

Virtual Element Methods for the Neutron Diffusion and Neutron Transport Equations with Applications in Nuclear Reactor Physics

Academic Supervisors:

Dr Mathew Eaton

Dr Michael Bluck

Industrial Supervisors:

Dr Paul Warner

Dr Alan Copestake

Author: John Alexander Ferguson

Department of Mechanical Engineering

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Nuclear Engineering at Imperial College London and
the Diploma of Imperial College London

March 2022

Abstract

In this thesis we propose a new spatial discretisation for the solution of the multigroup neutron diffusion equation (NDE) and the first-order form of the multigroup, discrete-ordinates, neutron transport equation (S_N -NTE). The proposed spatial discretisation is called the NURBS-Enhanced virtual element method (NE-VEM). The NE-VEM is a generalisation of the Galerkin finite element method which expands its applicability to meshes comprised of arbitrary polygonal elements with curved edges. Non-uniform rational B-splines (NURBS) are used as the geometric description of curved edges as they are the standard geometry encoding used by the computer-aided geometric design (CAGD) industry.

We developed a continuous-Galerkin VEM (NECG-VEM) to discretise the NDE and a discontinuous Galerkin VEM (NEDG-VEM) to discretise the S_N -NTE. For both of these discretisations we prove the well-posedness and derive a priori error estimates. We also present parallelisation strategies for both methods: a distributed memory strategy for the NECG-VEM and shared-memory strategy for the NEDG-VEM. The benefits of maintaining exact geometry are demonstrated through numerical examples. We show that the NE-VEM attains the theoretical order of convergence for problems with curved geometry for both the NDE and the S_N -NTE. The convergence rate of conventional, straight-sided VEM however, saturates due to the geometric error induced by approximating curved geometry with line segments.

Additionally, we present a 2D geometric modelling kernel and mesh generator to both model nuclear reactor core geometries and to create polygonal meshes which exactly represent the geometry of the reactor from the lowest level of mesh refinement.

Declaration of Originality

I hereby declare that the work contained in this thesis is my own unless stated otherwise. In such cases the work will be appropriately referenced

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

I would like to thank my principal academic supervisor Dr Matthew Eaton for his advice and guidance during my PhD studies

I would also like to thank Dr József Kópházi, Dr Charlie Latimer and Dr Seth Wilson for some useful discussions with regard to computational neutron transport and nuclear reactor physics.

I would like to acknowledge the financial support of the Engineering and Physical Sciences Research Council (EPSRC) and Rolls-Royce.

Finally, and most of all, I would like to thank my parents for their love and support. It is because of them that I have made it this far.

Contents

1	Introduction	14
1.1	Motivation	14
1.2	Project Scope	20
1.3	Thesis Structure	21
2	Numerical Modelling of Neutron Transport	23
2.1	The Neutron Transport Equation	23
2.1.1	Material Isotropy	26
2.1.2	The Fixed Source and k -Eigenvalue Problems	28
2.2	Angular Discretisation	29
2.2.1	The Spherical Harmonics (P_N) Method	29
2.2.2	The Discrete Ordinates (S_N) Method	29
2.2.3	The Diffusion Approximation	34
2.3	Energy Discretisation	35
2.4	Spatial Discretisation	38
2.4.1	The Finite Difference Method	38
2.4.2	The Finite Element Method	39
2.4.3	Isogeometric Analysis	41
3	A Review of Functional Analysis	43
3.1	Vector Space, Normed Space, Inner Product Space	43
3.2	Banach Space, Hilbert Space	45
3.3	Linear Operator, Linear Functional	46
3.4	Lebesgue Spaces, Weak Derivatives, Sobolev Spaces	49
3.5	Traces	52
3.6	Well Posedness of Variational Problems	52
3.6.1	Riesz Representation Theorem	53
3.6.2	Lax-Milgram Theorem	53
3.6.3	Banach-Nečas-Babuška Theorem	54
3.7	Galerkin Methods	54
4	The Virtual Element Method	57
4.1	Introduction	57
4.2	Preliminaries	60
4.2.1	The Polygonal Mesh	60
4.2.2	Polynomial Basis	61
4.3	The Local Curved Virtual Element Space	66
4.3.1	The Initial Space	66
4.3.2	The Enhanced Space	68
4.4	Approximation Properties of the Local Space	71

4.5	Practical Implementation	73
4.5.1	Boundary Shape Functions	73
4.5.2	Energy Projector	76
4.5.3	Gradient Projector	78
4.5.4	L^2 Projector	80
4.5.5	Change of Polynomial Basis	81
4.5.6	Computation of Local Bilinear Forms	84
4.6	Discussion	88
5	Geometry and Topology	89
5.1	Introduction	89
5.2	A Review of NURBS	91
5.2.1	B-spline Basis	91
5.2.2	B-spline Curves and Surfaces	93
5.2.3	NURBS Curves and Surfaces	96
5.2.4	Geometric Operations on NURBS	100
5.2.5	Surface Construction	104
5.3	Curved Polygon Tessellation and Numerical Integration	110
5.3.1	Tessellation	110
5.3.2	Integration	112
5.4	The Topology Data Structure	119
5.4.1	The Double-Connected Edge List	119
5.4.2	The Euler Operators	122
5.5	Mesh Generation	128
5.5.1	A Review of the Centroidal Voronoi Tessellation	129
5.5.2	Initial Seed Point Distribution	132
5.5.3	Seed Point Smoothing	135
5.5.4	CVT Clipping	138
5.5.5	Post-processing	146
5.6	Discussion	151
6	Virtual Element Discretisation of the Multigroup Neutron Diffusion Equation	153
6.1	Introduction	153
6.2	The Multigroup Diffusion Equations	155
6.3	The Continuous Galerkin VEM	157
6.3.1	The Model Problem	157
6.3.2	The Global CG-VEM Space	158
6.3.3	The VE Discretisation	158
6.3.4	Well Posedness and H^1 Error Estimate	161
6.4	The Variational Form of the NDE	163
6.4.1	Well-Posedness of Variational Problems	165
6.5	The Virtual Element Discretisation of the NDE	169
6.5.1	Well-Posedness of Discretisation	170
6.6	Practical Implementation	174
6.6.1	Global NECG-VEM Degrees of Freedom	174
6.6.2	The Global Matrices	179
6.6.3	The Fixed Source Problem	180
6.6.4	The k -Eigenvalue Problem	184
6.6.5	Parallelisation Strategy	187

6.7	Numerical Results	192
6.7.1	Method of Manufactured Solutions (MMS)	192
6.7.2	Burnable Supercell Benchmark Verification Test Case	197
6.7.3	The 2D OECD-NEA C5G7 Benchmark Verification Test Case	201
6.8	Discussion and Future Work	206
7	Virtual Element Discretisation of the Multigroup Neutron Transport Equation	208
7.1	Introduction	208
7.2	The Multigroup, S_N Equations	210
7.3	The Discontinuous Galerkin VEM	213
7.3.1	The Discrete Setting	213
7.3.2	The Model Problem	215
7.3.3	The DG-VEM Discretisation	217
7.3.4	Well Posedness and DG Error Estimate	219
7.4	The Transport Sweep	222
7.4.1	Local Problem on a Single Face	223
7.4.2	Local Problem on a Pair of Faces	225
7.4.3	Building the Digraph	227
7.4.4	Detecting Strongly Connected Components	229
7.4.5	Finding the Reduced Graph and the Final Ordering	232
7.5	The Virtual Element Discretisation of the NTE	233
7.6	Practical Implementation	237
7.6.1	Global NEDG-VEM Degrees of Freedom	237
7.6.2	The Global Matrices	239
7.6.3	The Fixed Source Problem	241
7.6.4	The k -eigenvalue Problem	243
7.6.5	The Sweep Task, Source Iteration and Power Iteration	244
7.6.6	Parallelisation Strategy	249
7.7	Numerical Results	252
7.7.1	Method of Manufactured Solutions 1	252
7.7.2	Method of Manufactured Solutions 2	254
7.7.3	Reed Cell Problem	256
7.7.4	The C5G7 UO_2 Pincell Problem	258
7.8	Discussion and Future Work	262
8	Conclusions and Further Work	264
8.1	Conclusions	264
8.2	Future work	266
A	Appendix to Chapter 2	267
A.1	Definition of Legendre Polynomials and Real Spherical Harmonics	267
A.2	Derivation of Isotropic Scatter Source	268
B	Appendix to Chapter 3	270
C	Appendix to Chapter 4	271
C.1	Proof of Proposition 4.3.1	271
C.2	Proof of Proposition 4.3.2	271
C.3	Proof of Proposition 4.3.6	271
C.4	Proof of lemma 6.3.2	272

C.5	Proof of Corollary 4.4.1.1	273
C.6	Proof of lemma 4.4.4	273
D	Appendix to Chapter 5	275
D.1	Proof of Lasserre on Edge	275
E	Appendix to Chapter 6	277
E.1	Proof of theorem 6.3.1	277
E.2	Proof of Lemma 6.3.4	278
E.3	Proof of lemma 6.3.5	279
E.4	Proof of Lemma 6.3.6	280
E.5	Proof of Theorem 6.3.7	281
E.6	Proof of Theorem 6.3.8	282
E.7	Derivation of MMS Forcing Function	283
E.8	C5G7 Supplementary Data	286
F	Appendix to Chapter 7	290
F.1	Proof of lemma 7.3.3	290
F.2	Proof of Lemma 7.3.4	290

List of Figures

2.1	Illustration of a particle (shown in red) with position $\mathbf{x}$ and angular $\boldsymbol{\Omega}$ components of the phase space.	26
2.2	Ordinate indices for the S_8 scheme.	31
2.3	Level symmetric angular quadratures of order 2, 4, 8 and 16. Each red disc is an ordinate and its size denotes its weight relative to the others.	32
2.4	Ray effects present in the solution to a S_{16} pincell problem, left image shows a less spatially refined mesh and the right image a more spatially refined mesh. . .	34
2.5	The microscopic total cross section $\sigma_t(E)$ as a function of incident neutron energy.	36
2.6	Structure of a typical group-to-group scattering matrix for a nuclear reactor physics problem with neutron up-scattering from lower energy neutron groups. .	38
4.1	An example mesh with the boundary face set $\mathcal{F}_h^b$ highlighted in darker grey. . .	60
4.2	Illustration of a star shaped domain (left and with respect to B), and a non-star shaped domain.	61
4.3	Illustration of local curvilinear VEM space (recreation of figure 6 in [1])	66
4.4	DoF distribution and numbering for $p = 3$	69
4.5	An example B-spline curve with a C^0 -continuous Jacobian.	74
4.6	Illustration of mapping from $\mathbb{P}_p([0, 1])$ to $\tilde{\mathbb{P}}_p(e_1)$ and $\tilde{\mathbb{P}}_p(e_1)$ for $p = 3$	75
5.1	Simplified schematic of a PWR core, the left-most image is the full core with a length scale around 4-5m, the middle image is a fuel assembly with a length scale of around 15-20cm and the right-most image is a pincell with a length-scale of <1 cm. The pincell may have additional features which are smaller still, such as annular air gaps or cladding with thicknesses on the millimetre scale.	89
5.2	Plots of the basis functions of orders $p = 0, 1, 2$ defined on the knot vector $\mathbf{u} = [0, 0, 0, 0.25, 0.75, 1, 1, 1]$	92
5.3	The first B-spline curve example.	93
5.4	The second B-spline curve example.	94
5.5	An example B-spline surface with the control net shown on the left and the surface on the right. The iso-lines in each parameter have been added.	96
5.6	Each type of conic section: from left to right we have 1) hyperbola, 2) parabola, 3) ellipse and 4) circle	97
5.7	A circular NURBS curve $\mathbf{C}(u)$ and its counterpart in homogeneous coordinates $\mathbf{C}^w(u)$. Based on an image presented in [2].	98
5.8	Curve and basis before knot insertion.	101
5.9	Curve and basis after knot insertion.	101
5.10	Curve after 2 knot insertions at $\hat{u} = 0.75$	102
5.11	The curve of example 4 after order elevation.	104
5.12	A bilinear quadrilateral	105

5.13	On the left we have the two original curves, in the middle we have the the curves after they have been made compatible and on the right we have the final ruled surface.	107
5.14	On the left we have four NURBS curves arranged in a loop, arrows represent the orientation of the curves. On the right a Coons surface, represented in NURBS form.	108
5.15	Example of an straight polygon (left) and a curved polygon (right), both extracted from a typical polygonal mesh used in the VEM.	110
5.16	A quad-tessellation of a straight-sided polygon. The $u-v$ axis shows the direction of the increasing u and v parameters of the underlying NURBS surface.	111
5.17	A quad tessellation of a curved polygon.	111
5.18	The ruled surface constructed to represent S_5	112
5.19	The ruled surface constructed to represent S_5	112
5.20	Polygon F with an example hyperplane $\mathcal{H}_i$. This is a recreation of ([3] figure 1)	113
5.21	An example curved polygon (left), its NURBS tessellation (middle) and its quadrature points (right).	117
5.22	Illustration of the vertex record (labelled v) and half-edge record (labelled he).	120
5.23	An example DCEL	121
5.24	Illustration of algorithms 7 (left) and 8 (right).	124
5.25	The series of Euler operators used to create a DCEL representing a square with a hole in it.	124
5.26	Relevant data in an <code>mev</code> operation.	125
5.27	The <code>mef</code> operation performed on face f	125
5.28	Initial geometry $\mathcal{D}_\Omega$	129
5.29	An example of a Voronoi Tessellation. The seeds are shown in dark green, the Voronoi vertices are shown in red and the Voronoi edges are shown in black, with bounded edges marked with a solid line and infinite edges shown with dashed lines. The convex hull of the seeds are shown in dashed red.	130
5.30	The Delaunay triangulation is shown in black while the VT is shown in dashed grey.	130
5.31	A clipped Voronoi tessellation after 0, 20 and 60 Lloyd's iterations (left to right).	132
5.32	A B-spline curve with a set of points distributed at equal arc-lengths.	133
5.33	Illustration of the sweep algorithm. The red dots denote the intersection points of the sweep line and the light red box denotes the range over which points are randomly distributed.	134
5.34	Initial point distribution with $N_{\text{init}} = 500$, the actual number of seeds is 729.	134
5.35	The VT corresponding to the initial seed distribution $\mathbf{P}^{(0)}$	135
5.36	Illustration of point reflection.	136
5.37	The smoothed VT corresponding to the final seed distribution $\mathbf{P}^{(i)}$	138
5.38	Upper endpoint event.	139
5.39	Lower endpoint event.	140
5.40	Intersection event.	140
5.41	The intersection points between the linearised DCEL and the Voronoi Tessellation. The intersections are shown in black.	141
5.42	Illustration of the boundary and internal interface half-edges.	143
5.43	Illustration of half-edges constructed from Voronoi edges.	143
5.44	Illustration of the half-edges going in and out of vertex v	144
5.45	Illustration of finding the bounding loops of the mesh DCEL $\mathcal{D}_h$	145
5.46	Illustration of finding the bounding loop of face $f_{i,h}$, shown in purple.	145
5.47	The final mesh DCEL, $\mathcal{D}_h$, after steps 1-3 of this section have been executed.	146

5.48	An example of a small Voronoi edge (left) and its cause in the dual graph (right).	147
5.49	An example of a polygon with a small edge and the definition of the swept angle β .	147
5.50	An illustration of the edge collapsing algorithm. The mesh before edge-collapse is shown on the left and after edge-collapse on the right.	148
5.51	Distribution of block IDs on mesh faces after the end of the first phase of the algorithm.	149
5.52	Mesh adaptivity scheme presented in [4]	151
6.1	A small mesh with two faces and the degrees of freedom for a NECG-VEM scheme of order $p = 3$.	175
6.2	An illustration of the algorithm used to partition the mesh.	187
6.3	A polygonal mesh and the sparsity pattern that results from the mesh numbering scheme.	189
6.4	The partitions of the mesh of figure 6.3 into $P = 10$ parts and the corresponding sparsity patterns arising from the partition numbering system.	189
6.5	The domain, $V = V_1 \cup V_2$ used for this MMS test case.	192
6.6	Plot of the MMS solution with isolines in white.	193
6.7	Three of the polygonal meshes used to solve the MMS problem.	193
6.8	The errors E_{L^2} and E_{H^1} for the case where both exact geometry and polynomial basis orthogonalisation are enabled.	195
6.9	The errors E_{L^2} and E_{H^1} for the case where exact geometry has been disabled.	196
6.10	Geometry and materials of the burnable supercell benchmark verification test case.	197
6.11	Two meshes used to solve the burnable supercell benchmark verification test case.	197
6.12	The scalar neutron flux contour of the reference solution	198
6.13	Convergence plots of I_{mod} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).	199
6.14	Convergence plots of I_{fuel} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).	199
6.15	Convergence plots of I_{ab1} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).	200
6.16	Convergence plots of I_{ab2} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).	200
6.17	The geometry of the 2D C5G7 benchmark verification test case.	201
6.19	Scalar neutron flux contours for energy groups 1, 3, 5 and 7.	204
6.20	Errors in QoIs for the C5G7.	205
7.1	Illustration of the jump and average operators, the figure on the right plots the functions $u _{F_1}$ and $u _{F_2}$ as one travels across the interface edge e .	214
7.2	Illustration of the inner and outer traces of v with respect to F .	215
7.3	Illustration of the first six steps of a transport sweep, the elements shown in black have been solved and the ones in red are the elements to be solved.	222
7.4	Illustration of the upwind flux vectors $\mathbf{u}_e^-$ for two edges e_1 and e_2 of face F . In this scenario the local matrix problem on faces F_1 and F_2 are assumed to have been solved.	224
7.5	Illustration of mutual dependency resulting from curved polygons F_1 and F_2 (left) and the set of DoFs for VE order $p = 3$ (right).	225
7.6	A small mesh and the sweep digraph for the direction $\boldsymbol{\Omega} = (-1, 1)$.	227

7.7	An example curvilinear mesh where the two faces shown on the right, F_1 and F_2 , form an SCC. This is because information propagates in both directions, as detected by the fact that the sign of $\mathbf{\Omega} \cdot \mathbf{n}$ changes.	228
7.8	An example graph with the SCCs highlighted. The other vertices are also considered SCCs but containing a single vertex.	229
7.9	Transpose graph G^T	232
7.10	The reduced graph of figure 7.8	233
7.11	A small mesh with three faces and the degrees of freedom for a NEDG-VEM scheme of order $p = 3$, vertex DoFs are shown on the left and edge DoFs on the right	237
7.12	Illustration of the set of shape functions which contribute to $\mathbf{R}_{g,k,F}$. The upwind DoFs are shown in red, the local DoFs on F are shown in black.	240
7.13	Example mesh with the 4 direction vectors in an S_4 level-symmetric angular quadrature.	249
7.14	The task forest for the mesh in figure 7.13 with an S_4 level-symmetric quadrature.	251
7.15	Plot of analytic solution with iso-lines shown in white.	252
7.16	Three of the polygonal meshes used in this test case.	252
7.17	Convergence curves for the NEDG-VEM, the curves for $E_{L^2}(u^h)$ are on the left and the curves for $E_{dg}(u^h)$ are on the right.	253
7.18	The DG errors in the two scalar fluxes: ϕ_1^h on the left and ϕ_2^h on the right.	255
7.19	Analytic solution for the Reed cell problem from [5].	256
7.20	Three of the meshes used to solve the Reed cell problem.	257
7.21	Three of the meshes used to solve the Reed cell problem.	257
7.22	Convergence curves in the DG norm for the Reed cell problem.	257
7.23	The geometry of the UO_2 pincell test case.	258
7.24	Three of the meshes used to solve the UO_2 pincell problem.	258
7.25	Convergence curves for each QoI given in table 7.4. QoIs I_1^f , I_4^f , I_7^f are on the left, I_1^f , I_4^f , I_7^f are on the right and K_{eff} is shown at the bottom. In all figures, the $p = 1$ curve is shown in blue, the $p = 2$ curve in orange and the $p = 3$ curve shown in green.	259
7.26	Scalar flux maps for the reference VEM solution with $p = 3$ and mesh with 6116 elements. Note the appearance of ray effects.	260
D.1	Mapping from $\mathbb{R}^2$ to $\mathbb{R}$	275
E.1	On the left are the isolines of $d(\mathbf{x})$, on the right is an illustration of the properties of $\nabla d(\mathbf{x})$ for $\mathbf{x} \in V_1$	284

List of Tables

2.1	Definitions of terms in the steady-state NTE.	25
5.1	The set of 5 Euler operators and their inverses.	122
6.1	Table of macroscopic nuclear cross section material data.	192
6.2	The nuclear data for the burnable supercell benchmark verification test case. . .	198
6.3	The reference values of the QoIs. They were computed using the NECG-VEM scheme defined on a polygonal mesh with 29401 elements and VE order $p = 4$, resulting in a scheme with $N_{\text{gdof}} = 486238$	198
6.4	Reference results for the C5G7 (2,028,163 DoFs).	201
7.1	Predecessors detected when traversing G	232
7.2	Cross section data for this test case.	254
7.3	Geometry and nuclear data for the Reed cell problem.	256
7.4	Reference values of the QoIs for and S_{16} , $p = 3$ and 6116 elements.	258

Chapter 1

Introduction

In this chapter, we shall provide a summary of current and future developments within the energy sector and nuclear industry. The nuclear industry has undergone a renaissance recently because of a renewed focus on low-carbon, base-load, energy generation to replace existing fossil-fuelled power plants. Additionally, energy security has recently become a major issue for countries around the world. This renaissance in nuclear energy has focussed on a new generation of modern nuclear power plants (NPPs). These generation III, III+ and IV nuclear reactor designs are designed to be cheaper, safer, more reliable and with improved performance compared to earlier generation I and II designs. As with all modern engineering design this new generation of NPPs have been designed using current nuclear reactor design software. However, in order to reduce pessimisms in nuclear reactor designs, and to reduce modelling errors, nuclear engineers are developing a new generation of high-fidelity numerical models. These high-fidelity numerical models aim to reduce uncertainties such as discretisation errors as well as epistemic and aleatoric uncertainties. The requirement for high-fidelity numerical models in modern nuclear reactor design has motivated the research presented in this PhD thesis. Specifically, high-fidelity **spatial discretisation**. This chapter shall also present the overall scope and structure of the PhD thesis.

1.1 Motivation

The nuclear energy industry in both the UK and around the world will likely see a renaissance in the coming decades. Many nations, around the world, have agreed to ambitious reductions in greenhouse gas (GHG) [6] emissions as part of their commitment to climate change treaties such as the Kyoto protocol [7], and the more recent, Paris agreement [8]. The vast majority of low-carbon, renewable energy comes from variable renewable energy (VRE) or intermittent renewable energy sources (IRES) that cannot be dispatched on demand at the request of power grid operators, according to market needs such as solar, tidal as well as offshore and onshore wind power. However, high-density energy storage systems are still in their infancy in terms of their technological development and widespread use within smart energy grids [9]. A key

issue is that even smart energy grids need a constant base-load electrical power supply [10]. All energy grids need to supply their customers with at least a minimal, yet reliable, electrical power supply, to satisfy any anticipated demand during the day[11]. This means that there is a need for low-carbon, yet reliable, energy sources to meet base-load electrical power generation requirements [12]. The property of continuous electrical power supply means that some energy sources are better suited than others to provide base-load [13]. Of the currently available base-load technologies only nuclear fission meets all the criteria of being low-carbon, immediately scalable and immediately commercially viable. These ambitious climate goals also come at a time when many factors are pushing up base-load energy demand from the grid [14]. These factors include the usual forces: such as the industrialisation of under-developed nations and the steady march of economic growth. However, there are two additional factors which are not only changing the total quantity of energy demand but are also altering the composition of energy demand. Chief amongst these is the electrification of transport, primarily passenger vehicles, which will require energy currently supplied by the fossil fuel infrastructure to be supplied by the energy grid [15]. Another factor is the continued growth of an electricity-intensive digital economy [16]. As more activities are moved online there will be a proportional increase in the number of servers required to power those activities and therefore greater demand put upon the electricity grid of many nations [17]. These two factors together mean that the quantity of energy required as base-load is set to increase in the near to mid-term. These appear to be growing trends in the developed world and the UK is no exception [12].

At present, the UK nuclear industry operates a fleet of eleven nuclear power plants producing approximately 16.1% of the UK's total electrical power generation capacity. There are ten advanced gas-cooled reactors (AGR) and one pressurised water reactor (PWR) which is Sizewell B in Suffolk. The UK's entire fleet of AGR NPPs will be decommissioned by 2028 [18]. A new fleet of NPPs is being constructed in the UK and comprises Generation III NPPs such as the Evolutionary (formally European) Power Reactor (EPR) [19] and the China General Nuclear Power Group (CGN) Hualong One (HPR1000) PWR [20]. The largest UK nuclear new-build programme is the £22 Billion, 3.26 GWe, twin EPR NPP at Hinkley Point C in Somerset. Hinkley Point C has an expected operational start date of June 2026 [21]. There are two other nuclear new-build programmes planned in Suffolk and Essex which are a twin EPR NPP which will be called Sizewell C [22] and a twin HPR1000 which will be called Bradwell B [23].

The UK Government is also currently renewing its fleet of submersible ship nuclear attack submarines (SSN) and submersible ship ballistic missile nuclear submarines (SSBN). The UK's latest designs of nuclear submarines are called the Astute class (SSN) and the Dreadnought class (SSBN). The Astute class submarine is fitted with the core H design Rolls-Royce PWR2 nuclear steam raising plant (NSRP) and the Dreadnought class is fitted with the Rolls Royce PWR3 NSRP. These reactors have been designed to provide up to twenty years of continuous operation without refuelling [24, 25].

There are several distinct advantages of naval nuclear power plants (NNPP) verses conventional

diesel-electric (SSK) [26] and air-independent powered (AIP) SSP submarines [27]. Modern non-nuclear submarines are potentially stealthier than current nuclear submarines as they rely upon natural circulation for cooling. Conversely, most NNPP use recirculation pumps (RCP) to constantly circulate reactor coolant (i.e. water) through the nuclear reactor core and primary circuit. Consequently, they can generate detectable noise during operation. However, nuclear submarines still have significantly better performance in terms of submergence times, speed, range and deep-ocean performance. It is important to note that modern non-nuclear attack submarines (SSK and SSP) still pose a significant threat to older, less stealthy and less manoeuvrable nuclear submarines in coastal environments [28, 29].

In addition to the UK civil nuclear build programme, and the renewal of the UK nuclear submarine fleet, the UK Government is also investing in a new generation of nuclear technologies, such as small modular reactors (SMR), advanced modular reactors (AMR) and new nuclear fusion reactors [30]. The UK government’s aims for the nuclear sector have been set out within the UK nuclear sector deal [31]. This deal provides a broad industrial strategy for the UK nuclear industry. The nuclear sector deal aims to radically improve the UK’s industrial productivity, while still delivering affordable and reliable base-load electrical power generation. In order to deliver this vision, the nuclear sector deal aims to provide a new approach to building NPP with a target for a 30% reduction in the cost of new build nuclear projects by 2030. It provides a longer-term vision of innovation-led growth that delivers successively lower nuclear power generation costs as well as a 20% reduction in nuclear decommissioning costs to the UK taxpayer. A final aim of the nuclear sector deal is to produce a more competitive nuclear supply chain with UK companies using the latest advanced manufacturing methods and entering both domestic and export markets for nuclear goods and services. The nuclear sector deal also aims to support the growth of a highly skilled workforce and a globally unique set of technologies and skills which can benefit other industries and services; thus leading to significant potential in overseas markets. Finally, it aims to produce a lasting contribution to UK communities that host nuclear facilities both currently and in the future. The nuclear sector deal was developed by the nuclear industry council, whose members are drawn from manufacturing, engineering, science and research, in partnership with the UK government [31].

A key aspect of achieving these goals will be to digitalise the whole UK nuclear industry, through the extensive use of digital twin models of NPPs [32, 33]. Improvements in modern, multicore and many-core, exascale, high-performance computing (HPC) hardware architectures is enabling high-fidelity numerical models to be developed and applied within a wide range of different engineering and scientific fields [34]. Modern exascale HPC hardware architectures and software are enabling a new generation of multiscale, multiphysics digital twin models to be developed for the nuclear industry. Examples include ENRICO (Exascale Nuclear Reactor Investigative Code) [35]. The department for Business, Energy and Industrial Strategy (BEIS) have funded several studies in digital nuclear reactor design in order to determine whether high-fidelity numerical modelling and simulation (M&S) can reduce the cost as well as improve the safety, reliability and performance of current and future NPP [36].

The engineering design and analysis of modern generation III and IV NPP represents a significant computational challenge due to the multiscale and multiphysical nature of the phenomena that must be modelled within these NPP [37, 38]. In nuclear reactor physics and reactor shielding, typical geometric length scales range from sub-millimetre (e.g. air gaps between nuclear fuel pin and clad), to centimetres (e.g. nuclear fuel pin radius to fuel assembly pitch) and metres (nuclear reactor diameter and reactor shield size) [39]. In nuclear fuel performance and structural integrity analyses, the length scale can range from the atomic, mesoscale to macroscale [40, 41]. In computational fluid dynamics (CFD) and nuclear thermal-hydraulics, the length scales can range from the Kolmogorov microscales to macroscale geometrical length scales of components within the reactor pressure vessel (RPV) and primary nuclear thermal-hydraulic circuit [42, 43]. Moreover, many of these physical phenomena can be weakly or strongly coupled over a range of different length scales. The latest innovations in exascale HPC hardware architectures and software are enabling nuclear engineers to model these multiscale problems and multiphysics phenomena in unprecedented levels of fidelity [44]. Exascale HPC is enabling high-fidelity digital twin models to be developed of the latest gen III and IV NPPs. This will reduce their through-life costs as well as improve their safety, reliability and performance.

The current generation of nuclear reactor analysis codes decompose multiscale and multiphysics problems into a multistage computational modelling process. This involves first measuring differential nuclear cross sections and compiling them into pointwise continuous energy nuclear data libraries such as the ENDF- BVIII (Evaluated Nuclear Data File)[45], JEFF-3.3 (Joint European Fission and Fusion)[46], JENDL-4.0 (Japanese Evaluated Nuclear Data Library) [47], BROND-3.1 (Russian Evaluated Nuclear Data Library) [48], CENDL-3.2 (Chinese Evaluated Nuclear Library) [49], TENDL (TALYS Evaluated Nuclear Data Library) [50] and CIELO (Collaborative International Evaluated Library Organisation) [51]. For Monte Carlo codes such as Serpent [52], MCNP [53] (Monte Carlo N-Particle) and OpenMC [54], these pointwise continuous energy nuclear data libraries can be used after processing into ACE (A Compact ENDF) file format using the nuclear processing code NJOY [55].

However, for deterministic nuclear reactor physics codes, these pointwise continuous energy nuclear libraries are energy group condensed to a couple of hundred discrete energy group microscopic nuclear cross sections. This condensation is performed using the GROUPR module within NJOY [56] which uses the multigroup approximation and a suitable condensation spectrum for the energy dependence of the angular neutron flux (e.g. Maxwell-Boltzmann energy spectrum for thermal neutrons, $1/E$ slowing down energy spectrum for epi-thermal neutrons and the Watt Fission energy spectrum for fast neutrons) [44]. These multigroup microscopic nuclear cross sections are then used within nuclear reactor lattice physics calculations which solve the integral and differential forms of the neutron transport equation (NTE) on single and multiple nuclear fuel assemblies with prescribed periodic boundary conditions [57]. Examples of typical lattice physics codes include WIMS (Winfrith Improved Multigroup Scheme) [58], CASMO [59], HELIOS [60], DRAGON [61], SCALE [62], APOLLO [63], STREAM [64] and LANCER [65]. These nuclear reactor lattice physics codes then produce between 2-4 group mi-

cross section and macroscopic nuclear cross sections and assembly discontinuity factors (ADF) for use in whole core nuclear reactor physics simulations. These whole-core nuclear reactor physics calculations are performed using an approximation to the NTE called the neutron diffusion equation (NDE). The NDE assumes that the angular dependence of the neutrons is linearly anisotropic and is therefore more tractable to solve than the NTE for whole-core nuclear physics problems. Nodal methods, which are a type of coarse mesh finite difference method, are used to discretise the NDE [66]. Whole-core nuclear reactor physics simulations are used for performing nuclear fuel burn-up/depletion calculations as well as determining the detailed pin powers within the nuclear reactor core using flux reconstruction methods. The nodal whole-core nuclear reactor physics calculations can also be coupled to both sub-channel/component (e.g. COBRA-TF [67] and VIPRE [68]) and/or system nuclear thermal-hydraulic models (e.g. RELAP5-3D [69] and TRACE [70]) to perform integrated nuclear reactor kinetics and nuclear thermal-hydraulic calculations.

More modern approaches to nuclear reactor design and analysis aim to eliminate many of the approximations and pessimisms inherent in the current generation of nuclear reactor analysis codes. Examples include the MOOSE (Multiphysics Object Oriented Simulation Environment), which was developed by Idaho National Laboratory (INL) [71], and VERA (Virtual Environment for Reactor Applications) developed by CASL (Consortium for Advanced Simulation of Light Water Reactors) at Oak Ridge National Laboratory (ORNL) [72]. More recently, nuclear reactor analysis codes such as ENRICO (Exascale Nuclear Reactor Investigative Code) have been developed for the very latest exascale HPC hardware architectures [35]. Codes such as MOOSE, VERA and ENRICO use high-fidelity massively parallel Monte Carlo (e.g. SHIFT [73] and OpenMC [54]) and deterministic neutron transport codes (e.g. MPACT [74] and Rattlesnake [75]) to resolve the angular neutron flux within the nuclear reactor core. The deterministic neutron transport code MPACT uses a method of characteristic (MoC) spatial discretisation whereas the Rattlesnake code uses a finite element spatial discretisation. The MoC is the de facto standard approach used for spatially discretising the NTE in commercial nuclear reactor lattice physics codes such as WIMS, CASMO, HELIOS, DRAGON, SCALE, APOLLO, STREAM and LANCER. However, most nuclear reactor lattice physics codes only solve nuclear fuel assemblies in two dimensions (2D), where the ray-tracing algorithms for the MoC are computationally efficient. In three-dimensions (3D) the ray-tracing algorithms are not as computationally efficient [76] and most MoC methods use hybrid 2D/1D algorithms where the axial direction is discretised using either simplified spherical harmonic (SP_N) or discrete ordinate (S_N) methods [77, 78]. Another issue with the MoC method is that for very thin absorbing layers the MoC requires very fine characteristic track spacing and a significant number of characteristic track directions, which can be very computationally demanding [79].

An alternative approach to the MoC for spatial discretisation is the finite element method (FEM) which takes the approach of decomposing the nuclear reactor geometry into a collection of non-overlapping triangles/quadrilaterals in 2D and tetrahedra/hexahedra in 3D. Local approximation spaces are constructed on each element of this decomposition and combined to

form a global set of basis functions. A discrete solution to a partial differential equation (PDE) is then sought within the finite-dimensional function space spanned by the global set of basis functions [80]. Typically, the discontinuous Galerkin FEM is used to discretise the first-order NTE [81, 82] while the continuous Galerkin FEM is used to discretise the neutron diffusion equation [83, 84]. A major limitation of these early FEM discretisations was the limitation of the finite elements to linear triangles or quadrilaterals. This limitation meant that, for curved reactor geometries, there is a mismatch between the geometry of the reactor and the mesh used for FE analysis. The errors induced in quantities such as reaction rates and neutron leakages can be substantial. High-order, isoparametric finite elements [85, 86] can ameliorate this issue to some degree however, any geometry described using polynomials will be unable to exactly represent conic sections, such as circles and ellipses [87]. Therefore, there will always be a geometric error induced using this methodology. There have been some recent advances in the FEM whereby the geometry of the elements themselves are described using the geometry encoding of computer-aided geometric design (CAGD): non-uniform rational B-splines (NURBS) [88]. This method is called the NURBS-enhanced FEM (NE-FEM) and it solves the problem of geometric mismatch between CAGD geometry based upon NURBS and the geometry of the FE mesh. However, the NE-FEM has yet to be applied to solve nuclear reactor physics problems.

Isogeometric analysis (IGA) [89] is another recent innovation in spatial discretisation which solves the issue of inexact geometry. IGA takes the approach of using NURBS basis functions to represent both the numerical solution and the geometry. Therefore, IGA can exactly represent any NURBS-based CAGD geometry from the lowest level of mesh refinement. However, a key disadvantage of the IGA method is that the computational domain must be comprised of one or more connected NURBS patches, each of which must be isomorphic to quadrilaterals in 2D and hexahedra in 3D. This reduces the flexibility of IGA with respect to the topology of the domain it can model, and also means there is a mismatch between the CAGD description of topology, based on a boundary representation of volumes bounded by NURBS surfaces, and the topology of the IGA domain.

More recently, the virtual element method (VEM) was introduced [90], which is an extension of the FEM to meshes comprised of general polygonal (2D) and polyhedral (3D) elements. In [1], a VEM with arbitrary curves as edges was introduced and has been adapted in this thesis to use NURBS as the description of geometry. This method, which we are calling the NURBS-enhanced VEM (as it bears similarity to the NE-FEM), represents a new and interesting method to bridge the gap between CAGD geometry and the computational mesh used for analysis. The use of curvilinear polygons in 2D and polyhedra in 3D enables a wide range of complex geometries to be geometrically decomposed and meshed without introducing geometric errors. This is accomplished by ensuring that those polygons/polyhedra adjacent to curved features of the original domain are exactly conforming to those features. The use of NURBS geometry within the VEM will enable a novel NURBS-enhanced spatial discretisation to be developed which is both computationally efficient and accurate as well as a method that can streamline the CAGD and computer-aided engineering (CAE) pipeline.

The NURBS-enhanced VEM will be the focus of this PhD thesis and we aim to demonstrate the unique mathematical properties of this spatial discretisation scheme as well as computational efficiency and numerical accuracy. Moreover, we will demonstrate the capacity of the NURBS-enhanced VEM capacity to model highly complex curvilinear geometries with high fidelity.

1.2 Project Scope

The scope of this PhD thesis is to present the application of the NURBS-Enhanced VEM (NE-VEM) to the neutron diffusion equation (NDE) and the neutron transport equation (NTE). This PhD thesis will provide a detailed description of the mathematics, algorithms and data structures which were developed to solve these equations in a manner which is both computationally efficient, numerically accurate and scalable. More specifically, this PhD thesis shall focus on the application of the NE-VEM to (a) the steady-state, multigroup NDE and (b) the steady-state, multigroup, discrete ordinate (S_N) angularly discretised first-order form of NTE. Additionally, we restrict the discussion to two spatial dimensions (2D). The key novelties we present in this thesis are as follows:

- The development of a geometry and mesh data structure which can both represent the geometry of complex nuclear reactors 2D and also represent curvilinear, polygonal meshes with millions of polygons. This data structure uses non-uniform rational B-splines as its description of the geometry and the doubly-connected edge list as its description of the topology.
- The development of a mesh generation algorithm, based on centroidal Voronoi tessellation, which can produce curvilinear, polygonal meshes which exactly preserve the geometry of the input domain.
- The development of continuous Galerkin, NURBS-enhanced VEM (NECG-VEM) for the spatial discretisation of the steady-state, multigroup, neutron diffusion equation. In particular, we prove the well-posedness of the NECG-VEM discretisation and we show, through numerical examples, the improvement in convergence rate attained when using an exact-geometry VEM compared to a classical, straight-sided VEM. We also provide a detailed account of their practical implementation and finally, we present a distributed-memory parallelisation strategy.
- The development of a discontinuous Galerkin, NURBS-enhanced VEM (NEDG-VEM) for the solution of the steady-state, multigroup, discrete ordinates, neutron transport equation. The relaxed inter-element continuity is necessary in order to model first-order hyperbolic problems. We present a transport sweeping algorithm for this problem and also present a shared-memory parallelisation strategy based on task scheduling.

1.3 Thesis Structure

After this chapter, the thesis is structured into seven more chapters. The first three provide both a literature review and background theory which later chapters build upon. Consequently, the theory presented in these chapters is largely taken from other sources, with care taken to properly cite the original material. The first three chapters are organised as follows:

- In chapter 2 we introduce the neutron transport equation and describe its physical meaning. We then review some of the discretisation methods which are applied to each of the independent variables: neutron energy E , neutron angle Ω and spatial coordinates $\mathbf{x}$.
- In chapter 3 we review the parts of functional analysis which underpins both the finite element method and the virtual element method. This chapter is more for reference and introduces the functional analytic definitions which are used throughout the thesis.
- In chapter 4 we present the NURBS-enhanced virtual element method, we define the local NE-VEM function spaces and define the polynomial projection operators which are crucial in computing discrete VE bilinear forms. We also present a practical guide to computing the local projections, bilinear forms and linear forms.

The following three chapters present the research which was performed as part of this PhD and are organised as follows:

- In chapter 5 we present the 2D geometric modelling kernel (GMK) which was developed as part of this project. The GMK provides the topological and geometric description of both the reactor geometries and the curvilinear, polygonal meshes which were generated from them. We also present a polygonal mesh generator which was developed to create meshes from input geometry created for test cases.
- In chapter 6 we introduce the NURBS-enhanced, continuous-Galerkin virtual element method (NECG-VEM) and present its application to multigroup neutron diffusion equation (NDE). We prove the well-posedness of the discretisation and the a priori error estimate in the H^1 norm. We then present a distributed-memory parallelisation strategy based upon mesh partitioning between MPI processes. Finally, we present some numerical benchmark problems which substantiate the theory presented in the chapter.
- In chapter 7 we introduce the NURBS-enhanced, discontinuous Galerkin virtual element method (NEDG-VEM) and present its application to the multigroup, discrete ordinates neutron transport equation (S_N -NTE). We present the well-posedness and an error estimate for a suitably chosen mesh-dependent norm. We then present the transport sweep algorithm for curved, polygonal meshes and a shared-memory parallelisation strategy based on task scheduling. Finally, we present some numerical benchmark problems which substantiate the theory presented in the chapter.

Chapter 8 provides a summary, some concluding remarks and discusses directions for further research.

To date there have been two three publications related to the work presented in this thesis.
They are [91, 92, 93]

Chapter 2

Numerical Modelling of Neutron Transport

A nuclear fission reactor is a complex engineered system with many interdependent physical phenomena occurring simultaneously on many different length and time scales. However, the primary physical phenomenon that distinguishes a nuclear fission (or fusion) reactor from any other engineered system is the transport of neutral particle radiation such as neutrons and gamma-ray photons. The aims of this chapter are two-fold:

- The first aim is to formally pose the problems that this thesis is solving using the virtual element method (VEM). These are 1) the steady-state, multigroup neutron diffusion equation and 2) the steady-state, discrete-ordinates, multigroup neutron transport equation.
- The second is to provide the reader with context and to clarify where the virtual element method sits within the pantheon of methods that have preceded it.

2.1 The Neutron Transport Equation

The neutron transport equation (NTE) was first derived in 1937 by L.S. Ornstein and G.E. Uhlenbeck [94, 95] in the context of studying the kinetic properties of neutrons in paraffin. The NTE is a linearised form of the non-linear Boltzmann transport equation (BTE) which describes the average non-equilibrium statistical behaviour of a gas of atoms or molecules [96]. The NTE is a valid approximation of the migration and interaction within a host medium or material under the following set of physical assumptions [97]

- Neutrons do not interact with other neutrons.
- Neutrons are not affected by any external force fields, such as electric or magnetic fields.
- The lifespan of a neutron is sufficiently short so that the motion due to gravity may be safely neglected.

- Collisions between neutrons and nuclei occur instantaneously.
- The host media may be considered stationary.

The first assumption is a consequence of the neutron density being many orders of magnitude smaller than the density of host nuclei (typical values of which are 10^{10} cm^{-3} and 10^{22} cm^{-3} respectively [98]). Neutron transport theory is fundamentally a kinetic theory description of the migration and interaction of neutrons with a host medium or material. This contrasts with continuum descriptions of matter such as the neutron diffusion equation (NDE). Neutron transport theory takes a different approach to that taken in continuum descriptions of matter. The statistical nature of the interactions between neutrons and the host media means that finding a density function $n(\mathbf{x}, E, \boldsymbol{\Omega}, t)$ which says there are exactly n particles at point $\mathbf{x}$, with energy E , travelling with direction $\boldsymbol{\Omega}$ and at time t is not possible. Rather, in the context of neutron transport theory, $n(\mathbf{x}, E, \boldsymbol{\Omega}, t)$ must be understood as the mean number of particles at point $(\mathbf{x}, E, \boldsymbol{\Omega}, t)$ in a statistical phase-space. Therefore, the NTE describes the average, non-equilibrium, statistical behaviour of neutrons within a host medium or material [99]. If one were to take measurements, the true number would fluctuate slightly about $n(\mathbf{x}, E, \boldsymbol{\Omega}, t)$. Indeed, a fundamental assumption for the validity of transport theory as it is presented here, is that the variance is very small relative to the mean [98]. In the case where the variance is high relative to the mean, such as during reactor start-up, a different approach based on the Pál-Bell equation is required [100, 101].

The NTE can be derived phenomenologically (or heuristically) by using the conservation of neutrons within a differential phase-space volume. This approach to deriving the NTE is given in monographs and textbooks in nuclear reactor theory [102, 103]. Alternative, more mathematically rigorous derivations based on quantum statistical mechanics [99] or using a classical master (Chapman-Kolmogorov) equation [104], have also been developed. This thesis is solely concerned with time-independent (or steady-state) neutron diffusion and transport problems. Therefore, the physics of delayed neutrons has also been ignored. The primary objects of study are therefore the steady-state NDE and NTE without delayed neutrons.

Throughout this thesis, the spatial domain is an open, connected domain $V \subset \mathbb{R}^2$ with boundary ∂V . The energy domain is the positive real number line $\mathbb{R}^+$ and the angular domain is the unit sphere centred at $(0, 0, 0)$: $S = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}\| = 1\}$. The mean (average or expected value) of the neutron distribution is represented by the angular neutron flux distribution $\psi(\mathbf{x}, E, \boldsymbol{\Omega}) = vn(\mathbf{x}, E, \boldsymbol{\Omega})$ (where $v(E)$ is the magnitude of neutron velocity). Under the assumptions given in this chapter, the angular flux distribution $\psi(\mathbf{x}, E, \boldsymbol{\Omega})$ satisfies the steady-state NTE:

$$\begin{aligned}
& \boldsymbol{\Omega} \cdot \nabla \psi(\mathbf{x}, E, \boldsymbol{\Omega}) + \Sigma_t(\mathbf{x}, E) \psi(\mathbf{x}, E, \boldsymbol{\Omega}) \\
& = \\
& \int_0^\infty dE' \int_{4\pi} d\boldsymbol{\Omega}' \Sigma_s(\mathbf{x}, E' \rightarrow E, \boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega}) \psi(\mathbf{x}, E', \boldsymbol{\Omega}')
\end{aligned} \tag{2.1.1}$$

$$+ \frac{\chi(\mathbf{x}, E)}{4\pi} \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E') \int_{4\pi} d\Omega' \psi(\mathbf{x}, E', \Omega') + Q(\mathbf{x}, E, \Omega),$$

for all $(\mathbf{x}, E, \Omega) \in V \times \mathbb{R}^+ \times S$ and subject to boundary conditions:

$$\psi(\mathbf{x}, E, \Omega) = \psi_{\text{in}}(\mathbf{x}, E, \Omega) + \beta(\mathbf{x}, \Omega') \psi(\mathbf{x}, E, \Omega') \quad \forall \mathbf{x} \in \partial V_-$$

where $\partial V_- = \{\mathbf{x} \in \partial V : \mathbf{n} \cdot \Omega < 0\}$ and $\mathbf{n}(\mathbf{x})$ is the outward unit normal to ∂V and $\Omega' = \Omega - 2(\Omega \cdot \mathbf{n})\Omega$ is the direction resulting from specular reflection. The quantity $\beta(\mathbf{x}, \Omega')$ is the specular reflectivity and is equal to 0 for vacuum boundaries and equal to 1 for reflective boundaries. We present a table with the definitions of each of the terms in the NTE in table 2.1.

Table 2.1: Definitions of terms in the steady-state NTE.

Symbol	Definition
$\mathbf{x}, E, \Omega$	From left to right: Spatial location vector, neutron kinetic energy and neutron direction vector.
$\psi(\mathbf{x}, E, \Omega)$	The angular neutron flux.
$\Sigma_t(\mathbf{x}, E)$	The macroscopic total neutron cross section.
$\Sigma_s(\mathbf{x}, E' \rightarrow E, \Omega' \rightarrow \Omega)$	Double differential scattering cross section from energy E' to E and angle $\Omega' \rightarrow \Omega$.
$\chi(\mathbf{x}, E)$	The prompt spectrum of neutron energies released in fission.
$\nu(\mathbf{x}, E)$	The average number of neutrons released in a fission event by a neutron of energy E .
$\Sigma_f(\mathbf{x}, E)$	Macroscopic fission neutron cross section.
$Q(\mathbf{x}, E, \Omega)$	Extraneous, prescribed (or fixed) source of neutrons.

The physical interpretation of each of the terms in equation (2.1.1) are as follows:

- The terms:

$$\Omega \cdot \nabla \psi(\mathbf{x}, E, \Omega) + \Sigma_t(\mathbf{x}, E) \psi(\mathbf{x}, E, \Omega)$$

are the streaming and collision terms. These represent the neutrons lost in a differential control volume of phase-space through both streaming and interactions with the host medium such as neutron capture and neutron scattering.

- The term:

$$\int_0^\infty dE' \int_{4\pi} d\Omega' \Sigma_s(\mathbf{x}, E' \rightarrow E, \Omega' \rightarrow \Omega) \psi(\mathbf{x}, E', \Omega') \quad (2.1.2)$$

represents the neutron source resulting from neutrons entering a scattering event with energy E' and direction Ω' and exiting the scatter event with energy E and direction Ω .

- The term:

$$\frac{\chi(\mathbf{x}, E)}{4\pi} \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E') \int_{4\pi} d\Omega' \psi(\mathbf{x}, E', \Omega')$$

represents the neutron source resulting from neutrons entering a prompt fission event with energy E' and exiting with energy E .

The angular flux $\psi(\mathbf{x}, E, \Omega)$ represents a mean (average or expectation value) over a statistical

ensemble of potential states of the system according to kinetic theory [99]. Therefore it must be positive everywhere, a property which can only be guaranteed if the sources are all positive also [102].

2.1.1 Material Isotropy

At this stage, we shall introduce some additional information concerning the angular part of the phase space in order to properly explain the concept of **material isotropy**: an important simplifying assumption we shall apply to the NTE. In figure 2.1 we present an illustration of the spatial and angular components of the phase space following the convention presented in [103]. While the direction vector $\mathbf{\Omega} = (\mu, \eta, \xi)$ has three components, only two are independent, with the third being a dependent variable. The two independent components are best described using the angles shown in figure 2.1. The first angle which defines the particle direction is the **azimuthal angle**, denoted ω , which is defined as the counter-clockwise angle in the yz -plane between the positive y -axis and the projection of $\mathbf{\Omega}$ onto the yz -plane. It has the range $\omega \in [0, 2\pi]$. The second angle is the **polar angle**, which is defined as the counter-clockwise angle between the positive x -axis and $\mathbf{\Omega}$ in the plane spanned by $(\mathbf{e}_x, \mathbf{\Omega})$. It has range $[0, \pi]$. The components of $\mathbf{\Omega}$ are called the direction cosines and may be related to (ω, θ) as follows:

$$\mu = \mathbf{e}_x \cdot \mathbf{\Omega} = \cos(\theta) \quad \eta = \mathbf{e}_y \cdot \mathbf{\Omega} = \sqrt{1 - \mu^2} \cos(\omega) \quad \xi = \mathbf{e}_z \cdot \mathbf{\Omega} = \sqrt{1 - \mu^2} \sin(\omega)$$

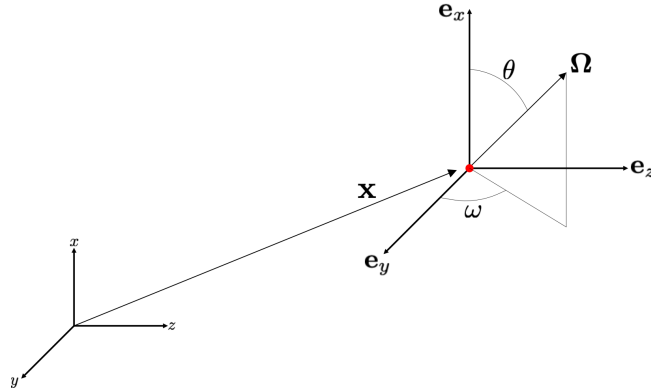


Figure 2.1: Illustration of a particle (shown in red) with position $\mathbf{x}$ and angular $\mathbf{\Omega}$ components of the phase space.

Using the azimuthal and polar angles one can define the differential solid angle as $d\Omega = \sin(\theta) d\theta d\omega$ and therefore integrals of the whole of the sphere $S = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}\| = 1\}$ may be written as:

$$\int_{4\pi} d\Omega f(\mathbf{\Omega}) = \int_0^\pi d\theta \sin(\theta) \int_0^{2\pi} d\omega f(\mathbf{\Omega}(\theta, \omega)).$$

It is typical to change variables with the substitution $d\mu = -\sin(\theta)d\theta$ to give:

$$\int_{4\pi} d\Omega f(\mathbf{\Omega}) = \int_{-1}^1 d\mu \int_0^{2\pi} d\omega f(\mathbf{\Omega}(\mu, \omega)).$$

An isotropic material is one where distributions of the form $f(\boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega})$ do not depend on the incoming angle $\boldsymbol{\Omega}'$, only the angle between $\boldsymbol{\Omega}'$ and $\boldsymbol{\Omega}$. Therefore, a function $f(\boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega})$ may be transformed as follows:

$$f(\boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega}) = \frac{1}{2\pi} f(\boldsymbol{\Omega} \cdot \boldsymbol{\Omega}') = \frac{1}{2\pi} f(\mu)$$

where the $1/2\pi$ term is a normalisation to preserve the angular integral over the unit-sphere S :

$$\int_{4\pi} d\Omega' f(\boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega}) = \int_{-1}^1 d\mu \frac{f(\mu)}{2\pi} \int_0^{2\pi} d\omega = \int_{-1}^1 d\mu f(\mu),$$

Angular distributions in just the polar angle $f(\mu)$, may be expanded in the L -order Legendre polynomials as follows [103]:

$$f(\mu) = \sum_{l=0}^L \left(\frac{2l+1}{2} \right) f_l P_l(\mu), \quad f_l = \int_{-1}^1 d\mu f(\mu) P_l(\mu).$$

where the definition of $P_l(\mu)$ may be found in section A.1. More general angular distributions $f(\boldsymbol{\Omega}) = f(\mu, \theta)$ may be expanded in the L -order real spherical harmonics:

$$f(\boldsymbol{\Omega}) = \sum_{l=1}^L \frac{2l+1}{4\pi} \sum_{m=-l}^l f_l^m R_l^m(\boldsymbol{\Omega}), \quad f_l^m = \int_{4\pi} d\Omega R_l^m(\boldsymbol{\Omega}) f(\boldsymbol{\Omega})$$

where the definition of $R_{l,m}(\boldsymbol{\Omega})$ may also be found in A.1. If a host medium is assumed isotropic then the macroscopic double differential scattering cross section may be written as:

$$\Sigma_s(\mathbf{x}, E' \rightarrow E, \boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega}) = \frac{1}{2\pi} \Sigma_s(\mathbf{x}, E' \rightarrow E, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega})$$

and the neutron scattering source (equation (2.1.2)) may be expressed as

$$\int_0^\infty dE' \sum_{l=0}^L \left(\frac{2l+1}{4\pi} \right) \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) \phi_l^m(\mathbf{x}, E') \quad (2.1.3)$$

where

$$\phi_l^m(\mathbf{x}, E) = \int_{4\pi} d\Omega' R_l^m(\boldsymbol{\Omega}') \psi(\mathbf{x}, E, \boldsymbol{\Omega}'),$$

is the (l, m) angular neutron flux moment and $\phi(\mathbf{x}, E) = \phi_0^0(\mathbf{x}, E)$ is the scalar neutron flux. We provide a full derivation of equation (2.1.3) in section A.2. The steady-state NTE for isotropic media is therefore given by:

$$\begin{aligned} & \boldsymbol{\Omega} \cdot \nabla \psi(\mathbf{x}, E, \boldsymbol{\Omega}) + \Sigma_t(\mathbf{x}, E) \psi(\mathbf{x}, E, \boldsymbol{\Omega}) \\ & = \\ & \int_0^\infty dE' \sum_{l=0}^L \frac{2l+1}{4\pi} \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) \phi_l^m(\mathbf{x}, E') \end{aligned} \quad (2.1.4)$$

$$+ \frac{\chi(\mathbf{x}, E)}{4\pi} \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E') \phi(\mathbf{x}, E') + Q(\mathbf{x}, E, \boldsymbol{\Omega}),$$

2.1.2 The Fixed Source and k -Eigenvalue Problems

In this thesis, we only consider two types of problems: the fixed (prescribed) neutron source problem (to be referred to as the **fixed-source** problem for the remainder of the thesis) and the **k-eigenvalue** (or criticality) problem. To pose these two kinds of problems we shall first reformulate equation (2.1.4) in operator form:

$$L\psi(\mathbf{x}, E, \boldsymbol{\Omega}) = S\psi(\mathbf{x}, E, \boldsymbol{\Omega}) + F\psi(\mathbf{x}, E, \boldsymbol{\Omega}) + q(\mathbf{x}, E, \boldsymbol{\Omega}) \quad (2.1.5)$$

where

$$\begin{aligned} L\psi(\mathbf{x}, E, \boldsymbol{\Omega}) &= \boldsymbol{\Omega} \cdot \psi(\mathbf{x}, E, \boldsymbol{\Omega}) + \Sigma_t(\mathbf{x}, E) \psi(\mathbf{x}, E, \boldsymbol{\Omega}), \\ S\psi(\mathbf{x}, E, \boldsymbol{\Omega}) &= \int_0^\infty dE' \sum_{l=0}^L \left(\frac{2l+1}{4\pi} \right) \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) \phi_l^m(\mathbf{x}, E'), \\ F\psi(\mathbf{x}, E, \boldsymbol{\Omega}) &= \frac{\chi(\mathbf{x}, E)}{4\pi} \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E') \phi(\mathbf{x}, E'), \\ \phi_l^m(\mathbf{x}, E) &= \int_{4\pi} d\boldsymbol{\Omega}' R_l^m(\boldsymbol{\Omega}') \psi(\mathbf{x}, E, \boldsymbol{\Omega}'). \end{aligned}$$

In the fixed source problem, we assume that there are no multiplying media in the domain V and that there is only a fixed, extraneous source $q(\mathbf{x}, E, \boldsymbol{\Omega})$ which is driving the system:

$$(L - S)\psi(\mathbf{x}, E, \boldsymbol{\Omega}) = q(\mathbf{x}, E, \boldsymbol{\Omega})$$

In the k -eigenvalue problem we assume there are no extraneous sources (therefore $q(\mathbf{x}, E, \boldsymbol{\Omega}) = 0$) and we are left with the eigenvalue problem:

$$(L - S)\psi(\mathbf{x}, E, \boldsymbol{\Omega}) = \frac{1}{\gamma} F\psi(\mathbf{x}, E, \boldsymbol{\Omega})$$

where we are only interested in the first and largest value of $1/\gamma$ and the corresponding fundamental mode $\psi(\mathbf{x}, E, \boldsymbol{\Omega})$. The largest value of $1/\gamma$ is commonly known as the effective multiplication factor and is denoted K_{eff} .

These are the two kinds of transport problems that this thesis seeks to discretise and solve using the virtual element method (VEM). However, the VEM is solely a spatial discretisation. Therefore, the NTE must first be semi-discretised in both energy and angle before any spatial discretisation may be considered. For this reason, energy and angular discretisation methods are the topics of the next two sections.

2.2 Angular Discretisation

There are many angular discretisation methods, each with its strengths and weaknesses. Some less common methods include angular wavelets [105, 106, 107] and angular finite elements [108, 109, 110, 111]. Both of these methods, while perhaps more complicated than the more traditional S_N and P_N methods, have the considerable benefit of being well-suited to angular adaptivity. However, this thesis aims to investigate the VEM, which is a spatial discretisation. Therefore, we simply review the two most common treatments in angle: the discrete-ordinate (S_N) method and the spherical harmonic (P_N) method. In the case of the steady-state NTE, the diffusion approximation is implicitly a P_1 angular expansion and therefore we consider it as a special case of P_N , albeit one with profoundly different mathematical properties. For this reason, we deal with it separately in section 2.2.3.

The time-dependent P_1 equations **do not** reduce to the time-dependent neutron diffusion equation. Rather the time-dependent P_1 equations are equivalent to the Telegrapher's equation (see [97], page 156).

2.2.1 The Spherical Harmonics (P_N) Method

The P_N method was first presented by J.H. Jeans [112] and developed further by J.C Mark [113, 114] and involves representing the angular dependence of $\psi(\mathbf{x}, E, \boldsymbol{\Omega})$ with a spherical harmonic expansion as follows:

$$\begin{aligned}\psi(\mathbf{x}, E, \boldsymbol{\Omega}) &\approx \sum_{l=0}^N \frac{2l+1}{4\pi} \sum_{m=-l}^l \phi_l^m(\mathbf{x}, E) R_l^m(\boldsymbol{\Omega}), \\ \phi_l^m(\mathbf{x}, E) &= \int_{4\pi} d\Omega \psi(\mathbf{x}, E, \boldsymbol{\Omega}) R_l^m(\boldsymbol{\Omega})\end{aligned}\tag{2.2.1}$$

The real spherical harmonics are known to be complete in $L^2(S)$ therefore as $N \rightarrow \infty$, equation (2.2.1) becomes exact. To solve for the angular moments, one substitutes the angular expansion of equation (2.2.1) into the steady-state NTE (equation (2.1.5)), then multiplies the equations by a spherical harmonic of different order $R_l^{m'}(\boldsymbol{\Omega})$ and integrates over the unit sphere S . One may then invoke the orthogonality properties of the spherical harmonics (see section A.1) to yield a system of equations for the angular flux moments $\phi_l^m(\mathbf{x}, E)$. This is essentially a weighted residual Galerkin projection, using spherical harmonic basis functions in the angular domain. As we have not chosen to implement the P_N method in this thesis we shall end the discussion here, however for a thorough treatment of P_N methods we refer the reader to [103] section 3.7, or [115] chapter 12.

2.2.2 The Discrete Ordinates (S_N) Method

The discrete ordinates method was first developed in 1950 by S. Chandrasekhar in [116] for modelling radiative transfer in the field of astrophysics. B.G Carlson, K.D. Lathrop and C. Lee further refined the discrete ordinates method and applied it to the field of neutron transport

in a series of reports [117, 118, 119, 120] for the Los Alamos National Laboratory (LANL) in the 1950s and 1960s. It is a collocation method in the angular domain, meaning that a set of directions (ordinates) and weights are first constructed $\{(w_k, \boldsymbol{\Omega}_k) : k = 1, \dots, K\}$, the exact nature of which will be explained later, which can accurately integrate angular functions over S :

$$\int_{4\pi} d\Omega f(\boldsymbol{\Omega}) \approx \sum_{k=1}^K w_k f(\boldsymbol{\Omega}_k)$$

The steady-state NTE (equation (2.1.4)) is then solved in each of the discrete directions. Letting $\psi_k(\mathbf{x}, E) = \psi(\mathbf{x}, E, \boldsymbol{\Omega}_k)$ we have:

$$\begin{aligned} & [\boldsymbol{\Omega}_k \cdot \nabla + \Sigma_t(\mathbf{x}, E)] \psi_k(\mathbf{x}, E) \\ &= \\ & \int_0^\infty dE' \sum_{l=0}^L \frac{2l+1}{4\pi} \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}_k) \phi_l^m(\mathbf{x}, E') \\ & + \frac{\chi(\mathbf{x}, E)}{4\pi} \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E') \phi(\mathbf{x}, E') + Q(\mathbf{x}, E, \boldsymbol{\Omega}_k), \end{aligned}$$

where the angular flux moments are given by:

$$\phi_l^m(\mathbf{x}, E) = \int_{4\pi} d\Omega R_l^m(\boldsymbol{\Omega}) \psi(\mathbf{x}, E, \boldsymbol{\Omega}) \approx \sum_{k=1}^K w_k R_l^m(\boldsymbol{\Omega}_k) \psi_k(\mathbf{x}, E).$$

and similarly, the spherical harmonic expansion of the angular flux is given by:

$$\psi_k(\mathbf{x}, E) \approx \sum_{l=0}^L \left(\frac{2l+1}{4\pi} \right) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}_k) \phi_l^m(\mathbf{x}, E).$$

It is common practice to write the full set of angular fluxes and angular flux moments as the following vectors:

$$\boldsymbol{\psi} = [\psi_1, \dots, \psi_K]^T, \quad \boldsymbol{\phi} = [\phi_0^0, \phi_1^{-1}, \phi_1^0, \phi_1^1, \dots, \phi_L^L]$$

where more details of the 1D-index to moment mapping: $i \leftrightarrow (l, m)$ may be found in equation (A.1.1). Using this notation, we may define the **direction-to-moment** (D) and **moment-to-direction** (M) operators as follows:

$$\boldsymbol{\phi} = D\boldsymbol{\psi}, \quad D_{i,k} = w_k R_{l(i)}^{m(i)}(\boldsymbol{\Omega}_k) \quad (2.2.2)$$

$$\boldsymbol{\psi} = M\boldsymbol{\phi}, \quad M_{k,i} = \left(\frac{2l(i)+1}{4\pi} \right) R_{l(i)}^{m(i)}(\boldsymbol{\Omega}_k) \quad (2.2.3)$$

where $(l(i), m(i))$ are the spherical harmonic indices corresponding to index i (equation (A.1.1)). Letting N_{mom}^L denote the number of spherical harmonic flux moments in a P_N expansion of order

L , then D and M have dimensions $N_{\text{mom}}^L \times K$ and $K \times N_{\text{mom}}^L$ respectively. We shall now introduce the angular quadrature set which we have implemented in this thesis, which is the well-known level symmetric quadrature (LSQ).

The Level Symmetric Discrete Ordinate (S_N) Quadrature

The level symmetric quadrature is one of the earliest quadratures developed for the S_N method [118], however, there are a few variants corresponding different choices of parameters. In this section, we shall review the version of the LSQ presented in [121], as this was the version we implemented.

The defining feature of the LSQ is that it has symmetry about all three axes, making it rotationally invariant with respect to 90-degree rotations about any axis [121]. For an LS quadrature of order N , with $N \in \{2, 4, 6, \dots, 16\}$ we first define a set $N/2$ of positive, increasing direction cosines on each axis:

$$\{\mu_i : i = 1, \dots, (N/2)\}, \quad \{\eta_i : i = 1, \dots, (N/2)\}, \quad \{\xi_i : i = 1, \dots, (N/2)\},$$

such that $\mu_i = \eta_i = \xi_i$. It can be shown (see [97], section 4.2) that, due to symmetry conditions, only one of the direction cosines μ_i is independent. Therefore, once the first direction cosine μ_1 is chosen then all others are determined. The direction cosines are given by [121]:

$$\mu_1^2 = \frac{1}{3(N-1)},$$

$$\mu_i^2 = \mu_1^2 + (i-1) \frac{2(1-3\mu_1^2)}{N-2} \quad i = 2, \dots, N/2.$$

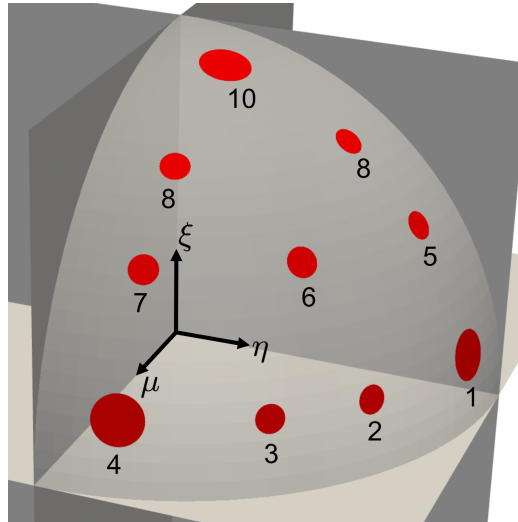


Figure 2.2: Ordinate indices for the S_8 scheme.

The full set of ordinates on the primary octant forms a triangle, with $N/2$ ordinates on the bottom row, $N/2 - 1$ on the next and so on. Row i corresponds the set of ordinates with the

z -axis direction cosines all equal to ξ_i . We give an example of the numbering scheme for an S_8 quadrature in figure 2.2. For a given S_N order (N) the number of directions on the primary octant are given by $K_{oct} = N(N+2)/8$. In 2D, where due to symmetry about the xy -axis we only require ordinates in the positive hemisphere, the total number of ordinates is given by $K = N(N+2)/2$ and in 3D we have $K = N(N+2)$. With the ordinates computed all that remains to be determined are the weights. Here we follow the procedure of [121] whereby we compute the set of weights which exactly integrate even-order polynomials of the direction cosines μ, η up to a prescribed order:

$$\sum_{k=1}^K w_k \mu_k^l \eta_k^m = \frac{2}{\pi} \int_0^1 d\mu \int_0^{\pi/2} d\omega \mu^l \eta^m, \quad \text{for even } l, m \text{ and where } l+m < N. \quad (2.2.4)$$

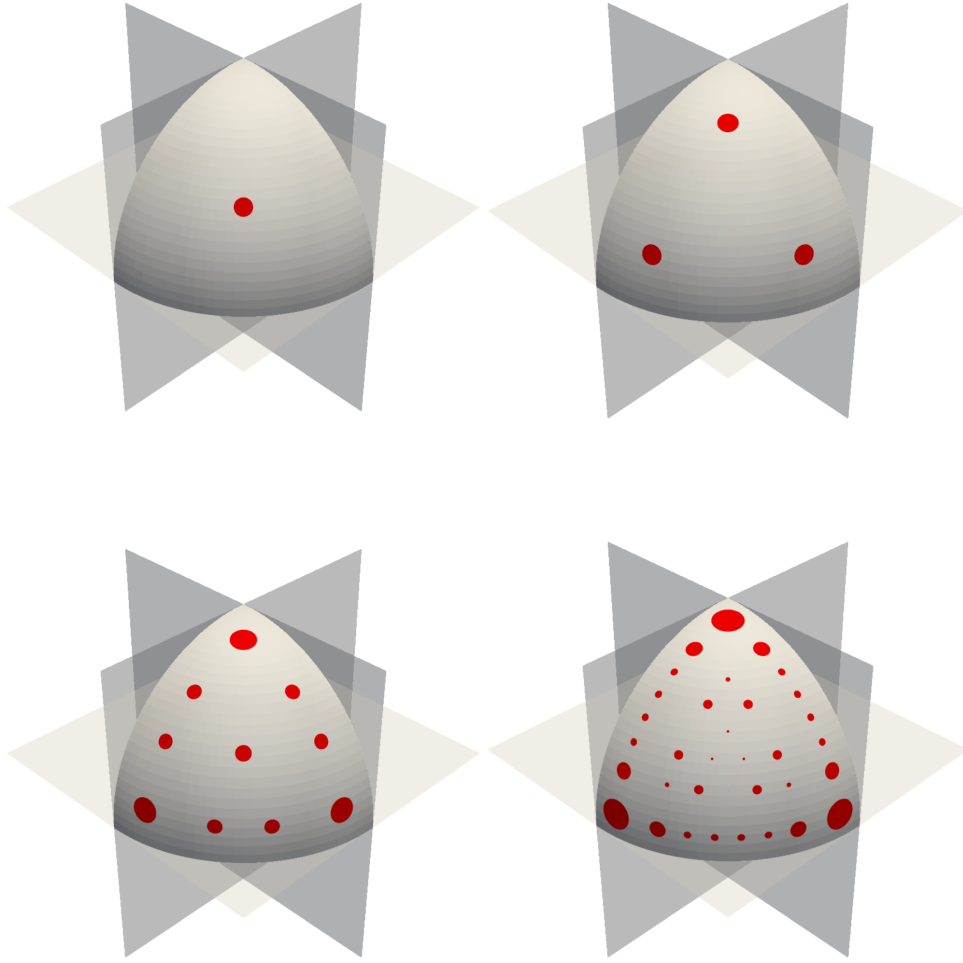


Figure 2.3: Level symmetric angular quadratures of order 2, 4, 8 and 16. Each red disc is an ordinate and its size denotes its weight relative to the others.

The integrals on the RHS of equation (2.2.4) are computed as follows [103]:

$$b_{m,l} = \frac{2}{\pi} \int_0^1 d\mu \int_0^{\pi/2} d\omega \mu^l \eta^m = \begin{cases} \frac{1}{m+1}, & \text{if } l = 0 \\ \frac{1}{m+l+1} \prod_{i=1}^{l-1} \frac{i}{m+i}, & \text{for } i \text{ odd where } l \geq 2 \end{cases}$$

To turn equation (2.2.4) into a linear system, we sort the set of pairs of even numbers (l, m) into lexicographic order and define a mapping $i \mapsto (l(i), m(i))$ such that equation (2.2.4) may be written as:

$$\mathbf{A}\mathbf{w} = \mathbf{b}, \quad \text{where} \quad (\mathbf{A})_{i,k} = \mu_k^{l(i)} \eta_k^{m(i)}, \quad (\mathbf{b})_i = b_{k(i),l(i)} \quad (2.2.5)$$

We may then solve equation (2.2.5) for the set of weights $\mathbf{w}$ on the positive octant. We now have the set of ordinates and the set of weights on the primary octant. The final step is to build the full quadrature set on the positive hemisphere for 2D problems and the full sphere for 3D problems. To do so we simply reflect the ordinates in the primary octant about each of the axes and finally we re-normalise the angular quadrature weights such that:

$$\sum_{k=1}^K w_k = 4\pi.$$

Ray Effects

Because there are only a finite number of directions along which neutrons are streaming, then at a given point the sources which are "visible" from that point may not be the same as would be the case if the angle were treated continuously and exactly, with say an infinite P_N expansion. This property of the S_N approximation manifests itself as **ray effects**, which are spurious oscillations in the scalar flux solution of the S_N equations. Ray effects are intrinsic to the S_N equations, they are not improved by increasing the mesh resolution, as we demonstrate in figure 2.4.

In figure 2.4, we may see that the cause of the ray effects is a dip in the flux at the interface between the fuel, where there is a fission source and the moderator, where there is no source. This problem has reflected boundary conditions so this dip is then reflected at the boundary back into the domain. Ray effects are most pronounced in problems with isolated sources within optically thin regions, such as a point source in a pure absorber. In optically thick and highly scattering problems, ray effects are less prevalent. One way to ameliorate ray effects is to simply increase the angular quadrature order until they have been reduced to an acceptable level. However, ray effects are known to be resistant to higher quadrature order therefore the increase in computational effort may not be feasible [122]. One important method for reducing ray effects is Lathrops' fictitious source method [123], which was developed further in [124, 125] and is based upon building S_N schemes which are more strongly coupled and are closer to the P_N equations, which do not exhibit ray effects. Ray effect mitigation is still an active area

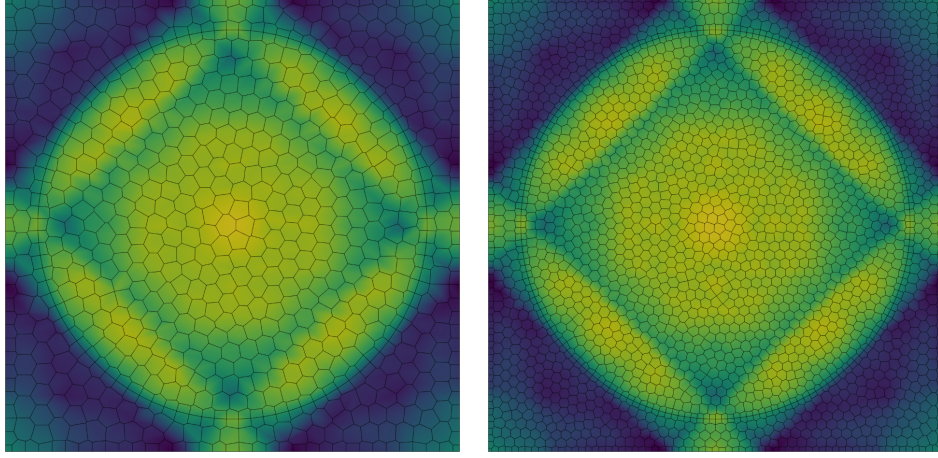


Figure 2.4: Ray effects present in the solution to a S_{16} pincell problem, left image shows a less spatially refined mesh and the right image a more spatially refined mesh.

of research within neutron transport theory. However, it is outside the scope of this thesis so we refer the reader to [122] for more on the topic. It is important to note that no ray effect mitigation scheme has been implemented as part of this thesis.

2.2.3 The Diffusion Approximation

It is well-known that as $\Sigma_t \rightarrow \infty$ and $\Sigma_a \rightarrow 0$ that the NTE, which is a hyperbolic equation, begins to behave like an elliptic equation [126, 127], a property known as the asymptotic diffusion limit. This property means that the neutron diffusion equation (NDE) may be a reasonable low-order approximation to the NTE for highly scattering problems. The NDE has found widespread use within the modelling and simulation of nuclear reactor physics problems. For instance, the NDE is used extensively within whole-core nuclear reactor physics calculations using coarse mesh finite difference (CMFD) [128] and nodal spatial discretisation schemes [66]. This is because the NDE is much less computational demanding to solve compared to the NTE. The NDE is also used for accelerating the solution of the NTE. In this case, the solution of the NDE is used to accelerate the solution of the S_N angularly discretised NTE [129].

To derive the diffusion equations let us first recall the definition of the scalar flux and introduce the flux current:

$$\phi(\mathbf{x}, E) = \int_{4\pi} d\Omega \psi(\mathbf{x}, E, \Omega), \quad \mathbf{J}(\mathbf{x}, E) = \int_{4\pi} d\Omega \Omega \psi(\mathbf{x}, E, \Omega).$$

Following the steps presented in chapter 4 of [44], we may derive the diffusion approximation by first assuming linear isotropic flux in angle, which is a P_1 expansion which may be rearranged to yield P_1 equations, which are given by equations (2.2.6) and (2.2.7):

$$\begin{aligned} \nabla \cdot \mathbf{J}(\mathbf{x}, E) + \Sigma_t(\mathbf{x})\phi(\mathbf{x}, E) \\ = \end{aligned} \tag{2.2.6}$$

$$\int_0^\infty dE' \Sigma_{s,0}(\mathbf{x}, E' \rightarrow E) \phi(\mathbf{x}, E') + \chi(\mathbf{x}, E) \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E' \rightarrow E) \phi(\mathbf{x}, E')$$

and

$$\frac{1}{3} \nabla \phi(\mathbf{x}, E) + \Sigma_t(\mathbf{x}, E) \mathbf{J}(\mathbf{x}, E) = \int_0^\infty dE' \Sigma_{s,1}(\mathbf{x}, E' \rightarrow E) \mathbf{J}(\mathbf{x}, E') \quad (2.2.7)$$

Next, we make the following assumption about the neutron currents [44, 130]:

$$\int_0^\infty dE' \Sigma_{s,1}(\mathbf{x}, E' \rightarrow E) \mathbf{J}(\mathbf{x}, E') = \int_0^\infty dE' \Sigma_{s,1}(\mathbf{x}, E \rightarrow E') \mathbf{J}(\mathbf{x}, E). \quad (2.2.8)$$

Equation (2.2.8) only holds true in the thermal energy range. At higher energies the diffusion approximation diverges from the P_1 equations. Letting:

$$\Sigma_{s,1}(\mathbf{x}, E) = \int_0^\infty dE' \Sigma_{s,1}(\mathbf{x}, E \rightarrow E')$$

and substituting equation (2.2.8) into equation (2.2.7) we get:

$$\frac{1}{3} \nabla \phi(\mathbf{x}, E) + \Sigma_t(\mathbf{x}, E) \mathbf{J}(\mathbf{x}, E) = \Sigma_{s,1}(\mathbf{x}, E) \mathbf{J}(\mathbf{x}, E)$$

therefore

$$\mathbf{J}(\mathbf{x}, E) = -\frac{1}{3 \left(\Sigma_t(\mathbf{x}, E) - \Sigma_{s,1}(\mathbf{x}, E) \right)} \nabla \phi(\mathbf{x}, E) \quad (2.2.9)$$

The diffusion coefficient may then be defined as:

$$D(\mathbf{x}, E) = \frac{1}{3 \left(\Sigma_t(\mathbf{x}, E) - \Sigma_{s,1}(\mathbf{x}, E) \right)} = \frac{1}{3 \Sigma_{tr}(\mathbf{x}, E)}$$

where Σ_{tr} is the transport-corrected cross section and is given by:

$$\Sigma_{tr}(\mathbf{x}, E) = \Sigma_t(\mathbf{x}, E) - \Sigma_{s,1}(\mathbf{x}, E) = \Sigma_t(\mathbf{x}, E) - \bar{\mu} \Sigma_{s,0}(\mathbf{x}, E)$$

where $\bar{\mu}$ is the average scattering angle. Finally, we substitute equation (2.2.9) which is known as Ficks' Law into equation (2.2.6) to yield the neutron diffusion equation:

$$\begin{aligned} & \nabla \cdot (D(\mathbf{x}, E) \nabla \phi(\mathbf{x}, E)) + \Sigma_t(\mathbf{x}) \phi(\mathbf{x}, E) \\ &= \\ & \int_0^\infty dE' \Sigma_{s,0}(\mathbf{x}, E' \rightarrow E) \phi(\mathbf{x}, E') + \frac{\chi(\mathbf{x}, E)}{4\pi} \int_0^\infty dE' \nu \Sigma_f(\mathbf{x}, E' \rightarrow E) \phi(\mathbf{x}, E') \end{aligned} \quad (2.2.10)$$

2.3 Energy Discretisation

The simplest means of discretising energy is to treat equation (2.1.4) as a one-speed equation. This is equivalent to integrating equation (2.1.4) over the full energy domain and solving for the energy-integrated angular flux ([98] section 2.2). This crude model is useful in describing

situations where the neutron energy distribution is very fast or very slow [98]. Additionally, the one-speed NTE is an invaluable model for establishing the mathematical properties of the space and angle dependence, such as existence and uniqueness of solutions and error estimates for spatial and angular discretisations, without the unnecessary complexity of maintaining energy dependence.

In most problems of interest, such as the study of light-water reactors, the neutron energy spectrum ranges from as low as 0.01eV up to 10MeV [102]. Over this energy range many materials exhibit detailed and complicated dependence on energy. In particular, all heavier nuclides exhibit a very large number of resonances: energies at which the probability of an interaction between the neutron and the nucleus becomes high. In figure 2.5 we present an example: the total microscopic cross section σ_t for Plutonium-239. This figure was produced using data from the ENDF-BVIII nuclear data library and visualised using JANIS [131]. In order to capture this complex dependence on energy, Hurwitz and Ehrlich first developed the multigroup method of energy discretisation within the context of the age diffusion equation [132]. It has since become the standard method of discretising energy. Over the decades a great deal of research has been conducted to both compile detailed data on the continuous energy dependence of many materials, such as ENDF [45, 133] and JEFF [134], and to design nuclear data processing codes such as NJOY [130].

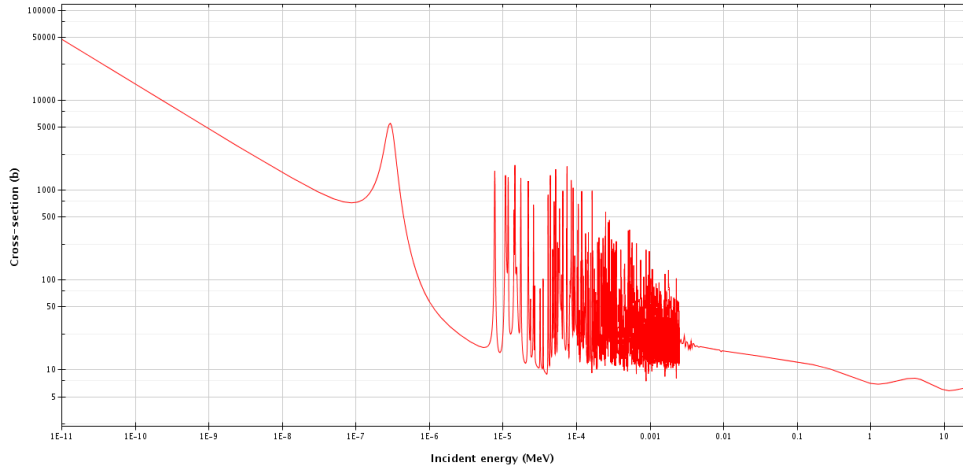


Figure 2.5: The microscopic total cross section $\sigma_t(E)$ as a function of incident neutron energy.

In the multigroup approximation, one begins by choosing upper and lower limits for the effective energy range and then splitting the range into G intervals:

$$E_{\min} = E_G < E_{G-1} < \dots < E_2 < E_1 < E_0 = E_{\max}.$$

and approximating the full integral over energy with:

$$\int_0^\infty dE f(E) \approx \int_{E_{\min}}^{E_{\max}} dE f(E) = \sum_{g=1}^G \int_{E_g}^{E_{g-1}} dE f(E).$$

The quantities that must be preserved in the multigroup approximation are the reaction rates

and the neutron currents. If these are not preserved then neutron conservation will be violated and vital quantities such as the K_{eff} will be incorrect. The multigroup angular flux, scalar flux and neutron cross sections are given below:

$$\begin{aligned}
\psi_g(\mathbf{x}, \boldsymbol{\Omega}) &= \int_{E_g}^{E_{g-1}} dE \psi(\mathbf{x}, E, \boldsymbol{\Omega}), \\
\phi_g(\mathbf{x}) &= \int_{E_g}^{E_{g-1}} dE \phi(\mathbf{x}, E), \quad \phi_{l,g}^m(\mathbf{x}) = \int_{E_g}^{E_{g-1}} dE \phi_l^m(\mathbf{x}, E) \\
\Sigma_t^g &= \frac{\int_{E_g}^{E_{g-1}} dE \Sigma_t(\mathbf{x}, E) \Psi(\mathbf{x}, E)}{\int_{E_g}^{E_{g-1}} dE \Psi(\mathbf{x}, E)}, \\
\Sigma_{s,l}^{g' \rightarrow g}(\mathbf{x}) &= \frac{\int_{E_g}^{E_{g-1}} dE \int_{E_g'}^{E_{g-1}'} dE' \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \Psi(\mathbf{x}, E')}{\int_{E_g}^{E_{g-1}} dE \Psi(\mathbf{x}, E)}, \\
\nu \Sigma_f^g(\mathbf{x}) &= \frac{\int_{E_g}^{E_{g-1}} dE \nu \Sigma_f(\mathbf{x}, E) \Psi(\mathbf{x}, E)}{\int_{E_g}^{E_{g-1}} dE \Psi(\mathbf{x}, E)}, \\
\chi_g(\mathbf{x}) &= \int_{E_g}^{E_{g-1}} dE \chi(\mathbf{x}, E), \quad q_g(\mathbf{x}, \boldsymbol{\Omega}) = \int_{E_g}^{E_{g-1}} dE q(\mathbf{x}, E, \boldsymbol{\Omega}).
\end{aligned}$$

The quantity $\Psi(\mathbf{x}, E)$ found in the expressions for the group cross sections is the neutron weighting spectrum [102], which is an approximation to the scalar neutron flux. The computation of $\Psi(\mathbf{x}, E)$ and other aspects of nuclear data preparation are wide-ranging areas of research and are outside the scope of this thesis so we refer the reader to [103, 130, 135] for a more thorough discussion. For the remainder of the thesis we shall simply assume the existence of pre-computed multigroup cross sections which are typically provided in the specification of nuclear reactor physics verification and validation (V&V) benchmark test cases. By integrating the NTE (equation (2.1.3)) over each energy interval $[E_g, E_{g-1}]$ and introducing the group averaged quantities defined before into the NTE one yields the multigroup NTE:

$$\begin{aligned}
&\boldsymbol{\Omega} \cdot \nabla \psi_g(\mathbf{x}, \boldsymbol{\Omega}) + \Sigma_t^g(\mathbf{x}) \psi_g(\mathbf{x}, \boldsymbol{\Omega}) \\
&= \\
&\sum_{g'=1}^G \sum_{l=0}^L \frac{2l+1}{4\pi} \Sigma_{s,l}^{g' \rightarrow g}(\mathbf{x}) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) \phi_{l,g}^m(\mathbf{x}) + \frac{\chi_g(\mathbf{x})}{4\pi} \sum_{g'=1}^G \nu \Sigma_f^{g'}(\mathbf{x}) \phi_g(\mathbf{x}) + q_g(\mathbf{x}, \boldsymbol{\Omega}). \quad (2.3.1)
\end{aligned}$$

The reason for the multigroup approximation should now be clear: it has turned the continuous-energy NTE, with its complicated dependence on energy, into a system of coupled hyperbolic PDEs, with no dependence upon energy. Finally, following a similar procedure for the continuous-energy neutron diffusion equation (equation (2.2.10)), we arrive at the multigroup NDE:

$$-\nabla \cdot (D^g(\mathbf{x}) \nabla \phi_g + \Sigma_r^g(\mathbf{x}) \phi_g) = \sum_{g' \neq g}^G \Sigma_s^{g' \rightarrow g}(\mathbf{x}) \phi_{g'} + \chi^g(\mathbf{x}) \sum_{g'=1}^G \nu \Sigma_f^{g'}(\mathbf{x}) \phi_{g'} \quad (2.3.2)$$

The PDEs in equations (2.3.1) and (2.3.2) are both coupled by the scattering and prompt fission

neutron sources. Therefore, the structure of the matrices given in equation (2.3.3) determines the coupling between energy groups :

$$(\Sigma_S(\mathbf{x}))_{g,g'} = \Sigma_s^{g' \rightarrow g}(\mathbf{x}), \quad (\Sigma_F)_{g,g'} = \chi_g \Sigma_f^{g'}(\mathbf{x}) \quad (2.3.3)$$

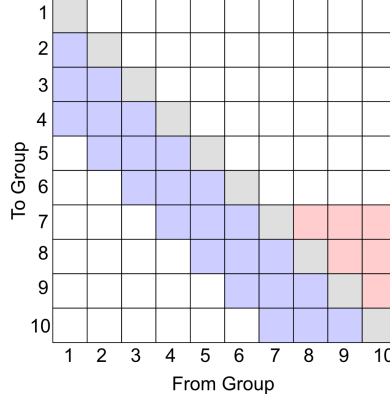


Figure 2.6: Structure of a typical group-to-group scattering matrix for a nuclear reactor physics problem with neutron up-scattering from lower energy neutron groups.

In pressurised water reactors (PWRs) and boiling water reactors (BWRs), where the energy of neutrons ranges from the thermal region to the fast region, the structure of a typical group-to-group neutron scattering matrix looks like figure 2.6, with most groups only exhibiting down-scattering (shown in blue in figure 2.6) and a small number of thermal groups which also exhibit up-scattering (shown in red in figure 2.6). The grey boxes represent within-group scattering. This is where the neutron has undergone a scattering event without a change in energy occurring.

2.4 Spatial Discretisation

After the multigroup approximation in energy, and the S_N or P_N approximations in angle have been applied to the NTE, one is left with a semi-discretised system of coupled partial differential equations (PDEs) in the spatial variables. Therefore, to yield a system of linear algebraic equations which may be solved numerically a spatial discretisation must also be applied to these coupled PDEs. Since the 1950s, many different methods have been developed to discretise these coupled PDEs, each with their own advantages and disadvantages. In this section we shall briefly review a few of the most important of these methods and discuss where the virtual element method resides in relation to them.

2.4.1 The Finite Difference Method

The finite difference method (FDM) is one of the oldest and in many ways one of simplest spatial discretisations. To implement an FD scheme we first define a Cartesian grid of points in space and define the value of the unknown field at each point as the degrees of freedom (DoF) which

we are solving for. The partial derivatives found in the PDE under consideration may therefore be replaced by finite difference stencils. These are derived by considering a Taylor expansion about each point and truncating at a desired order of accuracy. These FD stencils couple a DoF to its immediate neighbours in the Cartesian grid, therefore a PDE with continuous partial derivatives in space is converted into a linear algebraic system of equations amenable to numerical solution. The Cartesian structure of the unknowns allows for direct indexing of degrees of freedom and for a very simple implementation of multigrid preconditioners for iterative methods [136]. These two features, along with the simple implementation, make the FDM an excellent spatial discretisation for problems with strictly Cartesian geometry. However, the restriction to Cartesian geometry is also the biggest limitation of the FDM; most nuclear reactors have complicated geometries which often need to be resolved more accurately. This provides the motivation behind more advanced spatial discretisations such as the finite element method (FEM) and Isogeometric analysis (IGA).

A slightly different approach to finite differences which is often used in reactor physics is the diamond-difference (DD) scheme [117]. In the DD scheme, the degrees of freedom are the cell-averaged angular flux and the face-averaged angular fluxes and the assumption is made that the angular flux varies linearly inside each cell [137]. Using this assumption, one can derive linear equations in the unknowns which, when using the S_N method in angle, can be swept from downstream to upstream and solved cell-by-cell. A drawback of the DD scheme is the potential non-positivity of solutions [137]. In [138] the weighted diamond difference method was presented to solve this issue, albeit at the cost of slower convergence rates.

2.4.2 The Finite Element Method

The conventional, conforming, Galerkin finite element method [80, 139, 140, 141] is a method of discretising partial differential equations (PDEs). The basic idea is to pose the PDE as a variational problem in an infinite-dimensional function space where the true solution is proven to exist and is unique. The domain of interest is then decomposed into a set of non-overlapping elements. On each element a local polynomial function space is defined which has desirable local approximation properties. These local polynomial spaces are then combined in such a way as to construct a global, finite-dimensional approximation space in which the approximate solution is sought. When this finite dimensional space is a subspace of the original infinite-dimensional space, the method is said to be *conforming*. Otherwise, the spatial discretisation is called a *nonconforming* finite element method. As the dimension of the finite-dimensional subspace grows to infinity the discrete solution becomes arbitrarily close to the true solution in a chosen norm. Due to the local approximation properties of the local spaces, the error between the true solution and the discrete solution may be bounded in terms of the maximum element diameter of the mesh (h), the polynomial order of local spaces (p) and the data of the problem (the smoothness of boundary conditions, forcing function and the boundary of the domain).

The finite element method (FEM) has been used in nuclear reactor physics since the 1970s and

represented a major breakthrough in the types of reactor geometries which could be analysed accurately. This is because the FEM allows the use of unstructured meshes, as opposed to the Cartesian meshes of the finite difference codes that preceded it. The continuous-Galerkin FEM (CG-FEM) was applied to the neutron diffusion equation in [83, 84] where in [83] the authors focused exclusively on the steady-state mono-energetic NDE and the steady-state multigroup NDE. While in [84] the authors apply the CG-FEM with Hermite polynomials to both the steady-state, multigroup NDE in multiple spatial dimensions. and the point-kinetics equations. In [142] the authors solve the steady-state NDE with high-order CG-FEM (up to fifth order) and compare the solve time with finite differences. In [81] the authors present the first discontinuous-Galerkin FEM (DG-FEM) discretisation of the first-order form of the NTE and the first sweep solver based on unstructured triangular meshes. In [82] the authors formally analyse the well-posedness of the DG-FEM presented in [81] and prove error estimates in the L^2 norm. In [143] the authors present one of the earliest applications of phase-space finite elements to the 1D first-order NTE. The phase space finite element is the term used to describe the situation where finite element spaces are used to represent both the spatial domain and the angular domain. In [108] the authors extend the phase-space finite element to the 2D first-order NTE.

In the decades since these early publications, the finite element method has since become one of the most common spatial discretisations for both neutron diffusion and neutron transport codes in production today. Some notable examples include the **Rattlesnake** code [75] and the **ATILLA** code [144]. There have been many innovations in the application of the FEM to nuclear reactor physics since the early publications we listed above and we shall provide a non-exhaustive list of some important advances. In [145, 146] the authors presented h and hp adaptive FEM to solve the NDE. In [147] the author applies the piecewise linear discontinuous (PWLD) FEM with symmetric interior penalties to the NDE. In [148] the authors present a mixed-finite element method to discretise the NDE, where the diffusion operator is split into two coupled, first-order equations. In [149, 150, 151] goal-oriented, adaptive DG-FEM schemes to solve the S_N -NTE were presented. In [152] a 2D quadratic, polygonal DG-FEM was presented to solve the S_N -NTE.

The main limitation of the conventional FEM is that the mesh elements are either linear, or polynomial. Broadly speaking the geometry of elements may be categorised as:

- Sub-parametric: where the geometry of the finite element is described using a lower order polynomial than that of the finite elements themselves.
- Iso-parametric: where the geometry is described using the same polynomial order as that finite elements themselves, and usually the same basis.
- Super-parametric: where the geometry is described using a higher polynomial order than that of the finite elements themselves.

In all three of these categories, because they are all polynomial descriptions of geometry, there will be a discrepancy between some common reactor geometries, such as circular fuel pins and

annular cladding, and the FE mesh. The subsequent errors induced in important quantities of interest (QoIs), such as the effective multiplication factor K_{eff} , and fission rates in the assemblies, can be substantial. However, there have been recent advances in the FEM which solve this issue and we present two here. The first is the NURBS-enhanced FEM (NE-FEM) [88, 153], which takes an approach to encoding exact-geometry which most closely resembles our own. Therefore, we use a similar naming convention when presenting the NURBS-enhanced VEM. The NE-FEM uses NURBS curves and surfaces as the description of geometry for the edges and faces of standard finite elements, such as triangles/quadrilaterals in 2D and tetrahedra/hexahedra in 3D. A volume parameterisation of the element which exactly conforms to the NURBS boundary is then constructed and used to generate the shape-function nodal coordinates and the quadrature sets used to evaluate the integrals of bilinear forms. Therefore, all of the information encoding exact-geometry is embedded in the usual information used in FE codes: namely the shape function nodes and the quadrature rules on elements. In [88] the authors first present the 2D NE-FEM and in [153] the authors compare it to other approaches encoding higher-order geometry, in particular the isoparametric p -FEM. In [154] the authors present the generation of 2D exact-geometry quadrature sets for the NE-FEM and in [155] the 3D NE-FEM was presented. In [156] the authors present a 2D triangular mesh generator which preserve the NURBS geometry of the input domain. The second approach we shall discuss here is that presented by the authors of the **Nektar++** spectral/ hp finite element library [157, 158].

Rather than using NURBS to describe the geometry, their implementation uses a more conventional iso-parametric or super-parametric polynomial approach but with advanced mesh-quality improvement techniques to ensure that the geometric error is not the limiting factor in the convergence rate of the method. Some examples include using optimisation of a variational problem to improve high-order mesh quality [159] and solving a PDE problem to construct curvilinear quad meshes [160]. To the best of our knowledge the NE-FEM has not yet been applied in nuclear reactor physics to date. However, high-order meshes for the p -FEM have been applied to the multigroup S_N -NTE in [85]. Additionally, in [86] the authors also present a diffusion-synthetic acceleration scheme for the high-order mesh S_N scheme they presented in [85].

2.4.3 Isogeometric Analysis

Another recent innovation in spatial discretisation of PDEs is Isogeometric Analysis (IGA). First presented in [89], the fundamental idea in IGA is to extend the idea of isoparametric FEM, where the same basis is used to represent both the unknown solution field and the geometry of the domain, to use a basis of non-uniform rational B-spline (NURBS) basis functions. NURBS are the standard geometry description used in the computer-aided geometric design (CAGD) industry and are found in every major commercial CAGD package, including ACIS and Parasolid. The original vision for IGA was to bridge the gap between the CAGD description of the geometry, which is based on NURBS, and the FE analysis description of the geometry, which is based on linear meshes of triangles/quadrilaterals in 2D and tetrahedra/hexahedra in

3D. Since the original IGA paper [89], the literature has expanded substantially; in [161] the mathematical properties of the NURBS basis are formally analysed. In [162] efficient quadratures were developed for IGA. In [163] IGA using T-splines instead of NURBS was presented and in [164] an version of IGA using trimmed NURBS was presented. IGA has been applied to many areas of science and engineering, such as fluid dynamics and linear elasticity. We refer the reader to [2] for a thorough overview of the applications of IGA.

IGA has been applied to many problems in nuclear reactor physics. In [165] IGA was applied to the mono-energetic NDE and was later extended to the multigroup neutron diffusion equation in [166]. Subsequently, in [167] IGA was applied to the even-parity form the NTE with the P_N approximation in angle. Around the same time, the IGA was used to solve the multigroup S_N -NTE in [168] and was extended to include local h adaptivity with a hanging-node formulation in [169]. Finally, energy dependent meshes with local h adaptivity and field transfer operators between IGA meshes were implemented in [170]. In [171] IGA was applied to the self-adjoint angular flux (SAAF) form of the NTE with an S_N approximation in angle. In [172] IGA was applied to the weighted least-squares (WLS) form of the NTE, also with an S_N approximation in angle. Finally, in [173] spatially adaptive versions of IGA with hanging nodes were developed for both the SAAF-NTE and WLS-NTE with the S_N approximation in angle.

Chapter 3

A Review of Functional Analysis

We shall now briefly review the definitions and theorems from functional analysis which form the foundation of both finite element methods and virtual element methods. The primary sources we draw upon are [141, 174, 175, 176, 177]. This is included for convenience, as a quick reference for any definitions or theorems used elsewhere in the thesis and is not intended as a rigorous treatment of functional analysis.

3.1 Vector Space, Normed Space, Inner Product Space

Definition 3.1.1. (*Real Vector Space*) A real vector space V is a set with two operations: vector addition $+: V \times V \rightarrow V$ and scalar multiplication $\cdot: \mathbb{R} \times V \rightarrow V$ which satisfy:

- $u + v = v + u$ (*commutative*)
- $(u + v) + w = u + (v + w)$ (*associative*)
- $\alpha \cdot (u + v) = \alpha \cdot u + \alpha \cdot v$ (*distributive with respect to vector addition*)
- $(\alpha + \beta) \cdot u = \alpha \cdot u + \beta \cdot u$ (*distributive with respect to scalar addition*)

For all $u, v, w \in V$ and $\alpha, \beta \in \mathbb{R}$.

Additionally there must exist:

- An element $0 \in V$ called the additive identity: $0 + v = v$.
- For every $v \in V$ an element $-v$ called the additive inverse: $v + -v = 0$.
- An element 1 called the multiplicative identity: $1 \cdot v = v$.

Definition 3.1.2. (*Norm*)

A norm on a vector space V , denoted by $\|\cdot\|_V$ is a mapping $\|\cdot\|_V: V \rightarrow \mathbb{R}$ which has the following properties for all $x \in V$ and $\alpha \in \mathbb{R}$:

- **N1:** $\|x\|_V \geq 0$

- **N2:** $\|x\|_V = 0$ iff $x = 0$
- **N3:** $\|\alpha x\|_V = |\alpha| \|x\|_V$
- **N4:** $\|x + y\|_V \leq \|x\|_V + \|y\|_V$ (triangle inequality)

A vector space with a norm defined on its elements is a **normed vector space**.

Definition 3.1.3. (Seminorm) A seminorm on a vector space V , denoted by $|\cdot|_V$ is a mapping $|\cdot|_V : V \rightarrow \mathbb{R}$ which satisfies axioms N1, N3 and N4.

Definition 3.1.4. (Open Ball, Closed Ball) The open ball and closed ball in normed vector space V are defined as:

$$\begin{aligned} B(x_0, r) &= \{x \in V : \|x - x_0\|_V < r\} \quad \text{open ball} \\ \overline{B}(x_0, r) &= \{x \in V : \|x - x_0\|_V \leq r\} \quad \text{closed ball} \end{aligned}$$

Definition 3.1.5. (Open Set, Closed Set) A subset $M \subset V$ is said to be **open** if for every $x \in M$ there exists a ball $B(x, \epsilon) \subset M$. A subset $M \subset V$ is said to be **closed** if its complement $M^c = V - M$ is open.

Definition 3.1.6. (Closure) Let $M \subset V$ be an open set. A point $x \in V$ (which may or may not also be a member of M) is called a **limit point** if every ball $B(x, \epsilon)$ contains at least one point $y \in M$ different from x . The closure of M , denoted $\overline{M}$ is union of M and all of its limit points.

The closure $\overline{M}$ is the smallest closed set which contains M .

Definition 3.1.7. Diameter, Bounded Set, Bounded Sequence Let V a vector space and let $M \subset V$, the diameter of M is given by:

$$\delta(M) = \sup_{x, y \in M} \|x - y\|_V.$$

The set M is said to be **bounded** if $\delta(M)$ is finite. A sequence (x_n) in V is said to be bounded if the point set generated by the sequence is a bounded set.

Definition 3.1.8. (Basis and Dimension) A basis of vector space V is a minimal set of functions $\{\varphi_i\}_{i=1}^n$ which span V , meaning that every $v \in V$ may be written uniquely as:

$$v = \alpha_1 \varphi_1 + \dots + \alpha_n \varphi_n$$

The dimension of V , denoted $\dim(V)$, is number of basis functions which span the space.

Definition 3.1.9. (Bilinear Form) A bilinear form is a mapping $B(\cdot, \cdot) : V \times V \rightarrow \mathbb{R}$ which satisfies:

- $B(u + v, w) = B(u, w) + B(v, w)$ and $B(u, v + w) = B(u, v) + B(u, w)$.

- $B(\alpha u, v) = \alpha B(u, v)$ and $B(u, \alpha v) = \alpha B(u, v)$.

A bilinear form is said to be **bounded or continuous** if there exists a positive constant C such that:

$$B(u, v) \leq C \|u\|_V \|v\|_V \quad \forall u, v \in V$$

and bilinear form is said to be **coercive** if there exists a positive constant M which satisfies:

$$M \|u\|_V^2 \leq B(u, u) \quad \forall u \in V$$

Definition 3.1.10. (Continuous Bilinear Forms) Let X and Y be normed vector spaces. The set of continuous bilinear forms on $X \times Y$ is denoted $B(X \times Y; \mathbb{R})$ and is itself a vector space. Meaning that each bilinear form $a : X \times Y \rightarrow \mathbb{R}$ is a member of $B(X \times Y; \mathbb{R})$. Each $a \in B(X \times Y; \mathbb{R})$ has the norm:

$$\|a\|_{B(X \times Y; \mathbb{R})} = \sup_{x \in X, y \in Y} \frac{a(x, y)}{\|x\|_X \|y\|_Y}$$

Definition 3.1.11. (Inner Product Space) The inner product on vector space V , denoted $(u, v)_V$, is a bilinear form which satisfies:

- **IP1:** $(u, v)_V = (v, u)_V$ (symmetric)
- **IP2:** $(u, u)_V \geq 0$ and $(u, u)_V = 0$ iff $u = 0$. (positive definite)

A vector space with an inner product defined on its elements is an **inner product space**. The inner product also induces a norm:

$$\|u\|_V = \sqrt{(u, u)_V}.$$

Therefore, inner product spaces are also normed vector spaces.

Lemma 3.1.12. Schwarz Inequality, Triangle Inequality The inner product and the induced norm on V satisfy the Schwarz inequality:

$$|(u, v)_V| \leq \|u\|_V \|v\|_V \quad (3.1.1)$$

and the triangle inequality:

$$\|u + v\|_V \leq \|u\|_V + \|v\|_V$$

3.2 Banach Space, Hilbert Space

Definition 3.2.1. (Convergent Sequence) Sequence $\{u_n\}$ in normed vector space V is said to converge to u if for each $\epsilon > 0$ there exists an N such that:

$$\|u - u_n\|_V < \epsilon \quad \forall n > N$$

The difficulty with this definition of convergence is that, in order to verify that $\{u_n\}$ is convergent, we require knowledge of the limit u , which is dependent both on the space V and the choice of norm $\|\cdot\|_V$. It is useful to have a definition of convergence which does not require the limit u . We have exactly such a definition for Cauchy sequences.

Definition 3.2.2. (Cauchy sequence) A sequence $\{x_n\}$ in normed vector space V is said to be Cauchy if for an $\epsilon > 0$ there exists an N such that:

$$\|x_n - x_m\|_V < \epsilon \quad \forall n, m > N$$

Theorem 3.2.3. (Convergent Sequence) Every convergent sequence in normed space V is Cauchy.

Proof. See [174], theorem 1.4-5. □

Although every convergent sequence is Cauchy not every Cauchy sequence is convergent. The Cauchy convergence criterion allows one to determine if a sequence in normed vector space V is convergent without requiring knowledge of the limit point, which may not be a member of V .

Definition 3.2.4. (Compactness) A normed vector space V is compact if every sequence in V has a convergent sub-sequence.

Definition 3.2.5. (Completeness) A normed vector space V is said to be **complete** if every Cauchy sequence in V has a limit in V .

If a normed vector space is not complete then it is **incomplete**. All incomplete normed vector spaces may be completed by adding the limit points of all their Cauchy sequences [174].

Definition 3.2.6. (Banach Space) A Banach space is a complete, normed vector space.

Definition 3.2.7. (Hilbert Space) A Hilbert space is a complete inner product space. With completeness defined with respect to the norm induced by the inner product.

3.3 Linear Operator, Linear Functional

Definition 3.3.1. (Linear Operator) A linear operator T is a mapping from one vector space X to another vector space Y (both over the same field) which satisfies:

$$\begin{aligned} T(x + y) &= Tx + Ty, \\ T(\alpha x) &= \alpha Tx \end{aligned}$$

The first vector space is called the **domain** of T and is denoted $\mathcal{D}(T) \subset X$. The second vector space is called the **range** of T and is denoted $\mathcal{R}(T) \subset Y$. Additionally, the set of vectors for which $Tv = 0$ is called the **null space** of T , denoted $\mathcal{N}(T)$.

Definition 3.3.2. (Injection, Surjection, Bijection) A linear operator is said to be **injective** if it is "one-to-one", or more precisely that:

$$x_1 \neq x_2 \implies Tx_1 \neq Tx_2.$$

for $x_1, x_2 \in \mathcal{D}(T)$. An injective mapping T has an inverse $T^{-1} : \mathcal{R}(T) \rightarrow \mathcal{D}(T)$ which is also one-to-one. A mapping is $T : \mathcal{D}(T) \rightarrow Y$ is **surjective** or "onto Y " if $\mathcal{R}(T) = Y$. Finally, a mapping is **bijective** if it is both injective and surjective.

If T is injective then the operator T^{-1} is called the **inverse** of T . If $y = Tx$ then $x = T^{-1}y$ and we have the relations

$$\begin{aligned} T^{-1}Tx &= x \quad \forall x \in \mathcal{D}(T) \\ TT^{-1}y &= y \quad \forall y \in \mathcal{R}(T) \end{aligned}$$

Definition 3.3.3. (Bounded Linear Operator) A linear operator is bounded if there exists a positive real number m such that:

$$\|Tx\|_Y \leq m \|x\|_X \tag{3.3.1}$$

The smallest possible value of m for which equation (3.3.1) is satisfied is used to define the norm of the operator T .

Definition 3.3.4. (Linear Operator Norm) The norm of the linear operator T is defined as:

$$\|T\| = \sup_{\substack{x \in \mathcal{D}(T) \\ x \neq 0}} \frac{\|Tx\|_Y}{\|x\|_X}.$$

In the case where $x = 0$ the norm is defined as $\|T\| = 0$.

Letting X and Y be vector spaces over the same field, then the set of all bounded linear operators from X to Y is denoted $B(X, Y)$.

Theorem 3.3.5. (Space $B(X, Y)$) The set of bounded linear operators from normed vector space X to normed vector space Y is itself a normed vector space and has norm:

$$\|T\|_{B(X, Y)} = \sup_{\substack{x \in \mathcal{D}(T) \\ x \neq 0}} \frac{\|Tx\|_Y}{\|x\|_X}.$$

Theorem 3.3.6. (Completeness of $B(X, Y)$) If Y is complete, then $B(X, Y)$ is also com-

plete and is therefore a Banach space.

Before moving on to linear functionals, we shall introduce one more important concept: the isomorphic mapping in normed vector spaces.

Definition 3.3.7. (Isomorphism) Given two abstract normed vector spaces X and $\tilde{X}$, an isomorphism $T : X \rightarrow \tilde{X}$ is a mapping which preserves the structure of the normed vector space [174]. Meaning that the operations of vector addition, scalar multiplication and the norm are all preserved:

$$T(x + y) = Tx + Ty, \quad T(\alpha x) = \alpha Tx, \quad \|Tx\|_{\tilde{X}} = \|x\|_X$$

The existence of an isomorphism between X and $\tilde{X}$ implies, in the words of Erwin Kreyszig that [174]: we can regard X and $\tilde{X}$ as two copies of the same abstract space whenever the structure is the primary object of study. Other abstract spaces, such as Hilbert spaces for example, will have their own isomorphic mapping which preserves the structure particular to that space.

Now, we turn our attention to a special class of linear operator; one where the range of the operator lies within the real number line or complex plane. Such linear operators are called **linear functionals**. Linear functionals are so common and important that we use a different notation of lowercase letters. For example if x is a vector in real vector space V and f, l are linear functionals then $f(x)$ or $l(x)$ are both real numbers.

Definition 3.3.8. (Linear functional) A linear functional $l(\cdot) : V \rightarrow \mathbb{K}$ is a mapping which satisfies:

- $l(u + v) = l(u) + l(v)$.
- $l(\alpha u) = \alpha l(u)$

where $\mathbb{K} = \mathbb{R}$ if V is a real vector space and $\mathbb{K} = \mathbb{C}$ if V is complex. A linear functional is said to be bounded if there exists a positive constant m such that:

$$|l(u)| \leq m \|u\|_V$$

Definition 3.3.9. (Dual Space) The set of all bounded linear functionals $l(\cdot)$ on vector space V is itself a vector space called the dual space, it is denoted V' . The norm of the dual space is:

$$\|l\|_{V'} = \sup_{u \in V} \frac{|l(u)|}{\|u\|_V}$$

Remark 1. Given that $V' = B(X, Y)$ with Y equal to $\mathbb{R}$ or $\mathbb{C}$, both of which are complete, then by theorem 3.3.6, V' is also complete.

It is possible to also find the dual of the dual space, known as the **second dual space** and denoted with V'' for vector space V . Letting $x \in V$ and $f \in V'$, members of V'' are defined by

keeping x fixed and letting f vary:

$$g_x(f) = f(x)$$

where we have used the x subscript to denote that g corresponds to a particular, fixed value of x . The norm of the space V'' is given by (see [174], lemma 4.6-1):

$$\|g_x\|_{V''} = \|x\|_V$$

For every $x \in V$ there is a $g_x \in V''$, this is called the **canonical mapping**, and it is denoted $C : V \rightarrow V''$ where $g_x = Cx$.

Definition 3.3.10. (Embedding) A vector space X which is isomorphic with a subspace of Y is said to be **embeddable** in Y .

The canonical mapping is both injective (one-to-one) and linear, it is therefore an isomorphism between V and $\mathcal{R}(C) \subset V''$.

Definition 3.3.11. (Reflexivity) A normed vector space is **reflexive** if, for the canonical mapping C , $\mathcal{R}(C) = V''$.

Theorem 3.3.12. (Finite Dimension) Every finite dimensional normed vector space is reflexive.

Theorem 3.3.13. (Hilbert Space) Every Hilbert space is reflexive.

3.4 Lebesgue Spaces, Weak Derivatives, Sobolev Spaces

Definition 3.4.1. Lebesgue Space The Lebesgue space on domain $\Omega \subset \mathbb{R}^d$ is defined as:

$$L^p(\Omega) := \left\{ u : \Omega \rightarrow \mathbb{R} : \|u\|_{L^p(\Omega)} \leq \infty \right\}$$

where

$$\|u\|_{L^p(\Omega)} := \left(\int_{\Omega} |u(\mathbf{x})|^p dx \right)^{1/p}, \quad 1 \leq p < \infty$$

and for $p = \infty$

$$\|u\|_{L^\infty(\Omega)} = \sup_{\mathbf{x} \in \Omega} |u(\mathbf{x})|$$

Lebesgue spaces are complete and are therefore examples of Banach spaces [176]. The case $p = 2$ is different to all other cases as the norm $\|u\|_{L^2(\Omega)}$ may be expressed as an inner product: $(u, u)_{L^2(\Omega)}$. Consequently, $L^2(\Omega)$ is also a Hilbert Space. Functions in Hilbert spaces are, in general, not regular enough for classical point-wise derivatives to be well-defined. Therefore, the notion of the weak derivative is required to characterise derivatives in an averaged sense [141].

Definition 3.4.2. (Support of Function) The support of a function φ is defined as:

$$\text{supp}(\varphi) := \text{closure}(\{\mathbf{x} \in \Omega : \varphi(\mathbf{x}) \neq 0\})$$

We shall assume that φ is 0 outside of its support. In this way we may extend φ to all of $\mathbb{R}^d$. We use $\text{supp}(\varphi) \subset\subset \Omega$ to denote that the support of φ is **compactly contained** in Ω . If Ω is bounded, this means φ will vanish as it approaches $\partial\Omega$. We refer the reader to [176], for a fuller discussion of this point.

Definition 3.4.3. Let $\mathcal{D}(\Omega)$ denote the set of all infinitely differentiable functions which are strictly non-zero in Ω and vanish in the neighbourhood of the boundary $\partial\Omega$:

$$\mathcal{D}(\Omega) = \{\varphi \in C^\infty(\Omega) : \text{supp}(\varphi) \subset\subset \Omega\}$$

Definition 3.4.4. We now define the set of locally integrable functions $L^1_{loc}(\Omega)$ as:

$$L^1_{loc}(\Omega) := \{f : f \in L^1(K) \quad \forall K \subset\subset \Omega\}$$

We use the multi-index to simplify the notation of weak derivatives. Let $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_d)$ be a d-tuple of integers, then we define the following:

$$|\alpha| := \sum_{i=1}^d \alpha_i, \quad \mathbf{x}^\alpha := \prod_{i=1}^d x_i^{\alpha_i}, \quad D^\alpha \varphi := \prod_{i=1}^d \left(\frac{\partial}{\partial x_i} \right)^{\alpha_i} \varphi$$

for $\mathbf{x} \in \mathbb{R}^d$ and $\varphi \in \mathcal{D}(\Omega)$. We are now in a position to introduce the weak derivative.

Definition 3.4.5. (Weak Derivative) Let function $f \in L^1_{loc}(\Omega)$, if there exists a function $g \in L^1_{loc}(\Omega)$ such that:

$$\int_{\Omega} g \varphi \, dx = (-1)^{|\alpha|} \int_{\Omega} f D^\alpha \varphi \, dx \quad \forall \varphi \in \mathcal{D}(\Omega)$$

then the weak derivative is given by $D^\alpha f = g$.

With the weak derivative and Lebesgue spaces defined we may now define Sobolev spaces.

Definition 3.4.6. (Sobolev Space) The Sobolev space on domain Ω is defined as:

$$W_s^p(\Omega) = \{u \in L^1_{loc}(\Omega) : \|u\|_{W_s^p(\Omega)} \leq \infty\}$$

where the Sobolev norm is defined as:

$$\|u\|_{W_s^p(\Omega)} = \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|_{L_p(\Omega)}^p \right)^{1/p}, \quad 1 \leq p < \infty$$

and for $p = \infty$

$$\|u\|_{W_s^\infty(\Omega)} = \max_{|\alpha| \leq s} \|D^\alpha u\|_{L^\infty(\Omega)}$$

Additionally, we have the following definition for the Sobolev semi-norm with $1 \leq p < \infty$:

$$|u|_{W_s^p(\Omega)} = \left(\sum_{|\alpha|=s} \|D^\alpha u\|_{L_p(\Omega)}^p \right)^{1/p}, \quad 1 \leq p < \infty$$

and for $p = \infty$

$$\|u\|_{W_s^\infty(\Omega)} = \max_{|\alpha|=s} \|D^\alpha u\|_{L^\infty(\Omega)}$$

When $p = 2$, the spaces W_s^2 are also inner-product spaces with the inner products given by:

$$(u, v)_{W_s^2(\Omega)} = \sum_{|\alpha| \leq s} (D^\alpha u, D^\alpha v)_{L_2(\Omega)}$$

Theorem 3.4.7. (Sobolev Spaces) *The spaces $W_s^p(\Omega)$ are complete and are therefore, Banach spaces. Consequently, the spaces $W_s^2(\Omega)$ are also Hilbert Spaces.*

Proof. See [141], theorem 1.3.2 □

In this thesis we shall use the standard notation $H^s(\Omega) = W_s^2(\Omega)$. The case $s = 1$ deserves additional explanation as it is the space we use most in this thesis. When $s = 1$ and the spatial dimension is $d = 2$ there are three index tuples such that $|\alpha| \leq 1$, these are $(0, 0)$, $(1, 0)$ and $(0, 1)$. Therefore, the inner product for $H^1(\Omega)$ is:

$$\begin{aligned} (u, v)_{H^1(\Omega)} &= (u, v)_{L^2(\Omega)} + \left(\frac{\partial u}{\partial x_1}, \frac{\partial v}{\partial x_1} \right)_{L^2(\Omega)} + \left(\frac{\partial u}{\partial x_2}, \frac{\partial v}{\partial x_2} \right)_{L^2(\Omega)} \\ &= (u, v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)}. \end{aligned}$$

For any $\Omega \subset \mathbb{R}^d$ we shall use the shorthand.

$$\begin{aligned} (u, v)_{L_2(\Omega)} &= (u, v)_{0, \Omega}, & (u, v)_{H^s(\Omega)} &= (u, v)_{s, \Omega} \\ \|u\|_{L_2(\Omega)} &= \|u\|_{0, \Omega}, & \|u\|_{H^s(\Omega)} &= \|u\|_{s, \Omega}, & |u|_{H^s(\Omega)} &= |u|_{s, \Omega} \end{aligned}$$

Finally, we also define the norm and seminorm on Cartesian product Sobolev spaces $[H^s(\Omega)]^M$ as follows:

$$\|v\|_{[H^s(\Omega)]^M}^2 = \sum_{i=1}^M \|v_i\|_{s, \Omega}^2, \quad |v|_{[H^s(\Omega)]^M}^2 = \sum_{i=1}^M |v_i|_{s, \Omega}^2$$

where $v = (v_1, \dots, v_M) \in [H^s(\Omega)]^M$.

3.5 Traces

Functions in Sobolev spaces are defined using the Lebesgue integral, consequently their value is undefined on domains with non-zero Lebesgue measure. This raises the issue of how to define the value of Sobolev space members on the boundary of a domain $\partial\Omega$. This is the role of the trace operator which, loosely speaking, maps a function $v \in W_1^p(\Omega)$ to the limit of a sequence of smooth approximations to v in $C^0(\overline{\Omega})$. If the limit exists then the restriction to the boundary $\gamma : C^0(\overline{\Omega}) \rightarrow C^0(\partial\Omega)$ is called the trace (see [176] chapter 5). With some abuse of notation we shall denote the extension of the trace with $\gamma : H^1(\Omega) \rightarrow L^2(\partial\Omega)$ and shall present a fundamental results for trace operators, the trace inequality.

Theorem 3.5.1. *Trace Inequality* *Let $v \in H^1(\Omega)$ then we have:*

$$\|\gamma v\|_{L^2(\partial\Omega)} \leq C_{tr} \|v\|_{0,\Omega}^{1/2} \|v\|_{1,\Omega}^{1/2}$$

Which may also be rewritten in the following simpler form:

Corollary 3.5.1.1. *Trace Inequality*

$$\|\gamma v\|_{L^2(\partial\Omega)} \leq C_{tr} \|v\|_{1,\Omega} \quad (3.5.1)$$

Proof. Using geometric-arithmetic mean inequality $2ab \leq a^2 + b^2$ the trace inequality may be rewritten as:

$$\begin{aligned} \|\gamma v\|_{L^2(\partial\Omega)}^2 &\leq C_{tr}^2 \|v\|_{0,\Omega} \|v\|_{1,\Omega} \\ &\leq \frac{C_{tr}^2}{2} (\|v\|_{0,\Omega}^2 + \|v\|_{1,\Omega}^2) \\ &= \frac{C_{tr}^2}{2} (2\|v\|_{0,\Omega}^2 + \|\nabla v\|_{0,\Omega}^2) \\ &\leq C_{tr}^2 (\|v\|_{0,\Omega}^2 + \|\nabla v\|_{0,\Omega}^2) \end{aligned}$$

the result follows. □

3.6 Well Posedness of Variational Problems

Following the presentation of [139], we shall now present three theorems, in order of increasing generality, used to establish the well-posedness of the abstract variational problem:

$$\begin{cases} \text{Find } u \in U \text{ which satisfies:} \\ a(u, v) = l(v) \quad \forall v \in V \end{cases} \quad (3.6.1)$$

where U and V are Banach spaces and V is reflexive. The space U is called the **solution space** and V is the **test space**. Additionally we assume that $a(\cdot, \cdot) \in B(U \times V; \mathbb{R})$ and

$l(\cdot) \in V' = B(V; \mathbb{R})$. Well-posedness in this context is defined by Hadamard:

Definition 3.6.1. (*Well-posedness*) *The abstract variational problem of equation (3.6.1) is said to be well posed if it admits only one unique solution and if there exists a $c > 0$ such that:*

$$\|u\|_U \leq c \|f\|_{V'}, \quad \forall f \in V'$$

Therefore, each of the three theorems presented in this section, if satisfied for a given problem, would imply well-posedness as defined by Hadamard.

3.6.1 Riesz Representation Theorem

The first theorem we shall present is the least general. In order to use this theorem to prove the well-posedness of problem (3.6.1) we must assume that:

- $U = V = H$ where H is a Hilbert space.
- In addition to being bounded, $a(\cdot, \cdot)$ is symmetric and coercive.

Boundedness, symmetry and coercivity mean that $a(\cdot, \cdot)$ defines an inner product on $H \times H$. Therefore, we may use the following theorem to prove well-posedness:

Theorem 3.6.2. *Riesz Representation Theorem:* *Let H be a Hilbert space with bilinear form $a(\cdot, \cdot) : H \times H \rightarrow \mathbb{R}$ which satisfies:*

$$a(u, v) = a(v, u), \quad a(u, v) \leq C \|u\|_H \|v\|_H, \quad \alpha \|u\|_V^2 \leq a(u, u), \quad l(v) \leq C_l \|v\|_V$$

Every continuous linear form $l(\cdot)$ on H may be uniquely represented by $u \in H$ as:

$$l(v) = a(u, v) \quad \forall v \in H$$

Proof. See [174], theorem 3.8-1. □

3.6.2 Lax-Milgram Theorem

The second theorem we shall present makes the assumptions:

- $U = V = H$ where H is a Hilbert space.
- In addition to being bounded, $a(\cdot, \cdot)$ is coercive but **not** symmetric.

Under these conditions we may invoke the Lax-Milgram theorem:

Theorem 3.6.3. *Lax-Milgram Theorem:* *Let H be a Hilbert space and let $a(\cdot, \cdot)$ be a bilinear form on H which is both **continuous** and **coercive** and let $l(v)$ be a **continuous** linear form on H :*

$$a(u, v) \leq C \|u\|_V \|v\|_V, \quad \alpha \|u\|_V^2 \leq a(u, u), \quad l(v) \leq C_l \|v\|_V$$

then the problem:

$$\begin{cases} \text{Find } u \in H \text{ which satisfies:} \\ a(u, v) = l(v) \quad \forall v \in H \end{cases}$$

has a unique solution which satisfies:

$$\|u\|_H \leq \frac{1}{\alpha} \|f\|_{H'}$$

Proof. See [141], theorem 2.7.7. □

3.6.3 Banach-Nečas-Babuška Theorem

Finally, we present the most general case: where the bilinear form is defined over the product of different Banach spaces, not Hilbert spaces. In this case we have no recourse to coercivity, which is a Hilbertian property therefore well-posedness must be established by other means. Following [139], we shall refer to the following well-posedness theorem as the Banach-Nečas-Babuška (BNB) theorem.

Theorem 3.6.4. (*Banach-Nečas-Babuška*) *Let U and V be Banach spaces, and let V also be reflexive. Let $a \in B(U \times V; \mathbb{R})$ and let $f \in V'$. The variational problem of equation (3.6.1) is well posed if and only if the following two conditions are satisfied:*

- **BNB1:** *There exists an $\alpha > 0$ such that:*

$$\inf_{u \in U} \sup_{v \in V} \frac{a(u, v)}{\|u\|_U \|v\|_V} \geq \alpha$$

- **BNB2:** *If for all $u \in U$ and for all $v \in V$ that:*

$$a(u, v) = 0 \implies v = 0$$

and the unique solution $u \in U$ satisfies the a priori estimate:

$$\|u\|_U \leq \frac{1}{\alpha} \|f\|_{V'}$$

3.7 Galerkin Methods

In this section we shall briefly review Galerkin methods in an abstract setting. Both the finite element method and the virtual element method are Galerkin methods therefore the definitions and results of this section are used throughout the thesis. The primary sources we draw upon for this section are [139, 177]. The basic idea in Galerkin methods, when applied to the abstract variational problem of equation (3.6.1) is to replace the infinite-dimensional Banach spaces U and V with finite-dimensional spaces U_h and V_h . The space U_h is the solution space

and V_h is the trial space. These spaces are equipped with the norms $\|\cdot\|_{U_h}$ and $\|\cdot\|_{V_h}$. We also define the space $U(h) = U + U_h$ and define the norm on $U(h)$ in any way which satisfies [139]:

$$\begin{aligned}\|u^h\|_{U(h)} &= \|u^h\|_{U_h} \quad \forall u^h \in U_h \\ \|u\|_{U(h)} &\leq c \|u\|_U \quad \forall u \in U\end{aligned}$$

The second of these conditions means U is continuously embedded in $U(h)$. The error term $u - u^h$ is a member of $U(h)$ therefore it must be bounded in the norm $\|u^h\|_{U(h)}$. The discrete, Galerkin approximation to the abstract variational problem of equation (3.6.1) therefore reads:

$$\begin{cases} \text{Find } u^h \in U_h \text{ which satisfies:} \\ a^h(u^h, v^h) = l^h(v^h) \quad \forall v^h \in V_h \end{cases} \quad (3.7.1)$$

The case where $U_h \neq V_h$ is termed a **Petrov-Galerkin** method and the case where $U_h = V_h$ is termed a **Bubnov-Galerkin** method. In this thesis we shall focus exclusively on the Bubnov-Galerkin method.

Definition 3.7.1. (Conformity) When $U_h \subset U$ and $V_h \subset V$ the Galerkin approximation is said to be conformal and is nonconformal otherwise.

Definition 3.7.2. (Approximability) The space U_h has the approximability property if for all $u \in U$

$$\lim_{h \rightarrow 0} \left[\inf_{u^h \in U_h} \|u - u^h\|_{U(h)} \right] = 0$$

The rate at which $\inf_{u^h \in U_h} \|u - u^h\|_{U(h)}$ converges to zero depends upon the approximation properties of the space U_h and is the topic of error analysis. Another important concept when designing a Galerkin method is consistency.

Definition 3.7.3. (Consistency) Let u be the solution to the exact variational problem of equation (3.6.1). The discrete variational method used to construct equation (3.7.1) is said to be consistent if the discrete bilinear form $a^h(\cdot, \cdot)$ may be extended from $U_h \times V_h$ to $U(h) \times V_h$ and if the exact solution u also satisfies the discrete problem of equation (3.7.1):

$$a^h(u, v^h) = l^h(v^h)$$

If the discrete problem does not satisfy both of these conditions it is said to be non-consistent.

The well-posedness of the approximate variational problem of equation (3.7.1), in the non-conformal and/or non-consistent case, is not inherited from the well-posedness of the exact variational problem and must therefore be established in the discrete setting. Specifically, the

discrete problem must satisfy conditions BNB1

$$\forall \alpha_h > 0, \quad \inf_{u^h \in U_h} \sup_{v^h \in V_h} \frac{a^h(u^h, v^h)}{\|u^h\|_{U_h} \|v^h\|_{V_h}} \geq \alpha_h.$$

Condition BNB2 is automatically satisfied as U_h and V_h are finite-dimensional and if $\dim(U_h) = \dim(V_h)$, as shall always be the case in this thesis.

Chapter 4

The Virtual Element Method

4.1 Introduction

The conventional finite element method is limited to meshes composed of triangles or quadrilaterals in 2D and tetrahedra or hexahedra in 3D. On polygonal/polyhedral elements, conventional conforming FE spaces are not rich enough to construct global conforming spaces which also attain the required local approximation properties. Therefore, in order to extend the FEM to polygonal meshes, one of two approaches must be taken. The first is to abandon conforming spaces altogether by removing inter-element continuity from the definition of the global FE space. This approach comes at the cost of a large increase in the number of degrees of freedom when compared to a conforming discretisation however, the flexibility gained with respect to element shape is substantial. An example of this approach is the Discontinuous Galerkin method [178, 179]. The second approach is to maintain conformity but to "enrich" the local spaces with non-polynomial functions. Examples of this approach are polygonal finite elements (PFEM) [180, 181, 182, 183] and generalised/extended finite element methods (GFEM/XFEM) [184, 185, 186, 187, 188]. The latter are also used to enrich local elements with discontinuous functions to accurately represent discontinuities in the solution without the need for re-meshing or interface tracking [187]. This has been particularly successful in solid mechanics for accurate modelling of cracks and dislocations.

The virtual element method (VEM) was first introduced in [90]. It is in many ways the latest evolution of mimetic finite differences [189, 190, 191, 192, 193], one which brings it into the Galerkin finite element framework. In other ways it may be viewed as another generalisation of the Galerkin finite element method (FEM), one which allows it to handle polygonal and polyhedral meshes. It is similar to both the PFEM and GFEM/XFEM in that it takes the enrichment approach to constructing local conforming spaces. However, where the VEM differs from these methods (and the thing which makes the VEM unique) is that these non-polynomial functions which are added to the local VEM space are neither known explicitly nor computed in practice. Rather, the VE space is constructed to have two components: the polynomial compo-

ment which guarantees the accuracy of the method and the non-polynomial component, which is manipulated solely through its degrees of freedom and its projection onto the polynomial component.

The first publication of the VEM in [90] introduced the canonical H^1 -conforming, C^0 -continuous virtual element for solving the Poisson problem on polygonal meshes. Around the same time the H^2 -conforming, C^1 -continuous virtual element method for plate bending problems [194] was also introduced. The VEM literature has since expanded substantially. In [195] the VEM was extended to include schemes with arbitrary regularity, i.e. C^m continuous with any $m > 0$, while in [196, 197] the H^1 -conforming VEM for general second-order elliptic problems was presented. Other major breakthroughs in VEM theory include the development of the computable L^2 projection [198], the reduction of internal degrees of freedom [199, 200], nonconforming discretisations [201, 202, 203, 204, 205], L^2 -orthogonalised local polynomial bases [206, 207, 208], rigorous analyses of stability estimates and error estimates [209, 210, 211]. The VEM has found useful application simulating a wide range of physical phenomena. Here we provide a broad but non-exhaustive list:

- Eigenvalue problems: [212, 213, 214, 215, 216].
- Friction and contact problems: [217, 218].
- Electromagnetism: [219].
- Stokes Problems: [220, 205].
- Hyperbolic and Parabolic problems: [221, 222].
- Linear Elasticity: [223, 224, 225, 226, 227].

Recently, formulations involving H^1 virtual elements with curvilinear edges have been presented [1, 228, 229], with the version presented in [1] being the one that has been implemented in this PhD to solve the Neutron Diffusion Equation (NDE) and the Neutron Transport Equation (NTE). One of the important advantages of the VEM presented in [1] is its simplicity. An algorithm developed for a straight-sided VEM, such as that in [90] can easily be modified to accommodate curvilinear edges.

In this chapter we review the curvilinear VEM presented in [1] and explain in detail how we have adapted it for the case where curvilinear polygons have NURBS edges. To this end, we organise this chapter as follows:

- Section 4.2 shall introduce the notation we use throughout the thesis to describe the VEM and shall present the definitions and assumptions made about the polygonal mesh. Finally, this section will introduce how we describe and compute multi-dimensional polynomial spaces.
- Section 4.3 will then explain how to construct the local curvilinear VEM space from first principles.

- Section 4.4 reviews the approximation properties of the local NE-VEM space, specifically a priori polynomial estimates on curved polygons and interpolation estimates for the local NE-VEM space.
- Section 4.5 discusses the practical implementation of the local VEM space, including the computation of local projections and bilinear forms.
- Finally, section 4.6 will provide some concluding remarks about the curvilinear VEM and will suggest some areas of future research.

4.2 Preliminaries

Vectors in $\mathbb{R}^d$ are denoted with lower case bold, e.g. $\mathbf{v}$, and the element at index i of a vector is denoted $(\mathbf{v})_i$. Matrices in $\mathbb{R}^{m \times n}$ are denoted using upper case bold e.g. $\mathbf{A}$ and the element at index (i, j) of a matrix is denoted $(\mathbf{A})_{i,j}$. Given a domain $E \subset \mathbb{R}^d$ the diameter, centroid and measure are denoted h_E , $\mathbf{x}_E$ and $|E|$ respectively. If E is a polygon then the number of vertices, edges and degrees of freedom present in it are denoted N_v^E , N_e^E and N_{dof}^E respectively.

4.2.1 The Polygonal Mesh

Let the domain of interest Ω be bounded by a set of n general NURBS curves, denoted $\{\Gamma_i\}_{i=1}^n$, with the only restriction being that each curve is a member of C^m , with $m \geq 0$. We shall let $\{\mathcal{T}_h\}_h$ denote a family of partitions of Ω where h denoted the maximum face diameter in the mesh: $h = \max_{F \in \mathcal{T}_h} h_F$. The mesh shall satisfy the usual restrictions: that faces are non-overlapping and share one or more edges only if they are adjacent. The faces themselves are each bounded polygons with possibly curved edges. The set of faces in the mesh $\mathcal{F}_h$ may be split into two disjoint sets: $\mathcal{F}_h^i$ and $\mathcal{F}_h^b$ such that $\mathcal{F}_h = \mathcal{F}_h^i \cup \mathcal{F}_h^b$. The set $\mathcal{F}_h^i$ are the internal faces, defined as the set of faces that are not incident with a geometric curve. The set $\mathcal{F}_h^b$ are the boundary faces, defined as those face which are incident with a geometric curve. An example is presented in figure 4.1.

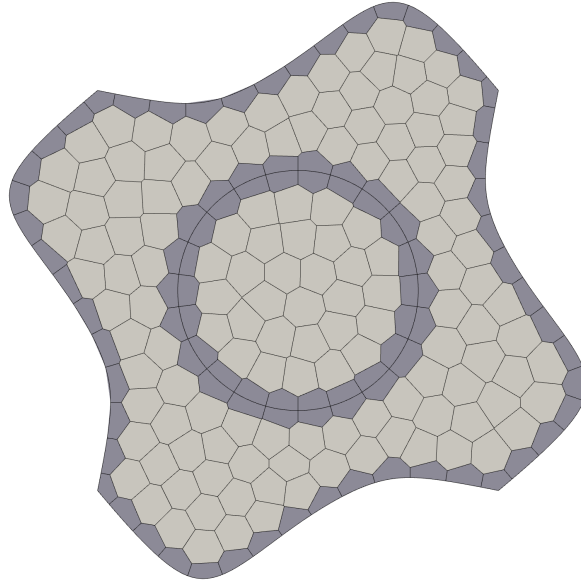


Figure 4.1: An example mesh with the boundary face set $\mathcal{F}_h^b$ highlighted in darker grey.

The reason for making the distinction between these two sets of faces is that the faces $F \in \mathcal{F}_h^i$ are guaranteed to have all their edges being straight lines. Therefore, the local curved virtual element spaces coincide with classical VEM described in [90, 230, 231]. The faces $F \in \mathcal{F}_h^b$ however, may have one or more of their edges be curvilinear. Therefore, the local curved VEM spaces diverge from the classical ones. Similarly we may split the set of edges in $\mathcal{T}_h$, denoted $\mathcal{E}_h$ into boundary edges $\mathcal{E}_h^b$ and internal edges $\mathcal{E}_h^i$. All edges may either be straight

or curvilinear depending on whether they are incident on either curved boundaries or curved internal interfaces.

Before stating the regularity assumptions placed on the mesh we must first introduce the concept of a star-shaped domain (taken from [141] definition 4.2.2).

Definition 4.2.1. (*Star shaped Polygon*)

A domain $F \in \mathbb{R}^2$ is said to be star-shaped with respect to a ball $B \subset F$ if for all $\mathbf{x} \in F$ the closed convex hull of $\{\mathbf{x}\} \cup B$ is a subset of F .

In figure 4.2 we present two domains, the one on the left is star-shaped with respect to the ball B but not with respect to ball B' . For a given point $\mathbf{x}$ the convex hulls of both $\{\mathbf{x}\} \cup B$ and $\{\mathbf{x}\} \cup B'$ are shown. The domain on the right is an example of a domain which is not star-shaped with respect to any ball.

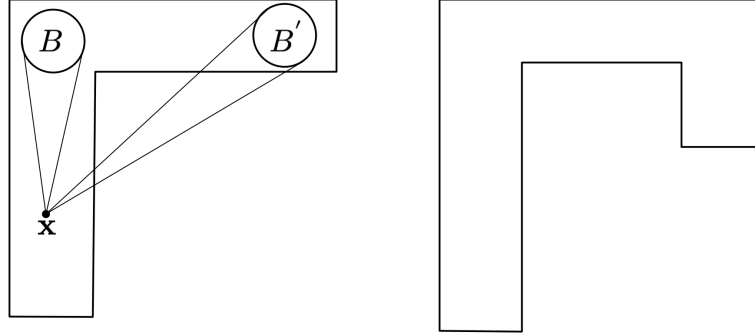


Figure 4.2: Illustration of a star shaped domain (left and with respect to B), and a non-star shaped domain.

We make the following mesh regularity assumptions:

Assumption 4.2.1. (*Mesh Regularity*)

There exists a positive constant η such that

- *for every face $F \in \mathcal{T}_h$ the length of all the edges of F have length greater than or equal to ηh_F .*
- *every face $F \in \mathcal{T}_h$ is star-shaped with respect to a ball B_F with radius ηh_F .*

4.2.2 Polynomial Basis

To compute 2D polynomial spaces on polygons we must first define a polynomial basis. In this section we define two such bases. The first is the set of scaled monomials, the second is an orthogonalised basis with respect to the L^2 norm on each polygon.

The Scaled Monomial Basis

In the VEM literature, the standard choice of basis for the space of polynomials $\mathbb{P}_p(E)$ on $E \subset \mathbb{R}^d$, are the scaled monomials. However, we shall also review them here in a more general

setting. The set of scaled monomials in d -dimensions and order p are given by:

$$\mathcal{M}_p^1(E) := \left\{ m_\alpha^1(\mathbf{x}) := \left(\frac{\mathbf{x} - \mathbf{x}_E}{h_E} \right)^\alpha : |\alpha| \leq p \right\}.$$

where $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_d^{\alpha_d}$ and $|\alpha| = \alpha_1 + \dots + \alpha_d$. The number of scaled monomials up to order p , and therefore the dimension of $\mathbb{P}_p(E)$, is given by

$$n_p := \dim(\mathbb{P}_p(E)) = \binom{p+d}{d} = \frac{\prod_{i=1}^d (p+i)}{d}$$

Any polynomial q_p has a unique representation in the scaled monomial basis:

$$q_p = \sum_{\alpha=1}^{n_p} c_\alpha m_\alpha^1 = \mathbf{c}^T \mathbf{m}_p^1, \quad c_\alpha \in \mathbb{R}$$

where $\mathbf{m}_p^1(\mathbf{x}) = [m_1(\mathbf{x}), \dots, m_{n_p}^1(\mathbf{x})]$ and $\mathbf{c} = [c_1, \dots, c_{n_p}]$. We may construct a mapping between a 1D index α and the exponent vector $\alpha = (\alpha_1, \dots, \alpha_d)$ as follows. The first task is to establish an ordering on the set $\mathbb{N}^d$, for which we choose the **degree-reverse lexicographic order** (see [232] chapter 2.2 for full discussion of the different orderings). This ordering involves first comparing the degree of each tuple and in the case of equal degree, breaking the tie by applying a reverse lexicographic order. More formally we say for $\alpha = (\alpha_1, \dots, \alpha_d)$ and $\beta = (\beta_1, \dots, \beta_d)$ that $\alpha \prec \beta$ if either:

1. $|\alpha| < |\beta|$.
2. Or $|\alpha| = |\beta|$ and $\alpha_i > \beta_i$ for the smallest i for which $\alpha_i \neq \beta_i$.

We present an example for the case $p = 2$ and $d = 3$ in equation (4.2.1)

$$\begin{aligned} 1 &\Leftrightarrow (0, 0, 0), \quad 2 \Leftrightarrow (1, 0, 0) \quad 3 \Leftrightarrow (0, 1, 0), \quad 4 \Leftrightarrow (0, 0, 1), \quad 5 \Leftrightarrow (2, 0, 0), \\ 6 &\Leftrightarrow (1, 1, 0), \quad 7 \Leftrightarrow (1, 0, 1), \quad 8 \Leftrightarrow (0, 2, 0), \quad 9 \Leftrightarrow (0, 1, 1), \quad 10 \Leftrightarrow (0, 0, 2), \end{aligned} \quad (4.2.1)$$

In practice, we store the tuples in an array `expon`. Therefore the mapping $\alpha \rightarrow \alpha$ may be performed in constant time by direct indexing: $\alpha = \text{expon}[\alpha]$. Given that the array `expon` is sorted with respect to the exponent tuple precedence operator, $\prec$, it is possible to compute the inverse mapping $\alpha \rightarrow \alpha$ in log time by using a binary search on `expon`. To compute the array `expon` we use algorithm 4.2 of [233] which partitions an integer p into d parts allowing for 0-valued partitions.

The Orthogonalised Basis

It was shown in [206, 207, 208] that, for elliptic problems, orthogonalising the scaled monomial basis with respect to the $L^2(E)$ norm resulted in improved conditioning of the final system of equations. In this thesis we use the method of orthogonalisation presented in [234]. This is a modified Gram-Schmidt (MGS) procedure. The set of orthogonalised basis functions, denoted

$\mathcal{M}_p^2(E)$, is defined as follows:

$$\mathcal{M}_p^2(E) := \left\{ m_\alpha^2(\mathbf{x}) := \sum_{\beta=1}^{n_p} (\mathbf{M}_p)_{\alpha,\beta} m_\beta^1 : (m_i^2(\mathbf{x}), m_j^2(\mathbf{x}))_{0,E} = \delta_{i,j} \right\}.$$

The $n_p \times n_p$ matrix $\mathbf{M}_p$ is a lower-triangular matrix computed using the MGS procedure. In order to efficiently perform the MGS procedure, one must be able to efficiently compute the $L^2(E)$ inner product of two polynomials, $p_k \in \mathbb{P}_k(E)$ and $p_l \in \mathbb{P}_l(E)$:

$$(p_k, q_l)_{0,E} = \left(\sum_{i=1}^{n_p} \eta_i m_i^1, \sum_{j=1}^{n_l} \zeta_j m_j^1 \right)_{0,E} = \sum_{i=1}^{n_p} \sum_{j=1}^{n_l} \eta_i \zeta_j (m_i^1, m_j^1)_{0,E}. \quad (4.2.2)$$

As will be explained in section 5.3.2 on numerical integration, the integrals of each of the scaled monomials are computed once using Lessere's method and stored for re-use. The product of two scaled monomials is also a scaled monomial, therefore the inner product $(m_i^1, m_j^1)_{0,E}$ has already been computed and stored. Consequently, the sum in equation (4.2.2) reduces to simply retrieving pre-computed integrals and multiplying them with coefficients. The MGS procedure of [234] proceeds as follows. We begin by initialising two sets of vectors in $\mathbb{R}^{n_p}$: $\{\boldsymbol{\eta}_i\}_{i=1}^{n_p}$ and $\{\hat{\boldsymbol{\eta}}_i\}_{i=1}^{n_p}$ with $\boldsymbol{\eta}_i$ initialised to the Euclidean basis vectors and $\hat{\boldsymbol{\eta}}_i$ initialised to the zero vector. Then we perform the steps of algorithm 1. Once the MGS procedure has been completed the coefficients of the $L^2(E)$ orthonormalised basis $\{\hat{\boldsymbol{\eta}}_i\}_{i=1}^{n_p}$ are assigned to be the rows of $\mathbf{M}_p$. Using $\mathbf{M}_p$ the orthogonalised basis functions $\mathbf{m}_p^2 = [m_1^2, \dots, m_{n_p}^2]$ may be written as $\mathbf{m}_p^2 = \mathbf{M}_p \mathbf{m}_p^1$. Additionally, we may perform the inverse map by simply inverting $\mathbf{M}_p$, $\mathbf{m}_p^1 = (\mathbf{M}_p)^{-1} \mathbf{m}_p^2$.

Remark 2. Both the scaled monomial basis and the orthogonalised basis possess the property that $\mathcal{M}_q^1(E) \subset \mathcal{M}_p^1(E)$ and $\mathcal{M}_q^2(E) \subset \mathcal{M}_p^2(E)$, where $q < p$. An important implication of this property is that the matrix $\mathbf{M}_q$ is a submatrix of $\mathbf{M}_p$ consisting of the first n_q rows and columns.

Algorithm 1 The MGS procedure of [234].

```

for  $i = 1$  to  $n_p$  do
  for  $j = 1$  to  $i$  do
     $r_{i,j} \leftarrow (\boldsymbol{\eta}_i^T \mathbf{m}_p, (\hat{\boldsymbol{\eta}}_j)^T \mathbf{m}_p)_{0,E}$ 
     $\boldsymbol{\eta}_i \leftarrow \boldsymbol{\eta}_i - r_{i,j} \hat{\boldsymbol{\eta}}_j$ 
  end for
   $r_{i,i} \leftarrow \sqrt{(\boldsymbol{\eta}_i^T \mathbf{m}_p, \boldsymbol{\eta}_i^T \mathbf{m}_p)_{0,E}}$ 
   $\boldsymbol{\eta} \leftarrow (1/r_{i,i}) \boldsymbol{\eta}$ 
   $\hat{\boldsymbol{\eta}} \leftarrow \boldsymbol{\eta}$ 
end for

```

Polynomial Derivatives

The derivative of a polynomial is a polynomial of lower order. Therefore, the mapping between a polynomial and its derivative may be represented as a matrix. Letting $\partial_{x_j}^n := \partial^n / \partial x_j^n$, we may

represent the derivative operator $\partial_{x_j}^n : \mathbb{P}_p(F) \rightarrow \mathbb{P}_{p-n}(F)$ with a matrix $\boldsymbol{\partial}_j^n$. Letting $q_p = \mathbf{c}^T \mathbf{m}_p^1$ be an order p polynomial expanded in $\mathcal{M}_p^1(E)$ then the derivative is given by:

$$\partial_{x_j}^n q_p = \mathbf{c}^T \left(\boldsymbol{\partial}_j^n \mathbf{m}_p^1 \right) \quad (4.2.3)$$

The entries of $\boldsymbol{\partial}_j^n$ are computed by considering the derivative of each monomial:

$$\begin{aligned} \frac{\partial^n m_\alpha^1}{\partial x_j^n} &= \frac{\partial^n}{\partial x_j^n} \prod_{i=1}^d \left(\frac{(\mathbf{x})_i - (\mathbf{x}_E)_i}{h_E} \right)^{\alpha_i} \\ &= \underbrace{\left(\frac{\alpha_j!}{(\alpha_j - n)! h_E^n} \right)}_A \underbrace{\left[\left(\frac{(\mathbf{x})_j - (\mathbf{x}_E)_j}{h_E} \right)^{\alpha_j - n} \prod_{i \neq j} \left(\frac{(\mathbf{x})_i - (\mathbf{x}_E)_i}{h_E} \right)^{\alpha_i} \right]}_B. \end{aligned} \quad (4.2.4)$$

Term A is a constant, which we shall denote $c_{\alpha,j,n}$ and term B is scaled monomial m_a^1 where a is the 1D index corresponding to the exponent vector $(\alpha_1, \dots, \alpha_j - n, \dots, \alpha_d)$. We therefore set $(\boldsymbol{\partial}_j^n)_{\alpha,a} = c_{\alpha,j,n}$. If $\partial_{x_j}^n m_\alpha^1$ is zero, then all element of row α in $\boldsymbol{\partial}_j^n$ will also be zero, if not then row α will have one non-zero element. A matrix of particular importance in the VEM is the matrix corresponding to the Laplacian $\Delta : \mathbb{P}_p(E) \rightarrow \mathbb{P}_{p-2}(E)$:

$$\Delta q_p = \sum_{j=1}^d \partial_{x_j}^2 q_p = \left(\sum_{j=1}^d \boldsymbol{\partial}_j^2 \right) \mathbf{m}_{p-2}^1 = \mathbf{L} \mathbf{m}_{p-2}^1 \quad (4.2.5)$$

So far, we have considered the case of expanding both a polynomial and its derivative in the scaled monomial basis. We shall now consider the case where we wish to represent the polynomial and its derivative in the orthogonalised basis. Letting $r_p = \mathbf{d}^T \mathbf{m}_p^2$ be an order p polynomial expanded in $\mathcal{M}_p^2(E)$ then the derivative is given by:

$$\begin{aligned} \partial_{x_j}^n r_p &= \partial_{x_j}^n \left(\mathbf{d}^T \mathbf{M}_p \mathbf{m}_p^1 \right) \\ &= \mathbf{d}^T \left(\mathbf{M}_p \boldsymbol{\partial}_j^n \mathbf{m}_{p-n}^1 \right) \\ &= \mathbf{d}^T \left(\mathbf{M}_p \boldsymbol{\partial}_j^n (\mathbf{M}_{p-n})^{-1} \mathbf{m}_{p-n}^2 \right) \\ &= \mathbf{d}^T \left(\overline{\boldsymbol{\partial}_j^n} \mathbf{m}_{p-n}^2 \right) \end{aligned}$$

The derivative matrix in the orthogonalised basis is therefore given by

$$\overline{\boldsymbol{\partial}_j^n} = \mathbf{M}_p \boldsymbol{\partial}_j^n (\mathbf{M}_{p-n})^{-1}$$

Using the orthogonalised derivative matrix we may then compute the Laplacian matrix in the orthogonalised basis as follows:

$$\overline{\mathbf{L}} = \sum_{j=1}^d \overline{\boldsymbol{\partial}_j^2} = \sum_{j=1}^d \mathbf{M}_p \boldsymbol{\partial}_j^2 (\mathbf{M}_{p-2})^{-1} = \mathbf{M}_p \left(\sum_{j=1}^d \boldsymbol{\partial}_j^2 \right) (\mathbf{M}_{p-2})^{-1} = \mathbf{M}_p \mathbf{L} (\mathbf{M}_{p-2})^{-1} \quad (4.2.6)$$

therefore $\Delta r_p = \mathbf{d}^T \left(\bar{\mathbf{L}} \mathbf{m}_{p-2}^2 \right)$.

4.3 The Local Curved Virtual Element Space

4.3.1 The Initial Space

Given a curved polygon $F \in \mathcal{T}_h$, on each edge e , straight or curved, let $\gamma_e : [0, 1] \rightarrow e$ be the NURBS parametrisation of that edge. The forward mapping is denoted $\mathbf{x} = \gamma_e t$ and the inverse mapping $t = \gamma_e^{-1} \mathbf{x}$. We define a 1D polynomial space $\mathbb{P}_p([0, 1])$ from which we may define a pseudo-polynomial space on each edge e

$$\tilde{\mathbb{P}}_p(e) := \{\tilde{q}_p(\gamma_e t) = q_p(t) \quad : \quad p_p \in \mathbb{P}_p([0, 1]) \quad \forall t \in [0, 1]\}. \quad (4.3.1)$$

Note $\tilde{\mathbb{P}}_p(e)$ will not be a polynomial unless γ_e is an affine map. Let F be bounded by the union of a set of curvilinear edges $\tilde{\mathcal{E}}_F$ and a set of straight edges $\mathcal{E}_F$ such that $\partial F = \tilde{\mathcal{E}}_F \cup \mathcal{E}_F$. Following the usual process of constructing local VEM spaces for elliptic operators we begin by defining a **boundary space** $\mathcal{B}_p(\partial F)$:

$$\mathcal{B}_p(\partial F) := \left\{ v : v \in C^0(\partial F) \quad \text{and} \quad v|_{\tilde{e}} \in \tilde{\mathbb{P}}_p(\tilde{e}) \quad \forall \tilde{e} \in \tilde{\mathcal{E}}_F \quad \text{and} \quad v|_e \in \mathbb{P}_p(e) \quad \forall e \in \mathcal{E}_F \right\}.$$

With $\mathcal{B}_p(\partial F)$ defined, the **initial local VEM space** may be constructed as follows:

$$\tilde{\mathcal{V}}_p(F) := \left\{ v \in H^1(F) : v|_{\partial F} \in \mathcal{B}_p(\partial F) \quad \text{and} \quad \Delta v \in \mathbb{P}_p(F) \right\}$$

We now prove two propositions about the dimension of $\mathcal{B}_p(\partial F)$ and $\tilde{\mathcal{V}}_p(F)$.

Proposition 4.3.1. *For a polygon F with N_e^F edges, $\mathcal{B}_p(\partial F)$ has dimension pN_e^F .*

Proposition 4.3.2. *For a polygon F with N_e^F edges, $\tilde{\mathcal{V}}_p(F)$ has dimension $pN_e^F + n_p$.*

We present the proofs in the appendix, sections C.1 and C.2.

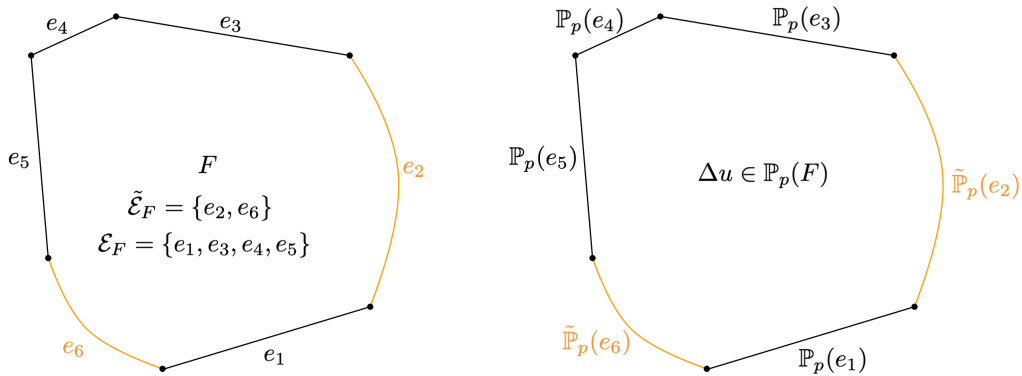


Figure 4.3: Illustration of local curvilinear VEM space (recreation of figure 6 in [1])

Figure 4.3 illustrates $\tilde{\mathcal{V}}_p(F)$ on an example curved polygon where e_2, e_6 (shown in orange) are curved and the remaining edges are straight. We may define the following linear functionals on $\tilde{\mathcal{V}}_p(F)$ [1]:

- $\mathcal{D}_1$: The value of v at each of the vertices of F .

- $\mathcal{D}_2$: The value of v at the $p - 1$ internal points of the $p + 1$ Gauss-Lobatto quadrature on $e \in \mathcal{E}_F$.
- $\mathcal{D}_3$: The value of v at each of the points given by $\mathbf{x} = \gamma_{\tilde{e}} t$ where t is a member of the $p - 1$ internal points of the $p + 1$ Gauss-Lobatto rule on $[0, 1]$ and $\gamma_{\tilde{e}} : [0, 1] \rightarrow \tilde{e}$ is the NURBS parameterisation for curve $\tilde{e} \in \tilde{\mathcal{E}}_F$.
- $\mathcal{D}_4$: The moments with the members of $\mathcal{M}_{p-2}^1(F)$:

$$\frac{1}{|F|} \int v m_{\alpha}^1 dx \quad \forall m_{\alpha}^1 \in \mathcal{M}_{p-2}^1(F).$$

Using these linear functionals, it is possible to compute three important projection operators: the energy projector $\Pi_p^{\nabla} : H^1(F) \rightarrow \mathbb{P}_p(F)$, the gradient projector $\Pi_{p-1}^0 \nabla : H^1(F) \rightarrow [\mathbb{P}_{p-1}(F)]^2$ and the $L^2(F)$ projection $\Pi_{p-2}^0 : L^2(F) \rightarrow \mathbb{P}_{p-2}(F)$.

Definition 4.3.3. (The energy projector) *The energy projector is defined by the following projection onto constants and the following orthogonality condition:*

$$\int_{\partial F} \Pi_p^{\nabla} v - v dx = 0 \quad \forall v \in H^1(F) \quad (4.3.2)$$

$$\left(\nabla \left(\Pi_p^{\nabla} v - v \right), \nabla q_p \right)_{0,F} = 0 \quad \forall v \in H^1(F) \quad \text{and} \quad \forall q_p \in \mathbb{P}_p(F). \quad (4.3.3)$$

To see how Π_p^{∇} is computed using $\mathcal{D}_1 - \mathcal{D}_4$ we shall rewrite equation (4.3.3) as:

$$\left(\nabla \Pi_p^{\nabla} v, \nabla q_p \right)_{0,F} = \left(\nabla v, \nabla q_p \right)_{0,F} \quad (4.3.4)$$

The left-hand side of equation (4.3.4) is simply the inner product of scaled monomials, and is therefore computable using Lassere's method (see section 5.3.2). The right-hand side of equation (4.3.4) is computable from the linear functionals as follows:

$$\left(\nabla v, \nabla q_p \right)_{0,E} = - \underbrace{\int_F \Delta q_p v dx}_A + \underbrace{\int_{\partial F} \nabla q_p \cdot \mathbf{n} v dx}_B.$$

Term A is computable using the internal linear functionals $\mathcal{D}_4$ as $\Delta q_p \in \mathbb{P}_{p-2}(F)$. Term B is computable using the boundary linear functionals as $v|_e \in \tilde{\mathbb{P}}_p(e)$ for all $e \subset \partial F$ and so the integral may be computed using the $p + 1$ Lobatto rule on $[0, 1]$ mapped forward to e by γ_e . We shall revisit this point in section 4.5.

Definition 4.3.4. (The gradient projector) *The gradient projector is defined by the following orthogonality condition:*

$$\left(\Pi_{p-1}^0 \nabla v - \nabla v, \mathbf{q}_{p-1} \right)_{0,F} = 0 \quad \text{for} \quad \mathbf{q}_{p-1} \in [\mathbb{P}_{p-1}(F)]^2, \quad (4.3.5)$$

Letting $\mathbf{q}_{p-1} = (q_{p-1,1}, q_{p-1,2})$, equation (4.3.5) may be written as:

$$\sum_{i=1}^2 \left(\Pi_{p-1}^0 \left(\frac{\partial v}{\partial x_i} \right) - \frac{\partial v}{\partial x_i}, q_{p-1,i} \right)_{0,F} = 0$$

Therefore, the gradient projector is actually an L^2 projector operating on the first derivatives of v .

Equation (4.3.5) may be rearranged to give:

$$\left(\Pi_{p-1}^0 \nabla v, \mathbf{q}_{p-1} \right)_{0,F} = (\nabla v, \mathbf{q}_{p-1})_{0,F} = \underbrace{- \int_F \nabla \cdot \mathbf{q}_{p-1} v \, dx}_A + \underbrace{\int_{\partial F} \mathbf{q}_{p-1} \cdot \mathbf{n} v \, ds}_B.$$

Term A is computable from the internal linear functionals $\mathcal{D}_4$ as $\nabla \cdot \mathbf{q}_{p-1} \in \mathbb{P}_{p-2}(F)$. As with the energy projector, term B is computable using the boundary linear functionals as $v|_e \in \tilde{\mathbb{P}}_p(e)$ for all $e \subset \partial F$. Therefore, the integral may be computed using the $p+1$ Lobatto rule on $[0, 1]$ mapped forward to e by γ_e .

Definition 4.3.5. (The L^2 projector) The L^2 projector is defined by the following orthogonality condition.

$$\left(\Pi_{p-2}^0 v - v, q_{p-2} \right)_{0,F} = 0 \quad \forall v \in H^1(F) \quad \text{and} \quad \forall q_{p-2} \in \mathbb{P}_{p-2}(F).$$

Rearranging gives:

$$\left(\Pi_{p-2}^0 v, q_{p-2} \right)_{0,F} = (v, q_{p-2})_{0,F}. \quad (4.3.6)$$

The left-hand side of equation (4.3.6) is simply the inner product of scaled monomials, and are therefore computable using Lassere's method. The right-hand side is computable from the internal linear functionals $\mathcal{D}_4$.

4.3.2 The Enhanced Space

The initial virtual element space $\tilde{\mathcal{V}}_p(F)$ is sufficient to solve pure diffusion problems. However, if one wishes to solve problems involving either convection or reaction terms then the full $L^2(F)$ projector is required. Following the method of [198], it is possible to construct the **enhanced virtual element space** in which the full L^2 projection onto $\mathbb{P}_p(F)$ is computable from just the linear functionals $\mathcal{D}_1 - \mathcal{D}_4$. The enhanced virtual element space is defined as follows

$$\mathcal{V}_p(F) := \left\{ v \in \tilde{\mathcal{V}}_p(F) : (v, q_p)_{0,F} = \left(\Pi_p^\nabla v, q_p \right)_{0,F} \quad \forall q_p \in \bar{\mathbb{P}}_p(F) \cup \bar{\mathbb{P}}_{p-1}(F) \right\},$$

where $\bar{\mathbb{P}}_p(F) = \mathbb{P}_p(F)/\mathbb{P}_{p-1}(F)$ is the set of polynomials of order strictly equal to p . Therefore, $\mathcal{V}_p(F) \subset \tilde{\mathcal{V}}_p(F)$, and is the set comprised of all functions $v \in \tilde{\mathcal{V}}_p(F)$ for which the internal moments of v and $\Pi_p^\nabla v$ coincide for orders $p-1$ and p . The linear functionals $\mathcal{D}_1, \dots, \mathcal{D}_4$ are

well defined on $\mathcal{V}_p(F)$ and may be taken as a set of unisolvent degrees of freedom, meaning that every $v \in \mathcal{V}_p(F)$ is uniquely determined by the values of $\mathcal{D}_1 - \mathcal{D}_4$.

Proposition 4.3.6. *For a polygon F with N_e^F edges the dimension of $\mathcal{V}_p(F)$ is $pN_e^F + n_{p-2}$ and the linear functionals $\mathcal{D}_1, \dots, \mathcal{D}_4$ may be taken as a unisolvent set of degrees of freedom.*

We present the proof in section C.3. With unisolvence established, the linear functionals $\mathcal{D}_1, \dots, \mathcal{D}_4$ shall now be referred to as **degrees of freedom** (DoFs) and $\dim(\mathcal{V}_p(F))$ shall be denoted N_{dof}^F . The i 'th degree of freedom shall be denoted $\text{dof}_i(v) : \mathcal{V}_p(F) \rightarrow \mathbb{R}$ and the local Lagrangian basis $\varphi_i \in \mathcal{V}_p(F)$ satisfies:

$$\text{dof}_i(\varphi_j) = \delta_{i,j} \quad \text{for } i, j = 1, \dots, N_{\text{dof}}^F.$$

Every $v \in \mathcal{V}_p(F)$ has a unique representation in the local Lagrangian basis:

$$v = \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(v) \varphi_i.$$

Remark 3. *Due to the fact that boundary spaces $\tilde{\mathbb{P}}_p(e)$ are not polynomial unless $\gamma_e : [0, 1] \rightarrow e$ is an affine map, then in general the set of polynomials $\mathbb{P}_p(F)$ are not contained in $\mathcal{V}_p(F)$ (i.e. $\mathbb{P}_p(F) \not\subset \mathcal{V}_p(F)$). However we still have $\mathbb{P}_0(F) \subset \mathcal{V}_p(F)$ [1].*

The choice of local DoF numbering, and their mapping to $\mathcal{D}_1, \dots, \mathcal{D}_4$ is arbitrary so long as it is consistent. Here we opt to number the vertex DoFs counter-clockwise first ($\mathcal{D}_1$), then edge DoFs counter-clockwise ($\mathcal{D}_2$ and $\mathcal{D}_3$ irrespective of whether edge e is curvilinear) and finally internal DoFs in order of increasing monomial ID. An example is presented in figure 4.4 for VEM order $p = 3$. We also use $\mathcal{D}^e$ to denote the set of DoFs (two vertex DoFs and the edge DoFs) that reside on edge e . In $\mathcal{V}_p(F)$ the full $L^2(F)$ projection $\Pi_p^0 : \mathcal{V}_p(F) \rightarrow \mathbb{P}_p(F)$ is computable using just the degrees of freedom as follows:

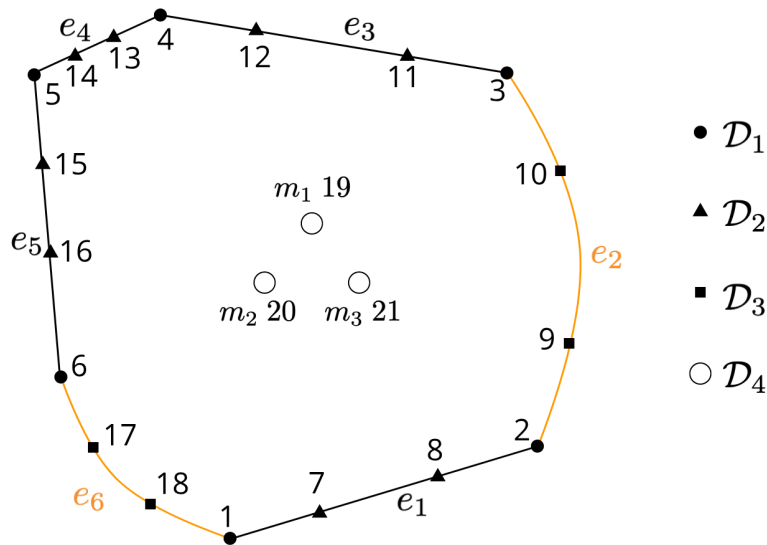


Figure 4.4: DoF distribution and numbering for $p = 3$.

$$\left(\Pi_p^0 v, q_p\right)_{0,F} = \begin{cases} (v, q_p)_{0,F} & \text{for } q_p \in \mathbb{P}_{p-2}(F) \\ \left(\Pi_p^\nabla v, q_p\right)_{0,F} & \text{for } q_p \in \bar{\mathbb{P}}_{p-1}(F) \cup \bar{\mathbb{P}}_p(F) \end{cases} \quad (4.3.7)$$

Additionally, in $\mathcal{V}_p(F)$ the full gradient projection $\Pi_p^0 \nabla : \mathcal{V}_p(F) \rightarrow [\mathbb{P}_p(F)]^2$ is also computable using just the degrees of freedom. Letting $\mathbf{q}_p = (q_{p,1}, q_{p,2}) \in [\mathbb{P}_p(F)]^2$ the orthogonality condition becomes:

$$\left(\Pi_p^0 \nabla u - \nabla u, \mathbf{q}_p\right)_{0,F} = \sum_{i=1}^2 \left(\Pi_p^0 \frac{\partial u}{\partial x_i} - \frac{\partial u}{\partial x_i}, q_{p,i}\right)_{0,F} = 0.$$

Therefore, for each coordinate component we have:

$$\left(\Pi_p^0 \frac{\partial u}{\partial x_i}, q_{p,i}\right)_{0,F} = \left(\frac{\partial u}{\partial x_i}, q_{p,i}\right)_{0,F} = - \left(u, \frac{\partial q_{p,i}}{\partial x_i}\right)_{0,F} + (\mathbf{n}_i u, q_{p,i})_{0,\partial F}.$$

The integral over the boundary ∂F is computable from the boundary linear functionals $\mathcal{D}_1 - \mathcal{D}_3$. The integral over the element F must employ the same trick we used for the L^2 projection:

$$\left(u, \frac{\partial q_{p,i}}{\partial x_i}\right)_{0,F} = \begin{cases} \left(u, \frac{\partial q_{p,i}}{\partial x_i}\right)_{0,F}, & \text{if } \frac{\partial q_{p,i}}{\partial x_i} \in \mathbb{P}_{p-2}(F) \\ \left(\Pi_p^\nabla u, \frac{\partial q_{p,i}}{\partial x_i}\right)_{0,F}, & \text{if } \frac{\partial q_{p,i}}{\partial x_i} \in \bar{\mathbb{P}}_{p-1}(F) \end{cases}$$

recalling that $\bar{\mathbb{P}}_{p-1}(F) = \mathbb{P}_{p-1}(F)/\mathbb{P}_{p-2}(F)$. The top case corresponds to the internal degrees of freedom $\mathcal{D}_4$, the bottom case may be computed by expanding $\Pi_p^\nabla u$ out in the polynomial basis and computing inner products of polynomials.

We shall postpone a practical guide to computing all of the projections, and using them to construct local bilinear forms, to section 4.5. First, we shall review the local approximation properties of the space $\mathcal{V}_p(F)$ in next section.

4.4 Approximation Properties of the Local Space

With the local curvilinear VEM spaces defined we shall now recall some interpolation estimates defined on star-shaped, curvilinear polygons.

Lemma 4.4.1. (*Local Polynomial Estimate*) *Let $F \in \mathcal{T}_h$ and let m, s be two integers such that $0 \leq m \leq s$ and $1 \leq s \leq p+1$. Then for all $v \in H^s(F)$ there exists a $v_\pi \in \mathbb{P}_p(F)$ such that:*

$$|v - v_\pi|_{m,F} \leq Ch_F^{s-m} |v|_{s,F}$$

where $C = C(p, \eta)$ and η is the mesh regularity constant (cf. assumption 4.2.1).

This is a well-known result so the proof will be omitted here, however it may be found in many canonical texts on the finite element method (FEM), see for example [141] chapter 4, [139] chapter 1.4 or [80] chapter 3.1.

Corollary 4.4.1.1. *For $v \in H^s(F)$ with $1 \leq s \leq p+1$, the L^2 projection on F satisfies:*

$$\|v - \Pi_p^0 v\|_{0,F} \leq Ch_F^s |v|_{s,F}$$

Additionally, for $u \in H^s(F)$ we have the following estimate:

$$\|\nabla u - \Pi_p^0 \nabla u\|_{0,F} \leq Ch_F^{s-1} |\nabla u|_{s-1,F}$$

The proof for both these estimates is presented in the appendix, section C.5.

Remark 4. *The result of lemma 4.4.1 is often presented in the VEM literature as:*

$$\|v - v_\pi\|_{0,F} + h_F |v - v_\pi|_{1,F} \leq Ch_F^s |v|_{s,F}$$

Next, we present the interpolation estimate in $\mathcal{V}_p(F)$

Lemma 4.4.2. (*Local Interpolation estimate*) *Let $F \in \mathcal{T}_h$ and let m, s be two integers such that $0 \leq m \leq s$ and $2 \leq s \leq p+1$. Then for all $v \in H^s(F)$ there exists a $v_I \in \mathcal{V}_p(F)$ which interpolates v*

$$\text{dof}_i(v - v_I) = 0 \quad \text{for } i = 1, \dots, N_{\text{dof}}^F$$

for which the following bound holds:

$$|v - v_I|_{m,F} \leq Ch_F^{s-m} |v|_{s,F},$$

where $C = C(p, \eta)$ and η is the mesh regularity constant (cf. assumption 4.2.1).

The proof of this lemma may be found in [213].

Remark 5. *As with the local polynomial estimate lemma 4.4.1, the result of lemma 4.4.2 is*

also often presented in the VEM literature as:

$$\|v - v_I\|_{0,F} + h_F |v - v_I|_{1,F} \leq Ch_F^s |v|_{s,F}.$$

We also present the following result concerning the scaling properties of the Lagrange basis functions $\{\varphi_i\}_{i=1}^{N_{\text{dof}}^F}$ (first presented and proven in [202], lemma 4.9).

Lemma 4.4.3. (*Basis scaling*): Let $F \in \mathcal{T}_h$ and let $\{\varphi_i\}_{i=1}^{N_{\text{dof}}^F}$ be the Lagrangian basis of $\mathcal{V}_p(F)$. For $r \in \{0, 1\}$ the following scaling holds:

$$C^{-1}h_F^{1-r} \leq |\varphi_i|_{r,E} \leq Ch_F^{1-r},$$

where $C = C(p, \eta)$.

The scaling properties are required when deriving stabilising bilinear forms for the VEM discretisation of diffusion and reaction terms. This point shall be explained further in section 6.3. Finally, we present some useful bounds for the VE L^2 projection operators Π_p^0 and $\Pi_{p-1}^0 \nabla$.

Lemma 4.4.4. (*Projection bounds*)

$$\begin{aligned} \forall v \in L^2(V), \quad & \left\| \Pi_p^0 v \right\|_{0,V} \leq \|v\|_{0,V} \\ \forall v \in H^1(V), \quad & \left\| \Pi_{p-1}^0 \nabla v \right\|_{0,V} \leq \|\nabla v\|_{0,V} \end{aligned}$$

We present the proof in section C.6

4.5 Practical Implementation

In this section we shall review the guides presented in [197, 230] which explain how the projectors are computed in practice. We shall first present the case where $\mathcal{M}_p^1(F)$ is used as the basis of $\mathbb{P}_p(F)$. Later, we will present a method to substitute the scaled monomial basis $\mathcal{M}_p^1(F)$ with the orthogonalised basis $\mathcal{M}_p^2(F)$. We shall make use of Matlab-style notation for sub-matrices, for example $\mathbf{A}(1 : 5, 1 : 5)$ will denote the submatrix consisting of the first 5 rows and columns of matrix $\mathbf{A}$. We shall assume that all scaled monomials up to order $2p$ have already been computed using Lassere's method and stored in array I_{sm} such that:

$$I_{sm}[\alpha] = \int_F m_\alpha^1(\mathbf{x}) \, dx.$$

4.5.1 Boundary Shape Functions

In the conventional straight-sided VEM, an explicit representation of the shape functions restricted to the edge is not required. Any integrals on the edges of a polygon are computed using the $p + 1$ point Gauss-Lobatto rule on each edge. These quadrature points correspond to the locations of the degrees of freedom. Therefore, the values of the shape functions at the boundary quadrature points are either 1 or 0 by definition. In the curvilinear VEM however, where the boundary edges are NURBS curves, simply using the $p + 1$ Gauss-Lobatto rule on $[0, 1]$ is not guaranteed to be sufficiently accurate to ensure optimal convergence rates. To understand why, consider the curve integral of function $f(\mathbf{x})$ on edge e with NURBS parametrisation γ_e :

$$\int_e f(\mathbf{x}) \, \varphi_i(\mathbf{x}) \, ds = \int_0^1 f(\gamma_e t) \varphi_i(\gamma_e t) \, J_e(t) \, dt, \quad \text{where} \quad J_e = \left\| \frac{d\gamma_e t}{dt} \right\| \quad (4.5.1)$$

If the NURBS parametrisation is C^m continuous then the Jacobian term, $J_e(t)$, is C^{m-1} continuous, so if a NURBS curve of order p has any knots which are repeated $p - 1$ times on the interior of $[0, 1]$ then the Jacobian will only be C^0 continuous at that knot. Any integration rule which is defined on $[0, 1]$ will therefore incur a significant numerical error in the integral. In equation (4.5.2) we provide an example of this situation, we plot both the curve and its Jacobian in figure 4.5.

$$\begin{aligned} p &= 2, \quad \mathbf{u} = [0, 0, 0, 0.2, 0.8, 1, 1, 1], \\ P &= \{(0, 0), (1, -1), (2, 1), (2, -3), (1, -2)\}. \end{aligned} \quad (4.5.2)$$

The meaning of the data in equation (4.5.2) is explained in section 5.2. The curve Jacobian in this example is only C^0 continuous at knots $u = 0.2, 0.8$, so if we were to simply use the $p + 1$ Gauss-Lobatto rule on $[0, 1]$ there would be a significant error in the integral. The integral over $[0, 1]$ must therefore be split into the sum of integrals over each knot span. Each knot-span integral may then be computed using a Gauss-Legendre rule defined on $[u_i, u_{i+1}]$, we use

a Gauss-Legendre rule with a sufficient number of points to integrate a polynomial of order $2p - 1$ on each knot span.

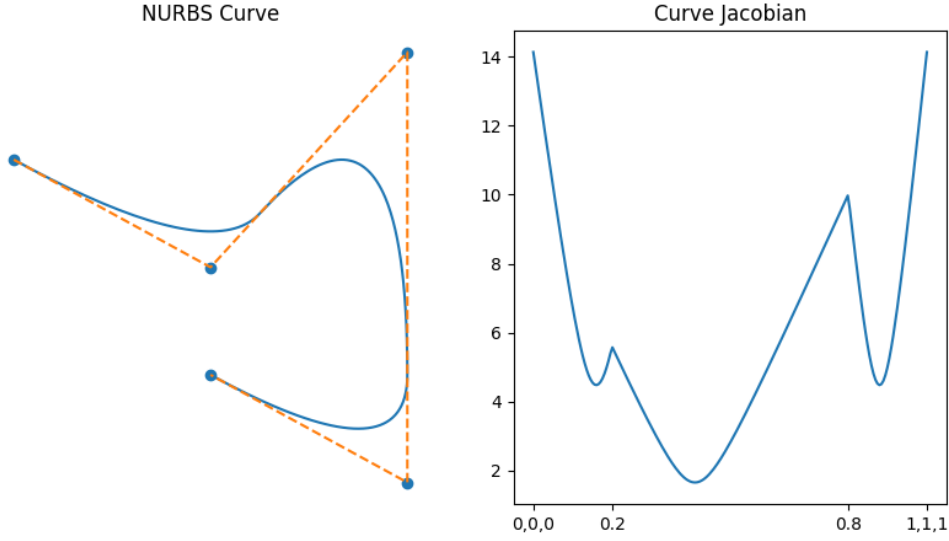


Figure 4.5: An example B-spline curve with a C^0 -continuous Jacobian.

Because we are unable to simply use the boundary DoF values of shape functions when evaluating edge integrals, we must evaluate the shape functions $\varphi_i|_e$ at points other than the boundary DoF locations. Consequently, an explicit representation of the shape functions restricted to the boundary is required for the curvilinear VEM.

With the motivation for the explicit representation explained we shall now explain how it is computed. In section 4.3, for polygon F , we treated the polynomial space on straight edges separately to curvilinear edges. While this is a useful distinction to make when developing the theory of the local VEM space, in practice there is no need to make the distinction explicitly. In our implementation the straight edge case is simply treated as NURBS curve with form:

$$p = 1, \quad \mathbf{u} = [0, 0, 1, 1], \quad \mathbf{w} = [1, 1], \quad P = \{\mathbf{v}_1, \mathbf{v}_2\}.$$

Therefore, regardless of whether e is straight or not, we first construct the 1D polynomial space on the parameter range, $\mathbb{P}_p([0, 1])$ and then map forward using the edge parametrisation γ_e to build the pseudo-polynomial space on edge e (the space $\tilde{\mathbb{P}}_p(e)$ given in equation (4.3.1)). When e is straight, we have $\tilde{\mathbb{P}}_p(e) = \mathbb{P}_p(e)$. To construct the shape functions restricted to edge e , (denoted $\varphi_i|_e$) the first task is to compute the Lagrangian shape functions $\hat{\varphi}_i \in \mathbb{P}_p([0, 1])$ which have the $p + 1$ Gauss-Lobatto points in $[0, 1]$ as interpolation points. The 1D Lagrangian shape functions $\hat{\varphi}_i$ have the following representation in the monomials, letting $\mu_\alpha = t^{\alpha-1}$:

$$\hat{\varphi}_i(t) = \sum_{\alpha=1}^{p+1} (\mathbf{\Pi})_{\alpha,i} \mu_\alpha(t), \quad i = 1, \dots, p + 1. \quad (4.5.3)$$

Let $\{\eta_i\}_{i=1}^{p+1}$ be the Gauss-Lobatto points on $[0, 1]$, then by the Lagrangian property of the basis

we have $\hat{\varphi}_i(\eta_j) = \delta_{i,j}$, which yields the following system of equations:

$$\sum_{\alpha=1}^{p+1} (\mathbf{\Pi})_{\alpha,i} \mu_{\alpha}(\eta_j) = \delta_{i,j}$$

which may be written as $\mathbf{V}\mathbf{\Pi} = \mathbf{I}$ where $\mathbf{V}$ is the Vandermonde matrix $(\mathbf{V})_{i,\alpha} = \mu_{\alpha}(\eta_i) = \eta_i^{\alpha-1}$. The 1D shape functions on $[0, 1]$ may therefore be treated as the "parent element" (also known as the "reference element"), to borrow a concept from FEA. The shape functions on each edge $\varphi_i(\mathbf{x})$ are then computed by mapping forward from the parent element using the NURBS parametrisation γ_e . Therefore we have: $\varphi_i(\gamma_e t) = \hat{\varphi}_i(t)$. In figure 4.6 we illustrate this mapping process from a parent element to a curved edge e_1 and a straight edge e_2 .

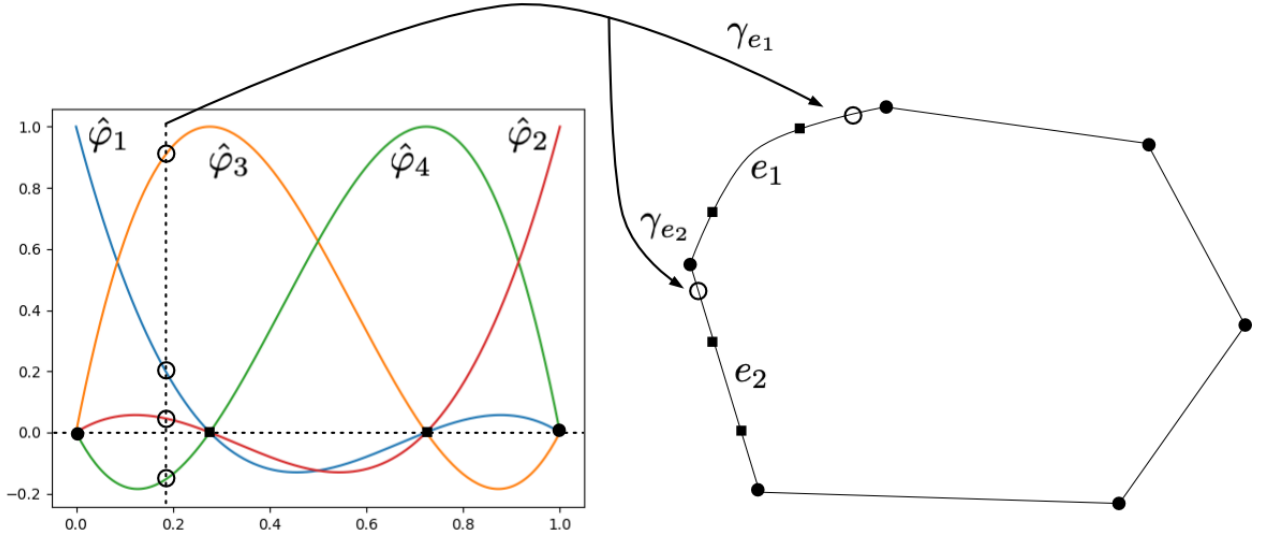


Figure 4.6: Illustration of mapping from $\mathbb{P}_p([0, 1])$ to $\tilde{\mathbb{P}}_p(e_1)$ and $\tilde{\mathbb{P}}_p(e_2)$ for $p = 3$.

Remark 6. The parent element defined on $[0, 1]$ with shape functions $\hat{\varphi}_i$, is in fact a 1D nodal spectral element (see [235], chapter 3), where the nodal basis is formed based on the Gauss-Lobatto points.

Returning to the problem of evaluating (4.5.1), we may write this as:

$$\begin{aligned} \int_e f(\mathbf{x}) \varphi_i(\mathbf{x}) ds &= \int_0^1 f(\gamma_e t) \hat{\varphi}_i(t) J_e(t) dt = \sum_{i=1}^{m-1} \int_{u_i}^{u_{i+1}} f(\gamma_e t) \hat{\varphi}_i(t) J_e(t) dt \\ &= \sum_{i=1}^{m-1} \sum_{j=1}^{\text{nqp}} f(\gamma_e \eta_j) \hat{\varphi}_i(\eta_j) J_e(\eta_j) w_j \end{aligned}$$

where m is the number of knots in the curve and (η_i, w_i) are points/weights of a Gauss-Legendre rule on $[u_i, u_{i+1}]$.

Remark 7. In the original paper on the curvilinear VEM, [1], the authors simply use the $p+1$ Gauss-Lobatto points on each edge to perform integration. However, the numerical tests they present have boundary curves which are infinitely differentiable. They therefore will not observe

any degradation in convergence. However, for the reasons we have presented in this section, this approach is will not guarantee sufficient accuracy when using NURBS.

4.5.2 Energy Projector

We will now present a practical guide to compute the energy projector $\Pi_p^\nabla : [\mathcal{V}_p(F) + \mathbb{P}_p(F)] \rightarrow \mathbb{P}_p(F)$ on curvilinear VEM spaces. We shall denote the corresponding $n_p \times N_{\text{dof}}^F$ matrix as $\mathbf{\Pi}_p^{\nabla,*}$. For shape function φ_i we have:

$$\Pi_p^\nabla \varphi_i = \sum_{\alpha=1}^{n_p} (\mathbf{\Pi}_p^{\nabla,*})_{\alpha,i} m_\alpha^1 \quad (4.5.4)$$

To compute $\mathbf{\Pi}_p^{\nabla,*}$ we begin with the orthogonality conditions of equations (4.3.2) and (4.3.3):

$$\begin{aligned} \int_{\partial F} \Pi_p^\nabla \varphi_i - \varphi_i \, dx &= 0, \quad i = 1, \dots, N_{\text{dof}}^F \\ (\nabla (\Pi_p^\nabla \varphi_i - \varphi_i), \nabla m_\beta^1)_{0,F} &= 0, \quad i = 1, \dots, N_{\text{dof}}^F \quad \beta = 2, \dots, n_p \end{aligned}$$

substituting the expansion of equation (4.5.4) gives:

$$\begin{aligned} \sum_{\alpha=1}^{n_p} \int_{\partial F} (\mathbf{\Pi}_p^\nabla)_{\alpha,i} m_\alpha^1 \, ds &= \int_{\partial F} \varphi_i \, ds \\ \sum_{\alpha=1}^{n_p} (\mathbf{\Pi}_p^\nabla)_{\alpha,i} (\nabla m_\alpha^1, \nabla m_\beta^1)_{0,F} &= (\nabla \varphi_i, \nabla m_\beta^1)_{0,F} \end{aligned}$$

which yields the system of equations $\mathbf{G} \mathbf{\Pi}_p^{\nabla,*} = \mathbf{B}$ where:

$$\begin{aligned} \mathbf{G} &= \begin{bmatrix} \int_{\partial F} m_1^1 \, ds & \int_{\partial F} m_2^1 \, ds & \dots & \int_{\partial F} m_{n_p}^1 \, ds \\ 0 & (\nabla m_2^1, \nabla m_2^1)_{0,F} & \dots & (\nabla m_2^1, \nabla m_{n_p}^1)_{0,F} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & (\nabla m_{n_p}^1, \nabla m_2^1)_{0,F} & \dots & (\nabla m_{n_p}^1, \nabla m_{n_p}^1)_{0,F} \end{bmatrix}, \\ \mathbf{B} &= \begin{bmatrix} \int_{\partial F} \varphi_1 \, ds & \int_{\partial F} \varphi_2 \, ds & \dots & \int_{\partial F} \varphi_{N_{\text{dof}}^F} \, ds \\ (\nabla m_2^1, \nabla \varphi_1)_{0,F} & (\nabla m_2^1, \nabla \varphi_2)_{0,F} & \dots & (\nabla m_2^1, \nabla \varphi_{N_{\text{dof}}^F})_{0,F} \\ \vdots & \vdots & \ddots & \vdots \\ (\nabla m_{n_p}^1, \nabla \varphi_1)_{0,F} & (\nabla m_{n_p}^1, \nabla \varphi_2)_{0,F} & \dots & (\nabla m_{n_p}^1, \nabla \varphi_{N_{\text{dof}}^F})_{0,F} \end{bmatrix}. \end{aligned}$$

Matrix $\mathbf{G}$ is computed in two stages. The first stage is to compute the first row by direct quadrature on the boundary of F :

$$(\mathbf{G})_{1,\alpha} = \int_{\partial F} m_\alpha^1(\mathbf{x}) \, ds = \sum_{e \in \partial F} \int_0^1 m_\alpha^1(\gamma_e t) J_e(t) dt = \sum_{e \in \partial F} \sum_{i=1}^{\text{nqp}} m_\alpha^1(\gamma_e t_i) w_i J_e(t_i)$$

In the second stage we compute the remaining rows. Recalling the properties of scaled monomial derivatives from equation (4.2.4) there exist constants $c_1, c_2, d_1, d_2 \in \mathbb{R}$ and $m_{a_1}^1, m_{a_2}^1, m_{b_1}^1, m_{b_2}^1 \in \mathcal{M}_{p-1}^1(F)$ such that:

$$\nabla m_\alpha^1 = c_1 m_{a_1}^1 \mathbf{e}_1 + c_2 m_{a_2}^1 \mathbf{e}_2, \quad \nabla m_\beta^1 = d_1 m_{b_1}^1 \mathbf{e}_1 + d_2 m_{b_2}^1 \mathbf{e}_2.$$

Therefore:

$$(\mathbf{G})_{\alpha,\beta} = (\nabla m_\alpha^1, \nabla m_\beta^1)_{0,F} = c_1 d_1 (m_{a_1}^1, m_{b_1}^1)_{0,F} + c_2 d_2 (m_{a_2}^1, m_{b_2}^1)_{0,F}$$

The terms $(m_{a_1}^1, m_{b_1}^1)_{0,F}$ and $(m_{a_2}^1, m_{b_2}^1)_{0,F}$ each correspond to an entry in I_{sm} where the integral has been pre-computed. By avoiding a full quadrature for each element of $\mathbf{G}$ we save a great deal of computation.

The terms $(\mathbf{B})_{\alpha,i} = (\nabla m_\alpha^1, \nabla \varphi_i)_{0,F}$ are computed by applying integration by parts and Gauss' theorem to give:

$$(\mathbf{B})_{\alpha,i} = (\nabla m_\alpha^1, \nabla \varphi_i)_{0,F} = -(\Delta m_\alpha^1, \varphi_i)_{0,F} + \int_{\partial F} \nabla m_\alpha^1 \cdot \mathbf{n} \varphi_i ds$$

The first term is only non-zero when φ_i corresponds to an internal DoF (i.e. when $i > p N_v$). To compute the first term we expand Δm_α^1 in $\mathcal{M}_{p-2}^1(F)$ using the Laplacian matrix derived in equation (4.2.5):

$$\begin{aligned} (\Delta m_\alpha^1, \varphi_i)_{0,F} &= \left(\sum_{\beta=1}^{n_{p-2}} (\mathbf{L})_{\alpha,\beta} m_\beta^1, \varphi_i \right)_{0,F} = \sum_{\beta=1}^{n_{p-2}} (\mathbf{L})_{\alpha,\beta} (m_\beta^1, \varphi_i)_{0,F} \\ &= \sum_{\beta=1}^{n_{p-2}} (\mathbf{L})_{\alpha,\beta} |F| \text{dof}_{pN_v+\beta}(\varphi_i) = \sum_{\beta=1}^{n_{p-2}} |F| (\mathbf{L})_{\alpha,\beta} \delta_{pN_v+\beta,i} \end{aligned}$$

where δ is the kronecker-delta. The second term is only non-zero when φ_i corresponds to a boundary DoF (i.e. when $i \leq p N_v$) and is computed as follows:

$$\begin{aligned} \int_{\partial F} \nabla m_\alpha^1 \cdot \mathbf{n} \varphi_i ds &= \sum_{e \subset \partial F} \int_e \nabla m_\alpha^1(\mathbf{x}) \cdot \mathbf{n}(t) \varphi_i(\mathbf{x}) ds \\ &= \sum_{e \subset \partial F} \int_0^1 \nabla m_\alpha^1(\gamma_e t) \cdot \mathbf{n}(t) \hat{\varphi}_i(t) J_e(t) dt \end{aligned}$$

where we have used the identity $\varphi_i(\gamma_e t) = \hat{\varphi}_i(t)$. The integrals on each edge are then computed using the Gauss-Legendre quadrature on each knot span, as was explained in section 4.5.1.

Once $\mathbf{G}$ and $\mathbf{B}$ have been computed, the projection matrix may then be computed by any standard solution procedure for small, dense linear systems: $\mathbf{\Pi}_p^{\nabla,*} = \mathbf{G}^{-1} \mathbf{B}$. In this thesis we use Lapack (or a high performance implementation of it) to solve these linear systems. Finally, we may also compute the equivalent representation of $\mathbf{\Pi}_p^{\nabla} \varphi_i$ in the Lagrangian basis, which may be thought of as the mapping $[\mathcal{V}_p(F) + \mathbb{P}_p(F)] \rightarrow \mathbb{P}_p(F) \rightarrow \mathcal{V}_p(F)$. We represent this mapping

using the $N_{\text{dof}}^F \times N_{\text{dof}}^F$ matrix $\mathbf{\Pi}_p^\nabla$. We begin by expanding $\mathbf{\Pi}_p^\nabla \varphi_i$ in the VEM basis:

$$\mathbf{\Pi}_p^\nabla \varphi_i = \sum_{j=1}^{N_{\text{dof}}^F} (\mathbf{\Pi}_p^\nabla)_{j,i} \varphi_j$$

from the Lagrangian property of the basis we therefore have:

$$(\mathbf{\Pi}_p^\nabla)_{j,i} = \text{dof}_j (\mathbf{\Pi}_p^\nabla \varphi_i) = \text{dof}_j \left(\sum_{\alpha=1}^{n_p} (\mathbf{\Pi}_p^{\nabla,*})_{\alpha,i} m_\alpha^1 \right) = \sum_{\alpha=1}^{n_p} \text{dof}_j (m_\alpha^1) (\mathbf{\Pi}_p^{\nabla,*})_{\alpha,i}$$

Defining the $N_{\text{dof}}^F \times n_p$ matrix $\mathbf{D}$ as:

$$\mathbf{D} = \begin{bmatrix} \text{dof}_1(m_1^1) & \text{dof}_1(m_2^1) & \dots & \text{dof}_1(m_{n_p}^1) \\ \text{dof}_2(m_1^1) & \text{dof}_2(m_2^1) & \dots & \text{dof}_2(m_{n_p}^1) \\ \vdots & \vdots & \ddots & \vdots \\ \text{dof}_{N_{\text{dof}}^F}(m_1^1) & \text{dof}_{N_{\text{dof}}^F}(m_2^1) & \dots & \text{dof}_{N_{\text{dof}}^F}(m_{n_p}^1) \end{bmatrix}$$

we may write express the projection matrix as $\mathbf{\Pi}_p^\nabla = \mathbf{D} \mathbf{\Pi}_p^{\nabla,*}$.

4.5.3 Gradient Projector

Next we consider the gradient projector $\mathbf{\Pi}_p^0 \nabla : [\mathcal{V}_p(F) + \mathbb{P}_p(F)] \rightarrow [\mathbb{P}_p(F)]^2$. Letting $\mathbf{e}_1 = [1, 0]$ and $\mathbf{e}_2 = [0, 1]$ denote the Cartesian basis vectors in $\mathbb{R}^2$, then the projection $\mathbf{\Pi}_p^0 \nabla \varphi_i$ may be expressed as:

$$\mathbf{\Pi}_p^0 \nabla \varphi_i = (\mathbf{\Pi}_p^0 \partial_{x_1} \varphi_i) \mathbf{e}_1 + (\mathbf{\Pi}_p^0 \partial_{x_2} \varphi_i) \mathbf{e}_2.$$

where we have used the notation $\partial_{x_i} = \partial/\partial x_i$. Each component of the projection, $\mathbf{\Pi}_p^0 \partial_{x_j} \varphi_i$, has the following expansion in $\mathcal{M}_p^1(F)$:

$$\mathbf{\Pi}_p^0 \partial_{x_j} \varphi_i = \sum_{\alpha=1}^{n_p} (\mathbf{\Pi}_p^{0,j})_{\alpha,i} m_\alpha^1 \quad (4.5.5)$$

To determine the $n_p \times N_{\text{dof}}^F$ projection matrix $\mathbf{\Pi}_p^{0,j}$ we use the orthogonality condition of equation (4.3.5) one coordinate component at a time:

$$(\mathbf{\Pi}_p^0 \partial_{x_j} \varphi_i, m_\beta^1)_{0,F} = (\partial_{x_j} \varphi_i, m_\beta^1)_{0,F}, \quad \beta = 1, \dots, n_p \quad (4.5.6)$$

Substituting the expansion of equation (4.5.5) into equation (4.5.6) we get:

$$\sum_{\beta=1}^{n_p} (\mathbf{\Pi}_p^{0,j})_{\alpha,i} (m_\beta^1, m_\alpha^1)_{0,F} = (\partial_{x_j} \varphi_i, m_\beta^1)_{0,F}.$$

which yields the following system of equations $\mathbf{H}_p \mathbf{\Pi}_p^{0,j} = \mathbf{R}^j$ where

$$(\mathbf{H}_p)_{\alpha,\beta} = (m_\alpha^1, m_\beta^1)_{0,F}, \quad \alpha, \beta = 1, \dots, n_p$$

$$(\mathbf{R}^j)_{\alpha,i} = (m_\alpha^1, \partial_{x_j} \varphi_i)_{0,F} = -(\partial_{x_j} m_\alpha^1, \varphi_i)_{0,F} + \int_{\partial F} m_\alpha^1 n_j \varphi_i ds$$

where n_j is the j 'th component of the outward normal to ∂F . The inner products $(m_\alpha^1, m_\beta^1)_{0,F}$ have been pre-computed and stored in I_{sm} , making them very fast to compute. The matrix $\mathbf{R}_j$ is the sum of two matrices:

$$\mathbf{R}_j = \mathbf{R}_j^F + \mathbf{R}_j^{\partial F}$$

where

$$(\mathbf{R}_j^F)_{\alpha,i} = \begin{cases} -(\partial_{x_j} m_\alpha^1, \varphi_i)_{0,F}, & \forall \alpha = 1, \dots, n_{p-1}, \quad (\text{Case 1}) \\ -(\partial_{x_j} m_\alpha^1, \Pi_p^\nabla \varphi_i)_{0,F}, & \forall \alpha = n_{p-1} + 1, \dots, n_p, \quad (\text{Case 2}) \end{cases}. \quad (4.5.7)$$

Case 1 corresponds to the set of internal DoFs ($\mathcal{D}_4$) and may therefore be computed as follows: recalling that the derivative $\partial_{x_j} m_\alpha^1$ may be expanded out in $\mathcal{M}_{p-2}(F)$ (equation (4.2.3)):

$$\partial_{x_j} m_\alpha^1 = \sum_{\beta=1}^{n_{p-2}} (\boldsymbol{\partial}_j^1)_{\alpha,\beta} m_\beta^1$$

we may write case 1 as:

$$-(\partial_{x_j} m_\alpha^1, \varphi_i)_{0,F} = -\sum_{\beta=1}^{n_{p-2}} (\boldsymbol{\partial}_j^1)_{\alpha,\beta} (m_\beta^1, \varphi_i)_{0,F} = -|F| \sum_{\beta=1}^{n_{p-2}} (\boldsymbol{\partial}_j^1)_{\alpha,\beta} \delta_{\beta,(i-pN_v)}$$

Case 2 corresponds to the situation where $m_\alpha^1 \in \mathcal{M}_p(F)/\mathcal{M}_{p-1}(F)$, therefore $\partial_{x_j} m_\alpha^1 \in \mathcal{M}_{p-1}(F)$ and we cannot use the internal DoFs. Instead, we use the defining property of $\mathcal{V}_p(F)$, that moments of order p and $p-1$ of φ_i and $\Pi_p^\nabla \varphi_i$ coincide. Therefore we may write case 2 as:

$$-(\partial_{x_j} m_\alpha^1, \Pi_p^\nabla \varphi_i)_{0,F} = -\sum_{\beta=1}^{n_{p-1}} \sum_{\gamma=1}^{n_p} (\boldsymbol{\partial}_j^n)_{\alpha,\beta} (m_\beta^1, m_\gamma^1)_{0,F} (\Pi_p^\nabla)_{\gamma,i}.$$

Therefore:

$$(\mathbf{R}_j^F)_{\alpha,i} = \begin{cases} -|F| \sum_{\beta=1}^{n_{p-2}} (\boldsymbol{\partial}_j^1)_{\alpha,\beta} \delta_{\beta,(i-pN_v)}, & \forall \alpha = 1, \dots, n_{p-1}, \\ (\boldsymbol{\partial}_j^1 \mathbf{H}_{p-1,p} \Pi_p^\nabla)_{\alpha,i}, & \forall \alpha = n_{p-1} + 1, \dots, n_p, \end{cases}.$$

where $(\mathbf{H}_{p-1,p})_{\alpha,\beta} = (m_\alpha^1, m_\beta^1)_{0,F}$ for $\alpha = 1, \dots, n_{p-1}$ and $\beta = 1, \dots, n_p$. The boundary matrix:

$$(\mathbf{R}_j^{\partial F})_{\alpha,i} = \int_{\partial F} m_\alpha^1 n_j \varphi_i ds$$

is only non-zero when φ_i is a boundary DoF. It is computed as follows:

$$\int_{\partial F} m_\alpha^1 n_j \varphi_i ds = \sum_{e \subset \partial F} \int_e m_\alpha^1(\mathbf{x}) n_j \varphi_i(\mathbf{x}) ds = \sum_{e \subset \partial F} \int_0^1 m_\alpha^1(\gamma_e t) n_j(t) \hat{\varphi}_i(t) J_e(t) dt.$$

Where the integrals over the parameter space $[0, 1]$ are evaluated using the numerical quadrature on curves discussed in section 4.5.1. Once $\mathbf{R}_j = \mathbf{R}_j^F + \mathbf{R}_j^{\partial F}$ has been evaluated, the projection is then computed by solving the system of equations: $\mathbf{\Pi}_p^{0,j} = (\mathbf{H}_p)^{-1} \mathbf{R}_j$.

In some situations, such as when discretising the diffusion bilinear form $(D\nabla u, \nabla v)_{0,F}$, the full gradient projection onto $[\mathbb{P}_p(F)]^2$ is not needed to guarantee the optimal rate of convergence and the projection $\mathbf{\Pi}_{p-1}^0 \nabla : (\mathcal{V}_p(F) + \mathbb{P}_p(F)) \rightarrow [\mathbb{P}_{p-1}(F)]^2$ is sufficient. In these situations, the projector $\mathbf{\Pi}_{p-1}^0 \nabla$ may be computed in the same way as we have detailed in this section, with the notable differences that the projections are expanded in $\mathcal{M}_{p-1}(F)$:

$$\mathbf{\Pi}_{p-1}^0 \partial_{x_j} \varphi_i = \sum_{\alpha=1}^{n_{p-1}} \left(\mathbf{\Pi}_{p-1}^{0,j} \right)_{\alpha,i} m_{\alpha}^1.$$

This means that the matrix $\mathbf{R}_j^F$ may be computed using just the internal degrees of freedom (case 1 in equation (4.5.7)).

4.5.4 L^2 Projector

Finally, we consider the L^2 projection $\mathbf{\Pi}_p^0 : [\mathcal{V}_p(F) + \mathbb{P}_p(F)] \rightarrow \mathbb{P}_p(F)$ which we shall denote with the $n_p \times N_{\text{dof}}^F$ matrix $\mathbf{\Pi}_p^{0,*}$. For each shape function φ_i we have:

$$\mathbf{\Pi}_p^0 \varphi_i = \sum_{\alpha=1}^{n_p} \left(\mathbf{\Pi}_p^{0,*} \right)_{\alpha,i} m_{\alpha}^1. \quad (4.5.8)$$

We compute $\mathbf{\Pi}_p^{0,*}$ using the L^2 orthogonality conditions:

$$\left(\mathbf{\Pi}_p^0 \varphi_i, m_{\beta}^1 \right)_{0,F} = \left(\varphi_i, m_{\beta}^1 \right)_{0,F}, \quad i = 1, \dots, N_{\text{dof}}^F, \quad \beta = 1, \dots, n_p.$$

Substituting the expansion of equation (4.5.8) gives:

$$\sum_{\alpha=1}^{n_p} \left(\mathbf{\Pi}_p^{0,*} \right)_{\alpha,i} \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F} = \left(\varphi_i, m_{\beta}^1 \right)_{0,F}, \quad \alpha, \beta = 1, \dots, n_p,$$

which yields the system of equations $\mathbf{H}_p \mathbf{\Pi}_p^{0,*} = \mathbf{C}$ where

$$(\mathbf{H}_p)_{\alpha,\beta} = \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F}, \quad (\mathbf{C})_{\alpha,i} = \left(m_{\alpha}^1, \varphi_i \right)_{0,F}.$$

The first n_{p-2} rows of $\mathbf{C}$ are given by the internal DoFs of each φ_i :

$$(\mathbf{C})_{\alpha,i} = |F| \delta_{\alpha,(i-pN_v)}, \quad \alpha = 1, \dots, n_{p-2}.$$

where δ is the kronecker-delta. We compute rows $\alpha = n_{p-2} + 1, \dots, n_p$ of $\mathbf{C}$ using the defining property of $\mathcal{V}_p(F)$ which is:

$$(m_\alpha^1, \varphi_i)_{0,F} = (m_\alpha^1, \Pi_p^\nabla \varphi_i)_{0,F} = \sum_{\beta=1}^{n_p} (m_\alpha^1, m_\beta^1)_{0,F} (\Pi_p^{\nabla,*})_{\beta,i}$$

Therefore, we have

$$(\mathbf{C})_{\alpha,i} = (\mathbf{H}_p \Pi_p^{\nabla,*})_{\alpha,i}, \quad \text{for } \alpha = n_{p-2} + 1, \dots, n_p$$

We may then compute the projector by inverting the system of equations: $\Pi_p^{0,*} = (\mathbf{H}_p)^{-1} \mathbf{C}$. As was the case with the energy projector, we also require the equivalent projection Π_p^0 onto the local Lagrangian basis, denoted Π_p^0 . Following exactly the same procedure as for the energy projection, Π_p^0 is given by

$$\Pi_p^0 = \mathbf{D} \Pi_p^{0,*}$$

4.5.5 Change of Polynomial Basis

At higher polynomial orders ($p > 2$) the local bilinear forms can become ill-conditioned, particularly for problems where there is element distortion [208]. To ameliorate this issue, we use the orthogonalised basis presented in section 4.2.2, which we denoted with $\mathcal{M}_p^2(F)$. Following the notation of [207], we shall denote orthogonalised versions of local matrices with an overbar. For example, the orthogonalised version of matrix $\mathbf{G}$ shall be denoted $\overline{\mathbf{G}}$. We shall denote the VEM basis associated with the orthogonalised polynomial basis with $\{\varphi_i^2\}$ and the original VEM basis with $\{\varphi_i^1\}$. Finally, we shall denote the orthogonalised degrees of freedom with $\text{dof}_i^2(\cdot)$ and the original degrees of freedom with $\text{dof}_i^1(\cdot)$.

Remark 8. *Both the boundary VE shape functions and the boundary DoFs for both the standard and orthogonalised bases coincide. Only the internal shape functions and DoFs differ. This means that $\varphi_i^1 = \varphi_i^2$ and $\text{dof}_i^1(v) = \text{dof}_i^2(v)$ for $i = 1, \dots, pN_v$.*

In practice, changing the polynomial basis means two things [207]: the first is changing the internal degrees of freedom ($\mathcal{D}_4$) to be the moments with the new basis:

$$\text{dof}_{pN_v+\alpha}^2(v) = \frac{1}{|F|} \int_F m_\alpha^2 v \, dx, \quad \forall m_\alpha^2 \in \mathcal{M}_{p-2}^2(F) \quad \text{and} \quad v \in \mathcal{V}_p(F).$$

The second is computing orthogonalised projectors $\overline{\Pi}_p^\nabla$, $\overline{\Pi}_p^0 \nabla$ and $\overline{\Pi}_p^0$ which project onto the new polynomial basis:

$$\begin{aligned} \overline{\Pi}_p^\nabla \varphi_i^2 &= \sum_{\alpha=1}^{n_p} (\overline{\Pi}_p^{\nabla,*})_{\alpha,i} m_\alpha^2, \\ \overline{\Pi}_p^0 \nabla \varphi_i^2 &= \left(\sum_{\alpha=1}^{n_p} (\overline{\Pi}_p^{0,1})_{\alpha,i} m_\alpha^2 \right) \mathbf{e}_1 + \left(\sum_{\alpha=1}^{n_p} (\overline{\Pi}_p^{0,2})_{\alpha,i} m_\alpha^2 \right) \mathbf{e}_2, \end{aligned}$$

$$\bar{\Pi}_p^0 \varphi_i^2 = \sum_{\alpha=1}^{n_p} \left(\bar{\Pi}_p^{0,*} \right)_{\alpha,i} m_\alpha^2$$

Note, that the orthogonalised projection matrices are also denoted with overbar.

Orthogonalised Energy Projector

This section shall explain how to compute the orthogonalised energy projector matrices $\bar{\Pi}_p^{\nabla,*}$ and $\bar{\Pi}_p^{\nabla}$ beginning with the orthogonalised matrix $\bar{\mathbf{G}}$, which is given by:

$$\left(\bar{\mathbf{G}} \right)_{\alpha,\beta} = \begin{cases} \int_{\partial F} m_\beta^2 ds & \text{for } \alpha = 1 \\ \left(\nabla m_\alpha^2, \nabla m_\beta^2 \right)_{0,F} & \text{for } \alpha > 1 \end{cases}$$

We may write the first row of $\bar{\mathbf{G}}$ in terms of $\mathbf{G}$ as follows:

$$\left(\bar{\mathbf{G}} \right)_{1,\beta} = \int_{\partial F} \sum_{\gamma=1}^{n_p} (\mathbf{M})_{\beta,\gamma} m_\gamma^1 ds = \sum_{\gamma=1}^{n_p} (\mathbf{M})_{\beta,\gamma} \int_{\partial F} m_\gamma^1 ds$$

therefore we have $\bar{\mathbf{G}}(1, 1 : n_p) = \mathbf{G}(1, 1 : n_p) \mathbf{M}^T$. For all other rows of $\bar{\mathbf{G}}$ we have:

$$\begin{aligned} \left(\bar{\mathbf{G}} \right)_{\alpha,\beta} &= \left(\nabla m_\alpha^2, \nabla m_\beta^2 \right)_{0,F} = \sum_{\gamma=1}^{n_p} \sum_{\eta=1}^{n_p} (\mathbf{M})_{\alpha,\gamma} (\mathbf{M})_{\beta,\eta} \left(\nabla m_\gamma^1, \nabla m_\eta^1 \right)_{0,F} \\ &= \left(\mathbf{M} \mathbf{G} \mathbf{M}^T \right)_{\alpha,\beta} \end{aligned}$$

for $2 \leq \alpha, \beta \leq n_p$. The orthogonalised matrix $\bar{\mathbf{B}}$ is given by:

$$\left(\bar{\mathbf{B}} \right)_{\alpha,i} = \begin{cases} \int_{\partial F} \varphi_i^2 ds, & \text{for } \alpha = 1 \\ \left(\nabla m_\alpha^2, \nabla \varphi_i^2 \right)_{0,F}, & \text{for } \alpha = 1, \dots, n_p \end{cases}.$$

The first row of $\bar{\mathbf{B}}$ and $\mathbf{B}$ are equal as they only involve boundary shape functions (c.f. remark 8). We may rewrite the remaining rows of $\bar{\mathbf{B}}$ as:

$$\left(\bar{\mathbf{B}} \right)_{\alpha,i} = - \left(\Delta m_\alpha^2, \varphi_i^2 \right)_{0,F} + \int_{\partial F} \nabla m_\alpha^2 \cdot \mathbf{n} \varphi_i^2 ds.$$

The first term is only non-zero when φ_i^2 corresponds to an internal DoF, following the same process we did for $\mathbf{B}$ we may write this term as:

$$\left(\bar{\mathbf{B}} \right)_{\alpha,i} = \sum_{\beta=1}^{n_p-2} |F| (\bar{\mathbf{L}})_{\alpha,\beta} \delta_{pN_v+\beta,i}$$

Note, that we have used the orthogonalised version of the Laplacian matrix $\bar{\mathbf{L}}$ introduced in equation (4.2.6). The second term is only non-zero when φ_i^2 corresponds to a boundary DoF

and is given by:

$$\begin{aligned} (\bar{\mathbf{B}})_{\alpha,i} &= \int_{\partial F} \nabla m_{\alpha}^2 \cdot \mathbf{n} \varphi_i^2 ds = \int_{\partial F} \sum_{\beta}^{n_p} (\mathbf{M})_{\alpha,\beta} \nabla m_{\beta}^1 \cdot \mathbf{n} \varphi_i^2 ds \\ &= \sum_{\beta=1}^{n_p} (\mathbf{M})_{\alpha,\beta} \int_{\partial F} \nabla m_{\beta}^1 \cdot \mathbf{n} \varphi_i^1 ds. \end{aligned}$$

Which may be written in Matlab notation as:

$$\bar{\mathbf{B}}(2 : n_p, 1 : pN_v) = \mathbf{M}(2 : n_p, 2 : n_p) \mathbf{B}(2 : n_p, 1 : pN_v)$$

The orthogonalised matrix $\bar{\mathbf{D}}$ is given by:

$$(\bar{\mathbf{D}})_{i,\alpha} = \text{dof}_i^2(m_{\alpha}^2) = \sum_{\beta=1}^{n_p} (\mathbf{M})_{\alpha,\beta} \text{dof}_i^2(m_{\beta}^1).$$

For the boundary DoFs, $i = 1, \dots, pN_v$, we have $\text{dof}_i^1(\cdot) = \text{dof}_i^2(\cdot)$ therefore:

$$(\bar{\mathbf{D}})_{i,\alpha} = (\mathbf{D}\mathbf{M}^T)_{i,\alpha}$$

For the internal DoFs, $i = pN_v + 1, \dots, pN_v + n_{p-2}$ we have:

$$(\bar{\mathbf{D}})_{i,\alpha} = \text{dof}_i^2(m_{\alpha}^2) = \frac{1}{|F|} \int_F m_{\beta}^2 m_{\alpha}^2 dx = \frac{1}{|F|} \delta_{\alpha,i}$$

where we have used the L^2 orthogonality property which defines the space $\mathcal{M}_p^2(F)$. Once the orthogonalised matrices $\bar{\mathbf{G}}$, $\bar{\mathbf{B}}$ and $\bar{\mathbf{D}}$ have been computed, the orthogonalised projectors are computed as before:

$$\bar{\Pi}_p^{\nabla,*} = (\bar{\mathbf{G}})^{-1} \bar{\mathbf{B}}, \quad \bar{\Pi}_p^{\nabla} = \bar{\mathbf{D}} \bar{\Pi}_p^{\nabla,*}$$

Orthogonalised Gradient Projector

Computing the orthogonalised gradient projector $\bar{\Pi}_p^0 \nabla$, means computing the two matrices: $\bar{\Pi}_p^{0,1}$ and $\bar{\Pi}_p^{0,2}$ corresponding to orthogonalised projections $\bar{\Pi}_p^0 \partial_{x_1}$ and $\bar{\Pi}_p^0 \partial_{x_2}$ respectively. The orthogonalised matrix $\bar{\mathbf{R}}_j$ is given by:

$$(\bar{\mathbf{R}}_j)_{\alpha,i} = (m_{\alpha}^2, \partial_{x_j} \varphi_i^2)_{0,F} = -(\partial_{x_j} m_{\alpha}^2, \varphi_i^2)_{0,F} + \int_{\partial F} m_{\alpha}^2 n_j \varphi_i^2 ds = (\bar{\mathbf{R}}_j^F)_{\alpha,i} + (\bar{\mathbf{R}}_j^{\partial F})_{\alpha,i}$$

The boundary matrix is only non-zero when φ_i^2 corresponds to a boundary DoF and is given by:

$$(\bar{\mathbf{R}}_j^{\partial F})_{\alpha,i} = \int_{\partial F} m_{\alpha}^2 n_j \varphi_i^2 ds = \sum_{\beta=1}^{n_p} (\mathbf{M})_{\alpha,\beta} \int_{\partial F} m_{\alpha}^1 n_j \varphi_i^1 ds.$$

therefore

$$\bar{\mathbf{R}}_j^{\partial F} = \mathbf{M} \mathbf{R}_j^{\partial F}.$$

The matrix $\overline{\mathbf{R}}_j^F$ may be computed in the same way as the unorthogonalised version $\mathbf{R}_j$ by simply replacing all the matrices with their orthogonalised versions:

$$(\overline{\mathbf{R}}_j^F)_{\alpha,i} = \begin{cases} -|F| \sum_{\beta=1}^{n_{p-2}} (\overline{\partial}_j^1)_{\alpha,\beta} \delta_{\beta,(i-pN_v)}, & \forall \alpha = 1, \dots, n_{p-1}, \\ (\overline{\partial}_j^1 \overline{\mathbf{H}}_{p-1,p} \overline{\mathbf{\Pi}}_p^\nabla)_{\alpha,i}, & \forall \alpha = n_{p-1} + 1, \dots, n_p, \end{cases}.$$

Recalling that the members of $\mathcal{M}_p^2(F)$ are L^2 orthogonal to one another, we therefore have $\overline{\mathbf{H}}_{p,p} = \mathbf{I}_{p,p}$ where $\mathbf{I}_{p,p}$ is the $n_p \times n_p$ identity matrix. Therefore, the orthogonalised derivative projection matrices are given by:

$$\overline{\mathbf{\Pi}}_{p-1}^{0,j} = \overline{\mathbf{R}}_j$$

Orthogonalised L^2 Projector

This section shall explain how to compute the orthogonalised L^2 projection matrices $\overline{\mathbf{\Pi}}_p^{0,*}$ and $\overline{\mathbf{\Pi}}_p^{0,*}$, beginning with the orthogonalised matrix $\overline{\mathbf{C}}$:

$$(\overline{\mathbf{C}})_{\alpha,i} = (m_\alpha^2, \varphi_i^2)_{0,F}.$$

As was the case with $\mathbf{C}$, the first n_{p-2} rows correspond to the internal DoFs:

$$(\overline{\mathbf{C}})_{\alpha,i} = |F| \text{dof}_i^2(\varphi_i^2) = |F| \delta_{\alpha,(i-pN_v)}, \quad \alpha = 1, \dots, n_{p-2}$$

The remaining rows are computed as follows:

$$(\overline{\mathbf{C}})_{\alpha,i} = (m_\alpha^2, \overline{\mathbf{\Pi}}_p^\nabla \varphi_i^2)_{0,F} = \sum_{\beta=1}^{n_p} (\overline{\mathbf{\Pi}}_p^\nabla)_{\beta,i} (m_\alpha^2, m_\beta^2)_{0,F} = \sum_{\beta=1}^{n_p} \delta_{\alpha,\beta} (\overline{\mathbf{\Pi}}_p^\nabla)_{\beta,i}.$$

Therefore, we have:

$$(\overline{\mathbf{C}})_{\alpha,i} = (\overline{\mathbf{\Pi}}_p^\nabla)_{\alpha,i}, \quad \alpha = n_{p-2} + 1, \dots, n_p$$

The projection matrices are then computed as follows:

$$\overline{\mathbf{\Pi}}_p^{0,*} = (\overline{\mathbf{H}}_{p,p})^{-1} \overline{\mathbf{C}} = \overline{\mathbf{C}}, \quad \overline{\mathbf{\Pi}}_p^0 = \overline{\mathbf{D}} \overline{\mathbf{\Pi}}_p^{0,*}$$

4.5.6 Computation of Local Bilinear Forms

In the neutron diffusion equation and neutron transport equation, there are only four local element matrices which we require to construct the discrete bilinear forms. These are the consistency part of the stiffness matrix $\mathbf{K}_F$, the consistency part of the mass matrix $\mathbf{M}_F$, consistency part of the advection matrix $\mathbf{T}_F$ and the stability matrix $\mathbf{S}_F$. Letting $F \in \mathcal{T}_h$ be

a polygon on which the VEM space $\mathcal{V}_p(F)$ has been defined and let $\{\varphi_i\}$ be the set of VEM basis functions on F . Then the definitions of the local matrices are given by:

$$\begin{aligned} (\mathbf{K}_F)_{i,j} &= \left(\Pi_{p-1}^0 \nabla \varphi_i, \Pi_{p-1}^0 \nabla \varphi_j \right)_{0,F}, \quad (\mathbf{M}_F)_{i,j} = \left(\Pi_p^0 \varphi_i, \Pi_p^0 \varphi_j \right)_{0,F} \\ (\mathbf{T}_F)_{i,j} &= \left(\boldsymbol{\Omega} \cdot \Pi_p^0 \nabla \varphi_i, \Pi_p^0 \varphi_j \right)_{0,F}, \quad (\mathbf{S}_F)_{i,j} = \sum_{l=1}^{N_{\text{dof}}^F} \text{dof}_l \left((I - \Pi_p^\nabla) \varphi_i \right) \text{dof}_l \left((I - \Pi_p^\nabla) \varphi_j \right) \end{aligned}$$

We highlight that the diffusion bilinear form uses $\Pi_{p-1}^0 \nabla$ while the advection bilinear form uses $\Pi_p^0 \nabla$. Throughout this thesis, we make the assumption that material coefficients are piece-wise constant and that they are constant within elements. Therefore, differential forms involving material coefficients will become, after discretisation using the VEM, scaled combinations of $\mathbf{K}_F$, $\mathbf{M}_F$, $\mathbf{T}_F$ and $\mathbf{S}_F$. For example:

$$\begin{aligned} -\nabla \cdot (D \nabla \phi) + \Sigma \phi &\rightarrow D_F \mathbf{K}_F + \Sigma_F \mathbf{M}_F + (D_F + h_F^2 \Sigma_F) \mathbf{S}_F \\ \boldsymbol{\Omega} \cdot \nabla \phi + \Sigma \phi &\rightarrow \mathbf{T}_F + \Sigma_F \mathbf{M}_F + \Sigma_F h_F^2 \mathbf{S}_F \end{aligned}$$

The meaning of the stability term scaling parameters shall be explained in section 6.3. The case where material properties $D(\mathbf{x})$, $\Sigma(\mathbf{x})$ and $\boldsymbol{\Omega}(\mathbf{x})$ are not constant within elements shall not be discussed here. However, it is not difficult to extend the present discussion to those cases. We refer the reader to [196] and [197] for a description of how this is done. The local matrices we have presented in this section use the un-orthogonalised, scaled-monomials as the polynomial basis on F . To compute the bilinear forms when the orthogonalised basis has been chosen, one simply replaces the projection matrices and inner product matrices $\mathbf{H}$ with their orthogonalised counterparts.

In addition to the local element matrices, there are situations where we must compute mass matrices on the edges of elements $e \subset \partial F$. For example, when a problem involves Robin boundary conditions or in the discontinuous-Galerkin VEM to represent loss on outgoing element boundaries. For a possibly curved edge e we present two kinds of mass matrix:

$$(\mathbf{M}_e)_{i,j} = (\varphi_i, \varphi_j)_{0,e}, \quad (\mathbf{N}_e)_{i,j} = (\boldsymbol{\Omega} \cdot \mathbf{n} \varphi_i, \varphi_j)_{0,e}.$$

where $\boldsymbol{\Omega} \in \mathbb{R}^2$ is the same vector which appears in the advection matrix. As we showed in section 4.5.1, the shape functions on the boundary edges e of element F have an explicit expansion in the local monomial basis defined on the parameter space $[0, 1]$, a fact which we exploit to compute $\mathbf{M}_e$ and $\mathbf{N}_e$. Finally, we also require the forcing terms resulting both from a forcing function $f \in L^2(F)$ and a boundary flux term $\mathbf{q}(\mathbf{x}) \in L^2(\partial V)$ resulting from Neumann boundary conditions:

$$(\mathbf{f}_F)_i = \left(f, \Pi_p^0 \varphi_i \right)_{0,F}, \quad (\mathbf{r}_e)_i = (\mathbf{q}(\mathbf{x}) \cdot \mathbf{n}, \varphi_i)_{0,e}$$

Stiffness Matrix

We begin our discussion with the consistency stiffness matrix:

$$\begin{aligned}
(\mathbf{K}_F)_{i,j} &= \left(\Pi_{p-1}^0 \nabla \varphi_i, \Pi_{p-1}^0 \nabla \varphi_j \right)_{0,F} \\
&= \left(\Pi_{p-1}^0 \partial_{x_1} \varphi_i, \Pi_{p-1}^0 \partial_{x_1} \varphi_j \right)_{0,F} + \left(\Pi_{p-1}^0 \partial_{x_2} \varphi_i, \Pi_{p-1}^0 \partial_{x_2} \varphi_j \right)_{0,F} \\
&= \sum_{\alpha=1}^{n_{p-1}} \sum_{\beta=1}^{n_{p-1}} \left\{ \left(\Pi_{p-1}^{0,1} \right)_{\alpha,i} \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F} \left(\Pi_{p-1}^{0,1} \right)_{\beta,j} + \left(\Pi_{p-1}^{0,2} \right)_{\alpha,i} \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F} \left(\Pi_{p-1}^{0,2} \right)_{\beta,j} \right\} \\
&= \left(\left(\Pi_{p-1}^{0,1} \right)^T \mathbf{H}_{p-1} \Pi_{p-1}^{0,1} + \left(\Pi_{p-1}^{0,2} \right)^T \mathbf{H}_{p-1} \Pi_{p-1}^{0,2} \right)_{i,j}
\end{aligned}$$

therefore,

$$\mathbf{K}_F = \left(\Pi_{p-1}^{0,1} \right)^T \mathbf{H}_{p-1} \Pi_{p-1}^{0,1} + \left(\Pi_{p-1}^{0,2} \right)^T \mathbf{H}_{p-1} \Pi_{p-1}^{0,2}.$$

Mass Matrix

Next, the consistency mass matrix:

$$\begin{aligned}
(\mathbf{M}_F)_{i,j} &= \left(\Pi_p^0 \varphi_i, \Pi_p^0 \varphi_j \right)_{0,F} = \sum_{\alpha=1}^{n_p} \sum_{\beta=1}^{n_p} \left(\Pi_p^{0,*} \right)_{\alpha,i} \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F} \left(\Pi_p^{0,*} \right)_{\beta,j} \\
&= \left(\left(\Pi_p^{0,*} \right)^T \mathbf{H}_p \Pi_p^{0,*} \right)_{i,j},
\end{aligned}$$

therefore:

$$\mathbf{M}_F = \left(\Pi_p^{0,*} \right)^T \mathbf{H}_p \Pi_p^{0,*}$$

Transport Matrix

Letting $\mathbf{\Omega} = (\Omega_1, \Omega_2)$, the transport matrix is computed as follows:

$$\begin{aligned}
(\mathbf{T}_F)_{i,j} &= \left(\mathbf{\Omega} \Pi_p^0 \nabla \varphi_i, \Pi_p^0 \varphi_j \right)_{0,F} \\
&= \Omega_1 \left(\Pi_p^0 \partial_{x_1} \varphi_i, \Pi_p^0 \varphi_j \right)_{0,F} + \Omega_2 \left(\Pi_p^0 \partial_{x_2} \varphi_i, \Pi_p^0 \varphi_j \right)_{0,F} \\
&= \sum_{\alpha=1}^{n_p} \sum_{\beta=1}^{n_p} \left\{ \Omega_1 \left(\Pi_p^{0,1} \right)_{\alpha,i} \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F} \left(\Pi_p^{0,*} \right)_{\beta,j} + \Omega_2 \left(\Pi_p^{0,2} \right)_{\alpha,i} \left(m_{\alpha}^1, m_{\beta}^1 \right)_{0,F} \left(\Pi_p^{0,*} \right)_{\beta,j} \right\} \\
&= \left(\left(\Omega_1 \Pi_p^{0,1} + \Omega_2 \Pi_p^{0,2} \right)^T \mathbf{H}_p \Pi_p^{0,*} \right)_{i,j}.
\end{aligned}$$

Therefore:

$$\mathbf{T}_F = \left(\Omega_1 \Pi_p^{0,1} + \Omega_2 \Pi_p^{0,2} \right)^T \mathbf{H}_p \Pi_p^{0,*}.$$

Stability Matrix

Finally, the stability matrix is computed as follows:

$$(\mathbf{S}_F)_{i,j} = \sum_{l=1}^{N_{\text{dof}}^F} \text{dof}_l \left((I - \Pi_p^0) \varphi_i \right) \text{dof}_l \left((I - \Pi_p^0) \varphi_j \right)$$

$$= \left((\mathbf{I} - \mathbf{\Pi}_p^0)^T (\mathbf{I} - \mathbf{\Pi}_p^0) \right)_{i,j},$$

therefore

$$\mathbf{S}_F = (\mathbf{I} - \mathbf{\Pi}_p^0)^T (\mathbf{I} - \mathbf{\Pi}_p^0).$$

Boundary Mass Matrices

As was explained in section 4.5.1, for every $F \in \mathcal{T}_h$ and each edge $e \subset \partial F$ there is a NURBS parameterisation γ_e which describes the geometry of e . For a VEM scheme of order p there will be $p + 1$ basis functions defined on parameter space $[0, 1]$, $\hat{\varphi}_i$ such that shape functions in real space φ_i satisfy $\varphi_i(\gamma_e t) = \hat{\varphi}_i(t)$. The coefficients in the local monomial basis are stored in matrix $\mathbf{\Pi}$ (equation (4.5.3)) therefore we may write $\mathbf{M}_e$ as:

$$(\mathbf{M}_e)_{i,j} = (\varphi_i, \varphi_j)_{0,e} = \sum_{\alpha=1}^{p+1} \sum_{\beta=1}^{p+1} (\mathbf{\Pi})_{\alpha,i} (\mathbf{\Pi})_{\beta,j} (\mu_\alpha, \mu_\beta)_{0,e} \implies \mathbf{M}_e = \mathbf{\Pi}^T \mathbf{H}_e \mathbf{\Pi}.$$

Similarly for $\mathbf{N}_e$ we have:

$$(\mathbf{N}_e)_{i,j} = (\mathbf{\Omega} \cdot \mathbf{n} \varphi_i, \varphi_j)_{0,e} = \sum_{\alpha=1}^{p+1} \sum_{\beta=1}^{p+1} (\mathbf{\Pi})_{\alpha,i} (\mathbf{\Pi})_{\beta,j} (\mathbf{\Omega} \mathbf{n} \mu_\alpha, \mu_\beta)_{0,e}$$

Letting $\mathbf{H}_{e,\mathbf{\Omega}}$ be the weighted inner product matrix $(\mathbf{\Omega} \mathbf{n} \mu_\alpha, \mu_\beta)_{0,e}$ we therefore have:

$$\mathbf{N}_e = \mathbf{\Pi}^T \mathbf{H}_{e,\mathbf{\Omega}} \mathbf{\Pi}.$$

Forcing Vectors

The forcing vector $\mathbf{f}_F$ is computed as follows:

$$(\mathbf{f}_F)_i = (f(\mathbf{x}), \Pi_p^0 \varphi_i)_{0,F} = \sum_{\alpha=1}^{n_p} (\Pi_p^0)_{\alpha,i} (f, m_\alpha)_{0,F} = ((\Pi_p^0)^T \hat{\mathbf{f}})_i$$

where $(\hat{\mathbf{f}})_\alpha = (f(\mathbf{x}), m_\alpha)_{0,F}$ is used to denote the $n_p \times 1$ vector of moments of $f(\mathbf{x})$. Given that we can't make many assumptions about $f(\mathbf{x})$ we must evaluate the moments by direct quadrature. The Neumann vector is computed as follows:

$$(\mathbf{r})_i = (\mathbf{q} \cdot \mathbf{n}, \varphi_i)_{0,e} = \sum_{\alpha=1}^{p+1} (\mathbf{\Pi})_{\alpha,i} (\mathbf{q} \cdot \mathbf{n}, \mu_\alpha)_{0,e}.$$

Therefore:

$$\mathbf{r} = \mathbf{\Pi}^T \mathbf{s}, \quad (\mathbf{s})_\alpha = (\mathbf{q} \cdot \mathbf{n}, \mu_\alpha)_{0,e}.$$

The vector $\mathbf{s}$ stores the moments of the function $r(\mathbf{x}) = \mathbf{q}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})$ on edge e . Again, we can make only minimal assumptions about the nature of $\mathbf{q}(\mathbf{x})$, therefore we evaluate the elements of $\mathbf{s}$ by direct quadrature on the edge e .

4.6 Discussion

In this chapter we have discussed the assumptions on the polygonal mesh necessary to guarantee the proper convergence rates of the VEM and we have reviewed the definition and manipulation of two polynomial bases: the set of scaled monomials and an L^2 orthogonal basis. We reviewed the theory of the local H^1 -conforming curvilinear VEM space which was initially presented in [1] and have gone into detail on how to implement such a local VEM space on curvilinear polygons with edges represented by NURBS curves. Finally, we gave a full account of how local projections are computed in practice, both for the scaled monomial basis and the orthogonalised basis.

The theory presented in this chapter will be applied to two problems in nuclear reactor physics: the multigroup NDE and the multigroup, S_N -NTE. The NDE is a coupled system of elliptic PDEs, therefore chapter 6 shall explain how a global, NURBS-enhanced, continuous-Galerkin VEM (NECG-VEM) space is constructed and used to discretise the NDE such that it may be solved in an efficient and scalable way. The S_N -NTE on the other hand, is a coupled system of hyperbolic PDEs for which continuous-Galerkin schemes are known to be unstable. Therefore, in chapter 7 we shall explain how a global, discontinuous-Galerkin VEM (DG-VEM) space is constructed and how it is used to discretise the multigroup S_N -NTE.

There are a number of interesting areas for future research with respect to the curvilinear VEM, these include:

- The extension of the H^1 -conforming virtual element to curvilinear polyhedra in 3D. As is the case in 2D, if one wishes to realise the full benefit of higher-order VEM schemes in 3D then development of an exact-geometry 3D VEM is a priority.
- The development of curvilinear versions of other virtual elements such as the $H(\text{div})$ and $H(\text{curl})$ conforming virtual elements [236]. This would allow a greater range of physical phenomena to be analysed using exact-geometry VEM schemes.
- The development of mesh to mesh interpolation and projection algorithms for future application in multi-physics problems.

Chapter 5

Geometry and Topology

5.1 Introduction

Typical geometries in nuclear reactor physics pose many challenges from a computational perspective. They possess geometric features which are both multi-scale and curvilinear. If one wishes to simulate the performance of such a reactor design without losing spatial information through homogenisation then the numerical method used to solve the governing equations must have the capability to exactly represent the reactor geometry.

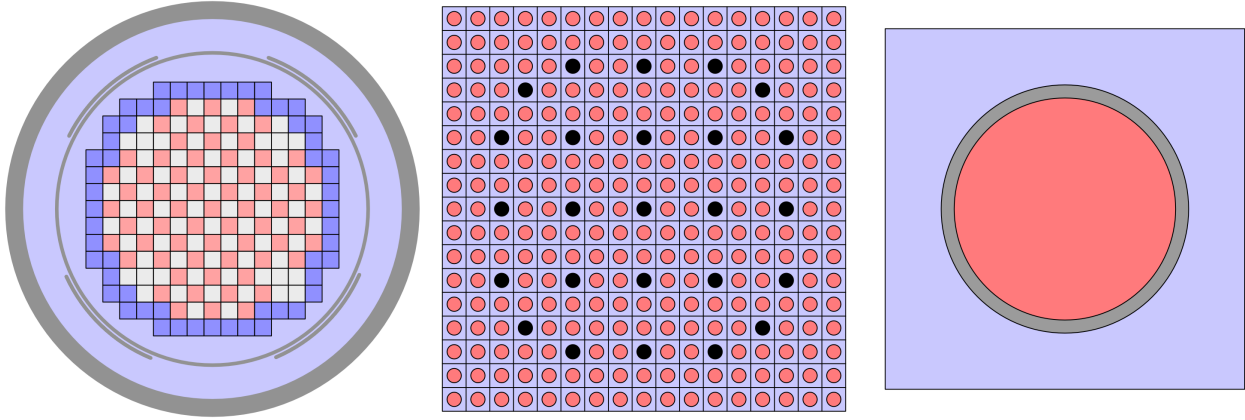


Figure 5.1: Simplified schematic of a PWR core, the left-most image is the full core with a length scale around 4-5m, the middle image is a fuel assembly with a length scale of around 15-20cm and the right-most image is a pincell with a length-scale of <1 cm. The pincell may have additional features which are smaller still, such as annular air gaps or cladding with thicknesses on the millimetre scale.

If the numerical method cannot exactly represent the geometry, then the error in the numerical solution shall become dominated by the geometric error. This is especially true when higher-order numerical methods such as the high-order FEM or VEM are used. These errors occur because quantities such as the arc length between a fuel pin and the moderator will be incorrect, thus affecting the rate of neutron leakage from the fuel to the moderator. A more serious error could be that the fissile mass of the pincells are not preserved between the geometric model

and the computational mesh. A direct consequence would be the inaccurate estimation of the reactivity of the core as a whole. Needless to say, this would have serious implications for reactor safety.

With these considerations in mind, this chapter will present the design and theory of the 2D geometric modelling kernel (GMK) developed as part of this project. The GMK had to be sufficiently general to represent a wide range of reactor geometries but also sufficiently memory-efficient to represent large, curved polygonal meshes. We have restricted the scope of the project to the 2D VEM and therefore all the reactor geometries are planar. Consequently, all reactor geometries considered in this thesis are 2-manifolds, meaning the neighbourhood of every point in the domain is homeomorphic with the unit disc in $\mathbb{R}^2$. This property simplifies the choice of data structure used to represent the topology of the reactor. The GMK is the underlying infrastructure required to implement a curvilinear VEM. This project makes no distinction between the "domain" and the "mesh" as they are commonly conceived in finite element analysis; both are represented using exactly the same data structure. The "mesh" is simply a more highly refined version of the "domain". However, we shall retain the terminology with "domain" used to describe the very low-refinement geometry which is constructed when posing a PDE problem while we shall use the term "mesh" to describe the highly refined, possibly curved, polygonal meshes used as an input for the curvilinear VEM.

This chapter is organised as follows.

- In section 5.2, a review of non-uniform rational B-splines (NURBS) shall be presented. NURBS are a cornerstone of computer aided geometric design (CAGD) and are used extensively in this project to represent parametric curves and surfaces. NURBS are the geometry description of the GMK.
- Section 5.3 shall apply some of the concepts and algorithms of NURBS to the problem of tessellating curved polygons. Two methods of numerical integration on curvilinear polygons are also presented: one for homogeneous functions and another for inhomogeneous functions.
- Section 5.4 will then introduce the doubly-connected edge list (DCEL), which we used to efficiently build, store and query the topology of the domain. The DCEL is the topology description of the GMK.
- Section 5.5 shall present the algorithm developed in this project for curvilinear, polygonal mesh generation using the Centroidal Voronoi Tessellation.
- Finally, section 5.6 shall provide a discussion of the strengths and limitations of the GMK and provide some suggestions for further research.

5.2 A Review of NURBS

Non-uniform rational B-splines (NURBS) became the standard description of geometry in the field of computer aided geometric design (CAGD) in the 1980s, primarily because they possess a number of properties, to be explained in due course, that make them uniquely suited to geometric modelling.

5.2.1 B-spline Basis

The **knot vector** is a non-decreasing set of real numbers $\mathbf{u} = [u_1, \dots, u_m]$ with each u_i referred to as a **knot**. Repeated knots are permitted and the number of times a knot appears in a knot vector is referred to as the **knot multiplicity**. The i 'th half-open interval $[u_i, u_{i+1})$, which may have zero length, are referred to as **knot spans**. From a given knot vector, a set of B-spline basis functions of polynomial order p (denoted $N_{i,p}(u)$) are defined as follows. For $p = 0$ [87]:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}.$$

and for $p > 0$ the basis functions are defined using the **Cox-deBoor** recursion [87, 237, 238]:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u). \quad (5.2.1)$$

From the definition of the B-spline basis functions it may be seen that for a knot vector with m knots there will be $r = m - (p + 1)$ basis functions of order p . A few important properties of B-spline basis functions are listed below. The proofs may be found in (section 2.1 of [87]).

Property 5.2.1. *The non-zero support of $N_{i,p}(u)$ is $[u_i, u_{i+p+1})$.*

Property 5.2.2. *On any given knot span $[u_i, u_{i+1})$ the only non-zero basis functions are $N_{i-p,p}, \dots, N_{i,p}$.*

Property 5.2.3. *On all knot spans $[u_i, u_{i+1})$ the local non-zero basis functions form a partition of unity:*

$$\sum_{j=i-p}^i N_{j,p}(u) = 1 \quad \forall \quad u \in [u_i, u_{i+1})$$

Property 5.2.4. *All $N_{i,p}(u)$ are non-negative everywhere in $\mathbb{R}$.*

Property 5.2.5. *Shape function $N_{i,p}$ is p times differentiable on the interior of the knot intervals between u_i and u_{i+p+1} and is $p - k$ times differentiable at a knot where k is the multiplicity of the knot.*

In the present work the following assumptions are made about all knot vectors used in curves

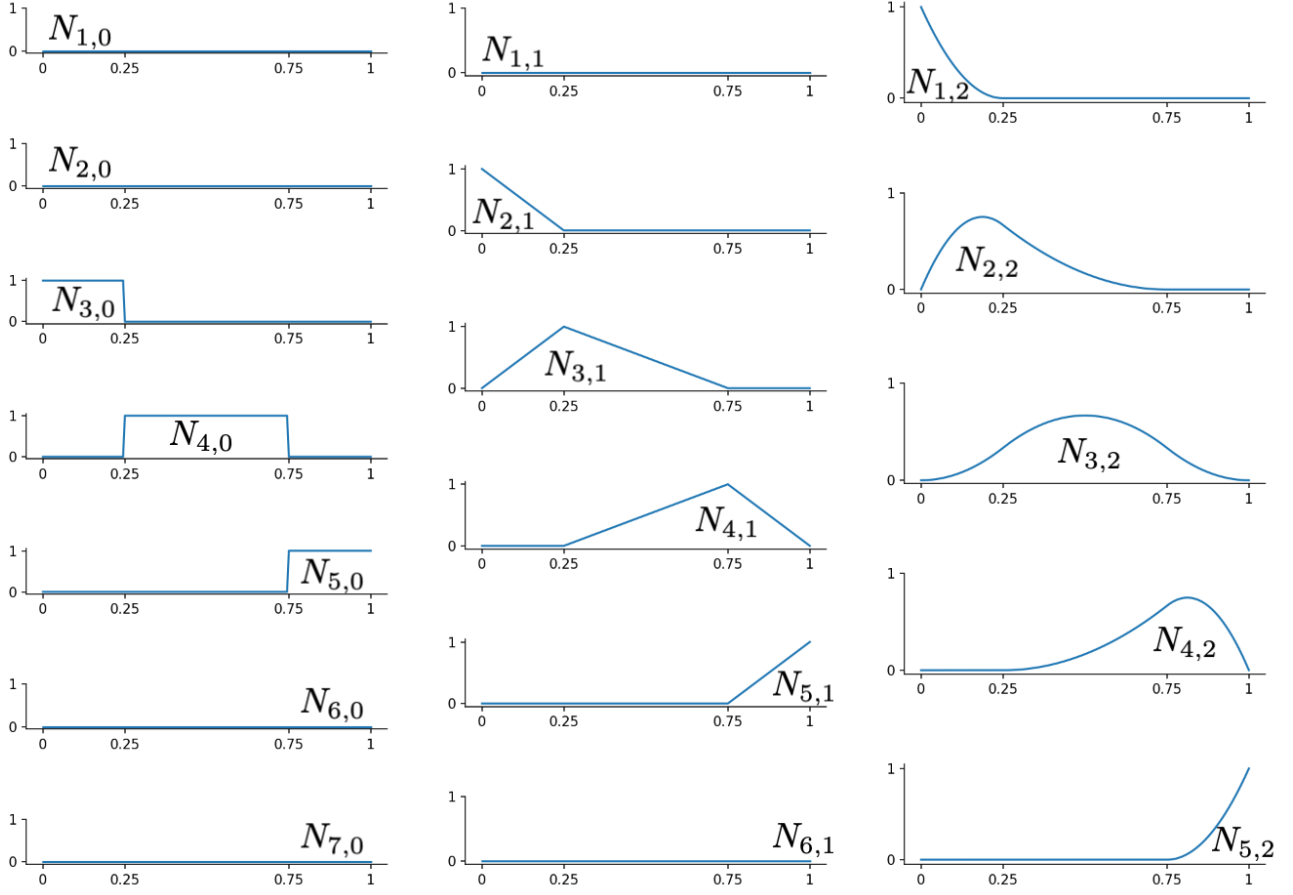


Figure 5.2: Plots of the basis functions of orders $p = 0, 1, 2$ defined on the knot vector $\mathbf{u} = [0, 0, 0, 0.25, 0.75, 1, 1, 1]$.

and surfaces: first that the non-zero support of all of the B-spline bases are limited to $[0, 1]$. Second, that the knot vectors are **open**, meaning for a B-spline basis of order p the knot vector has the following form [87]:

$$\mathbf{u} = [\underbrace{0, \dots, 0}_{p+1}, u_{p+1}, \dots, u_{m-p-1}, \underbrace{1, \dots, 1}_{p+1}].$$

The derivatives of the B-spline basis functions are computed using the formula [87]:

$$N_{i,p}^{(k)} := \frac{d^k N_{i,p}}{du^k} = \frac{p!}{(p-k)!} \sum_{j=0}^k \alpha_{k,j} N_{i+j,p-k} \quad (5.2.2)$$

where

$$\begin{aligned} \alpha_{0,0} &= 1 \\ \alpha_{k,0} &= \frac{\alpha_{k-1,0}}{u_{i+p-k+1} - u_i} \\ \alpha_{k,j} &= \frac{\alpha_{k-1,j} - \alpha_{k-1,j-1}}{u_{i+p+j-k+1} - u_{i+j}} \quad \text{for } j = 1, \dots, k-1 \\ \alpha_{k,k} &= \frac{\alpha_{k-1,k-1}}{u_{i+p+1} - u_{i+k}} \end{aligned}$$

The numerical algorithms used in this project to compute the B-spline basis functions and their derivatives are based on equations (5.2.1) and (5.2.2) and are simply C++ implementations of the algorithms presented in [87], section 2.5.

5.2.2 B-spline Curves and Surfaces

B-spline Curve

Given r points in $\mathbb{R}^d$, denoted $\{\mathbf{P}_i\}_{i=1}^r$ where $d = 2, 3$, a p -order B-spline curve is defined as [87]:

$$\mathbf{C}(u) = \sum_{i=1}^r N_{i,p}(u) \mathbf{P}_i.$$

B-spline curves are non-interpolatory, meaning that the curve does not pass through its control points unless the corresponding knot has a multiplicity of p . In which case the curve has C^0 continuity at that particular knot and will pass through the corresponding control point. We demonstrate this property using examples 1 and 2.

Example 1

In this example we have a cubic B-spline curve with the following knot vector:

$$\mathbf{u} = [0, 0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1, 1],$$

$$\{\mathbf{P}_i\} = \{(0, 0), (0.5, -0.5), (1, -0.5), (1, 0.5), (2, 0.5), (3, 1.5), (1, 2), (0, 1)\}.$$

Both the B-spline basis and the curve will therefore exhibit a minimum of C^2 continuity on the interior knots. This curve and its basis are shown in figure 5.3

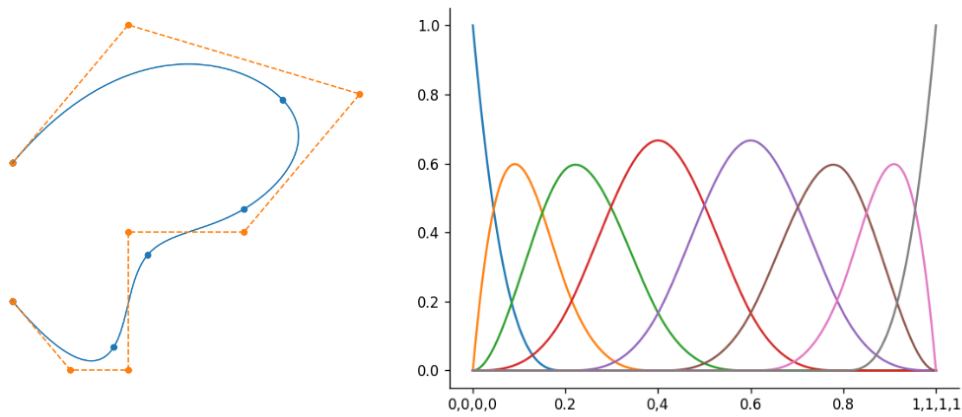


Figure 5.3: The first B-spline curve example.

As expected the curve does not pass through any of its interior control points.

Example 2

This example also uses a cubic B-spline however in this case the knot vector is given by:

$$\begin{aligned}\mathbf{u} &= [0, 0, 0, 0, 0.2, 0.4, 0.6, 0.6, 0.6, 0.8, 1, 1, 1, 1], \\ \{\mathbf{P}_i\} &= \{(0, 0), (0.5, -0.5), (1, -0.5), (1, 0.5), (2, 0.5), (3, 1.5), \\ &\quad (2.7, 2.5), (2.0, 3), (1.5, 1.5), (0, 1)\}\end{aligned}$$

The curve and basis shown figure 5.4

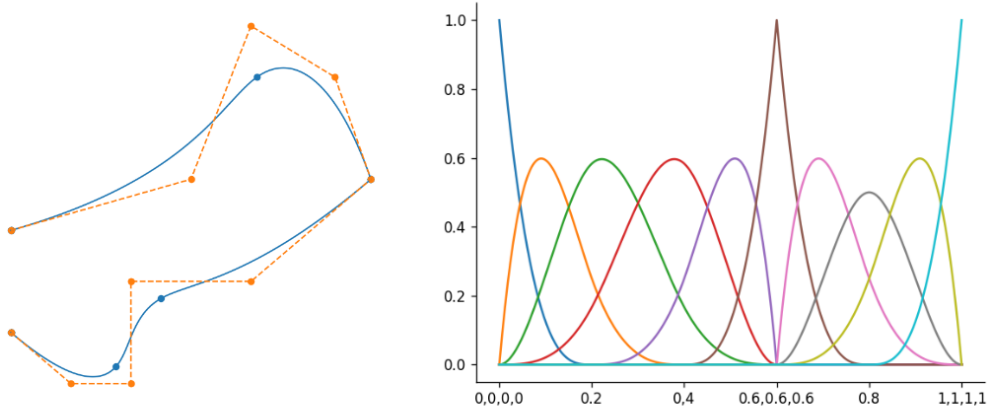


Figure 5.4: The second B-spline curve example.

As expected, at the knots $u_7 = u_8 = u_9 = 0.6$ the curve exhibits C^0 continuity and passes through control point $\mathbf{P}_6$.

Derivatives of B-spline curves are simple to compute and therefore so are quantities such as tangent ($\mathbf{T}(u)$) and normal ($\mathbf{N}(u)$) vectors:

$$\begin{aligned}\mathbf{C}^{(k)}(u) &:= \frac{d^k}{du^k} \mathbf{C}(u) = \sum_{i=1}^r N_{i,p}^{(k)}(u) \mathbf{P}_i, \\ \mathbf{T}(u) &= \frac{\mathbf{C}^{(1)}(u)}{\|\mathbf{C}^{(1)}(u)\|}, \quad \mathbf{N}(u) = \frac{\mathbf{C}^{(2)}(u)}{\|\mathbf{C}^{(2)}(u)\|}\end{aligned}\tag{5.2.3}$$

If $\mathbf{C}(u)$ is a plane curve then in practice the normal is computed using $\mathbf{N}(u) = \mathbf{T}(u) \times \mathbf{k}$ where $\mathbf{k}$ is the upward unit vector out of the plane. In the VEM it is often necessary to integrate functions along curves, there are two cases which appear most. The first is simply integrating a scalar function $f(\mathbf{x})$ along a curve, the second is integrating the dot product of a vector function $\mathbf{f}(\mathbf{x})$ with the normal vector. Letting $C = \{\mathbf{x} = \mathbf{C}(u) : u \in [0, 1]\}$ denote a B-spline curve then these integrations are performed in the standard way:

$$\int_C f(\mathbf{x}) ds = \int_0^1 f(\mathbf{C}(u)) J(u) du,\tag{5.2.4}$$

$$\int_C \mathbf{f}(\mathbf{x}) \cdot \mathbf{N} \, ds = \int_0^1 \mathbf{f}(\mathbf{C}(u)) \cdot \mathbf{N}(u) J(u) du \quad (5.2.5)$$

where $J(u)$ is the Jacobian of the curve which is evaluated as:

$$J(u) = \sqrt{\frac{d\mathbf{C}(u)}{du} \cdot \frac{d\mathbf{C}(u)}{du}} = \left\| \frac{d\mathbf{C}(u)}{du} \right\|_{\mathbb{R}^d}. \quad (5.2.6)$$

B-spline Surface

B-spline surfaces are constructed as follows: given two knot vectors $\mathbf{u} = [u_1, \dots, u_m]$ and $\mathbf{v} = [v_1, \dots, v_n]$, two polynomial orders p and q and an r -by- s 2D array of points $\mathbf{P}_{i,j}$ (where $r = m - (p + 1)$ and $s = n - (q + 1)$) the B-spline surface is defined as [87]:

$$\mathbf{S}(u, v) = \sum_{i=1}^r \sum_{j=1}^s N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j},$$

where $\{N_{i,p}(u)\}_{i=1}^r$ are the basis functions corresponding to $\mathbf{u}$ and $\{N_{j,q}(v)\}_{j=1}^s$ are the basis functions corresponding to $\mathbf{v}$. From the definition of the B-spline surface it is clear that the two-dimensional B-spline basis is constructed as the tensor-product of two one-dimensional B-spline bases. This is an important feature of B-spline surfaces as it means that their dependence on parameters u, v is separable. The first implication of this separability is that if v is held constant at some value v^* , the iso-curve along that fixed v is also a B-spline curve:

$$\mathbf{C}_{v^*}(u) := \mathbf{S}(u, v^*) = \sum_{i=1}^r N_{i,p}(u) \left(\sum_{j=1}^s N_{j,q}(v^*) \mathbf{P}_{i,j} \right) = \sum_{i=1}^r N_{i,p}(u) \mathbf{Q}_i.$$

Consequently, the differentiability of $\mathbf{S}(u, v)$ with respect to u is determined entirely by the B-spline basis in the u -parameter. So, if $p = 3$ and there is a knot in $\mathbf{u}$ with a multiplicity of 2 then the surface will have C^1 continuity at that knot with respect to u . By the same argument, the differentiability of the surface with respect to v is determined solely by the B-spline basis in the v -parameter. Derivatives of B-spline surfaces are computed as follows:

$$\mathbf{S}^{(l,m)}(u, v) := \frac{\partial^{m+l}}{\partial u^l \partial v^m} \mathbf{S}(u, v) = \sum_{i=1}^r \sum_{j=1}^s N_{i,p}^{(l)}(u) N_{j,q}^{(m)}(v) \mathbf{P}_{i,j},$$

and the normal vector to a B-spline surface may therefore be written as:

$$\mathbf{N}(u, v) = \mathbf{S}^{(1,0)}(u, v) \times \mathbf{S}^{(0,1)}(u, v) / \left\| \mathbf{S}^{(1,0)}(u, v) \times \mathbf{S}^{(0,1)}(u, v) \right\|_{\mathbb{R}^d}. \quad (5.2.7)$$

As with the B-spline curve, it is common in VEM to compute integrals on B-spline surfaces. Letting $S = \{\mathbf{x} = \mathbf{S}(u, v) : \forall (u, v) \in [0, 1]^2\}$ denote a B-spline surface the integral of a given function $f(\mathbf{x})$ is computed as follows:

$$\int_S f(\mathbf{x}) dx = \iint_0^1 f(\mathbf{S}(u, v)) J(u, v) du dv \quad (5.2.8)$$

where the Jacobian $J(u, v)$ is given by:

$$J(u, v) = \left\| \mathbf{S}^{(1,0)}(u, v) \times \mathbf{S}^{(0,1)}(u, v) \right\|_{\mathbb{R}^d} \quad (5.2.9)$$

A discussion of numerical integration to evaluate equations (5.2.4), (5.2.5) and (5.2.8) will be left to section 5.3, where numerical integration is discussed in detail.

Example 3

Here we give an example B-spline surface where $p = q = 2$, $\mathbf{u} = \mathbf{v} = [0, 0, 0, 0.5, 1, 1, 1]$ and the control points are computed using:

$$\mathbf{P}_{i,j} = \left(2 + 2\frac{(j-1)}{3} \right) \begin{bmatrix} \cos\left(\frac{\pi(i-1)}{6}\right) \\ \sin\left(\frac{\pi(i-1)}{6}\right) \end{bmatrix}$$

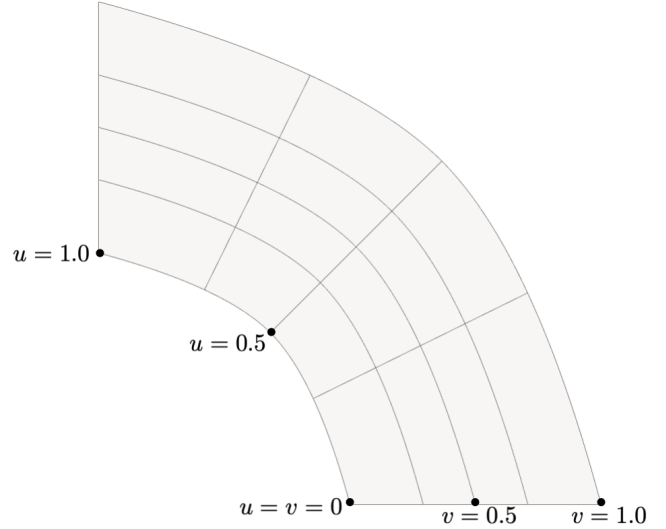
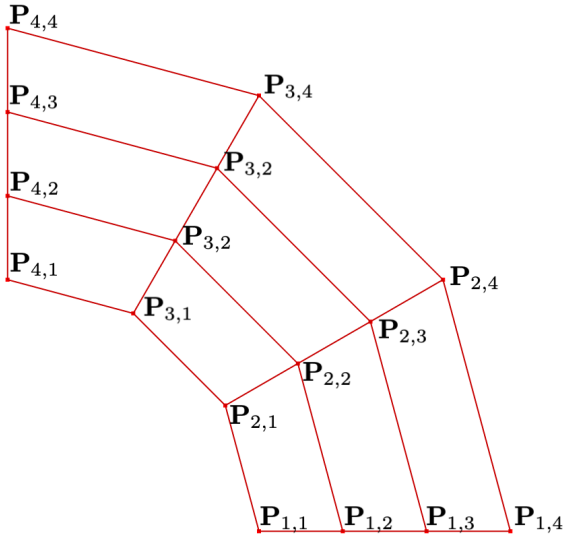


Figure 5.5: An example B-spline surface with the control net shown on the left and the surface on the right. The iso-lines in each parameter have been added.

5.2.3 NURBS Curves and Surfaces

While B-spline geometry is powerful and can represent a wide range of geometries, it has one major limitation: it cannot exactly represent conic sections. Conic sections derive their name from a method of constructing them, whereby each type of curve is constructed from the locus of intersection points between a cone and a plane in $\mathbb{R}^3$ (see figure 5.6). Conic sections are ubiquitous in engineering design, therefore a description of geometry which cannot accurately represent them will be limited in its application. The most important conic section in nuclear reactor design is the ellipse (of which the circle is a special case). To understand why B-spline

curves cannot accurately represent ellipses consider the parametric form of the ellipse:

$$x(t) = a \cos(t), \quad y(t) = b \sin(t) \quad \text{where} \quad t \in [0, 2\pi)$$

Using the transformation $u = \tan(t/2)$ we have:

$$x(u) = a \frac{1 - u^2}{1 + u^2}, \quad y(u) = b \frac{2u}{1 + u^2} \quad \text{where} \quad u \in [0, \infty).$$

The ellipse is therefore a rational polynomial, consequently there is no B-spline curve capable of exactly representing it.

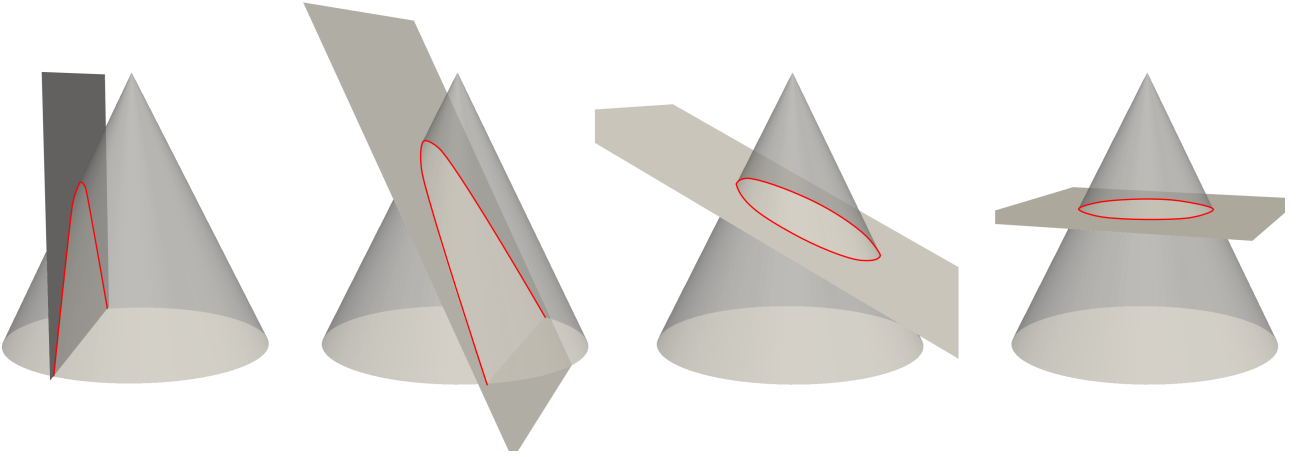


Figure 5.6: Each type of conic section: from left to right we have 1) hyperbola, 2) parabola, 3) ellipse and 4) circle

To solve this problem, NURBS curves and surfaces extend B-spline curves and surfaces to be piecewise rational polynomials as opposed to merely piecewise polynomials. The NURBS curve is defined using the same data as the B-spline curve: a polynomial order p , a knot vector $\mathbf{u}$ and a set of control points $\{\mathbf{P}_i\}$. In addition, the NURBS curve is defined using a set of control weights $\{w_i\}$, one for each control point. Using this data, the NURBS curve is defined as:

$$\mathbf{C}(u) = \frac{\sum_{i=1}^r N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{j=1}^r N_{j,p}(u) w_j} = \sum_{i=1}^r R_i^p(u) \mathbf{P}_i \quad (5.2.10)$$

where

$$R_i^p(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=1}^r w_j N_{j,p}(u)}$$

are the 1D rational basis functions. NURBS surfaces are defined using the same data as B-spline surfaces (orders p and q , knot vectors $\mathbf{u}$ and $\mathbf{v}$ and control net $\{\mathbf{P}_{i,j}\}$) and additionally a 2-dimensional array of weights, one for each control point $\{w_{i,j}\}$. Using this data, the NURBS surface is defined as:

$$\mathbf{S}(u, v) = \frac{\sum_{i=1}^r \sum_{j=1}^s N_{i,p}(u) N_{j,q}(v) w_{i,j} \mathbf{P}_{i,j}}{\sum_{k=1}^r \sum_{l=1}^s N_{k,p}(u) N_{l,q}(v) w_{k,l}} = \sum_{i=1}^r \sum_{j=1}^s R_{i,j}^{p,q}(u, v) \mathbf{P}_{i,j} \quad (5.2.11)$$

where

$$R_{i,j}^{p,q}(u, v) = \frac{N_{i,p}(u) N_{j,q}(v) w_{i,j}}{\sum_{k=1}^r \sum_{l=1}^s N_{k,p}(u) N_{l,q}(v) w_{k,l}}$$

are the 2D rational basis functions. In practice, NURBS curves and surfaces in this project were not computed using equations (5.2.10) and (5.2.11) directly but rather by computing them in homogeneous coordinates, where they have an equivalent B-spline representation, and then applying the perspective map to recover the NURBS representation in real coordinates [87]. Homogeneous coordinates are constructed as follows: given a point-weight pair, $(\mathbf{P}_i, w_i)$, the homogeneous control point, denoted $\mathbf{P}_i^w$ is given by $\mathbf{P}_i^w = [w_i \mathbf{P}_i, w_i]$. Once in homogeneous coordinates a NURBS object in d dimensions may be represented as a B-spline object in $d + 1$ dimensions. So a NURBS curve $\mathbf{C}(u)$ and a NURBS surface $\mathbf{S}(u, v)$ have the following equivalent forms in homogeneous coordinates:

$$\mathbf{C}^w(u) = \sum_{i=1}^r N_{i,p}(u) \mathbf{P}_i^w, \quad \mathbf{S}^w(u, v) = \sum_{i=1}^r \sum_{j=1}^s N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j}^w$$

The perspective map $H : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ then recovers the NURBS form: $\mathbf{C}(u) = H(\mathbf{C}^w(u))$ and $\mathbf{S}(u, v) = H(\mathbf{S}^w(u, v))$. The perspective map derives its name from its geometric interpretation, whereby points in homogeneous coordinates are projected into the $w = 1$ hyperplane by finding the intersection of the rays from the homogeneous control points to the origin. An example is given in figure 5.7 for a circular NURBS curve.

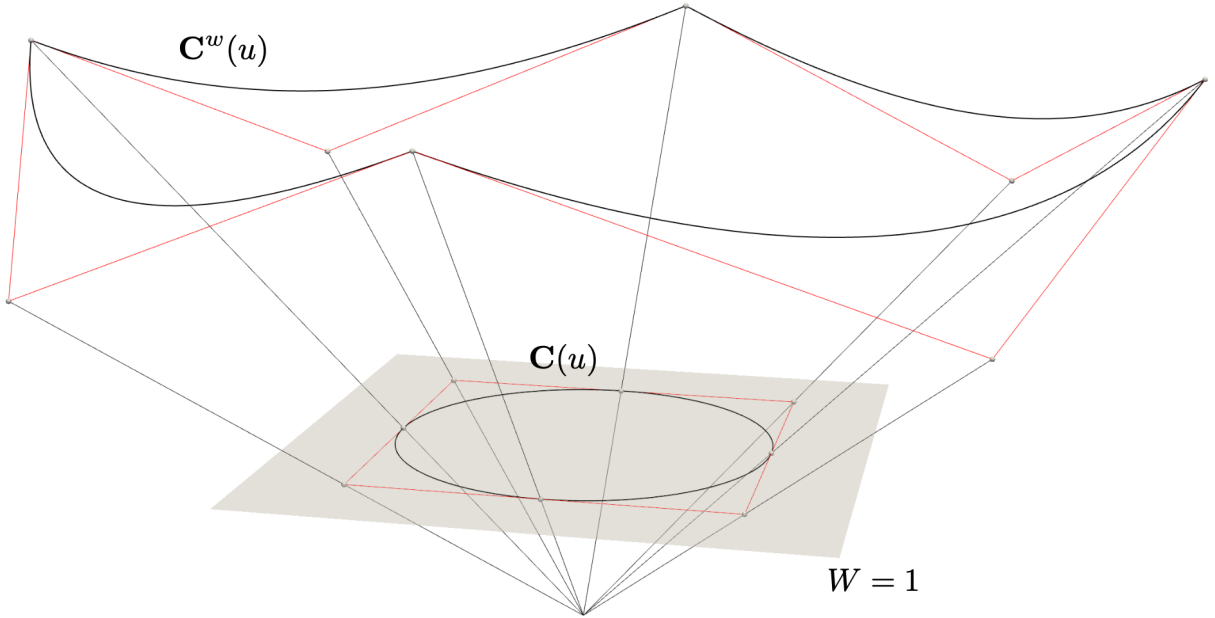


Figure 5.7: A circular NURBS curve $\mathbf{C}(u)$ and its counterpart in homogeneous coordinates $\mathbf{C}^w(u)$. Based on an image presented in [2].

Letting the weight functions be defined as:

$$w(u) = \sum_{i=1}^r N_{i,p}(u) w_i, \quad w(u, v) = \sum_{i=1}^r \sum_{j=1}^s N_{i,p}(u) N_{j,q}(v) w_{i,j}$$

the homogeneous form of both the NURBS curve and surface may be written as follows:

$$\mathbf{C}^w(u) = \begin{bmatrix} w(u)\mathbf{C}(u) \\ w(u) \end{bmatrix} = \begin{bmatrix} \mathbf{A}(u) \\ w(u) \end{bmatrix}, \quad \mathbf{S}^w(u, v) = \begin{bmatrix} w(u, v)\mathbf{S}(u, v) \\ w(u, v) \end{bmatrix} = \begin{bmatrix} \mathbf{A}(u, v) \\ w(u, v) \end{bmatrix}.$$

Using this notation the derivatives of the NURBS curve and surface may be computed as follows. Beginning with the NURBS curve:

$$\mathbf{A}^{(k)}(u) = (w(u)\mathbf{C}(u))^{(k)} = \sum_{i=0}^k \binom{k}{i} w^{(i)} \mathbf{C}^{(k-i)}(u) = w(u)\mathbf{C}^{(k)}(u) + \sum_{i=1}^k \binom{k}{i} w^{(i)} \mathbf{C}^{(k-i)}(u)$$

rearranging we get:

$$\mathbf{C}^{(k)}(u) = \frac{\mathbf{A}^{(k)}(u) - \sum_{i=1}^k \binom{k}{i} w^{(i)} \mathbf{C}^{(k-i)}(u)}{w(u)}.$$

Therefore the k 'th derivative may be computed from the lower derivatives $0, \dots, k-1$. We may perform exactly the same derivation for NURBS surfaces:

$$\begin{aligned} \mathbf{A}^{(k,l)}(u, v) &= ((w(u, v)\mathbf{S}(u, v))^{(k)})^{(l)} = \sum_{i=0}^k \binom{k}{i} (w^{(i,0)} \mathbf{S}^{(k-i,0)})^{(l)} \\ &= \sum_{i=0}^k \sum_{j=0}^l \binom{k}{i} \binom{l}{j} w^{(i,j)} \mathbf{C}^{(k-i, l-j)} \end{aligned}$$

which may be written as:

$$\mathbf{A}^{(k,l)}(u, v) = w \mathbf{S}^{(k,l)} + \sum_{i=1}^k \binom{k}{i} w^{(i,0)} \mathbf{S}^{(k-i,0)} + \sum_{j=1}^l \binom{l}{j} w^{(0,j)} \mathbf{S}^{(0, l-j)} + \sum_{i=1}^k \sum_{j=1}^l \binom{k}{i} \binom{l}{j} w^{(i,j)} \mathbf{C}^{(k-i, l-j)}.$$

Finally, rearranging for $\mathbf{S}^{(k,l)}$ gives:

$$\begin{aligned} \mathbf{S}^{(k,l)} &= \frac{1}{w} \left[\mathbf{A}^{(k,l)}(u, v) - \sum_{i=1}^k \binom{k}{i} w^{(i,0)} \mathbf{S}^{(k-i,0)} - \right. \\ &\quad \left. \sum_{j=1}^l \binom{l}{j} w^{(0,j)} \mathbf{S}^{(0, l-j)} - \sum_{i=1}^k \sum_{j=1}^l \binom{k}{i} \binom{l}{j} w^{(i,j)} \mathbf{C}^{(k-i, l-j)} \right]. \end{aligned}$$

Using these expressions for NURBS curve/surface derivatives, the formulae for tangents, normals and Jacobians (equations (5.2.3), (5.2.6) (5.2.7), (5.2.9)) are exactly the same as for the B-spline curves (and for any parametric curve/surface for that matter). NURBS curves/surfaces are infinitely differentiable on the interior of knot spans, where they are rational polynomials. However, on the knots themselves they have the same properties as B-spline curves/surfaces i.e. a curve of order p and a knot u_i of multiplicity k will be $p - k$ differentiable at that knot value. Similarly for NURBS surfaces, if a surface has order p in the u -parameter and u_i has multiplicity of k then $\mathbf{S}(u, v)$ will be $p - k$ differentiable with respect to u at (u_i, v) for all $v \in [0, 1]$.

Remark 9. *An important property of NURBS curves/surfaces is that when all the weights are*

equal they reduce to B-spline curves/surfaces. Therefore, B-splines are simply a special case of NURBS. This property also simplifies their implementation in C++ code as B-splines do not require separate classes/functions, they can be computed by simply making all the weights equal to 1 in a NURBS implementation.

All of the algorithms for computing NURBS curves and surfaces, based on the equations presented in this section, may be found in chapter 4 of [87].

5.2.4 Geometric Operations on NURBS

This section shall introduce two fundamental operations: **knot insertion** and **order elevation**. Their application within VEM shall be covered in greater detail in section 5.3.

Knot Insertion

Given a NURBS curve $\mathbf{C}^w(u)$ in homogeneous form with order p , knot vector $\mathbf{u}$ and control points $\{\mathbf{P}_i^w\}$, the process of knot insertion involves adding a knot $\hat{u} \in [u_k, u_{k+1})$ to the knot vector and generating a new set of homogeneous points $\{\mathbf{Q}_i^w\}$ such that the curve is geometrically and parametrically unchanged [87]. The relationship between $\mathbf{Q}_i^w$ and $\mathbf{P}_i^w$ is given by:

$$\mathbf{Q}_i^w = \alpha_i \mathbf{P}_i^w + (1 - \alpha_i) \mathbf{P}_{i-1}^w \quad (5.2.12)$$

where

$$\alpha_i = \begin{cases} 1 & \text{if } i \leq k - p \\ (\hat{u} - u_i)/(u_{i+p} - u_i) & \text{if } k - p + 1 \leq i \leq k \\ 0 & \end{cases} \quad (5.2.13)$$

In example 4 we demonstrate the effect of knot insertion on the B-spline basis and the set of control points. While knot insertion does not affect the continuity of the curve itself, which is geometrically unchanged, it does affect the continuity of the B-spline basis. Equations (5.2.12) and (5.2.13) can be generalised to the case where the same knot is inserted more than once (see [87], section 5.1). Knot insertion on surfaces may be performed on either knot vector $\mathbf{u}$ or $\mathbf{v}$ using the same algorithm as for curves. The main difference on surfaces is that the non-changed dimension remains constant, this amounts to applying equations (5.2.12) and (5.2.13) to rows or columns of surface control points $\{\mathbf{P}_{i,j}^w\}$.

Example 4

In this example we have $p = 2$ and $\mathbf{u} = [0, 0, 0, 0.5, 1, 1, 1]$ and the control points/weights are:

$$\{\mathbf{P}_i\} = \{(0, 0), (2, 1), (1, 2), (0.5, 1)\} \quad \{w_i\} = \{1, 2, 1, 2\}$$

and we insert the knot $\hat{u} = 0.75$. The curve and its B-spline basis before knot insertion are shown in figure 5.8.

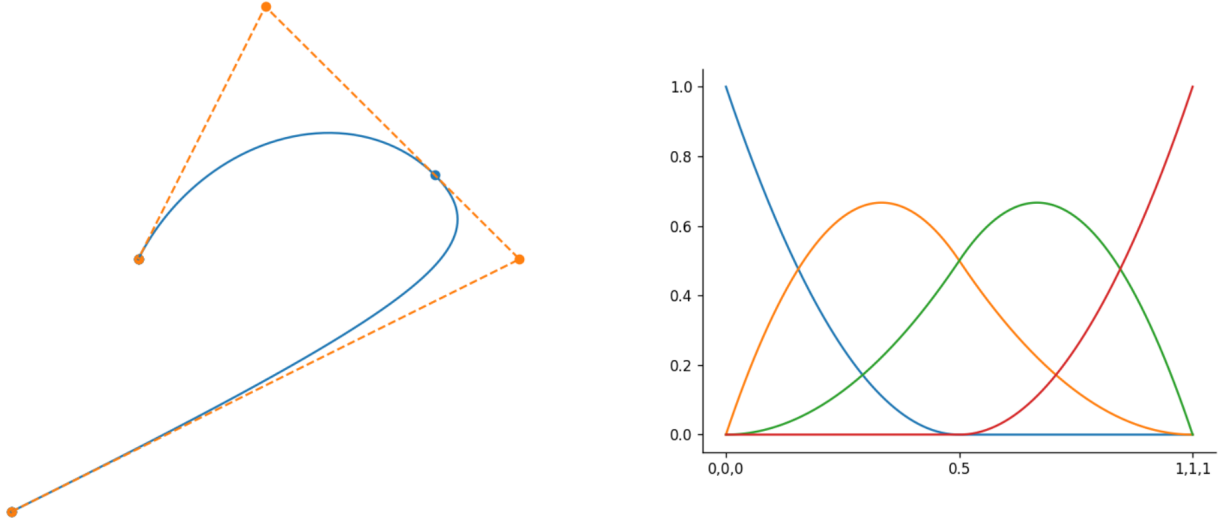


Figure 5.8: Curve and basis before knot insertion.

The curve and basis after knot insertion:

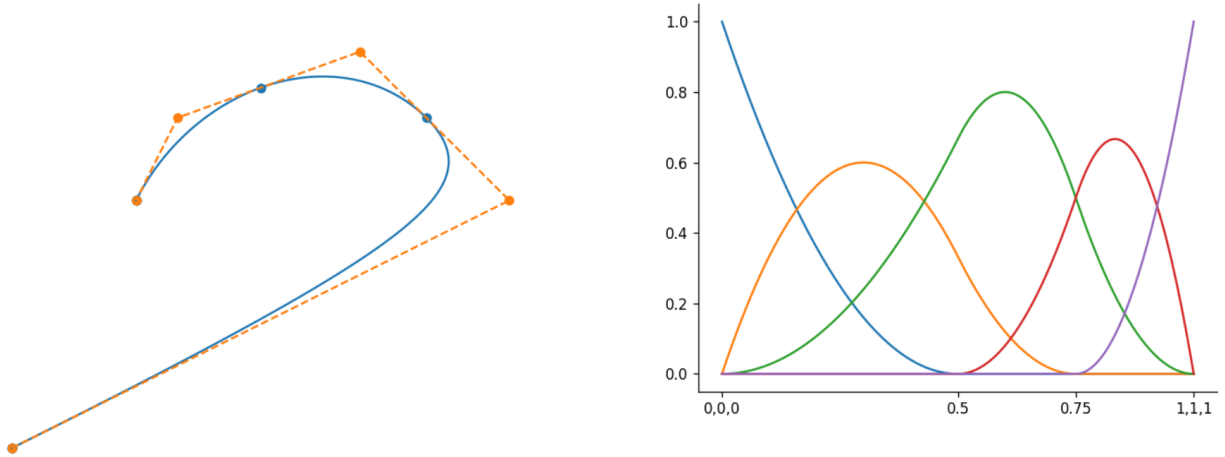


Figure 5.9: Curve and basis after knot insertion.

The main applications of knot insertion in the curved VEM are curve splitting and curve extraction. To split a curve at some parameter $\hat{u}$ that knot must be inserted into the curve until its multiplicity is p . While the curve will be unchanged, the basis will have become C^0 continuous at $\hat{u}$. The curve is then split by assigning one set of the newly created control points to what we shall call the left curve $\mathbf{C}_l(u)$ and the remaining set to the right curve $\mathbf{C}_r(u)$. We demonstrate this example 5:

Example 5

We begin with the curve shown in example 4, figure 5.8 and we wish to split this curve at $\hat{u} = 0.75$. After 2 knot insertions the curve and the B-spline basis look like this:

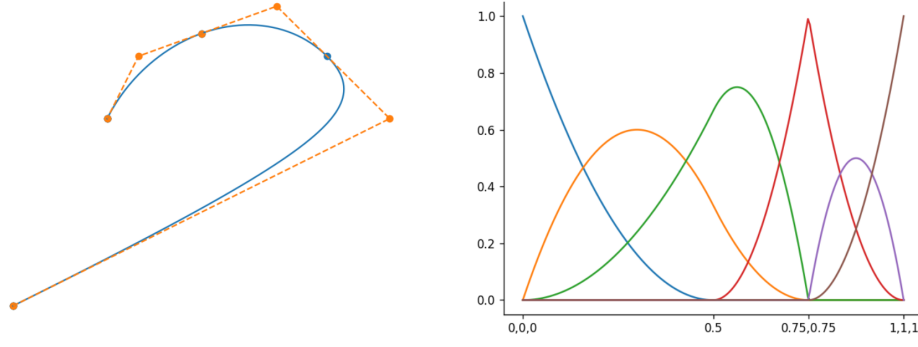


Figure 5.10: Curve after 2 knot insertions at $\hat{u} = 0.75$.

The curve after knot insertion has the following data (values rounded for brevity):

$$\begin{aligned}\mathbf{u} &= [0, 0, 0, 0.5, 0.75, 0.75, 1, 1, 1] \\ \{\mathbf{P}_i\} &= \{(0, 0), (2, 1), (1.4, 1.6), (1.0, 1.45), (0.67, 1.33), (0.5, 1.0)\} \\ \{w_i\} &= \{1.0, 2.0, 1.25, 1.375, 1.5, 2.0\}\end{aligned}$$

The left and right curves are assigned the following data:

$$\begin{aligned}\mathbf{u}^l &= [0, 0, 0, 0.5, 0.75, 0.75, 0.75] & \mathbf{u}^r &= [0.75, 0.75, 0.75, 1, 1, 1] \\ \{\mathbf{P}_i^l\} &= \{(0, 0), (2, 1), (1.4, 1.6), (1.0, 1.45)\} & \{\mathbf{P}_i^r\} &= \{(1.0, 1.45), (0.67, 1.33), (0.5, 1.0)\} \\ \{w_i^l\} &= \{1.0, 2.0, 1.25, 1.375\} & \{w_i^r\} &= \{1.375, 1.5, 2.0\}\end{aligned}$$

After the data for each curve has been computed the knot vectors for each curve are renormalised to span $[0, 1]$.

Curve extraction is the process of computing the sub-curve corresponding to a parameter interval $[u_l, u_r]$. The algorithm is simple and makes recursive use of the curve splitting routine:

Algorithm 2 Curve extraction algorithm.

```

1: Inputs A curve  $\mathbf{C}(u)$  and an interval  $[u_l, u_r] \subset [0, 1]$ 
2: if  $u_l = 0$  then
3:   Split  $\mathbf{C}(u)$  at  $u_r$  to give  $\{\mathbf{C}_l, \mathbf{C}_r\}$ .
4:   return  $\mathbf{C}_l$ 
5: else if  $u_r = 1$  then
6:   Split  $\mathbf{C}(u)$  at  $u_l$  and give  $\{\mathbf{C}_l, \mathbf{C}_r\}$ .
7:   return  $\mathbf{C}_r$ 
8: else
9:   Split  $\mathbf{C}(u)$  at  $u_l$  to give  $\{\mathbf{C}_l, \mathbf{C}_r\}$ 
10:  Split  $\mathbf{C}_r$  at  $u_r$  to give  $\{\mathbf{C}_{r,l}, \mathbf{C}_{r,r}\}$ 
11:  return  $\mathbf{C}_{r,l}$ 
12: end if
```

As shall be seen in later sections these two operations of splitting and extraction are invaluable

for the implementation of the curved VEM.

Order Elavation

Order elevation is the process of increasing the polynomial order of a NURBS curve and computing the set of knots, points and weights which leave the curve geometrically and parametrically unchanged [87]. Let $\mathbf{C}_p^w(u)$ be a NURBS curve in homogeneous coordinates with initial order p , initial control points $\{\mathbf{P}_i^w\}$ and initial knot vector of the form:

$$\mathbf{u}_p = [\underbrace{0, \dots, 0}_{p+1}, \underbrace{u_1, \dots, u_1}_{m_1}, \underbrace{u_2, \dots, u_2}_{m_2}, \dots, \underbrace{u_g, \dots, u_g}_{m_g}, \underbrace{1, \dots, 1}_{p+1}]$$

where, for each internal knot u_i , the number $m_i \geq 1$ is the corresponding multiplicity. We denote the length of the knot vector with $m = 2(p+1) + (m_1 + \dots + m_g)$ and we denote the number of unique internal knots with g . As usual we denote the number of control points with $r = m - p - 1$. The problem of order elevation may be formally posed as follows: find a new knot vector $\mathbf{u}_{p+1}$ and control points $\mathbf{Q}_i^w$ such that the curve $\mathbf{C}_{p+1}^w(u)$ of order $p+1$ satisfies:

$$\mathbf{C}_{p+1}^w(u) = \mathbf{C}_p^w(u) \quad \forall u \in [0, 1]$$

Given that order elevation must preserve geometry and therefore must preserve the continuity of the curve at each knot, then the multiplicity of each knot must increased by 1 to give:

$$\mathbf{u}_{p+1} = [\underbrace{0, \dots, 0}_{p+2}, \underbrace{u_1, \dots, u_1}_{m_1+1}, \underbrace{u_2, \dots, u_2}_{m_2+1}, \dots, \underbrace{u_g, \dots, u_g}_{m_g+1}, \underbrace{1, \dots, 1}_{p+2}].$$

Therefore, the number of knots in elevated knot vector will be $\hat{m} = m + g + 2$ and the number of elevated control points will be $\hat{r} = \hat{m} - p - 2$. The next item to compute are the new control points $\{\mathbf{Q}_i^w\}$. A simple but inefficient method of computing the control points is to set up a system of linear equations with the new coordinates of $\mathbf{Q}_i^w$ as unknowns:

$$\sum_{i=1}^{\hat{r}} \hat{N}_{i,p+1}(v_j) \mathbf{Q}_i^w = \sum_{i=1}^r N_{i,p}(v_j) \mathbf{P}_i^w \quad \text{for } v_j \in [0, 1] \quad \text{and } j = 1, 2, \dots, \hat{r} \quad (5.2.14)$$

where $\{\hat{N}_{i,p+1}\}$ is the B-spline basis associated with $\mathbf{u}_{p+1}$ and $\{N_{i,p}\}$ is the B-spline basis associated with the knot vector $\mathbf{u}_p$. Letting $\mathbf{I}$ denote the $(d+1) \times (d+1)$ identity matrix then equation (5.2.14) may be written as:

$$\begin{bmatrix} \hat{N}_{1,p+1}(v_1)\mathbf{I} & \hat{N}_{2,p+1}(v_1)\mathbf{I} & \dots & \hat{N}_{\hat{r},p+1}(v_1)\mathbf{I} \\ \hat{N}_{1,p+1}(v_2)\mathbf{I} & \hat{N}_{2,p+1}(v_2)\mathbf{I} & \dots & \hat{N}_{\hat{r},p+1}(v_2)\mathbf{I} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{N}_{1,p+1}(v_{\hat{r}})\mathbf{I} & \hat{N}_{2,p+1}(v_{\hat{r}})\mathbf{I} & \dots & \hat{N}_{\hat{r},p+1}(v_{\hat{r}})\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_1^w \\ \mathbf{Q}_2^w \\ \vdots \\ \mathbf{Q}_{\hat{r}}^w \end{bmatrix} = \begin{bmatrix} \mathbf{C}_p^w(v_1) \\ \mathbf{C}_p^w(v_2) \\ \vdots \\ \mathbf{C}_p^w(v_{\hat{r}}) \end{bmatrix}$$

The order-elevation algorithm presented in [87] algorithm 5.9, which we have implemented for the curved VEM, operates by sweeping through the knot vector from low to high and for each knot span $[u_i, u_{i+1})$ doing the following:

1. Extract a Bezier segment on $[u_i, u_{i+1})$, denoted $\mathbf{B}_i(u)$. This involves inserting u_i and u_{i+1} until their multiplicity is p and then extracting the control points and knot vector which affect the curve restricted to $[u_i, u_{i+1})$.
2. Next, the Bezier segment is order elevated, this is a simple operation for Bezier segments.
3. The final step is to remove the unnecessary copies of u_i and u_{i+1} , in effect this "stitches" the Bezier segment back to the rest of the curve.

The full details of this algorithm are outside the scope of this thesis, the reader may consult [87], Chapter 5 for a more complete treatment. Finally, we shall present an example of order elevation.

Example 6

Again, we begin with the curve shown in example 4, figure 5.8 and we elevate it to $p = 3$ and we get the following knots, weights and points:

$$\mathbf{u} = [0, 0, 0, 0, 0.5, 0.5, 1, 1, 1, 1] \quad \{w_i\} = \{1.0000, 1.6667, 1.8333, 1.1667, 1.3333, 2.0000\}$$

$$\{\mathbf{P}_i\} = \{(0, 0), (1.6, 1.8), (1.91, 1.09), (1.29, 1.71), (0.75, 1.5), (0.5, 1)\}$$

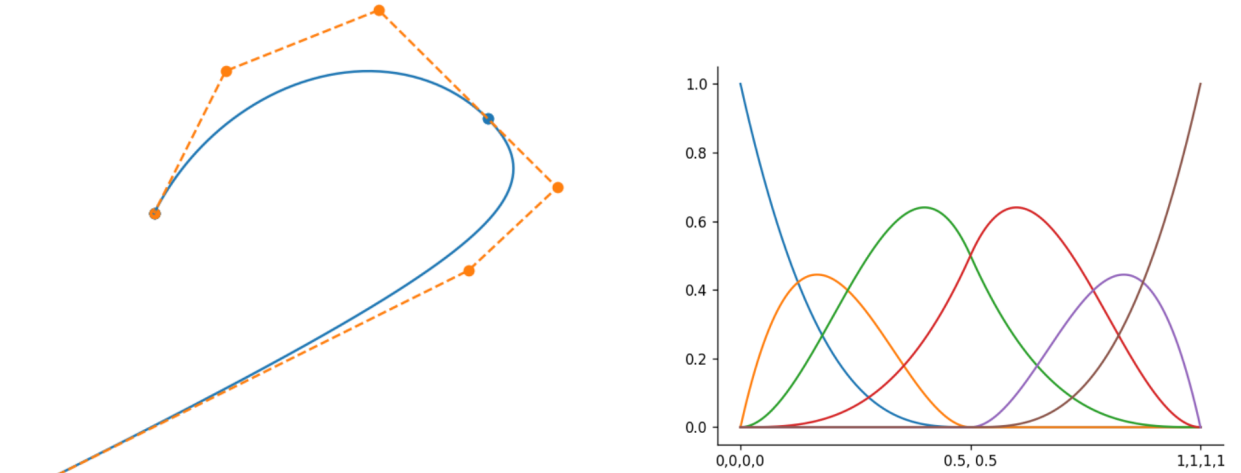


Figure 5.11: The curve of example 4 after order elevation.

5.2.5 Surface Construction

The curved VEM developed in this project requires the subdivision of curved polygons with NURBS edges into curved NURBS quadrilaterals. To this end we use three types of surface

construction from boundary curves: **bilinear quadrilaterals**, **ruled surfaces** and **Coons surfaces**. We shall discuss them here in general terms and discuss their application in the curved VEM to later sections.

The Bilinear Quadrilateral

This is the simplest of the three surface constructions presented here. The bilinear quad is the ideal surface representation if one wishes to create a surface which has 4 straight lines as its boundary curves. More formally, we wish to compute a NURBS surface $\mathbf{S}(u, v)$ which satisfies (see figure 5.12):

$$\begin{aligned} \mathbf{S}(0, 0) &= \mathbf{v}_1, & \mathbf{S}(1, 0) &= \mathbf{v}_2, & \mathbf{S}(1, 1) &= \mathbf{v}_3, & \mathbf{S}(0, 1) &= \mathbf{v}_4, \\ \mathbf{S}(u, 0) &= \mathbf{C}_{1,u}(u), & \mathbf{S}(u, 1) &= \mathbf{C}_{2,u}(u), & \mathbf{S}(0, v) &= \mathbf{C}_{1,v}(v), & \mathbf{S}(1, v) &= \mathbf{C}_{2,v}(v). \end{aligned}$$

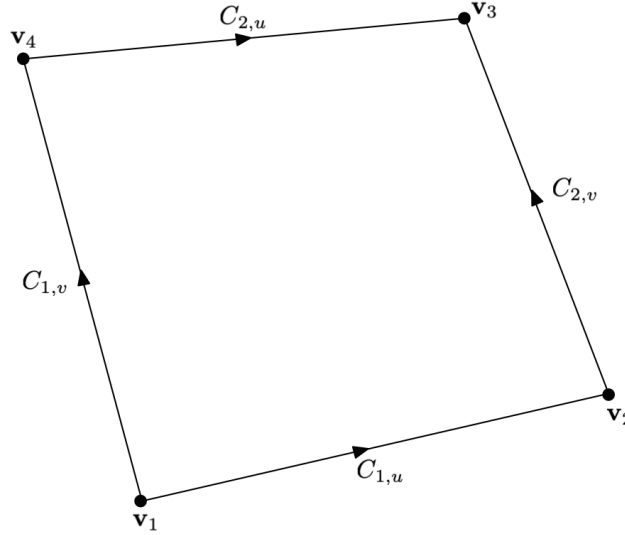


Figure 5.12: A bilinear quadrilateral

The simplest NURBS surface which satisfies these constraints is:

$$\begin{aligned} p = q = 1, \quad \mathbf{u} = \mathbf{v} = [0, 0, 1, 1], \quad \{\mathbf{P}_{1,1} = \mathbf{v}_1, \mathbf{P}_{2,1} = \mathbf{v}_2, \mathbf{P}_{1,2} = \mathbf{v}_4, \mathbf{P}_{2,2} = \mathbf{v}_3\} \\ \{w_{i,j} = 1 \quad \text{for } i, j = \{1, 2\}\} \end{aligned}$$

The Ruled Surface

The ruled surface is a method of constructing a NURBS surface which linearly interpolates two NURBS curves, denoted $\mathbf{C}_1(u)$ and $\mathbf{C}_2(u)$ with the first defined with order p_1 , knot vector $\mathbf{u}_1$, control point set $P_1 = \{\mathbf{P}_i^1\}$ and weight set $W_1 = \{w_i^1\}$ and the second defined with order p_2 , knot vector $\mathbf{u}_2$ and control point set $P_2 = \{\mathbf{P}_i^2\}$ and weight set $W_2 = \{w_i^2\}$. Therefore we seek

a NURBS surface $\mathbf{S}(u, v)$ given by:

$$\mathbf{S}(u, v) = \sum_{i=1}^r \sum_{j=1}^s R_{i,j}^{p,1}(u, v) \mathbf{Q}_{i,j}$$

which satisfies

$$\mathbf{S}(u, v) = v \mathbf{C}_1(u) + (1 - v) \mathbf{C}_2(u). \quad (5.2.15)$$

where $\mathbf{v} = [0, 0, 1, 1]$ and the order in the v -parameter is $q = 1$. Therefore, the problem is to find order p , knot vector $\mathbf{u}$ and control points and weights $\mathbf{Q}_{i,j}$, $w_{i,j}$ such that equation (5.2.15) is satisfied. The algorithm to do so follows these steps [87]: Steps 3-6 of the ruled surface

Algorithm 3 Ruled surface construction.

- 1: **Inputs:** NURBS curves $\mathbf{C}_1$ and $\mathbf{C}_2$
- 2: **Returns:** Ruled surface $\mathbf{S}(u, v)$
- 3: The order of the surface is found to be $p = \max(p_1, p_2)$.
- 4: Whichever curve has the lower order, let us say for the sake of argument it is $\mathbf{C}_1(u)$, is order-elevated to p . It's knot vector $\mathbf{u}_1$ now holds the order-elevated knot vector.
- 5: Find the merged knot vector $\mathbf{u}$, this is made up of all the knots in $\mathbf{u}_1 \cup \mathbf{u}_2$ and the multiplicity of each knot is set to the maximum of either $\mathbf{u}_1$ or $\mathbf{u}_2$. For example, if $\mathbf{u}_1 = [0, 0, 0, 0.1, 0.2, 1, 1, 1]$ and $\mathbf{u}_2 = [0, 0, 0, 0.2, 0.2, 0.7, 1, 1, 1]$ then $\mathbf{u} = [0, 0, 0, 0.1, 0.2, 0.2, 0.7, 1, 1, 1]$.
- 6: For both $\mathbf{C}_1$ and $\mathbf{C}_2$, perform knot insertion until $\mathbf{u}_1 = \mathbf{u}_2 = \mathbf{u}$. Both curves are now the same order and have the same knot vector, therefore they both have the same number of control points/weights. Let us denote them with $\hat{\mathbf{P}}_i^1$, $\hat{w}_i^1$ and $\hat{\mathbf{P}}_i^2$, $\hat{w}_i^2$.
- 7: Finally, the surface $\mathbf{S}(u, v)$ is constructed by setting $p = \max(p_1, p_2)$, $q = 1$, $\mathbf{u}$ to the knot vector found in steps 3-6, $v = [0, 0, 1, 1]$, $\mathbf{Q}_{i,1} = \hat{\mathbf{P}}_i^1$, $\mathbf{Q}_{i,2} = \hat{\mathbf{P}}_i^2$, $w_{i,1} = \hat{w}_i^1$ and $w_{i,2} = \hat{w}_i^2$.
- 8: Construct NURBS surface:

$$\mathbf{S}(u, v) = \text{makeSurface}(p, q, \mathbf{u}, \mathbf{v}, \{\mathbf{Q}_{i,j}\}, \{w_{i,j}\})$$

algorithm make curves $\mathbf{C}_1$ and $\mathbf{C}_2$ **compatible**, meaning they both have the same order and are defined on the same knot vector. Due to the tensor-product nature of NURBS surfaces, compatibility is a prerequisite for interpolation of NURBS curves. We provide an example of a ruled surface in example 7.

Example 7

Consider two curves defined with the following, for $\mathbf{C}_1(u)$:

$$p_1 = 2, \quad \mathbf{u}_1 = [0, 0, 0, 0.75, 1, 1, 1], \quad P_1 = \{(0, 0), (0.5, -0.5), (2, 0), (3, -1.5)\}$$

$$W_1 = \{1, 2, 1, 3\}$$

and for $\mathbf{C}_2(u)$:

$$p_2 = 3, \quad \mathbf{u}_2 = [0, 0, 0, 0, 0.5, 1, 1, 1, 1], \quad P_2 = \{(0, 2), (1, 3), (2, 2), (3, 4), (4, 2)\}$$

$$W_2 = \{1, 2, 1, 3, 1\}.$$

Figure 5.13 provides a graphical explanation of how the ruled surface is constructed for these two curves.

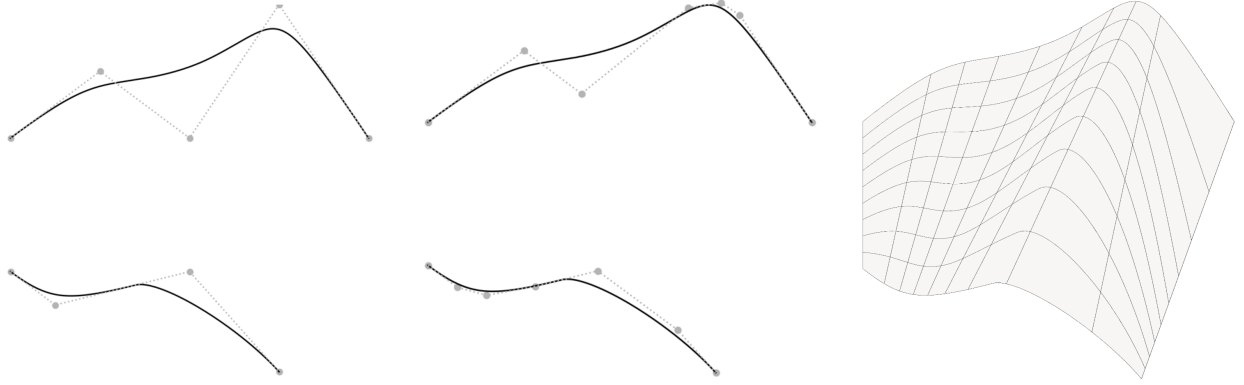


Figure 5.13: On the left we have the two original curves, in the middle we have the the curves after they have been made compatible and on the right we have the final ruled surface.

The Coons Surface

The Coons surface, named after Steven A. Coons [239, 240], is another type of interpolation surface. One which bilinearly interpolates four bounding curves such that the outer edges of the surface conform to the boundary curves. Let $\mathbf{C}_{u,1}$, $\mathbf{C}_{u,2}$, $\mathbf{C}_{v,1}$ and $\mathbf{C}_{v,2}$ be four NURBS curves arranged in a loop such as that shown in figure 5.14. The constraints on the curves, conditions which must be satisfied in order to have a well-defined Coons surface, are:

$$\begin{aligned} \mathbf{C}_{u,1}^w(0) &= \mathbf{C}_{v,1}^w(0), & \text{Bottom left corner} \\ \mathbf{C}_{u,1}^w(1) &= \mathbf{C}_{v,2}^w(0), & \text{Bottom right corner} \\ \mathbf{C}_{u,2}^w(1) &= \mathbf{C}_{v,2}^w(1), & \text{Top right corner} \\ \mathbf{C}_{u,2}^w(0) &= \mathbf{C}_{v,1}^w(1), & \text{Top left corner.} \end{aligned}$$

In these corner-compatibility constraints, the curves are in homogeneous form, meaning that both the end-points and the end-weights of the curves must coincide. We also assume that the curves on opposing sides have already been made compatible using steps 3-6 of algorithm 3. The Coons surface has the following definition:

$$\mathbf{S}(u, v) = \sum_{i=1}^r \sum_{j=1}^s R_{i,j}^{p,q}(u, v) \mathbf{Q}_{i,j} = \mathbf{S}_1(u, v) + \mathbf{S}_2(u, v) + \mathbf{S}_3(u, v) \quad (5.2.16)$$

where $\mathbf{S}_1(u, v)$ is the ruled surface between $\mathbf{C}_{u,1}$ and $\mathbf{C}_{u,2}$, $\mathbf{S}_2(u, v)$ is the ruled surface between $\mathbf{C}_{v,1}$ and $\mathbf{C}_{v,2}$ and $\mathbf{S}_3(u, v)$ is the bilinear quad with corners equal to the endpoints of the boundary curves. We therefore seek to find the orders p, q , knot vectors $\mathbf{u}, \mathbf{v}$, control points $\mathbf{P}_{i,j}$ and weights $w_{i,j}$ such that equation (5.2.16) is satisfied. We denote the data associated with each surface with a subscript, for example the control points belonging to surface $\mathbf{S}_1$ are denoted $\mathbf{P}_{1,i,j}$. The Coons surface procedure is given by algorithm 4 where the expressions for the Coons control points and weights (equations (5.2.17) and (5.2.18)) were originally presented in [241] and where the representation of Coons surface in NURBS form was discussed.

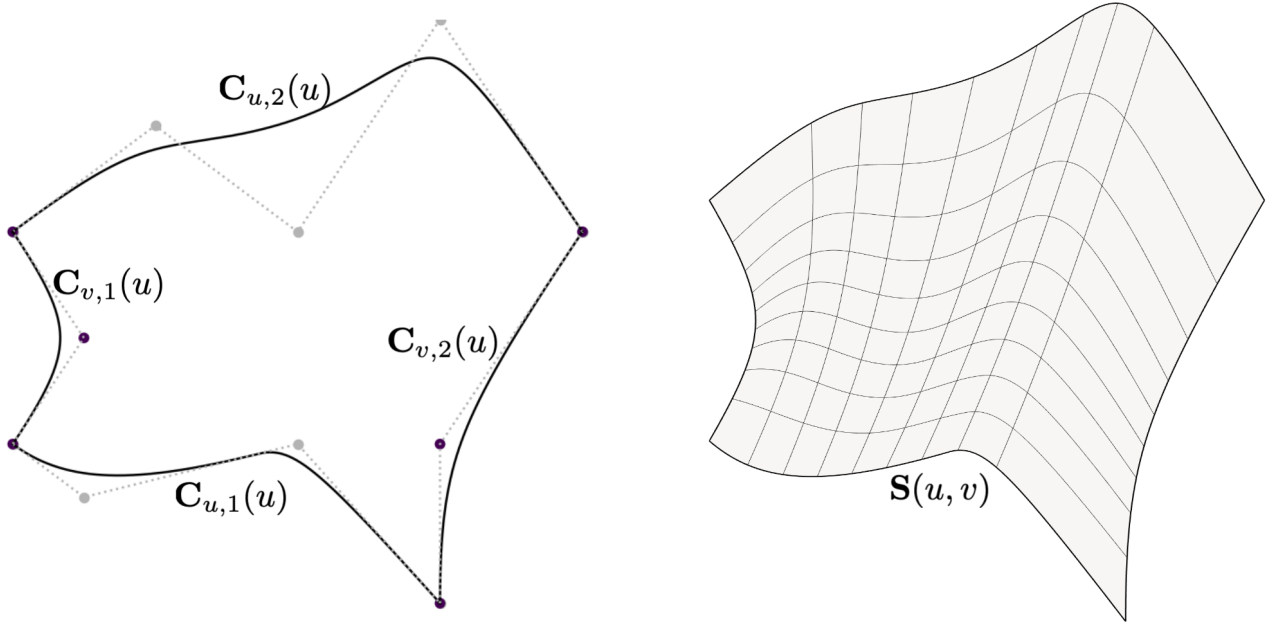


Figure 5.14: On the left we have four NURBS curves arranged in a loop, arrows represent the orientation of the curves. On the right a Coons surface, represented in NURBS form.

Algorithm 4 Coons surface construction.

- 1: **Inputs:** Set of NURBS curves $\mathbf{C}_{u,1}$, $\mathbf{C}_{u,2}$, $\mathbf{C}_{v,1}$ and $\mathbf{C}_{v,2}$
- 2: **Returns:** Coons surface $\mathbf{S}(u, v)$
- 3: The order of the Coons patch in each dimension is $p = \max(p_1, p_2, p_3)$, and $q = \max(q_1, q_2, q_3)$.
- 4: **for** Each surface $\mathbf{S}_i \in \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\}$ **do**
- 5: **if** $p_i < p$ **then**
- 6: Elevate the order of $\mathbf{S}_i$ in the u -parameter to p .
- 7: **end if**
- 8: **if** $q_i < q$ **then**
- 9: Elevate the order of $\mathbf{S}_i$ in the v -parameter to q .
- 10: **end if**
- 11: **end for**
- 12: Find common knot vectors $\mathbf{u} = \mathbf{u}_1 \cup \mathbf{u}_2 \cup \mathbf{u}_3$ and $\mathbf{v} = \mathbf{v}_1 \cup \mathbf{v}_2 \cup \mathbf{v}_3$.
- 13: **for** Each surface $\mathbf{S}_i \in \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\}$ **do**
- 14: Insert missing knots into $\mathbf{u}_i$ until $\mathbf{u}_i = \mathbf{u}$.
- 15: Insert missing knots into $\mathbf{v}_i$ until $\mathbf{v}_i = \mathbf{v}$.
- 16: **end for**
- 17: Let r and s denote the number of control points in the u and v directions.
- 18: Set the internal control points/weights using:

$$\begin{bmatrix} w\mathbf{Q} \\ w \end{bmatrix}_{i,j} = \begin{bmatrix} w_1\mathbf{P}_1 + w_2\mathbf{P}_2 - w_3\mathbf{P}_3 \\ w_1 + w_2 - w_3 \end{bmatrix}_{i,j} \quad \text{for } i = 2, \dots, r-1 \quad j = 2, \dots, s-1 \quad (5.2.17)$$

- 19: Set the boundary control points/weights using:

$$\begin{bmatrix} w\mathbf{Q} \\ w \end{bmatrix}_{i,j} = \begin{bmatrix} w_1w_2(\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_3) \\ w_1w_2 \end{bmatrix}_{i,j} \quad \text{for } i = 1, r \quad j = 1, s \quad (5.2.18)$$

- 20: Construct NURBS surface:

$$\mathbf{S}(u, v) = \text{makeSurface}(p, 1, \mathbf{u}, \mathbf{v}, \{\mathbf{Q}_{i,j}\}, \{w_{i,j}\})$$

5.3 Curved Polygon Tessellation and Numerical Integration

Using the tools presented in the previous section we shall now detail how curved 2-dimensional polygons were represented and manipulated for the purpose of implementing the curved VEM. This section shall also explain how efficient numerical integration was performed on such polygons. We emphasize that the method of polygon subtessellation that we present in this section is purely for the purpose of numerical integration. We do not tessellate individual polygons to apply the VEM to the quadrilaterals.

5.3.1 Tessellation

In the context of the curved VEM, and standard FEM for that matter, a common operation is to loop over the elements of a mesh and integrate expressions involving the shape functions that exist locally on each element. When extracting these polygons from a mesh there are two cases to consider: the first is where the polygon is in the interior of the mesh, away from curved boundaries and away from curved internal interfaces, in which case all of its edges will be straight lines. The second case is where a polygon is incident on a curved boundary or internal interface, in which case one or more of its edges will be curved.

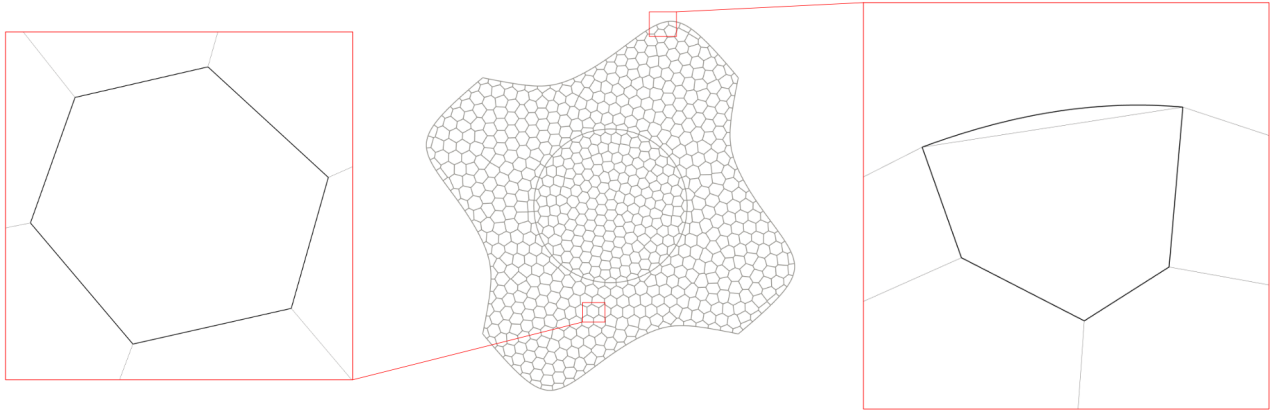


Figure 5.15: Example of a straight polygon (left) and a curved polygon (right), both extracted from a typical polygonal mesh used in the VEM.

In the case where all edges are straight lines, the polygon F is broken into a set of bilinear quadrilaterals (quads). The quad vertices consist of the centroid of the face, denoted $\mathbf{x}_F$, one of the vertices of the polygon $\mathbf{v}_i$ and the midpoints of the two edges that share that vertex, denoted $\mathbf{v}_{e_i}$ and $\mathbf{v}_{e_{i-1}}$. Each of these quads has the following NURBS representation:

$$\begin{aligned} p = 1, \quad q = 1, \quad \mathbf{u} = [0, 0, 1, 1], \quad \mathbf{v} = [0, 0, 1, 1] \\ \mathbf{w} = \{1, 1, 1, 1\}, \quad \left\{ \mathbf{P}_{1,1} = \mathbf{v}_i, \mathbf{P}_{2,1} = \mathbf{v}_{e_i}, \mathbf{P}_{1,2} = \mathbf{v}_{e_{i-1}}, \mathbf{P}_{2,2} = \mathbf{x}_f \right\}. \end{aligned} \quad (5.3.1)$$

An example is given in figure 5.16.

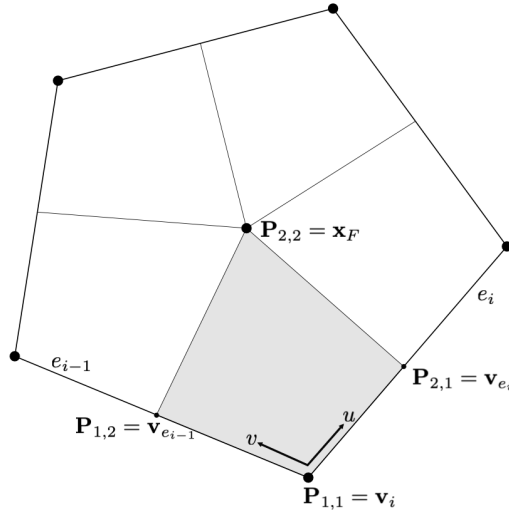


Figure 5.16: A quad-tessellation of a straight-sided polygon. The $u-v$ axis shows the direction of the increasing u and v parameters of the underlying NURBS surface.

The case where one or more of the polygon edges are curved is more involved. While the underlying principle is the same: splitting the polygon into quadrilaterals using the edge mid-points and the centroid, there are some additional steps involved in constructing curved quads. Consider the following polygon with two curved NURBS edges e_1 and e_5 :

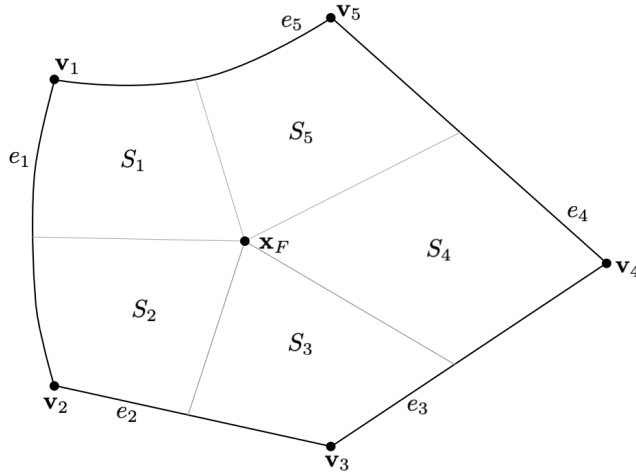


Figure 5.17: A quad tessellation of a curved polygon.

The first step is to perform knot insertion on each of the edges at $u = 0.5$ in order to split each of them into two parts which then may be used to construct each of the quads. The quads S_3 and S_4 have all their sides straight edges therefore they are constructed using the same bilinear quads from equation (5.3.1). Quads S_2 and S_5 have only one curved edge and are therefore constructed using a **ruled surfaces**. For example S_5 is constructed by splitting e_5 at $u = 0.5$ to give $\mathbf{C}_1(u)$, the second curve $\mathbf{C}_2(u)$ is a straight line between $\mathbf{x}_F$ and $\mathbf{x}_{e_4}$ (see figure 5.18). The ruled surface is then constructed using algorithm 3 with $\mathbf{C}_1(u)$ and $\mathbf{C}_2(u)$ as inputs.

The final quad to consider is S_1 which has two curved sides, therefore it must be constructed using a Coons surface with $\mathbf{C}_{u,1}$ set to the curved segment between $\mathbf{v}_1$ and $\mathbf{x}_{e_1}$, $\mathbf{C}_{u,2}$ is the

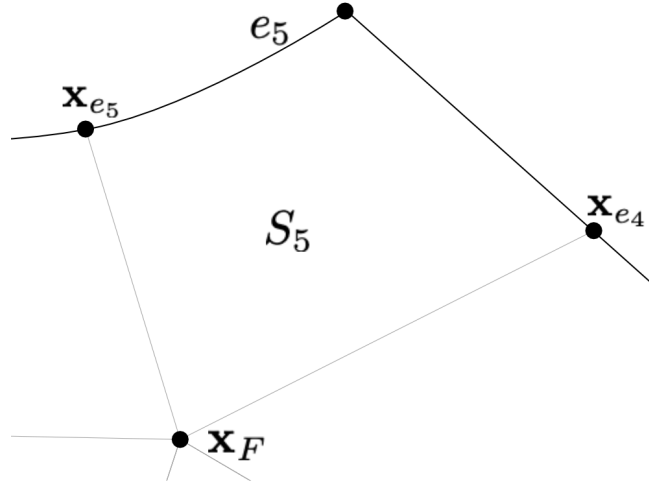


Figure 5.18: The ruled surface constructed to represent S_5 .

straight line between x_{e_5} and x_F , $C_{v,1}$ is set to the curve segment between v_1 and x_{e_5} and finally $C_{v,2}$ is the straight line between x_{e_1} and x_F (see figure 5.19).

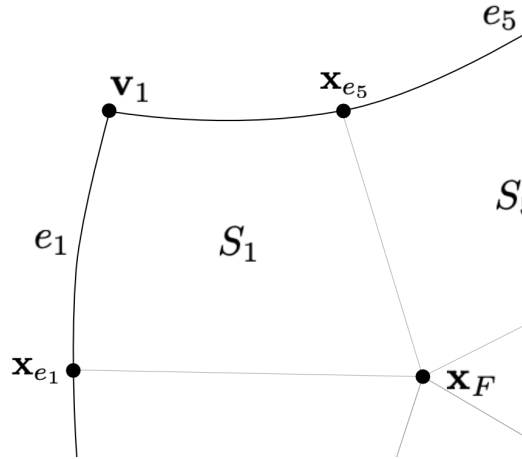


Figure 5.19: The ruled surface constructed to represent S_5 .

5.3.2 Integration

Accurate integration on curved polygons is accomplished using one of two methods: **Lasserre's Method** [242] for homogenous functions, and a **Subtessellation Method** for non-homogenous functions.

Lasserre's Method

Definition 5.3.1. Homogeneous function. A homogenous function $g(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ of degree q is any function that satisfies:

$$g(\lambda \mathbf{x}) = \lambda^q g(\mathbf{x}), \quad (5.3.2)$$

Monomials fall into this class of function therefore many computations in VEM are greatly

accelerated by Lasserre's Method. Consider a polygon $F \subset \mathbb{R}^2$ which has a boundary ∂F made up of a set of curved edges $\hat{\mathcal{E}}_F$ and straight edges $\mathcal{E}_F$. Using the terminology and notation of [3] each straight edge is a subset of a hyperplane in $\mathbb{R}^2$ defined as:

$$e_i \subset \mathcal{H}_i = \{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \cdot \mathbf{n}_i = b_i \}$$

where $\mathbf{n}_i$ is the outward unit-normal vector of edge e_i and b_i is the perpendicular distance of the hyperplane from the origin, see figure 5.20 for an example.

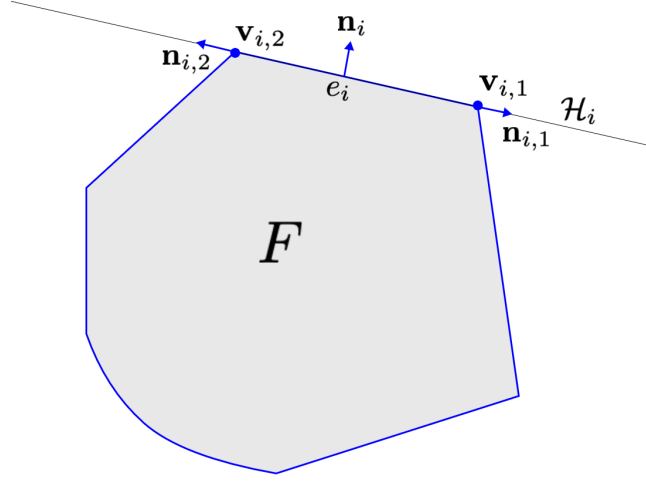


Figure 5.20: Polygon F with an example hyperplane $\mathcal{H}_i$. This is a recreation of ([3] figure 1)

Lasserre's Method makes use of two results from vector calculus: the first is Euler's homogeneous function theorem, which states that for a homogeneous function of degree q the following identity holds:

$$qg(\mathbf{x}) = \nabla g(\mathbf{x}) \cdot \mathbf{x}, \quad (5.3.3)$$

the second result is the generalised Stokes' Theorem which states for a vector field $\mathbf{f} : F \rightarrow \mathbb{R}^d$ that:

$$\int_F \nabla \cdot \mathbf{f} g(\mathbf{x}) dx + \int_F \mathbf{f} \cdot \nabla g(\mathbf{x}) dx = \int_{\partial F} \mathbf{f} \cdot \mathbf{n} g(\mathbf{x}) ds. \quad (5.3.4)$$

Choosing the vector field to be $\mathbf{f} = \mathbf{x}$ and substituting equation (5.3.3) into equation (5.3.4) yields Lasserre's method for homogenous functions:

$$\begin{aligned} \int_F g(\mathbf{x}) dx &= \frac{1}{2+q} \int_{\partial F} \mathbf{x} \cdot \mathbf{n} g(\mathbf{x}) ds \\ &= \frac{1}{2+q} \left[\sum_{e_i \in \hat{\mathcal{E}}_F} \int_{e_i} \mathbf{x} \cdot \mathbf{n} g(\mathbf{x}) ds + \sum_{e_i \in \mathcal{E}_F} b_i \int_{e_i} g(\mathbf{x}) ds \right] \end{aligned} \quad (5.3.5)$$

If e_i is curved then numerical integration is performed using Gauss-Legendre quadrature on the underlying NURBS curve, which assumed to be defined on $[0, 1]$. Denoting the NURBS curve parameterisation as $\mathbf{c}(t)$, the quadrature is computed as follows:

$$\int_{e_i} \mathbf{x} \cdot \mathbf{n} g(\mathbf{x}) ds = \int_0^1 \mathbf{c}(t) \cdot \mathbf{n}(t) g(\mathbf{c}(t)) J(t) dt$$

$$\approx \sum_{i=1}^{\text{nqp}} \mathbf{c}(t_i) \cdot \mathbf{n}(t_i) g(\mathbf{c}(t_i)) w_i J(t_i) \quad (5.3.6)$$

where t_i, w_i are the quadrature points and weights respectively and $\mathbf{n}(t)$ and $J(t)$ are the normal vector and Jacobian respectively. The full quadrature rule on a curve is computed by mapping a standard Gauss-Lobatto quadrature defined in $[-1, 1]$ to each non-zero length knot span $[u_i, u_{i+1}) \subset [0, 1]$. So, given a knot vector with M non-zero knot spans and a quadrature rule with m points then $\text{nqp} = m \times M$. This is a simplistic way to generate quadrature points for NURBS curves and more efficient methods exist [162] which take into account continuity of the B-spline basis. However, the vast majority of edges in the EG-mesh are straight lines so the additional computational cost of this quadrature is negligible.

If e_i is a straight line then we follow the technique introduced in [242], whereby Lasserre's Method is applied recursively to each straight edge in equation (5.3.5). Using the notation of figure 5.20, if we define $\mathbf{x}_{0,i} = \mathbf{v}_{i,1}$ to be the origin of $\mathcal{H}_i$ we have:

$$\int_{e_i} g(\mathbf{x}) dx = \frac{1}{(1+q)} \left[|e_i| g(\mathbf{v}_{i,2}) + \int_{e_i} \mathbf{v}_{i,1} \cdot \nabla g(\mathbf{x}) ds \right]. \quad (5.3.7)$$

The derivation may be found in the appendix; section D.1. If $g(\mathbf{x})$ is a polynomial then the term $\mathbf{v}_{i,1} \cdot \nabla g(\mathbf{x})$ is a polynomial of lower order. This property of polynomials lends itself to the design of a recursive algorithm for computing the integrals of families of monomials up to a specified order. In [3] one such recursive algorithm was presented and has been slightly modified here to allow for polygons with curved edges. The set of scaled monomials of order p on face F , denoted $\mathcal{M}_p(F)$, is used as a basis for the set of polynomials in the VEM. It is defined as:

$$\mathcal{M}_p(F) = \left\{ m_\alpha(\mathbf{x}) = \left(\frac{x - x_F}{h_F} \right)^{\alpha_x} \left(\frac{y - y_F}{h_F} \right)^{\alpha_y} : \alpha_x + \alpha_y \leq p \right\},$$

The dimension of $\mathbb{P}_p(P)$, which shall be denoted as $n_p = \dim \mathbb{P}_p(P)$ is given by:

$$n_p = \frac{(p+1)(p+2)}{2}.$$

We establish the following ordering between 1D index α and multi-index (α_x, α_y) :

$$1 \Leftrightarrow (0, 0), \quad 2 \Leftrightarrow (1, 0), \quad 3 \Leftrightarrow (0, 1), \quad 4 \Leftrightarrow (2, 0), \quad 5 \Leftrightarrow (1, 1), \quad 6 \Leftrightarrow (0, 2), \dots \quad (5.3.8)$$

This mapping shall be denoted as $\alpha(i, j)$. The reason for introducing Lasserre's Method is to compute the array:

$$I_p^F = \left[\int_F m_1 dx, \int_F m_2 dx, \dots, \int_F m_{n_p} dx \right]$$

as efficiently as possible and without simply resorting to numerical quadrature. Before detailing the algorithm, a quick review of properties of scaled monomials is needed.

Property 5.3.1. *Let $\hat{F} \subset \mathbb{R}^2$ be polygon as F translated by $-\mathbf{x}_F$. Therefore, $\mathbf{x}_{\hat{F}} = \mathbf{0}$. The following identity holds:*

$$\int_F m_i(\mathbf{x}) \, dx = \frac{1}{h_{\hat{F}}^{\alpha_x + \alpha_y}} \int_{\hat{F}} u_i(\mathbf{x}) \, dx \quad \text{where} \quad u_i(\mathbf{x}) = x^{\alpha_x} y^{\alpha_y} \quad (5.3.9)$$

Property 5.3.2. *The derivative of a monomial of order p is another monomial of order $p - 1$, recalling the single index to multi-index mapping of equation (5.3.8), $\alpha(i, j) : \mathbb{N}^2 \rightarrow \mathbb{N}$ we have:*

$$\frac{\partial}{\partial x} u_{\alpha(a,b)}(\mathbf{x}) = a u_{\alpha(a-1,b)}(\mathbf{x}) \quad \text{and} \quad \frac{\partial}{\partial y} u_{\alpha(a,b)}(\mathbf{x}) = b u_{\alpha(a,b-1)}(\mathbf{x})$$

Property 5.3.1 is crucial if one wishes to use Lassere's Method as the un-scaled monomials $u_i(\mathbf{x})$ are homogenous while the scaled monomials $m_i(\mathbf{x})$ are not. The procedure for efficiently computing all monomials on $\hat{F}$ up to order p is given in algorithms 5 and 6, which together cover the full procedure. Algorithm 5, which calculates the integrals of the scaled monomials $m_i(\mathbf{x})$ on a face F , must call a routine which computes the integrals of the (non-scaled) monomials $u_i(\mathbf{x})$ on a scaled face $\hat{F}$. The routine `integrateAll` is defined for faces, edges and vertices. The top of the recursion is the `integrateAll` for the face which calls `integrateAll` on each of its edges which in turn calls `integrateAll` on each of their vertices. Once the array of scaled monomial integrals has been computed, it may be stored and used to compute many important quantities in the VEM. Most notably the L^2 inner product $(m_i, m_j)_{0,F}$ which appears in many of the VE bilinear forms.

Algorithm 5 Algorithm for integrating all scaled monomials $m_i(\mathbf{x})$ up to order p

- 1: **Input:** Order p and a polygon F
 - 2: **Output:** Set of scaled monomial integrals on F , I_p^F .
 - 3: Translate F by $-\mathbf{x}_F$ to get $\hat{F}$
 - 4: $I_p^F \leftarrow \text{IntegrateAll}(p, \hat{F})$
 - 5: **for** $\alpha = 1$ to n_p **do**
 - 6: $q \leftarrow \text{order of } m_\alpha(\mathbf{x})$
 - 7: $I_p^F[\alpha] \leftarrow \frac{1}{h_F^q} I_p^F[\alpha]$ ▷ property (5.3.9)
 - 8: **end for**
-

Algorithm 6 Algorithm for integrating all monomials $u_i(\mathbf{x})$ up to order p on face F

```

1: procedure INTEGRATEALL( $p, F$ ) ▷ Lassere on face
2:    $I_p^F \leftarrow [0, \dots, 0]$  ▷ Initialise integrals to zero
3:   for  $e_i \in \partial F$  do ▷ For each edge
4:     if  $e_i$  is a linear then
5:        $I_p^{e_i} \leftarrow \text{IntegrateAll}(p, e_i)$  ▷ integrate all monomials on edge
6:     else
7:        $I_p^{e_i} \leftarrow [0, \dots, 0]$ 
8:       for  $\alpha = 1$  to  $n_p$  do
9:         Integrate by quadrature (application of equation (5.3.6)):

$$I_p^{e_i}[\alpha] \leftarrow \sum_{j=1}^{\text{nqp}} \mathbf{c}(t_j) \cdot \mathbf{n}(t_j) u_\alpha(\mathbf{c}(t_j)) w_j |J(t_j)|$$

10:      end for
11:    end if

12:    for  $\alpha = 1$  to  $n_p$  do ▷ Add contribution from edge  $e_i$  to face integral
13:      Let  $(\alpha_x, \alpha_y)$  be the exponents of  $u_\alpha(\mathbf{x})$ 
14:      Let  $q = \alpha_x + \alpha_y$  ▷ Monomial order
15:       $I_p^F[\alpha] \leftarrow I_p^F[\alpha] + \frac{1}{2+q} I_p^{e_i}[\alpha]$ 
16:    end for
17:  end for
18:  return  $I_p^F$ 
19: end procedure

20:
21: procedure INTEGRATEALL( $p, e$ ) ▷ Lassere on edge
22:    $I_p^e \leftarrow [0, \dots, 0]$ 
23:    $I_p^{\mathbf{v}_{i,2}} \leftarrow \text{IntegrateAll}(p, \mathbf{v}_{i,2})$  ▷ integrate on second vertex of edge
24:   for  $\alpha = 1$  to  $n_p$  do
25:     Let  $(\alpha_x, \alpha_y)$  be the exponents of  $u_\alpha(\mathbf{x})$ 
26:     Let  $q = \alpha_x + \alpha_y$  ▷ Monomial order
27:     Let  $\beta_1 = \alpha(\alpha_x - 1, \alpha_y)$  and  $\beta_2 = \alpha(\alpha_x, \alpha_y - 1)$ 
28:     Apply equation (5.3.7):

$$I_p^e[\alpha] = \frac{1}{1+q} \left[ |e| I_p^{\mathbf{v}_{i,2}}[\alpha] + (\mathbf{x}_{0,i})_1 \alpha_x I_p^e[\beta_1] + (\mathbf{x}_{0,i})_2 \alpha_y I_p^e[\beta_2] \right]$$

29:   end for
30:   return  $I_p^e$ 
31: end procedure
32:
33: procedure INTEGRATEALL( $p, \mathbf{v}$ ) ▷ Lassere on vertex
34:    $I_p^\mathbf{v} \leftarrow [0, \dots, 0]$ 
35:   for  $\alpha = 1$  to  $n_p$  do
36:      $I_p^\mathbf{v}[\alpha] = u_\alpha(\mathbf{v})$ 
37:   end for
38:   return  $I_p^\mathbf{v}$ 
39: end procedure

```

Subtessellation Method

If the integrand $f(\mathbf{x})$ is not homogeneous, i.e. does not satisfy equation (5.3.2), then the method of integration used in this thesis is the Subtessellation Method. Let us assume that polygon F has been decomposed into a set of (possibly curved) NURBS quads S_i using the methods of section 5.3.1

$$F = \bigcup_{i=1}^{N_e^F} S_i \quad \text{where} \quad S_i = \left\{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = \mathbf{s}_i(\mathbf{u}) \quad \forall \mathbf{u} \in [0, 1]^2 \right\}.$$

Then tensor-product Gauss-Legendre quadrature rules of any required order may be generated on $[0, 1]^2$ and then mapped onto each NURBS surface S_i . An example is given in figure 5.21.

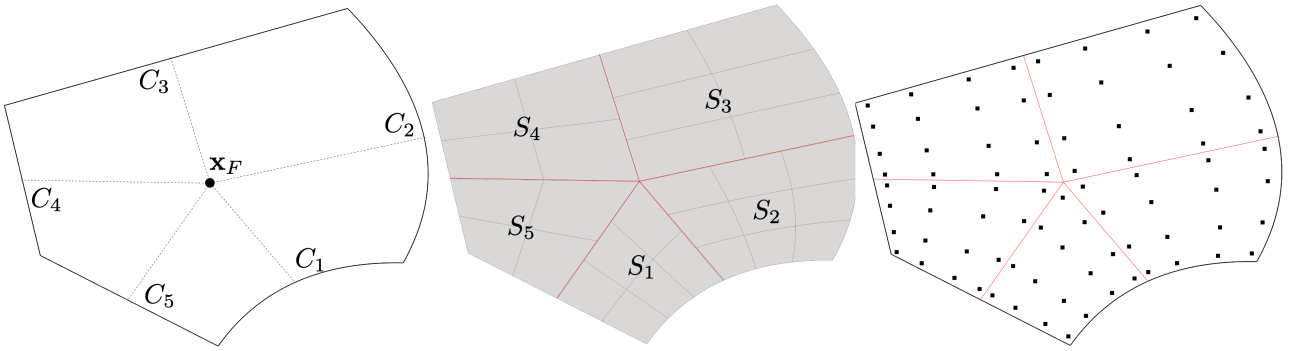


Figure 5.21: An example curved polygon (left), its NURBS tessellation (middle) and its quadrature points (right).

Let $\mathbf{x} = (x, y) \in S_i$ and $\mathbf{u} = (u, v) \in [0, 1]^2$, then the integral of $f(\mathbf{x})$ on S_i may be computed as follows:

$$\int_F f(\mathbf{x}) dx = \sum_{i=1}^{N_e^F} \int_{S_i} f(\mathbf{x}) dx = \sum_{i=1}^{N_e^F} \iint_{[0,1]^2} f(\mathbf{s}_i(u, v)) J_i(u, v) du dv$$

where $J_i(u, v)$ is the surface Jacobian given by:

$$J_i(u, v) = \det(\mathbf{J}_i(u, v)), \quad \text{where} \quad \mathbf{J}_i(u, v) = \begin{bmatrix} \frac{\partial \mathbf{s}_i(u, v)}{\partial u} & \frac{\partial \mathbf{s}_i(u, v)}{\partial v} \end{bmatrix}.$$

Let $\{u_l\}_{l=1}^{\text{nqp}_u}$ denote the nqp_u Gauss-Legendre points on $[0, 1]$ for the u surface parameter, let $\{v_m\}_{m=1}^{\text{nqp}_v}$ denote the nqp_v Gauss-Legendre points on $[0, 1]$ for the v surface parameter and let $w_{l,m} = w_l w_m$ denote the weight at point (u_l, v_m) then the integral over each NURBS surface of the sub-tessellation may be written as:

$$\iint_{[0,1]^2} f(\mathbf{s}_i(u, v)) J_i(u, v) du dv \approx \sum_{l=1}^{\text{nqp}_u} \sum_{m=1}^{\text{nqp}_v} f(\mathbf{s}_i(u_l, v_m)) w_{l,m} J(u_l, v_m)$$

The subtessellation method presented here is an alternative to the method of integration presented in [243], which does not require sub-tessellation but rather uses Green's integration

formula to convert an integral of function $f(\boldsymbol{x})$ over the polygon into an integral of the antiderivative of $F(\boldsymbol{x})$ over the boundary. The values of the antiderivative at the boundary quadrature points are then generated by integrating $f(\boldsymbol{x})$ along a line perpendicular to a "base line" and incident on the boundary quadrature point [243].

5.4 The Topology Data Structure

In a computer-aided geometric design context the storage of topological data entails the encoding of topological entities: 0-cells (vertices), 1-cells (edges), 2-cells (faces), 3-cells (cells) and even n-cells depending on the application. When choosing a format in which to store discrete topological data there is often a trade-off between low storage requirements and having fast access to adjacency and incidence relations between topological entities. Therefore, a careful consideration of the functional requirements of the application and the likely size of the final meshes is needed to make a good decision. An additional consideration is which topological entities need explicit representation for a given application. For example, in the VEM, vertices, edges and faces all have degrees of freedom attached to them. Therefore, having each of these entities explicitly represented in the mesh greatly eases many common operations: such as finding all the degrees of freedom coupled to a given degree of freedom, or enumerating the set of degrees of freedom attached to the closure of a face. In [244] a framework for analysing these considerations is presented and applied to examples of mesh data structures presented in the numerical analysis literature. Examples such as an edge-based analysis and refinement [245] and volumetric meshes with adaptive refinement [246].

The topology data structure developed as part of this geometric modelling kernel (GMK) is an implementation of the double-connected edge list (DCEL) [247, 248]. This is a well-known data structure in computer-aided geometric design (CAGD) and two well-known open-source projects based upon it already exist [249, 250]. However, a bespoke implementation, one which would be better suited to support NURBS and one where the underlying data structure is under my own control was desired. So, this section shall discuss our implementation of the DCEL for the purpose of implementing an exact-geometry VEM.

5.4.1 The Double-Connected Edge List

The DCEL is capable of representing any 2-manifold geometry. Vertices, edges and faces each have explicit representations in the DCEL making it an ideal topology description for the VEM, where degrees of freedom are attached to each of these entities. The basic principle of the DCEL is to split each edge going between vertices (v_1, v_2) into two half-edges, one being $v_1 \rightarrow v_2$ and the other $v_2 \rightarrow v_1$. This means that the orientation of edges and faces are explicitly encoded into the DCEL.

The DCEL is defined as a triple $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ where $\mathcal{V}$ is the set of vertex records, $\mathcal{E}$ is the set of half-edge records and $\mathcal{F}$ is the set of face records. Each vertex record $v \in \mathcal{V}$ stores two items [248]:

- A reference to an arbitrary outgoing half-edge, denoted `outHedge(v)`, which has v as its start point.
- A point in the plane $\mathbf{x} \in \mathbb{R}^2$ which represents the embedding of the vertex in $\mathbb{R}^2$, this is denoted by `point(v)`.

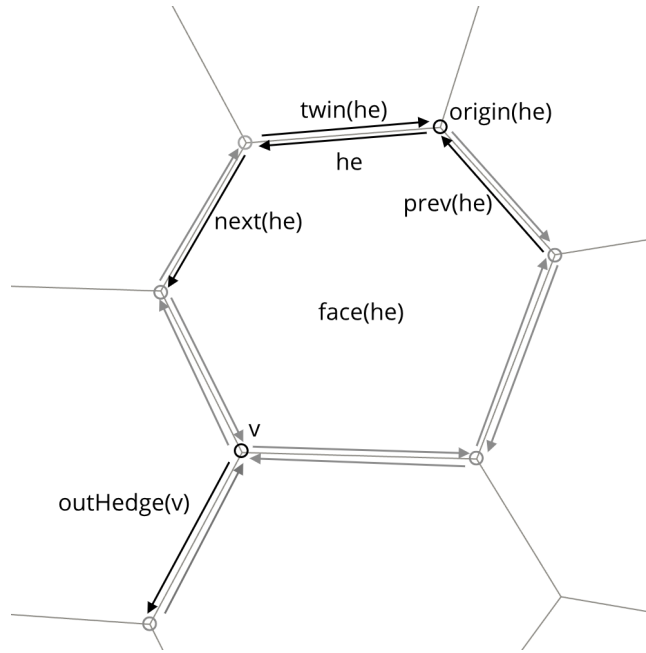


Figure 5.22: Illustration of the vertex record (labelled v) and half-edge record (labelled he).

Each half-edge record $he \in \mathcal{E}$ stores six items [248]:

- A reference to the twin half edge, which is the second half-edge of a pair when an edge is split. This is denoted $twin(he)$.
- A reference to the face f on which he is incident. Incidence in this case is defined as being a member of a ring of half edges which bounds face f . This is denoted $face(he)$
- A reference to the edge that comes directly after he in the ring bounding f , this is denoted $next(he)$.
- A reference to the edge that comes directly before he in the ring bounding f , this is denoted $prev(he)$.
- A reference to the vertex v which is the start point of he , this is denoted $origin(he)$.
- A curve which represents the embedding of the half edge in $\mathbb{R}^2$. This could be any continuous curve however, in this GMK we restrict it to be a NURBS curve in the plane. This is denoted $curve(he)$. Note, that the curve belonging to $twin(he)$ must be the exact same curve with the orientation reversed.

The face records $f \in \mathcal{F}$ are best explained using an example, consider the DCEL in figure 5.23: Each bounded face in the DCEL has **one** outer ring of half-edges that bound the face from the outside, for example face f_2 is bounded by $\{he_{10}, he_{12}\}$. Additionally, each bounded face may have an arbitrary number of rings that bound the face from the inside, in the case of f_1 these two rings are $\{he_9, he_{11}\}$ and $\{he_{13}, he_{21}, he_{19}, he_{17}\}$. A ring bounding a face will always be oriented such that the face will be to its left (with respect to the tangent vector). Therefore, outer rings run counter clockwise and inner rings will run clockwise. The unbounded face, which in this GMK will always be defined as f_0 , is a face used to delineate the boundary

of the DCEL. This face stretches into infinity in all directions and is also the face bounded by any rings that form holes in the DCEL. In this GMK all of the geometries will be assumed to be connected, therefore the unbounded face has exactly one inner ring that bounds it from the inside, in this case it is $\{he_1, he_7, he_5, he_3\}$. The number of outer rings belonging to f_0 is equal to the number of holes in the DCEL, in this case there is just one hole and it is bounded by the ring $\{he_{14}, he_{16}, he_{18}, he_{20}\}$.

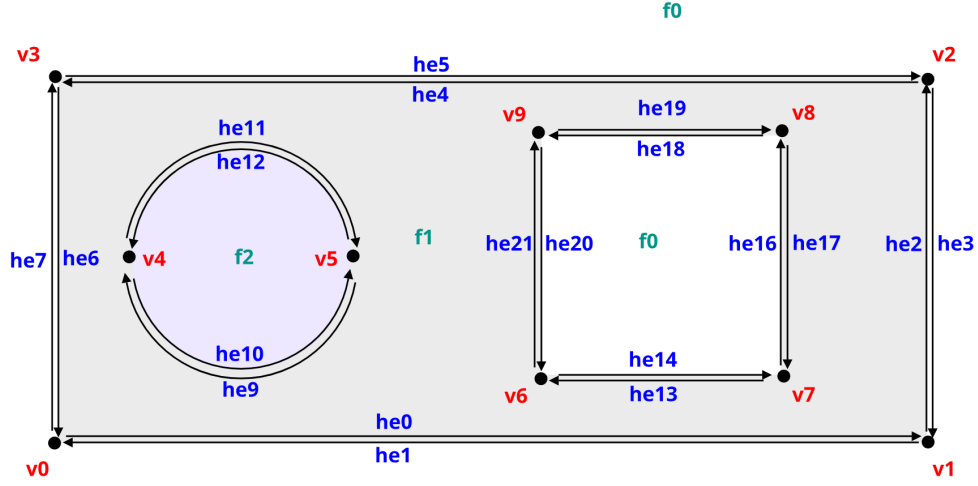


Figure 5.23: An example DCEL

To traverse the rings that bound it, each face need only store a reference to an arbitrary half-edge in each ring. The `next()` and `prev()` pointers may then be followed to traverse each ring. Each face record $f \in \mathcal{F}$ therefore stores the following [248]:

- An array of outer-ring half edges, for bounded faces this will have just one half-edge which is any half-edge in the outer ring that bounds f . For the unbounded face this array will have one half-edge for every hole in the domain. This array shall be denoted `outerHedges(f)`.
- An array of inner-ring half edges, for bounded faces this may have any number of half-edges in it. For the unbounded face f_0 this will have just one half-edge. This array shall be denoted `innerHedges(f)`.

For a given DCEL the face records are not unique, there would be many ways to describe the same DCEL. In the case of figure 5.23, a possible set of values for the face records could be:

Face	outerHedges(f)	innerHedges(f)
f_0	he_{14}	he_5
f_1	he_0	$\{he_{11}, he_{13}\}$
f_2	he_{12}	$\emptyset$

5.4.2 The Euler Operators

If the DCEL is in a valid state, with validity defined as being a 2-manifold, then the DCEL will satisfy the Euler-Poincarè formula:

$$V - E + F = 2(S - H) + R \quad (5.4.1)$$

where V , E , F , S , H and R are the number of vertices, edges, faces, shells, holes and rings respectively. In $\mathbb{R}^2$ a shell is simply a connected component of the DCEL. In this GMK all geometries will consist of one connected component so $S = 1$. In the case of figure 5.23 we have $V = 10$, $E = 10$, $F = 3$, $H = 1$ and $R = 3$. Plugging these values into equation (5.4.1) gives $3 = 3$, therefore the DCEL of figure 5.23 is a valid 2-manifold geometry.

In this GMK, Euler operators [251, 252, 253] are used to construct geometry. These operators add or remove vertices, edges, faces, shells, holes and rings in such a way as to preserve the Euler-Poincarè formula at every stage [251]. The Euler operators serve as manipulation primitives which may be chained to form complex and intuitive geometric operations which are guaranteed to ensure that the geometry remains a valid 2-manifold. Although the terms of equation (5.4.1) may only have positive integer values: if one were to briefly consider (V, E, D, S, H, R) to be a vector in $\mathbb{R}^6$, then the Euler-Poincarè formula may be thought of as a hyperplane with normal vector $(1, -1, 1, -2, 2, -1)$. Under this interpretation, the Euler operators may be thought of as a set of 5 linearly independent vectors in $\mathbb{R}^6$, with integer values, that span the hyperplane [252]. There are many valid choices for this spanning set, however the literature on Euler operators has converged to the set given in table 5.1 [252].

Table 5.1: The set of 5 Euler operators and their inverses.

Operator	Meaning	V	E	F	S	H	R
mvsf	make vertex shell face	+1	0	+1	+1	0	0
kvsf	kill vertex shell face	-1	0	-1	-1	0	0
mev	make edge, vertex	+1	+1	0	0	0	0
kev	kill edge, vertex	-1	-1	0	0	0	0
mef	make edge, face	0	1	1	0	0	0
kef	kill edge, face	0	-1	-1	0	0	0
mekr	make edge, kill ring	0	+1	0	0	0	-1
kemr	kill edge, make ring	0	-1	0	0	0	+1
mfkrh	make face, kill ring, hole	0	0	+1	0	-1	-1
kfmrh	kill face, make ring, hole	0	0	-1	0	+1	+1

To demonstrate the use of Euler operators, figure 5.25 in shows the construction of a square with a hole inside of it. This is probably one of the simplest geometries that uses all five operators. The Euler operators are a good system for constructing a 2-manifold geometry for an additional reason: they do not presuppose or require any particular data structure. An application designer must create the implementations of the Euler operators for their particular case. In the case of the DCEL the implementations come naturally, as we shall demonstrate

with some examples. But first, we shall present two algorithms for querying adjacency and incidence information which will be reused later. In particular we present the algorithms for finding all of the half-edges bounding a face (the external ring) and for finding all of the outgoing half-edges from a vertex.

Algorithm 7 Returns all of the edge bounding a face from the outside.

```

1: procedure OUTERHEDGES( $D, f$ )
2:   Inputs: A DCEL  $D$  and a face  $f$ .
3:   Returns: A list  $L$  of all half-edges bounding  $f$ 
4:    $L \leftarrow []$ 
5:    $he_1 \leftarrow \text{outerHedges}(f)[0]$ 
6:    $he_2 \leftarrow he_1$ 
7:   do
8:      $\text{append}(L, he_2)$ 
9:      $he_2 \leftarrow \text{next}(he_2)$ 
10:  while  $he_2 \neq he_1$ 
11: end procedure

```

If one wishes to return all of the half edges bounding a face from an internal ring, for example face f_1 in figure 5.23 has two rings bounding it from the inside: $\{he_{11}, he_9\}$ and $\{he_{19}, he_{17}, he_{13}, he_{21}\}$, then algorithm 7 only needs to be slightly altered with line 5 being changed to: $he_1 \leftarrow \text{innerHedges}(f)[r]$, where r is the number of the ring.

Algorithm 8 Returns all of the outgoing half-edges from a vertex

```

1: procedure OUTHEDGES( $D, v$ )
2:   Inputs: A DCEL  $D$  and a vertex  $v$ .
3:   Outputs: A list  $L$  of all outgoing half-edges from  $v$ .
4:    $L \leftarrow []$ 
5:    $he_1 \leftarrow \text{outHedge}(v)$ 
6:    $he_{\text{out}} \leftarrow he_1$ 
7:   do
8:      $\text{append}(L, he_{\text{out}})$ 
9:      $he_{\text{in}} \leftarrow \text{twin}(he_{\text{out}})$ 
10:     $he_{\text{out}} \leftarrow \text{next}(he_{\text{in}})$ 
11:  while  $he_{\text{out}} \neq he_1$ 
12: end procedure

```

Algorithm 8 can be easily altered to return all of the incoming half edges or to return all of the vertices connected to vertex v . The actions of both these algorithms is illustrated in figure 5.24.

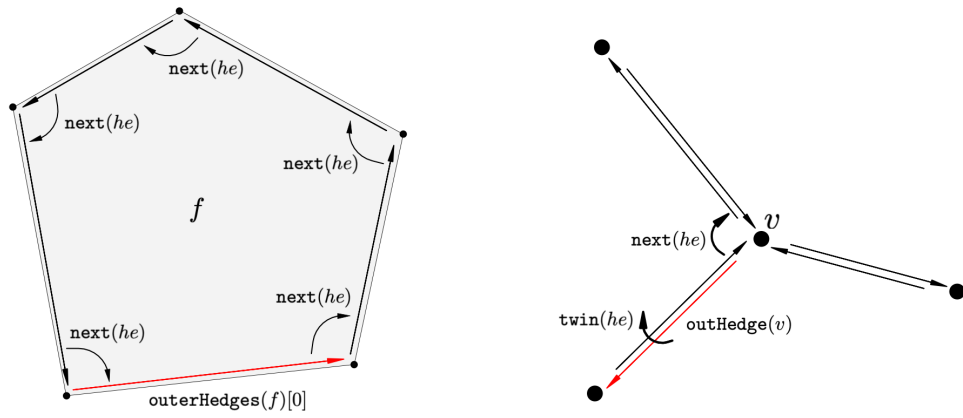


Figure 5.24: Illustration of algorithms 7 (left) and 8 (right).

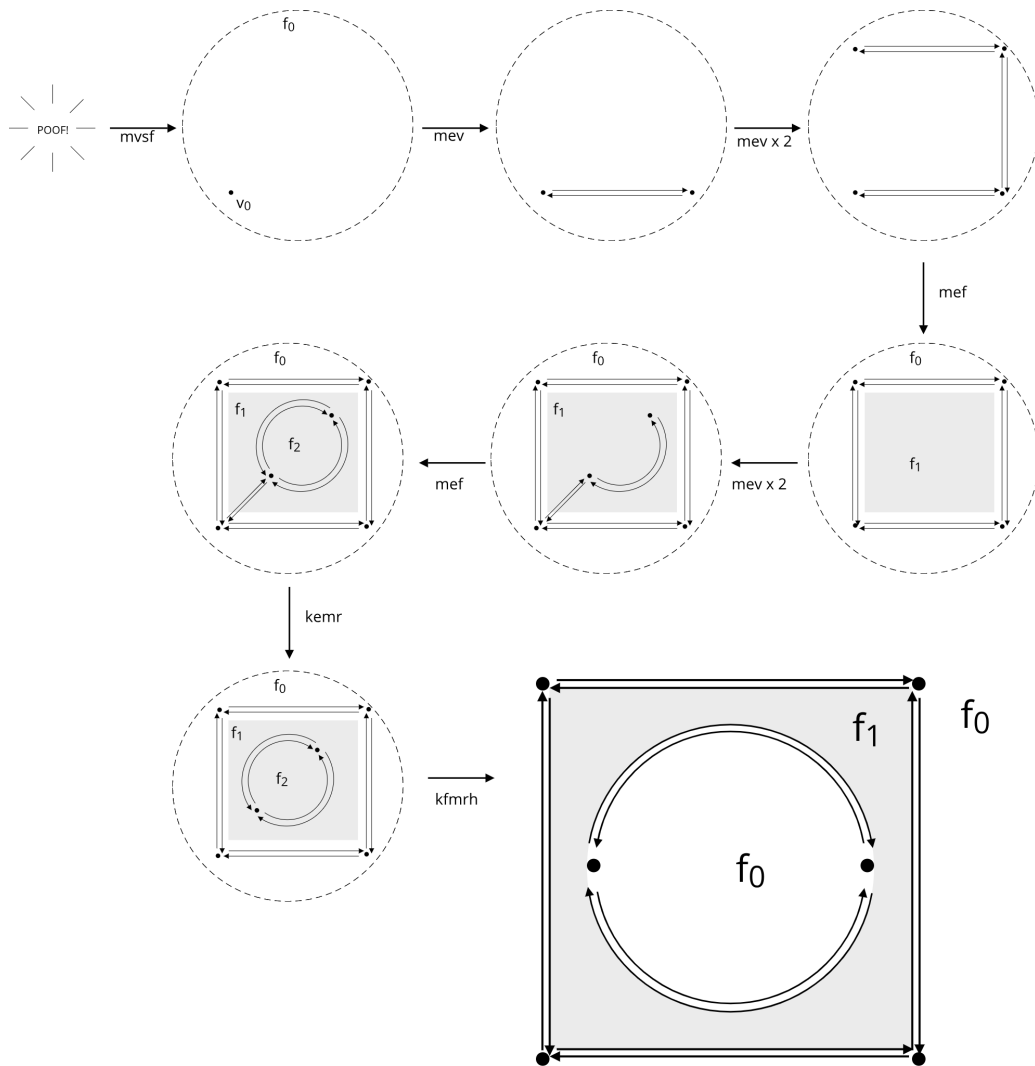


Figure 5.25: The series of Euler operators used to create a DCEL representing a square with a hole in it.

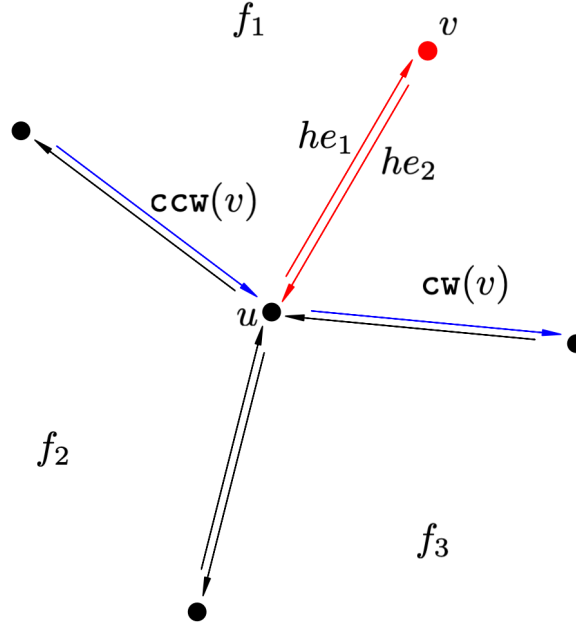


Figure 5.26: Relevant data in an `mev` operation.

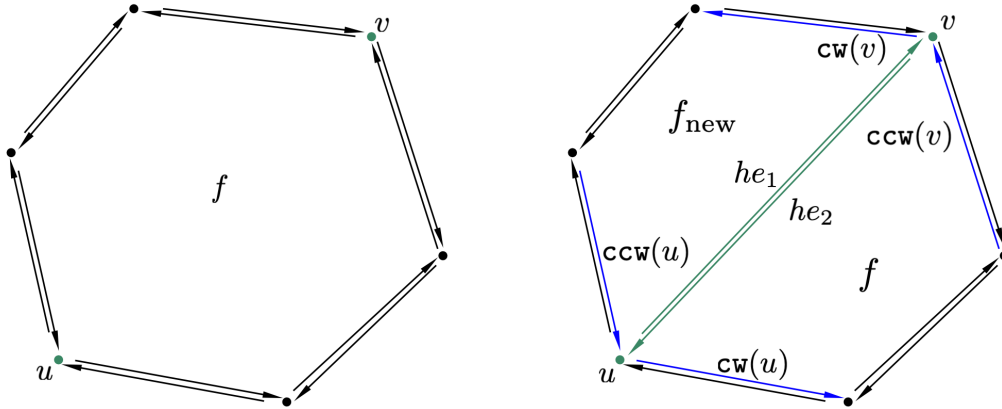


Figure 5.27: The `mef` operation performed on face f .

We will now discuss how two of the Euler operators, make-edge-vertex (`mev`) and make-edge-face (`mef`) are implemented for the DCEL. Beginning with the `mev` operation; consider the situation of figure 5.26, where vertex v and half-edges he_1, he_2 (shown in red) are to be connected to existing vertex u . The half-edge $ccw(v)$ is defined as the first incoming half-edge to u which is counter-clockwise to he_1 . The half-edge $cw(v)$ is the first outgoing half-edge from u which is clockwise to he_1 , both are shown in blue. The algorithm to compute both is simple, first the outgoing edges from u are computed using algorithm 8 and are sorted with respect to their angle relative to the positive x -axis. Next, the angle of he_1 is computed and the two nearest (with respect to angle) outgoing half-edges are stored, one will be clockwise to he_1 and the second counter-clockwise. Next, $ccw(v)$ is set to the twin of the counter-clockwise half-edge and $cw(v)$ is set to the clockwise half-edge. With the notation explained we may now present

the **mev** operation, given in algorithm 9.

Algorithm 9 Performs the **mev** operation.

```

1: procedure MEV( $D, u, v$ )
2:   Inputs: A DCEL  $D$ , an existing vertex  $u$  and a new vertex  $v$ .
3:   Outputs: The altered DCEL  $D$ .
4:   Add  $v$  and  $he_1, he_2$  to  $D$ .
5:                                                                 ▷ Set vertex fields
6:   outHedge( $v$ )  $\leftarrow he_2$ 
7:                                                                 ▷ Set  $he_1$  fields
8:   twin( $he_1$ )  $\leftarrow he_2$ 
9:   face( $he_1$ )  $\leftarrow f_1$ 
10:  next( $he_1$ )  $\leftarrow he_2$ 
11:  prev( $he_1$ )  $\leftarrow ccw(v)$ 
12:  next( $ccw(v)$ )  $\leftarrow he_1$ 
13:  origin( $he_1$ )  $\leftarrow u$ 
14:                                                                 ▷ Set  $he_2$  fields
15:  twin( $he_2$ )  $\leftarrow he_1$ 
16:  face( $he_2$ )  $\leftarrow f_1$ 
17:  next( $he_2$ )  $\leftarrow cw(v)$ 
18:  prev( $cw(v)$ )  $\leftarrow he_2$ 
19:  prev( $he_2$ )  $\leftarrow he_1$ 
20:  origin( $he_2$ )  $\leftarrow v$ 
21: end procedure

```

Next, consider the situation in figure 5.27 where an **mef** operation is creating a new pair of half-edges between two existing vertices u and v , with face f being split into faces f and f_{new} in the process. The original face f is kept in the DCEL but has its field `outerHedges(f)[0]` set to the new half edge he_2 . This ensures that it does not point to a half-edge that it no longer owns. The half-edges $ccw(u)$ and $cw(u)$ are counter-clockwise and clockwise with respect to he_1 while $ccw(v)$ and $cw(v)$ are counter-clockwise and clockwise with respect to he_2 , these are shown in blue in figure 5.26. The **mef** operation is given in algorithm 10; this version assumes that f does not have any internal rings. If this is not the case then additional steps must be performed to ensure that 1) the new half-edges do not intersect the internal rings and 2) to determine if the internal rings belong to f_{new} or to f after the **mef** operation is completed. The method for doing so shall be omitted here for brevity but may be found in [248], section 2.3.

Algorithm 10 Performs the **mef** operation.

```

1: procedure MEF( $D, u, v, f$ )
2:   Inputs: A DCEL  $D$ , an existing vertices  $u$  and  $v$  and existing face  $f$ .
3:   Outputs The altered DCEL  $D$ .
4:   Add  $he_1, he_2$  and  $f_2$  to  $D$ .
5:                                     ▷ Set the fields for  $he_1$ 
6:    $\text{origin}(he_1) = u$ 
7:    $\text{face}(he_1) = f_{\text{new}}$ 
8:    $\text{twin}(he_1) = he_2$ 
9:    $\text{next}(he_1) = \text{cw}(v)$ 
10:   $\text{prev}(\text{cw}(v)) = he_1$ 
11:   $\text{prev}(he_1) = \text{ccw}(u)$ 
12:   $\text{next}(\text{ccw}(u)) = he_1$ 
13:                                     ▷ Set the fields for  $he_2$ 
14:   $\text{origin}(he_2) = v$ 
15:   $\text{face}(he_2) = f$ 
16:   $\text{twin}(he_2) = he_1$ 
17:   $\text{next}(he_2) = \text{cw}(u)$ 
18:   $\text{prev}(\text{cw}(u)) = he_2$ 
19:   $\text{prev}(he_2) = \text{ccw}(v)$ 
20:   $\text{next}(\text{ccw}(v)) = he_2$ 
21:                                     ▷ Set the fields for  $f$ 
22:   $\text{outerHedges}(f) = [he_2]$ 
23:                                     ▷ Set the fields for  $f_{\text{new}}$ 
24:   $\text{outerHedges}(f) = [he_1]$ 
25: end procedure

```

5.5 Mesh Generation

The curved VEM is able to construct local approximation spaces on arbitrary n-sided polygons with curved edges, subject only to minimal limitations on polygon shape. To fully realise the benefit of this capability the need arose to design a polygonal mesh generator which could also preserve the initial geometry of the domain. Most of the publicly available mesh generators were designed to serve the scientific community using the FEM. Consequently, most are focussed on producing high-quality triangular/quadrilateral meshes in 2D and tetrahedral/hexahedral in 3D. Some notable examples are Gmsh [254], Triangle [255], TetGen [256]. Unfortunately, to the best of our knowledge, there are currently no publicly available mesh generators for curved, polygonal meshes which are of the same level of development and adoption as Gmsh or TetGen. While there are some proprietary tools which can produce polygonal/polyhedral meshes, such as STAR-CCM, the somewhat esoteric requirements of the curved VEM meant that a bespoke mesh generator was required.

The mesh generator developed as part of this project is a Centroidal Voronoi Diagram (CVT) based mesh generator. It draws upon ideas presented in [257, 258] and has three components:

1. A module which distributes seed points on the domain. Special care is taken to ensure seeds on boundary curves and internal interfaces are properly spaced to ensure a good final result.
2. A smoothing module which takes the initial seed distribution and minimises an energy functional to compute a CVT. Again, the boundary and internal interface seeds must be treated differently to volume seeds to ensure the final CVT conforms to the initial geometry.
3. A clipping module which intersects a CVT with the initial domain to give a final mesh.

The modular nature of the design means that different implementations may be used so long as they conform to the code interface for each module. Viewed at a high level, the mesh generation algorithm is quite simple as most of the complexity resides in the three modules.

Many of the sub-problems which arise in the implementation of this mesh generator involve the solution of a small, possibly constrained nonlinear optimisation problem. Unless otherwise stated all such problems were solved using the NLOpt C++ library [259].

The specifications of the mesh generator are as follows:

1. It must take as input arguments a DCEL representing a domain, $\mathcal{D}_\Omega = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, and a target number of faces for the final mesh, N_{init} .
2. The output will be another DCEL, $\mathcal{D}_h = (\mathcal{V}_h, \mathcal{E}_h, \mathcal{F}_h)$, which is an exact-geometry tessellation of $\mathcal{D}_\Omega$. Both the boundary and the internal interfaces of $\mathcal{D}_\Omega$ must be preserved exactly.
3. The faces of $\mathcal{D}_\Omega$ will have block IDs mapped to them and the boundary edges of $\mathcal{D}_\Omega$ will

have bound IDs mapped to them. These must be preserved in the final mesh. For faces this means that if face $f \in \mathcal{F}$ has block ID b then all faces $f_h \in \mathcal{F}_h$ such that $f_h \subset f$ are also mapped to block ID b . For half-edges this means that if $e \in \mathcal{E}$ has bound ID b then all half edges $e_h \in \mathcal{E}_h$ such that $e_h \subset e$ are also mapped to bound ID b .

If the mesh generator meets these specifications then it contains all the information required to implement a curved VEM in 2D. The domain of figure 5.28 shall be used as an example to discuss the salient features of the mesh generator.

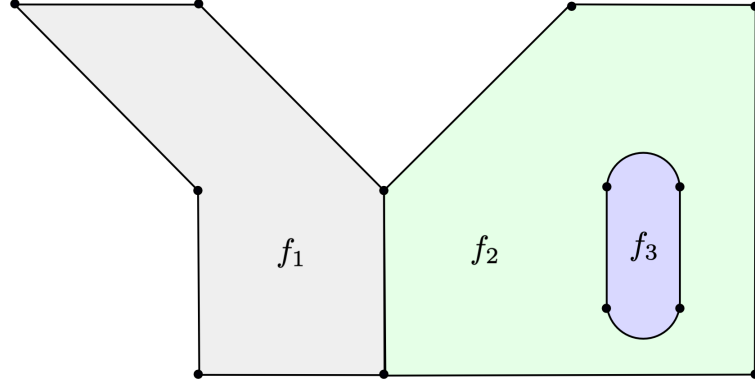


Figure 5.28: Initial geometry $\mathcal{D}_\Omega$.

Before we dive into the details of the mesh generator, we shall briefly introduce the CVT.

5.5.1 A Review of the Centroidal Voronoi Tessellation

The Voronoi Tessellation

The Voronoi Tessellation (VT) is fundamental entity in the natural sciences, mathematics and computational geometry. It has a long history, with some of the earliest known work published by Ren  Descartes in the 17th century. Many excellent texts exist on the subject, in particular we point to [260, 261, 262, 263]. Therefore, we shall only provide a minimal review here needed to put the mesh generator into context.

We shall restrict our discussion of VTs to $\mathbb{R}^2$ under the Euclidean distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$. Generalisations of the VT exist in higher dimensions and with different metrics however these are not required in the present discussion.

Let $S \subset \mathbb{R}^2$ be a set of discrete points called *seed points* and let $\mathbf{s} \in S$ be a particular seed point. The **Voronoi region** corresponding to $\mathbf{s}$, $R(\mathbf{s}, S)$, is the set of points in $\mathbb{R}^2$ closer to $\mathbf{s}$ than to any other point in S . Or more formally:

$$R(\mathbf{s}, S) = \left\{ \mathbf{x} \in \mathbb{R}^2 : \|\mathbf{s} - \mathbf{x}\| \leq \|\mathbf{q} - \mathbf{x}\| \quad \forall \mathbf{q} \in S - \{\mathbf{s}\} \right\}$$

We give an example in figure 5.29. The edges which separate adjacent Voronoi regions are called

Voronoi edges and the points of intersection between Voronoi edges are called *Voronoi vertices*. The set of seed points may be partitioned into seeds which lie on the boundary of the convex hull and those which lie on the interior of the convex hull. The regions which correspond to boundary seeds are unbounded while those on the interior are both bounded and convex [260].

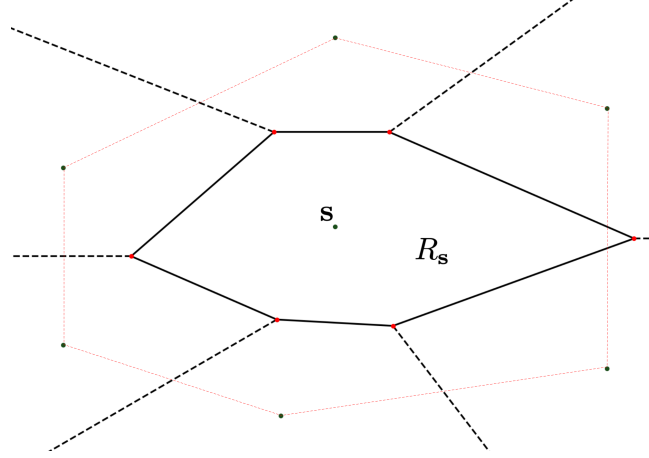


Figure 5.29: An example of a Voronoi Tessellation. The seeds are shown in dark green, the Voronoi vertices are shown in red and the Voronoi edges are shown in black, with bounded edges marked with a solid line and infinite edges shown with dashed lines. The convex hull of the seeds are shown in dashed red.

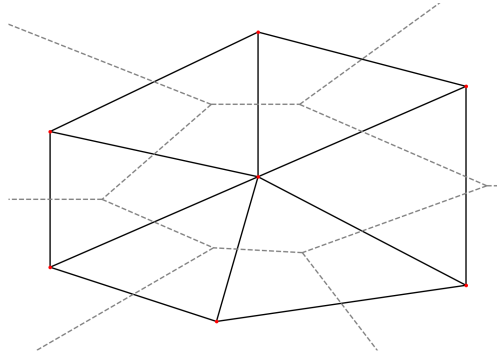


Figure 5.30: The Delaunay triangulation is shown in black while the VT is shown in dashed grey.

The **Delaunay Triangulation** (DT) is the dual graph of the VT, it may be constructed by setting the seeds to be the Delaunay vertices and connecting with a Delaunay edge any pair of seeds whose corresponding Voronoi regions are edge-adjacent. The DT corresponding to figure 5.29 is shown in figure 5.30.

The Centroidal Voronoi Tessellation

Let $\Omega \subset \mathbb{R}^2$ be some domain of interest and let $S = \{\mathbf{s}_i : i = 1, \dots, N_S\}$ be a set of seed points inside Ω . We shall denote the seed vector with $\mathbf{P} = [\mathbf{s}_1, \dots, \mathbf{s}_{N_S}] \in \mathbb{R}^{2N_S}$ and will use it interchangeably with S depending on the context. We denote the clipped Voronoi Tessellation

of Ω with:

$$VT(S, \Omega) = \{R_s \cap \Omega : \forall s \in S\}$$

The clipped centroidal Voronoi tessellation is a special case of the clipped VT where, for each Voronoi region R_s , the centroid $\mathbf{c}_s$ and the seed point $\mathbf{s}$ coincide [263]:

$$\mathbf{c}_s = \mathbf{s} \quad \text{where} \quad \mathbf{c}_s = \frac{\int_{R_s \cap \Omega} \mathbf{x} \, dx}{\int_{R_s \cap \Omega} 1 \, dx}.$$

One method to compute a clipped CVT is to minimise an energy functional which penalises the distance between a seed and the centroid of its region:

$$E(\mathbf{P}, \Omega) = \sum_{\mathbf{s} \in \mathbf{P}} \int_{R_s \cap \Omega} \|\mathbf{x} - \mathbf{s}\|^2 \, dx \quad (5.5.1)$$

The gradient of the energy functional with respect to a seed $\mathbf{s} \in S$ is given by:

$$\nabla_{\mathbf{s}} E(\mathbf{P}, \Omega) = 2A_s (\mathbf{s} - \mathbf{c}_s) \quad \text{where} \quad A_s = \int_{R_s \cap \Omega} 1 \, dx$$

One of the oldest and simplest algorithms for minimising $E(\mathbf{P}, \Omega)$ is Lloyd's Method [263], which is a fixed-point iteration given by the map $L : \mathbb{R}^{2N_s} \rightarrow \mathbb{R}^{2N_s}$ operating on the seed vector $\mathbf{P}$

$$L(\mathbf{P}) = (L_s(\mathbf{P}))_{\mathbf{s} \in \mathbf{P}} \quad \text{where} \quad L_s(\mathbf{P}) = \frac{\int_{R_s \cap \Omega} \mathbf{x} \, dx}{\int_{R_s \cap \Omega} 1 \, dx}. \quad (5.5.2)$$

The CVT is the fixed point of the Lloyd map, which is defined as the seed vector $\mathbf{P}^*$ which satisfies $\mathbf{P}^* = L(\mathbf{P}^*)$. The iteration takes an initial seed point distribution $\mathbf{P}^{(0)}$ as an input and the iteration proceeds as follows: $\mathbf{P}^{(k+1)} = L(\mathbf{P}^{(k)})$ until some convergence criteria is reached. In [257] the following non-dimensional error parameter was derived:

$$e(\mathbf{P}, \Omega) = \frac{N_s}{|\Omega|^{3/2}} \|\nabla E(\mathbf{P}, \Omega)\| = \frac{N_s}{|\Omega|^{3/2}} \left(\sum_{\mathbf{s} \in \mathbf{P}} A_s^2 \|\mathbf{s} - \mathbf{c}_s\|^2 \right)^{1/2} \quad (5.5.3)$$

This error parameter ensures that, for a given tolerance ϵ , the convergence criterion $e(\mathbf{P}, \Omega) < \epsilon$ yields a mesh of comparable smoothness independent of the size of the domain Ω or the number of seeds N_s . The convergence properties of Lloyd's Method were investigated in [264]. An example of Lloyd's iteration is shown in figure 5.31 for a unit square geometry.

It was shown in [265] that the energy functional of (5.5.1) is C^2 continuous when the domain Ω is convex and C^1 continuous otherwise. Therefore, unconstrained descent algorithms are a reasonable alternative to Lloyd's method for finding the a local minimum, given an initial distribution of seeds. The use of quasi-Newton methods to compute the CVT have been studied in [265, 266], where the authors presented preconditioned L-BFGS solvers for the CVT. Additionally, in [266] a preconditioned nonlinear conjugate gradient was also studied for the case where Ω is non-convex and only C^1 continuity may be assumed. All of these descent methods were shown to have better rates of convergence than Lloyd's method, albeit at the cost of higher

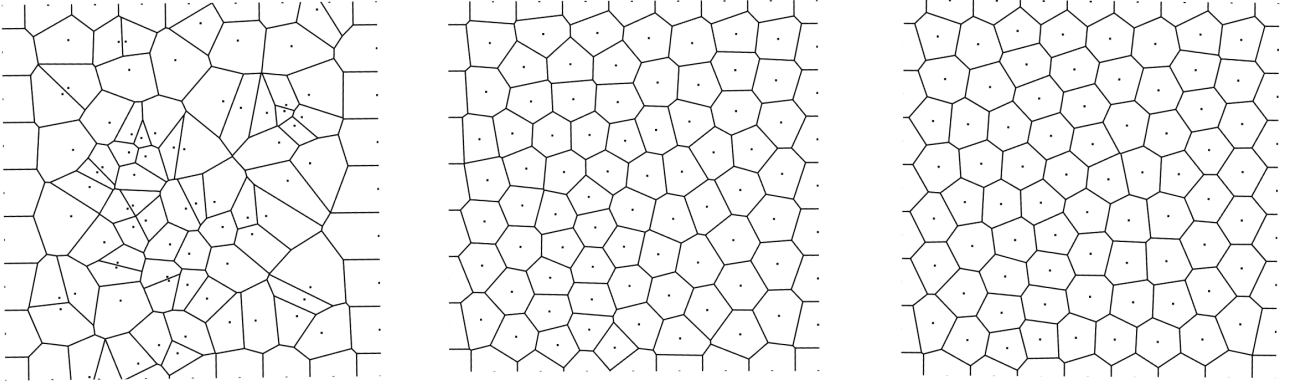


Figure 5.31: A clipped Voronoi tessellation after 0, 20 and 60 Lloyd's iterations (left to right).

complexity.

5.5.2 Initial Seed Point Distribution

The aims of the initial seed placement is to have:

1. A set of seeds which are uniformly distributed in the arc length along each of the curves of the input geometry, both boundary and internal. These seeds remain fixed throughout the remainder of the mesh generation algorithm and shall be referred to as **boundary seeds**.
2. A set of seeds which are randomly and evenly distributed on the interior of the input geometry. These seeds are left free to move and shall be referred to as **volume seeds**.

Generation of Boundary Seeds

Consider the problem of distributing N points evenly along a curve with respect to arc length, such as that shown in figure 5.32: We solve this problem using a stepping algorithm. The arc length between parameters u_1, u_2 is given by:

$$L(u_1, u_2) = \int_{u_1}^{u_2} J(t) dt \approx \sum_{i=1}^{\text{nqp}} J(t_i) w_i$$

with the total arc length denoted $L = L(0, 1)$ and the number of quadrature points (nqp) chosen to accurately compute the integral. The length of each segment is therefore given by $\bar{L} = L/(N - 1)$. We begin at $u = 0$ and compute the parameter u^* which minimises the following objective function:

$$f(t) = (L(0, t) - \bar{L})^2, \quad \text{where} \quad \frac{df}{dt} = 2(L(0, t) - \bar{L})J(t)$$

subject to the constraint that $t \in [0, 1]$. Once the parameter $u^* = \arg \min f(t)$ has been computed we set the lower bound of t to u^* and repeat the process until we have a set of

parameter values $[0, u_1, \dots, u_{N-2}, 1]$ which are equidistant with respect to their arc lengths.

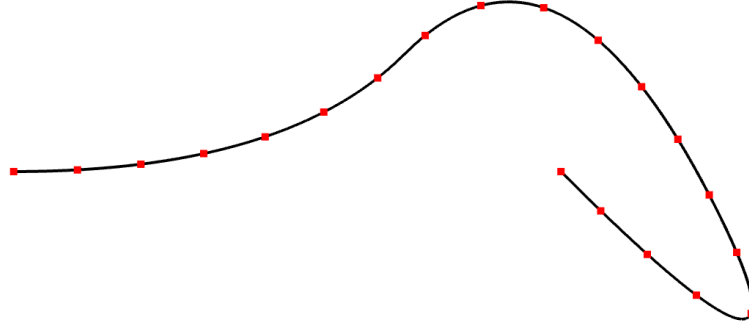


Figure 5.32: A B-spline curve with a set of points distributed at equal arc-lengths.

We shall denote the DCEL representation of the domain Ω with $\mathcal{D}_\Omega$. To generate the set of seeds on the boundary of the input domain $\mathcal{D}_\Omega$, first the number of segments required on each edge is computed using the average element diameter $\alpha = \sqrt{|\mathcal{D}_\Omega|/N_{init}}$ where $|\mathcal{D}_\Omega|$ is the area of the input domain. Each edge in $\mathcal{D}_\Omega$ will have an arc length L associated with it and the number of segments on each edge is given by $N = \text{ceil}(\beta(L/\alpha))$ where β is a user-selected parameter, $\beta \geq 1$, which clusters more points on the boundary. Once the number of segments required on each edge is known, the point distribution algorithm presented in the next section is then applied to each curve in $\mathcal{D}_\Omega$ to yield the set of boundary seeds.

Generation of Volume Seeds

To generate the set of volume seeds, we developed a sweep-line algorithm where a horizontal line beginning at the north edge of the bounding box of the domain travels downward in steps of length α , the average element diameter. At each step, the sweep-line l_s has y -coordinate y_s . Let us assume that at this y -coordinate the line enters and exits the DCEL domain $\mathcal{D}_\Omega$ twice (see the top sweep line in figure 5.33). Each interval of l_s which lies inside of $\mathcal{D}_\Omega$ has start and end x -coordinates denoted x_l, x_r from which a rectangle $R = [x_l + \alpha, x_r - \alpha] \times [y_s - 0.5\alpha, y_s + 0.5\alpha]$ is constructed over which a uniform 2D probability distribution is constructed and a set of points randomly generated inside of R . On each rectangle R the number of points distributed is given by $N_R = \text{ceil}((x_r - x_l)/\alpha)$. After sweeping through the whole domain the number of volume seeds distributed should be a reasonable approximation to N_{init} . An example initial point distribution with $N_{init} = 500$ is shown in figure 5.34. The primary benefit of this algorithm for volume seed distribution is that all the information it requires about the domain $\mathcal{D}_\Omega$, and therefore the DCEL is the set of boundary curves of $\mathcal{D}_\Omega$, no parameterisation of the faces of $\mathcal{D}_\Omega$ is required. We present the formal algorithm in algorithm 11.

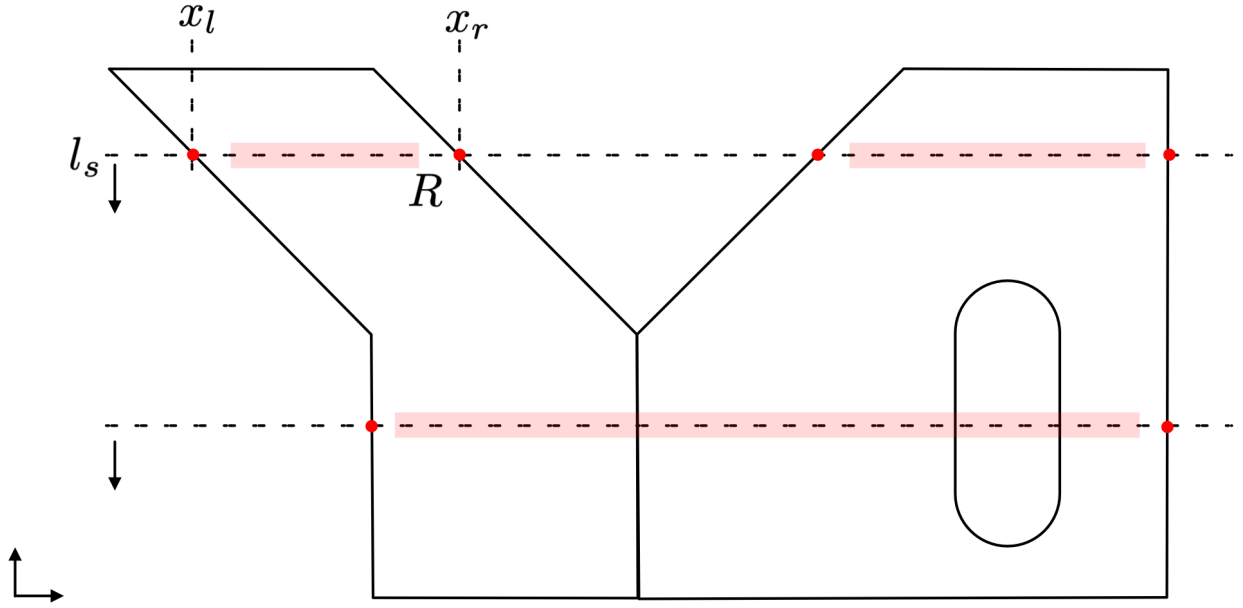


Figure 5.33: Illustration of the sweep algorithm. The red dots denote the intersection points of the sweep line and the light red box denotes the range over which points are randomly distributed.

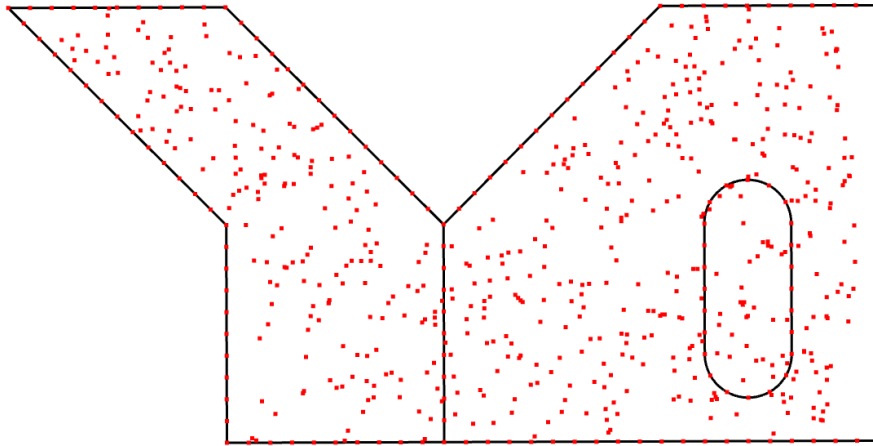


Figure 5.34: Initial point distribution with $N_{\text{init}} = 500$, the actual number of seeds is 729.

Algorithm 11 The initial seed placement routine.

```
1: procedure INITIALPOINTS( $\mathcal{D}_\Omega$ ,  $N_{\text{init}}$ ,  $\beta$ )
2:   Input: A DCEL  $\mathcal{D}_\Omega$ , target no. of elements  $N_{\text{init}}$  and a boundary factor  $\beta$ .
3:   Output: An initial seed set  $\mathbf{P}^{(0)}$ .
4:   Find average element diameter  $\alpha = \sqrt{|\mathcal{D}_\Omega|/N_{\text{init}}}$ 
5:   Set the first vertices of  $\mathbf{P}^{(0)}$  to be those of  $\mathcal{D}_\Omega$ .
6:   for Each forward half-edge  $he \in \mathcal{D}_\Omega$  do
7:     Compute number of points on curve( $he$ ),  $N_E = \beta(L/\alpha)$ 
8:     Compute linearly spaced points  $\mathbf{P}_{he}$ .
9:     Append interior edge points: append( $\mathbf{P}^{(0)}$ ,  $\mathbf{P}_{he}$ )
10:  end for
11:  Compute the bounding box of  $\mathcal{D}_\Omega$ , denoted  $B$ .
12:  Set horizontal sweep line  $l_s$  to  $B.\text{ymax}$ .
13:  while  $l_s.y > B.\text{ymin}$  do
14:    Sweep  $l_s$  downward in  $y$ -coordinate by step  $\alpha$ .
15:    Find the intervals  $[x_l, x_r]$  where  $l_s$  intersects  $\mathcal{D}_\Omega$ .
16:    for Each interval  $[x_l, x_r]$  do
17:      Construct the rectangle  $R = [x_l + \alpha, x_r - \alpha] \times [y_s - 0.5\alpha, y_s + 0.5\alpha]$ 
18:      Generate uniformly distributed points  $\mathbf{P}_R$ .
19:      Append to points to set: append( $\mathbf{P}^{(0)}$ ,  $\mathbf{P}_R$ )
20:    end for
21:  end while
22:  return:  $\mathbf{P}^{(0)}$ 
23: end procedure
```

5.5.3 Seed Point Smoothing

The smoothing algorithm we have implemented is Lloyd's method. We use Qhull [267] to compute the Voronoi tessellation at each iteration. To ensure that all the Voronoi regions (VR) corresponding to seeds in the domain are bounded, we also place a set of fixed seeds along the edges of a bounding box of the domain. An example is presented in figure 5.35.

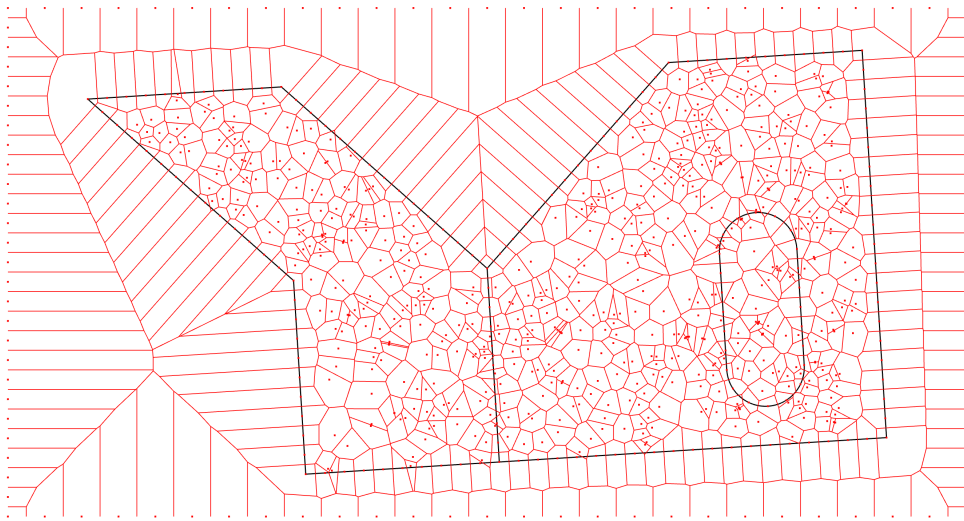


Figure 5.35: The VT corresponding to the initial seed distribution $\mathbf{P}^{(0)}$.

During Lloyd's iteration, volume seeds can occasionally cross the boundary of the domain and become stuck outside of it. This is an occurrence we wish to avoid to ensure the final mesh has the desired number of elements. Therefore, a simple reflection algorithm similar to that presented in [257] was developed. We begin by constructing a pseudo-distance function $d(\mathbf{x}, \Omega)$, which is negative if $\mathbf{x} \in \Omega$, is positive otherwise and returns the minimum distance of $\mathbf{x}$ to the boundary $\partial\Omega$, the corresponding point $\mathbf{x}_{min} \in \partial\Omega$ and the curve parameter $t_{min} \in [0, 1]$. For each curve $C = \{\mathbf{c}(t) : t \in [0, 1]\} \in \partial\Omega$ we find the minimum distance of $\mathbf{x}$, denoted $d(\mathbf{x}, C)$, the corresponding point $\mathbf{x}_C$ and parameter t_C by solving the following constrained minimisation problem:

$$\begin{aligned} &\text{Minimize} \quad f(t) = \|\mathbf{c}(t) - \mathbf{x}\|^2, \quad \text{where} \quad \frac{df}{dt} = 2(\mathbf{c}(t) - \mathbf{x}) \cdot \frac{d\mathbf{c}}{dt} \\ &\text{subject to} \quad t \geq 0 \quad \text{and} \quad t \leq 1 \end{aligned}$$

Once the minimum distances to each of the curves in $\partial\Omega$ have been computed, the curve with the minimum value of $|d(\mathbf{x}, C)|$ is the curve in $\partial\Omega$ closest to $\mathbf{x}$. Therefore $d(\mathbf{x}, \Omega)$, $\mathbf{x}_{min}$ and t_{min} are equal to the $d(\mathbf{x}, C)$, $\mathbf{x}_C$ and t_C of the nearest curve. With the pseudo-distance function computed for $\mathbf{x}$, the reflected point $\mathbf{r}_\Omega(\mathbf{x})$ is given by:

$$\mathbf{r}_\Omega(\mathbf{x}) = \mathbf{x}_{min} - d(\mathbf{x}, \Omega)\mathbf{n}(t_{min}). \quad (5.5.4)$$

The reflection is illustrated in figure 5.36.

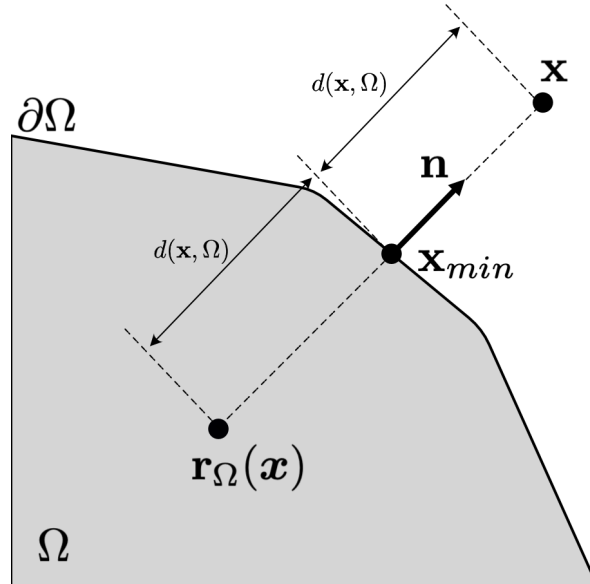


Figure 5.36: Illustration of point reflection.

At each Lloyd's iteration the neighbouring volume seeds of each of the boundary seeds are labelled as being "outside" if $d(\mathbf{x}, \Omega) > 0$ and are reflected back into the domain. The output of Qhull contains a full set of connectivity relations between Voronoi vertices, Voronoi edges and Voronoi regions. Therefore, in line 5 of the reflection algorithm, the set of seeds neighbouring

a seed can be easily computed by querying the VR-to-VR connectivity, given every seed maps on-to-one with a VR.

Algorithm 12 Seed point reflection.

```

1: procedure REFLECTION( $\mathcal{D}_\Omega$ ,  $VT(\mathbf{P}, \Omega)$ )
2:   Input: The DCEL  $\mathcal{D}_\Omega$  and a Voronoi tessellation  $VT(\mathbf{P}, \Omega)$ 
3:   Output: The same seed set  $\mathbf{P}$  but with external seeds reflected.
4:   for Each boundary seed  $\mathbf{s}$  do
5:     Find set of neighbouring volume seeds,  $S_{neigh}$ 
6:     for Each neighbour  $\mathbf{p} \in S_{neigh}$  do
7:       if  $d(\mathbf{p}, \Omega) > 0$  then
8:         Reflect point using equation (5.5.4):  $\mathbf{p} \leftarrow \mathbf{r}_\Omega(\mathbf{p})$ .
9:       end if
10:    end for
11:  end for
12: end procedure

```

The full smoothing algorithm may now be presented in algorithm 13, this algorithm is the same as that presented in [257].

Algorithm 13 Lloyd's smoother.

```

procedure LLOYDSMOOTH( $\mathcal{D}_\Omega$ ,  $\mathbf{P}^0$ ,  $N_{max}$ ,  $\epsilon$ )
  Inputs The DCEL  $\mathcal{D}_\Omega$ , the set of initial seed points  $\mathbf{P}^{(0)}$ , maximum number of lloyds
  iterations  $N_{max}$  and error tolerance  $\epsilon$ .
  Outputs Smoothed points  $\mathbf{Q}$ , Voronoi tessellation  $VT(\mathbf{Q}, \Omega)$ .
  Initialise variables:  $i \leftarrow 0$ ,  $e \leftarrow 1$ 
  Initialise variables:  $\mathbf{P}^c \leftarrow \mathbf{P}^{(0)}$ ,  $VT(\mathbf{P}^{(0)}, \Omega) \leftarrow \text{QHULL}(\mathbf{P}^{(0)})$ 
  while  $i < N_{max}$  and  $e > \epsilon$  do
    Compute reflections:  $\mathbf{P}^{(i)} \leftarrow \text{REFLECTION}(\mathcal{D}_\Omega, VT(\mathbf{P}^c, \Omega))$ 
    Compute Voronoi Tessellation  $VT(\mathbf{P}^{(i)}, \Omega) = \text{QHULL}(\mathbf{P}^{(i)})$ 
    Compute the centroids using Lloyd's map, equation (5.5.2):  $\mathbf{P}^c = L(\mathbf{P}^{(i)})$ 
    Compute the error parameter  $e$  using equation (5.5.3)
     $i \leftarrow i + 1$ 
  end while
  Return: The final Voronoi tessellation  $VT(\mathbf{P}^{(i)}, \Omega)$ .
end procedure

```

In figure 5.37 we present the final, smoothed VT corresponding to the initial, un-smoothed VT presented in figure 5.35.

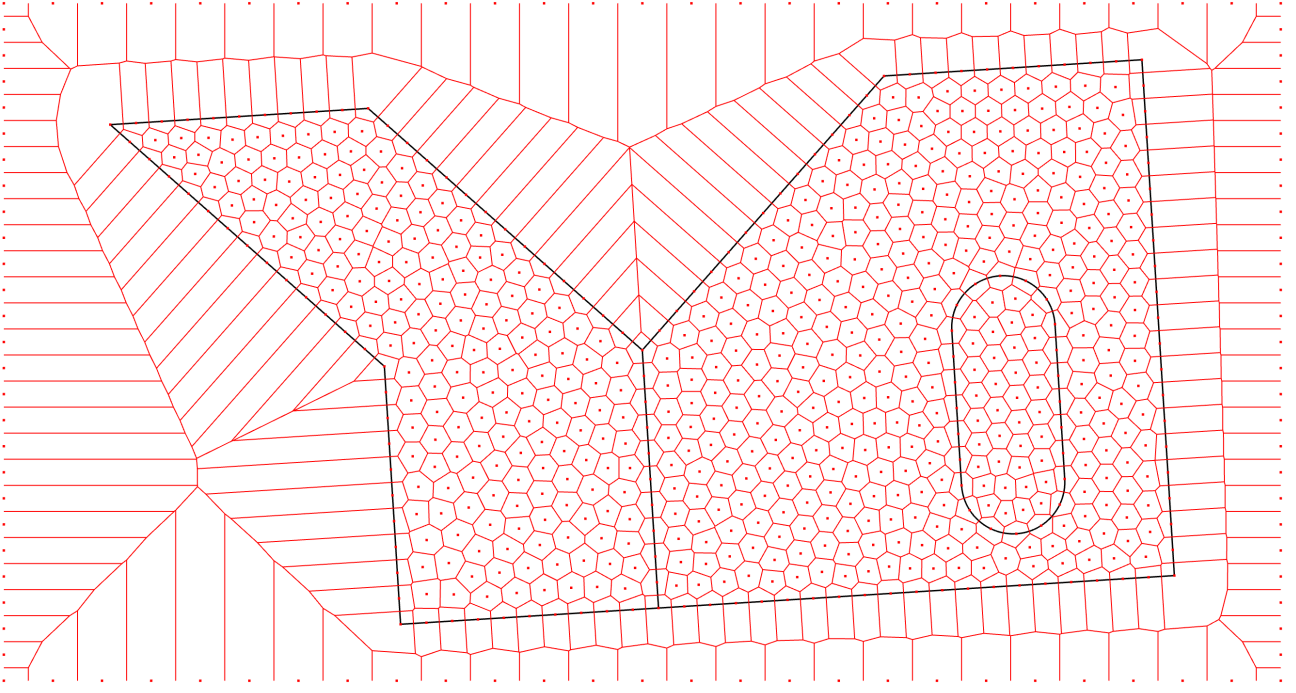


Figure 5.37: The smoothed VT corresponding to the final seed distribution $\mathbf{P}^{(i)}$.

5.5.4 CVT Clipping

Boundary Recovery

The next stage in the mesh-generation pipeline is to compute the intersection of the smoothed Voronoi tessellation, denoted $VT(\mathbf{P}, \Omega)$ and the initial DCEL domain $\mathcal{D}_\Omega$. The first stage is to linearise $\mathcal{D}_\Omega$. This simply means subdividing each of the curves in $\mathcal{D}_\Omega$ into a chain of straight edges using the curve point distribution algorithm we presented in section 5.5.2. The linearised DCEL shall be denoted $\hat{\mathcal{D}}_\Omega$. The full set of intersection points between $\hat{\mathcal{D}}_\Omega$ and $VT(\mathbf{P}, \Omega)$ are then computed using a slightly modified version of the plane sweep algorithm presented in section 2.1 of [248], which we shall review here briefly.

Consider the problem of finding all of the intersection points between two sets of line segments in the plane, $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_m\}$, where the line segments are pairs of points, $a_i = (\mathbf{p}, \mathbf{q})$. Intersections between segments of the same set are ignored. The algorithm begins by merging the sets $S = A \cup B$ and mapping an $ID = 1$ to elements of A and $ID = 2$ to elements of B . This ID may be accessed using $ID(s)$ for all $s \in S$. The basic idea of the sweep line algorithm is to order the set of segments by y -coordinate and sweep a horizontal line downward, keeping track of the set segments which intersect the sweep line at every stage, called the **status**. The set of segments stored in the status are themselves ordered left to right in the order in which they intersect with the sweep line. At each stage of the sweep, only segments which are adjacent in the status need to be tested for intersection in order to guarantee that all intersections are found [248]. We therefore avoid testing every possible pair of segments in S , which of course scales quadratically with the number of segments.

The algorithm proceeds through **event points**, which are stored in an **event queue** $\mathcal{Q}$ with an

order $\prec$ defined as follows. Given two event points, p and q , we say $p \prec q$ if (a) $p_y > q_y$ or (b) if $p_y = q_y$ and $p_x < q_x$. The status data structure, denoted $\mathcal{S}$ is a set of segments ordered left to right by their intersection with the sweep line. Both of these data structures are implemented as binary search trees, which allow for insertion, deletion and search of elements in $\log(n)$ time. The event points come in three categories: the upper endpoints of segments, lower endpoints of segments and intersection points. When each kind of event is encountered the algorithm does the following:

- When an upper endpoint is encountered, the segment it belongs to must be inserted into the status at the position which preserves the left-to-right ordering. It is then tested for intersection with its neighbours to the left and right. If an intersection point is found below the sweep line, it is added to the event queue.
- When a lower endpoint is encountered, the segment it belongs to is deleted from the status. This means its left and right neighbours are now adjacent to one another and must be tested for intersection. Again, if an intersection point is found below the sweep line, it is added to the event queue.
- When an intersection point is encountered, the position of the two intersecting segments in $\mathcal{S}$ are reversed. The segment which is now on the left is tested for intersection with its left neighbour and the segment on the right is tested for intersection with its right neighbour. Again, if an intersection point is found below the sweep line, it is added to the event queue.

We shall provide an example of each of these event points, beginning with the upper endpoint event illustrated in figure 5.38. Prior to reaching the upper endpoint of s_j the status would be $\mathcal{S} = \{s_i, s_k, s_l\}$. Once the upper event is reached it is inserted into $\mathcal{S}$ to give $\mathcal{S} = \{s_i, s_j, s_k, s_l\}$. The left and right neighbours of s_j would then be s_i and s_k . Each neighbour is tested for intersection with s_j . The intersection between s_i and s_j would be found and inserted to $\mathcal{Q}$.

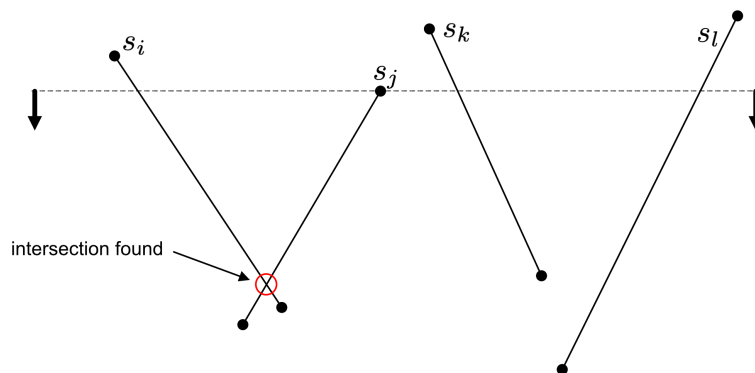


Figure 5.38: Upper endpoint event.

Next, a lower endpoint event is illustrated in figure 5.39. Prior to reaching the lower endpoint the status would be $\mathcal{S} = \{s_i, s_j, s_k, s_l\}$. Once the lower endpoint is reached, s_k is deleted. Segments s_j and s_l would then be adjacent and would be tested for intersection. They are

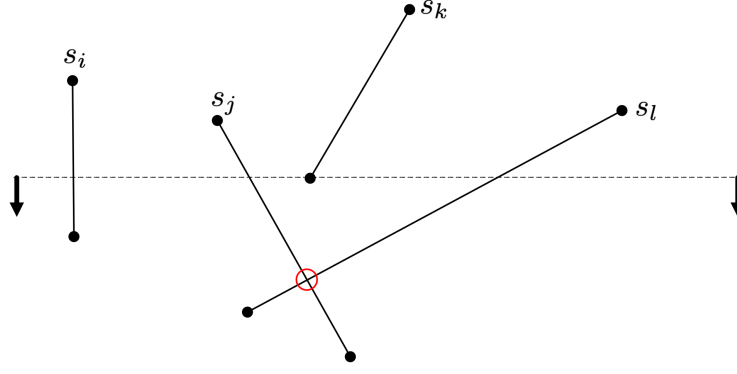


Figure 5.39: Lower endpoint event.

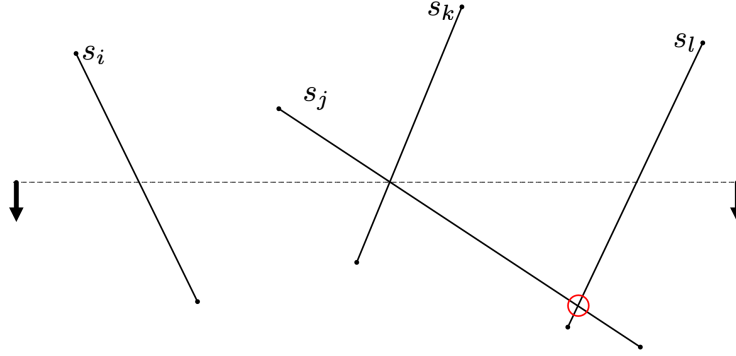


Figure 5.40: Intersection event.

found to intersect below the sweep line and therefore their intersection is added to $\mathcal{Q}$.

Finally, we give an example of an intersection event in figure 5.40. Prior to reaching the event the status would be $\mathcal{S} = \{s_i, s_j, s_k, s_l\}$. Once the intersection event is reached, the intersecting segments are swapped in the status to give $\mathcal{S} = \{s_i, s_k, s_j, s_l\}$. Therefore, s_i and s_k would be tested for intersection and so would s_j and s_l . The intersection between s_j and s_l is below the sweep line and would be added to the event queue. The formal line-sweep procedure is presented in algorithm 14, and is presented in the same way as in [248].

Returning to the problem of finding all the intersection points between the linearised DCEL $\hat{D}_\Omega$ and the smoothed Voronoi tessellation $VT(\mathbf{P}, \Omega)$, clearly this problem may be reduced to a line-sweep problem by treating each as a collection of line segments. Taking this approach has the benefit that no assumption is made as to the smoothness of the Voronoi Tessellation; the algorithm will work even when $VT(\mathbf{P}, \Omega)$ has not been smoothed at all. We present the output of the line sweep algorithm in figure 5.41. The final step is to project the intersection points onto the curves of the DCEL. Therefore every intersection point $\mathbf{p} \in P$ is also mapped to the half-edge $he \in \mathcal{D}_\Omega$ it intersects along with the curve parameter at the intersection (we map to the even-numbered member of a half-edge pair). This information is required to preserve the geometry of $\mathcal{D}_\Omega$ in the mesh.

Note that in lines 17 and 18 of algorithm 14, which are deleting and reinserting $C(p)$ in $\mathcal{S}$, have the effect of reversing the order of the segments $C(p)$ in $\mathcal{S}$. Additionally, lines 19-28 assume that

the left and right neighbour segments s_l, s_r exist. If they do not then these lines are skipped.

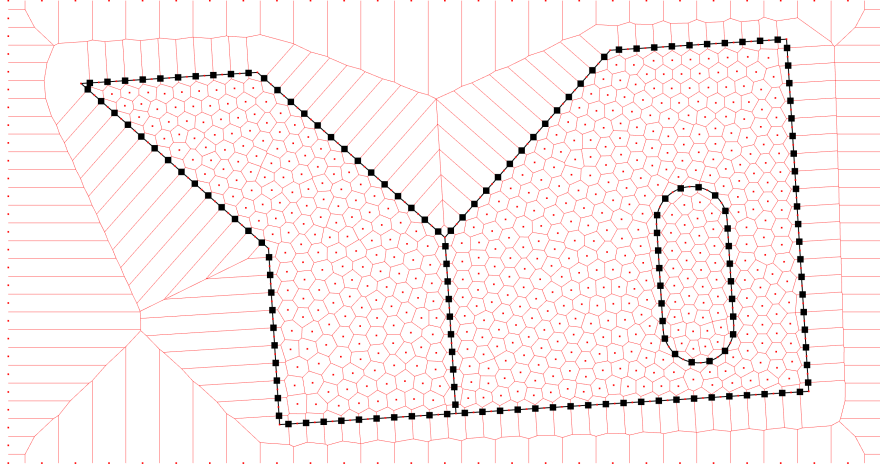


Figure 5.41: The intersection points between the linearised DCEL and the Voronoi Tessellation. The intersections are shown in black.

Algorithm 14 The line sweep algorithm for finding all intersections between sets of segments A and B .

```

1: procedure FINDINTERSECTIONS( $A, B$ )
2:   Inputs: Line segment sets  $A$  and  $B$ .
3:   Outputs: The set of intersections between  $A$  and  $B$ , denoted  $P$ .
4:   Set  $ID(s) = 1$  for all  $s \in A$  and  $ID(s) = 2$  for all  $s \in B$ 
5:   Construct the union of the sets  $S = A \cup B$ ,
6:   Initialise  $P \leftarrow \{ \}$ 
7:   Initialise empty queue  $\mathcal{Q}$  and insert all vertices of  $S$ , store the segment each event
   belongs to with the event.
8:   while  $\mathcal{Q}$  is not empty do
9:     Pop the top event in the queue  $p$  and delete  $p$  from  $\mathcal{Q}$ .
10:    Let  $U(p)$  be the set of segments for which  $p$  is the upper endpoint.
11:    Let  $L(p)$  be the set of segments for which  $p$  is the lower endpoint.
12:    Let  $C(p)$  be the set of segments for which  $p$  is in the interior.
13:    Let  $T = U(p) \cup L(p) \cup C(p)$ 
14:    if  $T$  contains one or more segments from both  $A$  and  $B$  then
15:      Add event point to set of intersections append( $P, p$ )
16:    end if
17:    Delete the segments  $L(p) \cup C(p)$  from  $\mathcal{S}$ 
18:    Insert the segments  $U(p) \cup C(p)$  into  $\mathcal{S}$ 
19:    if  $U(p) \cup C(p)$  is empty then
20:      Let  $s_l$  and  $s_r$  be the left and right neighbours of  $p$ 
21:      Find intersection of  $s_l$  and  $s_r$ , if it is below sweep line insert onto  $\mathcal{Q}$ 
22:    else
23:      Let  $s'$  be the left-most segment of  $U(p) \cup C(p)$ 
24:      Let  $s_l$  be its left neighbour in  $\mathcal{S}$ .
25:      Find intersection of  $s_l$  and  $s'$ , if it is below sweep line insert onto  $\mathcal{Q}$ 
26:      Let  $s''$  be the right-most segment of  $U(p) \cup C(p)$ 
27:      Let  $s_r$  be its right neighbour in  $\mathcal{S}$ .
28:      Find intersection of  $s''$  and  $s_r$ , if it is below sweep line insert onto  $\mathcal{Q}$ 
29:    end if
30:  end while
31: end procedure

```

Constructing the Mesh DCEL

The next step is to construct the DCEL representing the mesh $\mathcal{D}_h = \{\mathcal{V}_h, \mathcal{E}_h, \mathcal{F}_h\}$. This algorithm has three parts.

Step 1: The vertices of the new DCEL are added. The first set of these vertices are the intersection vertices. These are sorted first by the half-edge ID and second by the curve parameter value. The second set are the Voronoi vertices which reside inside the domain Ω . The vertex records $v \in \mathcal{V}_h$ added in step 1 will all be initialised with $\text{outHedge}(v) = -1$.

Step 2: Half-edges are constructed between the vertices added in step 1. We begin by connecting the vertices which are incident on the curves of $\mathcal{D}_\Omega$ with half-edges. A temporary mapping from half-edges in $\mathcal{D}_\Omega$ to half-edges in $\mathcal{D}_h$ are stored in a "geometry half-edge" to "mesh half-

edge" connectivity, denoted **GE2ME**. Given a half-edge $he \in \mathcal{D}_\Omega$ the ordered set of incident mesh half-edges (sorted by starting curve parameter) are stored in **GE2ME**[he]. This data structure will be used later in step 3. As was mentioned at the end of the previous section, the intersection points are mapped to a curve parameter. Therefore, the geometry of the original DCEL, $\mathcal{D}_\Omega$, may be preserved in $\mathcal{D}_h$ by extracting the curve segments corresponding to each interval $[t_i, t_j]$. The NURBS curves corresponding to segments of the curves in $\mathcal{D}_\Omega$ are then assigned to the **curve**() fields of the incident mesh half-edges.

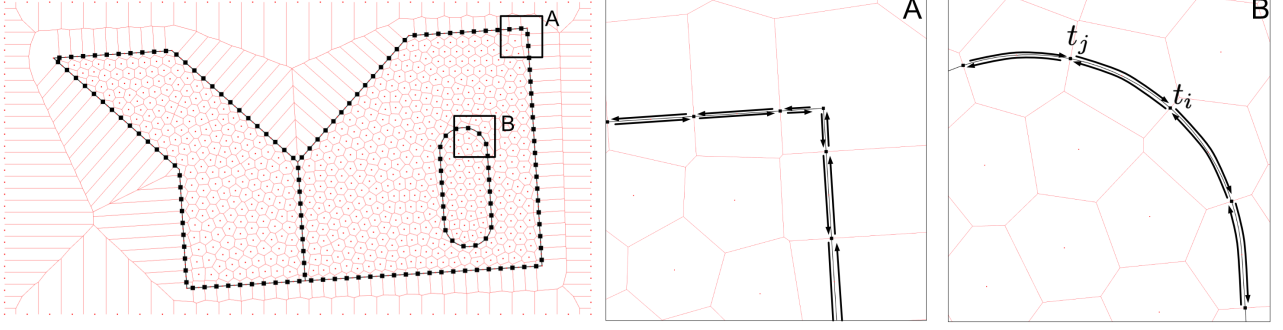


Figure 5.42: Illustration of the boundary and internal interface half-edges.

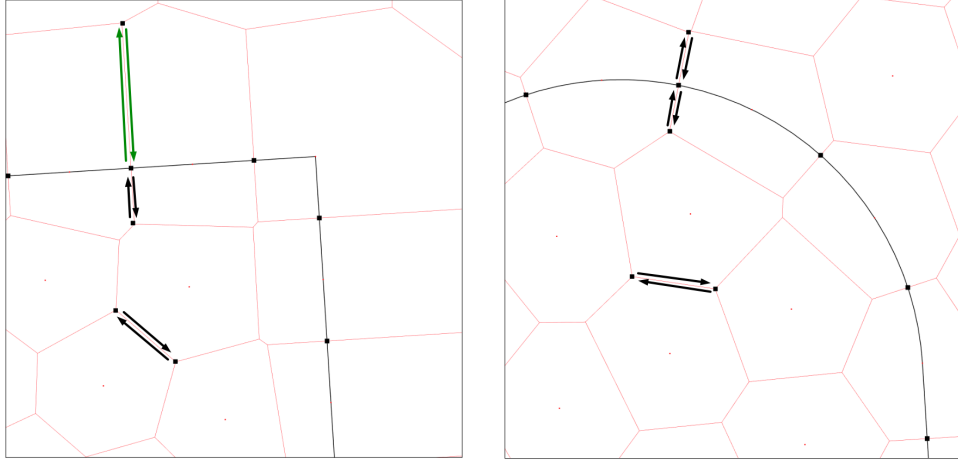


Figure 5.43: Illustration of half-edges constructed from Voronoi edges.

The remaining edges to be added are the Voronoi edges from $VT(\mathbf{P}, \Omega)$. Some of these edges will have been split at one or more locations where they have intersected $\mathcal{D}_\Omega$. Only the subsections which are inside of Ω are added to the mesh DCEL $\mathcal{D}_h$. We demonstrate this procedure in figure 5.43. In the left figure, we show the half-edges corresponding to a Voronoi edge which has been split in half by a boundary curve. The pair of half-edges shown in black are added to $\mathcal{D}_h$ while those shown in green are not as they connect to a Voronoi vertex which is outside of Ω . The right figure shows a Voronoi edge which has been split in half by an internal interface curve. In this case, both pairs of half-edges are added to $\mathcal{D}_h$ as both pairs connect to vertices which are inside of Ω . As half-edges are being added to $\mathcal{D}_h$, we also construct a temporary vertex-to-outgoing-half-edge connectivity **V2E** which will be required to compute the **next**() and **prev**() fields of each half-edge. Once all of the half-edges have been added, the connectivity **V2E** will contain all of the outgoing half-edges starting at each vertex. Therefore, for each vertex

$v \in \mathcal{V}_h$ the set of outgoing half-edges, ordered in counter clockwise order relative to the positive x -axis (angle β in figure 5.44), are stored in $\text{V2E}[v]$. For example, in figure 5.44 we would have $\text{V2E}[v] = \{he_1, he_2, he_3, he_4\}$.

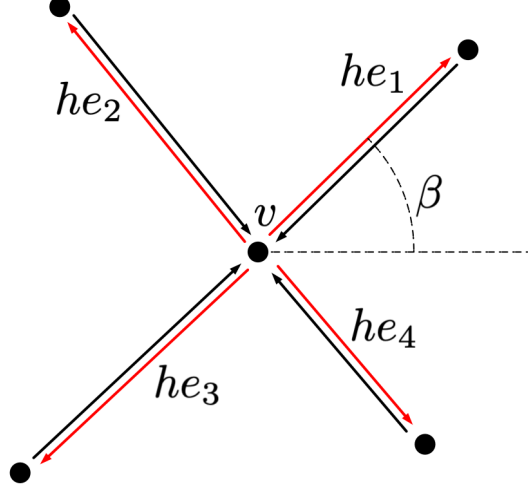


Figure 5.44: Illustration of the half-edges going in and out of vertex v .

Let $\text{ccw}(he)$ denote the first outgoing half-edge counter-clockwise to he , for example $\text{ccw}(he_1) = he_2$ and let $\text{cw}(he)$ denote the first outgoing half-edge clockwise to he , for example $\text{cw}(he_1) = he_4$. For each outgoing half-edge $he_i \in \text{V2E}[v]$, the next and previous fields are set at follows:

$$\text{prev}(he_i) = \text{twin}(\text{ccw}(he_i)), \quad \text{next}(\text{twin}(he_i)) = \text{cw}(he_i) \quad (5.5.5)$$

Applying these formulae to he_4 we see that $\text{prev}(he_4) = \text{twin}(he_1)$ and $\text{next}(\text{twin}(he_4)) = he_3$, as expected. By looping over all of the vertices $v \in \mathcal{V}_h$ and applying equation (5.5.5) to all of the outgoing half-edges connected to v we will set the **next** and **previous** fields of all of the half-edges in $\mathcal{E}_h$.

Step 3: The final step is to add faces to $\mathcal{D}_h$. This is done by appending face records to $\mathcal{F}_h$ corresponding to closed loops of half-edges we identify in $\mathcal{D}_h$. We begin this process by constructing an array of flags, `visited[ ]`, which returns whether a half-edge has been visited by the algorithm yet. Initially `visited[he] = false` for all $he \in \mathcal{E}_h$. The first face to identify is the unbounded face $f_{0,h}$. We must therefore identify the closed loop of half-edges corresponding to the outer boundary and inner boundaries (holes) in $\mathcal{D}_h$ and assign one of them to the `innerHedges( $f_{0,h}$ )` and `outerHedges( $f_{0,h}$ )` fields respectively. We illustrate this process for a simplified example in figure 5.45. To compute this information we use the **GE2ME** connectivity constructed in step 2. The starting half-edge of the outer boundary of $\mathcal{D}_\Omega$ is $he_s = \text{innerHedges}(f_0)[0]$ (marked with an A in figure 5.45), therefore we may set the starting

half-edge for $\mathcal{D}_h$ to:

$$\text{innerHedges}(f_{0,h}) = \{\text{GE2ME}[he_s][0]\}$$

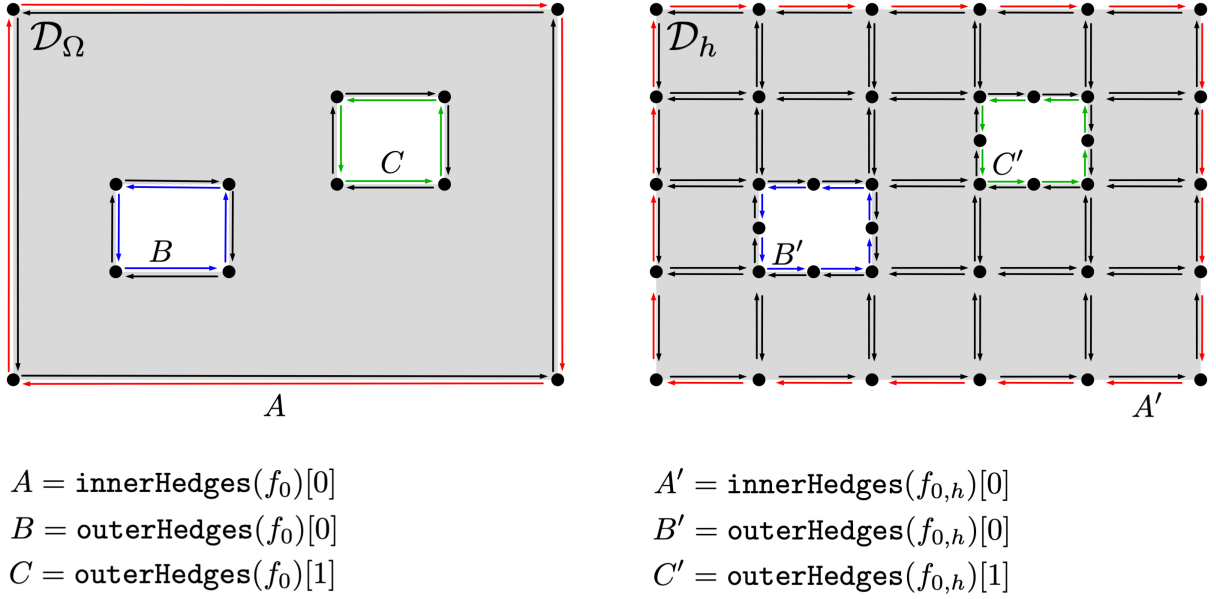


Figure 5.45: Illustration of finding the bounding loops of the mesh DCEL $\mathcal{D}_h$.

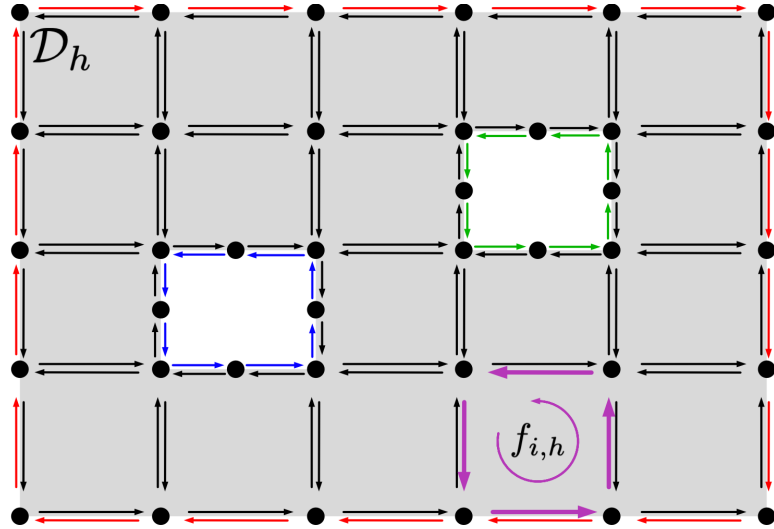


Figure 5.46: Illustration of finding the bounding loop of face $f_{i,h}$, shown in purple.

This is marked A' in figure 5.45. The choice of element 0 of $\text{GE2ME}[he_s]$ is arbitrary, any member would have been correct. We may follow the same procedure for each of the inner loops stored in $\text{outerHedges}(f_0)$, these are marked in figure 5.45 with B and C .

$$\text{outerHedges}(f_{0,h})[i] = \text{GE2ME}[\text{outerHedges}(f_0)[i]][0], \quad \text{for } i = 1, \dots, H,$$

where H is the number of holes in the DCEL, in this case $H = 2$. For each of the half-edges the bounding half-edge loops (marked in red, blue and green on $\mathcal{D}_h$ of figure 5.45) we set $\text{face}(he) = f_{0,h}$ and we also mark those half-edges as "visited" i.e. we set $\text{visited}[he] = \text{true}$.

Next, we must identify all of the bounded faces $f_{i,h}$. We do this by looping through all of the half-edges in $he \in \mathcal{E}_h$, for each half-edge we check if `visited[he] = false`. If so, we then follow the `next()` pointers of each half-edge in the loop until we reach he again. For each of the half-edges $\hat{he}$ in the loop we set `face( $\hat{he}$ ) =  $f_{i,h}$` and `visited[ $\hat{he}$ ] = true`. We then set `outerHedges( $f_{i,h}$ ) =  $he$` , the first half-edge in the loop. Finally, we append $f_{i,h}$ to $\mathcal{F}_h$ and increment the counter $i = i + 1$. We illustrate this procedure in figure 5.46. Once all of the half-edges in $\mathcal{E}_h$ have been visited then all of the loops have been detected and therefore all of the bounded faces have been added to $\mathcal{F}_h$.

Returning to our example geometry, $\mathcal{D}_\Omega$ (figure 5.28), the final mesh DCEL $\mathcal{D}_h$ is shown below in figure 5.47.

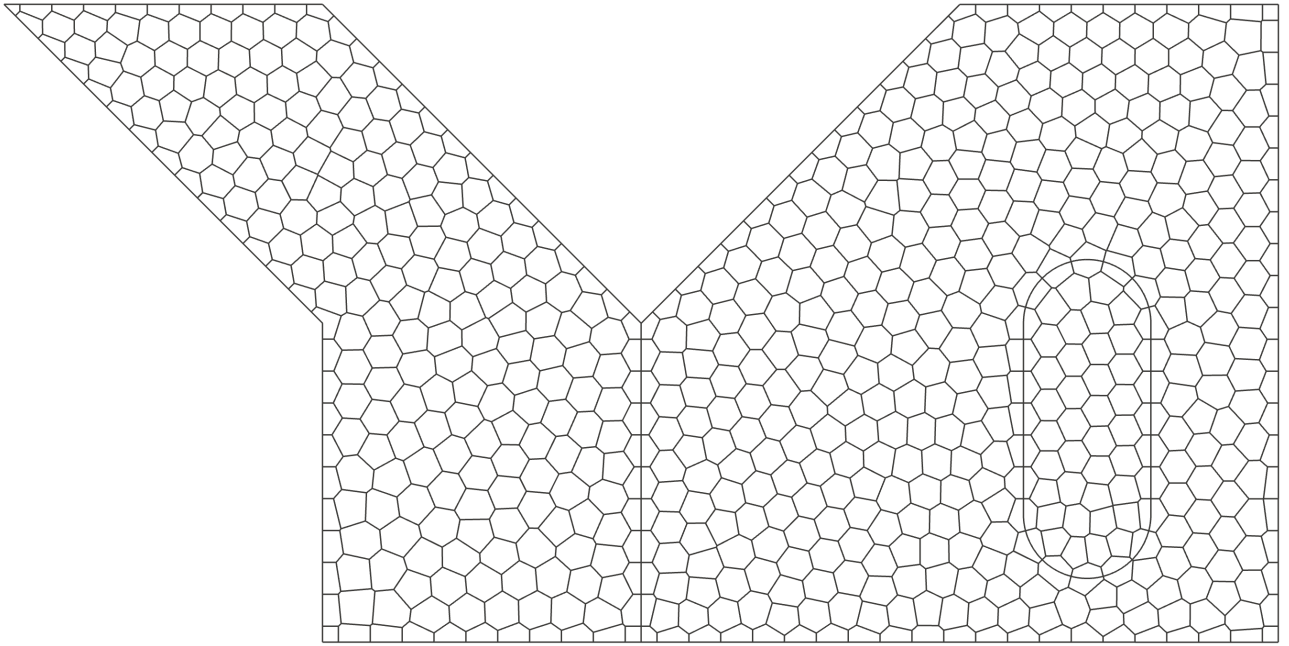


Figure 5.47: The final mesh DCEL, $\mathcal{D}_h$, after steps 1-3 of this section have been executed.

The final operations to be performed on $\mathcal{D}_h$, to be explained in the next section, are post-processing operations which add the final pieces of data which $\mathcal{D}_h$ needs to store for the purpose of implementing a curvilinear VEM.

5.5.5 Post-processing

There are three post-processing operations which need to be performed before the mesh DCEL $\mathcal{D}_h$ is ready for use in the VEM. First of which is the deletion of small edges in the mesh. Small edges lead to both an inefficient distribution of degrees of freedom and to ill-conditioning of the final system of equations. The second post-processing operation is to map block IDs to the mesh faces. The third and final operation is to map boundary IDs to those half-edges which are on the boundaries. The correct mapping of block and boundary IDs is necessary if one wishes to correctly prescribe boundary conditions, material properties and forcing functions when posing a PDE problem.

Small Edge Deletion

The appearance of small edges in a Voronoi tessellation are the result of two adjacent triangles in the dual graph having circumcircles which almost coincide. We present an example of this occurrence in figure 5.48 (this is a recreation of figure 3 in [268]):

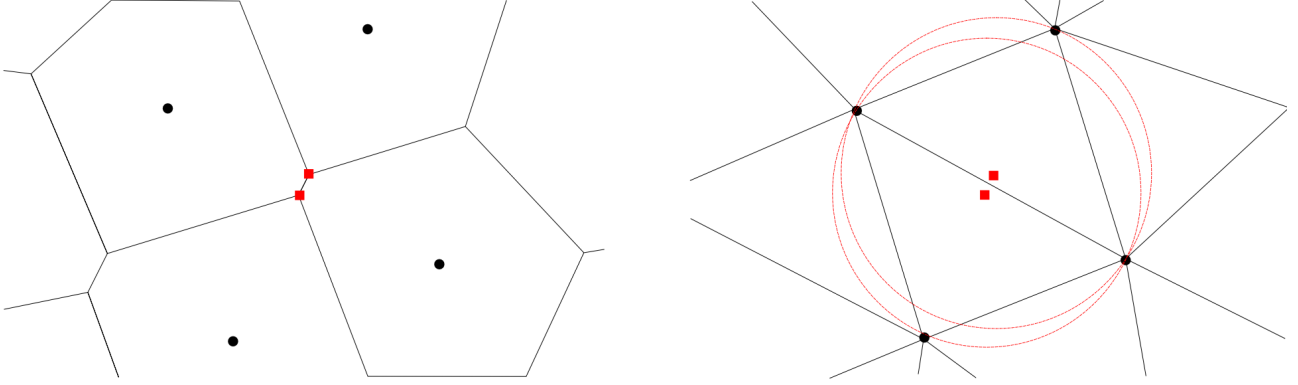


Figure 5.48: An example of a small Voronoi edge (left) and its cause in the dual graph (right).

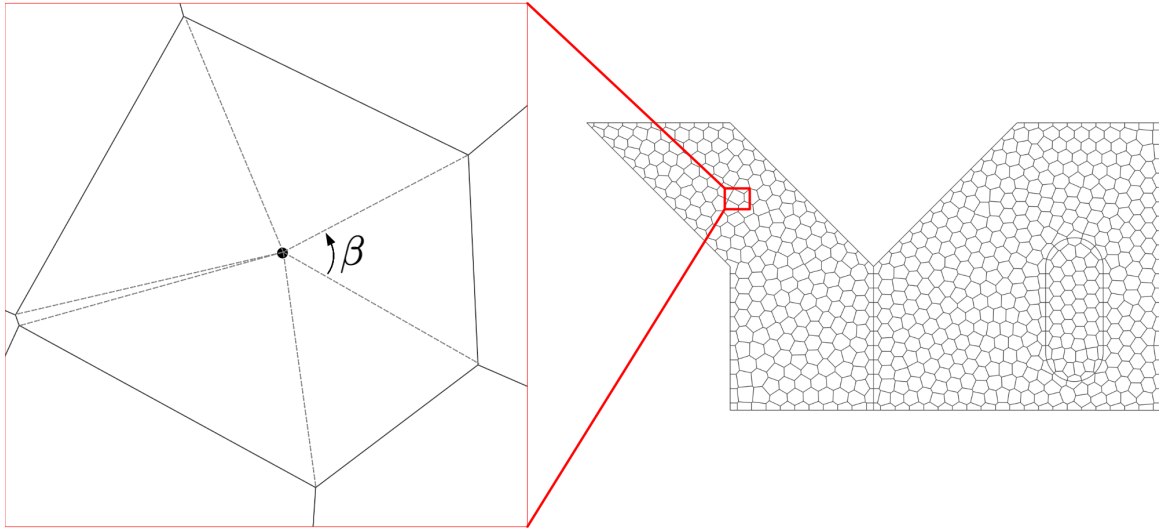


Figure 5.49: An example of a polygon with a small edge and the definition of the swept angle β .

As may be seen in figure 5.48 this occurs when four Delaunay vertices (Voronoi seeds in the VT) are almost co-circular [268]. Even a highly smoothed CVT can exhibit this phenomenon as nothing in the energy functional of equation (5.5.1) penalises small edges. An interesting approach to eliminating these small edges was presented in [268], where the authors derive an energy functional which does penalise small edges. The energy functional is C^1 continuous and is therefore amenable to the standard methods of nonlinear minimisation such as nonlinear conjugate gradients. While the authors show this approach to be highly effective we chose to implement a simpler method which was presented in [257]. In this approach, edges are defined as being small if the angle swept by their vertices is less than:

$$\hat{\beta} = \epsilon_a \frac{2\pi}{N_v},$$

where ϵ_a is a user-specified tolerance (we chose 0.1) and N_v is the number of vertices which define the polygon. We illustrate the edge-collapse algorithm using figure 5.50 below:

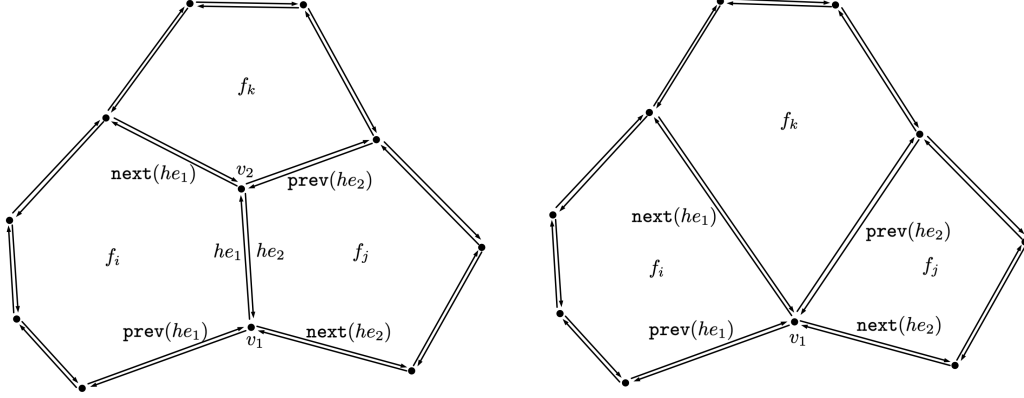


Figure 5.50: An illustration of the edge collapsing algorithm. The mesh before edge-collapse is shown on the left and after edge-collapse on the right.

Using the notation of figure 5.50 the edge-collapse procedure is given by algorithm 15.

Algorithm 15 The edge-collapse algorithm

- 1: **procedure** COLLAPSEEDGE(he_1)
- 2: **Input:** he_1 the half-edge to collapse.
- 3: **Output:** An altered DCEL with he_1 and he_2 collapsed.
- 4: Let v_1 and v_2 be the start and end vertices of half-edge he_1
- 5: Set the outgoing half-edge of the remaining vertex:

$$\text{outHedge}(v_1) = \text{next}(he_1)$$

- 6: Set the origin field of the half-edges which now start at v_1 :

$$\text{origin}(\text{next}(he_1)) = v_1, \quad \text{origin}(\text{twin}(\text{prev}(he_2))) = v_1$$

- 7: Set the next and prev fields of the remaining half-edges:

$$\begin{aligned} \text{prev}(\text{next}(he_1)) &= \text{prev}(he_1), & \text{next}(\text{prev}(he_1)) &= \text{next}(he_1), \\ \text{prev}(\text{next}(he_2)) &= \text{prev}(he_2), & \text{next}(\text{prev}(he_2)) &= \text{next}(he_2) \end{aligned}$$

- 8: Ensure that faces f_i and f_j reference half-edges which exist:

$$\text{outerHedges}(f_i)[0] = \text{next}(he_1), \quad \text{outerHedges}(f_j)[0] = \text{next}(he_2)$$

- 9: **end procedure**
-

By looping over all of the faces in the mesh and checking the angles swept by their edges we are able to identify all of the small edges in the mesh and collapse them using algorithm 15.

Block Identifier Assignment

Each of the faces in the geometry DCEL, $\mathcal{D}_\Omega$, has a block ID mapped to which we shall denote with $\text{block}(f)$ for all $f \in \mathcal{F}$. We wish for these block IDs to be preserved in $\mathcal{D}_h$. The algorithm has two phases.

Phase 1. The first phase begins by initialising a boolean array **visited** which stores whether mesh face $f_h \in \mathcal{F}_h$ has been visited yet, it is initialised to false. Next, we loop over the faces $f \in \mathcal{F}$ of the geometry and identify the mesh faces $f_h \in \mathcal{F}_h$ which are incident with each of the half-edges which bound f . These faces are added to a FIFO queue holding mesh faces; we have one such queue for each geometry face and we shall denote it with $\mathcal{Q}_f$. With each incident mesh face we visit we set $\text{visited}[f_h] = \text{true}$ and we set $\text{block}(f_h) = \text{block}(f)$. The end result of these operations, when applied to our example geometry (figure 5.28), is shown in figure 5.51.

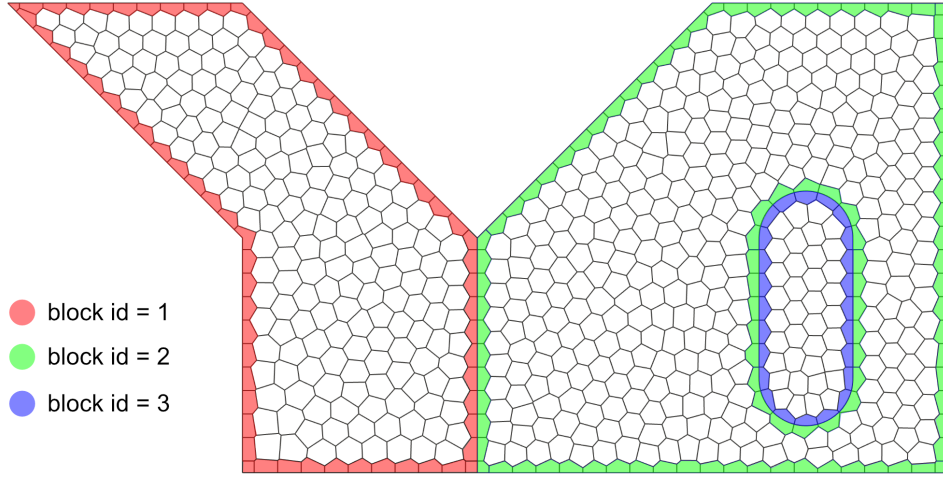


Figure 5.51: Distribution of block IDs on mesh faces after the end of the first phase of the algorithm.

Phase 2. In the second phase we loop over each of the geometry faces and perform a breadth-first search (BFS) of mesh faces beginning with those stored in $\mathcal{Q}_f$. The BFS is a fundamental algorithm from graph theory, in it we use the dual of the mesh as the graph, therefore vertices in this graph correspond to faces in $\mathcal{D}_h$ and an edge between vertices means the corresponding mesh faces are adjacent. New faces are only added to $\mathcal{Q}_f$ if they have not already been visited. This ensures that mesh faces which belong to a different geometry face are not added to the queue. As unvisited mesh faces are visited, their block ID is set to that of the parent geometry face and the flag in the **visited** array is set to **true**. We may now present the full algorithm in algorithm 16

Bound Identifier Assignment

The last post-processing procedure is to map boundary IDs to each of the half-edges in the mesh $\mathcal{D}_h$. If a half-edge is not on the boundary we assign -1 as the boundary ID. This allows us to easily distinguish boundary half-edges from internal half-edges. The procedure is simple, we

Algorithm 16 Sets the block IDs of face of the mesh DCEL.

```

1: procedure SETBLOCKIDS( $\mathcal{D}_\Omega$ ,  $\mathcal{D}_h$ , GE2ME)
2:   Inputs: The geometry DCEL  $\mathcal{D}_\Omega$ , the mesh DCEL  $\mathcal{D}_h$  and the geometry half-edge to
   mesh half-edge mapping GE2ME.
3:   This is the beginning of phase 1.
4:   for Each geometry face  $f \in \mathcal{F}$  do
5:     Initialise FIFO queue  $\mathcal{Q}_f \leftarrow []$ 
6:     for Each bounding half-edge  $he_g$  of  $f$  do
7:       The half-edges the outer loop and inner boundaries are visited.
8:       for Each mesh half-edge  $he_m$  in GE2ME[ $he_g$ ] do
9:         Find parent mesh face:  $f_h \leftarrow \text{face}(he_m)$ 
10:        Set visited flag  $\text{visited}[f_h] \leftarrow \text{true}$ 
11:        Set block ID:  $\text{block}(f_h) \leftarrow \text{block}(f)$ 
12:        Add mesh face to queue:  $\text{push}(\mathcal{Q}_f, f_h)$ 
13:      end for
14:    end for
15:  end for
16:  This is the beginning of phase 2.
17:  for Each geometry face  $f \in \mathcal{F}$  do
18:    while  $\mathcal{Q}_f$  is not empty do
19:      Pop top face:  $f_h \leftarrow \text{pop}(\mathcal{Q}_h)$ 
20:      Find set of neighbouring faces of  $f_h$ , denoted  $\text{neigh}(f_h)$ 
21:      for Each neighbouring face  $f_{h,2} \in \text{neigh}(f_h)$  do
22:        if  $\text{visited}[f_{h,2}]$  is false then
23:          Set the block ID:  $\text{block}(f_{h,2}) \leftarrow \text{block}(f)$ 
24:          Set flag to visited:  $\text{visited}[f_{h,2}] \leftarrow \text{true}$ 
25:          Add face to queue:  $\text{push}(\mathcal{Q}_f, f_{h,2})$ 
26:        end if
27:      end for
28:    end while
29:  end for
30: end procedure

```

loop over all the half-edges in the geometry DCEL $he_g \in \mathcal{D}_\Omega$ and for each of the mesh half-edges $he_m \in \text{GE2ME}[he]$ we simply assign $\text{bound}(he_m) = \text{bound}(he_g)$. The geometry half-edge to mesh half-edge map GE2ME therefore contains all of the information required to correctly assign the boundary IDs to mesh half-edges.

5.6 Discussion

The main strength of the GMK presented in this section is its generality. The DCEL can represent any 2-manifold topology. Additionally, the use of NURBS as the description of geometry means that any realistic reactor design can be represented exactly: no approximation to reactor geometry is required. The dual use of the DCEL for both posing the geometry of a reactor physics problem and for representing the curvilinear polygonal mesh greatly simplifies the practical implementation of a curvilinear VEM in code, as there is greater code re-use between the geometry and mesh modules.

The GMK does have two major limitations however. The first is that it is limited to 2D and second there is not yet any support for mesh adaptivity. These limitations form the basis for some interesting future research for the GMK.

Extension of the GMK to 3D. Reactor geometries in 3D are not 2-manifold due to the existence of internal interfaces between material subdomains. There are a number of data structures in the literature for the representation of 3D subdivisions [244, 269, 270] however the most natural extension would be the combinatorial map (CM) presented in [271, 272]. In 2D the CM reduces to the doubly-connected edge list. In 3D it provides an efficient and complete data structure for storing 3-manifold topology. In order to preserve the 3D geometry of the reactor in the mesh, the polygons which are incident with the boundary surfaces or internal-interface surfaces shall have to be described using a closed loop of trimming curves in the parameter space of the surface.

Mesh adaptivity. A major limitation of the curvilinear VEM presented in this thesis is the lack of posteriori error estimation and mesh adaptivity. These are necessary features of any FEM or VEM which will be used for industrial-scale problems, which routinely scale to millions of degrees of freedom. In [4] an error estimator for the VEM was presented and a scheme for adaptivity based on quadrilateral tessellation:

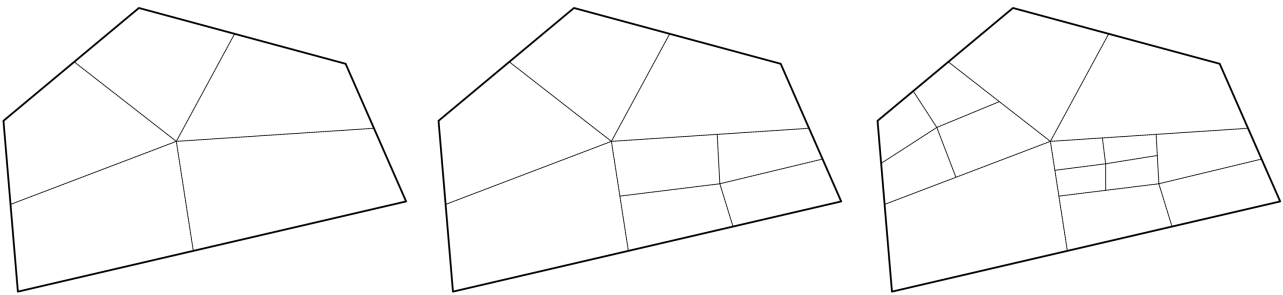


Figure 5.52: Mesh adaptivity scheme presented in [4]

This method of adaptivity has a number of desirable features. First, it would be easy to implement using the existing infrastructure presented in this chapter for the DCEL. Second, the hierarchical structure of this adaptivity scheme allows for simple and efficient transfer of solution field information between grids at different levels of refinement. Third, this method of

adaptivity generalises to 3D, where hexahedra would take the place of quadrilaterals. Lastly, high quality open source libraries for "forest of octrees" adaptivity already exist [273] and have been shown to be highly efficient and scalable in a distributed-memory environment.

Chapter 6

Virtual Element Discretisation of the Multigroup Neutron Diffusion Equation

6.1 Introduction

The multigroup neutron diffusion equation (NDE) is an important mathematical model for simulating nuclear reactor core performance. It is implicitly a $P1$ expansion in angle and is typically used to calculate global quantities of interest for the whole core, which has been spatially homogenised and energy group condensed to 2-4 groups. In the early years of nuclear reactor design in the 1950s and 1960s, computing resources were far less abundant than they are today, therefore the high degree of coarsening in space, angle and energy was necessary for performing any sort of whole-core calculation [97, 102]. As computing resources have increased, it is now feasible to perform whole-core diffusion calculations without any spatial homogenisation. Therefore a numerical method which can preserve the curved geometry of the reactor and handle large, complex meshes in an efficient and scalable way is required.

To this end, we shall present the application of the NURBS-enhanced, continuous Galerkin virtual element method (NECG-VEM) to the multigroup neutron diffusion equation. In particular we shall be solving the fixed source problem and the k -eigenvalue problem.

This chapter is organised as follows:

- In section 6.2, we formally introduce the multigroup, neutron diffusion equation which this chapter shall solve using the NECG-VEM.
- In section 6.3, we introduce the NECG-VEM itself in the context of solving the reaction-diffusion equation, which is mathematically equivalent to a mono-energetic NDE. We present the key properties of the NECG-VEM discretisation which make it well posed and we present the H^1 error estimate of the NECG-VEM discretisation.

- In section 6.4 we derive the variational form of the fixed-source and k -eigenvalue problems and prove their well-posedness under well-defined limits on the nuclear cross section data.
- In section 6.5, we use the tools introduced in section 6.3 to derive the discretised, NECG-VEM variational problems. and prove their well-posedness.
- In section 6.6 we derive the systems of linear equations which result from discretising the multigroup NDE using the NECG-VEM and discuss the iterative strategy which was implemented to solve them. We also introduce a distributed-memory parallelisation strategy for partitioning meshes, assembling the global systems of equations and reordering DoFs to improve the efficiency of the PETSc matrix-vector multiply operation.
- In section 6.7 we present a number of test cases to substantiate the claims made in this chapter. First, we present a method of manufactured solutions (MMS) test case. This is a two-group problem defined on a curved geometry in which we show that the NECG-VEM converges with the expected order of accuracy of $p + 1$ for a scheme of order p . We also show that the conventional, straight-sided VEM converges sub-optimally for the curved geometry, as would be expected given there is a geometric error. Second, we present a burnable supercell benchmark verification test case. This is a small, single-group, fixed-source problem which we use to demonstrate the improved performance of the NECG-VEM relative to the classical, straight-sided VEM when computing integral quantities of the scalar flux. Third, we present the 2D OECD-NEA C5G7 benchmark verification test case. This is a seven-group, k -eigenvalue problem which models a PWR core. This test case demonstrates the ability of the NECG-VEM to scale to larger problems and also provides a more advanced example of curved, polygonal mesh generation.
- Finally, in section 6.8 we provide a discussion of the strengths and limitations of the NECG-VEM presented in this chapter and make some recommendations for future research.

6.2 The Multigroup Diffusion Equations

Let $V \subset \mathbb{R}^2$ be the domain of interest, the boundary ∂V is the union of a prescribed vacuum boundary and reflective boundary Γ_V and Γ_R such that $\Gamma_R \cap \Gamma_V = \emptyset$. Then the steady-state, multi-group NDE is given by equation (6.2.1) [102, 146]:

$$\begin{aligned} -\nabla \cdot D^g(\mathbf{x}) \nabla \phi_g + \Sigma_r^g(\mathbf{x}) \phi_g &= Q_g(\mathbf{x}) \quad \forall \mathbf{x} \in V, \\ D^g(\mathbf{x}) \nabla \phi_g \cdot \mathbf{n} &= 0 \quad \forall \mathbf{x} \in \Gamma_R, \\ D^g(\mathbf{x}) \nabla \phi_g \cdot \mathbf{n} + \frac{1}{2} \phi_g &= 0 \quad \forall \mathbf{x} \in \Gamma_V, \end{aligned} \tag{6.2.1}$$

where $g = 1, \dots, G$, $D^g(\mathbf{x})$ is the group g diffusion coefficient, $\Sigma_r^g(\mathbf{x})$ is the group g macroscopic neutron removal cross section, defined as:

$$\Sigma_r^g(\mathbf{x}) = \Sigma_a^g(\mathbf{x}) + \sum_{g' \neq g}^G \Sigma_s^{g \rightarrow g'}(\mathbf{x}) = \Sigma_t^g(\mathbf{x}) - \Sigma_s^{g \rightarrow g}(\mathbf{x}),$$

where $\Sigma_t^g(\mathbf{x})$ is the macroscopic neutron total cross section. By convention $g = 1$ is the high energy (fast) flux and $g = G$ is the low energy (thermal) flux. $Q_g(\mathbf{x})$ is a neutron source for group g which will take one of two forms depending on the type of problem under consideration.

- In fixed-source problems $Q_g(\mathbf{x})$ takes the form:

$$Q_g(\mathbf{x}) = \sum_{g' \neq g}^G \Sigma_s^{g' \rightarrow g}(\mathbf{x}) \phi_{g'} + q_g(\mathbf{x})$$

where $q_g(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ is an extraneous, or prescribed (fixed) neutron source.

- In k -eigenvalue problems, $Q_g(\mathbf{x})$ takes the form:

$$Q_g(\mathbf{x}) = \frac{\chi^g(\mathbf{x})}{K_{\text{eff}}} \sum_{g'=1}^G \nu \Sigma_f^{g'}(\mathbf{x}) \phi_{g'} + \sum_{g' \neq g}^G \Sigma_s^{g' \rightarrow g}(\mathbf{x}) \phi_{g'}$$

where $\chi^g(\mathbf{x})$ is the group g prompt neutron fission spectrum, ν is the average number of neutrons per fission, $\Sigma_f^g(\mathbf{x})$ is the group g macroscopic fission cross section, $\Sigma_s^{g' \rightarrow g}(\mathbf{x})$ is the macroscopic scatter cross section from group g' to g and K_{eff} is the effective multiplication factor, an important quantity of interest (QoI) in nuclear reactor physics as it indicates the degree of criticality of the system.

Let $L_g = -\nabla \cdot D^g \nabla + \Sigma_r^g$, then the extraneous, or prescribed (fixed) neutron source problem has the following matrix form [97, 102]:

$$\begin{bmatrix} L_1 & 0 & 0 & \dots \\ 0 & L_2 & 0 & \dots \\ 0 & 0 & L_3 & \dots \\ \vdots & \vdots & \vdots & \dots \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} - \begin{bmatrix} 0 & \Sigma_s^{2 \rightarrow 1} & \Sigma_s^{3 \rightarrow 1} & \dots \\ \Sigma_s^{1 \rightarrow 2} & 0 & \Sigma_s^{3 \rightarrow 2} & \dots \\ \Sigma_s^{1 \rightarrow 3} & \Sigma_s^{2 \rightarrow 3} & 0 & \dots \\ \vdots & \vdots & \vdots & \dots \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} = \begin{bmatrix} q_1(\mathbf{x}) \\ q_2(\mathbf{x}) \\ \vdots \\ q_G(\mathbf{x}) \end{bmatrix}$$

or more succinctly in operator form:

$$(L - S)\phi = q(\mathbf{x}) \quad \text{where} \quad q(\mathbf{x}) = [q_1(\mathbf{x}), \dots, q_G(\mathbf{x})]^T. \quad (6.2.2)$$

Letting $F_{g,g'} = \chi_g \nu \Sigma_f^{g'}$, then the eigenvalue, or nuclear criticality (K_{eff}) problem has the following matrix form [97, 102]:

$$\begin{bmatrix} L_1 & 0 & 0 & \dots \\ 0 & L_2 & 0 & \dots \\ 0 & 0 & L_3 & \dots \\ \vdots & \vdots & \vdots & \dots \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} - \begin{bmatrix} 0 & \Sigma_s^{2 \rightarrow 1} & \Sigma_s^{3 \rightarrow 1} & \dots \\ \Sigma_s^{1 \rightarrow 2} & 0 & \Sigma_s^{3 \rightarrow 2} & \dots \\ \Sigma_s^{1 \rightarrow 3} & \Sigma_s^{2 \rightarrow 3} & 0 & \dots \\ \vdots & \vdots & \vdots & \dots \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} = \frac{1}{\gamma} \begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} & \dots \\ F_{2,1} & F_{2,2} & F_{2,3} & \dots \\ F_{3,1} & F_{3,2} & F_{3,3} & \dots \\ \vdots & \vdots & \vdots & \dots \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix},$$

or equivalently in operator form:

$$(L - S)\phi = \frac{1}{\gamma} F\phi \quad \text{where} \quad \phi = [\phi_1, \dots, \phi_G]^T, \quad (6.2.3)$$

from which it becomes clear that γ are the eigenvalues of the operator $(L - S)^{-1}F$. It has been shown that the spectrum of this operator forms a positive decreasing series $\gamma_1 > \gamma_2 > \gamma_3 > \dots$ (see [274] chapter 3.3). The effective multiplication factor is defined as the first eigenvalue of the spectrum $K_{\text{eff}} = \gamma_1$. The corresponding eigenvector ϕ is the fundamental mode of the system. The discrete NE-VEM approximation of the pair (ϕ_1, K_{eff}) is the desired solution of the nuclear reactor physics benchmark verification test case in this chapter.

Equations (6.2.2) and (6.2.3) have been discretised in angle (implicit P_1 expansion) and in energy to form a system of coupled elliptic PDEs. The final step in discretising equations (6.2.2) and (6.2.3) is to apply a spatial discretisation, in this case the NURBS-Enhanced, continuous Galerkin VEM (NECG-VEM). To this end, we shall first present the case $G = 1$, which reduces to a simple reaction-diffusion equation. Once we have established the key properties of the NECG-VEM for this simpler problem we shall then apply them to the full multigroup neutron diffusion equations.

6.3 The Continuous Galerkin VEM

In this section we shall present the NURBS-enhanced, continuous-Galerkin VEM (NECG-VEM), in which we construct the classical H^1 -conforming and C^0 continuous global virtual element space. This is the most natural space in which to pose a VE discretisation of elliptic problems. In this section we shall use the reaction-diffusion (RD) equation with Neumann boundary conditions as our model problem. We shall prove the well-posedness of a VE discretisation and to derive a priori error estimates for that discretisation. This shall serve as a template for a similar derivation which we shall perform for the neutron diffusion equation.

6.3.1 The Model Problem

The reaction-diffusion equation may be stated as follows:

$$\begin{aligned} -\nabla \cdot (D(\mathbf{x}) \nabla u) + \Sigma(\mathbf{x})u &= f(\mathbf{x}) \quad \forall \mathbf{x} \in V \subset \mathbb{R}^2, \\ -D(\mathbf{x}) \nabla u \cdot \mathbf{n} &= r(\mathbf{x}) \quad \forall \mathbf{x} \in \partial V, \end{aligned} \quad (6.3.1)$$

where we assume $f \in L^2(V)$ and $r \in L^2(\partial V)$. The diffusion and reaction coefficients are assumed to be piece-wise constant. Let the following quantities be defined for the material coefficients:

$$\begin{aligned} D_* &= \inf_{\mathbf{x} \in V} D(\mathbf{x}), \quad D^* = \sup_{\mathbf{x} \in V} D(\mathbf{x}) \\ \Sigma_* &= \inf_{\mathbf{x} \in V} \Sigma(\mathbf{x}), \quad \Sigma^* = \sup_{\mathbf{x} \in V} \Sigma(\mathbf{x}). \end{aligned}$$

In addition to these assumptions on the data we also assume that equation (6.3.1) is in the diffusion-dominated regime. The reaction-dominated regime is defined as the case where $D(\mathbf{x}) \ll \Sigma(\mathbf{x})h^2$. The variational form of equation (6.3.1) is:

$$\begin{cases} \text{Find } u \in H^1(V) \text{ such that:} \\ B(u, v) = l(v) \quad \forall v \in H^1(V) \end{cases} \quad (6.3.2)$$

where:

$$\begin{aligned} B(u, v) &= (D(\mathbf{x}) \nabla u, \nabla v)_{0,V} + (\Sigma(\mathbf{x})u, v)_{0,V}, \\ l(v) &= (v, f)_{0,V} + (v, r)_{0,\partial V}. \end{aligned}$$

Theorem 6.3.1. *The variational problem of equation (6.3.2), under the current assumptions on the material and boundary data is well posed.*

We present the proof of theorem 6.3.1 in section E.1. We shall now present the global CG-VEM space and present some interpolation estimates.

6.3.2 The Global CG-VEM Space

The global VEM space, denoted $\mathcal{V}_p(V)$ is defined in the usual way:

$$\mathcal{V}_p(V) := \left\{ v^h \in H^1(V) : v^h|_F \in \mathcal{V}_p(F) \quad \forall F \in \mathcal{T}_h \right\}.$$

We shall now use the approximation properties of the local VEM spaces $\mathcal{V}_p(F)$ for each $F \in \mathcal{T}_h$ to estimate the approximation properties of the global CG-VEM space. First, we use the local polynomial estimate of lemma 4.4.1 to prove the global polynomial estimate:

Lemma 6.3.2. (*Global Polynomial Estimate*)

Let $\mathcal{T}_h$ be a decomposition of Ω which satisfies the mesh regularity assumption and let $u \in H^s(\Omega)$. Letting $0 \leq m \leq s$ and $1 \leq s \leq p+1$ there exists a $u_\pi \in \mathbb{P}_p(\mathcal{T}_h)$ such that:

$$\|u - u_\pi\|_{m,h} \leq Ch^{s-m} |u|_{s,\Omega}.$$

We present the proof of lemma 6.3.2 in section C.4. From the local interpolation estimate of lemma 4.4.2 the following global interpolation estimate may also be derived:

Lemma 6.3.3. (*Global Interpolation estimate*) Let $u \in H^s(V)$ and let $v_I \in \mathcal{V}_p(V)$ and m, s be two integers such that $0 \leq m \leq s$ and $2 \leq s \leq p+1$ then the following holds:

$$\|u - v_I\|_{m,V} \leq Ch^{s-m} |u|_{s,V}.$$

The proof is almost identical to that for lemma 6.3.2.

6.3.3 The VE Discretisation

At this stage we make the following assumption on the mesh, in addition to those presented in assumption 4.2.1.

Assumption 6.3.1. *We assume that the mesh conforms to the material interfaces of the mesh. Therefore, no element crosses a material interface so the material coefficients are constant inside each element.*

In this thesis we seek to discretise the following local bilinear forms on polygon F :

$$\begin{aligned} B_F(u, v) &= D_F(\nabla u, \nabla u)_{0,F} + \Sigma_F(u, v)_{0,F} \\ l_F(v) &= (v, f(\mathbf{x}))_{0,F}, \end{aligned}$$

where we have removed the material coefficients from the integrals on F in accordance with assumption 6.3.1 and the material constants for face F are denoted D_F and Σ_F . In the VEM setting, $B_F(u, v)$ cannot be computed directly but must be approximated by a computable (in

the sense of being computable from the degrees of freedom) bilinear form $B_F^h(\cdot, \cdot)$ that preserves the key properties of symmetry, continuity and coercivity that make the continuous problem (6.3.2) well-posed. Let $u, v \in [\mathcal{V}_p(F) + \mathbb{P}_p(F)]$, and let us decompose each function using the projections $\Pi_{p-1}^0 \nabla$ and Π_p^0 :

$$\nabla u = \Pi_{p-1}^0 \nabla u + (I - \Pi_{p-1}^0) \nabla u, \quad u = \Pi_p^0 u + (I - \Pi_p^0) u.$$

Substituting these decompositions into $B_F(u, b)$ gives:

$$\begin{aligned} B_F(u, v) &= D_F \left(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla v \right)_{0,F} + \Sigma_F \left(\Pi_p^0 u, \Pi_p^0 v \right)_{0,F} \\ &\quad + D_F \left((I - \Pi_{p-1}^0) \nabla u, (I - \Pi_{p-1}^0) \nabla v \right)_{0,F} \\ &\quad + \Sigma_F \left((I - \Pi_p^0) u, (I - \Pi_p^0) v \right)_{0,F}, \end{aligned} \quad (6.3.3)$$

where we eliminated the terms $(\Pi_{p-1}^0 \nabla u, (I - \Pi_{p-1}^0) \nabla u)_{0,F}$ and $(\Pi_p^0 u, (I - \Pi_p^0) u)_{0,F}$ by recalling the orthogonality conditions of equations (4.3.5) and (4.3.7). The first two terms of equation (6.3.3) are the **consistency terms** and are computable from the degrees of freedom of $\mathcal{V}_p(F)$. The second two terms are the **stability terms** and are not computable from the DoFs and must therefore be approximated by computable bilinear forms. Let the stability terms be denoted as follows:

$$\begin{aligned} T_F^\nabla(u, v) &= D_F \int_F (I - \Pi_{p-1}^0) \nabla u \cdot (I - \Pi_{p-1}^0) \nabla v \, dx, \\ T_F^0(u, v) &= \Sigma_F \int_F (I - \Pi_p^0) u (I - \Pi_p^0) v \, dx. \end{aligned}$$

The bilinear forms T_F^∇, T_F^0 induce seminorms on $\mathcal{V}_p(F)$, they are both zero on the polynomial space $\mathbb{P}_k(F)$ and induce a norms on the quotient space $\mathcal{V}_p(F)/\mathbb{P}_0(F)$ (see remark 3). Therefore, we must construct two *computable* bilinear forms $S_F^\nabla, S_F^0 : \mathcal{V}_p(F)/\mathbb{P}_0(F) \times \mathcal{V}_p(F)/\mathbb{P}_0(F) \rightarrow \mathbb{R}$ which approximate T_F^∇, T_F^0 . By the equivalence of norms on finite-dimensional spaces, there exists constants c_0, c_1, d_0, d_1 with $0 < c_0 < c_1$ and $0 < d_0 < d_1$ such that for $u \in \mathcal{V}_p(F)/\mathbb{P}_0(F)$:

$$\begin{aligned} c_0 T_F^\nabla(u, u) &\leq S_F^\nabla(u, u) \leq c_1 T_F^\nabla(u, u) \\ d_0 T_F^0(u, u) &\leq S_F^0(u, u) \leq d_1 T_F^0(u, u) \end{aligned}$$

The constants c_0, c_1, d_0, d_1 are independent of mesh parameter h_F because the bilinear forms S_F^∇ and S_F^0 are designed to scale like T_F^∇ and T_F^0 respectively. Therefore, S_F^∇ must scale like $D_F h_F^{d-2}$ and S_F^0 must scale like $\Sigma_F h_F^d$, where $d = 2$ in this case. There are many choices for S_F^∇, S_F^0 in the literature (see for example [206, 207, 211]) however here we use the original one [90]:

$$S_F^\nabla(u, v) = D_F \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(u) \text{dof}_i(v) \quad S_F^0(u, v) = (h_F^2 \Sigma_F) \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(u) \text{dof}_i(v),$$

which we shall collect into a single stability term $S_F(u, v)$:

$$S_F(u, v) = (D_F + h_F^2 \Sigma_F) \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(u) \text{dof}_i(v) \quad \forall u, v \in \mathcal{V}_p(F)/\mathbb{P}_0(F)$$

For $u, v \in [\mathcal{V}_p(F) + \mathbb{P}_k(F)]$ the discrete VEM bilinear form is therefore given by:

$$\begin{aligned} B_F^h(u, v) \\ = \\ D_F \left(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla v \right)_{0,F} + \Sigma_F \left(\Pi_p^0 u, \Pi_p^0 v \right)_{0,F} + S_F((I - \Pi_p^\nabla)u, (I - \Pi_p^\nabla)v) \end{aligned}$$

The proof of the stability of B_F^h shall be left to the next section.

Remark 10. *Under the current assumptions of materials: piecewise constant, and constant within elements (assumption 6.3.1), we could have made the following choice for the diffusion term:*

$$D_F \left(\nabla \Pi_p^\nabla u, \nabla \Pi_p^\nabla v \right)_{0,F}.$$

However, we wished to present a more general formulation which could be extended in the future to problems with varying material coefficients.

The discretised forcing term, denoted $l_F^h(v)$ is given by:

$$l_F^h(v) = \left(f(\mathbf{x}), \Pi_p^0 v \right)_{0,F} + (r(\mathbf{x}), v)_{0, \partial F \cap \partial V}$$

The global discretised bilinear form is computed from its local counterparts in the usual way. We start with recalling the global space of piecewise polynomials $\mathbb{P}_p(\mathcal{T}_h)$:

$$\mathbb{P}_p(\mathcal{T}_h) := \left\{ q_p \in L^2(V) : q_p|_F \in \mathbb{P}_p(F) \quad \forall F \in \mathcal{T}_h \right\},$$

using this definition we may restrict $u, v \in [\mathcal{V}_p(V) + \mathbb{P}_p(\mathcal{T}_h)]$ and compute the global bilinear form and forcing term as follows:

$$B^h(u, v) = \sum_{F \in \mathcal{T}_h} B_F^h(u, v), \quad l^h(v) = \sum_{F \in \mathcal{T}_h} l_F^h(v)$$

From which the discrete VEM problem may be posed:

$$\begin{cases} \text{Find } u^h \in \mathcal{V}_p(V) \text{ such that} \\ B^h(u^h, v^h) = l^h(v^h) \quad \forall v^h \in \mathcal{V}_p(V) \end{cases} \quad (6.3.5)$$

6.3.4 Well Posedness and H^1 Error Estimate

The purpose of this section is twofold: To establish the well-posedness of the discrete problem of equation (6.3.5) and to prove the optimal error estimate in the H^1 norm. The proof of the error estimate in the L^2 norm is omitted here as it would follow the usual Aubin-Nitsche duality argument [275], which may also be found in [202] or [231] in the context of VEM discretisation.

Well Posedness

To prove the well-posedness of the discrete problem (equation (6.3.5)) we must show that $B^h(u^h, v^h)$ is both continuous and coercive. Under these conditions the solution u^h exists and is unique. First, we show that the bilinear form $B_F^h(u, v)$ satisfies the p -consistency property.

Lemma 6.3.4. (*Consistency*) (p -consistency): *If either $u, v \in \mathbb{P}_p(F)$, the discrete bilinear form $B_F^h(u, v)$ satisfies:*

$$B_F^h(u, v) = B_F(u, v)$$

The proof of lemma 6.3.4 may be found in section E.2

Remark 11. *The discrete bilinear form $B_F^h(\cdot, \cdot)$ satisfies this stronger consistency property because, in this particular case, we have assumed that the material coefficients in each element constant. In the case of variable, non-polynomial coefficients it would not hold that $B_F^h(u, p) = B_F(u, p)$ for $p \in \mathbb{P}_k(F)$. For a more in-depth discussion of this point the reader may consult [196].*

Next, we must show that the discrete bilinear form $B_F^h(u, v)$ satisfies the stability property.

Lemma 6.3.5. (*Stability*): *The discrete local bilinear form $B_F^h(u, v)$ satisfies:*

$$\beta_* B_F(u, u) \leq B_F^h(u, u) \leq \beta^* B_F(u, u)$$

where β_*, β^* are constants independent of mesh parameter h and element F .

The proof of lemma 6.3.5 may be found in section E.3. With the consistency and stability properties of $B_F^h(u, v)$ established we may now state the main result of this section:

Lemma 6.3.6. (*Well-posedness*) *The discrete variational problem of equation (6.3.5) is well posed.*

We present the proof of lemma 6.3.6 in section E.4

H^1 Estimate

We shall first present an abstract error bound in the style of Strang's first lemma (see for example [140] chapter 3.1 or [141] chapter 10.1):

Theorem 6.3.7. Abstract error bound: *Let $u \in H^1(V)$ be the solution to the variational problem of equation (6.3.2) and let $u^h \in \mathcal{V}_p(V)$ be the solution to the discrete VEM problem of*

equation (6.3.5). Then there exists a constant $C > 0$ which depends only on the continuity and coercivity constants such that:

$$\|u - u^h\|_{1,V} \leq C \left(\inf_{v^h \in \mathcal{V}_p(V)} \|u - v^h\|_{1,V} + \sup_{v^h \in \mathcal{V}_p(V) \setminus \{0\}} \frac{|l^h(v^h) - l(v^h)|}{\|v^h\|_{1,V}} + \inf_{p \in \mathbb{P}_p(\mathcal{T}_h)} \|u - p\|_{1,h} \right) \quad (6.3.6)$$

The proof of theorem 6.3.7 is a standard one in the VEM literature (see for example [90]) therefore we present it in the appendix, section E.5. With the abstract bound proven we may now place concrete error bounds on each of its terms. The result of this process is given in theorem 6.3.8:

Theorem 6.3.8. Error bound in H^1 norm: Let $u \in H^{m+1}(V)$ be the solution to the variational problem of equation (6.3.2) and let $u^h \in \mathcal{V}_p(V)$ be the solution to the discrete VEM problem of equation (6.3.5). We define $s = \min(m, p)$ and let $f(\mathbf{x}) \in H^{m-1}(V)$ and $r(\mathbf{x}) \in H^{m-3/2}(\partial V)$ then the following bound holds:

$$\|u - u^h\|_{1,V} \leq Ch^s \left(|u|_{s+1,V} + |f|_{s-1,V} \right)$$

Again, the proof of theorem 6.3.8 follows a series of steps which are typical in both finite element and virtual element methods. Therefore, the proof is given in the appendix, section E.6.

6.4 The Variational Form of the NDE

First, we define the infinite-dimensional space $\mathcal{W}(V) := [H^1(V)]^G$ where we pose the variational forms of equations (6.2.2) and (6.2.3). Letting $u, v \in \mathcal{W}(V)$, the inner product and norm of $\mathcal{W}(V)$ are defined as:

$$(u, v)_{\mathcal{W}(V)} := \sum_{g=1}^G (u_g, v_g)_{1,V}, \quad \|u\|_{\mathcal{W}(V)}^2 := \sum_{g=1}^G \|u_g\|_{1,V}^2.$$

Additionally, we define the space $\mathcal{W}_p(V) := [\mathcal{V}_p(V)]^G$ which we use to pose the VEM discretisation of equations (6.2.3) and (6.2.2). Given that $\mathcal{V}_p(V) \subset H^1(V)$ then we have $\mathcal{W}_p(V) \subset \mathcal{W}(V)$. We make three assumptions about the computational mesh $\mathcal{T}_h = (\mathcal{V}_h, \mathcal{E}_h, \mathcal{F}_h)$ in addition to assumption 4.2.1:

Assumption 6.4.1. *Material properties:* the neutron diffusion coefficient and the macroscopic neutron cross sections are all constant, finite and positive within each material sub-domain. For example within a nuclear fuel pin or the moderator.

Assumption 6.4.2. *Face Conformity:* in $\mathcal{T}_h$ there exists no polygon that crosses a material boundary. This means that every polygon must reside entirely within a material sub-domain and must conform to internal boundaries.

Assumption 6.4.3. *Edge Conformity:* the set of edges in $\mathcal{E}_h$ that lie on the boundary ∂V may be split into two disjoint sets of edges $\mathcal{E}_R$ and $\mathcal{E}_V$:

$$\mathcal{E}_V = \{e \in \mathcal{E}_V : e \subset \Gamma_V\} \quad \mathcal{E}_R = \{e \in \mathcal{E}_V : e \subset \Gamma_R\}$$

that lie solely within Γ_R and Γ_V respectively. In other words, there exists no boundary edge that crosses the interface between Γ_R and Γ_V .

We also define the following notation:

$$\begin{aligned} \underline{D^g} &= \inf_{\mathbf{x} \in V} D^g(\mathbf{x}), & \overline{D^g} &= \sup_{\mathbf{x} \in V} D^g(\mathbf{x}) \\ \underline{\Sigma_X^g} &= \inf_{\mathbf{x} \in V} \Sigma_X^g(\mathbf{x}), & \overline{\Sigma_X^g} &= \sup_{\mathbf{x} \in V} \Sigma_X^g(\mathbf{x}) \end{aligned}$$

where Σ_X^g is a placeholder for any macroscopic neutron cross section. Starting with equation (6.2.1), in operator form and letting $Q = [Q_1, \dots, Q_G]^T$ we take the inner product of each term with a weighting function $v = [v_1, v_2, \dots, v_G]^T \in [H^1(V)]^G$ and integrate over V to give:

$$\int_V v^T L \phi \, dx - \int_V v^T S \phi \, dx = \int_V v^T Q \, dx. \quad (6.4.1)$$

We shall now derive bilinear and linear forms for each of the terms in equation (6.4.1). Let us begin with the removal term: first integration by parts is applied to the diffusion term, followed

by Gauss' theorem together with the boundary conditions of equation (6.2.1) to give:

$$\begin{aligned} \int_V v^T L \phi \, dx &= \sum_{g=1}^G \int_V (-\nabla \cdot D^g \nabla \phi_g) v_g \, dx + \int_V \Sigma_r^g \phi_g v_g \, dx \\ &= \sum_{g=1}^G \int_V (D^g \nabla \phi_g) \cdot \nabla v_g \, dx + \int_V \Sigma_r^g \phi_g v_g \, dx + \frac{1}{2} \int_{\Gamma_V} \phi_g v_g \, ds, \end{aligned}$$

from which the removal bilinear form, $\hat{L}(\phi, v)$, is defined:

$$\hat{L}(\phi, v) = \sum_{g=1}^G (D^g \nabla \phi_g, \nabla v_g)_{0,V} (\Sigma_r^g \phi_g, v_g)_{0,V} + \frac{1}{2} (\phi_g, v_g)_{0,\Gamma_V}.$$

Next the scatter term:

$$\int_V v^T S \phi \, dx = \sum_{g=1}^G \sum_{g' \neq g}^G \int_V \Sigma_s^{g' \rightarrow g} \phi_{g'} v_g \, dx,$$

from which the scatter bilinear form, $\hat{S}(\phi, v)$, is defined:

$$\hat{S}(\phi, v) = \sum_{g=1}^G \sum_{g' \neq g}^G (\Sigma_s^{g' \rightarrow g} \phi_{g'}, v_g)_{0,V}$$

Finally, the source term has two possibilities:

- The fixed source case:

$$\int_V v^T Q \, dx = \sum_{g=1}^G (q_g(\mathbf{x}), v_g)_{0,V},$$

from which the fixed source linear functional $\hat{Q}(v)$ (note the circumflex) may be defined:

$$\hat{Q}(v) = \sum_{g=1}^G (q_g(\mathbf{x}), v_g)_{0,V}.$$

- The eigenvalue case:

$$\int_V v^T Q \, dx = \int_V v^T \left(\frac{1}{\gamma} F \phi \right) \, dx = \frac{1}{\gamma} \sum_{g=1}^G \sum_{g'=1}^G \int_V \chi_g \nu \Sigma_f^{g'} \phi_{g'} v_g \, dx,$$

from which the fission bilinear form, $\hat{F}(\phi, v)$ is defined:

$$\frac{1}{\gamma} \hat{F}(\phi, v) = \frac{1}{\gamma} \sum_{g=1}^G \sum_{g'=1}^G (F_{g,g'} \phi_{g'}, v_g)_{0,V}$$

Note that the symbols for the bilinear forms $\hat{L}$, $\hat{S}$ and $\hat{F}$ have a circumflex above them to differentiate them from their operator counterparts L , S and F . Using these bilinear forms we may now formally pose the two variational problems we shall solve using the CG-VEM. The first is the **fixed source** variational problem:

$$\begin{cases} \text{Find } \phi \in \mathcal{W}(V) \text{ which satisfies:} \\ \hat{L}(\phi, v) - \hat{S}(\phi, v) = \hat{Q}(v) \quad \forall v \in \mathcal{W}(V) \end{cases} . \quad (6.4.2)$$

The second is the **eigenvalue** variational problem:

$$\begin{cases} \text{Find } \lambda \in \mathbb{R} \text{ such that there exists a } \phi \in \mathcal{W}(V) \text{ which satisfies:} \\ \hat{L}(\phi, v) - \hat{S}(\phi, v) = \lambda \hat{F}(\phi, v) \quad \forall v \in \mathcal{W}(V) \end{cases} . \quad (6.4.3)$$

6.4.1 Well-Posedness of Variational Problems

Letting $\hat{A}(u, v) = \hat{L}(u, v) - \hat{S}(u, v)$ for $u, v \in \mathcal{W}(V)$ we shall first prove the following:

- $\hat{A}(u, v)$ is continuous: $\hat{A}(u, v) \leq C_1 \|u\|_{\mathcal{W}(V)} \|v\|_{\mathcal{W}(V)}$.
- $\hat{A}(u, v)$ is coercive: $\hat{A}(u, u) \geq C_2 \|u\|_{\mathcal{W}(V)}^2$.
- $\hat{Q}(v)$ is continuous: $\hat{Q}(v) \leq C_3 \|v\|_{\mathcal{W}(V)}$.
- $\hat{F}(v)$ is continuous: $\hat{F}(u, v) \leq C_4 \|u\|_{\mathcal{W}(V)} \|v\|_{\mathcal{W}(V)}$.

After which we shall then present the well-posedness results.

Continuity of $\hat{A}(\cdot, \cdot)$

We shall begin with the continuity of $\hat{A}(u, v)$:

$$\hat{A}(u, v) = \sum_{g=1}^G \left\{ (\nabla u_g, D^g \nabla v_g)_{0,V} + (u_g, \Sigma_r^g v_g)_{0,V} + \frac{1}{2} (u_g, v_g)_{0,\Gamma_V} - \sum_{g' \neq g}^G (u_g, \Sigma_s^{g' \rightarrow g} v_{g'})_{0,V} \right\}.$$

The scatter terms may be bounded as follows:

$$-(u_g, \Sigma_s^{g' \rightarrow g} v_{g'})_{0,V} \leq |(u_g, \Sigma_s^{g' \rightarrow g} v_{g'})_{0,V}| \leq \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{0,V} \|v_{g'}\|_{0,V} \leq \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|v_{g'}\|_{1,V},$$

and we bound the reaction-diffusion term in the same way we did in equation (E.1.1), by applying the supremum to the material data followed by the Schwarz inequality and lemma B.0.1:

$$(\nabla u_g, D^g \nabla v_g)_{0,V} + (u_g, \Sigma_r^g v_g)_{0,V} \leq \max(\overline{D^g}, \overline{\Sigma_r^g}) \|u_g\|_{1,V} \|v_g\|_{1,V}.$$

Finally, we bound the boundary term using Schwarz inequality followed by the trace inequality (equation (3.5.1)):

$$(u_g, v_g)_{0,\Gamma_V} \leq \|u_g\|_{0,\Gamma_V} \|v_g\|_{0,\Gamma_V} \leq C_{tr}^2 \|u_g\|_{1,V} \|v_g\|_{1,V}. \quad (6.4.4)$$

Putting these three upper bounds together we have:

$$\hat{A}(u, v) \leq \sum_{g=1}^G \left\{ \left(\max(\overline{D^g}, \overline{\Sigma_r^g}) + C_{tr}^2 \right) \|u_g\|_{1,V} \|v_g\|_{1,V} + \sum_{g' \neq g}^G \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|v_{g'}\|_{1,V} \right\}$$

Which may be written as $\hat{A}(u, v) \leq \mathbf{u}^T \overline{\mathcal{A}} \mathbf{v}$ where

$$\mathbf{u} = \begin{bmatrix} \|u_1\|_{1,V} \\ \|u_2\|_{1,V} \\ \vdots \\ \|u_G\|_{1,V} \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \|v_1\|_{1,V} \\ \|v_2\|_{1,V} \\ \vdots \\ \|v_G\|_{1,V} \end{bmatrix}, \quad \overline{\mathcal{A}} = \begin{bmatrix} \overline{R_1} & \overline{\Sigma_s^{2 \rightarrow 1}} & \overline{\Sigma_s^{3 \rightarrow 1}} & \dots & \overline{\Sigma_s^{G \rightarrow 1}} \\ \overline{\Sigma_s^{1 \rightarrow 2}} & \overline{R_2} & \overline{\Sigma_s^{3 \rightarrow 2}} & \dots & \overline{\Sigma_s^{G \rightarrow 2}} \\ \vdots & \vdots & \ddots & \dots & \vdots \\ \overline{\Sigma_s^{1 \rightarrow G}} & \overline{\Sigma_s^{2 \rightarrow G}} & \overline{\Sigma_s^{3 \rightarrow G}} & \dots & \overline{R_G} \end{bmatrix} \quad (6.4.5)$$

where $\overline{R_g} = \max(\overline{D^g}, \overline{\Sigma_r^g}) + C_{tr}^2$. Therefore, we may bound $\hat{A}(u, v)$ in $\mathcal{W}(V) \times \mathcal{W}(V)$ as follows:

$$\begin{aligned} \hat{A}(u, v) &\leq \mathbf{u}^T \overline{\mathcal{A}} \mathbf{v} = \left(\mathbf{u}, \overline{\mathcal{A}} \mathbf{v} \right)_{\mathbb{R}^G} \leq \|\mathbf{u}\|_{\mathbb{R}^G} \|\overline{\mathcal{A}} \mathbf{v}\|_{\mathbb{R}^G} \leq \|\mathbf{u}\|_{\mathbb{R}^G} \|\overline{\mathcal{A}}\|_2 \|\mathbf{v}\|_{\mathbb{R}^G} \\ &= \|\overline{\mathcal{A}}\|_2 \left(\sum_{g=1}^G \|u\|_{1,V}^2 \right)^{1/2} \left(\sum_{g=1}^G \|v\|_{1,V}^2 \right)^{1/2} = C_1 \|u\|_{\mathcal{W}(V)} \|v\|_{\mathcal{W}(V)} \end{aligned} \quad (6.4.6)$$

where

$$\|\overline{\mathcal{A}}\|_2 = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\overline{\mathcal{A}} \mathbf{x}\|_{\mathbb{R}^G}}{\|\mathbf{x}\|_{\mathbb{R}^G}}$$

Coercivity of $\hat{A}(\cdot, \cdot)$

Again, we begin with the definition of $\hat{A}(u, v)$:

$$\hat{A}(u, v) = \sum_{g=1}^G \left\{ (\nabla u_g, D^g \nabla v_g)_{0,V} + (u_g, \Sigma_r^g v_g)_{0,V} + \frac{1}{2} (u_g, v_g)_{0,\Gamma_V} - \sum_{g' \neq g}^G (u_g, \Sigma_s^{g' \rightarrow g} v_{g'})_{0,V} \right\}.$$

We bound the reaction-diffusion term from below using the same procedure we used in equation (E.1.2):

$$\begin{aligned} (\nabla u_g, D^g \nabla u_g)_{0,V} + (u_g, \Sigma_r^g u_g)_{0,V} + \frac{1}{2} (u_g, u_g)_{0,\Gamma_V} &\geq \min(\underline{D^g}, \underline{\Sigma_r^g}) \|u_g\|_{1,V}^2 + \frac{1}{2} \|u_g\|_{0,\Gamma_V}^2 \\ &\geq \min(\underline{D^g}, \underline{\Sigma_r^g}) \|u_g\|_{1,V}^2 \end{aligned} \quad (6.4.7)$$

Next, we consider the scatter terms which due to the Schwarz inequality satisfy:

$$(u_g, \Sigma_s^{g' \rightarrow g} u_{g'})_{0,V} \leq \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{0,V} \|u_{g'}\|_{0,V} \leq \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|u_{g'}\|_{1,V}$$

therefore the full scattering term satisfies:

$$- \sum_{g' \neq g}^G (u_g, \Sigma_s^{g' \rightarrow g} u_{g'})_{0,V} \geq - \sum_{g' \neq g}^G \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|u_{g'}\|_{1,V} \quad (6.4.8)$$

Substituting equations (6.4.7) and (6.4.8) into $\hat{A}(u, u)$ yields:

$$\hat{A}(u, u) \geq \sum_{g=1}^G \left\{ \min(\underline{D}_s^g, \underline{\Sigma}_r^g) \|u_g\|_{1,V}^2 - \sum_{g' \neq g}^G \overline{\Sigma}_s^{g' \rightarrow g} \|u_g\|_{1,V} \|u_{g'}\|_{1,V} \right\} \quad (6.4.9)$$

We may write equation (6.4.9) as: $\hat{A}(u, u) \geq \mathbf{u}^T \underline{\mathbf{A}} \mathbf{u}$, where

$$\mathbf{u} = \begin{bmatrix} \|u_1\|_{1,V} \\ \|u_2\|_{1,V} \\ \vdots \\ \|u_G\|_{1,V} \end{bmatrix}, \quad \underline{\mathbf{A}} = \begin{bmatrix} R_1 & -\overline{\Sigma}_s^{2 \rightarrow 1} & -\overline{\Sigma}_s^{3 \rightarrow 1} & \dots & -\overline{\Sigma}_s^{G \rightarrow 1} \\ -\overline{\Sigma}_s^{1 \rightarrow 2} & R_2 & -\overline{\Sigma}_s^{3 \rightarrow 2} & \dots & -\overline{\Sigma}_s^{G \rightarrow 2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\overline{\Sigma}_s^{1 \rightarrow G} & -\overline{\Sigma}_s^{2 \rightarrow G} & -\overline{\Sigma}_s^{3 \rightarrow G} & \dots & R_G \end{bmatrix} \quad (6.4.10)$$

where $R_g = \min(\underline{D}_s^g, \underline{\Sigma}_r^g)$. If $\underline{\mathbf{A}}$ is positive-definite then there exists a constant $\alpha > 0$ such that: $\mathbf{u}^T \underline{\mathbf{A}} \mathbf{u} \geq \alpha \mathbf{u}^T \mathbf{u}$. Therefore we have:

$$\hat{A}(u, u) \geq \alpha \mathbf{u}^T \mathbf{u} = \alpha \left(\sum_{g=1}^G \|u_g\|_{1,V}^2 \right) = \alpha \|u\|_{\mathcal{W}(V)}^2$$

Therefore, provided $\underline{\mathbf{A}}$ is positive-definite, $\hat{A}(u, v)$ is coercive with $C_2 = \alpha$. The matrix $\underline{\mathbf{A}}$ is simple to compute using the nuclear cross section data which is typically used in neutron diffusion codes. Therefore, the problem of verifying whether $\hat{A}(u, v)$ is coercive reduces to verifying that $\underline{\mathbf{A}}$ is positive definite, which may be accomplished by verifying that all of the eigenvalues of $\underline{\mathbf{A}}$ are positive real numbers.

Continuity of $\hat{Q}(\cdot)$

Next, we show that $\hat{Q}(v)$ is continuous with respect to $\|\cdot\|_{\mathcal{W}(V)}$. We use the Schwarz inequality and the fact that $\|v\|_{0,V} \leq \|v\|_{1,V}$ to get:

$$\hat{Q}(v) = \sum_{g=1}^G (v_g, q_g(\mathbf{x}))_{0,V} \leq \sum_{g=1}^G \|v_g\|_{0,V} \|q_g\|_{0,V} \leq \sum_{g=1}^G \|v_g\|_{1,V} \|q_g\|_{0,V}.$$

Next, we use the Cauchy-Schwarz inequality for sums to get:

$$\hat{Q}(v) \leq \sum_{g=1}^G \|q_g\|_{0,V} \|v_g\|_{1,V} \leq \left(\sum_{g=1}^G \|q_g\|_{0,V}^2 \right)^{1/2} \left(\sum_{g=1}^G \|v_g\|_{1,V}^2 \right)^{1/2} = C_3 \|v\|_{\mathcal{W}(V)}$$

Continuity of $\hat{F}(\cdot, \cdot)$

Finally, we show the continuity of $\hat{F}(u, v)$ with respect to the norm $\|\cdot\|_{\mathcal{W}(V)}$. Recalling that $F_{g,g'} = \chi_g \nu \Sigma_f^{g'}$ and that $\|u\|_{0,V} \leq \|u\|_{1,V}$ have:

$$\hat{F}(u, v) = \sum_{g=1}^G \sum_{g'=1}^G (u_g, F_{g,g'} v_{g'})_{0,V} \leq \sum_{g=1}^G \sum_{g'=1}^G \overline{F_{g,g'}} \|u_g\|_{1,V} \|v_{g'}\|_{1,V}$$

which may be written as $\hat{F}(u, v) \leq \mathbf{u}^T \mathcal{F} \mathbf{v}$ where:

$$(\mathbf{u})_g = \|u_g\|_{1,V}, \quad (\mathbf{v})_g = \|v_g\|_{1,V} \quad \text{and} \quad (\mathcal{F})_{g,g'} = \overline{F_{g,g'}},$$

which may be rewritten using the Euclidean inner product and bounded as follows:

$$\hat{F}(u, v) \leq \mathbf{u}^T \mathcal{F} \mathbf{v} = (\mathbf{u}, \mathcal{F} \mathbf{v})_{\mathbb{R}^G} \leq \|\mathbf{u}\|_{\mathbb{R}^G} \|\mathcal{F} \mathbf{v}\|_{\mathbb{R}^G} \leq \|\mathcal{F}\|_2 \|\mathbf{u}\|_{\mathbb{R}^G} \|\mathbf{v}\|_{\mathbb{R}^G} \quad (6.4.11)$$

where $\|\mathcal{F}\|_2$ denotes the matrix 2-norm. Finally, we have:

$$\hat{F}(u, v) \leq \|\mathcal{F}\|_2 \|\mathbf{u}\|_{\mathbb{R}^G} \|\mathbf{v}\|_{\mathbb{R}^G} = C_4 \left(\sum_{g=1}^G \|u_g\|_{1,V}^2 \right)^{1/2} \left(\sum_{g=1}^G \|v_g\|_{1,V}^2 \right)^{1/2} = C_4 \|u\|_{\mathcal{W}(V)} \|v\|_{\mathcal{W}(V)} \quad (6.4.12)$$

Well-Posedness

Theorem 6.4.1. *The fixed source variational problem:*

$$\begin{cases} \text{Find } \phi \in \mathcal{W}(V) \text{ which satisfies:} \\ \hat{A}(\phi, v) = \hat{Q}(v) \quad \forall v \in \mathcal{W}(V) \end{cases}. \quad (6.4.13)$$

has a unique solution $\phi \in \mathcal{W}(V)$.

Proof. We have shown in this section that $\hat{A}(\cdot, \cdot)$ is coercive and continuous and that $\hat{Q}(\cdot)$ is also continuous. Therefore we may invoke the Lax-Milgram theorem (theorem 3.6.3) to prove that equation (6.4.13) has a unique solution. $\square$

In the case of the k -eigenvalue problem, we are only interested in the first and largest eigenvalue K_{eff} , and the corresponding eigenvector ϕ . This pair is referred to as the fundamental solution and it is the only eigen-pair with physical significance. This fact reduces the complexity of solving the k -eigenvalue problem as we are only interested in on eigenvalue and it is known that this eigenvalue has a multiplicity of 1.

Theorem 6.4.2. *The eigenvalue variational problem:*

$$\begin{cases} \text{Find } \lambda \in \mathbb{R} \text{ such that there exists a } \phi \in \mathcal{W}(V) \text{ which satisfies:} \\ \hat{A}(\phi, v) = \lambda \hat{F}(\phi, v) \quad \forall v \in \mathcal{W}(V) \end{cases}.$$

has a largest eigenvalue λ_1 and an associated fundamental mode ϕ equal to those for the strong form (equation (6.2.3)).

Proof. We have shown in this section that $\hat{A}(\cdot, \cdot)$ is coercive and continuous and that $\hat{F}(\cdot, \cdot)$ is continuous: the result follows. See [276] chapter 8 for more details. $\square$

6.5 The Virtual Element Discretisation of the NDE

The trial and test functions are defined as follows:

$$v^h = [v_1^h, v_2^h, \dots, v_G^h] \in \mathcal{W}_p(V), \quad \phi^h = [\phi_1^h, \phi_2^h, \dots, \phi_G^h] \in \mathcal{W}_p(V)$$

where $\mathcal{W}_p(V) := [\mathcal{V}_p(V)]^G$. Therefore, the functions v_g^h and ϕ_g^h have the following expansion in the global VE basis:

$$v_g^h = \sum_{i=1}^{N_{\text{gdof}}} \text{dof}_i(v_g^h) \varphi_i, \quad \phi_g^h = \sum_{i=1}^{N_{\text{gdof}}} \text{dof}_i(\phi_g^h) \varphi_i$$

For each function ϕ_g^h, v_g^h we denote the global DoF vectors with

$$\mathbf{v}_g = [\text{dof}_1(v_g^h), \dots, \text{dof}_{N_{\text{gdof}}}(v_g^h)], \quad \boldsymbol{\phi}_g = [\text{dof}_1(\phi_g^h), \dots, \text{dof}_{N_{\text{gdof}}}(\phi_g^h)].$$

To discretise the variational problems of equations (6.4.2) and (6.4.3) we must derive computable VE bilinear and linear forms $\hat{L}^h(\cdot, \cdot)$, $\hat{S}^h(\cdot, \cdot)$, $\hat{Q}^h(\cdot)$ and $\hat{F}^h(\cdot, \cdot)$ which approximate their continuous counterparts $\hat{L}(\cdot, \cdot)$, $\hat{S}(\cdot, \cdot)$, $\hat{Q}(\cdot)$ and $\hat{F}(\cdot, \cdot)$. We begin with the discrete removal operator, which is defined as follows:

$$\hat{L}^h(\phi^h, v^h) = \sum_{g=1}^G \left(\sum_{F \in \mathcal{T}_h} B_{g,F}^h(\phi_g^h, v_g^h) + \frac{1}{2} \sum_{e \in \mathcal{E}_V} (\phi_g^h, v_g^h)_{0,e} \right)$$

with

$$\begin{aligned} & B_{g,F}^h(\phi_g^h, v_g^h) \\ &= \\ & D_F^g \left(\Pi_{p-1}^0 \nabla \phi_g^h, \Pi_{p-1}^0 \nabla v_g^h \right)_{0,F} + \Sigma_{r,F}^g \left(\Pi_p^0 \phi_g^h, \Pi_p^0 v_g^h \right)_{0,F} + c_{g,F} S_F \left(\phi_g^h - \Pi_p^\nabla \phi_g^h, v_g^h - \Pi_p^\nabla v_g^h \right) \end{aligned}$$

where $c_{g,F} := (D_F^g + h_F^2 \Sigma_{r,F}^g)$ is a stabilisation scaling parameter for face F and

$$S_F \left(\phi_g^h - \Pi_p^\nabla \phi_g^h, v_g^h - \Pi_p^\nabla v_g^h \right) = \sum_{l=1}^{N_{\text{dof}}^F} \text{dof}_l(\phi_g^h - \Pi_p^\nabla \phi_g^h) \text{dof}_l(v_g^h - \Pi_p^\nabla v_g^h).$$

The scatter bilinear form is given by:

$$\hat{S}^h(\phi^h, v^h) = \sum_{g=1}^G \sum_{g' \neq g}^G \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 \phi_{g'}^h, \Pi_p^0 v_g^h \right)_{0,F},$$

the fission bilinear form is given by:

$$\hat{F}^h(\phi^h, v^h) = \sum_{g=1}^G \sum_{g'=1}^G \sum_{F \in \mathcal{T}_h} \chi_g \nu \Sigma_{f,F}^{g'} \left(\Pi_p^0 \phi_{g'}^h, \Pi_p^0 v_g^h \right)_{0,F}$$

and the fixed source linear form is given by:

$$\hat{Q}^h(v^h) = \sum_{g=1}^G \sum_{F \in \mathcal{T}_h} (\Pi_p^0 v_g^h, q_g)_{0,F}$$

We may now define the discrete VE variational problems. First, the **discrete fixed source** problem:

$$\begin{cases} \text{Find } \phi^h \in \mathcal{W}_p(V) \text{ which satisfies:} \\ \hat{L}^h(\phi^h, v^h) - \hat{S}^h(\phi^h, v^h) = \hat{Q}^h(v^h) \quad \forall v^h \in \mathcal{W}_p(V) \end{cases} . \quad (6.5.1)$$

Second, is the **discrete eigenvalue** problem:

$$\begin{cases} \text{Find } \lambda^h \in \mathbb{R} \text{ such that there exists a } \phi^h \in \mathcal{W}_p(V) \text{ which satisfies:} \\ \hat{L}^h(\phi^h, v^h) - \hat{S}^h(\phi^h, v^h) = \lambda^h \hat{F}^h(\phi^h, v^h) \quad \forall v^h \in \mathcal{W}_p(V) \end{cases} . \quad (6.5.2)$$

6.5.1 Well-Posedness of Discretisation

Letting $\hat{A}^h(u, v) = \hat{L}^h(u, v) - \hat{S}^h(u, v)$ for $u, v \in \mathcal{W}_p(V)$ (we've dropped the h superscript for lighter notation) we shall first prove the following:

- $\hat{A}^h(u, v)$ is continuous: $\hat{A}^h(u, v) \leq C_1 \|u\|_{\mathcal{W}} \|v\|_{\mathcal{W}}$.
- $\hat{A}^h(u, v)$ is coercive: $\hat{A}^h(u, u) \geq C_2 \|u\|_{\mathcal{W}}^2$.
- $\hat{Q}^h(v)$ is continuous: $\hat{Q}^h(v) \leq C_3 \|v\|_{\mathcal{W}}$.
- $\hat{F}^h(v)$ is continuous: $\hat{F}^h(u, v) \leq C_4 \|u\|_{\mathcal{W}} \|v\|_{\mathcal{W}}$.

Before proving the properties listed above we shall briefly recall the stability property (lemma 6.3.5) of the local bilinear form $B_{g,F}^h(\cdot, \cdot)$:

$$\beta_* B_{g,F}(u, u) \leq B_{g,F}^h(u, u) \leq \beta^* B_{g,F}(u, u).$$

where $B_{F,g}(u, v) = D_F^g(\nabla u, \nabla v)_{0,F} + \Sigma_{r,F}^g(u, v)_{0,V}$. We shall also recall that $B_{g,F}^h(u, v)$ is an inner product on $\mathcal{V}_p(F)$ and therefore it satisfies the Schwarz inequality:

$$B_{g,F}^h(u, v) \leq (B_{g,F}^h(u, u))^{1/2} (B_{g,F}^h(v, v))^{1/2}$$

Finally, we also recall the upper bound on the L^2 projection Π_p^0 (lemma 4.4.4):

$$\|\Pi_p^0 u\|_{0,F} \leq \|u\|_{0,F} . \quad (6.5.3)$$

Continuity of $\hat{A}^h(\cdot, \cdot)$

Starting from the definition of $\hat{A}^h(u, v)$ and letting $u, v \in \mathcal{W}_p(V)$ we have:

$$\begin{aligned} \hat{A}^h(u, v) &= \sum_{g=1}^G \left\{ \sum_{F \in \mathcal{T}_h} B_{g,F}^h(u_g, v_g) + \sum_{e \in \Gamma_V} (u_g, v_g)_{0,e} - \sum_{g' \neq g} \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 u_g, \Pi_p^0 v_{g'} \right)_{0,F} \right\} \\ &\leq \sum_{g=1}^G \left\{ \sum_{F \in \mathcal{T}_h} B_{g,F}^h(u_g, v_g) + (u_g, v_g)_{0,\Gamma_V} + \sum_{g' \neq g} \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left| \left(\Pi_p^0 u_g, \Pi_p^0 v_{g'} \right)_{0,F} \right| \right\} \end{aligned} \quad (6.5.4)$$

Using the stability property and the Cauchy-Schwarz inequality we have the following bound for the VE bilinear form (for more details see equation (E.4.2)):

$$\sum_{F \in \mathcal{T}_h} B_{g,F}^h(u_g, v_g) \leq \beta^* \max(\overline{D^g}, \overline{\Sigma_r^g}) \|u_g\|_{1,V} \|v_g\|_{1,V}. \quad (6.5.5)$$

Additionally, the scatter term satisfies:

$$\begin{aligned} \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left| \left(\Pi_p^0 u_g, \Pi_p^0 v_{g'} \right)_{0,F} \right| &\leq \overline{\Sigma_s^{g' \rightarrow g}} \sum_{F \in \mathcal{T}_h} \left\| \Pi_p^0 u_g \right\|_{0,F} \left\| \Pi_p^0 v_{g'} \right\|_{0,F} \leq \overline{\Sigma_s^{g' \rightarrow g}} \sum_{F \in \mathcal{T}_h} \|u_g\|_{0,F} \|v_{g'}\|_{0,F} \\ &\leq \overline{\Sigma_s^{g' \rightarrow g}} \sum_{F \in \mathcal{T}_h} \|u_g\|_{1,F} \|v_{g'}\|_{1,F} \\ &\leq \overline{\Sigma_s^{g' \rightarrow g}} \left(\sum_{F \in \mathcal{T}_h} \|u_g\|_{1,F}^2 \right)^{1/2} \left(\sum_{F \in \mathcal{T}_h} \|v_{g'}\|_{1,F}^2 \right)^{1/2}, \quad (\text{theorem B.0.2}) \\ &= \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|v_{g'}\|_{1,V} \end{aligned} \quad (6.5.6)$$

Substituting equations (6.5.5) and (6.5.6) into equation (6.5.4) and applying the trace inequality to the vacuum term (cf. (6.4.4)) yields:

$$\hat{A}^h(u, v) \leq \sum_{g=1}^G \left\{ \left(\beta^* \max(\overline{D^g}, \overline{\Sigma_r^g}) + C_{tr}^2 \right) \|u_g\|_{1,V} \|v_g\|_{1,V} + \sum_{g' \neq g} \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|v_{g'}\|_{1,V} \right\} \quad (6.5.7)$$

Therefore, using the same notation introduced in equation (6.4.5), with the difference that $(\overline{\mathcal{A}})_{g,g} = \beta^* \max(\overline{D^g}, \overline{\Sigma_r^g}) + C_{tr}^2$, we may write equation (6.5.7) as:

$$\hat{A}^h(u, v) \leq \mathbf{u}^T \overline{\mathcal{A}} \mathbf{v} \leq \left\| \overline{\mathcal{A}} \right\|_2 \|u\|_{\mathcal{W}(V)} \|v\|_{\mathcal{W}(V)}$$

The derivation is identical to that presented in equation (6.4.6).

Coercivity of $\hat{A}^h(\cdot, \cdot)$

To prove the coercivity we shall first recall the result from equation (E.4.1):

$$\sum_{F \in \mathcal{T}_h} B_{g,F}^h(u, u) \geq \beta_* \min(\underline{D}^g, \underline{\Sigma}_r^g) \|u\|_{1,V}^2, \quad \forall u \in \mathcal{V}_p(V) \quad (6.5.8)$$

Next, we must find an upper bound on the scatter terms using the inequality of equation (6.5.3):

$$\begin{aligned} \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 u_g, \Pi_p^0 u_{g'} \right)_{0,F} &\leq \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left\| \Pi_p^0 u_g \right\|_{0,F} \left\| \Pi_p^0 u_{g'} \right\|_{0,F} \\ &\leq \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \|u_g\|_{0,F} \|u_{g'}\|_{0,F} \\ &\leq \overline{\Sigma_s^{g' \rightarrow g}} \sum_{F \in \mathcal{T}_h} \|u_g\|_{1,F} \|u_{g'}\|_{1,F} \\ &\leq \overline{\Sigma_s^{g' \rightarrow g}} \left(\sum_{F \in \mathcal{T}_h} \|u_g\|_{1,F}^2 \right)^{1/2} \left(\sum_{F \in \mathcal{T}_h} \|u_{g'}\|_{1,F}^2 \right)^{1/2} \\ &= \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|u_{g'}\|_{1,V} \end{aligned} \quad (6.5.9)$$

therefore:

$$- \sum_{g' \neq g}^G \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 u_g, \Pi_p^0 u_{g'} \right)_{0,F} \geq - \sum_{g' \neq g}^G \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|u_{g'}\|_{1,V}. \quad (6.5.10)$$

Substituting equations (6.5.8) and (6.5.10) into $\hat{A}^h(u, u)$ gives:

$$\hat{A}^h(u, u) \geq \sum_{g=1}^G \left\{ \beta_* \min(\underline{D}^g, \underline{\Sigma}_r^g) \|u_g\|_{1,V}^2 - \sum_{g' \neq g}^G \overline{\Sigma_s^{g' \rightarrow g}} \|u_g\|_{1,V} \|u_{g'}\|_{1,V} \right\}$$

We may now bound $\hat{A}^h(u, u)$ in exactly the same way we bounded $\hat{A}(u, u)$. The only difference being that the diagonal elements are given by: $(\underline{\mathcal{A}})_{g,g} = \beta_* \min(\underline{D}^g, \underline{\Sigma}_r^g)$ (c.f equation (6.4.10)). Therefore, verifying that $\hat{A}^h(\cdot, \cdot)$ is coercive again reduces to verifying $\underline{\mathcal{A}}$ is positive-definite.

Continuity of $\hat{Q}^h(\cdot)$

Next, we prove the continuity of $\hat{Q}^h(\cdot)$ with respect to the $\|\cdot\|_{\mathcal{W}}$:

$$\begin{aligned} \hat{Q}^h(v) &= \sum_{g=1}^G \sum_{F \in \mathcal{T}_h} \left(\Pi_p^0 v_g, q_g \right)_{0,F} \leq \sum_{g=1}^G \sum_{F \in \mathcal{T}_h} \left\| \Pi_p^0 v_g \right\|_{0,F} \|q_g\|_{0,F} \\ &\leq \sum_{g=1}^G \sum_{F \in \mathcal{T}_h} \|v_g\|_{1,F} \|q_g\|_{0,F} \leq \sum_{g=1}^G \left(\sum_{F \in \mathcal{T}_h} \|q_g\|_{0,F}^2 \right)^{1/2} \left(\sum_{F \in \mathcal{T}_h} \|v_g\|_{1,F}^2 \right)^{1/2} \end{aligned}$$

$$= \sum_{g=1}^G \|q_g\|_{0,V} \|v_g\|_{1,V} \leq \left(\sum_{g=1}^G \|q_g\|_{0,V}^2 \right)^{1/2} \left(\sum_{g=1}^G \|v_g\|_{1,V}^2 \right)^{1/2} = D_3 \|v\|_{\mathcal{W}}$$

Continuity of $\hat{F}^h(\cdot, \cdot)$

Finally, we shall prove the continuity of the fission bilinear form with respect to $\|\cdot\|_{\mathcal{W}}$. Letting $F_{g,g'}^F = \chi_g \nu \Sigma_{f,F}^{g'}$ and following the same steps we followed for the scatter terms (equation (6.5.9)) we have:

$$\hat{F}^h(u, v) = \sum_{g=1}^G \sum_{g'=1}^G \sum_{F \in \mathcal{T}_h} F_{g,g'}^F (\Pi_p^0 u_g, \Pi_p^0 v_{g'})_{0,F} \leq \sum_{g=1}^G \sum_{g'=1}^G \overline{F_{g,g'}^F} \|u_g\|_{1,V} \|v_{g'}\|_{1,V}.$$

Performing exactly the same steps we did to bound $\hat{F}(u, v)$ (see equations (6.4.11) and (6.4.12)) we have: $\hat{F}^h(u, v) \leq D_4 \|u\|_{\mathcal{W}} \|v\|_{\mathcal{W}}$ with $D_4 = \|\mathcal{F}\|_2$.

Well Posedness

Theorem 6.5.1. *The fixed source discrete variational problem:*

$$\begin{cases} \text{Find } \phi^h \in \mathcal{W}_p(V) \text{ which satisfies:} \\ \hat{A}^h(\phi^h, v^h) = \hat{Q}^h(v^h) \quad \forall v^h \in \mathcal{W}_p(V) \end{cases}. \quad (6.5.11)$$

as a unique solution $\phi^h \in \mathcal{W}_p(V)$.

Proof. We have shown in this section that $\hat{A}^h(\cdot, \cdot)$ is coercive and continuous and that $\hat{Q}^h(\cdot)$ is also continuous. Therefore we may invoke the Lax-Milgram theorem (theorem 3.6.3) to prove that equation (6.5.11) has a unique solution. $\square$

Theorem 6.5.2. *The eigenvalue variational problem:*

$$\begin{cases} \text{Find } \lambda^h \in \mathbb{R} \text{ such that there exists a } \phi^h \in \mathcal{W}_p(V) \text{ which satisfies:} \\ \hat{A}^h(\phi^h, v^h) = \lambda^h \hat{F}^h(\phi^h, v^h) \quad \forall v^h \in \mathcal{W}_p(V) \end{cases}.$$

has a largest eigenvalue λ_1^h and fundamental mode ϕ^h which approximate the largest eigenvalue and fundamental mode of the continuous problem (equation (6.2.3)).

Proof. We have shown in this section that $\hat{A}^h(\cdot, \cdot)$ is coercive and continuous and that $\hat{F}^h(\cdot)$ is also continuous, the result follows, see [276] chapter 8 for more details. $\square$

6.6 Practical Implementation

In this section we shall explain how we implemented various key aspects of the NECG-VEM discretisation of the multigroup NDE. These aspects include the enumeration and retrieval of NECG-VEM degrees of freedom on the mesh, the assembly of global matrices for the removal, scatter and fission operators, the strategy for solving the global system of equations using source-iteration (SI) for the fixed source problem and power-iteration (PI) for the k -eigenvalue problem and finally we present the distributed-memory parallelisation strategy using PETSc [277] and MPI [278, 279].

6.6.1 Global NECG-VEM Degrees of Freedom

In this section we shall explain how the mapping from topological entities (vertices, half-edges and faces) to degrees of freedom is performed in the case of the NECG-VEM. Letting N_v , N_e and N_f be the number of vertices, edges and faces in the mesh $\mathcal{T}_h$ respectively, and noting that an edge is defined as a twin-pair of half-edges in the DCEL, then the dimension of $\mathcal{V}_p(V)$ (and the number of global degrees of freedom) is given by:

$$N_{\text{dof}} = N_v + (p - 1)N_e + n_{p-2}N_f.$$

The question remains as to how to initially distribute the degrees of freedom on the mesh. The ideal DoF ordering is problem-dependent: the ordering which optimises some important property of the global system of equations, for example matrix bandwidth, for one type of problem may not be the ideal for another. An additional factor which one must consider when numbering the DoFs is the parallelisation strategy: an ordering which makes sense in a distributed-memory implementation may be sub-optimal in a shared-memory implementation. Therefore, we choose an initial DoF ordering which does not seek to attain some desired global matrix structure but rather facilitates the easy mapping between topological entities (vertices, edges and faces) and degrees of freedom. This ordering, which we shall call the **mesh numbering system** (MNS) makes it easy to perform common operations such as retrieving all of the DoFs on the closure of a face (this means the face itself, which is considered to be an open set in $\mathbb{R}^2$, its bounding half-edges and its vertices) or finding all of the degrees of freedom which are coupled to a degree of freedom. The MNS is defined as follows:

- The degrees of freedom attached to vertices are numbered first, with DoFs belonging to vertex v numbered contiguously. Let us denote the number of DoFs per vertex with N_{dof}^v , in the present case this value is 1 but for vector-valued problems it may be more, then the DoFs attached to vertex v are:

$$vN_{\text{dof}}^v \rightarrow (v + 1)N_{\text{dof}}^v - 1,$$

for all $v = 0, \dots, N_v - 1$ (note the 0-based indexing).

- The degrees of freedom attached to half-edges are numbered second. As with the vertices, the DoFs belonging to each half-edge are numbered contiguously and we denote the number of DoFs per edge to be N_{dof}^e (in this case $N_{\text{dof}}^e = p - 1$). In section 5.4 we defined each pair of twin half-edges to be even-odd pairs, for example half-edges he_0 and he_1 will be twins and so will he_{100} and he_{101} . Therefore, on the even member of each pair we set the edge ID to be $e = he/2$ and we assign the following degrees of freedom to the even half-edge:

$$N_v N_{\text{dof}}^v + e N_{\text{dof}}^e \rightarrow N_v N_{\text{dof}}^v + (e + 1) N_{\text{dof}}^e - 1$$

and we assign the same DoFs to the odd half-edge but in reverse:

$$N_v N_{\text{dof}}^v + (e + 1) N_{\text{dof}}^e - 1 \rightarrow N_v N_{\text{dof}}^v + e N_{\text{dof}}^e.$$

This is how we encode shared DoFs between adjacent faces sharing an edge.

- Finally, we number the DoFs attached to faces. These correspond to internal DoFs: the number of DoFs per face is $N_{\text{dof}}^f = n_{p-2}$. Recalling that in the DCEL, we define face 0 to be the unbounded face. For the purpose of distributing DoFs we only consider bounded faces, so to maintain 0-indexing we decrement the IDs of each bounded face by 1. With this in mind, we number the face DoFs for face f as follows:

$$N_v N_{\text{dof}}^v + N_e N_{\text{dof}}^e + f N_{\text{dof}}^f \rightarrow N_v N_{\text{dof}}^v + N_e N_{\text{dof}}^e + (f + 1) N_{\text{dof}}^f - 1$$

In figure 6.1 we present a simple example of a mesh with two faces and with a NECG-VEM scheme of order $p = 3$ defined on it.

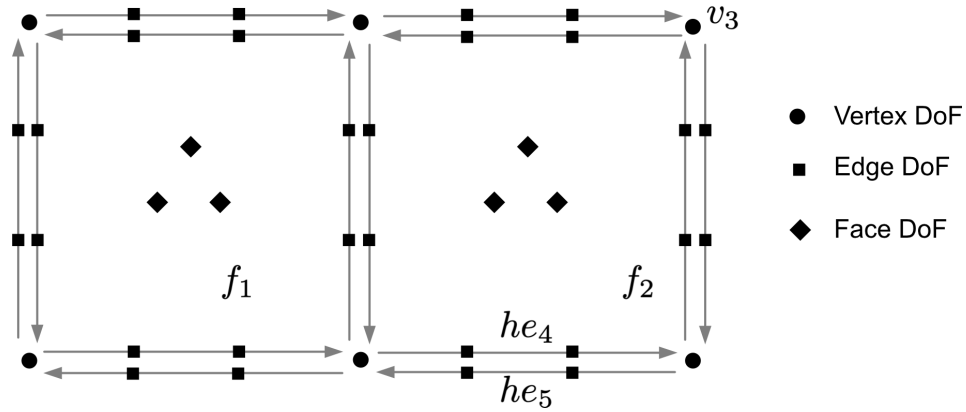


Figure 6.1: A small mesh with two faces and the degrees of freedom for a NECG-VEM scheme of order $p = 3$.

Letting $\text{dofset}(\cdot)$ be an operator which returns the DoFs on a vertex, half-edge or face, then in this example we would have the following mappings:

$$\begin{aligned}
\text{dofset}(v_3) &= \{3\}, \\
\text{dofset}(he_4) &= \{14, 15\}, \quad \text{dofset}(he_5) = \{15, 14\}, \\
\text{dofset}(f_1) &= \{20, 21, 22\},
\end{aligned}$$

where the DoFs also have 0-indexing.

A fundamental operation required for matrix assembly is the retrieval of the degrees of freedom on the closure of a face. The retrieved global DoFs must be in the same order as the local DoFs: where vertices are numbered first in counter-clockwise order, followed by half-edges also in counter-clockwise order and finally the face DoFs (see figure 4.4). We present this operation in algorithm 17.

Another fundamental operation is the retrieval of all DoFs which are coupled to a DoF. The DoF-to-DoF coupling determines the non-zeros on a given row of the global matrix, for example if DoF i is coupled to DoFs j, k, l then the row i will have non-zeros at columns j, k, l . This information is required for memory preallocation of global matrices and to cheaply set up the global matrix sparsity pattern without performing a full assembly procedure. The coupling between DoFs depends upon whether a given DoF belongs to a vertex, half-edge or face. If a DoF belongs to a vertex then it is coupled to the DoFs belonging to all faces which share that vertex. If a DoF belongs to a half-edge he then it is coupled to the DoFs of the faces $\text{face}(he)$ and $\text{face}(\text{twin}(he))$. Finally, if a DoF belongs to a face then it is only coupled to the DoFs of that face. A key benefit of the MNS is that the mapping from DoFs back to topological entities is cheap to compute. We define an operator $\text{topo}(\cdot)$ which takes a DoF as an argument and returns a pair of integers (i_1, i_2) . Integer i_1 denotes the type of topological entity (0 for vertex, 1 for half-edge and 2 for face) and i_2 denotes the index of the topological entity. The output of $\text{topo}(\cdot)$ is computed as follows: if $i < N_v N_{\text{dof}}^v$ the DoF belongs to a vertex so we set i_1 and i_2 to

$$i_1 = 0, \quad i_2 = \text{floor} \left(\frac{i}{N_{\text{dof}}^v} \right),$$

where the vertex ID is i_2 . If $N_v N_{\text{dof}}^v \leq i < (N_v N_{\text{dof}}^v + N_e N_{\text{dof}}^e)$ then the DoF belongs to two twin half edges:

$$i_1 = 1, \quad i_2 = \text{floor} \left(\frac{i - N_v N_{\text{dof}}^v}{N_{\text{dof}}^e} \right),$$

where the twin half-edges have index $2i_2$ and $2i_2 + 1$. If $(N_v N_{\text{dof}}^v + N_e N_{\text{dof}}^e) \leq i < (N_v N_{\text{dof}}^v + N_e N_{\text{dof}}^e + N_f N_{\text{dof}}^f)$ then the DoF belongs to a face

$$i_1 = 2, \quad i_2 = \text{floor} \left(\frac{i - N_v N_{\text{dof}}^v - N_e N_{\text{dof}}^e}{N_{\text{dof}}^f} \right)$$

where the face has index i_2 . We present the DoF-to-DoF mapping in algorithm 18.

Algorithm 17 Algorithm to return all of the DoFs on the closure of a face.

procedure FACEDOFS(f)**Input:** A face ID f .**Output:** An array **facedofs** holding the degrees of freedom on the face.Initialise empty array: **facedofs** $\leftarrow []$

Find the DoFs on the vertices first...

 $he_1 \leftarrow \text{outerHedges}(f)[0]$ $he_2 \leftarrow he_1$ **do** $v \leftarrow \text{origin}(he_2).$ Find DoFs on v using $\text{dofset}(v)$ and append to **facedofs**. $he_2 \leftarrow \text{next}(he_2).$ **while** $he_2 \neq he_1$

Find the DoFs on the half-edges second...

 $he_1 \leftarrow \text{outerHedges}(f)[0]$ $he_2 \leftarrow he_1$ **do**Find DoFs on he_2 using $\text{dofset}(he_2)$ and append to **facedofs**. $he_2 \leftarrow \text{next}(he_2).$ **while** $he_2 \neq he_1$

Find the DoFs on the face last...

Find DoFs on face using $\text{dofset}(f)$ and append to **facedofs**.**Return:** The DoF array **facedofs**.**end procedure**

Algorithm 18 Algorithm to return all of the DoFs coupled to a DoF

procedure COUPLEDDOFS(i)

Inputs: A DoF index i .

Outputs: An array of coupled DoFs **dofs** sorted in ascending order.

Initialise empty array: **dofs** $\leftarrow []$

Find topological entity the DoF belongs to: $(i_1, i_2) = \text{topo}(i)$

if $i = 0$ **then**

 Set vertex ID $v \leftarrow i_2$.

 Find set of faces which share vertex

 Find set union of DoFs for all these faces and assign to **dofs**

else if $i = 1$ **then**

 Set twin half-edge IDs $he_1 \leftarrow 2i_2$ and $he_2 \leftarrow 2i_2 + 1$.

 Find incident faces $f_1 = \text{face}(he_1)$ and $f_2 = \text{face}(he_2)$

 Find union of DoFs: **dofs** = FaceDofs(f_1) $\cup$ FaceDofs(f_2)

else if $i = 2$ **then**

 Set the face ID $f = i_2$

 Find set of DoFs on face: **dof** = FaceDofs(f)

end if

Return: The array **dof**.

end procedure

6.6.2 The Global Matrices

Each of the discrete bilinear and linear forms $\hat{L}^h(\cdot, \cdot)$, $\hat{S}^h(\cdot, \cdot)$, $\hat{F}^h(\cdot, \cdot)$ and $\hat{Q}^h(\cdot)$ have an equivalent matrix or vector form which we shall derive here. Let $u^h, v^h \in \mathcal{W}_p$ and let us recall the property that for any $v_g^h \in \mathcal{V}_p(V)$ we have on every face $v_g^h|_F \in \mathcal{V}_p(F)$ and on every edge e : $v_g^h|_e \in \tilde{\mathbb{P}}_p(e)$. Therefore, v_g^h may be expanded in the local basis of face F and of edge e :

$$\begin{aligned} v_g^h|_F &= \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(v_g^h) \varphi_i = \sum_{i=1}^{N_{\text{dof}}^F} (\mathbf{v}_{g,F})_i \varphi_i, \\ v_g^h|_e &= \sum_{i=1}^{N_{\text{dof}}^e} \text{dof}_i(v_g^h) \varphi_i = \sum_{i=1}^{N_{\text{dof}}^e} (\mathbf{v}_g^e)_i \varphi_i \end{aligned}$$

where $\mathbf{v}_{g,F}$, $\mathbf{v}_g^e$ are the vectors of local degrees of freedom of function v_g^h on face F and edge e respectively. Substituting these local VE expansions into $B_{g,F}^h(\cdot, \cdot)$ yields:

$$\begin{aligned} B_{g,F}^h(u_g^h, v_g^h) &= \sum_{i=1}^{N_{\text{dof}}^F} \sum_{j=1}^{N_{\text{dof}}^F} \text{dof}_i(u_g^h) \text{dof}_j(v_g^h) B_{g,F}^h(\varphi_i, \varphi_j) \\ &= (\mathbf{u}_{g,F})^T \mathbf{B}_{g,F} \mathbf{v}_{g,F}. \end{aligned}$$

Applying the formulae we presented in section 4.5.6 we may write the local bilinear form as the following matrix.

$$\mathbf{B}_{g,F} = D_F^g \mathbf{K}_F + \Sigma_{r,F}^g \mathbf{M}_F + c_{g,F} \mathbf{S}_F.$$

Applying similar steps to the edge mass-matrices gives:

$$(\Pi_p^0 u_g^h, \Pi_p^0 v_g^h)_{0,e} = \sum_{i=1}^{N_{\text{dof}}^e} \sum_{j=1}^{N_{\text{dof}}^e} \text{dof}_i(u_g^h) \text{dof}_j(v_g^h) (\Pi_p^0 \varphi_i, \Pi_p^0 \varphi_j)_{0,e} = (\mathbf{u}_g^e)^T \mathbf{M}_e \mathbf{v}_g^e.$$

Following the same procedure for the local bilinear forms for scatter and fission gives: With the local removal matrices defined we may now derive the global removal matrices in equation (6.6.1):

$$\begin{aligned} \hat{L}^h(u^h, v^h) &= \sum_{g=1}^G \left\{ \sum_{F \in \mathcal{T}_h} B_{g,F}^h(u_g^h, v_g^h) + \frac{1}{2} \sum_{e \in \Gamma_V} (u_g^h, v_g^h)_{0,e} \right\} \\ &= \sum_{g=1}^G \left\{ \sum_{F \in \mathcal{T}_h} (\mathbf{u}_{g,F})^T \mathbf{B}_{g,F} \mathbf{v}_{g,F} + \frac{1}{2} \sum_{e \in \Gamma_V} (\mathbf{u}_g^e)^T \mathbf{M}_e \mathbf{v}_g^e \right\} \\ &= \sum_{g=1}^G \left\{ (\mathbf{u}_g)^T \left(\sum_{F \in \mathcal{T}_h} \mathbf{B}_{g,F} + \frac{1}{2} \sum_{e \in \Gamma_V} \mathbf{M}_e \right) \mathbf{v}_g \right\} \\ &= \sum_{g=1}^G (\mathbf{u}_g)^T \mathbf{L}_g \mathbf{v}_g. \end{aligned} \tag{6.6.1}$$

The local scatter and fission matrices are given by:

$$\begin{aligned}\Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 u_g^h, \Pi_p^0 v_g^h \right)_{0,F} &= (\mathbf{u}_{g,F})^T \left(\Sigma_{s,F}^{g' \rightarrow g} \mathbf{M}_F \right) \mathbf{v}_{g,F} \\ \chi_g \nu \Sigma_{f,F}^{g'} \left(\Pi_p^0 u_g^h, \Pi_p^0 v_g^h \right)_{0,F} &= (\mathbf{u}_{g,F})^T \left(\chi_g \nu \Sigma_{f,F}^{g'} \mathbf{M}_F \right) \mathbf{v}_{g,F}.\end{aligned}$$

Therefore, the global scatter and fission matrices may be expressed as:

$$\begin{aligned}\hat{S}^h(u^h, v^h) &= \sum_{g=1}^G \sum_{g' \neq g}^G \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 u_g^h, \Pi_p^0 v_{g'}^h \right)_{0,F} \\ &= \sum_{g=1}^G \sum_{g' \neq g}^G (\mathbf{u}_g)^T \left(\sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} \mathbf{M}_F \right) \mathbf{v}_{g'} = \sum_{g=1}^G \sum_{g' \neq g}^G (\mathbf{u}_g)^T \mathbf{S}_{g,g'} \mathbf{v}_{g'},\end{aligned}$$

and

$$\begin{aligned}\hat{F}^h(u^h, v^h) &= \sum_{g=1}^G \sum_{g'=1}^G \sum_{F \in \mathcal{T}_h} \chi_g \nu \Sigma_{f,F}^{g'} \left(\Pi_p^0 u_g^h, \Pi_p^0 v_{g'}^h \right)_{0,F} \\ &= \sum_{g=1}^G \sum_{g'=1}^G (\mathbf{u}_g)^T \left(\sum_{F \in \mathcal{T}_h} \chi_g \nu \Sigma_{f,F}^{g'} \mathbf{M}_F \right) \mathbf{v}_{g'} = \sum_{g=1}^G \sum_{g'=1}^G (\mathbf{u}_g)^T \mathbf{F}_{g,g'} \mathbf{v}_{g'}.\end{aligned}$$

Finally, we have the vector form of the local fixed sources:

$$\left(\Pi_p^0 u_g^h, q_g(\mathbf{x}) \right)_{0,F} = \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(u_g^h) \left(\Pi_p^0 \varphi_i, q_g(\mathbf{x}) \right)_{0,F} = (\mathbf{u}_{g,F})^T \mathbf{q}_{g,F}.$$

Therefore, the global fixed source vectors are given by:

$$\hat{Q}^h(v^h) = \sum_{g=1}^G \sum_{F \in \mathcal{T}_h} \left(\Pi_p^0 v_g^h, q_g(\mathbf{x}) \right)_{0,F} = \sum_{g=1}^G \sum_{F \in \mathcal{T}_h} (\mathbf{v}_{g,F})^T \mathbf{q}_{g,F} = \sum_{g=1}^G (\mathbf{v}_g)^T \left(\sum_{F \in \mathcal{T}_h} \mathbf{q}_{g,F} \right) = \sum_{g=1}^G (\mathbf{v}_g)^T \mathbf{q}_g$$

6.6.3 The Fixed Source Problem

Substituting each of the expressions for the global matrices into the fixed source discrete VE variational problem of equation (6.5.1) yields the following, equivalent matrix problem:

$$\sum_{g=1}^G (\mathbf{v}_g)^T \left(\mathbf{L}_g \phi_g - \sum_{g' \neq g}^G \mathbf{S}_{g,g'} \phi_{g'} - \mathbf{q}_g \right) = 0 \quad (6.6.2)$$

The vector $\mathbf{v}_g$ is arbitrary, therefore equation (6.6.2) can must hold for any $\mathbf{v}_g$. Consequently, the term in brackets in equation (6.6.2) must be equal to the $N_{\text{gdof}} \times 1$ zero vector for all g .

Therefore, equation (6.6.2) may be written as:

$$\begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{L}_G \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} - \begin{bmatrix} \mathbf{0} & \mathbf{S}_{1,2} & \dots & \mathbf{S}_{1,3} \\ \mathbf{S}_{2,1} & \mathbf{0} & \dots & \mathbf{S}_{2,3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{3,1} & \mathbf{S}_{3,2} & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_G \end{bmatrix}. \quad (6.6.3)$$

which may also be written as: $(\mathbf{L} - \mathbf{S})\phi = \mathbf{q}$. An obvious but inefficient way to solve equation (6.6.3) would be to assemble the full non-symmetric operator $\mathbf{A} = \mathbf{L} - \mathbf{S}$ and use a Krylov subspace method (KSM) for non-symmetric systems such as a GMRES [280]. This approach is impractical for two reasons, the first is that the block-matrix $\mathbf{A}$, although sparse, has dimensions $(G N_{\text{gdof}}) \times (G N_{\text{gdof}})$ and so may become very large for large problems. Storing $\mathbf{A}$ explicitly could therefore exhaust memory resources even on modern hardware. Second, in the case of the restarted GMRES, a pre-defined number of global vectors would need to be stored in order to construct the Krylov subspace. These vectors would each have dimension $G \times N_{\text{gdof}}$ and would also be memory-intensive to store. An alternative to this method, and the most common way to solve this problem, is source iteration (SI). Let $\phi_g^{(i)}$ denote the i 'th iterate of the group g global flux vector. In the SI, the scalar neutron fluxes are first initialised to zero scalar neutron flux $\phi_g^{(0)} = \mathbf{0}$. The SI then proceeds as follows:

$$\phi_g^{(i+1)} = \mathbf{L}_g^{-1} \left(\sum_{g' < g} \mathbf{S}_{gg'} \phi_{g'}^{(i+1)} + \sum_{g' > g} \mathbf{S}_{gg'} \phi_{g'}^{(i)} + \mathbf{q}_g \right). \quad (6.6.4)$$

If there is no up-scatter then equation (6.6.4) will converge after a single outer iteration. This is because the scattering source for each energy group is a linear combination of scalar neutron fluxes that have already been computed. In the case where up-scatter is present, a suitable convergence criteria must be defined. In the present work we choose the following criteria:

$$\max_g(\delta_g) < \epsilon \quad \text{where} \quad \delta_g = \frac{\|\phi_g^{(i+1)} - \phi_g^{(i)}\|_{\mathbb{R}^{N_{\text{gdof}}}}}{\|\phi_g^{(i)}\|_{\mathbb{R}^{N_{\text{gdof}}}}}. \quad (6.6.5)$$

The value of the tolerance, ϵ , is problem-dependent, therefore we shall state its value when specifying each numerical test case later in this chapter. The removal operators $\mathbf{L}_g$ are symmetric positive-definite (SPD), therefore they may be solved using the preconditioned conjugate gradient method (PCG). The PCG does not require the storage of a new vector after each PCG iteration and the amount of computation performed inside each PCG iteration does not depend on the PCG iteration count [281]. This is in stark contrast to GMRES, which both stores a vector after each iteration and requires more computation at each iteration. Another major consideration is the ease with which the operator $\mathbf{L}_g$ may be preconditioned: the reaction-diffusion problem is a canonical problem in the finite element literature and many efficient preconditioners have been developed for it. A full performance comparison of each type of pre-

conditioner for inverting the removal operators is beyond the scope of this thesis. However, as a matter of trial and error, we have found that both the Hypr BoomerAMG [282] and PETSc's own algebraic multigrid algorithms perform well for VEM orders $p = 1, 2$. They perform less well for higher orders, $p = 3, 4$, however they are still superior to preconditioners based on incomplete factorisations such as the incomplete LU factorisation (ILU). We present the full SI procedure for the NDE in algorithm 19. We define the parameter `nfg` to denote the "number of fast groups", which is the number of energy groups which do not have any up-scatter source. For problems with no up-scatter we simply set `nfg` = G .

Algorithm 19 The SI procedure for the NDE.

1: Initialise $\phi_g^{(0)} = \mathbf{0}$ for all $g = 1, \dots, G$.

2: **for** $g = 1, \dots, \text{nfg}$ **do**

3: Compute group g down-scatter source:

$$\mathbf{q}_{s,g}^\downarrow = \sum_{g'=1}^{g-1} \mathbf{S}_{g,g'} \phi_{g'}$$

4: Compute the group g fixed source $\mathbf{q}_g$

5: Solve:

$$\mathbf{L}_g \phi_g = \mathbf{q}_{s,g}^\downarrow + \mathbf{q}_g$$

6: **end for**

7: Compute fast-to-thermal scatter sources for all $g = \text{nfg} + 1, \dots, G$

$$\mathbf{q}_g^{fast} = \sum_{g'=1}^{\text{nfg}} \mathbf{S}_{g,g'} \phi_{g'}$$

8: **for** up-scatter iteration $l = 1$ **to** max_iter **do**

9: **for** $g = \text{nfg} + 1, \dots, G$ **do**

10: Compute group g thermal down-scatter source:

$$\mathbf{q}_{s,g}^\downarrow = \sum_{g'=\text{nfg}+1}^g \mathbf{S}_{g,g'} \phi_{g'}^{(l+1)}$$

11: Compute group g thermal up-scatter source:

$$\mathbf{q}_{s,g}^\uparrow = \sum_{g'=g+1}^G \mathbf{S}_{g,g'} \phi_{g'}^{(l)}$$

12: Compute the group g fixed source $\mathbf{q}_g$

13: Solve:

$$\mathbf{L}_g \phi_g^{(l+1)} = \mathbf{q}_g^{fast} + \mathbf{q}_{s,g}^\downarrow + \mathbf{q}_{s,g}^\uparrow + \mathbf{q}_g$$

14: **end for**

15: Test for convergence using equation (6.6.5):

$$\max_{g=\text{nfg}+1, \dots, G} \frac{\|\phi_g^{(i+1)} - \phi_g^{(i)}\|}{\|\phi_g^{(i)}\|} < \epsilon$$

16: **end for**

6.6.4 The k -Eigenvalue Problem

Substituting each of the expressions for the global matrices into the k -eigenvalue discrete VE variational problem of equation (6.5.2) yields the following, equivalent matrix problem:

$$\sum_{g=1}^G (\mathbf{v}_g)^T \left(\mathbf{L}_g \phi_g - \sum_{g' \neq g}^G \mathbf{S}_{g,g'} \phi_{g'} - \lambda^h \sum_{g'=1}^G \mathbf{F}_{g,g'} \phi_{g'} \right) = 0 \quad (6.6.6)$$

As was the case with the fixed source matrix problem (equation (6.6.2)) the vector $\mathbf{v}_g$ is arbitrary, therefore the term in brackets must be equal to the zero-vector if equation (6.6.6) is to hold in all cases. Therefore we have the following system of equations:

$$\begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{L}_G \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} - \begin{bmatrix} \mathbf{0} & \mathbf{S}_{1,2} & \dots & \mathbf{S}_{1,3} \\ \mathbf{S}_{2,1} & \mathbf{0} & \dots & \mathbf{S}_{2,3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{3,1} & \mathbf{S}_{3,2} & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} = \lambda^h \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} & \dots & \mathbf{F}_{1,3} \\ \mathbf{F}_{2,1} & \mathbf{F}_{2,2} & \dots & \mathbf{F}_{2,3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}_{3,1} & \mathbf{F}_{3,2} & \dots & \mathbf{F}_{3,3} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix}, \quad (6.6.7)$$

which we may write equivalently as: $(\mathbf{L} - \mathbf{S})\phi = \lambda^h \mathbf{F}\phi$. Equation (6.6.7) is a generalised eigenvalue problem, the equivalent standard eigenvalue problem would be

$$(\mathbf{L} - \mathbf{S})^{-1} \mathbf{F} \phi = \gamma^h \phi \quad \text{where} \quad \gamma^h = 1/\lambda^h$$

The most common method of solving equation (6.6.7) for the largest eigenvalue ($\gamma_1^h = K_{\text{eff}}^h$), is the power iteration (PI). This is the method we have implemented in this thesis. Let $\phi_g^{(i)}$ denote the i 'th iterate of the group g global flux vector and $K_{\text{eff}}^{(i)}$ the i 'th iterate of the effective multiplication factor. The power iteration begins by initialising the fluxes to some initial flux profile, in this case unit neutron scalar flux $\phi_g^{(0)} = \mathbf{1}$ and $K_{\text{eff}}^{(0)} = 1$. The iterative procedure then follows equation (6.6.8) below [102]:

$$\phi_g^{(i+1)} = \mathbf{L}_g^{-1} \left(\frac{1}{K_{\text{eff}}^{(i)}} \sum_{g'=1}^G \mathbf{F}_{gg'} \phi_{g'}^{(i)} + \sum_{g' < g} \mathbf{S}_{gg'} \phi_{g'}^{(i+1)} + \sum_{g' > g} \mathbf{S}_{gg'} \phi_{g'}^{(i)} \right). \quad (6.6.8)$$

After each full power iteration a new set of fluxes will have been computed:

$$\{\phi_1^{(i+1)}, \phi_2^{(i+1)}, \dots, \phi_G^{(i+1)}\}.$$

After each power iteration the next iterate of the effective multiplication factor $K_{\text{eff}}^{(i+1)}$ may be computed as follows

$$K_{\text{eff}}^{(i+1)} = K_{\text{eff}}^{(i)} \frac{\int_V S_f^{(i+1)} dx}{\int_V S_f^{(i)} dx} \quad \text{where} \quad S_f^{(i)} = \sum_{g'=1}^G \nu \Sigma_f^{g'} \phi_g^{h,(i)}$$

Following [92] the fission source integrals were computed as follows:

$$\int_V \sum_{g'=1}^G \nu \Sigma_f^{g'} \phi_{g'}^h = \sum_{g'=1}^G \sum_{F \in \mathcal{T}_h} \nu \Sigma_{f,F}^{g'} \int_F \phi_{g'}^h dx,$$

where each flux integral in face F is computed as follows:

$$\int_F \phi_{g'}^h dx = \begin{cases} \int_F \Pi_p^0 \phi_{g'}^h dx & \text{if } p = 1 \\ |F| \text{dof}_{pN_v^F+1}(\phi_{g'}^h) & \text{if } p > 1 \end{cases}. \quad (6.6.9)$$

The iteration procedure is continued until the convergence criteria presented in equation (6.6.5) is satisfied. The convergence criteria for the PCG solver (PETScs own implementation) was altered throughout the power iteration in order to improve the rate of convergence. Given a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, then for for PCG iteration i the residual is given by $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$. The PCG iterations are continued until the following convergence criteria are met [277]:

$$\|\mathbf{r}_i\| < \max(r_{\text{tol}} \|\mathbf{b}\|, a_{\text{tol}})$$

The absolute tolerance, a_{tol} , was held constant at the PETSc default value of 1×10^{-50} . The value of the relative tolerance r_{tol} was varied as the power iteration progressed. Each removal operator matrix $\mathbf{L}_g$ had its own relative tolerance, denoted r_{tol}^g , which was computed as follows:

$$r_{\text{tol}}^g = \min(0.1 \delta_g, 5 \times 10^{-3}), \quad (6.6.10)$$

where δ_g is given in equation (6.6.5). Therefore, r_{tol}^g is capped at a maximum value of 5×10^{-3} and decreases as the relative difference in flux iterates decreases, thus providing a more accurate solution as the power iteration approaches convergence. Varying r_{tol}^g throughout the power iteration did not effect the final solution and would greatly improve the runtime to convergence; especially for the higher VEM orders where the number of PCG iterations can grow large. The expression for r_{tol}^g given in equation (6.6.10) was simply a heuristic based on trial and error, not the result of a rigorous attempt to find the largest admissible r_{tol}^g . Upon completion of the power iteration the neutron scalar flux is typically normalised such that the fission source is unity:

$$\frac{1}{K_{\text{eff}}} \int_V \sum_{g=1}^g \nu \Sigma_f^g(\mathbf{x}) \phi_g(\mathbf{x}) dx = 1.$$

However, the choice of neutron scalar flux normalisation is arbitrary and different benchmark problems will often specify different normalisations. We present the full PI procedure in algorithm 20. We chose to implement a simpler form of the PI where we do not do any additional inner iterations within each power iteration to converge the scatter source.

Algorithm 20 The PI procedure for the NDE.

- 1: Initialise $\phi_g^{(0)} = \mathbf{1}$ for $g = 1, \dots, G$.
- 2: Initialise $K_{\text{eff}}^{(0)} = 1$.
- 3: Initialise the fission integral:

$$I_f^{(0)} = \int_V \left(\sum_{g=1}^G \nu \Sigma_f^g \phi_g^{(0)} \right) dx$$

- 4: **for** Outer iteration $l = 0 = 1, \dots, \text{niter_max}$ **do**
- 5: **for** $g = 1, \dots, G$ **do**
- 6: Compute the group g fission source:

$$\mathbf{q}_{f,g} = \frac{1}{K_{\text{eff}}^{(l)}} \sum_{g'=1}^G \mathbf{F}_{g,g'} \phi_{g'}^{(l)}$$

- 7: Compute the group g down-scatter source:

$$\mathbf{q}_{s,g}^{\downarrow} = \sum_{g'=1}^g \mathbf{S}_{g,g'} \phi_{g'}^{(l+1)}$$

- 8: Compute the group g up-scatter source:

$$\mathbf{q}_{s,g}^{\uparrow} = \sum_{g'=g+1}^G \mathbf{S}_{g,g'} \phi_{g'}^{(l)}$$

- 9: Solve:

$$\mathbf{L}_g \phi_g^{(l+1)} = \mathbf{q}_{f,g} + \mathbf{q}_{s,g}^{\downarrow} + \mathbf{q}_{s,g}^{\uparrow}$$

- 10: **end for**
- 11: Compute new fission integral:

$$I_f^{(l+1)} = \int_V \left(\sum_{g=1}^G \nu \Sigma_f^g \phi_g^{(l+1)} \right) dx$$

- 12: Update the multiplication factor:

$$K_{\text{eff}}^{(i+1)} = K_{\text{eff}}^{(i)} \frac{I_f^{(i+1)}}{I_f^{(i)}}$$

- 13: Check convergence:

$$\max_{g=1, \dots, G} \frac{\|\phi_g^{(i+1)} - \phi_g^{(i)}\|}{\|\phi_g^{(i)}\|} < \epsilon_\phi \quad \text{or} \quad |K_{\text{eff}}^{(l+1)} - K_{\text{eff}}^{(l)}| < \epsilon_k$$

- 14: **end for**
-

6.6.5 Parallelisation Strategy

The parallel programming model used in the present work is distributed-memory message passing, using the message passing interface (MPI), the details of which may be found in [278, 279]. In this programming model, each concurrent process has direct access only to its own memory address space. If process A requires data owned by process B for computation, process A must request it explicitly, process B must first receive and acknowledge the request before sending the data in response. This is what is meant by message-passing. MPI is a standardised interface for performing and co-ordinating message passing in an efficient and portable way.

There are two parts of the NECG-VEM algorithm that require the majority of the computation time, these are the assembly and the linear solve. Both are also good targets for parallelisation. To achieve a high parallel efficiency in both assembly and the parallel Krylov solves there are two operations that must be implemented:

1. First, the mesh must be partitioned across all MPI processes such that the computational load is balanced: this ensures processes are not sitting idle.
2. Second, the DoFs must be renumbered across all processes such that the MPI communication costs for a distributed-memory matrix-vector multiply (MVM) are low.

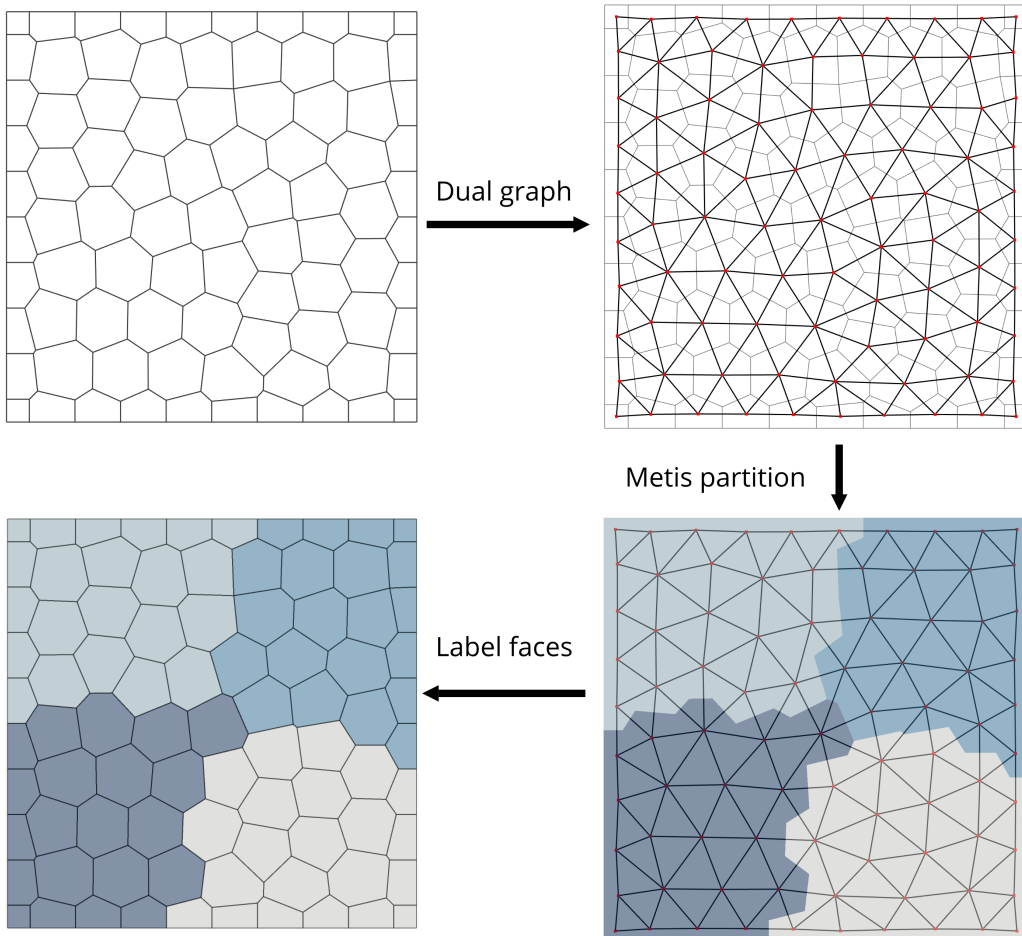


Figure 6.2: An illustration of the algorithm used to partition the mesh.

The first task, the partitioning of the mesh, consists of the following steps (see figure 6.2). First, the dual graph of the polygonal mesh is computed. A dual edge connects two dual vertices if their corresponding primal faces share a primal edge. Next, the open source library Metis [283] is used to partition the dual graph, there are options to assign weights to the vertices and edges of the graph in order to achieve better load balance or minimise a cut-cost function. For example, vertices could be weighted according to an a priori estimate of the cost of computing local bilinear forms. However, for the moment all the dual vertices and edges are weighted equally with 1.0. Once the dual graph has been partitioned by Metis, each dual vertex will have an MPI rank (process ID) mapped to it. These ID's are then simply mapped back to primal faces, after which the mesh partition is complete.

As a practical matter, a full copy of the mesh is stored on each MPI process with each process simply operating on a subset of the mesh which was assigned to it by Metis. This reduces the complexity of the code but is memory inefficient and ultimately does not scale as the mesh becomes large (millions of elements). However, the meshes used in the present work do not exceed 400,000 elements and are limited to 2D, therefore memory-scalability is not an issue here.

Before discussing the second task: renumbering the DoFs across all MPI processes such that the MPI communication costs for a MVM are low, we must first understand some of the details of the distributed MVM. In this thesis, distributed linear algebra was implemented using PETSc [277], which is an open-source library with implementations of MPI-parallel vectors, matrices, Krylov solvers and preconditioners. In the PETSc user manual it is stated that MPI-parallel vectors and matrices are partitioned contiguously by row, with rows split equally among processes. Let $\mathbf{A}$ be a $\mathbb{R}^{N \times N}$ global matrix and let $\mathbf{v}$, $\mathbf{w}$ be global $\mathbb{R}^N$ vectors. Let's assume that $\mathbf{A}$, $\mathbf{v}$, $\mathbf{w}$ are all distributed by row across P processes and consider a distributed memory MVM:

$$\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_P \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,P} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{P,1} & \mathbf{A}_{P,2} & \cdots & \mathbf{A}_{P,P} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_P \end{bmatrix}$$

and in particular the first operation of the MVM: the dot product between the first row of $\mathbf{A}$ and $\mathbf{v}$

$$\mathbf{w}_1 = \underbrace{\mathbf{A}_{1,1}\mathbf{v}_1}_A + \underbrace{\sum_{i=2}^P \mathbf{A}_{1,i}\mathbf{v}_i}_B.$$

The data required to compute part A is owned entirely by process 1, therefore no MPI communication is required. Meanwhile, each term in part B : $\mathbf{A}_{1,j}\mathbf{v}_j$, requires the portion of $\mathbf{v}$ owned by process j to be sent to process 1 before computation may be performed. If $\mathbf{A}$ is dense then there is an unavoidable all-to-all communication. If $\mathbf{A}$ is sparse, and many of the $\mathbf{A}_{1,j}$ are zero, then the communication overhead of the MVM is potentially much lower.

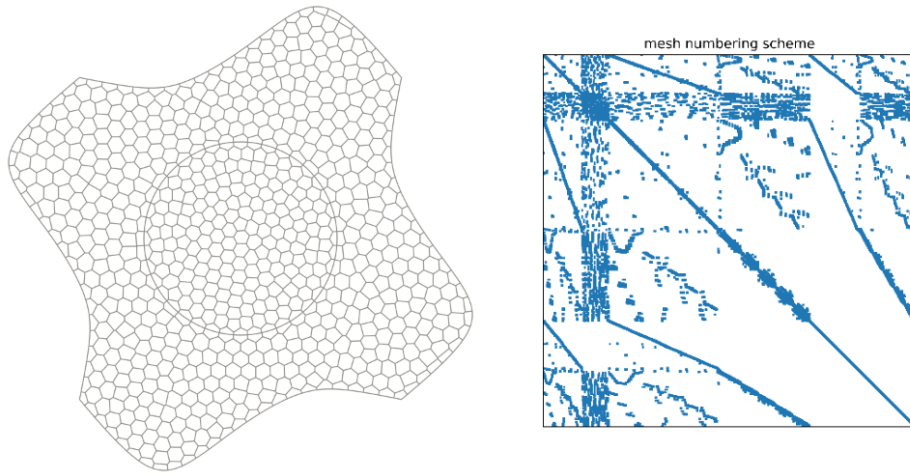


Figure 6.3: A polygonal mesh and the sparsity pattern that results from the mesh numbering scheme.

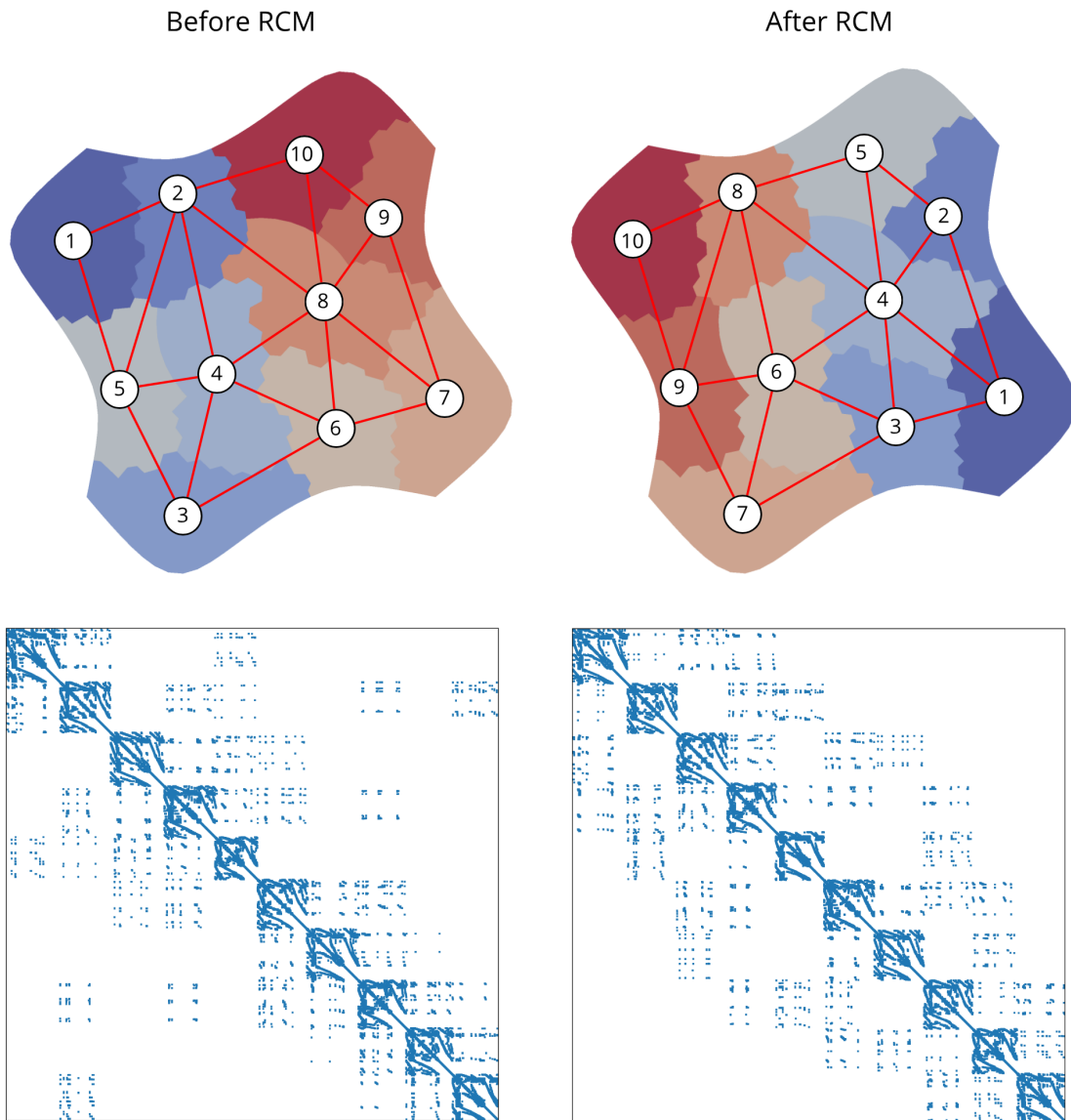


Figure 6.4: The partitions of the mesh of figure 6.3 into $P = 10$ parts and the corresponding sparsity patterns arising from the partition numbering system.

To demonstrate the DoF reordering algorithm consider the mesh presented in figure 6.3, along with the sparsity pattern produced when equation (6.3.5) is assembled with DoFs in the **mesh numbering scheme**, where DoFs are distributed to vertices first, edges second and faces last in ascending order. The NECG-VEM scheme is of order $p = 3$. For this example, the matrix presented in figure 6.3 (denoted $\mathbf{A}$), shall be partitioned across $P = 10$ processes. Once partitioned by row, the sparsity pattern is not optimal for MPI communication as every one of the off-diagonal sub-matrices has some non-zeros in it. Therefore, every process must communicate with every other process. To improve the MPI communication properties of the sparsity pattern, we must compute a DoF permutation vector $\mathbf{p}$ which maps each DoF in the mesh numbering system i to new index $\beta_i = \mathbf{p}[i]$, in what we shall call the **partition numbering system** (PNS). The partition numbering system is computed to maximise the number of matrix non-zeros that are clustered in the diagonal sub-matrices of $\mathbf{A}$. This approach maximises the amount of computation that may occur on each process independent of the others and minimises the number of other processes with which a process must communicate. Examples of the sparsity pattern for two different permutation vectors are presented in figure 6.4. Both were computed using the same algorithm, with the only difference being that the partition IDs corresponding to the right matrix sparsity pattern of figure 6.4 were reordered using a Reverse Cuthill-McKnee (RCM) algorithm [284] to minimise the global bandwidth of the reordered sparsity pattern. As was explained in section 6.6.1, the degrees of freedom that reside on the edges between two partitions are coupled only to the faces that are share those edges. Therefore, the partition-ID to partition-ID graphs of figure 6.4 are also the communication graphs. In other words, two processes only need to share information if their corresponding mesh partitions are physically adjacent to one another. With these ideas in mind, the algorithm to compute $\mathbf{p}$ proceeds as follows

1. On each process, loop over the elements assigned to it by Metis and find the union of all of the DoFs that exist on the process. Sort that set of DoFs so that it is in ascending order. These DoFs are in the mesh ordering system.
2. Let D_i and D_j be the sets of degrees of freedom, computed in step 1, that belong to processes i and j respectively. If processes i and j correspond to neighbouring partitions then find their intersection $D_{i,j} = D_i \cap D_j$. The set $D_{i,j}$ will be equal to the union of DoFs that lie on the edges and vertices that are shared between the partitions.
3. To ensure that the new DoF numbering does not double-count DoFs, we must assign ownership of the DoFs residing on partition interfaces to just one of the partitions. This is done by assigning the DoFs to the partition with **lower** ID number. This choice arbitrary but it must be consistently enforced. Therefore, if $i > j$ then D_j is unchanged and $D_i = D_i - D_{i,j}$. Conversely, if $j < i$ then D_i is unchanged and $D_j = D_j - D_{i,j}$.

After this stage every process shall have a set of DoFs in the mesh numbering system, $\{D_i\}_{i=1}^P$, stored in ascending order which are mutually disjoint and the union of which is

equal to full set of global DoFs:

$$D_i \cap D_j = \emptyset \quad \text{and} \quad D_1 \cup D_2 \dots \cup D_P = \{1, 2, \dots, N_{\text{gdof}}\}$$

4. Let $N_{\text{dof}}^i = |D_i|$ denote the number of DoFs in partition i . Then a DoF offset function, denoted $\text{offset}(i)$ may be defined as:

$$\text{offset}(j) = \sum_{i=1}^{j-1} N_{\text{dof}}^i$$

Where the argument to $\text{offset}(j)$ is a partition ID. Using the offset function, the permutation vector from the mesh numbering system to the partition numbering system may be computed on each process as follows:

Algorithm 21 Algorithm to find the partition numbering permutation vector.

- 1: **Input:** The process id pid and the corresponding set of on-process DoFs D_{pid} .
 - 2: **Output:** Forward and inverse permutation vectors $\mathbf{p}$ and $\hat{\mathbf{p}}$.
 - 3: Initialise the permutation vector: $\mathbf{p} \leftarrow [0, 0, \dots, 0]$
 - 4: Initialise the inverse permutation vector: $\hat{\mathbf{p}} \leftarrow [0, 0, \dots, 0]$
 - 5: Initialise new DoF number to the partition offset: $\beta = \text{offset}(\text{pid})$
 - 6: **for** each DoF $i \in D_{\text{pid}}$ **do** $\triangleright$ The loop is in ascending order of DoFs
 - 7: $\mathbf{p}[i] \leftarrow \beta$
 - 8: $\hat{\mathbf{p}}[\beta] \leftarrow i$
 - 9: $\beta \leftarrow \beta + 1$
 - 10: **end for**
-

During assembly, when the local bilinear matrices are computed, the DoFs on the face are calculated first in the mesh numbering system before being mapped forward into the partition numbering system. The local matrices are then inserted into the global matrix in the usual way. Performing assembly in this way avoids a reordering of rows and columns of the global matrix.

6.7 Numerical Results

6.7.1 Method of Manufactured Solutions (MMS)

Consider the following geometry comprised of two subdomains V_1 and V_2 shown below:

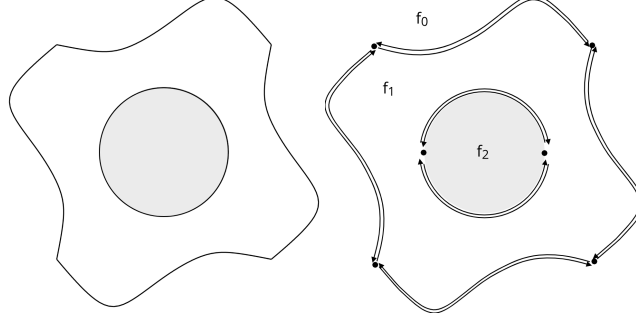


Figure 6.5: The domain, $V = V_1 \cup V_2$ used for this MMS test case.

The left image of figure 6.5 shows the control points of each curve and the right image shows the geometric domain created by building NURBS surfaces from the curves. The following two-group problem is defined on $V = V_1 \cup V_2$.

$$\begin{cases} -\nabla \cdot (D^1 \nabla \phi_1) + \Sigma_r^1 \phi_1 = f_1 & \forall \mathbf{x} \in V \\ -\nabla \cdot (D^2 \nabla \phi_2) + \Sigma_r^2 \phi_2 = f_2 + \Sigma_s^{1 \rightarrow 2} \phi_1 & \forall \mathbf{x} \in V \\ -D^1 \nabla \phi_1 \cdot \mathbf{n} = r_1 & \forall \mathbf{x} \in \partial V \\ -D^2 \nabla \phi_2 \cdot \mathbf{n} = r_2 & \forall \mathbf{x} \in \partial V \end{cases} \quad (6.7.1)$$

The values of the material coefficients may be found in table 6.1.

	D^1	D^2	Σ_r^1	Σ_r^2	$\Sigma_s^{1 \rightarrow 2}$
V_1	2.0	3.0	0.5	0.9	0.2
V_2	4.0	6.0	1.2	1.4	0.4

Table 6.1: Table of macroscopic nuclear cross section material data.

Letting $v = (v_1, v_2) \in [H^1(V)]^2$, $\phi = (\phi_1, \phi_2) \in [H^1(V)]^2$ and $f = (f_1, f_2)$, equation (6.7.1) has the following variational form:

$$\begin{cases} \text{Find } \phi \in [H^1(L^2(V))]^2 \text{ such that:} \\ A(v, \phi) = l(v) \quad \forall v \in [H^1(L^2(V))]^2 \end{cases} \quad (6.7.2)$$

where

$$\begin{aligned} A(v, \phi) = & (\nabla v_1, D^1 \nabla \phi_1)_{0,V} + (v_1, \Sigma_r^1 \phi_1)_{0,V} \\ & + \\ & (\nabla v_2, D^2 \nabla \phi_2)_{0,V} + (v_2, \Sigma_r^2 \phi_2)_{0,V} - (v_2, \Sigma_s^{1 \rightarrow 2} \phi_1)_{0,V} \end{aligned}$$

and

$$l(v) = (v_1, f_1)_{L^2(V)} + (v_1, r_1)_{\partial V} + (v_2, f_2)_{L^2(V)} + (v_2, r_2)_{\partial V}.$$

The source functions (f_1, f_2) , derived in section E.7, and boundary functions (r_1, r_2) are selected such that the exact solutions are given by the same function below:

$$\hat{\phi}_1 = \hat{\phi}_2 = (x + \cos(y))(y - \sin(x)). \quad (6.7.3)$$

Equation (6.7.3) is plotted in figure 6.6.

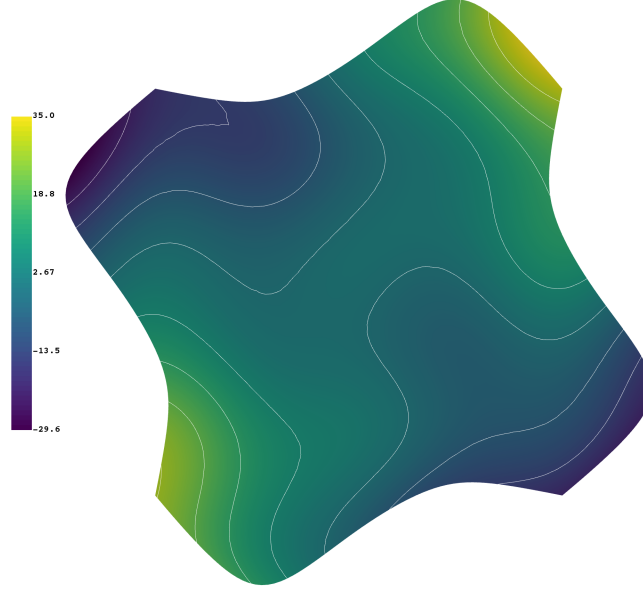


Figure 6.6: Plot of the MMS solution with isolines in white.

The variational form in equation (6.7.2) was discretised using the NECG-VEM and solved with a series of successively more refined polygonal meshes for orders $p = 1, 2, 3, 4, 5$. Some examples are given in figure 6.7.

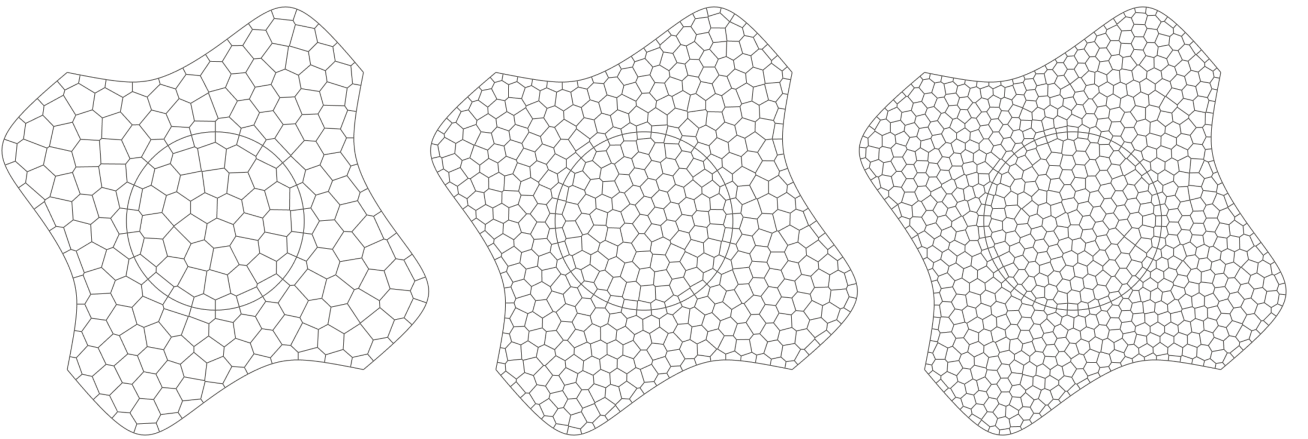


Figure 6.7: Three of the polygonal meshes used to solve the MMS problem.

Let h be the average element diameter for a given mesh and let ϕ_1^h and ϕ_2^h be the discrete VEM solutions to equation (6.7.2) then for every diameter-order pair, denoted (h, p) , both the $L^2(V)$ and $H^1(V)$ errors were computed as follows:

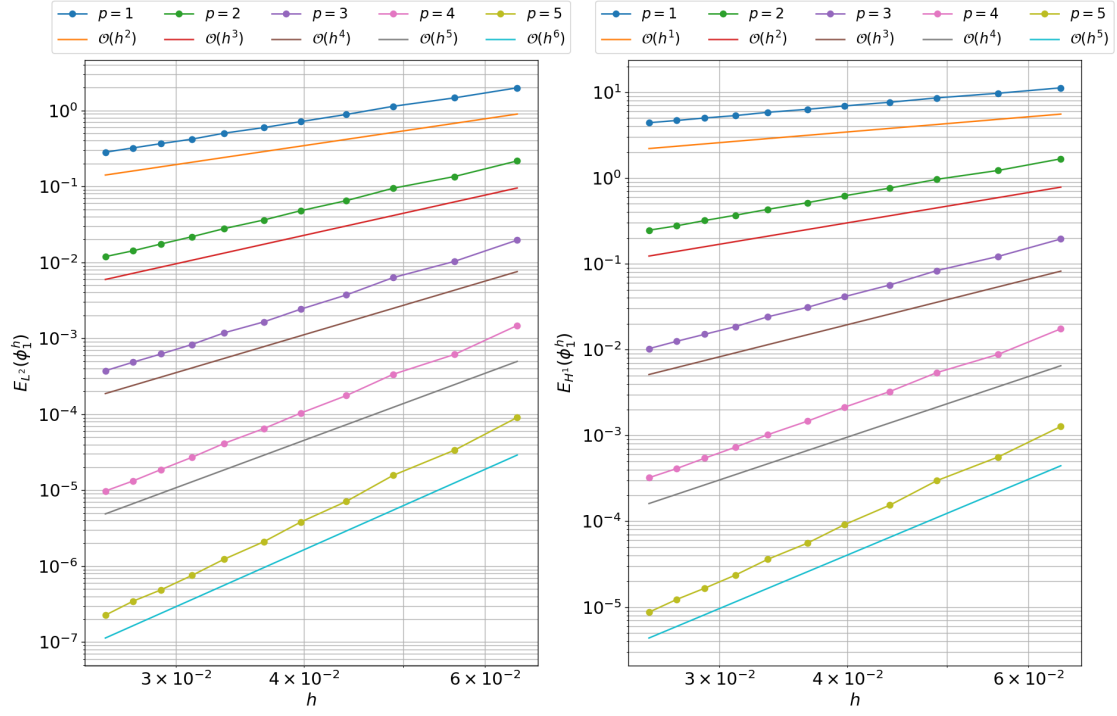
$$E_{L^2}(\phi_1^h) = \left(\sum_{F \in \mathcal{T}_h} \int_F (\hat{\phi}_1 - \Pi_p^0 \phi_1^h) (\hat{\phi}_1 - \Pi_p^0 \phi_1^h) dx \right)^{\frac{1}{2}},$$

and

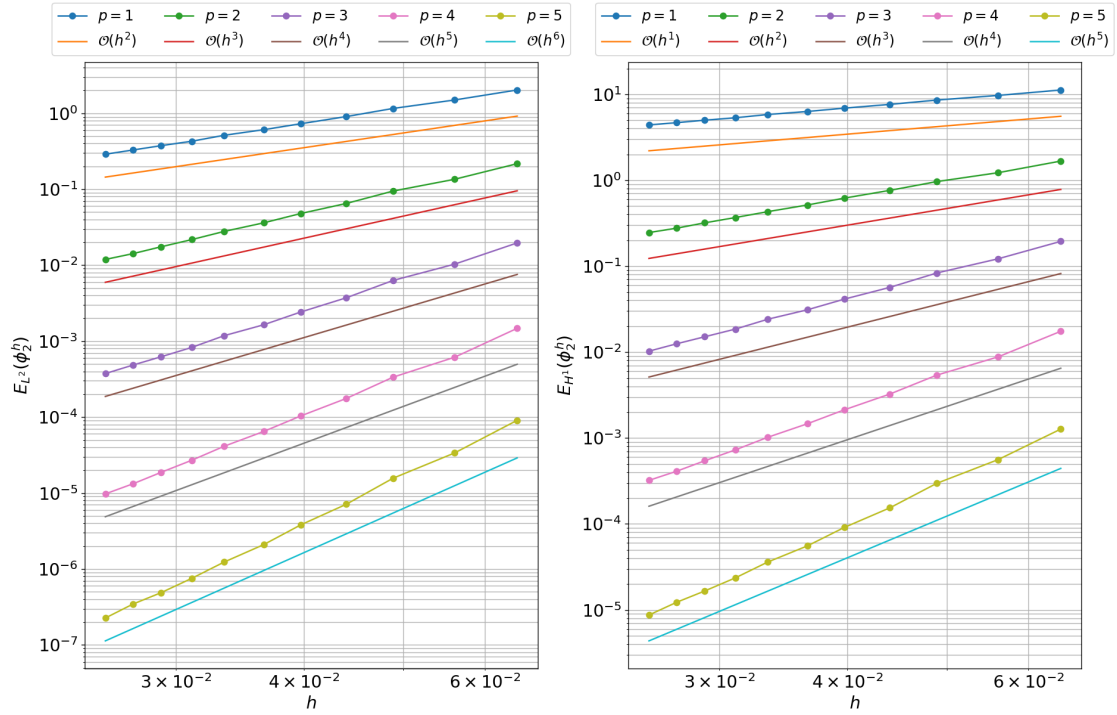
$$E_{H^1}(\phi_1^h) = \left(\sum_{F \in \mathcal{T}_h} \int_F (\nabla \hat{\phi}_1 - \Pi_{p-1}^0 \nabla \phi_1^h) \cdot (\nabla \hat{\phi}_1 - \Pi_{p-1}^0 \nabla \phi_1^h) dx \right)^{\frac{1}{2}},$$

with the errors for ϕ_2^h having the same definition. Given that the exact solutions $\hat{\phi}_1$ and $\hat{\phi}_2$ are infinitely differentiable everywhere in V then the both ϕ_1^h and ϕ_2^h should converge with $\mathcal{O}(h^p)$ and $\mathcal{O}(h^{p+1})$ accuracy in the H^1 and L^2 norms respectively.

The first result presented in this section, given in figures 6.8a and 6.8b shall be for the case where exact geometry is enabled. The results show that the expected order of accuracy is attained. The second result presented in this section, given in figures 6.9a and 6.9b, plots the errors when exact geometry has been disabled, and so the curvilinear edges in the V are approximated using straight line segments. As was explained in [1], this means the curved VEM becomes a standard straight sided VEM such as that presented in [231]. In this case the errors at higher orders (orders $p > 2$ for the L^2 error and orders > 3 for the H^1) are dominated by the geometric error. The rates of convergence for these higher orders are therefore limited to the rate of convergence of the geometric error, which is apparently $\mathcal{O}(h^2)$ for the L^2 error and $\mathcal{O}(h^3)$ for the H^1 error.

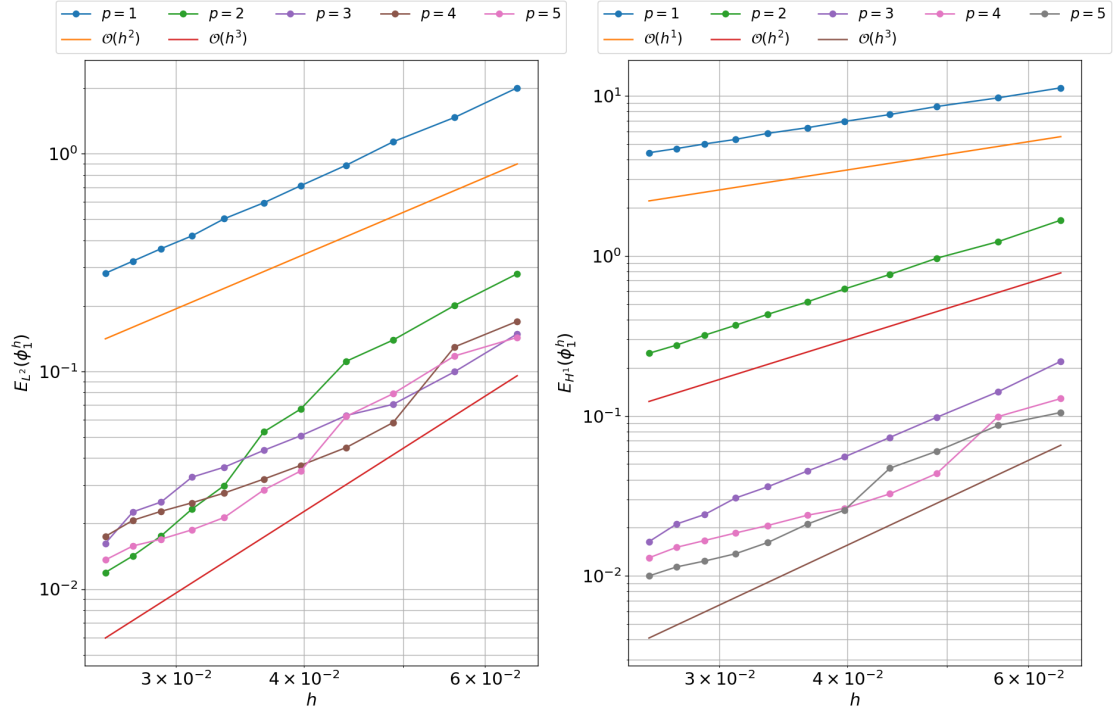


(a) Plots of $E_{L^2}(\phi_1^h)$ (left) and $E_{H^1}(\phi_1^h)$

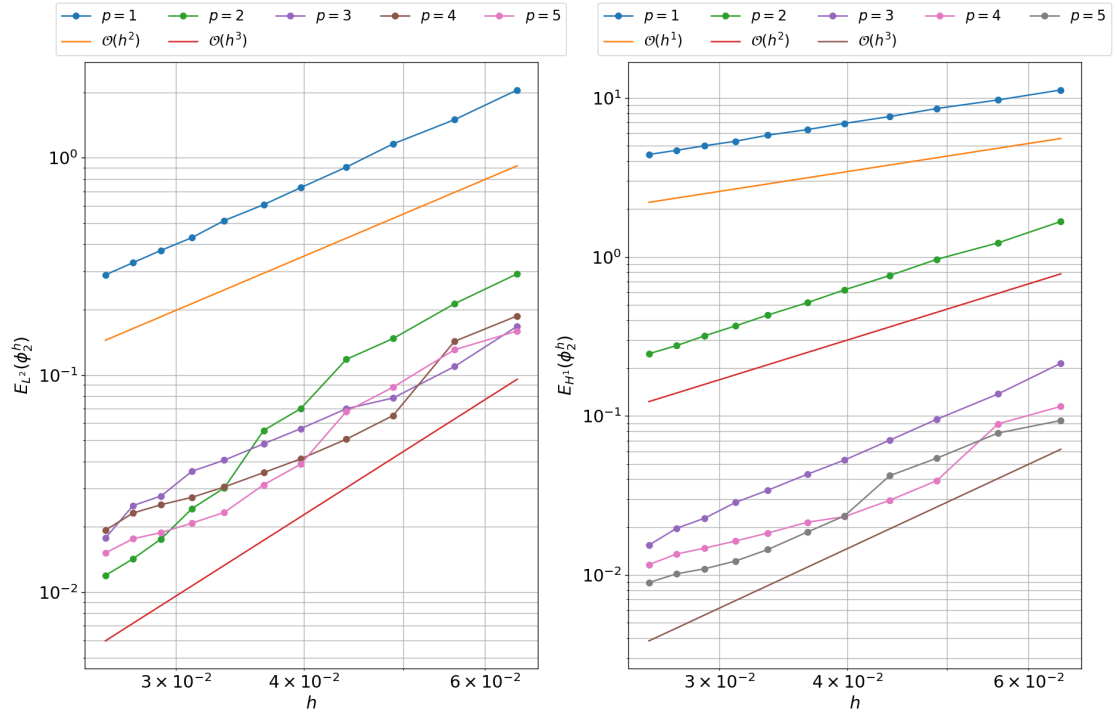


(b) Plots of $E_{L^2}(\phi_2^h)$ (left) and $E_{H^1}(\phi_2^h)$

Figure 6.8: The errors E_{L^2} and E_{H^1} for the case where both exact geometry and polynomial basis orthogonalisation are enabled.



(a) Plots of $E_{L^2}(\phi_1^h)$ (left) and $E_{H^1}(\phi_1^h)$



(b) Plots of $E_{L^2}(\phi_2^h)$ (left) and $E_{H^1}(\phi_2^h)$

Figure 6.9: The errors E_{L^2} and E_{H^1} for the case where exact geometry has been disabled.

6.7.2 Burnable Supercell Benchmark Verification Test Case

This problem, originally presented in [285] and modified in [286], is a single group, fixed-source problem consisting of 8 identical fuel pins arranged around a central burnable absorber pin. A schematic of the geometry is presented in figure 6.10.

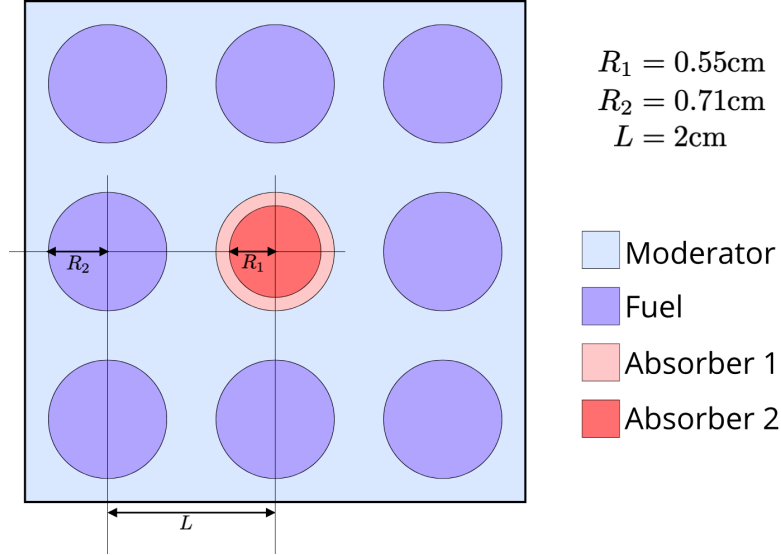


Figure 6.10: Geometry and materials of the burnable supercell benchmark verification test case.

Reflected boundary conditions are prescribed on all four sides of the supercell.

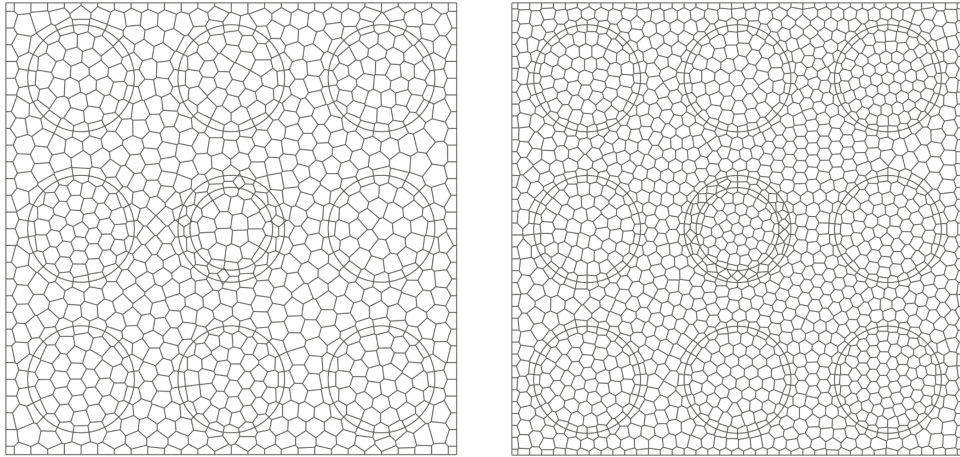


Figure 6.11: Two meshes used to solve the burnable supercell benchmark verification test case.

The computed values of the neutron macroscopic neutron cross sections and the value of the extraneous, prescribed (fixed) neutron source $q(\mathbf{x})$ are given in table 6.2.

Table 6.2: The nuclear data for the burnable supercell benchmark verification test case.

Material	D	Σ_t	Σ_s	$q(\mathbf{x})$
Moderator	0.4411	0.7557	0.7504	1
Fuel	0.6479	0.5145	0.3432	0
Absorber 1 (annulus)	1.5411e-02	21.630	0.3414	0
Absorber 2 (circle)	2.3359e-02	14.270	0.3400	0

The governing equations of this problem are:

$$\begin{cases} -\nabla \cdot (D(\mathbf{x}) \nabla \phi) + \Sigma_a \phi = q(\mathbf{x}) & \forall \mathbf{x} \in V \subset \mathbb{R}^2 \\ -D(\mathbf{x}) \nabla \phi \cdot \mathbf{n} = 0 & \forall \mathbf{x} \in \partial V \end{cases}$$

where $D = 1/3\Sigma_t$ and $\Sigma_a = \Sigma_t - \Sigma_s$. We shall define four quantities of interest (QoI): the integral of the flux over the moderator I_{mod} , the integral of flux over all eight fuel pins I_{fuel} , the integral of flux over the absorber annulus I_{ab1} and the integral of the flux over the absorber pin I_{ab2} . The reference values of these quantities are shown in table 6.3.

Table 6.3: The reference values of the QoIs. They were computed using the NECG-VEM scheme defined on a polygonal mesh with 29401 elements and VE order $p = 4$, resulting in a scheme with $N_{\text{g dof}} = 486238$.

QoI	I_{mod}	I_{fuel}	I_{ab1}	I_{ab2}
Reference Value	7.21396146e+01	1.42270560e-03	4.09144614e-01	1.24426030e02

Additionally, the scalar neutron flux contour of the reference solution is presented in figure 6.12

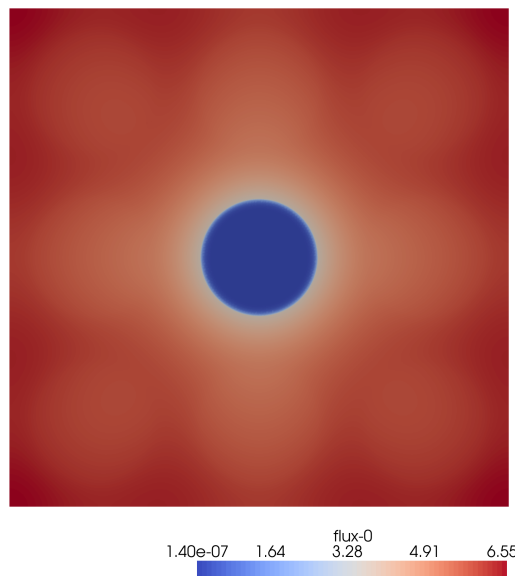


Figure 6.12: The scalar neutron flux contour of the reference solution

Qualitatively, the solution behaves as one would expect. It is flat and smooth far away from the moderator-absorber interface and at the interface exhibits a large gradient in the scalar neutron flux going radially out from the centre. This is caused by a large change in the macroscopic absorption neutron cross section. Finally, the solution is symmetric with 45 degree symmetry planes.

Figures 6.13-6.16 present the convergence curves for each of the QoIs and also compare the convergence curves of the exact-geometry VEM and the conventional, straight-sided VEM. For each QoI, the y -axis is shared between the exact-geometry VEM results (shown on the left) and the conventional VEM results shown on the right. This has been done to facilitate easy comparison between the two.

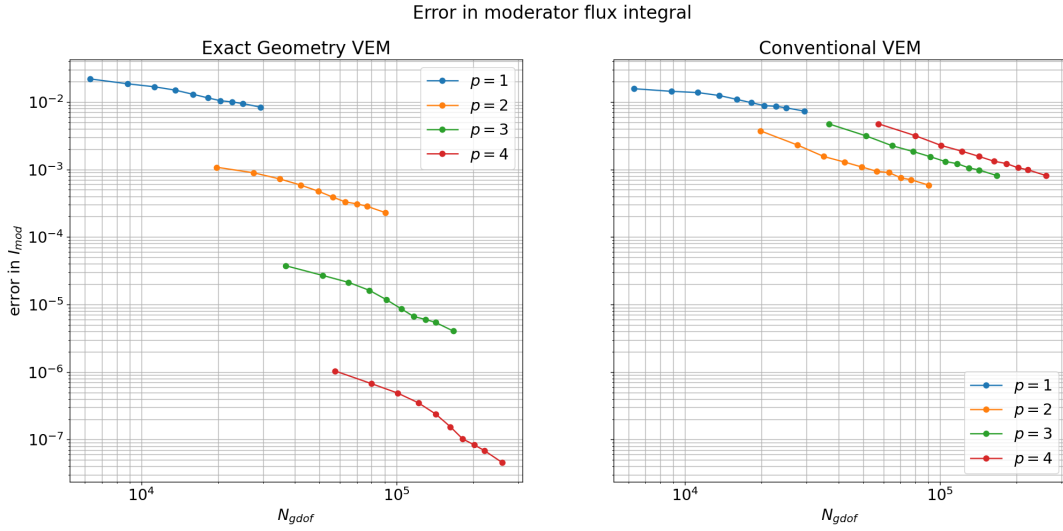


Figure 6.13: Convergence plots of I_{mod} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).

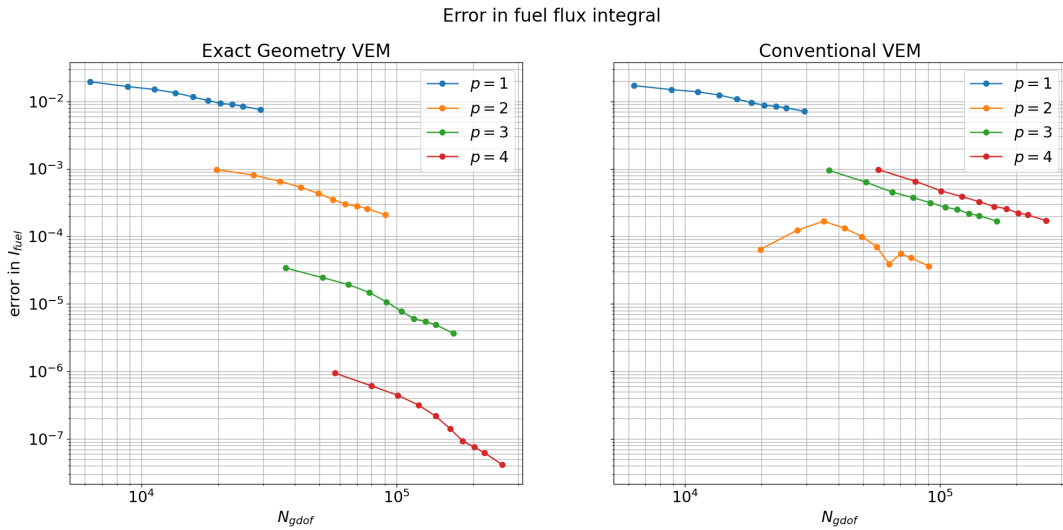


Figure 6.14: Convergence plots of I_{fuel} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).

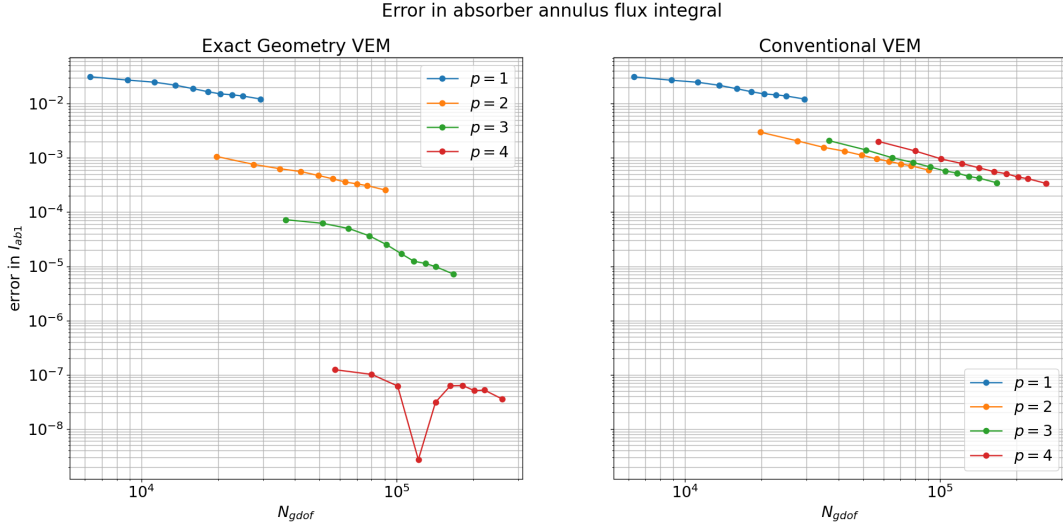


Figure 6.15: Convergence plots of I_{ab1} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).

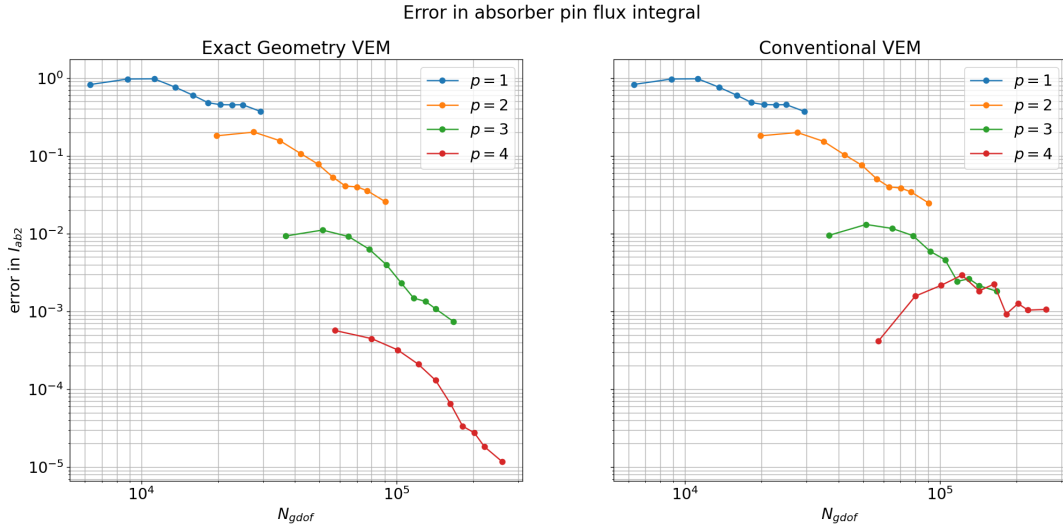


Figure 6.16: Convergence plots of I_{ab2} for the exact-geometry VEM (left) and the conventional, straight-sided VEM (right).

Figures 6.13-6.16 show that in the absence of exact-geometry, the VEM schemes are never better than the $p = 2$ scheme. This indicates that, for higher VEM orders and for the conventional VEM, the geometric errors are the dominant source of error. In the convergence curves of the exact-geometry VEM it is clear that the $p = 3, 4$ curves are shifted downward considerably, indicating that the major sources of error in the QoI are from the approximation error in the VEM discrete solution, as opposed to geometric error. Therefore, increasing the VEM order has a major effect on the error.

One artifact of figure 6.15 is the sudden dip of the exact-geometry, $p = 4$ curve. This was caused by the error changing sign, and therefore appearing to be anomalously low for one data point.

6.7.3 The 2D OECD-NEA C5G7 Benchmark Verification Test Case

The 2D OECD-NEA C5G7 Benchmark verification test case, presented in [287], is a nuclear reactor physics problem defined on a PWR geometry. The benchmark was developed in order to verify the ability of nuclear reactor physics codes to solve problems without spatial homogenisation. The geometry of the problem is presented in figure 6.17.

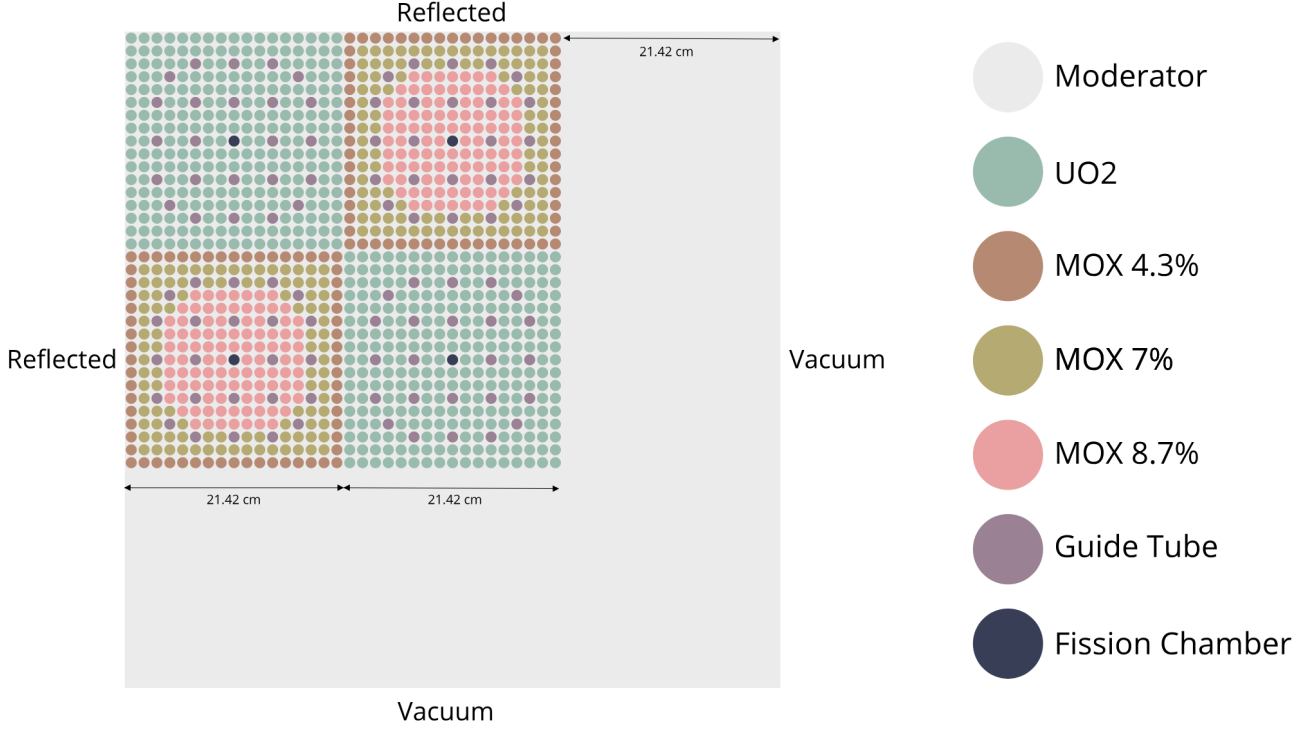


Figure 6.17: The geometry of the 2D C5G7 benchmark verification test case.

Symmetry along the x and y axes have reduced the problem to a quarter-core geometry consisting of two UOX assemblies and two MOX assemblies. Each assembly is a 17×17 array of nuclear fuel pins surrounded by a water reflector and each nuclear fuel pin has a radius of 0.54cm. The values of the macroscopic nuclear cross section material data, which may be found in the OECD/NEA benchmark specification [287], are also given in the appendix, section E.8.

The quantities of interest (QoIs) are the K_{eff} , the maximum pin power P_{min} , the minimum pin power P_{min} , the power of either MOX assembly P_{MOX} , the power of the inner UO_2 assembly $P_{\text{UO}_{2i}}$ and the power of the outer UO_2 assembly $P_{\text{UO}_{2o}}$. The full mathematical definitions of these QoIs are in the appendix, section E.8. A reference solution was calculated using the VEM with a 129,802-element polygonal mesh and an order of $p = 4$. This reference solution had 2,028,163 DoFs per group. The values of the QoIs for this reference solution are given in table 6.4.

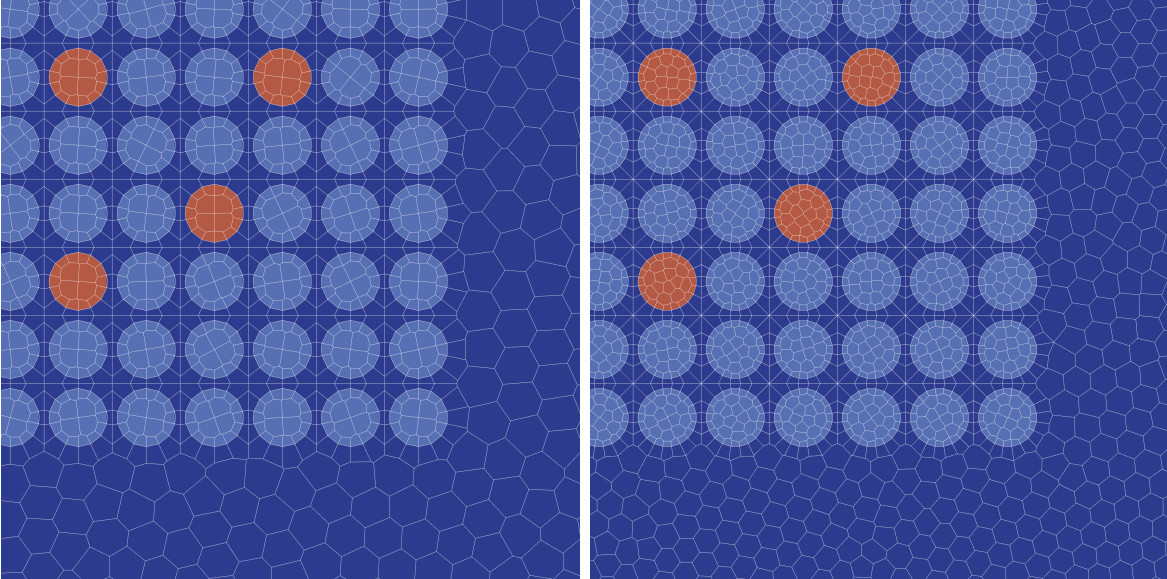
Table 6.4: Reference results for the C5G7 (2,028,163 DoFs).

K_{eff}	P_{max}	P_{min}	P_{MOX}	$P_{\text{UO}_{2o}}$	$P_{\text{UO}_{2o}}$
1.18324062	2.52089853	0.24063658	210.82158394	495.45445506	138.90237681

These values are in good agreement with the values presented in [166], where a high-fidelity and exact-geometry IGA solution was calculated. The scalar neutron flux contours of the reference solution are given in figure 6.19 for the $g = 1, 3, 5, 7$ energy groups. To investigate the numerical behaviour of the exact-geometry VEM, the C5G7 was solved using a series of 10 polygonal meshes ranging from 22,968 elements to 82,359 elements and for orders $k = 1, 2, 3, 4$. The QoIs were calculated for each mesh-order pair and the relative errors with respect to the reference values of table 6.4 were computed. These errors are plotted in figure 6.20. The convergence tolerance for this problem was set to 1×10^{-12} with respect to the convergence criterion of line 13 in algorithm 20. At this value, the C5G7 would require around 137 power iterations to converge. The scalar neutron fluxes were then normalised such that the fission source is equal to the number of nuclear fuel pins: 1056 [287].

$$\int_V \sum_{g=1}^G \nu \Sigma_f^g \phi_g(\mathbf{x}) dx = 1056.$$

Figures 6.18a and 6.18b show a portion of the 44,802 element mesh and the 82,359 element mesh from which the uniform refinement strategy may be seen. No adaptive refinement was used, additional elements were simply added to each region in proportion the surface area of each region.

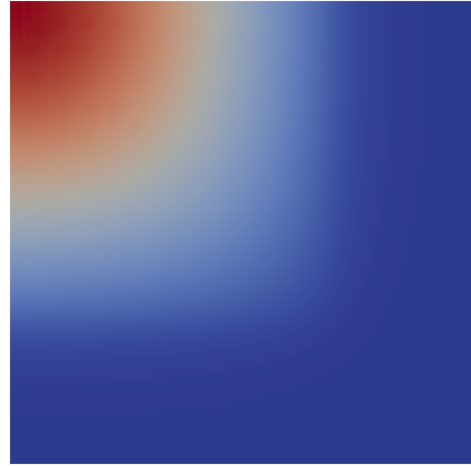
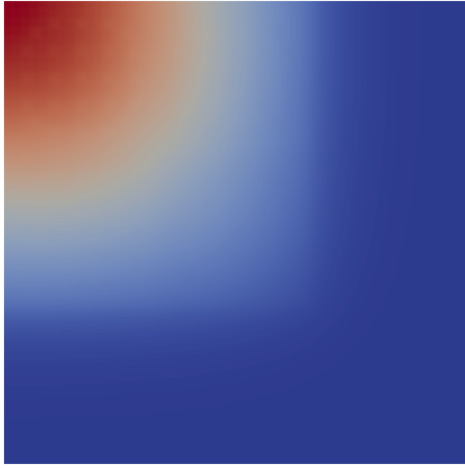


(a) Polygonal mesh with 44,802 elements. (b) Polygonal mesh with 82,359 elements.

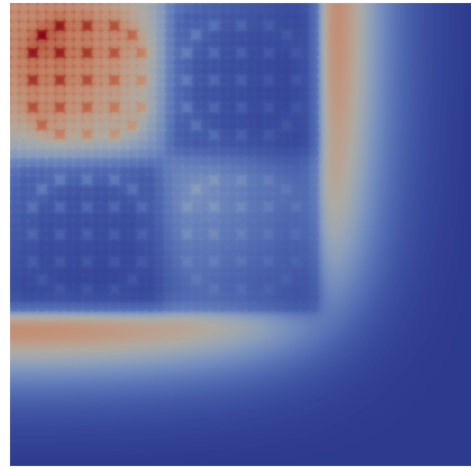
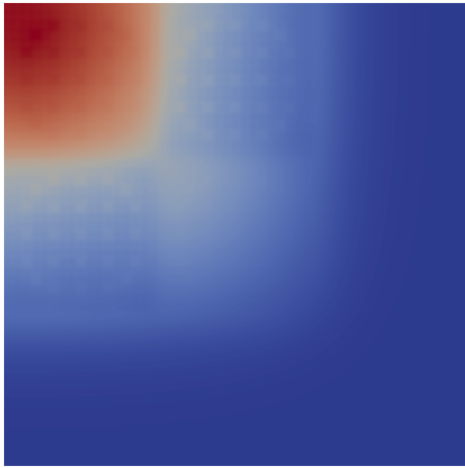
Looking at the convergence curves of figure 6.20 it is clear that for $p = 2, 3, 4$ there is a considerable amount of numerical noise. Particularly for $p = 2, 3$, the errors jump considerably. The reason for this is that the errors change sign, making the absolute value of the error appear to jump. However, it is still clear that the higher order curves are shifted downward relative to lower order curves and the errors themselves become very small, showing good agreement with the reference solution.

An additional complication is that the QoIs are a mix of global quantities, such as the K_{eff} , and

more localised quantities such as the min/max pin powers. Therefore, their error curves will exhibit different convergence properties depending upon how well converged the flux field is in the vicinity of the QoI. This is as problem that will be overcome in the future with adaptive mesh refinement (AMR) techniques, which for each region of the nuclear reactor will converge the flux to roughly the same tolerance.



(a) Scalar neutron flux contour for group 1. (b) Scalar neutron flux contour for group 3



(c) Scalar neutron flux contour for group 5 (d) Scalar neutron flux contour for group 7

Figure 6.19: Scalar neutron flux contours for energy groups 1, 3, 5 and 7.

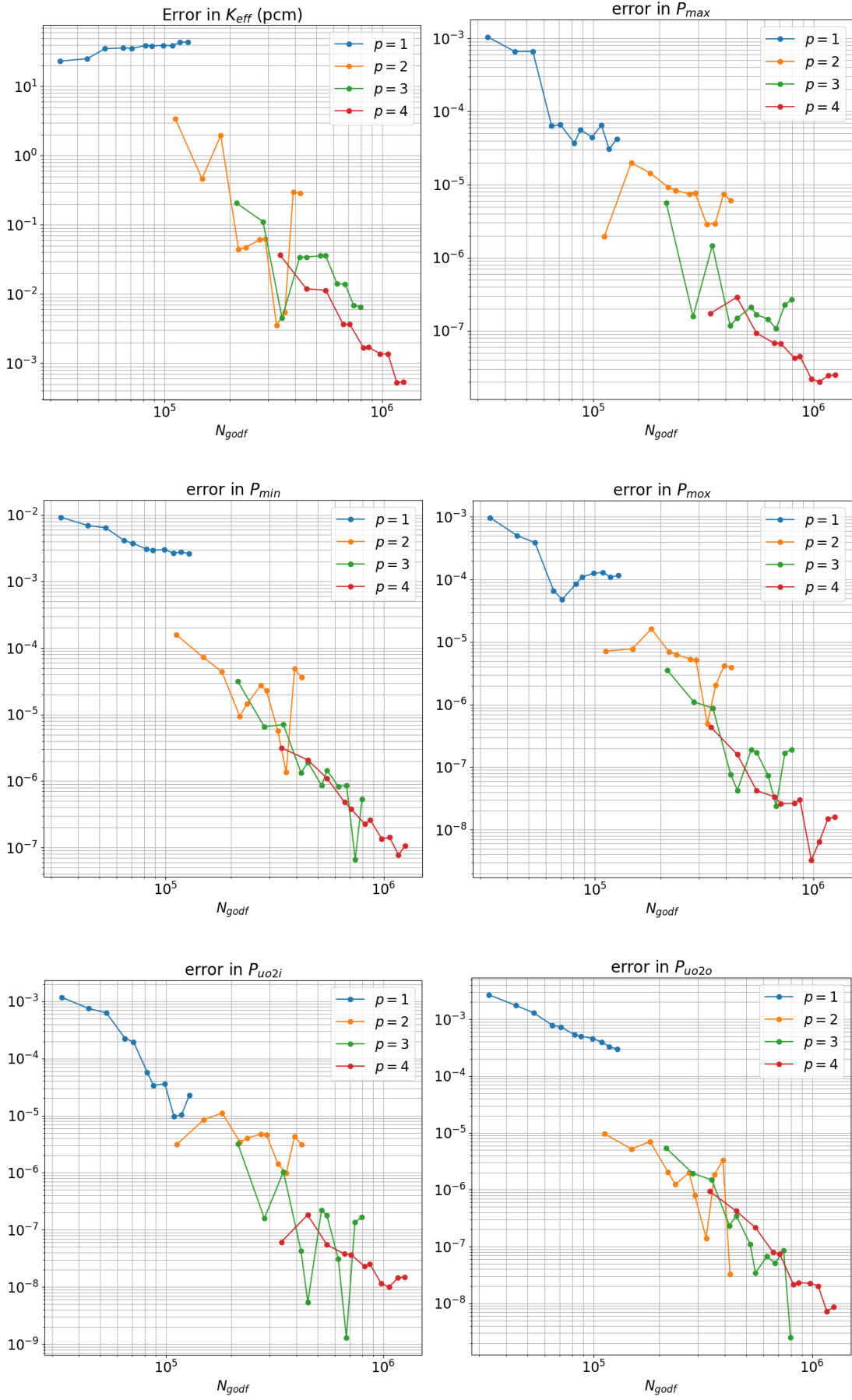


Figure 6.20: Errors in QoIs for the C5G7.

6.8 Discussion and Future Work

In this chapter we have presented the application of the NECG-VEM to the steady-state, multigroup, neutron diffusion equation, both for the fixed source and k -eigenvalue problems. We have presented the variational forms of both of these problems and proven that, under well defined and verifiable conditions on neutron cross sections, they are both well posed. We then presented the method by which the infinite-dimensional function space $\mathcal{W} := [H^1(V)]^G$ is restricted to a finite-dimensional space $\mathcal{W}_p := [\mathcal{V}_p(V)]^G \subset \mathcal{W}$ where the discrete, NECG-VEM, solution is sought. We then proved, using the properties of discrete VE bilinear forms first introduced in 6.3.3, that the discrete, VE variational problems are also well posed. We then presented an in-depth discussion of how the global systems of equations are assembled and solved iteratively and we have presented a distributed-memory parallelisation strategy for doing so. Finally, we presented three numerical test cases which demonstrate application of the NECG-VEM to multigroup neutron diffusion problems. First, we presented a method of manufactured solutions test case, to establish the optimal convergence rate of the method and to demonstrate the marked improvement of higher order NECG-VEM schemes over straight-sided VEM for problems with curvilinear geometry. Second, a burnable supercell fixed-source problem was presented to show the benefits of exact-geometry when calculating QoIs such flux integrals when in a reactor physics context. Finally, a C5G7 k -eigenvalue problem was presented to demonstrate the ability NECG-VEM to handle larger reactor physics problems with complex geometry.

A strength of the NECG-VEM presented in this chapter is that it has exact-geometry from the lowest level of mesh refinement. By implementing the kinds of mesh refinement strategies alluded to in section 5.6, it would be possible to preserve the exact-geometry of the reactor while performing h -refinement without recourse to an external CAD file. In this sense the NECG-VEM is similar in spirit to IGA.

As we have already mentioned in sections 4.6 and 5.6, the major limitation of the CG-VEM we have presented in this chapter is the lack of adaptive refinement. Thermal neutrons have a shorter mean-free path, consequently the thermal scalar flux profiles exhibit larger gradients and features at shorter length scales when compared to the fast scalar flux profile. The overall effect is that fast scalar fluxes typically have smoother flux profiles over larger distances. This means the optimal mesh refinements for fast and thermal fluxes are different. In this chapter, we used the same computational mesh for all energy groups. Consequently, the mesh will be insufficiently p -refined for the fast fluxes and will be insufficiently h -refined for the thermal fluxes. In [145, 146], both h and hp refinement were presented in the context of a continuous Galerkin finite element discretisation of the NDE. In those papers, each energy group had its own mesh with varying levels of hp -refinement and transfer operators were defined to project the solution of one energy group onto another to compute fission and scatter sources. In [167, 169, 170], similar ideas of group-dependent meshes and transfer operators were also presented for the case of Isogeometric Analysis (IGA) applied the multigroup neutron transport

equation. It is likely that any future research on group-dependent adaptivity for the NECG-VEM would follow a similar path to the papers cited above.

Chapter 7

Virtual Element Discretisation of the Multigroup Neutron Transport Equation

7.1 Introduction

The multigroup, S_N neutron transport equation is arguably the most important mathematical model used in computational nuclear reactor physics. It is a much more faithful representation of physical reality than the neutron diffusion equation, which is a useful mathematical simplification applicable only in highly scattering problems. Early attempts to solve the NTE in the 1950s and 1960s were always limited in the size of problems they could realistically solve on the available computing resources at the time. Therefore, to yield problems of tractable size, many geometrical simplifications were made, such as spatial homogenisation. Consequently, the spatial discretisations used at the time, such as the finite difference method, were not required to treat complicated geometry. However, it is now computationally feasible to treat reactor geometries without any simplifying modifications. Therefore, numerical methods which can efficiently and accurately represent the exact geometry of the reactor are increasingly necessary. In chapter 6 we introduced the NECG-VEM to serve this purpose for the NDE. In this chapter we shall introduce the NURBS-Enhanced discontinuous-Galerkin VEM (NEDG-VEM) to serve this purpose for the multigroup, S_N -NTE.

This chapter is structured as follows:

- In section 7.2 we formally introduce the multigroup, S_N -NTE with isotropic scatter. These are the equations this chapter is concerned with solving.
- In section 7.3 we present the application of the NEDG-VEM to the advection-reaction equation, the purpose of this section is to present the key mathematical properties of the NEDG-VEM: well-posedness with respect to a mesh-dependent norm and the a priori

error estimate with respect to a mesh-dependent norm.

- In section 7.4 we present the transport sweep algorithm for curved, polygonal elements including a heuristic strategy for breaking cycles in the dependency graph.
- In section 7.5 we derive the variational form of the multigroup, S_N -NTE.
- In section 7.6 we describe how global degrees of freedom are distributed on the mesh in the NEDG-VEM. We derive the global system of equations which result from the NEDG-VEM discretisation. Finally, we describe the task-based framework for performing transport sweeps and the shared-memory parallelisation strategy based on task-scheduling.
- In section 7.7, we present four test cases to substantiate the claims made in this chapter. In section 7.7.1 we present a method of manufactured solutions (MMS) test case to verify that the NEDG-VEM has convergence rate of $\mathcal{O}(h^{p+1/2})$ in the DG-norm for a NEDG-VEM scheme of order p and a sufficiently regular manufactured solution. In section 7.7.2 we present a method of manufactured solutions test case for the single-group neutron transport equation with an S_4 approximation in angle. The purpose of this test case was to verify that the S_N solver implemented in this thesis converges with the expected order of accuracy of $\mathcal{O}(h^{p+1/2})$ in the DG-norm. In section 7.7.3 we present the Reed Cell problem. This is a fixed source, single-group neutron transport problem with an S_8 approximation in angle. Due to the low regularity of the analytic solution to this problem, the NEDG-VEM converges with an order of $\mathcal{O}(h^{1.5})$ in the DG-norm, as expected. Finally, in section 7.7.4 we solve a 7-group pincell problem taken from the C5G7 benchmark test case. In this test case, we compare the NEDG-VEM to a classical, straight-sided DG-VEM and show that for this problem, which has a curved geometry, the NEDG-VEM convergence rates are improved relative to the straight-sided case for integral quantities of the scalar neutron flux.
- Finally, in section 7.8 we provide a discussion of the strengths and limitations of the NEDG-VEM presented in this chapter and shall make some recommendations for future research.

7.2 The Multigroup, S_N Equations

Let $V \subset \mathbb{R}^2$ be an open and connected domain with boundary ∂V which may be decomposed into vacuum and reflective boundaries: $\partial V = \partial V_V \cup \partial V_R$. Additionally, the boundary may be decomposed into inflow and outflow boundaries ∂V_- and ∂V_+ which depend on the discrete ordinate direction $\mathbf{\Omega}_k$. With the domain and boundary defined the steady-state, multi-group, discrete-ordinates, NTE is given by equations (7.2.1a), (7.2.1b), (7.2.1c) and (7.2.1d):

$$\mathbf{\Omega}_k \cdot \nabla \psi_{g,k}(\mathbf{x}) + \Sigma_t^g(\mathbf{x}) \psi_{g,k}(\mathbf{x}) = \frac{1}{4\pi} \sum_{g'=1}^G \Sigma_s^{g' \rightarrow g}(\mathbf{x}) \phi_{g'} + Q_{g,k}(\mathbf{x}) \quad \forall \mathbf{x} \in V \quad (7.2.1a)$$

$$\phi_g(\mathbf{x}) = \sum_{k=1}^K w_k \psi_{g,k}(\mathbf{x}) \quad \forall \mathbf{x} \in V \quad (7.2.1b)$$

$$\psi_{g,k}(\mathbf{x}) = 0 \quad \forall \mathbf{x} \in \partial V_V \cap \partial V_-, \quad (7.2.1c)$$

$$\psi_{g,k}(\mathbf{x}) = \psi_{g,k'}(\mathbf{x}) \quad \forall \mathbf{x} \in \partial V_R \cap \partial V_-, \quad (7.2.1d)$$

where $g = 1, \dots, G$ are the energy groups and $k = 1, \dots, K$ are the discrete ordinates. We shall refer to equations (7.2.1a) to (7.2.1d) collectively as the S_N -NTE. Equations (7.2.1c) and (7.2.1d) are the vacuum and reflective boundary conditions respectively. The incoming angular flux on reflective boundaries, $\psi_{g,k'}(\mathbf{x})$, corresponds to the angular flux of direction $\mathbf{\Omega}_{k'} = \mathbf{\Omega}_k - 2(\mathbf{\Omega}_k \cdot \mathbf{n})\mathbf{n}$ where $\mathbf{n}$ is the outward normal to ∂V at $\mathbf{x}$. The boundary must have a normal such that $\mathbf{\Omega}_{k'}$ is also a member of the discrete ordinate set. In this chapter we have limited the discussion to isotropic scatter sources, therefore the scatter kernel is expanded in the 0-order Legendre polynomial, which is constant. We shall therefore write $\Sigma_s^{g \rightarrow g'}$ in the place of the Legendre moment $\Sigma_{s,0}^{g \rightarrow g'}$ to simplify notation. The source $Q_{g,k}(\mathbf{x})$ takes one of two forms depending on the type of problem under consideration:

$$Q_{g,k}(\mathbf{x}) = \begin{cases} \frac{1}{4\pi} q_{g,k}(\mathbf{x}) & \text{Fixed source,} \\ \frac{1}{\gamma} \frac{\chi_g}{4\pi} \sum_{g'=1}^G \nu \Sigma_f^{g'} \phi_{g'} & k\text{-eigenvalue} \end{cases}.$$

When recasting equation (7.2.1a) in operator form, an order must be imposed on the equations. In this thesis we choose to order first by group and second by angle, meaning that all angles corresponding to a group are ordered contiguously first:

$$\begin{aligned} \psi_g &= [\psi_{g,1}, \psi_{g,2}, \dots, \psi_{g,K}]^T, & \psi &= [\psi_1, \psi_2, \dots, \psi_G]^T, \\ \phi &= [\phi_1, \phi_2, \dots, \phi_G]^T, & q_g &= [q_{g,1}, q_{g,2}, \dots, q_{g,K}] \end{aligned} \quad (7.2.2)$$

Therefore, ψ_g is the vector of group g angular fluxes ordered by discrete ordinate, ψ is the full vector of angular fluxes of dimension $G \cdot K$ ordered by energy group first and discrete ordinate second and ϕ is the full vector of scalar fluxes ordered by energy group. We shall define the

removal operator using the same ordering of group first and angle second:

$$L_{g,k}\psi_{g,k} = \mathbf{\Omega}_k \cdot \nabla \psi_{g,k} + \Sigma_t^g \psi_{g,k},$$

$$L_g \psi_g = \begin{bmatrix} L_{g,1} & 0 & \dots & 0 \\ 0 & L_{g,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & L_{g,K} \end{bmatrix} \begin{bmatrix} \psi_{g,1} \\ \psi_{g,2} \\ \vdots \\ \psi_{g,K} \end{bmatrix}, \quad L\psi = \begin{bmatrix} L_1 & 0 & \dots & 0 \\ 0 & L_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & L_G \end{bmatrix} \cdot \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_G \end{bmatrix}$$

Before presenting the operator form of the scatter term we must first recall the direction-to-moment operator and moment-to direction operators of equations (2.2.2) and (2.2.3):

$$D_{i,k} = w_k R_{l(i)}^{m(i)}(\mathbf{\Omega}_k), \quad M_{k,i} = \left(\frac{2l(i) + 1}{4\pi} \right) R_{l(i)}^{m(i)}(\mathbf{\Omega}_k).$$

In the case of isotropic scatter, where there is only one angular flux moment $\phi = \phi_0^0$, D and M have dimensions $1 \times K$ and $K \times 1$ respectively and are given by:

$$D_{1,k} = w_k, \quad M_{k,1} = 1/4\pi, \quad k = 1, \dots, K.$$

For each energy group g , the set of angular fluxes $\psi_{g,k}$ and the scalar flux ϕ_g are related as follows:

$$\phi_g = D\psi_g, \quad \psi_g = M\phi_g$$

and we therefore define the operators $\tilde{D}$ and $\tilde{M}$ which operate on the full vectors ψ and ϕ as follows:

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix} = \begin{bmatrix} D & 0 & \dots & 0 \\ 0 & D & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & D \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_G \end{bmatrix}, \quad \Rightarrow \phi = \tilde{D}\psi$$

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_G \end{bmatrix} = \begin{bmatrix} M & 0 & \dots & 0 \\ 0 & M & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_G \end{bmatrix}, \quad \Rightarrow \psi = \tilde{M}\phi$$

The matrices $\tilde{D}$ and $\tilde{M}$ have dimensions $G \times (G \cdot K)$ and $(G \cdot K) \times G$ respectively. Using the operators $\tilde{D}$ and $\tilde{M}$ we may write the scatter operator as:

$$S\psi = \tilde{M}\Sigma^S\tilde{D}\psi \quad \text{where} \quad \Sigma^S = \begin{bmatrix} \Sigma_s^{1 \rightarrow 1} & \Sigma_s^{2 \rightarrow 1} & \dots & \Sigma_s^{G \rightarrow 1} \\ \Sigma_s^{1 \rightarrow 2} & \Sigma_s^{2 \rightarrow 2} & \dots & \Sigma_s^{G \rightarrow 2} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_s^{1 \rightarrow G} & \Sigma_s^{2 \rightarrow G} & \dots & \Sigma_s^{G \rightarrow G} \end{bmatrix}.$$

Similarly, for k -eigenvalue problems, we may write the fission operator as:

$$F\psi = \tilde{M}\Sigma^F\tilde{D}\psi \quad \text{where} \quad \Sigma^F = \begin{bmatrix} \chi_1\nu\Sigma_f^1 & \chi_1\nu\Sigma_f^2 & \dots & \chi_1\nu\Sigma_f^G \\ \chi_2\nu\Sigma_f^1 & \chi_2\nu\Sigma_f^2 & \dots & \chi_2\nu\Sigma_f^G \\ \vdots & \vdots & \ddots & \vdots \\ \chi_G\nu\Sigma_f^1 & \chi_G\nu\Sigma_f^2 & \dots & \chi_G\nu\Sigma_f^G \end{bmatrix}.$$

Note that when the scatter operator is represented using a higher-order P_N expansion the D and M operators will differ between the scatter and fission terms. We may now pose the fixed source and eigenvalue problems in operator form. The fixed source problem is given by:

$$(L - S)\psi = q \tag{7.2.3}$$

and the k -eigenvalue problem:

$$(L - S)\psi = \frac{1}{\gamma}F\psi \tag{7.2.4}$$

Solving these two problems using the VEM is the topic of this chapter. So far, both the energy and angular variables have been discretised to yield a system of $G \cdot K$ coupled, hyperbolic, first order PDEs. The final step in fully discretising equations (7.2.3) and (7.2.4) is to apply a spatial discretisation, in this case the NEDG-VEM. To this end, we shall first present the case where $G = K = 1$, which is a simple advection-reaction equation. Once we have established the key properties of the NEDG-VEM for this simpler problem we shall then apply them to the full multigroup, S_N equations.

7.3 The Discontinuous Galerkin VEM

In this section we shall present the NURBS-enhanced, discontinuous-Galerkin VEM (NEDG-VEM), in which we construct a space which is locally H^1 -conforming and C^0 continuous but which has no global continuity and is only L^2 -conforming globally. This is the most natural space in which to pose a VE discretisation of first-order, hyperbolic problems for which continuous Galerkin discretisations are known to be unstable. In this section we shall use the advection-reaction (AR) equation with Dirichlet boundary conditions as our model problem. We shall use this section to present the theory behind the NEDG-VEM and show how the NEDG-VEM can be used to construct stable and high-order accurate discretisations of first-order hyperbolic problems. We shall also prove the well-posedness of the NEDG-VEM discretisation and derive a priori error estimates. The theory we present in this section is largely drawn from [139, 177].

7.3.1 The Discrete Setting

In discontinuous Galerkin discretisations it is necessary to construct spaces which possess a specified local regularity on elements of the mesh but no global regularity, these are known as broken Sobolev spaces.

Definition 7.3.1. (*Broken Sobolev Space*) On mesh $\mathcal{T}_h$ the broken Sobolev space $H^s(\mathcal{T}_h)$ is defined as

$$H^s(\mathcal{T}_h) = \left\{ v \in L^2(\Omega) : v|_F \in H^s(F) \quad \forall F \in \mathcal{T}_h \right\}$$

and has the following norm and seminorm:

$$\|u\|_{s,h} = \left(\sum_{F \in \mathcal{T}_h} \|u\|_{s,F}^2 \right)^{1/2}, \quad |u|_{s,h} = \left(\sum_{F \in \mathcal{T}_h} |u|_{s,F}^2 \right)^{1/2}$$

One notable example is the broken polynomial space:

$$\mathbb{P}_p(\mathcal{T}_h) = \left\{ q_p \in L^2(\Omega) : q_p \in \mathbb{P}_p(F) \quad \forall F \in \mathcal{T}_h \right\}.$$

In these kinds of broken spaces, special care must be taken to properly specify the behaviour of member functions at the interfaces of elements; to this end we must revisit traces. Traces were formally defined section 3.5 as operators which map functions in Sobolev spaces defined on volumes $F \subset \mathbb{R}^d$ to the boundary of such volumes ∂F . In the case of the NECG-VEM, the traces of two adjacent faces restricted to a shared edge coincide. Consequently, there is no need to store the boundary values of both faces on that shared edge. In the discontinuous Galerkin VEM (DG-VEM) however, functions are only continuous within elements and are discontinuous at the interfaces between elements. Therefore the traces of two incident faces along an edge must be stored explicitly and a description of jumps and averages over interfaces between elements is required.

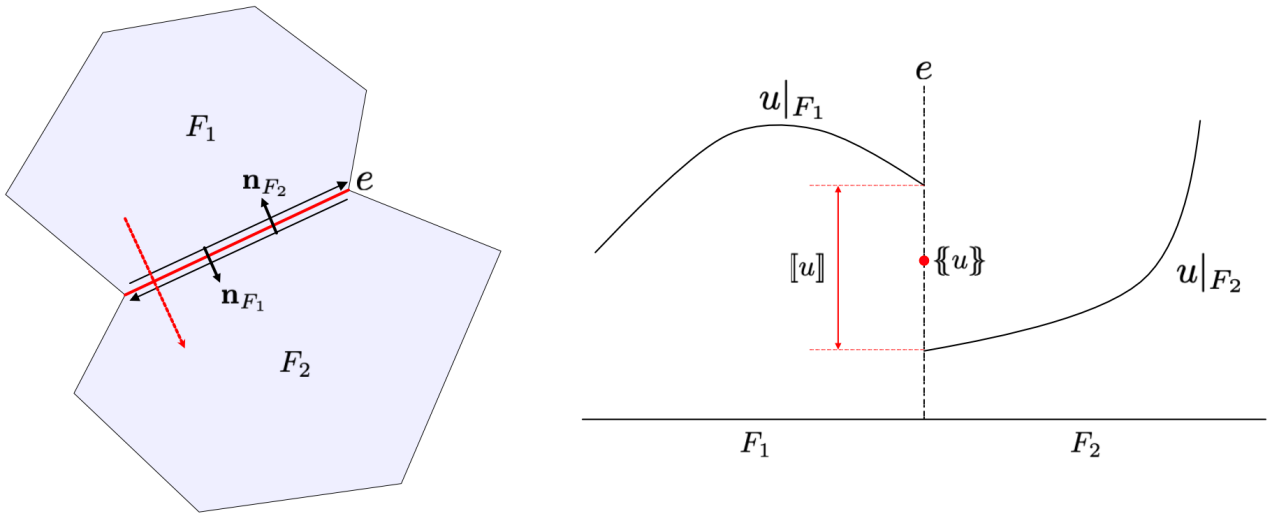


Figure 7.1: Illustration of the jump and average operators, the figure on the right plots the functions $u|_{F_1}$ and $u|_{F_2}$ as one travels across the interface edge e .

Let us begin with the definition of traces with respect to an edge e . Recalling that we have encoded each edge as a pair of half-edges with consecutive indices, the first even and the second odd. The forward orientation of a given edge e is chosen arbitrarily to be the orientation of the even half-edge. The incident faces on e , F_1 and F_2 (see figure 7.1), are ordered with F_1 being the face incident on the even half-edge and F_2 being incident on the odd half-edge. The choice of ordering is arbitrary so long as it is consistent. The normal of e , denoted $\mathbf{n}_e$, is defined as the outward normal of F_1 evaluated on e ($\mathbf{n}_{F_1}$ in figure 7.1). The traces of function v on an edge e are therefore defined as:

$$u|_{F_1} = \lim_{s \rightarrow +0} v(\mathbf{x} - s\mathbf{n}_e), \quad u|_{F_2} = \lim_{s \rightarrow +0} v(\mathbf{x} + s\mathbf{n}_e) \quad \forall \mathbf{x} \in e.$$

The jump ($[\![\cdot]\!]$) and average ($\{\!\{ \cdot \}\!\}$) operators over an edge e shared between faces F_1 and F_2 are defined as:

$$[\![u]\!] = u|_{F_1} - u|_{F_2}, \quad \{\!\{u\}\!\} = \frac{1}{2} (u|_{F_1} + u|_{F_2}).$$

In many places throughout this chapter we shall use the shorthand $u_1 = u|_{F_1}$, $u_2 = u|_{F_2}$. When deriving discontinuous Galerkin discretisations, it is common to formulate the problems locally on each face in a mesh $F \in \mathcal{T}_h$. Therefore, we also wish to define the inside trace and outside trace relative to a given face. Let F have an outward normal $\mathbf{n}_F$ defined on every point of ∂F then, for a given function v , the inner trace $(v)^+$ and outer trace $(v)^-$ are defined as:

$$(v)^+ = \lim_{s \rightarrow +0} v(\mathbf{x} - s\mathbf{n}_F), \quad (v)^- = \lim_{s \rightarrow +0} v(\mathbf{x} + s\mathbf{n}_F) \quad \forall \mathbf{x} \in \partial F. \quad (7.3.1)$$

We illustrate the meaning of the traces in figure 7.2.

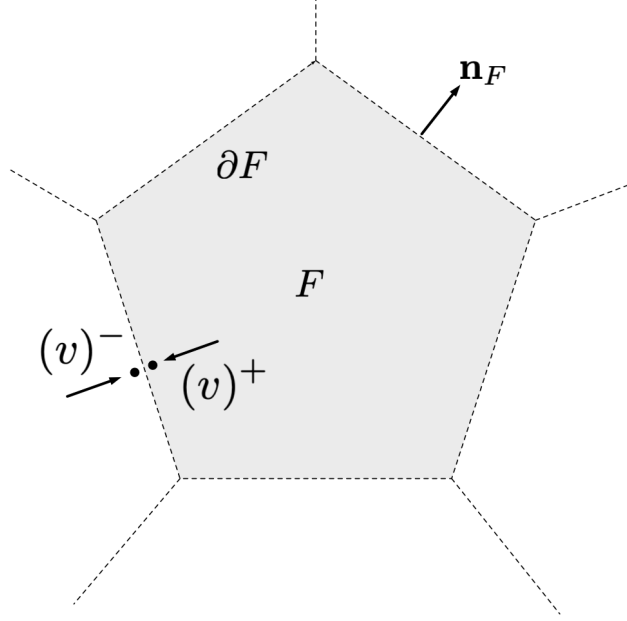


Figure 7.2: Illustration of the inner and outer traces of v with respect to F .

7.3.2 The Model Problem

The advection-reaction equation may be stated as follows:

$$\begin{aligned} \mathbf{\Omega} \cdot \nabla u + \Sigma(\mathbf{x})u &= f(\mathbf{x}) \quad \forall \mathbf{x} \in V \subset \mathbb{R}^2 \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \forall \mathbf{x} \in \partial V_- \end{aligned} \tag{7.3.2}$$

where we have split the boundary ∂V into inflow (∂V_-) and outflow (∂V_+) boundaries respectively.

$$\partial V_- := \{\mathbf{x} \in \partial V : \mathbf{\Omega} \cdot \mathbf{n} < 0\}, \quad \partial V_+ := \{\mathbf{x} \in \partial V : \mathbf{\Omega} \cdot \mathbf{n} \geq 0\}$$

and we have assumed that $f \in L^2(V)$. and that $\mathbf{\Omega}$ is a constant vector field. We define the quantities:

$$\Sigma_* = \inf_{\mathbf{x} \in V} \Sigma(\mathbf{x}), \quad \Sigma^* = \sup_{\mathbf{x} \in V} \Sigma(\mathbf{x}), \quad \Omega_\infty = \|\mathbf{\Omega}\|_\infty = \max_{i \in \{1,2\}} |(\mathbf{\Omega})_i|$$

The natural space in which to seek solutions to equation (7.3.2), in the continuous setting, is known as the graph space [177].

Definition 7.3.2. (Graph Space) *The graph space is defined as:*

$$\mathcal{G}(V) = \{u \in L^2(V) : \mathbf{\Omega} \cdot \nabla u \in L^2(V)\}.$$

and possesses the inner product:

$$(u, v)_{\mathcal{G}(V)} = (u, v)_{0,V} + (\mathbf{\Omega} \cdot \nabla u, \mathbf{\Omega} \cdot \nabla v)_{0,V},$$

and the induced graph norm: $\|u\|_{\mathcal{G}(V)}^2 = \|u\|_{0,V}^2 + \|\mathbf{\Omega} \cdot \nabla u\|_{0,V}^2$.

It may be proven (see [177], proposition 2.2) that $\mathcal{G}(V)$ is a Hilbert space. In order to meaningfully treat boundary integrals in the current setting we must also define the trace space [177]:

$$L^2_{\Omega}(\partial V) = \left\{ v \text{ measurable on } \partial V : \int_{\partial V} |\Omega \cdot \mathbf{n}| v^2 ds < \infty \right\}$$

and define the trace operator $\gamma(\cdot) : C^0(\overline{V}) \rightarrow L^2_{\Omega}(\partial V)$ where $\gamma(v) := v|_{\partial V}$. The trace operator satisfies the following trace inequality:

$$\|u\|_{L^2_{\Omega}(\partial V)} := \left(\int_{\partial V} |\Omega \cdot \mathbf{n}| u^2 ds \right)^{1/2} \leq C_{\gamma} \|u\|_V. \quad (7.3.3)$$

The proof may be found in ([177], lemma 2.5). The existence of the trace operator means that the boundary term which arises in integration by parts is well defined:

$$\int_V (\Omega \cdot \nabla u) v \, dx + \int_V u (\Omega \cdot \nabla v) \, dx = \int_{\partial V} \Omega \cdot \mathbf{n} \gamma(u) \gamma(v) \, ds.$$

We are now in a position to pose the continuous weak form:

$$\begin{cases} \text{Find } u \in \mathcal{G}(V) \text{ such that:} \\ a(u, v) = l(v) \quad \forall v \in \mathcal{G}(V) \end{cases} \quad (7.3.4)$$

where

$$\begin{aligned} a(u, v) &= \int_V \Omega \cdot \nabla u \, v \, dx + \int_V \Sigma \, uv \, dx - \int_{\partial V_-} \Omega \cdot \mathbf{n} \, uv \, ds, \\ l(v) &= \int_V v f \, dx - \int_{\partial V_-} \Omega \cdot \mathbf{n} \, vg \, ds \end{aligned}$$

We shall now present some results on the well-posedness of the continuous problem, specifically boundedness on $\mathcal{G}(V) \times \mathcal{G}(V)$ and coercivity in $L^2(V)$.

Lemma 7.3.3. (Boundedness) *The bilinear form $a(u, v)$ is bounded in $\mathcal{G}(V) \times \mathcal{G}(V)$, therefore there exists a positive real constant C such that:*

$$|a(u, v)| \leq C \|u\|_{\mathcal{G}(V)} \|v\|_{\mathcal{G}(V)}.$$

for all $u, v \in \mathcal{G}(V)$.

Lemma 7.3.4. (Coercivity in L^2) *The bilinear form $a(u, v)$ is coercive in $L^2(V)$, meaning there exists a positive constant C such that:*

$$a(u, u) \geq C \|u\|_{0,V}^2 + \frac{1}{2} \int_{\partial V} |\Omega \cdot \mathbf{n}| u^2 ds,$$

for all $u \in L^2(V)$.

The proofs for both may be found in appendix F. Finally, we state the main result of this section:

Lemma 7.3.5. (Well-posedness) *The variational problem of equation (7.3.4) is well posed.*

The proof is quite long, therefore we omit it here and refer the reader to [177] section 2.1.5.

7.3.3 The DG-VEM Discretisation

Let $\mathcal{T}_h$ be a (possibly) curved, polygonal decomposition of V , where we make assumption 6.3.1 about the material properties, then the global NEDG-VEM space on $\mathcal{T}_h$ is defined as follows:

$$\mathcal{V}_p(\mathcal{T}_h) = \left\{ v \in L^2(V) : v|_F \in \mathcal{V}_p(F) \quad \forall F \in \mathcal{T}_h \right\} \quad (7.3.5)$$

where $\mathcal{V}_p(F)$ is the local H^1 -conforming virtual element space introduced in section 4.3. Given that for each element $F \in \mathcal{T}_h$ we have $\mathcal{V}_p(F) \subset H^1(F)$ then we therefore have $\mathcal{V}_p(\mathcal{T}_h) \subset H^1(\mathcal{T}_h)$. Given that we have also assumed that any $F \in \mathcal{T}_h$ may be a curved polygon we can only assume that $\mathbb{P}_0(F) \subset \mathcal{V}_p(F)$ and that in general $\mathbb{P}_p(F) \not\subset \mathcal{V}_p(F)$. Letting $v^h \in \mathcal{V}_p(\mathcal{T}_h)$, discretising equation (7.3.2) in the DG framework begins by multiplying equation (7.3.2) with v^h , integrating over each face $F \in \mathcal{T}_h$ and summing over all faces:

$$\sum_{F \in \mathcal{T}_h} \left\{ \int_F (\boldsymbol{\Omega} \cdot \nabla u \, v^h + \Sigma(\mathbf{x}) u \, v^h) \, dx \right\} = \int_V v^h f \, dx.$$

Next, we apply integration by parts and Gauss' theorem to the advection term to give:

$$\sum_{F \in \mathcal{T}_h} \left\{ - \int_F u \, \boldsymbol{\Omega} \cdot \nabla v^h \, dx + \int_F \Sigma u \, v^h \, dx + \int_{\partial F} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds \right\} = \int_V v^h f \, dx \quad (7.3.6)$$

where $(\cdot)^+$ is the inner trace with respect to face F defined in equation (7.3.1). Let $\partial F_-^b = \partial F_- \cap \partial V_-$ denote the part of inflow boundary of F which is incident with the inflow boundary of V and let $\partial F_-^i = \partial F_- / \partial V_-$ denote the part of the inflow boundary which lies in the interior of V . Using this notation, we split the boundary integral as follows:

$$\begin{aligned} & \int_{\partial F} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds \\ &= \\ & \int_{\partial F_-^i} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds + \int_{\partial F_-^b} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds + \int_{\partial F_+} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds. \end{aligned} \quad (7.3.7)$$

On the internal inflow boundary ∂F_-^b we may replace the inner trace $(u)^+$ with the outer trace $(u)^-$. Doing so is called **upwinding** as the outer trace on ∂F_-^i also gives the upwind values of u . Additionally, on ∂V_-^b we may replace u with g . Applying these changes to equation (7.3.7) yields:

$$\int_{\partial F} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds$$

$$\begin{aligned}
&= \\
&\int_{\partial F_-^i} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^- \, (v^h)^+ \, ds + \int_{\partial F_-^b} \boldsymbol{\Omega} \cdot \mathbf{n} \, g \, (v^h)^+ \, ds + \int_{\partial F_+} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds.
\end{aligned} \tag{7.3.8}$$

Substituting equation (7.3.8) into equation (7.3.6) gives:

$$\begin{aligned}
&\sum_{F \in \mathcal{T}_h} \left\{ - \int_F u \, \boldsymbol{\Omega} \cdot \nabla v^h \, dx + \int_F \Sigma u \, v^h \, dx + \int_{\partial F_+} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+ \, (v^h)^+ \, ds \right\} \\
&= \\
&\sum_{F \in \mathcal{T}_h} \left\{ \int_F v^h f \, dx - \int_{\partial F_-^i} \boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^- \, (v^h)^+ \, ds - \int_{\partial F_-^b} \boldsymbol{\Omega} \cdot \mathbf{n} \, g \, (v^h)^+ \, ds \right\}
\end{aligned} \tag{7.3.9}$$

which we may write more succinctly as:

$$\sum_{F \in \mathcal{T}_h} \left\{ a_F(u, v^h) + \left(\boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^+, (v^h)^+ \right)_{0, \partial F_+} \right\} = \sum_{F \in \mathcal{T}_h} \left\{ (f, v^h)_{0, F} - \left(\boldsymbol{\Omega} \cdot \mathbf{n} \, (u)^\dagger, (v^h)^+ \right)_{0, \partial F_-} \right\} \tag{7.3.10}$$

where

$$a_F(u, v) = - (u, \boldsymbol{\Omega} \cdot \nabla v)_{0, F} + \Sigma_F(u, v)_{0, F}, \quad (u)^\dagger(\mathbf{x}) = \begin{cases} g(\mathbf{x}), & \text{if } \mathbf{x} \in \partial V_- \\ (u(\mathbf{x}))^- & \text{otherwise} \end{cases}.$$

To discretise equation (7.3.10) using the VEM, we must replace $a_F(u, v)$ with a computable bilinear form $a_F^h(u, v)$ as follows:

$$a_F^h(u, v^h) = - \left(\Pi_p^0 u, \boldsymbol{\Omega} \cdot \left(\Pi_p^0 \nabla v^h \right) \right)_{0, F} + \Sigma_F \left(\Pi_p^0 u \, \Pi_p^0 v^h \right)_{0, F} + c_F S_F \left(u - \Pi_p^0 u, v^h - \Pi_p^0 v^h \right). \tag{7.3.11}$$

where $c_F = \Sigma_F h_F^2 + \Omega_\infty^2$ and the stabilisation term is given by:

$$S_F(u, v) = \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(u) \, \text{dof}_i(v).$$

The VEM discretisation of equation (7.3.4) may be stated as:

$$\begin{cases} \text{Find } u^h \in \mathcal{V}_p(\mathcal{T}_h) \text{ such that:} \\ A^h(u^h, v^h) = l^h(v^h) \quad \forall v^h \in \mathcal{V}_p(\mathcal{T}_h) \end{cases} \tag{7.3.12}$$

where

$$\begin{aligned}
A^h(u^h, v^h) &= \sum_{F \in \mathcal{T}_h} \left\{ a_F^h(u^h, v^h) + \left(\boldsymbol{\Omega} \cdot \mathbf{n} \, (u^h)^+, (v^h)^+ \right)_{0, \partial F_+} \right\}, \\
l^h(v^h) &= \sum_{F \in \mathcal{T}_h} \left\{ (f, \Pi_p^0 v^h)_{0, F} - \left(\boldsymbol{\Omega} \cdot \mathbf{n} \, u^\dagger, (v^h)^+ \right)_{0, \partial F_-} \right\}
\end{aligned}$$

7.3.4 Well Posedness and DG Error Estimate

In this section we shall discretise equation (7.3.9) in a different, but mathematically equivalent way which shall ease the proof of inf-sup stability of the upwinded scheme we presented in the previous section. We stress that the formulation we present in this section is **not** what we used to solve the advection-reaction equation in practice. In practice we used the formulation presented in section 7.3.3, with the equations formulated on a face-by-face basis such that a sweep solver could be implemented.

In order to analyse the stability of the DG-VEM, and the effect of upwinding, we shall follow the procedure first presented in [288], where we express the boundary integral in equation (7.3.6) into equivalent integrals over mesh edges. Letting each edge be shared between two faces F_1 and F_2 such that $e = \partial F_1 \cap \partial F_2$ and letting $v_i = v|_{F_i}$ for $i \in \{1, 2\}$ we have:

$$\begin{aligned} \sum_{F \in \mathcal{T}_h} \int_{\partial F} (\mathbf{\Omega} \cdot \mathbf{n}_F) (u)^+ (v^h)^+ ds \\ = \\ \sum_{e \in \mathcal{E}_h^i} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) \llbracket uv^h \rrbracket ds + \sum_{e \in \mathcal{E}_h^b} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) uv^h ds. \end{aligned} \quad (7.3.13)$$

Note, that on the boundary edge we removed the face subscript as the functions are single-valued. At this stage we introduce the identity:

$$\llbracket uv^h \rrbracket = u_1 v_1^h - u_2 v_2^h = \llbracket u \rrbracket \llbracket v^h \rrbracket + \llbracket u \rrbracket \{ \{ v^h \} \}$$

and split the boundary-edge terms of equation (7.3.13) into the inflow boundary, where there are incoming boundary conditions, and the outflow boundary to give:

$$\begin{aligned} \sum_{F \in \mathcal{T}_h} \int_{\partial F} (\mathbf{\Omega} \cdot \mathbf{n}_F) (u)^+ (v^h)^+ ds \\ = \\ \sum_{e \in \mathcal{E}_h^i} \left\{ \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) \llbracket u \rrbracket \llbracket v^h \rrbracket ds + \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) \llbracket u \rrbracket \{ \{ v^h \} \} ds \right\} \\ + \sum_{e \subset \partial V_+} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) v^h u ds + \sum_{e \subset \partial V_-} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) v^h g(\mathbf{x}) ds. \end{aligned} \quad (7.3.14)$$

It may be shown that (see [177], lemma 2.14) that the exact solution satisfies: $\mathbf{\Omega} \cdot \mathbf{n} \llbracket u \rrbracket = 0$ for all internal edges $e \in \mathcal{E}_h^i$. Therefore, equation (7.3.14) becomes:

$$\begin{aligned} \sum_{F \in \mathcal{T}_h} \int_{\partial F} (\mathbf{\Omega} \cdot \mathbf{n}_F) (u)^+ (v^h)^+ ds \\ = \\ \sum_{e \in \mathcal{E}_h^i} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) \llbracket u \rrbracket \llbracket v^h \rrbracket ds + \sum_{e \subset \partial V_+} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) v^h u ds + \sum_{e \subset \partial V_-} \int_e (\mathbf{\Omega} \cdot \mathbf{n}_e) v^h g(\mathbf{x}) ds. \end{aligned} \quad (7.3.15)$$

Substituting equation (7.3.15) into equation (7.3.6) and defining the following bilinear forms:

$$\begin{aligned} a(u, v^h) &= \sum_{F \in \mathcal{T}_h} \left\{ - \int_F u \, \boldsymbol{\Omega} \cdot \nabla v^h \, dx + \Sigma_F \int_F u \, v^h \, dx \right\} \\ b(u, v^h) &= \sum_{e \in \mathcal{E}_h^i} \int_e (\boldsymbol{\Omega} \cdot \mathbf{n}_e) \llbracket u \rrbracket \llbracket v^h \rrbracket \, ds + \sum_{e \subset \partial V_+} \int_e (\boldsymbol{\Omega} \cdot \mathbf{n}_e) u \, v^h \, ds \\ (f, v^h)_{0,V} &= \int_V v^h f \, dx, \quad (g, v^h)_{\partial V_-} = \sum_{e \subset \partial V_-} \int_e (\boldsymbol{\Omega} \cdot \mathbf{n}_e) v^h g(\mathbf{x}) \, ds, \end{aligned}$$

we may then pose equation (7.3.6) as:

$$a(u, v^h) + b(u, v^h) = (f, v^h)_V - (g, v^h)_{\partial V_-} \quad (7.3.16)$$

Equation (7.3.16) is stable only in the L_2 norm ([288], [177] section 2.2). This means that any discontinuity in the boundary data will cause large, spurious oscillations which shall pollute the numerical solution. An alternative view of equation (7.3.16) is to consider it in terms of numerical fluxes. Specifically, the term $(\boldsymbol{\Omega} \cdot \mathbf{n}_e) \llbracket u \rrbracket$ in $b(v^h, u)$ corresponds to a centred numerical flux [177], whereby the numerical flux across interface $e = \partial F_1 \cap \partial F_2$ has an equal weighting of u_1 and u_2 . To make this idea more precise, we may rewrite $b(u, v^h)$ as:

$$b(u, v^h) = \sum_{e \in \mathcal{E}_h^i} \int_e \mathbf{n} \cdot \mathbf{F}(u) v^h \, ds + \sum_{e \subset \partial V_+} \int_e (\boldsymbol{\Omega} \cdot \mathbf{n}_e) u \, v^h \, ds \quad (7.3.17)$$

where $\mathbf{F}(u)$ is a numerical flux which depends upon the traces of the solution u . In the case of centred fluxes we have: $\mathbf{F}(u) = \boldsymbol{\Omega} \llbracket u \rrbracket$. The benefit of rewriting $b(v^h, u)$ in terms of numerical fluxes is that we may stabilise equation (7.3.16) in a stronger norm than $L^2(V)$ by making a different choice of numerical flux. The choice made in many settings, including computational fluid dynamics and neutron transport is the upwind numerical flux which is given by [288, 177]:

$$\mathbf{F}^\uparrow(u) = \begin{cases} \boldsymbol{\Omega} u_1 & \text{if } \boldsymbol{\Omega} \cdot \mathbf{n}_1 < 0 \\ \boldsymbol{\Omega} u_2 & \text{if } \boldsymbol{\Omega} \cdot \mathbf{n}_1 > 0 \end{cases}.$$

Therefore, depending on whether $\boldsymbol{\Omega}$ is incoming or outgoing across edge e , the numerical flux will always weight the upwind values of u and neglect the downwind values of u . The upwind numerical flux may be written as:

$$\mathbf{F}^\uparrow(u) = \boldsymbol{\Omega} \llbracket u \rrbracket + \frac{1}{2} |\boldsymbol{\Omega} \cdot \mathbf{n}| \llbracket u \rrbracket. \quad (7.3.18)$$

Substituting equation (7.3.18) into equation (7.3.17) gives the upwinded bilinear form [177]:

$$b^\uparrow(u, v^h) = \sum_{e \in \mathcal{E}_h^i} \left\{ \int_e (\boldsymbol{\Omega} \cdot \mathbf{n}_e) \llbracket u \rrbracket \llbracket v^h \rrbracket \, ds + \int_e \frac{1}{2} |\boldsymbol{\Omega} \cdot \mathbf{n}| \llbracket u \rrbracket \llbracket v^h \rrbracket \, ds \right\} + \sum_{e \subset \partial V_+} \int_e (\boldsymbol{\Omega} \cdot \mathbf{n}_e) u \, v^h \, ds$$

Therefore, upwinding is equivalent to the addition of a jump penalty term. This equivalence

eases the derivation of discrete stability [288] as we shall soon see. The upwinded, discrete variational problem therefore reads:

$$\begin{cases} \text{Find } u^h \in \mathcal{V}_p(\mathcal{T}_h) \text{ such that:} \\ a(u^h, v^h) + b^\uparrow(u^h, v^h) = (f, v^h)_{0,V} - (g, v^h)_{\partial V_-}, \quad \forall v^h \in \mathcal{V}_p(\mathcal{T}_h) \end{cases}$$

We are now in a position to state the NEDG-VEM variational approximation of equation (7.3.2), letting:

$$\begin{aligned} A^h(u^h, v^h) &= \sum_{F \in \mathcal{T}_h} \{a_F^h(u^h, v^h)\} + b^\uparrow(u^h, v^h) \\ l^h(v^h) &= (f, \Pi_p^0 v^h)_{0,V} - (g, v^h)_{\partial V_-} \end{aligned}$$

where $a_F^h(u^h, v^h)$ was introduced in equation (7.3.11). The DG-VEM variational problem may be stated as:

$$\begin{cases} \text{Find } u^h \in \mathcal{V}_p(\mathcal{T}_h) \text{ such that:} \\ A^h(u^h, v^h) = l^h(v^h), \quad \forall v^h \in \mathcal{V}_p(\mathcal{T}_h) \end{cases} \quad (7.3.19)$$

With this alternative but equivalent DG-VEM variational problem presented, we must now prove its well-posedness and also derive an error estimate in a discrete norm. Let $\mathcal{W}(h) = \mathcal{G}(V) + \mathcal{V}_p(\mathcal{T}_h)$, where $\mathcal{G}(V)$ is the graph space introduced in definition 7.3.2, and let us define the following norm on $\mathcal{W}(h)$ [177]:

$$\|v\|_{dg}^2 = \|v\|_{0,V}^2 + \frac{1}{2} \int_{\partial V} |\mathbf{\Omega} \cdot \mathbf{n}| v^2 ds + \sum_{e \in \mathcal{E}_h^i} \int_e \frac{1}{2} |\mathbf{\Omega} \cdot \mathbf{n}| \llbracket v \rrbracket^2 ds + \sum_{F \in \mathcal{T}_h} h_F \|\mathbf{\Omega} \cdot \nabla v\|_{0,F}^2.$$

We now present the main result of this section:

Theorem 7.3.6. (Well Posedness) *The discrete problem of equation (7.3.19) satisfies the discrete inf-sup condition, i.e. there exists a positive constant $C_{stab} > 0$ such that*

$$\inf_{u^h \in \mathcal{V}_p(\mathcal{T}_h)} \sup_{v^h \in \mathcal{V}_p(\mathcal{T}_h)} \frac{A^h(u^h, v^h)}{\|u^h\|_{dg} \|v^h\|_{dg}} \geq C_{stab}.$$

Problem (7.3.19) is therefore well posed.

Finally, we present the error estimate for problem (7.3.19):

Theorem 7.3.7. *Let $u \in H^{m+1}$ be the solution to the variational problem of equation (7.3.4) and $u^h \in \mathcal{V}_p(\mathcal{T}_h)$ be the solution to the discrete variational problem of equation (7.3.19). We define $s = \min(m, p)$ and let $f \in H^{m-1}$ and $g \in H^{m-3/2}(\partial V)$ then the following estimate holds:*

$$\|u - u^h\|_{dg} \leq Ch^{s-1/2} (|u|_{s,V} + |f|_{s,V})$$

Note that the theoretical order of convergence is less than what the approximation properties of the space $\mathcal{V}_p(\mathcal{T}_h)$ would suggest.

7.4 The Transport Sweep

The localised nature of DG methods, coupled with the property of first-order problems to propagate information directionally, both suggest an efficient method of solving equation (7.3.12). This method, first presented for unstructured triangular meshes in the seminal works of Reed and Hill [81] and Lesaint and Raviart [82], is known as the **transport sweep**. The basic idea is solve equation (7.3.12) element by element in an order which guarantees at every stage that the local unknowns depend only on values which have already been computed. This is equivalent to finding an ordering of elements and degrees of freedom which make the global system of equations block lower-triangular. We give an illustration of the beginning of a transport sweep in figure 7.3.

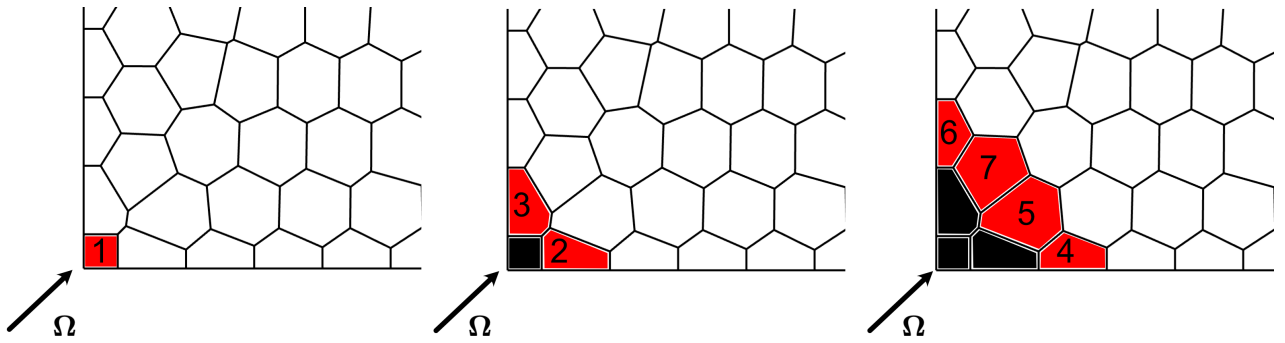


Figure 7.3: Illustration of the first six steps of a transport sweep, the elements shown in black have been solved and the ones in red are the elements to be solved.

In most cases each face $F \in \mathcal{T}_h$ may be solved in isolation. However, there are cases where two or more faces become mutually dependent on one another and have to be solved simultaneously. We discuss the former situation in section 7.4.1 and the latter in section 7.4.2. Performing a transport sweep is therefore a task scheduling problem, with each task being the assembly and solution of the local system of equations. The ordering in which the tasks are executed must be such that all upstream elements on which a task depends have already been executed when the task itself is executed. The algorithm to compute the ordering of faces has four main steps:

- Build a digraph from the mesh $\mathcal{T}_h$ using the mesh faces as vertices and where an edge $F_1 \rightarrow F_2$ means information propagates from face F_1 to F_2 .
- Identify the set of strongly-connected components (SCCs), these are cycles in the digraph.
- Compute the reduced digraph, which is the digraph where the vertices are the SCCs computed in step 2. This is guaranteed to be acyclic. Apply a depth-first search (DFS) of the reduced digraph to topologically sort the SCCs in the order that they must be solved.

These steps are the subject of sections 7.4.3, 7.4.4 and 7.4.5.

7.4.1 Local Problem on a Single Face

Let us consider the discrete problem of equation (7.3.12) but limited to a single face $F \in \mathcal{T}_h$:

$$a_F^h(u^h, v^h) + \left(\mathbf{\Omega} \cdot \mathbf{n} (u^h)^+, (v^h)^+ \right)_{0, \partial F_+} = (f, v^h)_{0, F} - \left(\mathbf{\Omega} \cdot \mathbf{n} (u^h)^\uparrow, (v^h)^+ \right)_{0, \partial F_-} \quad (7.4.1)$$

We then expand the functions out in the local basis of $\mathcal{V}_p(F)$:

$$u^h = \sum_{i=1}^{N_{\text{dof}}^F} \text{dof}_i(u^h) \varphi_i, \quad v^h = \sum_{j=1}^{N_{\text{dof}}^F} \text{dof}_j(v^h) \varphi_j.$$

Letting $(u^h)_i = \text{dof}_i(u^h)$ and substituting into equation (7.4.1) to give:

$$\sum_{j=1}^{N_{\text{dof}}^F} (v^h)_j \left\{ \sum_{i=1}^{N_{\text{dof}}^F} (u^h)_i \left(a_F^h(\varphi_i, \varphi_j) + \left(\mathbf{\Omega} \cdot \mathbf{n} (\varphi_i)^+, (\varphi_j)^+ \right)_{0, \partial F_+} \right) - (f, \varphi_j)_{0, F} + \left(\mathbf{\Omega} \cdot \mathbf{n} (u^h)^\uparrow, (\varphi_j)^+ \right)_{0, \partial F_-} \right\} = 0 \quad (7.4.2)$$

Given that v^h is arbitrary, the term inside the curly brackets in equation (7.4.2) must be zero for all $j = 1, \dots, N_{\text{dof}}^F$. Therefore, we are left with the following system of equations:

$$\sum_{i=1}^{N_{\text{dof}}^F} (u^h)_i \left\{ a_F^h(\varphi_i, \varphi_j) + \left(\mathbf{\Omega} \cdot \mathbf{n} (\varphi_i)^+, (\varphi_j)^+ \right)_{0, \partial F_+} \right\} = (f, \varphi_j)_{0, F} - \left(\mathbf{\Omega} \cdot \mathbf{n} (u^h)^\uparrow, (\varphi_j)^+ \right)_{0, \partial F_-}. \quad (7.4.3)$$

Equation (7.4.3) may be written in the following matrix form:

$$(\mathbf{L}_F + \mathbf{N}_F^+) \mathbf{u}_F = \mathbf{f}_F - \mathbf{u}_F^\uparrow$$

where $\mathbf{u}_F = [\text{dof}_1(u^h), \dots, \text{dof}_{N_{\text{dof}}^F}(u^h)]$ and $\mathbf{L}_F$ and $\mathbf{N}_F^+$ are $N_{\text{dof}}^F \times N_{\text{dof}}^F$ removal matrices given by:

$$(\mathbf{L}_F)_{i,j} = - \left(\mathbf{\Omega} \cdot \Pi_p^0 \nabla \varphi_i, \Pi_p^0 \varphi_j \right)_{0, F} + \Sigma_F \left(\Pi_p^0 \varphi_i, \Pi_p^0 \varphi_j \right)_{0, F} + c_F S_F(\varphi_i - \Pi_p^0 \varphi_i, \varphi_j - \Pi_p^0 \varphi_j) \\ (\mathbf{N}_F^+)_{i,j} = \sum_{e \subset \partial F_+} \left(\mathbf{\Omega} \cdot \mathbf{n} (\varphi_i)^+, (\varphi_j)^+ \right)_{0, e}$$

where $c_F = \Sigma_F h_F + \Omega_\infty^2$ and where the elements of $\mathbf{N}_F^+$ are zero if φ_i or φ_j do not correspond to DoFs which lie on an edge $e \subset \partial F_+$. The $N_{\text{dof}}^F \times 1$ forcing vector is given by:

$$(\mathbf{f}_F)_i = (f(\mathbf{x}), \Pi_p^0 \varphi_i)_{0, F},$$

To see how we compute the upwind source $\mathbf{u}_F^\uparrow$, consider the situation in figure 7.4. Splitting the incoming boundary of F into internal and boundary parts: $\partial F_-^i = \partial F_- / \partial V$ and $\partial F_-^b = \partial F_- \cap \partial V$. The upwind source vector is given by:

$$\left(\mathbf{u}_F^\uparrow\right)_i = \sum_{e \in \partial F_-^i} \left(\boldsymbol{\Omega} \cdot \mathbf{n} \left(\varphi_i \right)^+, \left(u^h \right)^- \right)_{0,e} + \sum_{e \in \partial F_-^b} \left(\boldsymbol{\Omega} \cdot \mathbf{n} \varphi_i, g(\mathbf{x}) \right)_{0,e} \quad (7.4.4)$$

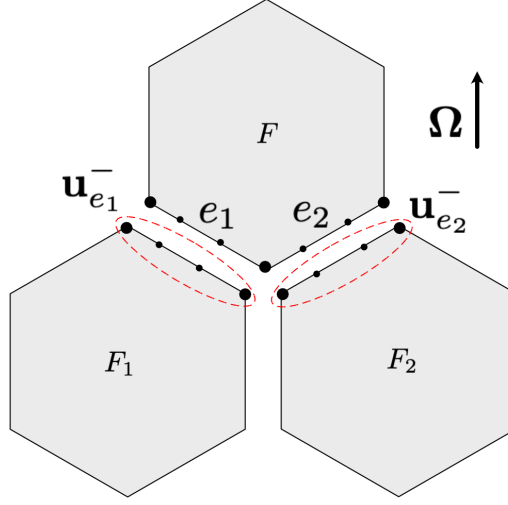


Figure 7.4: Illustration of the upwind flux vectors $\mathbf{u}_e^-$ for two edges e_1 and e_2 of face F . In this scenario the local matrix problem on faces F_1 and F_2 are assumed to have been solved.

On each edge $e \in \partial F_-^i$ the upwind flux may be expanded on the local basis on the upwind side. Letting $\mathbf{u}_e^-$ denote the $(p+1) \times 1$ vector of upwind DoFs on e we therefore have:

$$\left(u^h \right)^- = \sum_{i=1}^{p+1} \left(\mathbf{u}_e^- \right)_i \left(\varphi_i \right)^-.$$

Substituting this expansion into equation (7.4.4) gives:

$$\begin{aligned} \left(\mathbf{u}_F^\uparrow \right)_i &= \sum_{e \in \partial F_-^i} \sum_{j=1}^{p+1} \left(\boldsymbol{\Omega} \cdot \mathbf{n} \left(\varphi_i \right)^+, \left(\varphi_j \right)^- \right)_{0,e} \left(\mathbf{u}_e^- \right)_j + \sum_{e \in \partial F_-^b} \left(\boldsymbol{\Omega} \cdot \mathbf{n} \varphi_i, g(\mathbf{x}) \right)_{0,e} \\ &= \left(\mathbf{N}_F^- \mathbf{u}_F^- \right)_i + \left(\mathbf{g}_F \right)_i \end{aligned} \quad (7.4.5)$$

Let n_i be the number of edges in ∂V_-^i , then we may define the $(p+1) \cdot n_i \times 1$ vector $\mathbf{u}_F^-$ as the union of each of the edge upwind vectors $\mathbf{u}_e^-$. The matrix $\mathbf{N}_F^-$ therefore has dimension $N_{\text{dof}}^F \times (p+1) \cdot n_i$ and is given by

$$\left(\mathbf{N}_F^- \right)_{i,j} = \sum_{e \in \partial F_-^i} \left(\boldsymbol{\Omega} \cdot \mathbf{n} \left(\varphi_i \right)^+, \left(\varphi_j \right)^- \right)_{0,e}. \quad (7.4.6)$$

Finally, the vector $\mathbf{g}$ is defined as:

$$\left(\mathbf{g}_F \right)_i = \sum_{e \in \partial V_-^b} \left(\boldsymbol{\Omega} \cdot \mathbf{n} \varphi_i, g(\mathbf{x}) \right)_{0,e} \quad (7.4.7)$$

7.4.2 Local Problem on a Pair of Faces

In the case where $\mathcal{T}_h$ contains only convex polygons it is possible to construct a directed, acyclic graph (DAG) from $\mathcal{T}_h$ which may then be used to topologically sort the elements $F \in \mathcal{T}_h$ to yield the sweep ordering. Non-convex polygons occur when they have one or more curved edges. In the case of curved non-convex polygons, it is possible that the corresponding digraph contains cycles. For reasons we shall explain in section 7.4.3, the kinds of meshes we employ in this chapter do not result in mutually dependent clusters of more than three elements. Therefore, we solve these mutually coupled elements simultaneously using the same small, dense linear algebra routines we use to solve for single faces. To see how this is done consider the situation shown in figure 7.5.

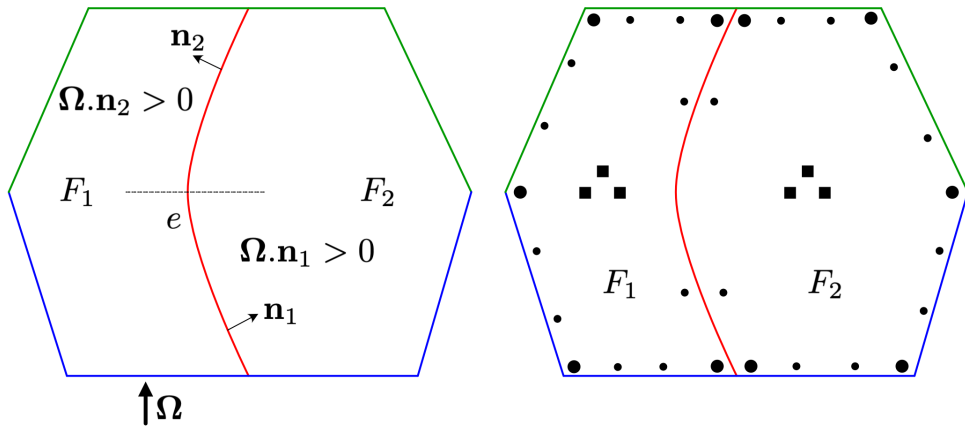


Figure 7.5: Illustration of mutual dependency resulting from curved polygons F_1 and F_2 (left) and the set of DoFs for VE order $p = 3$ (right).

In figure 7.5, the edges in $S = \mathcal{F}_1 \cup \mathcal{F}_2$ may be partitioned into three sets: the first set is the internal edge e shown in red, the second is the set of incoming edges shown in blue which we shall denote $\mathcal{E}_S^-$ and the third is the set of outgoing edges, shown in green and denoted by $\mathcal{E}_S^+$. The first $N_{\text{dof}}^{F_1}$ shape functions on $S = F_1 \cup F_2$ are those that lie on F_1 and the second $N_{\text{dof}}^{F_2}$ are those that lie on F_2 . We shall also denote the VE basis functions on F_1 and F_2 as:

$$\left\{ \varphi_i^1 \right\}_{i=1}^{N_{\text{dof}}^{F_1}}, \quad \left\{ \varphi_i^2 \right\}_{i=1}^{N_{\text{dof}}^{F_2}}$$

The local vector of unknowns on $S = F_1 \cup F_2$ is given by:

$$\mathbf{u}_S = \begin{bmatrix} \mathbf{u}_{F_1} \\ \mathbf{u}_{F_2} \end{bmatrix}$$

where $\mathbf{u}_{F_1}$ and $\mathbf{u}_{F_2}$ are the local vectors of DoFs of u^h on F_1 and F_2 . The total number of DoFs on S is denoted $N_{\text{dof}}^S = N_{\text{dof}}^{F_1} + N_{\text{dof}}^{F_2}$. and we may write the $N_{\text{dof}}^S \times N_{\text{dof}}^S$ removal operator and

the $N_{\text{dof}}^S \times 1$ force vector as:

$$\mathbf{L}_S = \begin{bmatrix} \mathbf{L}_{F_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{F_2} \end{bmatrix}, \quad \mathbf{f}_S = \begin{bmatrix} \mathbf{f}_{F_1} \\ \mathbf{f}_{F_2} \end{bmatrix}$$

To compute the coupling between F_1 and F_2 along the curved edge we first find the curve parameter t^* at which $\boldsymbol{\Omega} \cdot \mathbf{n}$ changes sign, this is accomplished using a standard Newton-method root-finder. We then split the edge e at t^* using knot insertion and generate high-order quadrature rules on each of the two halves e_1 (bottom half) and e_2 (top half). We are now in a position to compute the $N_{\text{dof}}^S \times N_{\text{dof}}^S$ coupling matrix:

$$\mathbf{N}_e = \begin{bmatrix} \mathbf{N}_e^{1 \rightarrow 1} & \mathbf{N}_e^{2 \rightarrow 1} \\ \mathbf{N}_e^{1 \rightarrow 2} & \mathbf{N}_e^{2 \rightarrow 2} \end{bmatrix}$$

where $\mathbf{N}_e^{1 \rightarrow 2}$ is an $N_{\text{dof}}^{F_2} \times N_{\text{dof}}^{F_1}$ matrix which represents the movement of information from F_1 to F_2 through e_1 and where $\mathbf{N}_e^{2 \rightarrow 1}$ is an $N_{\text{dof}}^{F_2} \times N_{\text{dof}}^{F_1}$ matrix which represents the movement of information from F_2 to F_1 through e_2 . They are defined as follows:

$$\left(\mathbf{N}_e^{1 \rightarrow 2} \right)_{i,j} = \left(\boldsymbol{\Omega} \cdot \mathbf{n}_2 \, \varphi_i^2, \varphi_j^1 \right)_{0,e_1}, \quad \left(\mathbf{N}_e^{2 \rightarrow 1} \right)_{i,j} = \left(\boldsymbol{\Omega} \cdot \mathbf{n}_1 \, \varphi_i^1, \varphi_j^2 \right)_{0,e_2}.$$

While the matrices $\mathbf{N}_e^{1 \rightarrow 1}$ and $\mathbf{N}_e^{2 \rightarrow 2}$ ($N_{\text{dof}}^{F_1} \times N_{\text{dof}}^{F_1}$ and $N_{\text{dof}}^{F_2} \times N_{\text{dof}}^{F_2}$ respectively) are the surface leakage matrices defined as:

$$\left(\mathbf{N}_e^{1 \rightarrow 1} \right)_{i,j} = \left(\boldsymbol{\Omega} \cdot \mathbf{n}_1 \, \varphi_i^1, \varphi_j^1 \right)_{0,e_1}, \quad \left(\mathbf{N}_e^{2 \rightarrow 2} \right)_{i,j} = \left(\boldsymbol{\Omega} \cdot \mathbf{n}_2 \, \varphi_i^2, \varphi_j^2 \right)_{0,e_2}$$

Additionally, we must compute the surface leakage matrix for $\mathcal{E}_S^+$ (the green edges of figure 7.5):

$$\mathbf{N}_S^+ = \begin{bmatrix} \mathbf{N}_{F_1}^+ & \mathbf{0} \\ \mathbf{0} & \mathbf{N}_{F_2}^+ \end{bmatrix}$$

where $\mathbf{N}_{F_1}^+$ and $\mathbf{N}_{F_2}^+$ have dimensions $N_{\text{dof}}^{F_1} \times N_{\text{dof}}^{F_1}$ and $N_{\text{dof}}^{F_2} \times N_{\text{dof}}^{F_2}$ respectively and are defined as:

$$\left(\mathbf{N}_{F_1}^+ \right)_{i,j} = \sum_{e \in \mathcal{E}_S^+ \cap \partial F_1} \left(\boldsymbol{\Omega} \cdot \mathbf{n}_1 \, \varphi_i^1, \varphi_j^1 \right)_{0,e}, \quad \left(\mathbf{N}_{F_2}^+ \right)_{i,j} = \sum_{e \in \mathcal{E}_S^+ \cap \partial F_2} \left(\boldsymbol{\Omega} \cdot \mathbf{n}_2 \, \varphi_i^2, \varphi_j^2 \right)_{0,e}.$$

Finally, we must compute the upwind sources on the blue edges of figure 7.5. These are computed in analogous way to the single element problem (see equations (7.4.5), (7.4.6), (7.4.7)).

$$\mathbf{u}_S^\uparrow = \begin{bmatrix} \mathbf{u}_{F_1}^\uparrow \\ \mathbf{u}_{F_2}^\uparrow \end{bmatrix}, \quad \mathbf{u}_{F_1}^\uparrow = \mathbf{N}_{F_1}^- \mathbf{u}_{F_1}^- + \mathbf{g}_{F_1}, \quad \mathbf{u}_{F_2}^\uparrow = \mathbf{N}_{F_2}^- \mathbf{u}_{F_2}^- + \mathbf{g}_{F_2}$$

where:

$$\left(\mathbf{N}_{F_k}^-\right)_{i,j} = \sum_{e \in \partial F_1 \cap \mathcal{E}_S^- / \partial V} \left(\boldsymbol{\Omega} \cdot \mathbf{n}_{F_k} \left(\varphi_i^k \right)^+, \left(\varphi_j^k \right)^- \right)_{0,e}, \quad (\mathbf{g}_{F_k})_i = \sum_{e \in \partial F_1 \cap \mathcal{E}_S^- \cap \partial V} \left(\boldsymbol{\Omega} \cdot \mathbf{n}_k \varphi_i^k, g(\mathbf{x}) \right)_{0,e}$$

for $k = 1, 2$. Using the matrices and vectors we have introduced in this section, the full system of equations on S may be written as:

$$\left(\mathbf{L}_S + \mathbf{N}_S^+ + \mathbf{N}_e \right) \mathbf{u}_S = \mathbf{f}_S - \mathbf{u}_S^\dagger$$

When $\mathbf{L}_S + \mathbf{N}_S^+ + \mathbf{N}_e$ is inverted, the mutual dependency between F_1 and F_2 is solved and we may compute the solution vector on both elements simultaneously.

7.4.3 Building the Digraph

The graph theory we present in this chapter is drawn primarily from [289, 290, 291, 292]. Let us first formally define a digraph as the pair $G = (V, E)$, where $V = \{v_1, \dots, v_{N_V}\}$ is a set of vertices and $E = \{e_1, \dots, e_{N_E}\}$ is a set of edges. Each edge is defined as a pair of vertices $e_i = (v_j, v_k)$, these edges are directed and therefore the order of the vertices matters. To ease the presentation of graph algorithms we shall use the object-oriented notation where $G.V$ accesses the set of vertices and $G.E$ accesses the set of edges. In our implementation we have used the adjacency list representation of the digraph, therefore G contains a data structure called `v2v` which stores the set of vertices connected to v in $G.v2v[v]$.

We wish to construct a digraph $G = (V, E)$ using $\mathcal{T}_h$ and a direction vector $\boldsymbol{\Omega}$ as inputs. Let us first consider the simplest case: where all the edges in $\mathcal{T}_h$ are linear and all the faces are convex. Under these assumptions the digraph is guaranteed to be acyclic. We give an example in figure 7.6:

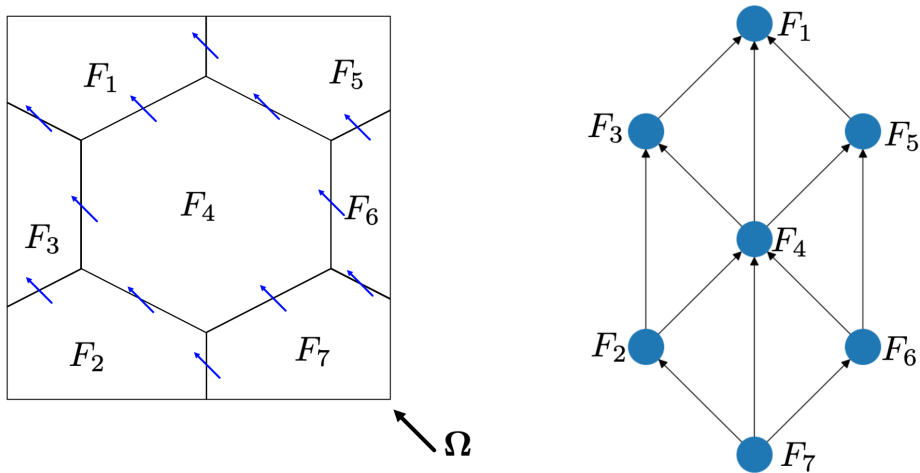


Figure 7.6: A small mesh and the sweep digraph for the direction $\boldsymbol{\Omega} = (-1, 1)$.

To compute the digraph under these assumptions, one can simply loop over the set of faces $F_i \in \mathcal{T}_h$, and for each half-edge $e \subset \partial F_i$ check the value of $\boldsymbol{\Omega} \cdot \mathbf{n}$ at the midpoint of e . If it is

negative then e is an inflow boundary to F and if it is positive then e is an outflow boundary from F . In the case of e being an outflow boundary we find the opposing face $F_j = \text{face}(\text{twin}(e))$ and add the directed edge $F_i \rightarrow F_j$ to G .

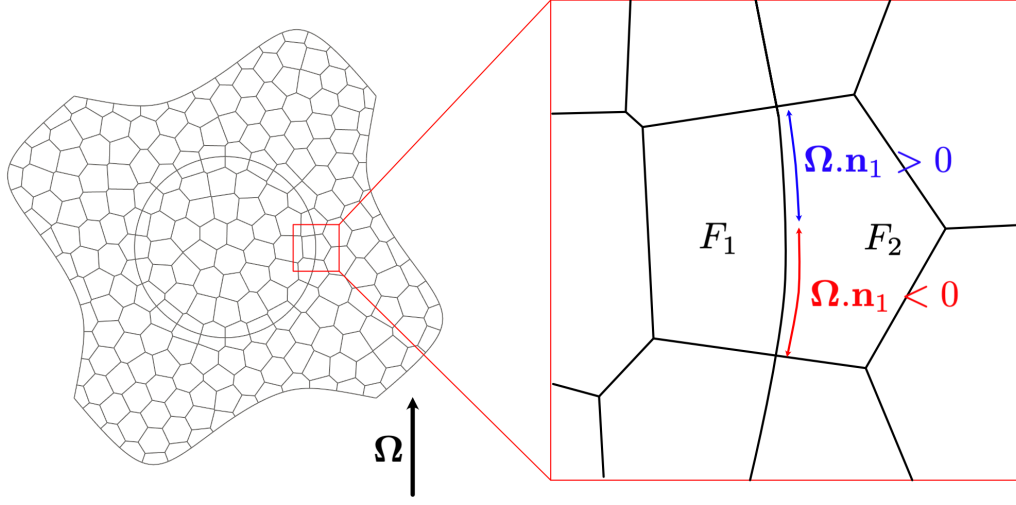


Figure 7.7: An example curvilinear mesh where the two faces shown on the right, F_1 and F_2 , form an SCC. This is because information propagates in both directions, as detected by the fact that the sign of $\Omega \cdot \mathbf{n}$ changes.

Algorithm 22 Algorithm for computing the digraph G .

```

1: procedure MAKEDIGRAPH( $\mathcal{T}_h, \Omega$ )
2:   Inputs: A mesh  $\mathcal{T}_h$  and a direction vector  $\Omega$ .
3:   Outputs: A digraph  $G$ .
4:   Add one vertex  $v_i$  to  $G$  for every face  $F_i \in \mathcal{T}_h$ .
5:   for Each face  $F_i \in \mathcal{T}_h$  do
6:     The vertex corresponding to  $F_i$  is  $v_i$ 
7:     for Each  $e \subset \partial F_i$  do
8:       Find neighbouring face  $F_j = \text{face}(\text{twin}(e))$ 
9:       The vertex corresponding to  $F_j$  is  $v_j$ 
10:      if  $e$  is curved then
11:        Evaluate  $\Omega \cdot \mathbf{n}$  at each knot on the underlying NURBS curve.
12:        if all  $\Omega \cdot \mathbf{n} > 0$  then
13:          Add edge  $e = (v_i, v_j)$  to  $G$ 
14:        else if the value of  $\Omega \cdot \mathbf{n}$  changes sign then
15:          Add edge  $e = (v_i, v_j)$  to  $G$ 
16:          Add edge  $e = (v_j, v_i)$  to  $G$ 
17:        end if
18:      else
19:        Evaluate  $\Omega \cdot \mathbf{n}$  at midpoint
20:        if  $\Omega \cdot \mathbf{n} > 0$  then add edge  $e = (v_i, v_j)$  to  $G$ 
21:      end if
22:    end for
23:  end for
24: end procedure

```

Let us next consider the case when $\mathcal{T}_h$ is curvilinear and so the resulting digraph G may con-

tain strongly connected components. These SCCs correspond to cycles of mutual dependency between elements in the mesh. In all of the meshes we have used to solve neutron transport problems, the arc length of curvilinear edges in $\mathcal{T}_h$ is small compared to the arc length of the curve in the geometry. Therefore, all of the SCCs we have so far observed have contained only two faces. We give an example of when this situation arises in figure 7.7. While it would not be difficult to contrive curvilinear polygonal meshes where SCCs contain larger numbers elements, these types of mesh simply do not occur in reactor geometries. Therefore, the assumption that SCCs remain small (2-3 elements) is a reasonable heuristic in this particular application. With this in mind, we may alter the procedure for building G to account for curvilinear edges. For each curvilinear half-edge $e \in \partial F_i$, rather than simply testing the sign of $\mathbf{\Omega} \cdot \mathbf{n}$ at the midpoint, we test the sign at each of the knots of the underlying NURBS curve. This is done so that the number of evaluations of $\mathbf{\Omega} \cdot \mathbf{n}$ scales with the geometric complexity of the curve. If all of the evaluations of $\mathbf{\Omega} \cdot \mathbf{n}$ are greater than zero then only the directed edge $F_i \rightarrow F_j$ is added to the digraph G . If the evaluations of $\mathbf{\Omega} \cdot \mathbf{n}$ change sign then the directed edges $F_i \rightarrow F_j$ and $F_j \rightarrow F_i$ are both added to G . We present the formal algorithm in algorithm 22.

7.4.4 Detecting Strongly Connected Components

In this section we shall present Kosaraju’s algorithm, which was first presented by Aho, Hopcroft and Ullman in [291] who attribute the algorithm to S.R. Kosaraju. An SCC of a digraph G is defined as a maximal subset of vertices $C \subseteq G.V$ such that for every pair $v_i, v_j \in C$ there exists a path from one to the other within C . We provide an example graph in figure 7.8 which we shall use to demonstrate the steps of Kosaraju’s algorithm.

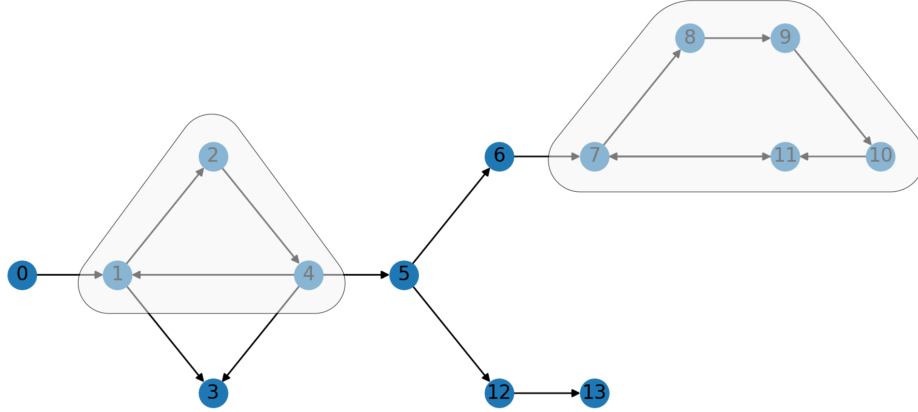


Figure 7.8: An example graph with the SCCs highlighted. The other vertices are also considered SCCs but containing a single vertex.

The fundamental building block in Kosaraju’s algorithm is the depth-first search (DFS). Typically, implementations given in textbooks present the recursive version of the DFS. However, given that some of the meshes used in this thesis are large, we opted to implement the iterative DFS so as to avoid bugs related to deep call-stacks. Following [290], we imbue vertices in $v \in G.V$ with four attributes: $v.\text{pred}$, $v.\text{color}$, $v.\text{dis}$, $v.\text{fin}$.

- The attribute $v.\text{pred}$ stores the immediate predecessor to v in the depth-first tree generated by the DFS. We set $v.\text{pred} = -1$ if v is the root of a depth first tree.
- The attribute $v.\text{color}$ stores the color of the vertex. White denotes that the vertex has not been discovered by the DFS. Gray denotes that the vertex has been discovered and its neighbours in the adjacency list are still being searched and black denotes that the vertex has been finished, meaning that both the vertex and all of its neighbours have been discovered.
- The attribute $v.\text{disc}$ stores the time-stamp when the vertex is discovered by the DFS and is turned from white to gray.
- The attribute $v.\text{fin}$ stores the time-stamp when the vertex is finished and turns from gray to black.

The time-stamp t ranges from 1 to $2N_v$ as it is incremented only when a vertex is discovered and when it is finished. We present the DFS formally in algorithm 23, where we have made use of a stack data structure S which has three operations: $S.\text{push}(u)$ adds a vertex to the top of the stack, $S.\text{pop}()$ removes the top vertex from the stack and $S.\text{top}()$ simply returns the top vertex on the stack but does not remove it. We also note that the main loop in the DFS (line 7 in algorithm 23) loops over the vertices in the order in which they are stored in $G.V$. This need not be the case, any permutation of $G.V$ is also admissible. We also note that in line 21 of algorithm 23, we loop over $G.\text{v2v}[v]$ in reverse order. We do this to ensure the same iterative DFS has the same output as the recursive DFS. Once the DFS has been called for a graph G , the discovery and finishing times are mapped to each vertex. Therefore, the order in which vertices are both discovered and finished may be found by sorting the vertices with respect to discovery and finishing times. These orderings shall be called the discovery and finishing orderings respectively and denoted by arrays D and F . In line 5 of algorithm 23 we set the predecessors of all vertices to -1. If a vertex is the root of a depth-first tree then this will remain unchanged while successor vertices will have the predecessor attribute changed. In this way we can easily detect the root of each depth-first tree in G . Kosaraju's algorithm also makes use of the digraph transpose $G^T = (V, E^T)$, which leaves the vertices unchanged but reverses the direction of each edge, therefore for every $e = (v_i, v_j) \in G.E$ we have an edge $e' = (v_j, v_i) \in (G^T).E^T$. It is a well-known result that the digraph G and its transpose G^T have the same strongly connected components [290]. We now present Kosaraju's algorithm in algorithm 24. In it we also give each vertex an additional attribute $v.\text{scc}$ which stores the ID of the SCC it belongs to. Additionally, the algorithm builds the array scc2v which stores the set of vertices that belongs to each SCC. This means that $\text{scc2v}[s]$ returns the set of vertices which belongs to the SCC with ID s .

Let us now follow the main steps of algorithm 24 using the digraph of figure 7.8 as an example. The first stage is the DFS of G and the computation of the discovery and finishing orderings

Algorithm 23 The iterative depth-first search.

```
1: procedure DFS( $G$ )
2:   Inputs: A digraph  $G$ 
3:   Outputs: The same digraph  $G$  but with the vertex attributes set.
4:   Initialise the color of all vertices  $v \in G.V$  to white
5:   Initialise the predecessor of all vertices  $v \in G.V$  to -1
6:   Initialise time counter  $t = 1$ 
7:   for  $u \in G.V$  do
8:     if  $u.\text{color} = \text{WHITE}$  then
9:       DFS-VISIT( $u$ )
10:    end if
11:  end for
12: end procedure
13: procedure DFS-VISIT( $u$ )
14:   Initialise empty stack  $S$  and call  $S.\text{push}(u)$ .
15:   while stack  $S$  is not empty do
16:      $v = S.\text{top}()$ 
17:     if  $v.\text{color} = \text{WHITE}$  then
18:       Vertex  $v$  has been discovered so set  $v.\text{color} = \text{GRAY}$ 
19:       Set discovery time  $v.\text{dis} = t$ 
20:        $t = t + 1$ 
21:       for each  $w \in G.v2v[v]$  in reverse order do
22:         if  $w.\text{color} = \text{WHITE}$  then
23:            $w.\text{pred} = v$ 
24:            $S.\text{push}(w)$ 
25:         end if
26:       end for
27:     else
28:        $S.\text{pop}()$ 
29:       if  $v.\text{color} = \text{GRAY}$  then
30:         Vertex  $v$  has been finished so set  $v.\text{color} = \text{BLACK}$ 
31:          $v.\text{fin} = t$ 
32:          $t = t + 1$ 
33:       end if
34:     end if
35:   end while
36: end procedure
```

▷

which are:

$$D = [0, 1, 2, 4, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13]$$

$$F = [3, 11, 10, 9, 8, 7, 6, 13, 12, 5, 4, 2, 1, 0]$$

Next, we compute the transpose G^T which is shown in figure 7.9 and we perform a DFS on

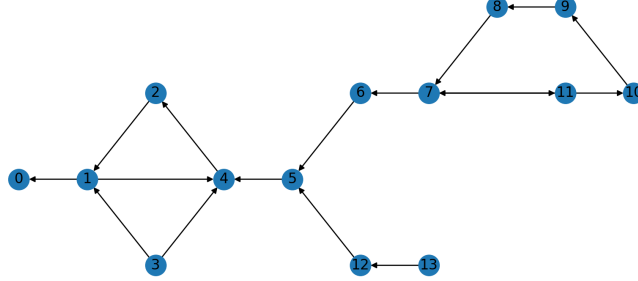


Figure 7.9: Transpose graph G^T .

G^T . However, in this second DFS we loop over the vertices (line 7 in algorithm 23) in the order $\text{reverse}(F) = [0, 1, 2, 4, 5, 12, 13, 6, 7, 8, 9, 10, 11, 3]$. Finally, in the loop of line 9 in algorithm 24, we loop over the vertices in the discovery order D and increment the SCC identifier, `sccid`, every time $v'.\text{pred} = -1$, which signifies a new depth-first tree root has been detected. Inside the loop we then set $v.\text{scc} = \text{sccid}$ and add v to the `scc2v` array. In table 2 we present the ordering D and the values of $v'.\text{pred}$ that would be encountered. Once Kosaraju's algorithm

Table 7.1: Predecessors detected when traversing G

Vertex $v \in D$	0	1	2	4	3	5	6	7	8	9	10	11	12	13
Predecessor $v'.\text{pred}$	-1	-1	4	1	-1	-1	-1	-1	9	10	11	7	-1	-1

concludes the value of `scc2v` would be:

$$\text{scc2v} = [\{0\}, \{1, 2, 4\}, \{3\}, \{5\}, \{6\}, \{7, 8, 9, 10, 11\}, \{12\}, \{13\}]$$

7.4.5 Finding the Reduced Graph and the Final Ordering

Once Kosaraju's algorithm has been performed and the array `scc2v` has been computed, it is straightforward to compute the reduced graph, which we shall denote with $\tilde{G} = (\tilde{V}, \tilde{E})$. Each vertex $\tilde{v} \in \tilde{V}$ corresponds to an SCC in G . Let each $\tilde{v}_i \in \tilde{V}$ correspond to an SCC $C_i \subseteq G.V$, then there is an edge $\tilde{e} = (\tilde{v}_i, \tilde{v}_j)$ if there exists a $u \in C_i$ and a $v \in C_j$ such that the edge $e = (u, v) \in G.E$. We present the algorithm to construct the reduced graph in algorithm 25. The reduced graph is acyclic, therefore the DFS may be used to topologically sort it into a depth-first tree. It is precisely this ordering of SCCs in the reduced graph which corresponds to the order in which elements must be solved in the transport sweep, with multi-element SCCs solved using the method presented in section 7.4.2.

Algorithm 24 Kosaraju's algorithm

```
1: procedure FINDSTRONGLYCONNECTEDCOMPONENTS( $G$ )
2:   Inputs: A digraph  $G$ 
3:   Outputs: List of SCC vertices scc2v.
4:   Call DFS( $G$ ).
5:   Find the discovery and finishing orderings of the DFS of  $G$ , denoted  $D$  and  $F$ .
6:   Compute the transpose graph  $G^T$ 
7:   Call DFS( $G^T$ ) but in the reverse order of  $F$ .  $\triangleright$  This shall set the  $v'.pred$  attributes for
   all  $v' \in G^T.V$ .
8:   Set sccid = -1
9:   for  $v \in D$  do
10:    Let  $v' \in G^T$  be the same vertex as  $v \in G$ .
11:    if  $v'.pred = -1$  then  $\triangleright$  If new DF tree root detected
12:      sccid = sccid + 1
13:    end if
14:     $v.scc$  = sccid
15:    Insert  $v$  into set of vertices stored in scc2v[sccid].
16:  end for
17: end procedure
```

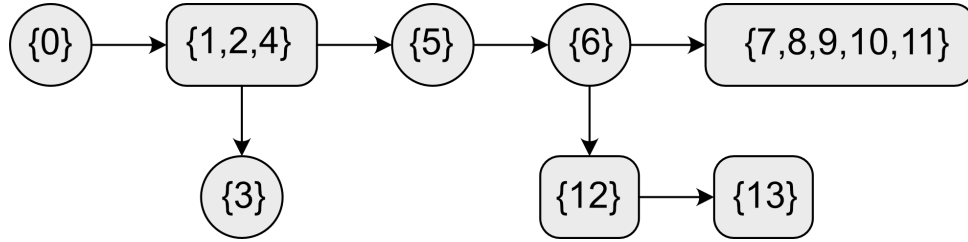


Figure 7.10: The reduced graph of figure 7.8

7.5 The Virtual Element Discretisation of the NTE

This section shall derive the global system of equations which result from discretising the multigroup, S_N NTE with the NEDG-VEM. We stress that in practice, these global matrices are not assembled and stored. Rather, they are solved in a matrix-free manner using the transport sweep presented in section 7.4. Additionally, we diverge from the discussion we presented in chapter 6 where we first presented the infinite dimensional variational problem (section 6.4) and then discretised it using the VEM (section 6.5). The reason for this is that the NEDG-VEM variational problem for the NTE does not inherit its well-posedness from the continuous variational problem for the NTE. Rather, well-posedness has to be proven in mesh-dependent norms which mimic the well-posedness of the continuous problem at the discrete level. Therefore, we only present the NEDG-VEM variational formulation of the NDE and omit a discussion of the continuous variational form of the NTE.

First, we define the Cartesian-product NEDG-VEM space $\mathcal{W}_p(\mathcal{T}_h) := [\mathcal{V}_p(\mathcal{T}_h)]^{G \cdot K}$ where $\mathcal{V}_p(\mathcal{T}_h)$ is the global NEDG-VEM space on mesh $\mathcal{T}_h$ defined in equation (7.3.5). Every $u \in \mathcal{W}_p(\mathcal{T}_h)$ is

Algorithm 25 Algorithm to compute the reduced graph.

```

1: procedure BUILDREDUCEDGRAPH( $G$ ,  $\text{scc2v}$ )
2:   Inputs: A digraph  $G$  and the SCC-to-vertex connectivity  $\text{scc2v}$ 
3:   Outputs: The reduced graph  $\tilde{G}$ .
4:   Insert one vertex to  $\tilde{G}.\tilde{V}$  for every SCC in  $G$ .
5:   for  $u \in G.V$  do
6:     for  $v \in G.\text{v2v}[u]$  do
7:       if  $u.\text{scc} \neq v.\text{scc}$  then
8:         Let  $\tilde{u} = u.\text{scc}$ 
9:         Let  $\tilde{v} = v.\text{scc}$ 
10:        Let  $\tilde{e} = (\tilde{u}, \tilde{v})$ 
11:        If  $\tilde{e} \notin \tilde{G}.\tilde{E}$  then add  $\tilde{e}$  to  $\tilde{G}$ .
12:      end if
13:    end for
14:  end for
15: end procedure

```

ordered in accordance with equation (7.2.2):

$$u = [u_1, \dots, u_G], \quad \text{where} \quad u_g = [u_{g,1}, \dots, u_{g,K}].$$

Following chapter 6 we make three assumptions about the computational mesh $\mathcal{T}_h = (\mathcal{V}_h, \mathcal{E}_h, \mathcal{F}_h)$ in addition to assumption 4.2.1:

Assumption 7.5.1. *Material properties:* the macroscopic neutron cross sections are all constant, finite and positive within each material sub-domain. For example within a nuclear fuel pin or the moderator.

Assumption 7.5.2. *Face Conformity:* in $\mathcal{T}_h$ there exists no polygon that crosses a material boundary. This means that every polygon must reside entirely within a material sub-domain and must conform to internal boundaries.

Assumption 7.5.3. *Edge Conformity:* the set of edges in $\mathcal{E}_h$ that lie on the boundary ∂V may be split into two disjoint sets of edges $\mathcal{E}_R$ and $\mathcal{E}_V$:

$$\mathcal{E}_V = \{e \in \mathcal{E}_V : e \subset \Gamma_V\} \quad \mathcal{E}_R = \{e \in \mathcal{E}_V : e \subset \Gamma_R\}$$

that lie solely within Γ_R and Γ_V respectively. In other words, there exists no boundary edge that crosses the interface between Γ_R and Γ_V .

We also define the following notation:

$$\underline{\Sigma}_X^g = \inf_{\mathbf{x} \in V} \Sigma_X^g(\mathbf{x}), \quad \overline{\Sigma}_X^g = \sup_{\mathbf{x} \in V} \Sigma_X^g(\mathbf{x})$$

where Σ_X^g is a placeholder for any macroscopic neutron cross section. Additionally, we shall continue to use $\mathcal{E}_h^i$ and $\mathcal{E}_h^b$ to denote the set of internal and boundary edges in $\mathcal{T}_h$ respectively.

To ease the notation in this section we shall define the following inner product notation:

$$(u, v)_F = (u, v)_{0,F}, \quad \langle u, v \rangle_e^k = (\mathbf{\Omega}_k \cdot \mathbf{n} u, v)_{0,e}, \quad \langle u, v \rangle_\Gamma^k = \sum_{e \in \Gamma} (\mathbf{\Omega}_k \cdot \mathbf{n} u, v)_{0,e}$$

where Γ can be any collection of edges, for example ∂V_- or $\mathcal{E}_h^i$. Let us recall the multigroup, S_N-NTE for group-ordinate (g, k) :

$$\mathbf{\Omega}_k \cdot \nabla \psi_{g,k}(\mathbf{x}) + \Sigma_t^g(\mathbf{x}) \psi_{g,k}(\mathbf{x}) = \frac{1}{4\pi} \sum_{g'=1}^G \Sigma_s^{g' \rightarrow g}(\mathbf{x}) \phi_{g'} + Q_{g,k}(\mathbf{x}) \quad (7.5.1)$$

Letting $v_{g,k} \in \mathcal{V}_p(\mathcal{T}_h)$, we derive the following bilinear forms on F using the same steps presented in section 7.3.3:

$$\begin{aligned} a_{g,k,F}^h(\psi_{g,k}, v_{g,k}) &= - \left(\Pi_p^0 \psi_{g,k}, \mathbf{\Omega}_k \cdot \Pi_p^0 \nabla v_{g,k} \right)_{0,F} + \Sigma_{t,F}^g \left(\Pi_p^0 \psi_{g,k}, \Pi_p^0 v_{g,k} \right)_{0,F} \\ &\quad + c_{g,k,F} S_F(\psi_{g,k} - \Pi_p^0 \psi_{g,k}, v_{g,k} - \Pi_p^0 v_{g,k}), \end{aligned}$$

where the stability constant is given by: $c_{g,k,F} = \Sigma_{t,F}^g h_F^2 + \Omega_{k,\infty}^2$ and:

$$b_{g,k,F}^h(\psi_{g,k}, v_{g,k}) = \left\langle (\psi_{g,k})^+, (v_{g,k})^+ \right\rangle_{\partial F_+}^k + \left\langle (\psi_{g,k})^-, (v_{g,k})^+ \right\rangle_{\partial F_-^i}^k + \left\langle \psi_{g,k}, (v_{g,k})^+ \right\rangle_{\partial V_R \cap \partial F_-^b}^k.$$

Using these bilinear forms, the streaming + collision bilinear form on face F may therefore be written as

$$\hat{L}_{g,k,F}^h(\psi_{g,k}, v_{g,k}) = a_{g,k,F}^h(\psi_{g,k}, v_{g,k}) + b_{g,k,F}^h(\psi_{g,k}, v_{g,k})$$

and the weak form of equation (7.5.1) on a single face $F \in \mathcal{T}_h$ is given the following for the fixed source and k -eigenvalue cases:

- The fixed source case

$$\hat{L}_{g,k,F}^h(\psi_{g,k}, v_{g,k}) - \frac{1}{4\pi} \sum_{g'=1}^G \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 \phi_{g'}, \Pi_p^0 v_{g,k} \right)_F = \frac{1}{4\pi} \left(q_{g,k}(\mathbf{x}), \Pi_p^0 v_{g,k} \right)_F \quad (7.5.2)$$

- The k -eigenvalue case:

$$\hat{L}_{g,k,F}^h(\psi_{g,k}, v_{g,k}) - \frac{1}{4\pi} \sum_{g'=1}^G \Sigma_{s,F}^{g' \rightarrow g} \left(\Pi_p^0 \phi_{g'}, \Pi_p^0 v_{g,k} \right)_F = \frac{1}{\gamma} \frac{\chi_g}{4\pi} \sum_{g'=1}^G \nu \Sigma_{f,F}^{g'} \left(\Pi_p^0 \phi_{g'}, \Pi_p^0 v_{g,k} \right)_F \quad (7.5.3)$$

To yield the full system of equations, we must sum equations (7.5.2) and (7.5.3) over all energy groups $g = 1, \dots, G$, all ordinates $k = 1, \dots, K$ and all faces $F \in \mathcal{T}_h$. We therefore define the global removal, scatter and fission bilinear forms as follows, beginning with the removal bilinear form

$\hat{L}^h(\psi, v) : \mathcal{W}_p(\mathcal{T}_h) \times \mathcal{W}_p(\mathcal{T}_h) \rightarrow \mathbb{R}$:

$$\hat{L}^h(\psi, v) = \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{F \in \mathcal{T}_h} \hat{L}_{g,k,F}^h(\psi_{g,k}, v_{g,k}) \right\}.$$

The scatter bilinear form $\hat{S}^h(\psi, v) : \mathcal{W}_p(\mathcal{T}_h) \times \mathcal{W}_p(\mathcal{T}_h) \rightarrow \mathbb{R}$:

$$\hat{S}^h(\psi, v) = \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{F \in \mathcal{T}_h} \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \left(\Pi_p^0 \psi_{g',k'}, \Pi_p^0 v_{g,k} \right)_F \right\}.$$

The fixed source linear form $\hat{Q}^h(v) : \mathcal{W}_p(\mathcal{T}_h) \rightarrow \mathbb{R}$:

$$\hat{Q}^h(v) = \frac{1}{4\pi} \sum_{g=1}^G \sum_{k=1}^K w_k \sum_{F \in \mathcal{T}_h} \left(q_{g,k}(\mathbf{x}), \Pi_p^0 v_{g,k} \right)_F.$$

Finally, the fission bilinear form $\hat{F}^h(\psi, v) : \mathcal{W}_p(\mathcal{T}_h) \times \mathcal{W}_p(\mathcal{T}_h) \rightarrow \mathbb{R}$:

$$\hat{F}^h(\psi, v) = \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \frac{1}{4\pi} \sum_{F \in \mathcal{T}_h} \chi_{g,F} \sum_{g'=1}^G \sum_{k'=1}^K \nu \Sigma_{f,F}^{g'} w_{k'} \left(\Pi_p^0 \psi_{g',k'}, \Pi_p^0 v_{g,k} \right)_F \right\}$$

Using these global bilinear forms we may now pose the global, NEDG-VEM variational problems for both the fixed source and k -eigenvalue problems. First the discrete, variational **fixed source problem**:

$$\begin{cases} \text{Find } \psi^h \in \mathcal{W}_p(\mathcal{T}_h) \text{ which satisfies:} \\ \hat{L}^h(\psi^h, v^h) - \hat{S}^h(\psi^h, v^h) = \hat{Q}^h(v^h) \quad \forall v^h \in \mathcal{W}_p(\mathcal{T}_h) \end{cases} \quad (7.5.4)$$

Second, the discrete, variational **k -eigenvalue problem**

$$\begin{cases} \text{Find } \lambda^h \in \mathbb{R} \text{ such that there exists a } \psi^h \in \mathcal{W}_p(\mathcal{T}_h) \\ \hat{L}^h(\psi^h, v^h) - \hat{S}^h(\psi^h, v^h) = \lambda^h \hat{F}^h(\psi, v^h) \quad \forall v^h \in \mathcal{W}_p(\mathcal{T}_h) \end{cases} \quad (7.5.5)$$

7.6 Practical Implementation

In this section we shall explain how we implemented various key aspects of the NEDG-VEM discretisation of the multigroup S_N -NTE. These aspects include the enumeration and retrieval of the NEDG-VEM degrees of freedom on the mesh, the strategy for solving the global system of equations using source-iteration (SI) for the fixed source problem and power-iteration (PI) for the k -eigenvalue problem and finally we present the shared-memory parallelisation strategy based on task-scheduling using Taskflow [293].

7.6.1 Global NEDG-VEM Degrees of Freedom

In this section we shall explain how the mapping from topological entities (vertices, half-edges and faces) to degrees of freedom is performed in the case of the NEDG-VEM. Let N_v , N_e and N_f be the number of vertices, edges and faces in the mesh $\mathcal{T}_h$ respectively, and let us recall that an edge is defined as a twin-pair of half-edges in the DCEL. In the case of the NEDG-VEM, DoFs on vertices and edges are not shared between adjacent faces. While this increases the total number of degrees of freedom, it also affords us the freedom to couple the DoFs between faces in a way which suits the problem under consideration, as was the case with upwinding. Therefore, the dimension of $\mathcal{V}_p(\mathcal{T}_h)$ is given by:

$$N_{\text{g dof}} = \sum_{F \in \mathcal{T}_h} N_{\text{dof}}^F, \quad \text{where} \quad N_{\text{dof}}^F = \dim(\mathcal{V}_p(F))$$

We wish to number the DoFs in $\mathcal{V}_p(\mathcal{T}_h)$ in the **mesh numbering system** (MNS) we presented in section 6.6.1, with vertex DoFs numbered first, half-edge DoFs numbered second and face DoFs numbered last. The MNS is more complicated to compute in the NEDG-VEM as we no longer have a constant number of DoFs per vertex and edge. The number of DoFs on a vertex is equal to the number of incident faces on that vertex while the number of DoFs on an edge depends upon whether that edge is internal or on the boundary. To illustrate this point we

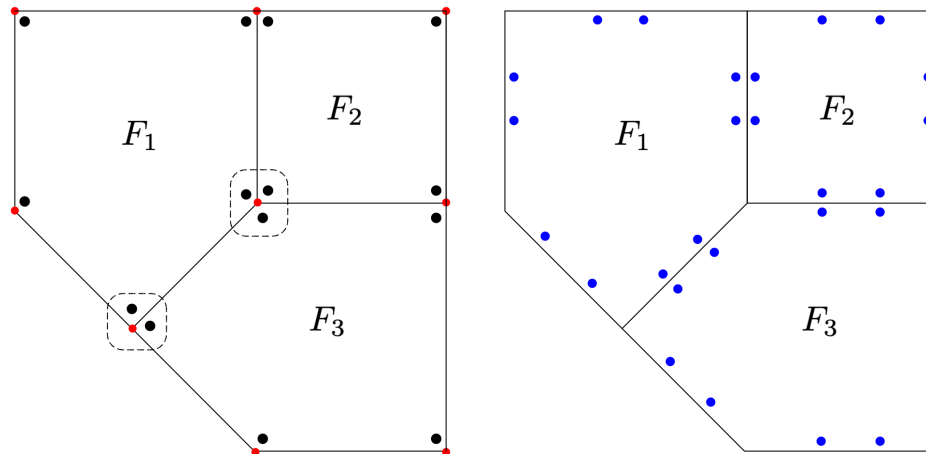


Figure 7.11: A small mesh with three faces and the degrees of freedom for a NEDG-VEM scheme of order $p = 3$, vertex DoFs are shown on the left and edge DoFs on the right

present a mesh with three elements in figure 7.11. On the left we show the vertices in red along with the vertex DoFs attached to them. On the right, we show the edge DoFs in blue attached to each edge. The MNS for the NEDG-VEM is defined as follows:

- The vertex DoFs are numbered first. As there are multiple DoFs on each vertex we therefore order them by the index of the face each DoF belongs to. We must also maintain a $(N_v + 1) \times 1$ array **v2dof** where the DoFs attached to vertex v are given by:

$$\text{dofset}(v) = \{\mathbf{v2dof}[v], \dots, \mathbf{v2dof}[v + 1] - 1\} \quad (7.6.1)$$

The total number of vertex DoFs is equal to the sum of the number of vertices on each face.

- The degrees of freedom attached to edges are numbered second. Recalling fact that each edge may be decomposed into half-edges $e_i = (he_{2i}, he_{2i+1})$ where $i = 0, \dots, N_e - 1$, then if edge e_i is internal ($e_i \in \mathcal{E}_h^i$) then there are $2(p-1)$ DoFs attached to it, one set of $(p-1)$ on each half-edge. If edge $e_i \in \mathcal{E}_h^b$ then there are only $p-1$ DoFs attached to it and the DoFs attached to one half-edge are simply the reverse of those attached to its twin, as was the case in the CG-VEM for all half-edges. We maintain a $(2N_e) \times 2$ array **he2dof** where the DoFs attached to each half-edge he are stored in:

$$\text{dofset}(he) = \{\mathbf{he2dof}[he, 0], \dots, \mathbf{he2dof}[he, 1] - 1\} \quad (7.6.2)$$

- The degrees of freedom attached to faces are numbered last and are numbered in exactly the same way they were for the CG-VEM. Therefore, we refer the reader to section 6.6.1 for the details.

The algorithm to retrieve all the DoFs on a face (algorithm 17) is unchanged in the case of the NEDG-VEM. The only difference is in how the $\text{dofset}(\cdot)$ operator is computed (equations (7.6.1) and (7.6.2)). As was the case in the CG-VEM for the elliptic equations, the MNS is a good DoF ordering for post-processing, but it is sub-optimal for the most computationally intensive part of solving the NTE: the transport sweep. It is advantageous to number DoFs on each face contiguously in the transport sweep, with the local ordering being the usual one for VEM DoFs (see figure 4.4). The reason being the improved memory access pattern when performing a transport sweep, with all locally needed data laid out nearby in memory. If there is only one direction vector $\mathbf{\Omega}$, as is the case in the advection-reaction equation then numbering contiguously by face in the sweep ordering is ideal. In the case of the S_N -NTE where there are many sweep directions, we choose to simply number DoFs on faces in the order they appear in the mesh: in this way we don't privilege any one direction over the others. As we described in section 6.6.5 with reference to the partition numbering system (PNS), we must compute a $N_{\text{g dof}} \times 1$ permutation vector $\mathbf{p}$ and its inverse $\hat{\mathbf{p}}$ which maps DoFs in the MNS to DoFs in the **sweep numbering system** (SNS) and back again. The algorithm for computing $\mathbf{p}$ and $\hat{\mathbf{p}}$ is given in algorithm 26.

Algorithm 26 Algorithm to build forward and inverse permutation vectors for the SNS.

```

1: procedure BUILDPERMUTATION( $\mathcal{T}_h, p$ )
2:   Inputs: A mesh  $\mathcal{T}_h$  and a VE order  $p$ .
3:   Outputs: A permutation vector  $\mathbf{p}$  and its inverse  $\hat{\mathbf{p}}$ .
4:   Initialise:  $\text{new\_dof} \leftarrow 0$ 
5:   Initialise:  $\mathbf{p} \leftarrow [0, \dots, 0]$ 
6:   Initialise:  $\hat{\mathbf{p}} \leftarrow [0, \dots, 0]$ 
7:   for Each  $F \in \mathcal{T}_h$  do
8:     for  $\text{old\_dof} \in \text{dofset}(F)$  do
9:        $\mathbf{p}[\text{old\_dof}] = \text{new\_dof}$ 
10:       $\hat{\mathbf{p}}[\text{new\_dof}] = \text{old\_dof}$ 
11:       $\text{new\_dof} = \text{new\_dof} + 1$ 
12:    end for
13:  end for
14: end procedure

```

7.6.2 The Global Matrices

As was the case with the NDE, the global bilinear forms $\hat{L}^h(\psi, v)$, $\hat{S}^h(\psi, v)$, $\hat{Q}^h(v)$ and $\hat{F}^h(\psi, v)$ have matrix and vector forms which we shall derive here. While this is instructive for devising iterative schemes for solving the global system of equations, we stress that these global matrices are not explicitly constructed in practice as we are implementing a transport sweep solver, which is a matrix-free algorithm. Let us expand $\psi_{g,k}$ and $v_{g,k}$ out in the basis of $\mathcal{V}_p(\mathcal{T}_h)$:

$$\psi_{g,k} = \sum_{i=1}^{N_{\text{gdof}}} \text{dof}_i(\psi_{g,k}) \varphi_i = \sum_{i=1}^{N_{\text{gdof}}} (\boldsymbol{\psi}_{g,k})_i \varphi_i, \quad v_{g,k} = \sum_{i=1}^{N_{\text{gdof}}} \text{dof}_i(v_{g,k}) \varphi_i = \sum_{i=1}^{N_{\text{gdof}}} (\mathbf{v}_{g,k})_i \varphi_i$$

We shall first derive the global removal operator, beginning with the (g, k) removal bilinear form:

$$\begin{aligned} \hat{L}_{g,k}^h(\psi_{g,k}, v_{g,k}) &= \sum_{F \in \mathcal{T}_h} \hat{L}_{g,k,F}^h(\psi_{g,k}, v_{g,k}) \\ &= \sum_{i=1}^{N_{\text{gdof}}} \sum_{j=1}^{N_{\text{gdof}}} (\mathbf{v}_{g,k})_i \left(\sum_{F \in \mathcal{T}_h} \hat{L}_{g,k,F}^h(\varphi_i, \varphi_j) \right) (\boldsymbol{\psi}_{g,k})_j \\ &= \mathbf{v}_{g,k}^T \mathbf{L}_{g,k} \boldsymbol{\psi}_{g,k} \end{aligned}$$

The local removal operator matrices $(\mathbf{R}_{g,k,F})_{i,j} = \hat{L}_{g,k,F}^h(\varphi_i, \varphi_j)$ are given by:

$$\mathbf{R}_{g,k,F} = \mathbf{L}_{g,k,F} + \mathbf{N}_F^+ + \mathbf{N}_F^-, \quad \text{where} \quad \mathbf{L}_{g,k,F} = -\mathbf{T}_F + \Sigma_{t,F}^g \mathbf{M}_F + c_{g,k,F} \mathbf{S}_F$$

and where the definition $\mathbf{T}_F$, $\mathbf{M}_F$ and $\mathbf{S}_F$ may be found in section 4.5.6. The definitions of $\mathbf{N}_F^+$ and $\mathbf{N}_F^-$ are:

$$(\mathbf{N}_F^+)_{i,j} = \sum_{e \in \partial F_+} \langle (\varphi_i)^+, (\varphi_j)^+ \rangle_e^k$$

and

$$\left(\mathbf{N}_F^-\right)_{i,j} = \sum_{e \in \partial F_- / \partial V_R} \left\langle (\varphi_i)^+, (\varphi_j)^- \right\rangle_e^k + \sum_{e \in \partial F_- \cap \partial V_R} \langle \varphi_i, \varphi_j \rangle_e^k$$

An important point to note about $\mathbf{R}_{g,k,F}$ is that, because it includes contributions from the upwind shape functions, its dimensions are larger than $N_{\text{dof}}^F \times N_{\text{dof}}^F$. More precisely, if F has N_F^- edges in ∂F_- then for a VEM scheme of order p there are $(p+1)N_F^-$ upwind DoFs which contribute terms to $\mathbf{R}_{g,k,F}$. Therefore, the dimensions of $\mathbf{R}_{g,k,F}$ are

$$(N_{\text{dof}}^F + (p+1)N_F^-) \times (N_{\text{dof}}^F + (p+1)N_F^-).$$

We illustrate the set of DoFs which contribute to $\mathbf{R}_{g,k,F}$ for a hexagonal element F with three upwind faces $F_1^\uparrow$, $F_2^\uparrow$ and $F_3^\uparrow$ in figure 7.12.

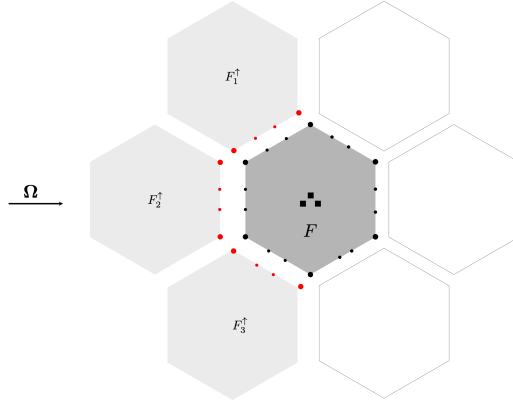


Figure 7.12: Illustration of the set of shape functions which contribute to $\mathbf{R}_{g,k,F}$. The upwind DoFs are shown in red, the local DoFs on F are shown in black.

The global removal operator is therefore given be:

$$\hat{L}^h(\psi, v) = \sum_{g=1}^G \sum_{k=1}^K w_k \left(\mathbf{v}_{g,k}^T \mathbf{L}_{g,k} \psi_{g,k} \right), \quad \text{where} \quad \mathbf{L}_{g,k} = \sum_{F \in \mathcal{T}_h} \mathbf{R}_{g,k,F}$$

Let us now derive the matrix form of the global scatter bilinear form:

$$\begin{aligned} \hat{S}^h(\psi, v) &= \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \left(\Pi_p^0 \psi_{g',k'}, \Pi_p^0 v_{g,k} \right)_F \right\} \\ &= \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \left[\sum_{i=1}^{N_{\text{gdof}}} \sum_{j=1}^{N_{\text{gdof}}} (\mathbf{v}_{g,k})_i \left(\sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \left(\Pi_p^0 \varphi_i, \Pi_p^0 \varphi_j \right)_F \right) (\psi_{g',k'})_j \right] \right\} \\ &= \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \left[\mathbf{v}_{g,k}^T \mathbf{S}_{g,k}^{g',k'} \psi_{g',k'} \right] \right\} \quad \text{where} \quad \mathbf{S}_{g,k}^{g',k'} = \sum_{F \in \mathcal{T}_h} \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \mathbf{M}_F. \end{aligned}$$

The matrix $\mathbf{S}_{g,k}^{g',k'}$ is the $(g',k') \rightarrow (g,k)$ scatter matrix, notice that the direction k does not appear in the expression for $\mathbf{S}_{g,k}^{g',k'}$. This a consequence of assuming isotropic scatter, which

means for all of the ordinates k in a given energy group g , the scatter source is the same. In the case of non-isotropic scatter, the scatter source would depend on k . Let us now derive the matrix form of the global fission bilinear form:

$$\begin{aligned}
\hat{F}^h(\psi, v) &= \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \sum_{F \in \mathcal{T}_h} \chi_{g,F} \nu \Sigma_{f,F}^{g'} w_{k'} \left(\Pi_p^0 \psi_{g',k'}, \Pi_p^0 v_{g,k} \right)_F \right\} \\
&= \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \sum_{i=1}^{N_{\text{gdof}}} \sum_{j=1}^{N_{\text{gdof}}} (\mathbf{v}_{g,k})_i \left(\sum_{F \in \mathcal{T}_h} \chi_{g,F} \nu \Sigma_{f,F}^{g'} w_{k'} \left(\Pi_p^0 \varphi_i, \Pi_p^0 \varphi_j \right)_F \right) (\psi_{g',k'})_j \right\} \\
&= \sum_{g=1}^G \sum_{k=1}^K w_k \left\{ \sum_{g'=1}^G \sum_{k'=1}^K \frac{1}{4\pi} \left[\mathbf{v}_{g,k}^T \mathbf{F}_{g,k}^{g',k'} \psi_{g',k'} \right] \right\} \quad \text{where} \quad \mathbf{F}_{g,k}^{g',k'} = \sum_{F \in \mathcal{T}_h} \chi_{g,F} \nu \Sigma_{f,F}^{g'} w_{k'} \mathbf{M}_F
\end{aligned}$$

The matrix $\mathbf{F}_{g,k}^{g',k'}$ is the $(g', k') \rightarrow (g, k)$ fission matrix. Like the scatter matrix, this is also an isotropic source so for every energy group g , the fission source is the same for all ordinates k . Finally, let us derive the vector form of the fixed source linear functional $\hat{Q}^h(v)$:

$$\begin{aligned}
\hat{Q}^h(v) &= \frac{1}{4\pi} \sum_{g=1}^G \sum_{k=1}^K w_k \sum_{F \in \mathcal{T}_h} (q_{g,k}(\mathbf{x}), \Pi_p^0 v_{g,k})_F \\
&= \frac{1}{4\pi} \sum_{g=1}^G \sum_{k=1}^K w_k \sum_{i=1}^{N_{\text{gdof}}} (\mathbf{v}_{g,k})_i \sum_{F \in \mathcal{T}_h} (q_{g,k}(\mathbf{x}), \Pi_p^0 \varphi_i)_F \\
&= \frac{1}{4\pi} \sum_{g=1}^G \sum_{k=1}^K w_k \left[\mathbf{v}_{g,k}^T \mathbf{q}_{g,k} \right] \quad \text{where} \quad \mathbf{q}_{g,k} = \sum_{F \in \mathcal{T}_h} \mathbf{q}_{g,k,F}
\end{aligned}$$

where

$$(\mathbf{q}_{g,k,F})_i = (q_{g,k}(\mathbf{x}), \Pi_p^0 \varphi_i)_F, \quad i = 1, \dots, N_{\text{dof}}^F.$$

7.6.3 The Fixed Source Problem

Substituting each of the expressions of the global matrices into equation (7.5.4) yields the following equivalent matrix problem:

$$\sum_{g=1}^G \sum_{k=1}^K w_k \mathbf{v}_{g,k}^T \left(\mathbf{L}_{g,k} \boldsymbol{\psi}_{g,k} - \frac{1}{4\pi} \sum_{g'=1}^G \sum_{k'=1}^K \mathbf{S}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'} - \frac{1}{4\pi} \mathbf{q}_{g,k} \right) = 0 \quad (7.6.3)$$

The discrete ordinate weights are assumed to be positive and the vector $\mathbf{v}_{g,k}$ is arbitrary. Therefore, the term inside the brackets in equation (7.6.8) must be equal to the $N_{\text{gdof}} \times 1$ zero-vector for every (g, k) pair. Let us define the following block vectors:

$$\begin{aligned}
\boldsymbol{\psi}_g &= [\boldsymbol{\psi}_{g,1}, \boldsymbol{\psi}_{g,2}, \dots, \boldsymbol{\psi}_{g,K}], & \boldsymbol{\psi} &= [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \dots, \boldsymbol{\psi}_G] \\
\mathbf{q}_g &= [\mathbf{q}_{g,1}, \mathbf{q}_{g,2}, \dots, \mathbf{q}_{g,K}], & \mathbf{q} &= [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_G]
\end{aligned}$$

and let us define the following block matrices:

$$\mathbf{L}_g = \begin{bmatrix} \mathbf{L}_{g,1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{g,2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_{g,K} \end{bmatrix}, \quad \mathbf{S}_{g,g'} = \begin{bmatrix} \mathbf{S}_{g,1}^{g',1} & \mathbf{S}_{g,1}^{g',2} & \cdots & \mathbf{S}_{g,1}^{g',K} \\ \mathbf{S}_{g,2}^{g',1} & \mathbf{S}_{g,2}^{g',2} & \cdots & \mathbf{S}_{g,2}^{g',K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{g,K}^{g',1} & \mathbf{S}_{g,K}^{g',2} & \cdots & \mathbf{S}_{g,K}^{g',K} \end{bmatrix}, \quad \mathbf{F}_{g,g'} = \begin{bmatrix} \mathbf{F}_{g,1}^{g',1} & \mathbf{F}_{g,1}^{g',2} & \cdots & \mathbf{F}_{g,1}^{g',K} \\ \mathbf{F}_{g,2}^{g',1} & \mathbf{F}_{g,2}^{g',2} & \cdots & \mathbf{F}_{g,2}^{g',K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}_{g,K}^{g',1} & \mathbf{F}_{g,K}^{g',2} & \cdots & \mathbf{F}_{g,K}^{g',K} \end{bmatrix}.$$

Each matrix in each block has dimensions $N_{\text{gdof}} \times N_{\text{gdof}}$, therefore $\mathbf{L}_g$, $\mathbf{S}_{g,g'}$ and $\mathbf{F}_{g,g'}$ each have dimensions $(K \cdot N_{\text{gdof}}) \times (K \cdot N_{\text{gdof}})$. Using these block matrices and vectors, equation (7.6.8) may be written as:

$$\begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_G \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \\ \vdots \\ \boldsymbol{\psi}_g \end{bmatrix} - \frac{1}{4\pi} \begin{bmatrix} \mathbf{S}_{1,1} & \mathbf{S}_{1,2} & \cdots & \mathbf{S}_{1,G} \\ \mathbf{S}_{2,1} & \mathbf{S}_{2,2} & \cdots & \mathbf{S}_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{G,1} & \mathbf{S}_{G,2} & \cdots & \mathbf{S}_{G,G} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \\ \vdots \\ \boldsymbol{\psi}_g \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_G \end{bmatrix}. \quad (7.6.4)$$

As was the case with the NDE, we may write equation (7.6.4) as $(\mathbf{L} - \mathbf{S})\boldsymbol{\psi} = \mathbf{q}$. The structure of equation (7.6.4) is very similar to that of the NDE presented in equation (6.6.3). To solve equation (7.6.4) iteratively we use source iteration (SI), which we already introduced in section 6.6.3. However, we shall recapitulate it here and explain some of the differences which arise when performing SI to solve the NTE as opposed to the NDE. First, we let $\boldsymbol{\psi}_g^{(i)}$ denote the i 'th iteration of the group g angular fluxes. Then the SI proceeds by initialising $\boldsymbol{\psi}_g^{(i)} = \mathbf{0}$ for $g = 1, \dots, G$ and performing the following iteration:

$$\boldsymbol{\psi}_g^{(i+1)} = \mathbf{L}_g^{-1} \left(\frac{1}{4\pi} \sum_{g' < g} \mathbf{S}_{g,g'} \boldsymbol{\psi}_{g'}^{(i+1)} + \frac{1}{4\pi} \sum_{g' \geq g} \mathbf{S}_{g,g'} \boldsymbol{\psi}_{g'}^{(i)} + \frac{1}{4\pi} \mathbf{q}_g \right) \quad (7.6.5)$$

For each (g, k) pair, the iteration of equation (7.6.5) is given by:

$$\boldsymbol{\psi}_{g,k}^{(i+1)} = \mathbf{L}_{g,k}^{-1} \left(\frac{1}{4\pi} \sum_{g' < g} \sum_{k'=1}^K \mathbf{S}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'}^{(i+1)} + \frac{1}{4\pi} \sum_{g' \geq g} \sum_{k'=1}^K \mathbf{S}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'}^{(i)} + \frac{1}{4\pi} \mathbf{q}_{g,k} \right) \quad (7.6.6)$$

The inversion of the (g, k) removal operator $\mathbf{L}_{g,k}$ is accomplished using a transport sweep. We say that equation (7.6.5) has converged if the following convergence criterion is met:

$$\max_g \delta_g < \epsilon \quad \text{where} \quad \delta_g = \frac{\|\boldsymbol{\phi}_g^{(i+1)} - \boldsymbol{\phi}_g^{(i)}\|}{\|\boldsymbol{\phi}_g^{(i)}\|} \quad (7.6.7)$$

where the tolerance ϵ is problem-dependent, so we shall state its value for every problem we present in section 7.7.

We shall postpone a full description of the SI procedure to section 7.6.5 as we must first discuss the sweep task.

7.6.4 The k -eigenvalue Problem

Substituting each of the expressions of the global matrices into equation (7.5.5) yields the following equivalent matrix problem:

$$\sum_{g=1}^G \sum_{k=1}^K w_k \mathbf{v}_{g,k}^T \left(\mathbf{L}_{g,k} \boldsymbol{\psi}_{g,k} - \frac{1}{4\pi} \sum_{g'=1}^G \sum_{k'=1}^K \mathbf{S}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'} - \frac{\lambda^h}{4\pi} \sum_{g'=1}^G \sum_{k'=1}^K \mathbf{F}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'} \right) = 0 \quad (7.6.8)$$

which, following the steps of the previous section, may be written as:

$$\begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{L}_G \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \\ \vdots \\ \boldsymbol{\psi}_g \end{bmatrix} - \frac{1}{4\pi} \begin{bmatrix} \mathbf{S}_{1,1} & \mathbf{S}_{1,2} & \dots & \mathbf{S}_{1,G} \\ \mathbf{S}_{2,1} & \mathbf{S}_{2,2} & \dots & \mathbf{S}_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{G,1} & \mathbf{S}_{G,2} & \dots & \mathbf{S}_{G,G} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \\ \vdots \\ \boldsymbol{\psi}_g \end{bmatrix} = \frac{\lambda^h}{4\pi} \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} & \dots & \mathbf{F}_{1,G} \\ \mathbf{F}_{2,1} & \mathbf{F}_{2,2} & \dots & \mathbf{F}_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}_{G,1} & \mathbf{F}_{G,2} & \dots & \mathbf{F}_{G,G} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \\ \vdots \\ \boldsymbol{\psi}_g \end{bmatrix}. \quad (7.6.9)$$

As was the case with the NDE, we may write equation (7.6.9) as $(\mathbf{L} - \mathbf{S})\boldsymbol{\psi} = \lambda^h \mathbf{F}\boldsymbol{\psi}$ which may be written in the following equivalent standard eigenvalue problem:

$$(\mathbf{L} - \mathbf{S})^{-1} \mathbf{F} \boldsymbol{\psi} = \gamma^h \boldsymbol{\psi} \quad \text{where} \quad \gamma^h = 1/\lambda^h$$

To solve equation (7.6.9) for the largest eigenvalue $\gamma_1^h = K_{\text{eff}}$ and the fundamental mode $\boldsymbol{\psi}$ we use power iteration (PI). The PI begins with initialising $\boldsymbol{\psi}_g^{(0)} = \mathbf{1}$ for $g = 1, \dots, G$ and setting $K_{\text{eff}}^{(0)} = 1$ and then proceeds as follows:

$$\boldsymbol{\psi}_g^{(i+1)} = \mathbf{L}_g^{-1} \left(\frac{1}{4\pi} \sum_{g' < g} \mathbf{S}_{g,g'} \boldsymbol{\psi}_{g'}^{(i+1)} + \frac{1}{4\pi} \sum_{g' \geq g} \mathbf{S}_{g,g'} \boldsymbol{\psi}_{g'}^{(i)} + \frac{1}{4\pi K_{\text{eff}}^{(i)}} \sum_{g'=1}^G \mathbf{F}_{g,g'} \boldsymbol{\psi}_{g'}^{(i)} \right) \quad (7.6.10)$$

For each pair (g, k) the iteration of equation (7.6.10) is given by:

$$\boldsymbol{\psi}_{g,k}^{(i+1)} = \mathbf{L}_{g,k}^{-1} \left(\frac{1}{4\pi} \sum_{g' < g} \sum_{k'=1}^K \mathbf{S}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'}^{(i+1)} + \frac{1}{4\pi} \sum_{g' \geq g} \sum_{k'=1}^K \mathbf{S}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'}^{(i)} + \frac{1}{4\pi K_{\text{eff}}^{(i)}} \sum_{g'=1}^G \sum_{k'=1}^K \mathbf{F}_{g,k}^{g',k'} \boldsymbol{\psi}_{g',k'}^{(i)} \right) \quad (7.6.11)$$

Again, the inversion of $\mathbf{L}_{g,k}$ is accomplished using a transport sweep. After a full power iteration, where we have solved for each energy group in succession, we will have a new set of angular fluxes:

$$\{\boldsymbol{\psi}_1^{(i+1)}, \boldsymbol{\psi}_2^{(i+1)}, \dots, \boldsymbol{\psi}_G^{(i+1)}\}$$

from which we may compute the next iterate of K_{eff} :

$$K_{\text{eff}}^{(i+1)} = K_{\text{eff}}^{(i)} \frac{\int_V S_f^{(i+1)} dx}{\int_V S_f^{(i)} dx}, \quad \text{where} \quad S_f = \sum_{g=1}^G \sum_{k=1}^K \nu \Sigma_f^g(\mathbf{x}) w_k \boldsymbol{\psi}_{g,k}.$$

We refer the reader to equation (6.6.9) to see how the integrals of the fission source are computed. In addition to equation (7.6.7) we also monitor the change in $K_{\text{eff}}^{(i+1)} - K_{\text{eff}}^{(i)}$ between

iteration to determine convergence.

We shall postpone a full description of the PI procedure to section 7.6.5 as we must first discuss the sweep task.

7.6.5 The Sweep Task, Source Iteration and Power Iteration

The global sparse matrices representing the discretised removal, scatter and fission operators ($\mathbf{L}$, $\mathbf{S}$ and $\mathbf{F}$ respectively) each have dimensions $(G \cdot K \cdot N_{\text{gdof}}) \times (G \cdot K \cdot N_{\text{gdof}})$. Therefore, constructing and storing the full matrices would exhaust memory resources even for moderately-sized problems. For very large problems, with hundreds of energy groups, thousands of ordinates and millions of spatial cells, storing the global matrices would be completely infeasible. Fortunately, we do not have to assemble and store these global matrices. Because we are using transport sweeps to compute the action of $\mathbf{L}_{g,k}^{-1}$ on a vector, we may also construct the scatter and fission sources on the fly, element-by-element. Implementing such a matrix-free algorithm reduces to an exercise in book-keeping: ensuring that, when constructing the local transport problem on every element $F \in \mathcal{T}_h$, that the correct angular fluxes on F are retrieved. Constructing and solving the local problem for the group-ordinate-face triple (g, k, F) is what we will call the **sweep task**: the basic building block of the transport sweep. Following the discussion of sections 7.4.1 and 7.5, let us define the following local matrices on (g, k, F) :

$$\begin{aligned} \mathbf{L}_{g,k,F} &= -\mathbf{T}_F + \Sigma_{t,F}^g \mathbf{M}_F + c_{g,k,F} \mathbf{S}_F \\ \mathbf{N}_F^+ &= \sum_{e \in \partial F_+} \left\langle (\varphi_i)^+, (\varphi_j)^+ \right\rangle_e^k, \quad \mathbf{N}_F^- = \sum_{e \in \partial F_-} \left\langle (\varphi_i)^+, (\varphi_j)^- \right\rangle_e^k \end{aligned}$$

and we define the upwind angular flux vector $\psi_{g,k,F}^\uparrow$ as being equal to the DoFs of the outside trace for all $e \in \partial F_- / \partial V_R$ and equal to the DoFs of the reflected angular flux $\psi_{g,k'}^{(i)}$ for all $e \in \partial F_- \cap \partial V_R$, see figure 7.4 for an illustration of this. We stress that when enforcing reflected boundary conditions, the flux from the previous iteration is used. The lagging of angular fluxes when enforcing reflected boundary conditions will slow the rate of convergence but it greatly simplifies the implementation of the sweep solver as reflected fluxes may then be treated as sources. It also avoids some of the large cyclic dependencies which may arise in the presence of reflected boundary conditions. The sweep task for the fixed source problem solves the following local problem:

$$\begin{aligned} & \left(\mathbf{L}_{g,k,F} + \mathbf{N}_F^+ \right) \psi_{g,k,F}^{(i+1)} \\ &= \\ & \frac{1}{4\pi} \mathbf{M}_F \left(\sum_{g' < g} \sum_{k'=1}^K \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \psi_{g',k',F}^{(i+1)} + \sum_{g' \geq g} \sum_{k'=1}^K \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \psi_{g',k',F}^{(i)} \right) + \frac{1}{4\pi} \mathbf{q}_{g,k,F} - \mathbf{N}_F^- \psi_{g,k,F}^\uparrow \end{aligned} \tag{7.6.12}$$

While the sweep task for the k -eigenvalue problem solves the following local problem:

$$\begin{aligned}
& \left(\mathbf{L}_{g,k,F} + \mathbf{N}_F^+ \right) \boldsymbol{\psi}_{g,k,F}^{(i+1)} \\
& = \\
& \frac{1}{4\pi} \mathbf{M}_F \left(\sum_{g' < g} \sum_{k'=1}^K \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \boldsymbol{\psi}_{g',k',F}^{(i+1)} + \sum_{g' \geq g} \sum_{k'=1}^K \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \boldsymbol{\psi}_{g',k',F}^{(i)} \right. \\
& \quad \left. + \frac{1}{K_{\text{eff}}^{(i)}} \sum_{g'=1}^G \sum_{k'=1}^K \chi_{g,F} \nu_{\Sigma_{f,F}^{g' \rightarrow g}} w_{k'} \boldsymbol{\psi}_{g',k',F}^{(i)} \right) - \mathbf{N}_F^- \boldsymbol{\psi}_{g,k,F}^\dagger
\end{aligned}$$

For both the fixed-source and k -eigenvalue problems, once $(\mathbf{L}_{g,k,F} + \mathbf{N}_F^+)$ has been inverted to yield the new iterate of the angular flux $\boldsymbol{\psi}_{g,k,F}^{(i+1)}$, it must then be stored in the global vector $\boldsymbol{\psi}^{(i+1)}$ so that it is accessible to downwind sweep tasks when they execute. In algorithm 27 we formally define the sweep task, which solves either equation (7.6.12) or (7.6.12) for every group-ordinate-face triple (g, k, F) and builds the new iterate of the global angular flux $\boldsymbol{\psi}^{(i+1)}$. Using the sweep task as the building block, we then define the transport sweep, also in algorithm 27, which for every group-ordinate pair (g, k) loops through all the faces $F \in \mathcal{T}_h$ in the sweep ordering of $\boldsymbol{\Omega}_k$ and applies the sweep task to each triple (g, k, F) . Algorithm 27 is what performs the action of the operator $\mathbf{L}_{g,k}^{-1}$ in equations (7.6.6) and (7.6.11). Using the sweep procedure described in this section we may now describe the SI and PI procedures for the multigroup S_N -NTE, which we give in algorithms 28 and 29 respectively.

Algorithm 27 The algorithm for the transport sweep for (g, k) the sweep task for triple (g, k, F) .

```

1: procedure SWEEP( $g, k$ )
2:   Inputs: Energy group  $g$  and direction  $k$ .
3:   for  $F \in \mathcal{T}_h$  in the sweep ordering with respect to  $\Omega_k$  do
4:     SWEEP TASK( $g, k, F$ )
5:   end for
6: end procedure

```

```

7: procedure SWEEP TASK( $(g, k, F)$ )
8:   Inputs: Energy group  $g$ , direction  $k$  and mesh face  $F$ 
9:   Compute the local matrices:  $\mathbf{L}_{g,k,F}$ ,  $\mathbf{N}_F^+$ ,  $\mathbf{N}_F^-$ ,  $\mathbf{M}_F$ .
10:  Retrieve angular fluxes  $\psi_{g',k',F}^{(i+1)}$  and compute the down-scatter source:

```

$$\mathbf{q}_{g,k,F}^{scat,\downarrow} = \frac{1}{4\pi} \mathbf{M}_F \left(\sum_{g' < g} \sum_{k'=1}^K \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \psi_{g',k',F}^{(i+1)} \right)$$

```

11:  Retrieve angular fluxes  $\psi_{g',k',F}^{(i)}$  and compute the up-scatter source:

```

$$\mathbf{q}_{g,k,F}^{scat,\uparrow} = \frac{1}{4\pi} \mathbf{M}_F \left(\sum_{g' \geq g} \sum_{k'=1}^K \Sigma_{s,F}^{g' \rightarrow g} w_{k'} \psi_{g',k',F}^{(i)} \right)$$

```

12:  if solving fixed-source problem then
13:    Compute the local fixed-source:

```

$$\mathbf{q}_{g,k,F}^* = \frac{1}{4\pi} \mathbf{q}_{g,k,F}$$

```

14:  else if solving  $k$ -eigenvalue problem then
15:    Retrieve angular fluxes  $\psi_{g',k',F}^{(i)}$  and compute the local fission source:

```

$$\mathbf{q}_{g,k,F}^* = \frac{1}{4\pi K_{\text{eff}}^{(i)}} \mathbf{M}_F \left(\sum_{g'=1}^G \sum_{k'=1}^K \chi_{g,F} \nu \Sigma_{f,F}^{g' \rightarrow g} w_{k'} \psi_{g',k',F}^{(i)} \right)$$

```

16:  end if
17:  Retrieve the upwind angular flux vector  $\psi_{g,k,F}^\uparrow$  and compute upwind source:

```

$$\mathbf{q}_{g,k,F}^{up} = \mathbf{N}_F^- \psi_{g,k,F}^\uparrow$$

```

18:  Solve local problem:

```

$$(\mathbf{L}_{g,k,F} + \mathbf{N}_F^+) \psi_{g,k,F}^{(i+1)} = \mathbf{q}_{g,k,F}^{scat,\downarrow} + \mathbf{q}_{g,k,F}^{scat,\uparrow} + \mathbf{q}_{g,k,F}^* - \mathbf{q}_{g,k,F}^{up}$$

```

19:  Store  $\psi_{g,k,F}^{(i+1)}$  in global  $\psi^{(i+1)}$ .
20: end procedure

```

Algorithm 28 The SI procedure for the multigroup S_N-NTE.

- 1: Initialise $\psi_{g,k}^{(0)}$ to $\mathbf{0}$ for all $g = 1, \dots, G$ and $k = 1, \dots, K$.
- 2: Initialise ϕ_g^0 to $\mathbf{0}$ for all $g = 1, \dots, G$.
- 3: **for** $l = 1, \dots, \text{niter_max}$ **do**
- 4: **for** $g = 1, \dots, G$ **do**
- 5: **for** $k = 1, \dots, K$ **do**
- 6: Call SWEEP(g, k) to compute $\psi_{g,k}^{(l+1)}$.
- 7: **end for**
- 8: Compute new iterate of scalar flux:

$$\phi_g^{(l+1)} = \sum_{k=1}^K w_k \psi_{g,k}^{(l+1)}$$

- 9: **end for**
- 10: Check for convergence:

$$\max_g \frac{\|\phi_g^{(l+1)} - \phi_g^{(l)}\|}{\|\phi_g^{(l)}\|} < \epsilon$$

- 11: **end for**
-

Algorithm 29 The PI procedure for the multigroup S_N-NTE.

- 1: Initialise $\psi_{g,k}^{(0)}$ to **1** for all $g = 1, \dots, G$ and $k = 1, \dots, K$.
- 2: Initialise the scalar fluxes:

$$\phi_g^{(0)} = \sum_{k=1}^K w_k \psi_{g,k}^{(0)}$$

- 3: **for** $l = 1, \dots, \text{niter_max}$ **do**
- 4: **for** $g = 1, \dots, G$ **do**
- 5: **for** $k = 1, \dots, K$ **do**
- 6: Call SWEEP(g, k) to compute $\psi_{g,k}^{(l+1)}$.
- 7: **end for**
- 8: Compute new iterate of scalar flux:

$$\phi_g^{(l+1)} = \sum_{k=1}^K w_k \psi_{g,k}^{(l+1)}$$

- 9: **end for**
- 10: Compute new fission integral:

$$I_f^{(l+1)} = \int_V \left(\sum_{g=1}^G \nu \Sigma_f^g \phi_g^{(l+1)} dx \right)$$

- 11: Update the multiplication factor:

$$K_{\text{eff}}^{(i+1)} = K_{\text{eff}}^{(i)} \frac{I_f^{(i+1)}}{I_f^{(i)}}$$

- 12: Check for convergence:

$$\max_g \frac{\|\phi_g^{(l+1)} - \phi_g^{(l)}\|}{\|\phi_g^{(l)}\|} < \epsilon_\phi \quad \text{or} \quad |K_{\text{eff}}^{(l+1)} - K_{\text{eff}}^{(l)}| < \epsilon_k$$

- 13: **end for**
-

7.6.6 Parallelisation Strategy

For the multigroup S_N -NTE we implemented a shared-memory, task-based parallelisation strategy. For each group-ordinate pair (g, k) , the full set of tasks over all faces $F \in \mathcal{T}_h$ must be executed in the sweep ordering over $\mathcal{T}_h$ with respect to direction Ω_k . Because the source iteration and the power iteration solve from high energy to low energy, the tasks must also be executed in order of increasing g (decreasing in energy). Additionally, because we are using lagged angular fluxes in treating reflected boundary conditions, the angles may be executed independently of one another. Solving the multigroup, S_N -NTE over all triples (g, k, F) is therefore a task-scheduling problem where a strict ordering must be enforced on task execution as to ensure that information propagates from upstream to downstream for all Ω_k . Parallelism must therefore be found by executing tasks which are independent of one another concurrently. Within a shared-memory framework there are a few C++ libraries available which allow a user to specify executable tasks, establish an order between them and execute them in parallel wherever possible. Two examples are OpenMP [294] and Intel Threaded Building Blocks (TBB) [295]. However, we chose to use Taskflow [293] as it has, in our opinion, the most intuitive and natural API for specifying and executing task-graphs and exhibits better performance than both OpenMP and TBB in the test cases that they present in their paper. To demonstrate the task graphs constructed using Taskflow, consider the example of figure 7.13; where the task graphs for an S_4 level-symmetric angular quadrature are constructed for a mesh of the unit-square with 37 elements.

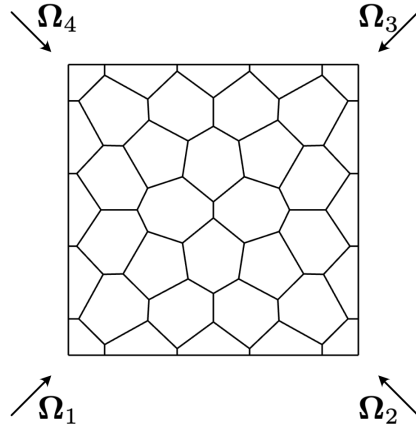


Figure 7.13: Example mesh with the 4 direction vectors in an S_4 level-symmetric angular quadrature.

For each direction $k = 1, 2, 3, 4$ we have a task tree (or DAG) for mesh $\mathcal{T}_h$ and the set of all task trees is what we shall term the **task forest**. The task forest is the same for every energy group g , therefore we construct just one task forest and pass g in as a variable to the tasks at runtime. As expected, each task tree is disjoint from the others; this is a consequence of each direction being independent from the others. Each node in the forest is a direction-face pair (k, F) and each directed edge $(k, F_1) \rightarrow (k, F_2)$ mean that for direction k , face F_1 must be solved before F_2 . In figure 7.14 we present the task forest for the mesh in figure 7.13.

This task-based parallel implementation is a natural fit for the implementation of sweep-based S_N codes. The fact that we may parallelise over both space and angle simultaneously using the same parallelisation strategy, rather than treating the parallelisation of space and angle separately (through domain decomposition in both for example) means that we may exploit the maximum amount of parallelism that is available in the problem. The task tree represents all of the computation done inside of the loop in line 5 in algorithms 28 and 29, therefore these are the parts of the SI and PI which the task-scheduler parallelises. Put differently, for a given energy group g , once all of the tasks in the task forest have been executed the set of angular fluxes:

$$\{\psi_{g,1}^{(l+1)}, \psi_{g,2}^{(l+1)}, \dots, \psi_{g,K}^{(l+1)}\}$$

will have been computed. A full description of the work-stealing scheduling algorithm implemented in Taskflow is outside the scope of this thesis so we refer the reader to [293] for more information.

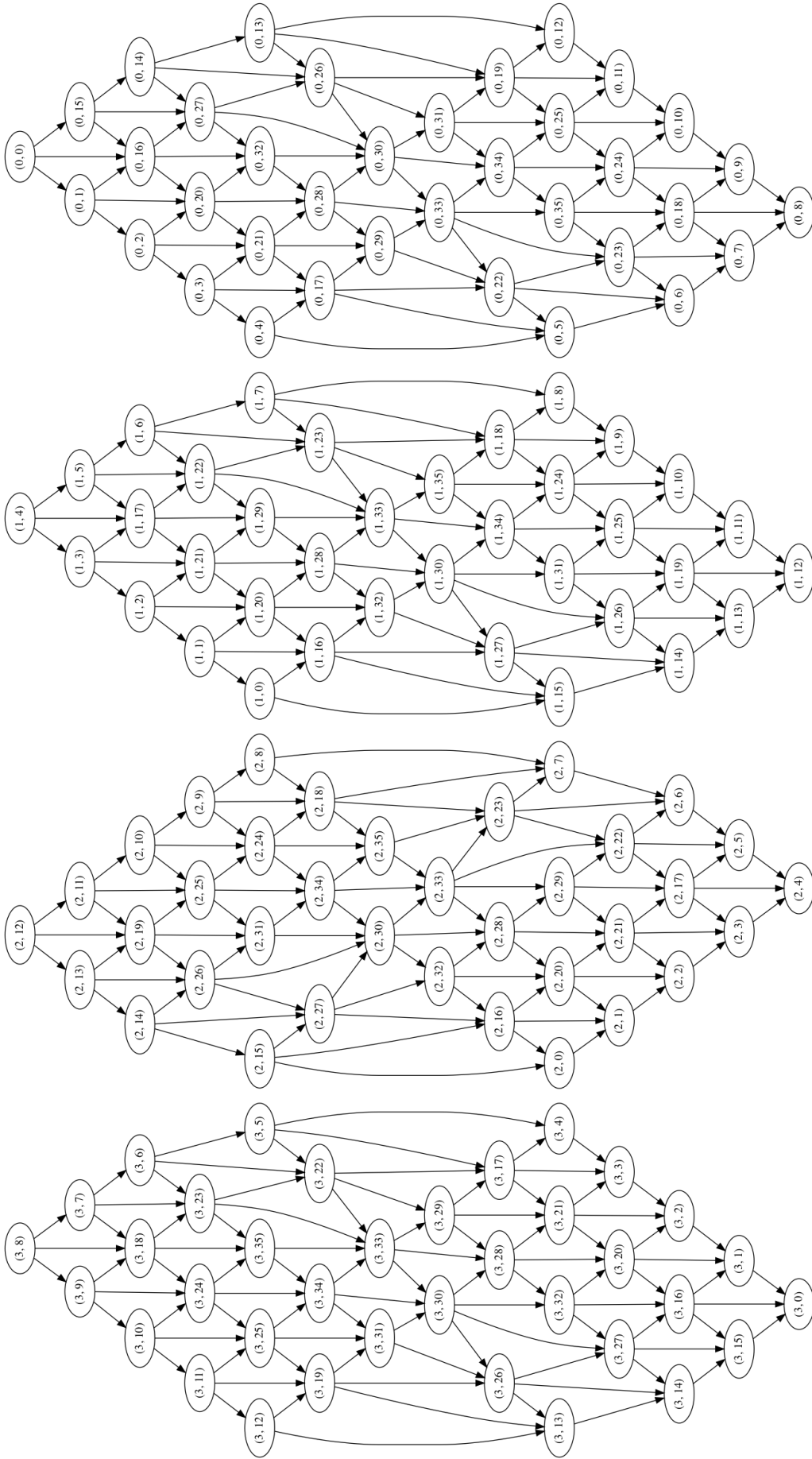


Figure 7.14: The task forest for the mesh in figure 7.13 with an S_4 level-symmetric quadrature.

7.7 Numerical Results

7.7.1 Method of Manufactured Solutions 1

The purpose of this test case is to verify that the DG-VEM converges with the expected order of accuracy for the advection-reaction equation:

$$\begin{aligned}\boldsymbol{\Omega} \cdot \nabla u + \Sigma(\mathbf{x})u &= f(\mathbf{x}) \quad \forall \mathbf{x} \in V \subset \mathbb{R}^2 \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \forall \mathbf{x} \in \partial V_-\end{aligned}\tag{7.7.1}$$

where we choose the domain to be the unit square $V = [0, 1]^2$. This is the simplest first-order hyperbolic equation and so is the logical starting point to investigate the properties of the DG-VEM discretisation without the added complexity of energy and angle dependence. Therefore we seek to solve equation (7.7.1) using the NEDG-VEM with $\boldsymbol{\Omega} = (100, -100)$ and $\Sigma = 1$. Letting $\mathbf{x} = (x, y)$, we choose the exact solution to be:

$$\hat{u}(\mathbf{x}) = (x - \cos(2y))(y + \sin(2x))$$

therefore the corresponding incoming boundary conditions and the forcing function are given by:

$$g(\mathbf{x}) = \hat{u}(\mathbf{x}), \quad f(\mathbf{x}) = \boldsymbol{\Omega} \cdot \nabla \hat{u}(\mathbf{x}) + \Sigma \hat{u}(\mathbf{x}).$$

A plot of the analytic solution $\hat{u}(\mathbf{x})$ may be found in figure 7.15. For this test case, we solved

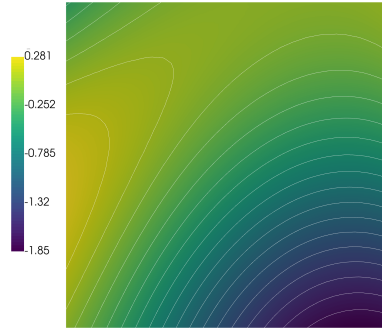


Figure 7.15: Plot of analytic solution with iso-lines shown in white.

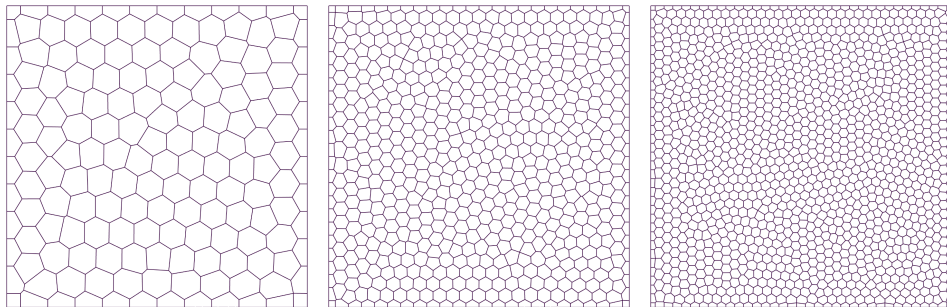


Figure 7.16: Three of the polygonal meshes used in this test case.

equation (7.7.1) using the discretisation we presented in section 7.3.3 and using the sweep solver

described in section 7.4 on a series of 12 increasingly refined polygonal meshes ranging from 166 faces to 1327 faces and for VEM order $p = 1, 2, 3, 4$. Letting h denote the average element diameter for each, then for each (h, p) pair we compute the $L^2(V)$ error:

$$E_{L^2}(u^h) = \left(\sum_{F \in \mathcal{T}_h} \|\hat{u} - \Pi_p^0 u^h\|_{0,F}^2 \right)^{\frac{1}{2}}$$

and the DG error:

$$E_{dg}(u^h) = \left[\left(\sum_{F \in \mathcal{T}_h} \|\hat{u} - \Pi_p^0 u^h\|_{0,F}^2 \right) + \left(\sum_{e \in \partial V} \frac{1}{2} \int_e |\mathbf{\Omega} \cdot \mathbf{n}| (\hat{u} - u^h)^2 ds \right) + \left(\sum_{e \in \mathcal{E}_i^h} \frac{1}{2} \int_e |\mathbf{\Omega} \cdot \mathbf{n}| [\![\hat{u} - u^h]\!]^2 ds \right) + \left(\sum_{F \in \mathcal{T}_h} \|\mathbf{\Omega} \cdot \nabla \hat{u} - \mathbf{\Omega} \cdot \Pi_p^0 \nabla u^h\|_{0,F}^2 \right) \right]^{1/2}$$

The analytic solution $\hat{u}(\mathbf{x})$ is infinitely differentiable, therefore the rate of convergence of $E_{dg}(u^h)$ we observe should be $\mathcal{O}(h^{p+1/2})$. In figure 7.17 we plot the convergence curves for the NEDG-VEM and indeed we do observe the expected rate of convergence while the rate of convergence in the L^2 norm appears to be slightly above $p + 1/2$.

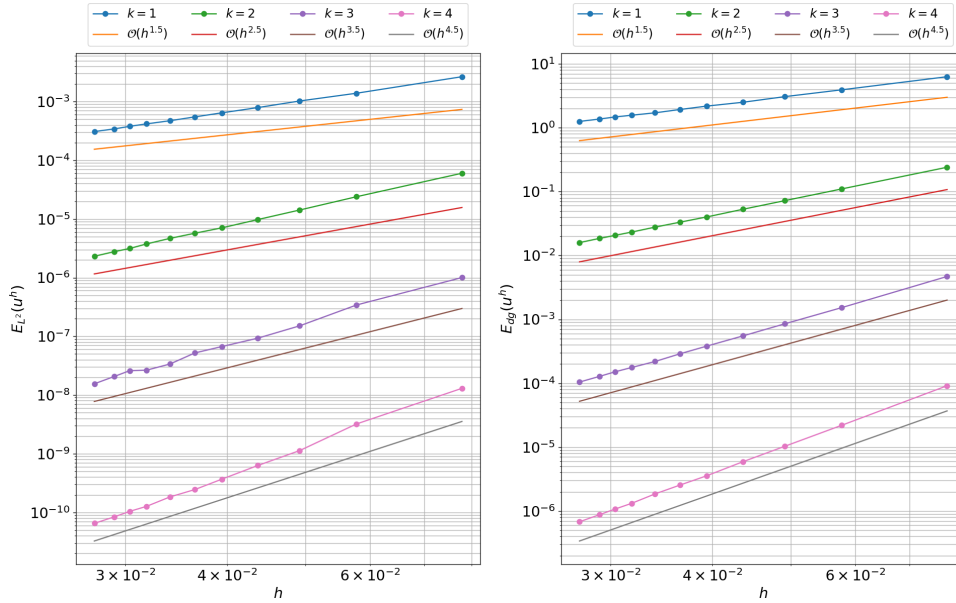


Figure 7.17: Convergence curves for the NEDG-VEM, the curves for $E_{L^2}(u^h)$ are on the left and the curves for $E_{dg}(u^h)$ are on the right.

7.7.2 Method of Manufactured Solutions 2

Having verified that our DG-VEM implementation converges with the expected order of accuracy for the advection-reaction equation, which is an S_N problem with $G = K = 1$, we must now verify that the full S_N solver converges with the expected order of accuracy. In this test case we are solving a two-group problem, with an S_4 approximation in angle. Therefore the governing equations are given by:

$$\begin{aligned}\boldsymbol{\Omega}_k \cdot \nabla \psi_{g,k}(\mathbf{x}) + \Sigma_t^g(\mathbf{x}) \psi_{g,k}(\mathbf{x}) &= \frac{1}{4\pi} \sum_{g'=1}^G \Sigma_s^{g' \rightarrow g}(\mathbf{x}) \phi_{g'} + \frac{1}{4\pi} q_{g,k}(\mathbf{x}) \quad \forall \mathbf{x} \in V \\ \phi_g(\mathbf{x}) &= \sum_{k=1}^K w_k \psi_{g,k}(\mathbf{x}) \quad \forall \mathbf{x} \in V \\ \psi_{g,k}(\mathbf{x}) &= f_{g,k}(\mathbf{x}) \quad \forall \mathbf{x} \in \partial V_-, \end{aligned} \quad (7.7.2)$$

where $G = 2$ and $K = 12$ and $f_{g,k}(\mathbf{x})$ is a boundary source function. Letting $(\mathbf{x}, \boldsymbol{\Omega}) = (x, y, \mu, \eta)$ denote the state-space vector, then the analytic angular flux solutions we choose for this problem are:

$$\hat{\psi}_1(\mathbf{x}, \boldsymbol{\Omega}) = \hat{\psi}_2(\mathbf{x}, \boldsymbol{\Omega}) = \hat{\psi}(\mathbf{x}, \boldsymbol{\Omega}) = X(\mathbf{x})A(\boldsymbol{\Omega}),$$

where

$$X(\mathbf{x}) = (x + \cos(y))(y - \sin(x)), \quad A(\boldsymbol{\Omega}) = 1 + \mu^2 + \eta^2$$

and the analytic scalar fluxes are given by:

$$\hat{\phi}_1(\mathbf{x}) = \hat{\phi}_2(\mathbf{x}) = \hat{\phi}(\mathbf{x}) = \left(\int_{4\pi} A(\boldsymbol{\Omega}) d\boldsymbol{\Omega} \right) X(\mathbf{x}) = \frac{20\pi}{3} X(\mathbf{x}).$$

In this test case we shall make use of the same curved pincell geometry we used in section 6.7.1 and the same curved, polygonal meshes. However, we only use one material for both sub-domains. The neutron cross section data for this material is given in table 7.2.

Table 7.2: Cross section data for this test case.

Σ_t^1	Σ_t^2	$\Sigma_s^{1 \rightarrow 1}$	$\Sigma_s^{1 \rightarrow 2}$	$\Sigma_s^{2 \rightarrow 1}$	$\Sigma_s^{2 \rightarrow 2}$
1.6	2.15	0.5	0.25	0.1	0.5

This is a fixed-source problem, therefore the fission cross sections are assumed to be zero. For each energy-ordinate pair (g, k) , the boundary condition source and the anisotropic, fixed-source are given by:

$$\begin{aligned}f_{g,k}(\mathbf{x}) &= \hat{\psi}(\mathbf{x}, \boldsymbol{\Omega}_k), \\ q_{g,k}(\mathbf{x}) &= 4\pi \left[\boldsymbol{\Omega}_k \cdot \nabla \hat{\psi}(\mathbf{x}, \boldsymbol{\Omega}_k) + \Sigma_t^g \hat{\psi}(\mathbf{x}, \boldsymbol{\Omega}_k) \right] - \sum_{g'=1}^G \Sigma_s^{g' \rightarrow g} \hat{\phi}(\mathbf{x})\end{aligned}$$

In this test case we shall compute the following DG error in the scalar fluxes.

$$E_{dg}(\phi_g^h) = \left[\left(\sum_{F \in \mathcal{T}_h} \|\hat{\phi} - \phi_g^h\|_{0,F}^2 \right) + \left(\sum_{e \in \partial V} \frac{1}{2} \int_e (\hat{\phi} - \phi_g^h)^2 ds \right) + \left(\sum_{e \in \mathcal{E}_i^h} \frac{1}{2} \int_e [\![\hat{\phi} - \phi_g^h]\!]^2 ds \right) \right]^{1/2} \quad (7.7.3)$$

for $g = 1, 2$. The convergence tolerance we chose for this was $\epsilon = 1 \times 10^{-10}$. In this test case we evaluate equation (7.7.2) on 9 meshes ranging from 400 elements to 1100 elements and for VEM orders $p = 1, 2, 3, 4$. For every diameter-order pair (h, p) we compute the DG error for ϕ_1^h and ϕ_2^h . These are shown in figure 7.18. The scalar flux is infinitely differentiable and the S_4 quadrature set integrates $A(\Omega)$ to within machine precision, therefore we again expect the maximum attainable order of convergence of $\mathcal{O}(h^{p+1/2})$ and indeed this is what we observe in figure 7.18

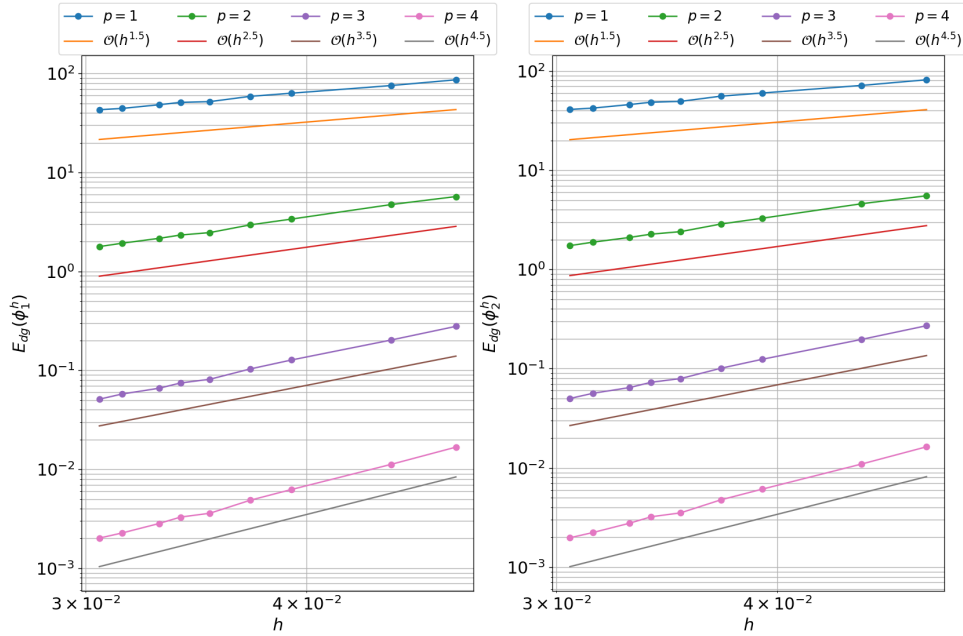


Figure 7.18: The DG errors in the two scalar fluxes: ϕ_1^h on the left and ϕ_2^h on the right.

The convergence curves presented in 7.18 are for the case where the exact-geometry VEM has been enabled. Therefore there is no geometric error slowing the convergence.

7.7.3 Reed Cell Problem

In this section we present the Reed cell problem. First presented in [296], this is a 1D mono-energetic, fixed-source S_8 -NTE problem with isotropic scatter defined on the domain $V = (0, 8)$. The geometry and nuclear data for the Reed cell problem are given in table 7.3. In [5] the

Table 7.3: Geometry and nuclear data for the Reed cell problem.

Material	domain	Σ_t	Σ_s	$q(\mathbf{x})$
1: Fuel	0cm - 2cm	50.0	0	50.0
2: Can	2cm - 3cm	5.0	0	0
3: Void	3cm - 5cm	0	0	0
4: Moderator	5cm - 6cm	1.0	0.9	1.0
5: Moderator	6cm - 8cm	1.0	0.9	0

analytic solution to the Reed cell problem was derived using the computer algebra package MAPLE and is shown in figure 7.19. We refer the reader the appendix of [5] for the closed-form solution to Reed cell problem.

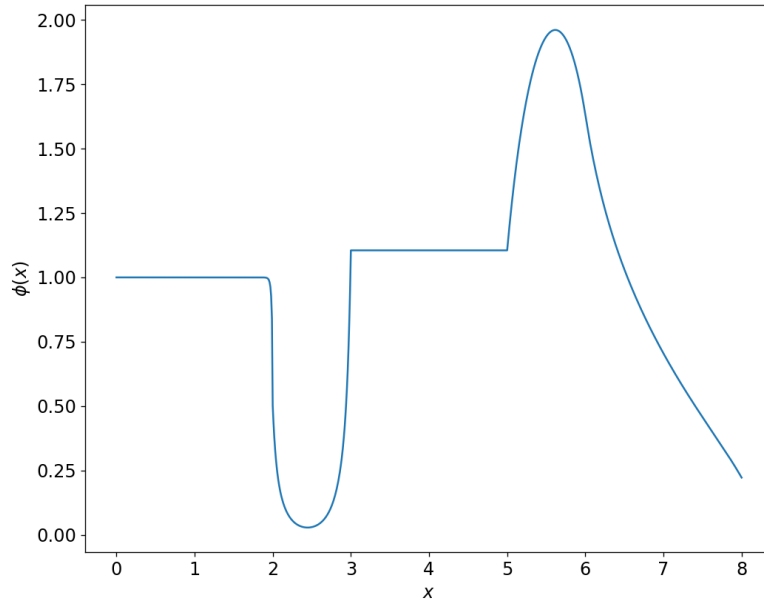


Figure 7.19: Analytic solution for the Reed cell problem from [5].

In this test case we adapt the Reed cell problem to be a 2D benchmark by using the geometry and boundary conditions presented in figure 7.20. Because we have prescribed reflective boundary conditions on the north and south sides, which are both parallel with the x -axis, the partial derivatives in y will be zero, so the 2D S_N equations reduce to the 1D S_N equations. We solve the Reed cell problem on a series of 11 meshes ranging from 1000 to 2500 elements and for VEM orders $p = 1, 2, 3, 4$. For every diameter-order pair (h, p) we compute the DG error (equation (7.7.3)) between the analytic solution shown in figure 7.19 and the discrete scalar flux ϕ^h . We present the convergence curves for the DG error in figure 7.22. Because the analytic solution to the Reed cell problem has discontinuities in the first derivative of $\phi(x)$ at $x = 3$

and $x = 5$, expect the maximum rate of convergence to be approximately $\mathcal{O}(h^{1.5})$ irrespective of the order of the VEM order p . Indeed this is what we observe in figure 7.22.

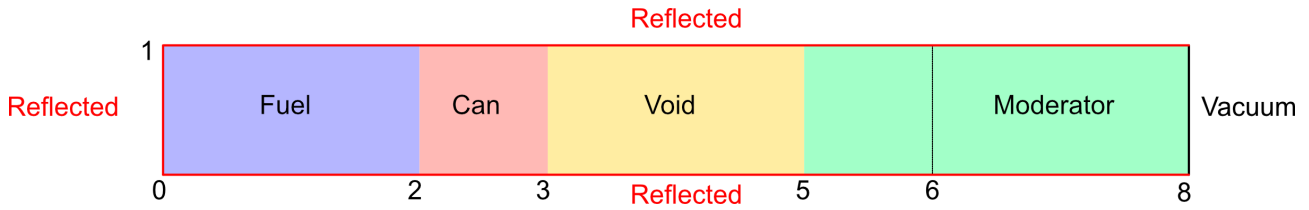


Figure 7.20: Three of the meshes used to solve the Reed cell problem.

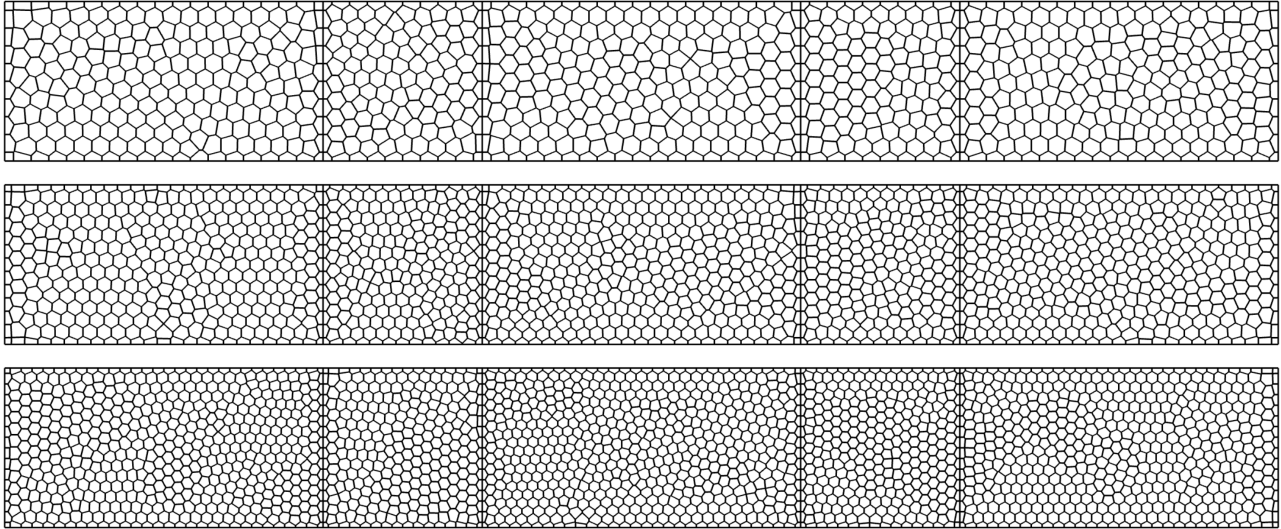


Figure 7.21: Three of the meshes used to solve the Reed cell problem.

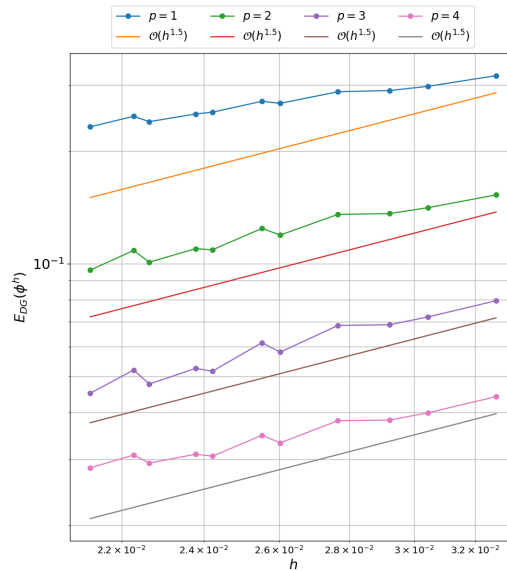


Figure 7.22: Convergence curves in the DG norm for the Reed cell problem.

7.7.4 The C5G7 UO₂ Pincell Problem

In this section we present a pincell problem taken from the C5G7 benchmark. The geometry of the problem is given in figure 7.23

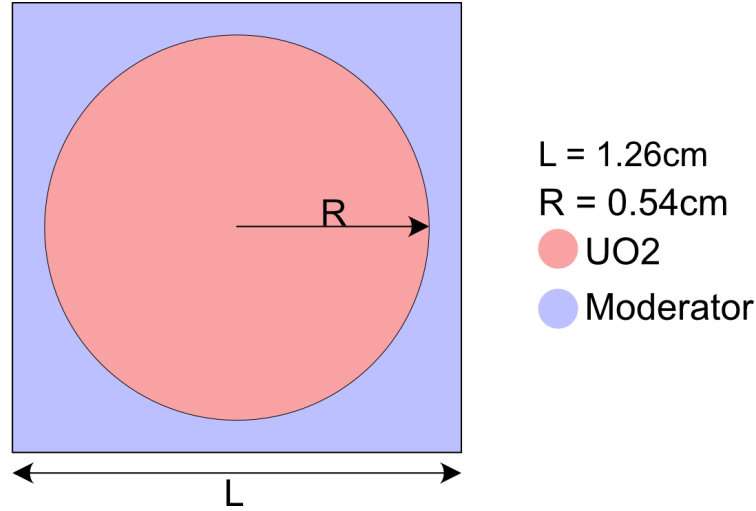


Figure 7.23: The geometry of the UO₂ pincell test case.

and the material properties are given in the appendix, section E.8. This is a 7 group problem and was solved using an S_{16} level-symmetric quadrature and a series of 12 polygonal meshes ranging from 800 to 2800 elements and with VEM orders $p = 1, 2, 3$.

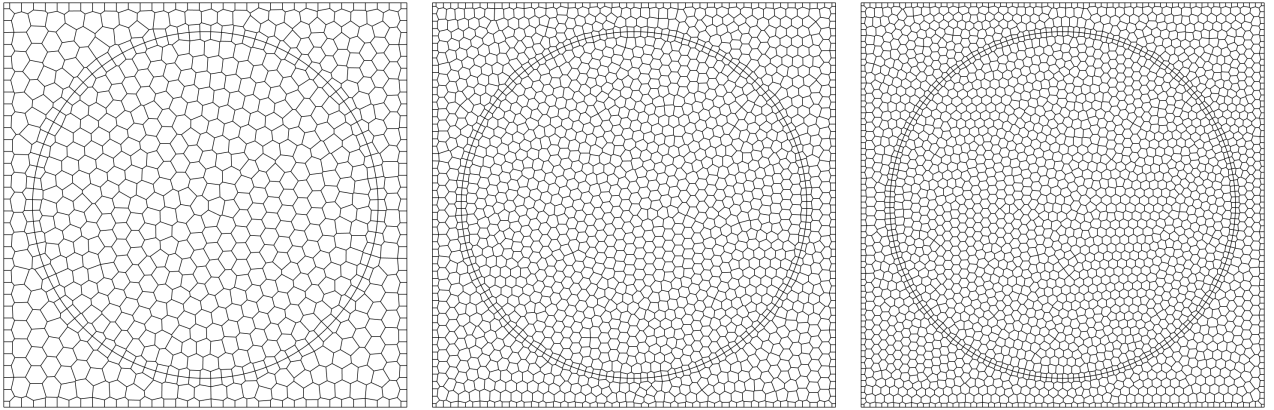


Figure 7.24: Three of the meshes used to solve the UO₂ pincell problem.

A reference solution was calculated using an S_{16} level-symmetric quadrature, a polygonal mesh with 6116 elements and a VEM order of $p = 3$. The quantities of interest in this benchmark are the K_{eff} , the integrals of the group 1, 4 and 7 scalar flux over the fuel (I_1^f , I_4^f , I_7^f) and the integrals of the scalar flux over the moderator (I_1^m , I_4^m , I_7^m). We provide the reference values of these QoIs in table 7.4.

Table 7.4: Reference values of the QoIs for and S_{16} , $p = 3$ and 6116 elements.

QoI	K_{eff}	I_1^f	I_4^f	I_7^f	I_1^m	I_4^m	I_7^m
Value	1.32366e+00	8.19153e+00	2.87762e+00	2.68911e+00	5.90458e+00	2.20023e+00	2.30451e+00

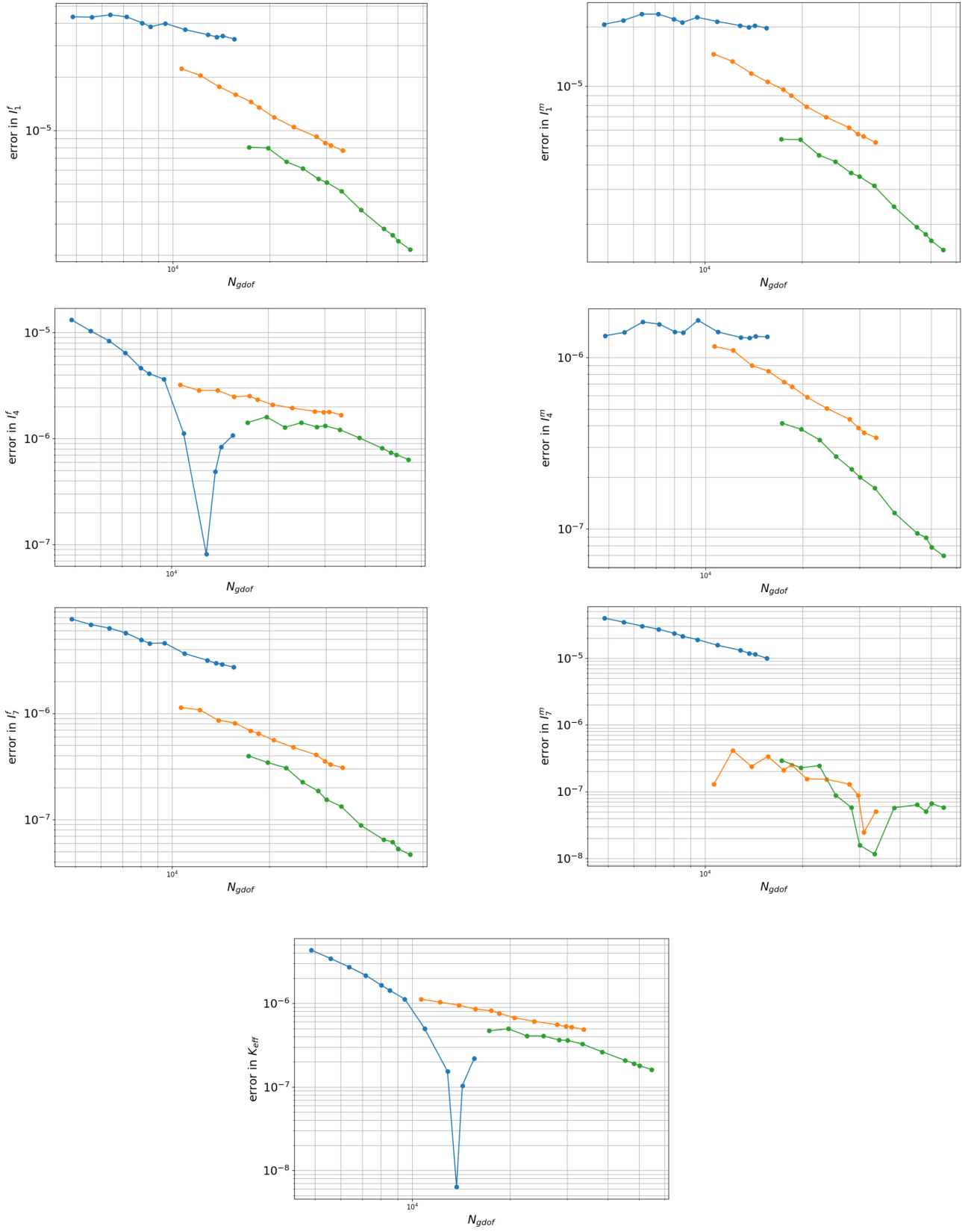


Figure 7.25: Convergence curves for each QoI given in table 7.4. QoIs I_1^f , I_4^f , I_7^f are on the left, I_1^f , I_4^f , I_7^f are on the right and K_{eff} is shown at the bottom. In all figures, the $p = 1$ curve is shown in blue, the $p = 2$ curve in orange and the $p = 3$ curve shown in green.

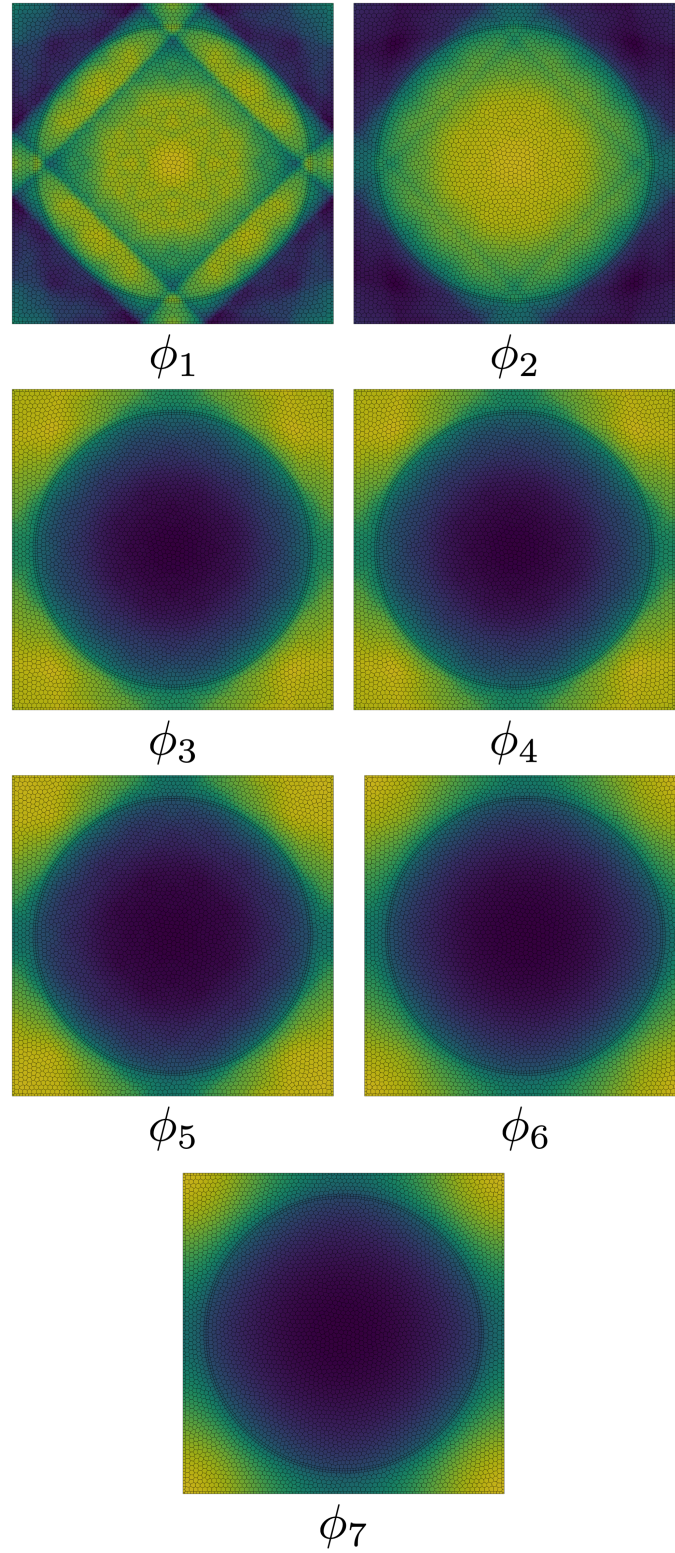


Figure 7.26: Scalar flux maps for the reference VEM solution with $p = 3$ and mesh with 6116 elements. Note the appearance of ray effects.

As would be expected for S_N methods, the solution exhibits strong ray effects for the fast scalar fluxes ϕ_1 and ϕ_2 . At these energy ranges, the fission source is at its strongest and given that reflected boundary conditions have been prescribed, the oscillations are reflected back into the domain. Meanwhile, in the thermal range (ϕ_5, ϕ_6, ϕ_7) the prompt fission multigroup spectrum $\chi_g = 0$ for $g = 5, 6, 7$, therefore there is no fission source and so there are no ray effects. As is common in k -eigenvalue problems, the different QoIs exhibit different convergence properties. This could be because different energy groups and regions of the domain converge at different rates therefore some QoIs may be under-converged and will therefore exhibit more numerical noise. The QoIs I_1^f , I_1^m , I_4^m and I_7^f all exhibit good convergence curves, where the curve for each order is both shifted downward and has a higher order of convergence. The anomalous $k = 1$ curves for I_4^f and the K_{eff} are probably due to cancellation of errors, this is usually the cause of anomalously low errors in a low-order discretisation.

7.8 Discussion and Future Work

In this chapter we have presented the application of the NEDG-VEM to the steady-state, multi-group, discrete ordinates, neutron transport equation, both for fixed-source and k -eigenvalue problems. We derived the variational form of the multigroup S_N -NTE in the discontinuous-Galerkin framework and showed how a system of linear equations may be assembled from the variational problem. We presented a transport sweep solver, which inverts the discretised advection-reaction operator $(\Omega \cdot \nabla + \Sigma)$ in a matrix-free manner. We presented a method of breaking cycles in the sweep graph by solving mutually dependent elements simultaneously, albeit under the heuristic that SCCs in the digraph are never larger than three elements. We presented a method of distributing NEDG-VEM degrees of freedom on the polygonal mesh and we presented a shared-memory, task-based parallel implementation of the S_N -NTE sweep solver. Finally, we presented four numerical test cases which demonstrate the application of the NEDG-VEM to multigroup, S_N -NTE problems. First, to verify our implementation of the NEDG-VEM, we presented a simple advection-reaction problem on a unit-square mesh. This problem is equivalent to solving the S_N -NTE for a single group-ordinate pair (g, k) . This test case exhibited the expected order of accuracy in the DG norm, specifically that for a NEDG-VEM scheme of order p the scheme converged with order $\mathcal{O}(h^{p+1/2})$ for a sufficiently regular solution. Second, to verify the source iteration procedure in addition to the NEDG-VEM discretisation, we presented a two-group, S_4 problem on a curved, polygonal mesh. The scalar flux solution of this problem exhibits the expected order of accuracy of $\mathcal{O}(h^{p+1/2})$ in the DG norm. From this, we may conclude two things: first that the NEDG-VEM discretisation was correctly approximating each angular flux $\psi_{g,k}$ and second, that the angular quadrature and source-iteration routines which compute the scalar flux are also performing as expected. We then presented an Reed cell problem, which is a mono-energetic S_8 problem which had been adapted to be a 2D problem. The analytics solution to this problem has low regularity therefore we do not expect $\mathcal{O}(h^{p+1/2})$ convergence but rather the best we can expect is $\mathcal{O}(h^{1.5})$ independent of NEDG-VEM order p . With that said, the higher order schemes still exhibit lower errors. Finally, we presented a C5G7 UO_2 pincell problem which had 7 energy groups and an S_{16} angular discretisation to demonstrate the NEDG-VEM applied to a k -eigenvalue problem. The convergence curves for most of the QoIs in this problem behaved as expected, with better convergence observed for higher-order VEM schemes.

A strength of the NEDG-VEM for the transport sweep over other methods such as IGA or high-order FEM, is that only a small subset of the edges in the mesh are curved. In IGA in particular, all of the IGA patches are curved and so large cycles involving many degrees of freedom can often arise. This observation informed the design of the cycle-breaking scheme presented in section 7.4, whereby clusters of 2-3 mutually dependent elements were solved simultaneously. Additionally, only a small subset of elements in the mesh require higher-order direct quadrature based on sub-tessellation (section 5.3.1) to compute the local projections, while the vast majority use Lassere’s method (section 5.3.2) which is much faster.

As was the case with the NECG-VEM, the primary limitation in the NEDG-VEM we have presented in this chapter is the lack of adaptive mesh refinement (AMR). Given that mesh flexibility is a key strength of the VEM, AMR would in theory be easier and more robust to implement than in methods with more stringent requirements on element shape. We would likely aim to replicate in the NEDG-VEM framework some of the ideas presented in [167, 169, 170] for IGA and [149, 150] for DG-VEM. Namely the derivation of dual-weighted residuals for guiding AMR and the creation of transfer operators between energy-dependent meshes.

Another aspect of the S_N -NTE sweep solver presented in this chapter which could be improved would be to use the scalar flux as the working vector as opposed to the angular flux, as we currently do. Given the matrix-free implementation of S_N sweep solvers, it is not necessary to store the full angular flux between iterations of an SI or PI procedure. This is the approach taken in [297, 298, 299, 300], where only those angular fluxes which are lagged, such as on reflected boundaries, are stored between iterations. This idea, called **significant angular fluxes**, [300], greatly reduces memory usage for problems using high-resolution angular quadrature. The memory savings become even more dramatic when using the GMRES iteration as opposed to the source iteration [301], as many previous iterations of the working vector must be stored to build the Krylov subspace.

It is well known that source iteration converges very slowly for highly scattering problems [302]. This property is independent of the spatial discretisation used and is an intrinsic property of SI itself. Therefore, diffusion acceleration for the NEDG-VEM will be crucial if the VEM is to be competitive with other spatial discretisations for which DSA has been developed. The DSA involves using the solution to a low-order (in angle) diffusion equation to accelerate the convergence of a high-order S_N equation. Optimal performance is achieved when both the low-order and high-order problems are solved using the same discretion. This ensures that both solutions are consistent [303]. To this end, the development of the symmetric interior penalty (SIP) VEM is a priority.

Finally, to achieve full scalability the shared-memory parallelisation strategy presented in this chapter will need to be replaced with distributed memory parallelisation. Many of the strategies for distributed-memory S_N sweeping on unstructured finite element meshes (see for example [304, 305, 306, 307]) would likely carry over to unstructured polygonal/polyhedral meshes quite naturally.

Chapter 8

Conclusions and Further Work

8.1 Conclusions

In chapter 2 we reviewed the background theory to the research presented in this thesis. We introduced the steady-state neutron transport equation in its most general form and made the simplifying assumption of isotropic materials: where the scatter cross section depends only on the change between incoming and outgoing angles, not the incoming angle. We then introduced the two types of problem this PhD has solved using the VEM: the fixed-source problem and the k -eigenvalue problem. We then briefly reviewed some common discretisations in angle, with emphasis placed on the angular discretisation which was implemented as part of this PhD: the S_N method. We then reviewed the multigroup approximation in angle and three kinds of spatial discretisation, the finite difference method, the finite element method and isogeometric analysis, which have all been applied extensively to problems in nuclear reactor physics.

In chapter 3 we provide a brief review of the ideas and definitions from functional analysis upon which both finite element methods and virtual element methods are built. This chapter is for reference.

In chapter 4 we presented the 2D, H^1 -conforming, NURBS-enhanced local virtual element space and presented its key features, including the set of local degrees of freedom, the interpolation estimates, the polynomial projections and a guide to computing the local bilinear forms. The local H^1 -conforming NE-VEM space is the local space used for both the NECG-VEM presented in chapter 6 to solve the NDE, and the NEDG-VEM presented in chapter 7 to solve the NTE. The difference between the NECG-VEM and NEDG-VEM is the implicit regularity of the global VEM space. The NECG-VEM assumes global H^1 regularity while the NEDG-VEM space assume global L^2 regularity.

In chapter 5 we described the 2D geometric modelling kernel (GMK) developed as part of this PhD. A description of topology based upon the double-connected edge list (DCEL) was implemented which can represent any 2-manifold topology. The geometric description of curves and surfaces in the GMK was the non-uniform rational B-spline (NURBS). NURBS are the

standard geometry encoding in the computer-aided geometric design industry. We also implemented a polygonal mesh generator which maintains the the geometry of the input domain exactly. In fact, both the mesh and the geometric domain are represented using exactly the same DCEL-plus-NURBS data structure. Conceptually, the mesh is simply a more refined geometric domain. This is in contrast to most finite element implementations where the domain and the mesh are logically separated and use different data structures.

In chapter 6 we present the application of the NECG-VEM to the multigroup NDE. We derive the discretised variational form of the NDE and show how to yield a system of equations from it. We provide a detailed account of the source iteration and power iteration procedures used to solve the fixed-source and k -eigenvalue problems respectively. We also present a distributed-memory parallelisation strategy based upon mesh partitioning and DoF renumbering to ensure that the distributed-memory matrix-vector multiply scales well. We demonstrate through numerical examples that for problems with curved domains, the NECG-VEM attains its optimal order of accuracy of $\mathcal{O}(h^{p+1})$ in the L^2 norms and $\mathcal{O}(h^p)$ in the H^1 norm while the order of accuracy of the conventional straight-sided VEM saturates at second-order in the L^2 norm and first order in the H^1 norm.

In chapter 7 we present the application of the NEDG-VEM to the multigroup, S_N -NTE. We derive the discretised variational form of the S_N -NTE and show how to yield a system of equations from it. A key difference between the S_N -NTE and the NDE is that the global equations are never assembled, rather they are solved using the transport sweep whereby the streaming + collision operator $\mathbf{\Omega} \cdot \nabla + \Sigma_t$ is inverted element-by-element. Similarly, the scatter and fission sources are computed element-by-element during the transport sweep. We adapted the standard algorithm for computing the sweep graph, which is a depth-first search and Kosaraju's algorithm, to the case of curved, polygonal meshes. We observed that cycles in the mesh, when they are present, usually consisted of two elements and never exceeded three for reasonable nuclear reactor geometries. Therefore, we proposed a method of breaking cycles which involved solving mutually dependent elements in a strongly-connected component simultaneously. We provide a detailed account of source iteration and power iteration procedures which used the transport sweep to invert each (g, k) global streaming + collision operator. We then presented a shared-memory parallelisation strategy based upon task-scheduling which parallelises in both angle and space simultaneously. Finally, we present a set of numerical examples which show that the NEDG-VEM converges with an expected order of accuracy of $\mathcal{O}(h^{p+1/2})$ in the DG norm.

To conclude, through a presentation of theory and numerical experiments we have shown the NECG-VEM and the NEDG-VEM to be viable alternatives to the finite element method and IGA for nuclear reactor physics problems. Using the VEM we may construct stable and high-order discretisations on very challenging domains while maintaining exact geometry from the lowest level of mesh refinement.

8.2 Future work

There are many fascinating possibilities for future research which lead directly from the research presented in this thesis. In the discussions of the chapters 5, 6 and 7, we already discussed some of the immediate improvements which could be made to work presented in each chapter: these included adaptive mesh refinement, error estimators, energy-dependent meshes, extension of both the NEG-VEM and NEDG-VEM to 3D, the development of a symmetric interior penalty NEDG-VEM for diffusion synthetic acceleration for the S_N -NTE and the development of a distributed memory parallelisation strategy for the S_N -NTE. In this section we shall discuss some longer-term future research.

Recently both p -multigrid and agglomeration based h -multigrid have been introduced in the VEM literature [308, 309] and multigrid algorithms for hp -DGFEM on polygonal meshes [310]. These multigrid schemes were devised for elliptic problems. Therefore, their primary application will be in preconditioning the neutron diffusion solver presented in chapter 6 and in preconditioning an NEDG-VEM, symmetric interior penalty solver which would be needed to perform diffusion synthetic acceleration. Good preconditioning of any NEDG-VEM elliptic solver used in DSA will be instrumental in ensuring that the diffusion solve in DSA does not dominate solve-times and eliminate the speed-up from faster convergence of the source iteration.

Angular virtual elements using a Voronoi tessellation of the sphere and subsequent adaptive refinement of the spherical surface mesh could form the basis of an interesting discretisation of angle. Similar idea have existed for finite elements for a long time [108, 109, 110, 111, 143], however the possibility of using exact-geometry, spherical 2D polygons could possibly confer additional accuracy over angular FEM.

The application of the NECG-VEM to the second-order forms of the NTE, such as the self-adjoint angular flux, weighted least-squares or even-parity would probably yield viable discretisation with all of the good properties which we have explored in this thesis. This would represent one of the most readily completed areas of future research.

The great strength of the VEM is its flexibility with regards to element shape and distortion. This makes it an excellent candidate to bridge the gap between different physics, where a mesh which is ideal from for one kind of physics could also be used for another without catastrophic loss of convergence. Therefore multiphysics applications, for example coupled neutronics and thermal-hydraulics could be an interesting area of research.

Appendix A

Appendix to Chapter 2

A.1 Definition of Legendre Polynomials and Real Spherical Harmonics

The Legendre polynomials $P_l(x) : \mathbb{R} \rightarrow \mathbb{R}$ are a set of orthogonal polynomials defined by the following 3-term recurrence relation:

$$\begin{aligned} P_0(x) &= 1, & P_1(x) &= x \\ P_{l+1} &= \left(\frac{2l+1}{l+1} \right) x P_l(x) + \left(\frac{l}{l+1} \right) P_{l-1}(x), \end{aligned}$$

and are L^2 -orthogonal on the range $[-1, 1]$:

$$\int_{-1}^1 P_l(x) P_m(x) dx = \frac{2}{2l+1} \delta_{l,m}$$

The real spherical harmonics $R_l^m(\boldsymbol{\Omega}) : S \rightarrow \mathbb{R}$ are distributions of the polar angle cosine $\mu = \cos(\theta)$ and the azimuthal angle ω defined as [103]:

$$R_l^m(\boldsymbol{\Omega}) = \sqrt{(2 - \delta_{m,0}) \frac{(l - |m|!)}{(l + |m|!)}} P_l^{|m|}(\mu) T_m(\omega)$$

where $P_l^{|m|}(\mu)$ are the associate Legendre functions defined as:

$$P_l^m(\mu) = (1 - \mu)^{m/2} \frac{d^m}{d\mu^m} P_l(\mu), \quad m \geq 0 \quad \text{and} \quad T_m(\omega) = \begin{cases} \cos(m\omega) & \text{if } m \geq 0 \\ \sin(|m|\omega) & \text{otherwise} \end{cases}.$$

Using this definition of the real spherical harmonics, where the Ferrer definition of the associated

Legendre function has been used, we may also express the direction vector as [103]:

$$\mathbf{\Omega} = \begin{bmatrix} \mu \\ \sqrt{1-\mu^2} \cos(\omega) \\ \sqrt{1-\mu^2} \sin(\omega) \end{bmatrix} = \begin{bmatrix} R_1^0 \\ R_1^1 \\ R_1^{-1} \end{bmatrix}.$$

The real spherical harmonics satisfy the following orthogonality condition:

$$\int_{4\pi} d\Omega R_l^m R_{l'}^{m'} = \frac{4\pi}{2l+1} \delta_{l,l'} \delta_{m,m'},$$

and are related to the spherical harmonics $Y_l^m(\mathbf{\Omega})$ as follows:

$$R_l^m(\mathbf{\Omega}) = \begin{cases} \sqrt{\frac{4\pi}{2l+1}} \text{Im}(Y_l^{|m|}(\mathbf{\Omega})) & \text{for } m < 0 \\ \sqrt{\frac{4\pi}{2l+1}} \text{Re}(Y_l^m(\mathbf{\Omega})) & \text{for } m \geq 0 \end{cases}.$$

The Legendre polynomials satisfy the **addition theorem**

$$P_l(\mathbf{\Omega} \cdot \mathbf{\Omega}') = \sum_{m=-l}^l R_l^m(\mathbf{\Omega}) R_l^m(\mathbf{\Omega}').$$

We shall also define the following mapping between 1D index i and the pair (l, m) for spherical harmonics.

$$\begin{aligned} 1 &\leftrightarrow (0, 0) \\ 2 &\leftrightarrow (1, -1), \quad 3 \leftrightarrow (1, 0), \quad 4 \leftrightarrow (1, 1) \\ 5 &\leftrightarrow (2, -2), \quad 6 \leftrightarrow (2, -1), \quad 7 \leftrightarrow (2, 0), \quad 8 \leftrightarrow (2, 1), \quad 9 \leftrightarrow (2, 2) \\ &\text{and so on...} \end{aligned} \tag{A.1.1}$$

Therefore, when we write $R_{l(i)}^{m(i)}$ we mean the spherical harmonic with indices (l, m) corresponding to 1D index i .

A.2 Derivation of Isotropic Scatter Source

Starting with the original scatter source:

$$Q_s(\mathbf{x}, E, \mathbf{\Omega}) = \int_0^\infty dE' \int_{4\pi} d\Omega' \Sigma_s(\mathbf{x}, E' \rightarrow E, \mathbf{\Omega}' \rightarrow \mathbf{\Omega}) \psi(\mathbf{x}, E', \mathbf{\Omega}'),$$

we begin by making the isotropic simplification to the differential scatter cross section:

$$Q_s(\mathbf{x}, E, \boldsymbol{\Omega}) = \frac{1}{2\pi} \int_0^\infty dE' \int_{4\pi} d\Omega' \Sigma_s(\mathbf{x}, E' \rightarrow E, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}) \psi(\mathbf{x}, E', \boldsymbol{\Omega}'). \quad (\text{A.2.1})$$

Next, we expand $\Sigma_s(\mathbf{x}, E' \rightarrow E, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega})$ in the Legendre polynomials and apply addition theorem to get:

$$\begin{aligned} \Sigma_s(\mathbf{x}, E' \rightarrow E, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}) &= \sum_{l=0}^L \left(\frac{2l+1}{2} \right) \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) P_l(\boldsymbol{\Omega} \cdot \boldsymbol{\Omega}') \\ &= \sum_{l=0}^L \left(\frac{2l+1}{2} \right) \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) R_l^m(\boldsymbol{\Omega}'). \end{aligned}$$

Substituting this real spherical harmonic expansion into equation (A.2.1) gives:

$$\begin{aligned} Q_s(\mathbf{x}, E, \boldsymbol{\Omega}) &= \int_0^\infty dE' \sum_{l=0}^L \left(\frac{2l+1}{4\pi} \right) \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) \int_{4\pi} d\Omega' R_l^m(\boldsymbol{\Omega}') \psi(\mathbf{x}, E', \boldsymbol{\Omega}') \\ &= \int_0^\infty dE' \sum_{l=0}^L \left(\frac{2l+1}{4\pi} \right) \Sigma_{s,l}(\mathbf{x}, E' \rightarrow E) \sum_{m=-l}^l R_l^m(\boldsymbol{\Omega}) \phi_l^m(\mathbf{x}, E') \end{aligned}$$

Appendix B

Appendix to Chapter 3

In this section we shall collect some theorems and inequalities, in no particular order, which are used throughout the thesis. We begin with a lemma for real numbers which will be reused elsewhere:

Lemma B.0.1. *For positive real numbers a, b, c, d the following inequality holds:*

$$ab + cd \leq (a^2 + c^2)^{1/2}(b^2 + d^2)^{1/2}$$

Proof. Let $A = ab + cd$ and let $B = (a^2 + c^2)^{1/2}(b^2 + d^2)^{1/2}$

$$\begin{aligned} B^2 - A^2 &= [(ab)^2 + (ad)^2 + (cb)^2 + (cd)^2] - [(ab)^2 + 2(ab)(cd) + (cd)^2] \\ &= (ad)^2 - 2(ad)(bc) + (bc)^2 = (ad - bc)^2 \geq 0 \end{aligned}$$

From which we conclude:

$$B^2 - A^2 \geq 0 \implies A^2 \leq B^2 \implies A \leq B$$

□

Theorem B.0.2. Cauchy-Schwarz Inequality for Sums *Let $\xi_i, \eta_i \in \mathbb{R}$ for $i = 1, \dots, n$ then the following inequality holds:*

$$\sum_{i=1}^n |\xi_i \eta_i| \leq \left(\sum_{i=1}^n \xi_i^2 \right)^{1/2} \left(\sum_{j=1}^n \eta_j^2 \right)^{1/2}$$

Proof. See [174], section 1.2.

□

Appendix C

Appendix to Chapter 4

C.1 Proof of Proposition 4.3.1

Proof. For each straight edge $e \subset \partial F$ the polynomial $q_p \in \mathbb{P}_p(e)$ is uniquely determined by the two vertex values and $p - 1$ values on the interior of the edge. On each curvilinear edge $\tilde{e} \subset \partial F$ a function $\tilde{q}_p \in \tilde{\mathbb{P}}_p(\tilde{e})$ is uniquely determined by the vertex values and the $p - 1$ interior values in $[0, 1]$ of $q_p(t)$ for which $\tilde{q}_p(\gamma_{\tilde{e}}t) = q_p(t)$. Therefore a function $g_p \in \mathcal{B}_p(\partial F)$ is uniquely determined by pN_e^F values. $\square$

C.2 Proof of Proposition 4.3.2

Proof. Let $g \in \mathcal{B}_p(\partial F)$ and $q_p \in \mathbb{P}_p(F)$ then there exists a unique $v \in \tilde{\mathcal{V}}_p(F)$ such that:

$$\begin{aligned}\Delta v &= q_p \quad \forall \mathbf{x} \in F \\ v &= g \quad \forall \mathbf{x} \in \partial F\end{aligned}$$

Therefore, $\dim(\tilde{\mathcal{V}}_p(F)) = \dim(\mathcal{B}_p(\partial F)) + \dim(\mathbb{P}_p(F)) = pN_e^F + n_p$. $\square$

C.3 Proof of Proposition 4.3.6

Proof. The dimension of $\bar{\mathbb{P}}_p(F)$ is $p+1$ so the dimension of $\bar{\mathbb{P}}_p(F) \cup \bar{\mathbb{P}}_{p-1}(F)$ is $2p+1$. Therefore, the minimum dimension that $\mathcal{V}_p(F)$ could be is $\dim(\tilde{\mathcal{V}}_p(F)) - (2p+1)$, this corresponds to the case where none of the additional $2p+1$ conditions are independent. To determine the independence of these $2p+1$ conditions consider a function $v \in \mathcal{V}_p(F)$ for which the linear functionals $\mathcal{D}_1, \dots, \mathcal{D}_4$ are all zero then, starting from equation (4.3.2) we have:

$$\begin{aligned}
(\nabla(\Pi_p^\nabla v, \nabla q_p))_{0,F} &= (\nabla v, \nabla q_p)_{0,F} \\
&= - \underbrace{\int_F \Delta q_p v \, dx}_A + \underbrace{\int_{\partial F} \nabla q_p \cdot \mathbf{n} v \, ds}_B = 0.
\end{aligned}$$

And the projection onto constants (equation (4.3.3)):

$$\int_{\partial F} \Pi_F^\nabla v \, ds = \underbrace{\int_{\partial F} v \, ds}_C = 0.$$

Term A is 0 as the internal DoFs are assumed zero and term B and C are both zero as $v|_{\partial F} = 0$. Therefore, $\Pi_F^\nabla v = 0$ and by the definition of $\mathcal{V}_p(F)$ this means the $p-1$ and p moments are also zero. Therefore the $p-1$ and p moments are not independent as they are fully determined by $\mathcal{D}_1, \dots, \mathcal{D}_4$. Consequently, $\dim(\mathcal{V}_p(F)) = pN_e^F + n_{p-2}$.

To prove the unisolvence of $\mathcal{D}_1, \dots, \mathcal{D}_4$ we must show that if $\mathcal{D}_1(v), \dots, \mathcal{D}_4(v) = 0$ this implies that $v = 0$ in F . Let's assume that for $v \in \mathcal{V}_p(F)$ that $\mathcal{D}_1(v), \dots, \mathcal{D}_4(v) = 0$, note that this implies $v|_{\partial F} = 0$. Taking the $H^1(F)$ seminorm of v we have:

$$|v|_{1,F}^2 = \int_F \nabla v \cdot \nabla v \, dx = - \underbrace{\int_F \Delta v \, v \, dx}_A + \underbrace{\int_{\partial F} \nabla v \cdot \mathbf{n} \, v \, ds}_B = 0$$

If $|v|_{1,F} = 0$ then v must be constant, given v is zero on ∂F this constant must be $v = 0$. Term A is zero $\Delta v \in \mathbb{P}_p(F)$ and as was shown earlier the moments of v up to order p are all zero. Term B is zero as $v|_{\partial F} = 0$.

□

C.4 Proof of lemma 6.3.2

Proof. We begin with the definition of the norm on $F \in \mathcal{T}_h$ and we assume that $h_F \leq h < 1$ (which can always be made true by rescaling the problem)

$$\begin{aligned}
\|u - u_\pi\|_{m,F}^2 &= \sum_{i=0}^m |u - u_\pi|_{i,F}^2 \\
&\leq \sum_{i=0}^m \left(C h_F^{s-i} |u|_{s,F} \right)^2 \\
&\leq C^2 m h_F^{2(s-m)} |u|_{s,F}^2 \quad (\text{Smallest power dominates as } h_F < 1).
\end{aligned}$$

Next, we shall bound the broken norm

$$\begin{aligned}
\|u - u_\pi\|_{m,h} &= \left(\sum_{F \in \mathcal{T}_h} \|u - u_\pi\|_{m,F}^2 \right)^{1/2} \\
&\leq \left(\sum_{F \in \mathcal{T}_h} C^2 m h_F^{2(s-m)} |u|_{s,F}^2 \right)^{1/2} \\
&\leq C_m h^{s-m} \left(\sum_{F \in \mathcal{T}_h} |u|_{s,F}^2 \right)^{1/2} \quad (\text{Letting } C_m = \sqrt{m}C) \\
&= C_m h^{s-m} |u|_{s,\Omega}.
\end{aligned}$$

□

C.5 Proof of Corollary 4.4.1.1

Proof. The proof relies on the fact that the L^2 projector satisfies the best approximation result

$$\|v - \Pi_p^0 v\|_{0,F} = \inf_{p \in \mathbb{P}_p(F)} \|v - p\|_{0,F} \leq \|v - v_\pi\|_{0,F} \leq C h_F^s |v|_{s,F}$$

To prove the estimate for the gradient projector we recall that if $u \in H^s(F)$ that $\partial u / \partial x_i \in H^{s-1}(F)$ and we perform the following steps:

$$\begin{aligned}
\|\nabla u - \Pi_p^0 \nabla u\|_{0,F}^2 &= \int_F [\nabla u - \Pi_p^0 \nabla u] \cdot [\nabla u - \Pi_p^0 \nabla u] \, dx = \sum_{i=1}^2 \int_F \left[\frac{\partial u}{\partial x_i} - \Pi_p^0 \left(\frac{\partial u}{\partial x_i} \right) \right]^2 dx \\
&= \sum_{i=1}^2 \left\| \frac{\partial u}{\partial x_i} - \Pi_p^0 \left(\frac{\partial u}{\partial x_i} \right) \right\|_{0,F}^2 \leq \sum_{i=1}^2 C^2 h_F^{2(s-1)} \left| \frac{\partial u}{\partial x_i} \right|_{s-1,F}^2 \\
&= C^2 h_F^{2(s-1)} \left(\sum_{i=1}^2 \left| \frac{\partial u}{\partial x_i} \right|_{s-1,F}^2 \right) = C^2 h_F^{2(s-1)} |\nabla u|_{s-1,F}^2
\end{aligned}$$

and the thesis follows easily. □

C.6 Proof of lemma 4.4.4

Proof. Beginning with the scalar-valued $L^2(V)$ projector we have:

$$\begin{aligned}
\|\Pi_p^0 v\|_{0,V}^2 &= (\Pi_p^0 v, \Pi_p^0 v)_{0,V} = (v, \Pi_p^0 v)_{0,V} \leq \|v\|_{0,V} \|\Pi_p^0 v\|_{0,V} \\
&\implies \|\Pi_p^0 v\|_{0,V} \leq \|v\|_{0,V}
\end{aligned}$$

Next, the for vector-valued $L^2(V)$ projector we have:

$$\begin{aligned}\left\|\Pi_{p-1}^0 \nabla v\right\|_{0,V}^2 &= \left(\Pi_{p-1}^0 \nabla v, \Pi_{p-1}^0 \nabla v\right)_{0,V} = \left(\nabla v, \Pi_{p-1}^0 \nabla v\right)_{0,V} \leq \|\nabla v\|_{0,V} \left\|\Pi_{p-1}^0 \nabla v\right\|_{0,V} \\ &\implies \left\|\Pi_{p-1}^0 \nabla v\right\|_{0,V} \leq \|\nabla v\|_{0,V}\end{aligned}$$

□

Appendix D

Appendix to Chapter 5

D.1 Proof of Lasserre on Edge

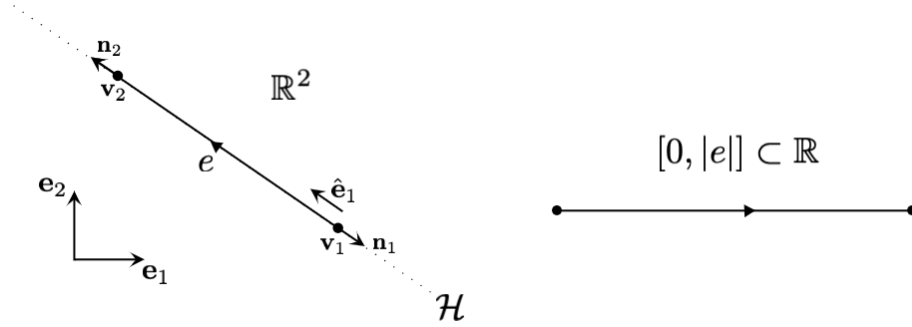


Figure D.1: Mapping from $\mathbb{R}^2$ to $\mathbb{R}$.

Proposition D.1.1. *Considering an edge $e \subset \mathcal{H} \subset \mathbb{R}^2$ such as that in figure D.1 then for homogenous function $g(\mathbf{x})$ of order q the following holds:*

$$\int_e g(\mathbf{x}) ds = \frac{1}{(1+q)} \left[|e| g(\mathbf{v}_2) + \int_e \mathbf{v}_1 \cdot \nabla g(\mathbf{x}) ds \right].$$

Proof. The proof given here is the restriction of the proof given for proposition 1 in [3] limited to the case where the domain of interest is an edge. Using the notation of figure D.1, we begin by defining a mapping $\gamma : \mathbb{R} \rightarrow \mathbb{R}^2$ from the coordinate system on the hyperplane $\tilde{\mathbf{x}} = (\tilde{x}_1)$ to $\mathbf{x} = (x_1, x_2)$:

$$\mathbf{x} = \gamma(\tilde{x}_1) = \mathbf{x}_0 + \tilde{x}_1 \hat{\mathbf{e}}_1.$$

The origin of $\mathcal{H}$ has arbitrarily been set $\mathbf{x}_0 = \mathbf{v}_1$. Let $\tilde{e} = [0, |e|] \subset \mathbb{R}$ then the following identity holds:

$$\int_e g(\mathbf{x}) ds = \int_{\tilde{e}} g(\gamma(\tilde{x}_1)) d\tilde{x}_1.$$

The gradient operator on the hyperplane is $\tilde{\nabla}g(\boldsymbol{\gamma}(\tilde{x}_1)) = (\partial g(\boldsymbol{\gamma}(\tilde{x}_1))/\partial \tilde{x}_1)$ and is related to the gradient in $\mathbb{R}^2$ by the chain rule:

$$\frac{\partial g}{\partial \tilde{x}_1} = \frac{\partial g}{\partial x_1} \frac{\partial x_1}{\partial \tilde{x}_1} + \frac{\partial g}{\partial x_2} \frac{\partial x_2}{\partial \tilde{x}_1} = \hat{\mathbf{e}}_1^T \nabla g. \quad (\text{D.1.1})$$

Given a function $f : \mathbb{R} \rightarrow \mathbb{R}$, the generalised Stokes' theorem in the hyperplane coordinate system is given by:

$$\underbrace{\int_{\tilde{e}} \frac{\partial f(\tilde{x}_1)}{\partial \tilde{x}_1} g(\boldsymbol{\gamma}(\tilde{x}_1)) d\tilde{x}_1}_A + \underbrace{\int_{\tilde{e}} f(\tilde{x}_1) \frac{\partial g(\boldsymbol{\gamma}(\tilde{x}_1))}{\partial \tilde{x}_1} d\tilde{x}_1}_B = -f(0)g(\mathbf{v}_1) + f(|e|)g(\mathbf{v}_2)$$

If we set $f = \tilde{x}_1$ then term A becomes:

$$\int_{\tilde{e}} \frac{\partial \tilde{x}_1}{\partial \tilde{x}_1} g(\boldsymbol{\gamma}(\tilde{x}_1)) d\tilde{x}_1 = \int_{\tilde{e}} g(\boldsymbol{\gamma}(\tilde{x}_1)) d\tilde{x}_1 = \int_e g(\mathbf{x}) ds$$

and term B becomes:

$$\begin{aligned} \int_{\tilde{e}} \tilde{x}_1 \frac{\partial g(\boldsymbol{\gamma}(\tilde{x}_1))}{\partial \tilde{x}_1} d\tilde{x}_1 &= \int_{\tilde{e}} \tilde{x}_1 \hat{\mathbf{e}}_1^T \nabla g(\boldsymbol{\gamma}(\tilde{x}_1)) d\tilde{x}_1 \\ &= \int_e (\mathbf{x} - \mathbf{x}_0) \cdot \nabla g(\mathbf{x}) ds \\ &= q \int_e g(\mathbf{x}) ds - \int_e \mathbf{x}_0 \cdot \nabla g(\mathbf{x}) ds \end{aligned}$$

Substituting all of this into equation (D.1.1) and noting that $f(0) = 0$, $f(|e|) = |e|$ and $\mathbf{x}_0 = \mathbf{v}_1$ we arrive at the desired result. $\square$

Appendix E

Appendix to Chapter 6

E.1 Proof of theorem 6.3.1

Proof. The well-posedness of equation (6.3.2) is proven by showing that $B(u, v)$ is an inner product on the space $H^1(V)$. Once this is established we may invoke Reisz representation theorem (theorem 3.6.2). To prove that $B(u, v)$ is an inner product it is sufficient to show it is **symmetric**, **continuous** and **coercive**. Additionally, we must prove $l(v)$ is continuous. The symmetry of $B(u, v)$ is obvious therefore we need only prove the other properties:

1. Continuity of $B(u, v)$:

$$\begin{aligned} B(u, v) &= (D(\mathbf{x})\nabla u, \nabla v)_{0,V} + (\Sigma(\mathbf{x})u, v)_{0,V} \\ &\leq D^*(\nabla u, \nabla v)_{0,E} + \Sigma^*(u, v)_{0,V} \\ &\leq D^* \|\nabla u\|_{0,V} \|\nabla v\|_{0,V} + \Sigma^* \|u\|_{0,E} \|v\|_{0,V} \quad \text{Schwarz Inequality (3.1.1)} \\ &\leq \max(D^*, \Sigma^*) \left(\|\nabla u\|_{0,V} \|\nabla v\|_{0,V} + \|u\|_{0,E} \|v\|_{0,V} \right) \quad (\text{E.1.1}) \\ &\leq \max(D^*, \Sigma^*) \left(\|u\|_{0,V}^2 + \|\nabla u\|_{0,V}^2 \right)^{1/2} \left(\|v\|_{0,V}^2 + \|\nabla v\|_{0,V}^2 \right)^{1/2} \quad \text{Lemma B.0.1} \\ &= \max(D^*, \Sigma^*) \|u\|_{1,V} \|v\|_{1,V}. \end{aligned}$$

Therefore, $B(u, v) \leq C_1 \|u\|_{1,V} \|v\|_{1,V}$.

2. Coercivity of $B(u, v)$:

$$\begin{aligned} B(u, u) &= (D(\mathbf{x})\nabla u, \nabla u)_{0,V} + (\Sigma(\mathbf{x})u, u)_{0,V} \\ &\geq D_*(\nabla u, \nabla u)_{0,E} + \Sigma_*(u, u)_{0,V} \\ &= D_* \|\nabla u\|_{0,V}^2 + \Sigma_* \|u\|_{0,V}^2 \quad (\text{E.1.2}) \\ &\geq \min(D_*, \Sigma_*) \left(\|u\|_{0,V}^2 + \|\nabla u\|_{0,V}^2 \right) \\ &= \min(D_*, \Sigma_*) \|u\|_{1,V}^2. \end{aligned}$$

Therefore, $B(u, v) \geq C_2 \|u\|_{1,V}^2$.

3. Continuity of $l(v)$:

$$\begin{aligned}
l(v) &= (f, v)_{0,V} + (r, v)_{0,\partial V} \\
&\leq \|f\|_{0,V} \|v\|_{0,V} + \|r\|_{0,\partial V} \|v\|_{0,\partial V} \quad \text{Schwarz inequality (3.1.1)} \\
&\leq \|f\|_{0,V} \|v\|_{1,V} + C \|r\|_{0,\partial V} \|v\|_{1,V} \quad \text{Trace inequality} \\
&= (\|f\|_{0,V} + C \|r\|_{0,\partial V}) \|v\|_{1,V}
\end{aligned}$$

Therefore, $l(v) \leq C_3 \|v\|_{1,V}$.

□

E.2 Proof of Lemma 6.3.4

Proof. Proof of lemma 6.3.4 The projections $\Pi_{p-1}^0 \nabla : \mathcal{V}_p(F) \rightarrow [\mathbb{P}_{p-1}(F)]^2$ and $\Pi_p^0 : \mathcal{V}_p(F) \rightarrow \mathbb{P}_p(F)$ satisfy:

$$\Pi_{p-1}^0 \nabla p = \nabla p, \quad \Pi_p^0 p = p \quad \forall p \in \mathbb{P}_p(F) \quad (\text{E.2.1})$$

We begin by noting that if either $u, v \in \mathbb{P}_p(F)$ that the stability bilinear form is zero: $S_F(u, v) = 0$. Next, the diffusion term:

$$\begin{aligned}
a_F^h(u, p) &= D_F(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla p)_{0,F} \\
&= D_F(\Pi_{p-1}^0 \nabla u, \nabla p)_{0,F} \quad (\text{Equation (E.2.1)}) \\
&= D_F(\nabla u, \nabla p)_{0,F} \quad \text{Equation (4.3.5)} \\
&= a_F(u, p)
\end{aligned}$$

and following exactly the same process for the reaction term:

$$\begin{aligned}
b_F^h(u, p) &= \Sigma_F(\Pi_p^0 u, \Pi_p^0 p)_{0,F} \\
&= \Sigma_F(\Pi_p^0 u, p)_{0,F} \quad (\text{Equation (E.2.1)}) \\
&= \Sigma_F(u, p)_{0,F} \quad \text{Equation (4.3.7)} \\
&= b_F(u, p).
\end{aligned}$$

The p -consistency of $B_F^h(\cdot, \cdot)$ follows from that of $a_F^h(\cdot, \cdot)$ and $b_F^h(\cdot, \cdot)$.

□

E.3 Proof of lemma 6.3.5

Proof. Proof of lemma 6.3.5 We begin the proof by considering the stability term $S_F((I - \Pi_p^\nabla)u, (I - \Pi_p^\nabla)v)$ where $u, v \in [\mathcal{V}_p(F) + \mathbb{P}_k(F)]$. Let $\hat{u} = u - \Pi_p^\nabla u \in \mathcal{V}_p(F)/\mathbb{P}_0(F)$, then:

$$c_0 T_F^\nabla(\hat{u}, \hat{u}) + d_0 T_F^0(\hat{u}, \hat{u}) \leq S_F(\hat{u}, \hat{u}) \leq c_1 T_F^\nabla(\hat{u}, \hat{u}) + d_1 T_F^0(\hat{u}, \hat{u}). \quad (\text{E.3.1})$$

Next, we have:

$$\begin{aligned} (I - \Pi_{p-1}^0) \nabla \hat{u} &= (I - \Pi_{p-1}^0) \nabla (u - \Pi_p^\nabla u) \\ &= \nabla u - \nabla \Pi_p^\nabla u - \Pi_{p-1}^0 \nabla u + \Pi_{p-1}^0 (\nabla \Pi_p^\nabla u) \\ &= \nabla u - \Pi_{p-1}^0 \nabla u \quad (\text{as } \nabla \Pi_p^\nabla u \in [\mathbb{P}_{p-1}(F)]^2) \\ &= (I - \Pi_{p-1}^0) \nabla u \end{aligned}$$

therefore $T_F^\nabla(\hat{u}, \hat{u}) = T_F^\nabla(u, u)$. Following the same procedure for L^2 projector:

$$\begin{aligned} (I - \Pi_p^0) \hat{u} &= (I - \Pi_p^0) (u - \Pi_p^\nabla u) \\ &= u - \nabla \Pi_p^\nabla u - \Pi_p^0 u + \Pi_p^0 (\Pi_p^\nabla u) \\ &= u - \Pi_p^0 u \quad (\text{as } \Pi_p^\nabla u \in \mathbb{P}_p(F)) \\ &= (I - \Pi_p^0) u \end{aligned}$$

therefore $T_F^0(\hat{u}, \hat{u}) = T_F^0(u, u)$. Substituting these results into equation (E.3.1) gives:

$$c_0 T_F^\nabla(u, u) + d_0 T_F^0(u, u) \leq S_F(u - \Pi_p^\nabla u, u - \Pi_p^\nabla u) \leq c_1 T_F^\nabla(u, u) + d_1 T_F^0(u, u)..$$

Beginning with the upper bound of $B_F^h(u, v)$ we have:

$$\begin{aligned} B_F^h(u, u) &\leq D_F(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla u)_{0,F} + \Sigma_F(\Pi_p^0 u, \Pi_p^0 u)_{0,F} + c_1 T_F^\nabla(u, u) + d_1 T_F^0(u, u) \\ &= \left[D_F(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla u)_{0,F} + c_1 D_F((I - \Pi_{p-1}^0) \nabla u, (I - \Pi_{p-1}^0) \nabla u)_{0,F} \right] \\ &\quad + \left[\Sigma_F(\Pi_p^0 u, \Pi_p^0 u)_{0,F} + d_1 \Sigma_F((I - \Pi_p^0) u, (I - \Pi_p^0) u)_{0,F} \right] \\ &\leq D_F \max(1, c_1) \left[(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla u)_{0,F} + ((I - \Pi_{p-1}^0) \nabla u, (I - \Pi_{p-1}^0) \nabla u)_{0,F} \right] \\ &\quad + \Sigma_F \max(1, d_1) \left[(\Pi_p^0 u, \Pi_p^0 u)_{0,F} + ((I - \Pi_p^0) u, (I - \Pi_p^0) u)_{0,F} \right] \\ &\leq \max(1, c_1, d_1) [D_F(\nabla u, \nabla u)_{0,F} + \Sigma_F(u, u)_{0,F}] = \max(1, c_1, d_1) B_F(u, u) \end{aligned}$$

Performing the same procedure for the lower bound we get:

$$\begin{aligned}
B_F^h(u, u) &\geq D_F(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla u)_{0,F} + \Sigma_F(\Pi_p^0 u, \Pi_p^0 u)_{0,F} + c_0 T_F^\nabla(u, u) + d_0 T_F^0(u, u) \\
&= \left[D_F(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla u)_{0,F} + c_0 D_F((I - \Pi_{p-1}^0) \nabla u, (I - \Pi_{p-1}^0) \nabla u)_{0,F} \right] \\
&\quad + \left[\Sigma_F(\Pi_p^0 u, \Pi_p^0 u)_{0,F} + d_0 \Sigma_F((I - \Pi_p^0) u, (I - \Pi_p^0) u)_{0,F} \right] \\
&\geq D_F \min(1, c_0) \left[(\Pi_{p-1}^0 \nabla u, \Pi_{p-1}^0 \nabla u)_{0,F} + ((I - \Pi_{p-1}^0) \nabla u, (I - \Pi_{p-1}^0) \nabla u)_{0,F} \right] \\
&\quad + \Sigma_F \min(1, d_0) \left[(\Pi_p^0 u, \Pi_p^0 u)_{0,F} + ((I - \Pi_p^0) u, (I - \Pi_p^0) u)_{0,F} \right] \\
&\geq \min(1, c_0, d_0) [D_F(\nabla u, \nabla u)_{0,F} + \Sigma_F(u, u)_{0,F}] = \min(1, c_0, d_0) B_F(u, u)
\end{aligned}$$

That concludes the stability proof. $\square$

E.4 Proof of Lemma 6.3.6

Proof. We begin by proving the coercivity of the global bilinear form $B^h(u^h, v^h)$.

$$\begin{aligned}
B^h(u^h, u^h) &= \sum_{F \in \mathcal{T}_h} B_F^h(u^h, u^h) \\
&\geq \sum_{F \in \mathcal{T}_h} \beta_* B_F(u^h, u^h) \quad (\text{Stability condition}) \\
&= \sum_{F \in \mathcal{T}_h} \beta_* \left(D_F \|\nabla u^h\|_{0,F}^2 + \Sigma_F \|u^h\|_{0,F}^2 \right) \\
&\geq \beta_* \min(D_*, \Sigma_*) \sum_{F \in \mathcal{T}_h} \left(\|\nabla u^h\|_{0,F}^2 + \|u^h\|_{0,F}^2 \right) \tag{E.4.1} \\
&= \beta_* \min(D_*, \Sigma_*) \sum_{F \in \mathcal{T}_h} \|u^h\|_{1,F}^2 \\
&= \beta_* \min(D_*, \Sigma_*) \|u^h\|_{1,V}^2
\end{aligned}$$

Next, we prove the continuity of the global bilinear form. Note, that the local bilinear form $B_F^h(\cdot, \cdot)$ is an inner product on $\mathcal{V}_p(F)$ and therefore satisfies the Cauchy-Schwarz inequality: $B_F^h(u, v) \leq B_F^h(u, u)^{1/2} B_F^h(v, v)^{1/2}$.

$$\begin{aligned}
B^h(u^h, v^h) &= \sum_{F \in \mathcal{T}_h} B_F^h(u^h, v^h) \\
&\leq \sum_{F \in \mathcal{T}_h} \left(B_F^h(u^h, u^h) \right)^{1/2} \left(B_F^h(v^h, v^h) \right)^{1/2} \quad (\text{Cauchy-Schwarz})
\end{aligned}$$

$$\begin{aligned}
&\leq \beta^* \sum_{F \in \mathcal{T}_h} \left(B_F(u^h, u^h) \right)^{1/2} \left(B_F(v^h, v^h) \right)^{1/2} \quad (\text{Stability property, lemma 6.3.5}) \\
&\leq \beta^* \left(\sum_{F \in \mathcal{T}_h} B_F(u^h, u^h) \right)^{1/2} \left(\sum_{F \in \mathcal{T}_h} B_F(v^h, v^h) \right)^{1/2} \quad (\text{Cauchy-Schwarz for sum (E.4.2)}) \\
&= \beta^* \left(B(u^h, u^h) \right)^{1/2} \left(B(v^h, v^h) \right)^{1/2} \\
&\leq \beta^* \left(\max(D^*, \Sigma^*) \|u^h\|_{1,V}^2 \right)^{1/2} \left(\max(D^*, \Sigma^*) \|v^h\|_{1,V}^2 \right)^{1/2} \quad (\text{Continuity of } B(\cdot, \cdot)) \\
&= \beta^* \max(D^*, \Sigma^*) \|u^h\|_{1,V} \|v^h\|_{1,V}
\end{aligned}$$

We have established that $B^h(u^h, v^h)$ is both continuous and coercive, it is trivial to verify that it is also symmetric. Therefore, we may invoke Reisz representation theorem (theorem 3.6.2) and thus prove that u^h both exists and is unique. $\square$

E.5 Proof of Theorem 6.3.7

Proof. Letting $v^h \in \mathcal{V}_p(V)$ we begin the proof with the following triangle inequality:

$$\|u - u^h\|_{1,V} \leq \|u - v^h\|_{1,V} + \|v^h - u^h\|_{1,V} \quad (\text{E.5.1})$$

Letting $w^h = u^h - v^h$ we seek to bound the term $\|v^h - u^h\|_{1,V}$ using the coercivity of $B^h(\cdot, \cdot)$:

$$\begin{aligned}
C_{coer} \|w^h\|_{1,V}^2 &\leq B^h(u^h - v^h, w^h) \\
&= B^h(u^h, w^h) - B^h(v^h, w^h) \\
&= l^h(w^h) - B^h(v^h, w^h) \\
&= l^h(w^h) - B^h(v^h - p + p, w^h), \quad (\text{use } \pm p \text{ where } p \in \mathbb{P}_p(\mathcal{T}_h)) \\
&= l^h(w^h) - \left(B^h(v^h - p, w^h) + B^h(p, w^h) \right) \\
&= l^h(w^h) - \left(B^h(v^h - p, w^h) + B(p, w^h) \right), \quad (\text{use } B^h(p, w^h) = B(p, w^h)) \\
&= l^h(w^h) - \left(B^h(v^h - p, w^h) + B(p - u + u, w^h) \right), \quad (\text{use } \pm u) \\
&= l^h(w^h) - \left(B^h(v^h - p, w^h) + B(p - u, w^h) + B(u, w^h) \right) \\
&= l^h(w^h) - l(w^h) - \left(B^h(v^h - p, w^h) + B(p - u, w^h) \right) \\
&= l^h(w^h) - l(w^h) + B^h(p - v^h, w^h) + B(u - p, w^h)
\end{aligned}$$

Next, we use the continuity of the bilinear forms to give:

$$C_{coer} \|w^h\|_{1,V}^2 \leq l^h(w^h) - l(w^h) + C_1 \|p - v^h\|_{1,h} \|w^h\|_{1,V} + C_2 \|u - p\|_{1,h} \|w^h\|_{1,V}$$

which, after dividing by $\|w^h\|_{1,V}$ and substituting the following triangle inequality:

$$\|p - v^h\|_{1,h} \leq \|p - u\|_{1,h} + \|u - v^h\|_{1,V}$$

gives

$$\begin{aligned} C_{coer} \|w^h\|_{1,V} &\leq \max(1, C_1, C_2) \left(\frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \|p - v^h\|_{1,h} + \|u - p\|_{1,h} \right) \\ &\leq C^* \left(\frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \|p - u\|_{1,h} + \|u - v^h\|_{1,V} + \|u - p\|_{1,h} \right) \\ &\leq 2C^* \left(\frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \|u - v^h\|_{1,V} + \|u - p\|_{1,h} \right). \end{aligned}$$

Therefore, we have the following bound:

$$\|u^h - v^h\|_{1,V} \leq \frac{2C^*}{C_{coer}} \left(\|u - v^h\|_{1,V} + \frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \|u - p\|_{1,h} \right).$$

Substituting this bound into equation (E.5.1) gives:

$$\begin{aligned} \|u - u^h\|_{1,V} &\leq \|u - v^h\|_{1,V} + \frac{2C^*}{C_{coer}} \left(\|u - v^h\|_{1,V} + \frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \|u - p\|_{1,h} \right) \\ &\leq \max \left(1, \frac{2C^*}{C_{coer}} \right) \left(\|u - v^h\|_{1,V} + \frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \|u - p\|_{1,h} \right) \\ &\leq C \left(\inf_{v^h \in \mathcal{V}_p(V)} \|u - v^h\|_{1,V} + \sup_{w^h \in \mathcal{V}_p(V)} \frac{l^h(w^h) - l(w^h)}{\|w^h\|_{1,V}} + \inf_{p \in \mathbb{P}_p(\mathcal{T}_h)} \|u - p\|_{1,h} \right) \end{aligned}$$

□

E.6 Proof of Theorem 6.3.8

Proof. For each term in the abstract error bound (cf. equation (6.3.6)) we derive bounds making use of the approximation properties of $\mathcal{V}_p(F)$ and $\mathbb{P}_p(F)$. Beginning with the first term we make use of lemma 6.3.3:

$$\inf_{v^h \in \mathcal{V}_p(F)} \|u - v^h\|_{1,V} \leq \|u - u_I\|_{1,V} \leq Ch^s |u|_{s+1,V}.$$

Next we bound the second term:

$$l^h(v^h) - l(v^h) = \underbrace{(\Pi_p^0 f, v^h)_{0,V} - (f, v^h)_{0,V}}_A + \underbrace{(r, v^h)_{0,\partial V}^* - (r, v^h)_{0,\partial V}}_B$$

Starting with term A we have:

$$\begin{aligned}
A &= (\Pi_p^0 f, v^h)_{0,V} - (f, v^h)_{0,V} = \left| \sum_{F \in \mathcal{T}_h} (f - \Pi_p^0 f, v^h)_{0,F} \right| = \left| \sum_{F \in \mathcal{T}_h} (f - \Pi_p^0 f, v^h - \Pi_p^0 v^h + \Pi_p^0 v^h)_{0,F} \right| \\
&= \left| \sum_{F \in \mathcal{T}_h} (f - \Pi_p^0 f, v^h - \Pi_p^0 v^h)_{0,F} + \underbrace{(f - \Pi_p^0 f, \Pi_p^0 v^h)_{0,F}}_{=0} \right| \\
&= \left| \sum_{F \in \mathcal{T}_h} (f - \Pi_p^0 f, v^h - \Pi_p^0 v^h)_{0,F} \right| \leq \sum_{F \in \mathcal{T}_h} \|f - \Pi_p^0 f\|_{0,F} \|v^h - \Pi_p^0 v^h\|_{0,F} \\
&\leq \sum_{F \in \mathcal{T}_h} C_{proj} h_F^{s-1} |f|_{s-1,F} C_{proj} h_F |v^h|_{1,F} \leq Ch^s \sum_{F \in \mathcal{T}_h} |f|_{s-1,F} |v^h|_{1,F} \\
&\leq Ch^s \left(\sum_{F \in \mathcal{T}_h} |f|_{s-1,F}^2 \right)^{1/2} \left(\sum_{F \in \mathcal{T}_h} |v^h|_{1,F}^2 \right)^{1/2} \quad (\text{Cauchy-Schwarz for sums}) \\
&\leq Ch^s |f|_{s-1,V} \|v^h\|_{1,V} \quad (\text{as } |v^h|_{1,V} \leq \|v^h\|_{1,V})
\end{aligned}$$

Concerning term B , it is possible to derive error estimates using tools presented in [80], theorems 4.1.3 to 4.1.5 however this would require introducing a great deal of additional information which outside the scope of this thesis. For the moment we shall simply say that guaranteeing the optimal rate of convergence for this term is simply a case of choosing a quadrature high enough to integrate any member of the space of polynomials $\mathbb{P}_p([0, 1])$ and mapping them to e using the NURBS parameterisation.

Finally we bound the last term using lemma 6.3.2

$$\inf_{p \in \mathbb{P}_p(\mathcal{T}_h)} \|u - p\|_{1,h} \leq Ch^s |u|_{s+1,V}.$$

Substituting each of these bounds into the abstract error bound yields the result.

□

E.7 Derivation of MMS Forcing Function

This section shall provide all the supplementary information needed to replicate the MMS benchmark verification test case of section 6.7.1. We begin with the construction of the forcing functions f_1 and f_2 shall be explained. Let $D_{V_1}^1 = D^1|_{V_1}$ and $D_{V_2}^1 = D^1|_{V_2}$, with the same subscript used for the other quantities to denote the same meaning. Let $d(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ be the signed distance function of V_1 , which is defined as:

$$d(\mathbf{x}) = \text{sgn}(\mathbf{x}) \min_{\mathbf{y} \in \partial V_1} \|\mathbf{x} - \mathbf{y}\|_{\mathbb{R}^2}, \quad \text{where} \quad \text{sgn}(\mathbf{x}) = \begin{cases} -1 & \text{if } \mathbf{x} \in V_1 \\ 1 & \text{otherwise} \end{cases}.$$

The distance function has two important properties which also provide useful information. First, for $\mathbf{x} \in V_1$ the gradient $\nabla d(\mathbf{x})$ points in the direction of the closest point on ∂V_1 , denoted $\mathbf{y}$. Second, provided $\mathbf{x}$ is not equidistant to more than one point on ∂V_1 , that $\nabla d(\mathbf{x})$ is equal to the outward normal vector at the nearest point $\mathbf{y}$, this normal vector shall be denoted $\mathbf{n}(\mathbf{x})$. An example is presented in figure E.1.

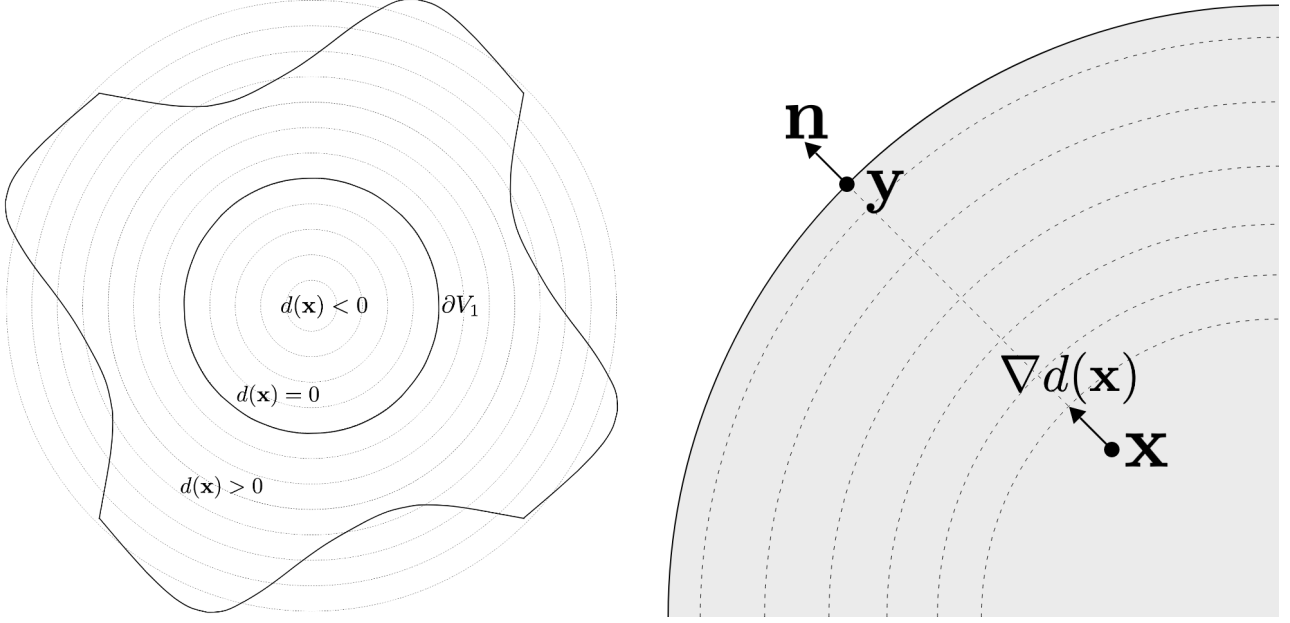


Figure E.1: On the left are the isolines of $d(\mathbf{x})$, on the right is an illustration of the properties of $\nabla d(\mathbf{x})$ for $\mathbf{x} \in V_1$.

Computing the signed distance function of a general curve is a non-trivial task typically accomplished by solving the Eikonal equation, $\|\nabla d\|_{\mathbb{R}^2} = 1$, using fast marching methods. A survey of which may be found in [311]. However, because the boundary curve ∂V_1 in the present test case is a circle of radius $R = 3$ and centre $(0, 0)$, albeit in NURBS form, the distance function has a simple expression. Letting $\mathbf{x} = (x_1, x_2)$, the distance function may be written as:

$$d(\mathbf{x}) = \sqrt{x_1^2 + x_2^2} - R$$

Therefore, all of the physical quantities may be written in the same way as D^1 is below:

$$D^1(\mathbf{x}) = D_{V_1}^1 + (D_{V_2}^1 - D_{V_1}^1) H(d(\mathbf{x})) = D_{V_1}^1 + \llbracket D^1 \rrbracket H(d(\mathbf{x})),$$

where $H(t) : \mathbb{R} \rightarrow \mathbb{R}$ is the Heaviside step function and $\llbracket D^1 \rrbracket$ denotes the jump in D^1 across the discontinuity. Substituting this description of the physical quantities and the manufactured solutions, $\hat{\phi}_1, \hat{\phi}_2$, into equation (6.7.1) to gives:

$$f_1 = -\nabla \cdot (D^1 \nabla \hat{\phi}_1) + \Sigma_r^1 \hat{\phi}_1$$

$$f_2 = -\nabla \cdot (D^2 \nabla \hat{\phi}_2) + \Sigma_r^2 \hat{\phi}_2 - \Sigma_s^{1 \rightarrow 2} \hat{\phi}_1.$$

Therefore, the source functions are piecewise continuous with a discontinuity at ∂V_1 . The diffusion terms require some further explanation as they include derivatives of discontinuous functions, beginning with the diffusion term for $\hat{\phi}_1$:

$$\begin{aligned} -\nabla \cdot (D^1 \nabla \hat{\phi}_1) &= -\nabla \cdot \left((D_{V_1}^1 + \llbracket D^1 \rrbracket H(d(\mathbf{x}))) \nabla \hat{\phi}_1 \right) \\ &= -\nabla \cdot (D_{V_1}^1 \nabla \hat{\phi}_1) - \nabla \cdot (\llbracket D^1 \rrbracket H(d(\mathbf{x})) \nabla \hat{\phi}_1) \\ &= \underbrace{-\left(D_{V_1}^1 + \llbracket D^1 \rrbracket H(d(\mathbf{x})) \right) \nabla^2 \hat{\phi}_1}_{\text{term 1}} - \underbrace{\llbracket D^1 \rrbracket \nabla H(d(\mathbf{x})) \cdot \nabla \hat{\phi}_1}_{\text{term 2}}. \end{aligned}$$

Term 1 is a volume source, defined for all V , and may be written as:

$$-\left(D_{V_1}^1 + \llbracket D^1 \rrbracket H(d(\mathbf{x})) \right) \nabla^2 \hat{\phi}_1 = \begin{cases} -D_{V_1}^1 \nabla^2 \hat{\phi}_1 & \text{if } \mathbf{x} \in V_1 \\ -D_{V_2}^1 \nabla^2 \hat{\phi}_1 & \text{if } \mathbf{x} \in V_2 \end{cases}.$$

Term 2 is a line source, defined only on ∂V_1 , and is computed as follows. Beginning with the $\nabla H(d(\mathbf{x}))$ term:

$$\frac{\partial}{\partial x_i} H(d(\mathbf{x})) = \frac{\partial H(d(\mathbf{x}))}{\partial d(\mathbf{x})} \cdot \frac{\partial d(\mathbf{x})}{\partial x_i} = \delta(d(\mathbf{x})) \frac{\partial d(\mathbf{x})}{\partial x_i},$$

where $\delta(t)$ is the Dirac delta distribution. Therefore, $\nabla H(d(\mathbf{x})) = \delta(d(\mathbf{x})) \nabla d(\mathbf{x})$. Substituting this back into term 2 gives:

$$\llbracket D^1 \rrbracket \nabla H(d(\mathbf{x})) \cdot \nabla \hat{\phi}_1 = \llbracket D^1 \rrbracket (\delta(d(\mathbf{x})) \nabla d(\mathbf{x})) \cdot \nabla \hat{\phi}_1$$

The same result can be reached for the group 2 diffusion term, $-\nabla \cdot (D^2 \nabla \hat{\phi}_2)$, using the same argument.

To see how the forcing linear forms are computed in practice, consider the forcing term:

$$\begin{aligned} (v_1, f_1)_{0,V} &= (v_1, -\nabla \cdot (D^1 \nabla \hat{\phi}_1) + \Sigma_r^1 \hat{\phi}_1)_{0,V} \\ &= -\left(v_1, D_{V_1}^1 \nabla^2 \hat{\phi}_1 \right)_{0,V_1} - \left(v_1, D_{V_2}^1 \nabla^2 \hat{\phi}_1 \right)_{0,V_2} + \llbracket D^1 \rrbracket (v_1, \mathbf{n}_1 \cdot \nabla \hat{\phi}_1)_{0,\partial V_1} + (v_1, \Sigma_r^1 \hat{\phi}_1)_{0,V} \end{aligned}$$

Where $\mathbf{n}_1(\mathbf{x}) = \nabla d(\mathbf{x})$, $\mathbf{x} \in \partial V_1$, is the outward normal to ∂V_1 . This is a property of the distance function which was explained earlier.

E.8 C5G7 Supplementary Data

The macroscopic neutron diffusion coefficients D^g are computed from the macroscopic nuclear cross section data given in [287] as follows:

$$D^g = \frac{1}{3\Sigma_{tr}^g}$$

where Σ_{tr}^g is the macroscopic transport-corrected total cross section.

The QoIs for the C5G7 are listed as follows, along with their full mathematical definition. Let $V_{i,j}$ denote the pin in row i and column j of the lattice, with $(0,0)$ being the bottom left and $(33,33)$ being the top right (see figure 6.17). The inner MOX assembly shall be denoted V_{MOX} , the outer UO_2 assembly V_{UO2o} and the inner UO_2 assembly V_{UO2i} .

- The nuclear criticality eigenvalue, K_{eff} this is the value of the largest eigenvalue of the system. It is computed during the power iteration.
- The maximum pin power P_{max} :

$$P_{\text{max}} = \max_{i,j} \int_{V_{i,j}} \sum_{g=1}^G \Sigma_f^g \phi_g dx$$

- The minimum pin power P_{min} :

$$P_{\text{min}} = \min_{i,j} \int_{V_{i,j}} \sum_{g=1}^G \Sigma_f^g \phi_g dx$$

- The minimum pin power P_{min} :

$$P_{\text{min}} = \min_{i,j} \int_{V_{i,j}} \sum_{g=1}^G \Sigma_f^g \phi_g dx$$

- The MOX assembly power

$$\int_{V_{\text{MOX}}} \sum_{g=1}^G \Sigma_f^g \phi_g dx$$

- The inner UO_2 assembly power:

$$\int_{V_{\text{UO2i}}} \sum_{g=1}^G \Sigma_f^g \phi_g dx$$

- The outer UO_2 assembly power:

$$\int_{V_{\text{UO2o}}} \sum_{g=1}^G \Sigma_f^g \phi_g dx$$

The material coefficients for the C5G7 are given for each material, these will be presented in two blocks. The first will give the values of χ^g , ν^g , Σ_f^g , Σ_a^g and D^g . The second will provide the scatter cross sections for all possible pairs of energies.

- Moderator macroscopic cross sections:

Group	χ^g	ν^g	Σ_f^g	Σ_a^g	D^g
1	0.00000e+00	0.00000e+00	0.00000e+00	6.01050e-04	2.09372e+00
2	0.00000e+00	0.00000e+00	0.00000e+00	1.57930e-05	8.07161e-01
3	0.00000e+00	0.00000e+00	0.00000e+00	3.37160e-04	5.64675e-01
4	0.00000e+00	0.00000e+00	0.00000e+00	1.94060e-03	5.70434e-01
5	0.00000e+00	0.00000e+00	0.00000e+00	5.74160e-03	4.64253e-01
6	0.00000e+00	0.00000e+00	0.00000e+00	1.50010e-02	2.65721e-01
7	0.00000e+00	0.00000e+00	0.00000e+00	3.72390e-02	1.25768e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	4.447770e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	1.134000e-01	2.823340e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	7.234700e-04	1.299400e-01	3.452560e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	3.749900e-06	6.234000e-04	2.245700e-01	9.102840e-02	7.143700e-05	0.000000e+00	0.000000e+00
To group 5	5.318400e-08	4.800200e-05	1.699900e-02	4.155100e-01	1.391380e-01	2.215700e-03	0.000000e+00
To group 6	0.000000e+00	7.448600e-06	2.644300e-03	6.373200e-02	5.118200e-01	6.999130e-01	1.324400e-01
To group 7	0.000000e+00	1.045500e-06	5.034400e-04	1.213900e-02	6.122900e-02	5.373200e-01	2.480700e+00

- UO₂ macroscopic cross sections:

Group	χ	ν	Σ_f^g	Σ_a^g	D^g
1	5.87910e-01	2.78145e+00	7.21206e-03	8.02480e-03	1.87320e+00
2	4.11760e-01	2.47443e+00	8.19301e-04	3.71740e-03	1.01070e+00
3	3.39060e-04	2.43383e+00	6.45320e-03	2.67690e-02	6.93884e-01
4	1.17610e-07	2.43380e+00	1.85648e-02	9.62360e-02	6.01286e-01
5	0.00000e+00	2.43380e+00	1.78084e-02	3.00200e-02	1.06906e+00
6	0.00000e+00	2.43380e+00	8.30348e-02	1.11260e-01	8.43523e-01
7	0.00000e+00	2.43380e+00	2.16004e-01	2.82780e-01	5.90591e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	1.275370e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	4.237800e-02	3.244560e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	9.437400e-06	1.631400e-03	4.509400e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	5.516300e-09	3.142700e-09	2.679200e-03	4.525650e-01	1.252500e-04	0.000000e+00	0.000000e+00
To group 5	0.000000e+00	0.000000e+00	0.000000e+00	5.566400e-03	2.714010e-01	1.296800e-03	0.000000e+00
To group 6	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.025500e-02	2.658020e-01	8.545800e-03
To group 7	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.002100e-08	1.680900e-02	2.730800e-01

- 4.3% MOX fuel-clad macroscopic cross sections:

Group	χ	ν	Σ_f^g	Σ_a^g	D^g
1	5.87910e-01	2.85209e+00	7.62704e-03	8.43390e-03	1.86500e+00
2	4.11760e-01	2.89099e+00	8.76898e-04	3.75770e-03	1.00751e+00
3	3.39060e-04	2.85486e+00	5.69835e-03	2.79700e-02	6.89030e-01
4	1.17610e-07	2.86073e+00	2.28872e-02	1.04210e-01	5.87970e-01
5	0.00000e+00	2.85447e+00	1.07635e-02	1.39940e-01	7.82056e-01
6	0.00000e+00	2.86415e+00	2.32757e-01	4.09180e-01	4.90920e-01
7	0.00000e+00	2.86780e+00	2.48968e-01	4.09350e-01	4.88149e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	1.288760e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	4.141300e-02	3.254520e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	8.229000e-06	1.639500e-03	4.531880e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	5.040500e-09	1.598200e-09	2.614200e-03	4.571730e-01	1.604600e-04	0.000000e+00	0.000000e+00
To group 5	0.000000e+00	0.000000e+00	0.000000e+00	5.539400e-03	2.768140e-01	2.005100e-03	0.000000e+00
To group 6	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	9.312700e-03	2.529620e-01	8.494800e-03
To group 7	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	9.165600e-09	1.485000e-02	2.650070e-01

- 7.0% MOX fuel-clad macroscopic cross sections:

Group	χ	ν	Σ_f^g	Σ_a^g	D^g
1	5.87910e-01	2.88498e+00	8.25446e-03	9.06570e-03	1.83834e+00
2	4.11760e-01	2.91079e+00	1.32565e-03	4.29670e-03	9.96906e-01
3	3.39060e-04	2.86574e+00	8.42156e-03	3.28810e-02	6.75058e-01
4	1.17610e-07	2.87063e+00	3.28730e-02	1.22030e-01	5.63810e-01
5	0.00000e+00	2.86714e+00	1.59636e-02	1.82980e-01	7.02941e-01
6	0.00000e+00	2.86658e+00	3.23794e-01	5.68460e-01	3.99872e-01
7	0.00000e+00	2.87539e+00	3.62803e-01	5.85210e-01	3.90502e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	1.304570e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	4.179200e-02	3.284280e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	8.510500e-06	1.643600e-03	4.583710e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	5.132900e-09	2.201700e-09	2.533100e-03	4.637090e-01	1.761900e-04	0.000000e+00	0.000000e+00
To group 5	0.000000e+00	0.000000e+00	0.000000e+00	5.476600e-03	2.823130e-01	2.276000e-03	0.000000e+00
To group 6	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.728900e-03	2.497510e-01	8.864500e-03
To group 7	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	9.001600e-09	1.311400e-02	2.595290e-01

- 8.7% MOX fuel-clad macroscopic cross sections:

Group	χ	ν	Σ_f^g	Σ_a^g	D^g
1	5.87910e-01	2.90426e+00	8.67209e-03	9.48620e-03	1.82105e+00
2	4.11760e-01	2.91795e+00	1.62426e-03	4.65560e-03	9.89986e-01
3	3.39060e-04	2.86986e+00	1.02716e-02	3.62400e-02	6.65991e-01
4	1.17610e-07	2.87491e+00	3.90447e-02	1.32720e-01	5.49897e-01
5	0.00000e+00	2.87175e+00	1.92576e-02	2.08400e-01	6.63015e-01
6	0.00000e+00	2.86752e+00	3.74888e-01	6.58700e-01	3.61914e-01
7	0.00000e+00	2.87808e+00	4.30599e-01	6.90170e-01	3.48956e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	1.315040e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	4.204600e-02	3.304030e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	8.697200e-06	1.646300e-03	4.617920e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	5.193800e-09	2.600600e-09	2.474900e-03	4.680210e-01	1.859700e-04	0.000000e+00	0.000000e+00
To group 5	0.000000e+00	0.000000e+00	0.000000e+00	5.433000e-03	2.857710e-01	2.391600e-03	0.000000e+00
To group 6	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.397300e-03	2.476140e-01	8.968100e-03
To group 7	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.928000e-09	1.232200e-02	2.560930e-01

- Guide tube macroscopic cross sections:

Group	χ	ν	Σ_f^g	Σ_a^g	D^g
1	0.000000e+00	0.000000e+00	0.000000e+00	5.11320e-04	2.64483e+00
2	0.000000e+00	0.000000e+00	0.000000e+00	7.58010e-05	1.13704e+00
3	0.000000e+00	0.000000e+00	0.000000e+00	3.15720e-04	1.17272e+00
4	0.000000e+00	0.000000e+00	0.000000e+00	1.15820e-03	1.18641e+00
5	0.000000e+00	0.000000e+00	0.000000e+00	3.39750e-03	9.96691e-01
6	0.000000e+00	0.000000e+00	0.000000e+00	9.18780e-03	5.89303e-01
7	0.000000e+00	0.000000e+00	0.000000e+00	2.32420e-02	2.84378e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	6.616590e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	5.907000e-02	2.403770e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	2.833400e-04	5.243500e-02	1.832970e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	1.462200e-06	2.499000e-04	9.239700e-02	7.885110e-02	3.733300e-05	0.000000e+00	0.000000e+00
To group 5	2.064200e-08	1.923900e-05	6.944600e-03	1.701400e-01	9.973720e-02	9.172600e-04	0.000000e+00
To group 6	0.000000e+00	2.987500e-06	1.080300e-03	2.588100e-02	2.067900e-01	3.167650e-01	4.979200e-02
To group 7	0.000000e+00	4.214000e-07	2.056700e-04	4.929700e-03	2.447800e-02	2.387700e-01	1.099120e+00

- Fission chamber macroscopic cross sections:

Group	χ	ν	Σ_f^g	Σ_a^g	D^g
1	5.87910e-01	2.76283e+00	4.79002e-09	5.11320e-04	2.64483e+00
2	4.11760e-01	2.46239e+00	5.82564e-09	7.58130e-05	1.13704e+00
3	3.39060e-04	2.43380e+00	4.63719e-07	3.16430e-04	1.17268e+00
4	1.17610e-07	2.43380e+00	5.24406e-06	1.16750e-03	1.18616e+00
5	0.000000e+00	2.43380e+00	1.45390e-07	3.39770e-03	9.96631e-01
6	0.000000e+00	2.43380e+00	7.14972e-07	9.18860e-03	5.89303e-01
7	0.000000e+00	2.43380e+00	2.08041e-06	2.32440e-02	2.84380e-01

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
To group 1	6.616590e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 2	5.907000e-02	2.403770e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 3	2.833400e-04	5.243500e-02	1.834250e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
To group 4	1.462200e-06	2.499000e-04	9.228800e-02	7.907690e-02	3.734000e-05	0.000000e+00	0.000000e+00
To group 5	2.064200e-08	1.923900e-05	6.936500e-03	1.699900e-01	9.975700e-02	9.174200e-04	0.000000e+00
To group 6	0.000000e+00	2.987500e-06	1.079000e-03	2.586000e-02	2.067900e-01	3.167740e-01	4.979300e-02
To group 7	0.000000e+00	4.214000e-07	2.054300e-04	4.925600e-03	2.447800e-02	2.387600e-01	1.099100e+00

Appendix F

Appendix to Chapter 7

F.1 Proof of lemma 7.3.3

Proof. Beginning with the definition of $a(v, u)$ we have:

$$\begin{aligned} a(v, u) &= \int_V v \boldsymbol{\Omega} \cdot \nabla u \, dx + \int_V \Sigma(\mathbf{x}) v u \, dx - \int_{\partial V_-} \boldsymbol{\Omega} \cdot \mathbf{n} v u \, ds \\ &= \int_V v \boldsymbol{\Omega} \cdot \nabla u \, dx + \int_V \Sigma(\mathbf{x}) v u \, dx + \int_{\partial V_-} |\boldsymbol{\Omega} \cdot \mathbf{n}| v u \, ds \end{aligned}$$

Letting $\Sigma^* = \sup_{\mathbf{x} \in V} \Sigma(\mathbf{x})(\mathbf{x})$ and applying Cauchy-Schwarz to each term gives:

$$|a(v, u)| \leq \|v\|_{0,V} \|\boldsymbol{\Omega} \nabla u\|_{0,V} + \Sigma^* \|v\|_{0,V} \|u\|_{0,V} + \|v\|_{L^2_{\boldsymbol{\Omega}}(V)} \|u\|_{L^2_{\boldsymbol{\Omega}}(V)}.$$

Recalling the Graph norm $\|u\|_{\mathcal{W}}^2 = \|u\|_{0,V}^2 + \|\boldsymbol{\Omega} \cdot \nabla u\|_{0,V}^2$ we may infer $\|u\|_{0,V} \leq \|u\|_{\mathcal{W}}$ and $\|\boldsymbol{\Omega} \cdot \nabla u\|_{0,V}^2 \leq \|u\|_{\mathcal{W}}^2$. Additionally, by applying the Graph space trace inequality (equation (7.3.3)) we get:

$$|a(v, u)| \leq (1 + \Sigma(\mathbf{x})^* + C_\gamma^2) \|u\|_{\mathcal{W}} \|v\|_{\mathcal{W}}.$$

□

F.2 Proof of Lemma 7.3.4

Proof. Beginning with the definition of $a(v, u)$ we have:

$$a(v, u) = \int_V v \boldsymbol{\Omega} \cdot \nabla u \, dx + \int_V \Sigma(\mathbf{x}) v u \, dx + \int_{\partial V_-} |\boldsymbol{\Omega} \cdot \mathbf{n}| v u \, ds.$$

Therefore:

$$a(u, u) = \int_V u \boldsymbol{\Omega} \cdot \nabla u \, dx + \int_V \Sigma(\mathbf{x}) u^2 \, dx + \int_{\partial V_-} |\boldsymbol{\Omega} \cdot \mathbf{n}| u^2 \, ds. \quad (\text{F.2.1})$$

Using the chain rule we have:

$$\mathbf{\Omega} \cdot \nabla u \, u = \frac{1}{2} \nabla \cdot (\mathbf{\Omega} u^2) - \frac{1}{2} \nabla \cdot \mathbf{\Omega} u^2.$$

Next, we integrate over V and use Gauss' theorem to yield:

$$\int_V \mathbf{\Omega} \cdot \nabla u \, u \, dx = \frac{1}{2} \int_{\partial V} \mathbf{\Omega} \cdot \mathbf{n} \, u^2 \, ds - \frac{1}{2} \int_V \nabla \cdot \mathbf{\Omega} \, u^2 \, dx.$$

Substituting this into equation (F.2.1) gives:

$$a(u, u) = \int_V \left(\Sigma(\mathbf{x}) - \frac{1}{2} \nabla \cdot \mathbf{\Omega} \right) u^2 \, dx + \frac{1}{2} \int_{\partial V} \mathbf{\Omega} \cdot \mathbf{n} \, u^2 \, ds + \int_{\partial V_-} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds.$$

The boundary terms may then be treated as follows:

$$\begin{aligned} & \frac{1}{2} \int_{\partial V} \mathbf{\Omega} \cdot \mathbf{n} \, u^2 \, ds + \int_{\partial V_-} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds \\ &= \\ & -\frac{1}{2} \int_{\partial V_-} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds + \frac{1}{2} \int_{\partial V_+} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds + \int_{\partial V_-} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds. \\ &= \\ & \frac{1}{2} \left(\int_{\partial V_-} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds + \int_{\partial V_+} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds \right) = \frac{1}{2} \int_{\partial V} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds. \end{aligned}$$

Therefore:

$$\begin{aligned} a(u, u) &= \int_V \left(\Sigma(\mathbf{x}) - \frac{1}{2} \nabla \cdot \mathbf{\Omega} \right) u^2 \, dx + \frac{1}{2} \int_{\partial V} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds \\ &\geq \Sigma_* \int_V u^2 \, dx + \frac{1}{2} \int_{\partial V} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds, \quad (\nabla \cdot \mathbf{\Omega} = 0) \\ &= \Sigma_* \|u\|_{0,V}^2 + \frac{1}{2} \int_{\partial V} |\mathbf{\Omega} \cdot \mathbf{n}| \, u^2 \, ds, \end{aligned}$$

□

Bibliography

- [1] L. Beirão da Veiga, A. Russo, G. Vacca, [The Virtual Element Method with curved edges](#), ESAIM: Mathematical Modelling and Numerical Analysis 53 (2) (2019) 375–404. doi:[10.1051/m2an/2018052](#).
URL <https://www.esaim-m2an.org/10.1051/m2an/2018052>
- [2] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, [Isogeometric Analysis](#), John Wiley and Sons, Ltd, Chichester, UK, 2009. doi:[10.1002/9780470749081](#).
URL <http://doi.wiley.com/10.1002/9780470749081>
- [3] P. F. Antonietti, P. Houston, G. Pennesi, [Fast Numerical Integration on Polytopic Meshes with Applications to Discontinuous Galerkin Finite Element Methods](#), Journal of Scientific Computing 77 (3) (2018) 1339–1370. doi:[10.1007/s10915-018-0802-y](#).
URL <http://link.springer.com/10.1007/s10915-018-0802-y>
- [4] A. Cangiani, E. H. Georgoulis, T. Pryer, O. J. Sutton, [A posteriori error estimates for the virtual element method](#), Numerische Mathematik 137 (4) (2017) 857–893. doi:[10.1007/s00211-017-0891-9](#).
URL <http://link.springer.com/10.1007/s00211-017-0891-9>
- [5] J. Warsa, [Analytical SN solutions in heterogeneous slabs using symbolic algebra computer programs](#), Annals of Nuclear Energy 29 (7) (2002) 851–874. doi:[10.1016/S0306-4549\(01\)00080-9](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454901000809>
- [6] J. H. Kim, C. Starr, [Global electrification and nuclear power: Toward sustainable growth in the new millennium](#), Progress in Nuclear Energy 37 (1-4) (2000) 11–18. doi:[10.1016/S0149-1970\(00\)00017-2](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197000000172>
- [7] C. Bohringer, [The Kyoto Protocol: A Review and Perspectives](#), Oxford Review of Economic Policy 19 (3) (2003) 451–466. doi:[10.1093/oxrep/19.3.451](#).
URL <https://academic.oup.com/oxrep/article-lookup/doi/10.1093/oxrep/19.3.451>
- [8] W. Liu, W. J. McKibbin, A. C. Morris, P. J. Wilcoxon, [Global economic and environmental outcomes of the Paris Agreement](#), Energy Economics 90 (2020) 104838.

- doi:10.1016/j.eneco.2020.104838.
URL <https://linkinghub.elsevier.com/retrieve/pii/S014098832030178X>
- [9] S. R. Sinsel, R. L. Riemke, V. H. Hoffmann, [Challenges and solution technologies for the integration of variable renewable energy sources—a review](#), *Renewable Energy* 145 (2020) 2271–2285. doi:10.1016/j.renene.2019.06.147.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0960148119309875>
- [10] A. A. Kebede, T. Kalogiannis, J. Van Mierlo, M. Berecibar, [A comprehensive review of stationary energy storage devices for large scale renewable energy sources grid integration](#), *Renewable and Sustainable Energy Reviews* 159 (2022) 112213. doi:10.1016/j.rser.2022.112213.
URL <https://linkinghub.elsevier.com/retrieve/pii/S1364032122001368>
- [11] L. Reichenberg, F. Hedenus, M. Odenberger, F. Johnsson, [Tailoring large-scale electricity production from variable renewable energy sources to accommodate baseload generation in europe](#), *Renewable Energy* 129 (2018) 334–346. doi:10.1016/j.renene.2018.05.014.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0960148118305330>
- [12] D. Gabaldon, E. Spiegel, J. Van den Berg, [Betting Big on Baseload](#), *The Electricity Journal* 19 (7) (2006) 74–85. doi:10.1016/j.tej.2006.08.004.
URL <https://linkinghub.elsevier.com/retrieve/pii/S1040619006000935>
- [13] M. Nicholson, T. Biegler, B. W. Brook, [How carbon pricing changes the relative competitiveness of low-carbon baseload generating technologies](#), *Energy* 36 (1) (2011) 305–313. doi:10.1016/j.energy.2010.10.039.
URL <https://linkinghub.elsevier.com/retrieve/pii/S036054421000602X>
- [14] V. Olkkonen, J. Ekström, A. Hast, S. Syri, [Utilising demand response in the future Finnish energy system with increased shares of baseload nuclear power and variable renewable energy](#), *Energy* 164 (2018) 204–217. doi:10.1016/j.energy.2018.08.210.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0360544218317468>
- [15] A. Chatzivasileiadi, E. Ampatzi, I. P. Knight, [The Implications of Demand Response Measures and Electrification of Transport on UK Household Energy Demand and Consumption](#), *Energy Procedia* 134 (2017) 89–98. doi:10.1016/j.egypro.2017.09.530.
URL <https://linkinghub.elsevier.com/retrieve/pii/S1876610217346581>
- [16] K. Vu, K. Hartley, [Effects of digital transformation on electricity sector growth and productivity: A study of thirteen industrialized economies](#), *Utilities Policy* 74 (2022) 101326. doi:10.1016/j.jup.2021.101326.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0957178721001594>
- [17] X. Zhou, D. Zhou, Z. Zhao, Q. Wang, [A framework to analyze carbon impacts of digital economy: The case of China](#), *Sustainable Production and Consumption* 31 (2022) 357–

369. doi:10.1016/j.spc.2022.03.002.
URL <https://linkinghub.elsevier.com/retrieve/pii/S2352550922000549>
- [18] E. Department for Business, I. Strategy, [UK energy in brief 2021](#), Tech. rep., Department for Business, Energy and Industrial Strategy, London (2021).
URL <https://www.gov.uk/government/statistics/uk-energy-in-brief-2021>
- [19] V. Maurel, [EPR, spearhead of the European nuclear renaissance](#), Nuclear Engineering and Design 236 (14-16) (2006) 1447–1454. doi:10.1016/j.nucengdes.2006.03.050.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0029549306003372>
- [20] J. Xing, D. Song, Y. Wu, [HPR1000: Advanced Pressurized Water Reactor with Active and Passive Safety](#), Engineering 2 (1) (2016) 79–87. doi:10.1016/J.ENG.2016.01.017.
URL <https://linkinghub.elsevier.com/retrieve/pii/S2095809916301515>
- [21] T. N. A. Office, [The decommissioning of the AGR nuclear power stations](#), Tech. rep., The National Audit Office, London (2022).
URL <https://www.nao.org.uk/report/the-decommissioning-of-the-agr-nuclear-power-sta>
- [22] E. energy, CGN, [The sizewell c project: Consultation document on further proposed project changes](#) (2021).
- [23] E. Energy, CGN, [Bradwell b: Stage one - consultation summary document](#) (2020).
- [24] H. of Commons Committee of Public Accounts, [Ministry of defence nuclear programme: Sixty-first report of session 2017–19](#) (2018).
- [25] J. Gates, [Astute Class Nuclear Submarine: 2010 to Date](#), J H Haynes & Co Ltd, 2018.
- [26] D. Brighton, P. Mart, G. Clark, M. Rowan, [The use of fuel cells to enhance the underwater performance of conventional diesel electric submarines](#), Journal of Power Sources 51 (3) (1994) 375–389. doi:10.1016/0378-7753(94)80106-1.
URL <https://linkinghub.elsevier.com/retrieve/pii/0378775394801061>
- [27] C. Fiori, A. Dell’Era, F. Zuccari, A. Santiangeli, A. D’Orazio, F. Orecchini, [Hydrides for submarine applications: Overview and identification of optimal alloys for air independent propulsion maximization](#), International Journal of Hydrogen Energy 40 (35) (2015) 11879–11889. doi:10.1016/j.ijhydene.2015.02.105.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0360319915005066>
- [28] J. Yu, [Marine Nuclear Power Technology](#), Springer Singapore, Singapore, 2020. doi:10.1007/978-981-15-2894-1.
URL <http://link.springer.com/10.1007/978-981-15-2894-1>
- [29] R. Burcher, L. J. Rydill, [Concepts in Submarine Design](#), Cambridge University Press, 1994. doi:10.1017/CB09781107050211.
URL <https://www.cambridge.org/core/product/identifier/9781107050211/type/book>

- [30] E. Department for Business, I. Strategy, Advanced modular reactor research, development & demonstration programme: Indicative programme outline (2022).
- [31] H. Government, Industrial strategy: Nuclear sector deal (2018).
- [32] C. Lu, J. Lyu, L. Zhang, A. Gong, Y. Fan, J. Yan, X. Li, [Nuclear Power Plants With Artificial Intelligence in Industry 4.0 Era: Top-Level Design and Current Applications—A Systemic Review](#), *IEEE Access* 8 (2020) 194315–194332. doi:10.1109/ACCESS.2020.3032529.
URL <https://ieeexplore.ieee.org/document/9233420/>
- [33] J. Leng, D. Wang, W. Shen, X. Li, Q. Liu, X. Chen, [Digital twins-based smart manufacturing system design in Industry 4.0: A review](#), *Journal of Manufacturing Systems* 60 (2021) 119–137. doi:10.1016/j.jmsy.2021.05.011.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0278612521001151>
- [34] F. Magoulès, M. Parsons, L. Smith, [Advances in High Performance Computing: on the path to Exascale software](#), *Advances in Engineering Software* 111 (2017) 1–2. doi:10.1016/j.advengsoft.2017.06.007.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0965997817305938>
- [35] P. K. Romano, S. P. Hamilton, R. O. Rahaman, A. Novak, E. Merzari, S. M. Harper, P. C. Shriwise, T. M. Evans, [A Code-Agnostic Driver Application for Coupled Neutronics and Thermal-Hydraulic Simulations](#), *Nuclear Science and Engineering* 195 (4) (2021) 391–411. doi:10.1080/00295639.2020.1830620.
URL <https://www.tandfonline.com/doi/full/10.1080/00295639.2020.1830620>
- [36] B. Lindley, D. Allen, J. Lillington, A. Smethurst, P. Smith, D. Bowman, L. Dwyer, K. Lai, A. Levers, K. Vikhorev, D. Litskevich, B. Merk, E. Patelli, E. Patterson, M. Bankhead, A. Peakman, S. de Haas, C. Jackson, J. Draup, E. Galenne, G. Parks, E. Shwageraus, M. D. Eaton, [Modelling and Simulation Activities in Support of the UK Nuclear R&D Programme on Digital Reactor Design](#), in: Volume 3: Nuclear Fuel and Material, Reactor Physics, and Transport Theory, American Society of Mechanical Engineers, 2018. doi:10.1115/ICONE26-81090.
URL <https://asmedigitalcollection.asme.org/ICONE/proceedings/ICONE26/51456/London,England/272751>
- [37] G. Locatelli, M. Mancini, N. Todeschini, [Generation IV nuclear reactors: Current status and future prospects](#), *Energy Policy* 61 (2013) 1503–1520. doi:10.1016/j.enpol.2013.06.101.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0301421513006083>
- [38] G. H. Marcus, A. E. Levin, [New Designs for the Nuclear Renaissance](#), *Physics Today* 55 (4) (2002) 54–60fig. doi:10.1063/1.1480783.
URL <http://physicstoday.scitation.org/doi/10.1063/1.1480783>

- [39] A. Santamarina, D. Bernard, P. Blaise, P. Leconte, C. Vaglio, J.-F. Vidal, [Recent advances in accurate neutronic calculation of GEN-3 LWR reactors](#), *Annals of Nuclear Energy* 87 (2016) 68–75. doi:10.1016/j.anucene.2015.05.035.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454915003540>
- [40] M. Ding, X. Zhou, H. Zhang, H. Bian, Q. Yan, [A review of the development of nuclear fuel performance analysis and codes for PWRs](#), *Annals of Nuclear Energy* 163 (2021) 108542. doi:10.1016/j.anucene.2021.108542.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454921004187>
- [41] B. Tomkins, [Structural Integrity Issues in the Nuclear Industry](#), in: *Comprehensive Structural Integrity*, Elsevier, 2003, pp. 173–192. doi:10.1016/B0-08-043749-4/01006-5.
URL <https://linkinghub.elsevier.com/retrieve/pii/B0080437494010065>
- [42] A. K. Nayak, J. B. Jyeshtharaj, [Advances of Computational Fluid Dynamics in Nuclear Reactor Design and Safety Assessment](#), Elsevier, 2019. doi:10.1016/C2016-0-04959-8.
URL <https://linkinghub.elsevier.com/retrieve/pii/C20160049598>
- [43] F. D’Auria, [Thermal-Hydraulics of Water Cooled Nuclear Reactors](#), Elsevier, 2017. doi:10.1016/C2015-0-00235-0.
URL <https://linkinghub.elsevier.com/retrieve/pii/C20150002350>
- [44] C. Demazière, *Modelling of Nuclear Reactor Multi-physics*, Academic Press, London, 2019.
- [45] D. Brown, M. Chadwick, R. Capote, A. Kahler, A. Trkov, M. Herman, A. Sonzogni, Y. Danon, A. Carlson, M. Dunn, D. Smith, G. Hale, G. Arbanas, R. Arcilla, C. Bates, B. Beck, B. Becker, F. Brown, R. Casperson, J. Conlin, D. Cullen, M.-A. Descalle, R. Firestone, T. Gaines, K. Guber, A. Hawari, J. Holmes, T. Johnson, T. Kawano, B. Kiedrowski, A. Koning, S. Kopecky, L. Leal, J. Lestone, C. Lubitz, J. Márquez Damián, C. Mattoon, E. McCutchan, S. Mughabghab, P. Navratil, D. Neudecker, G. Nobre, G. Noguere, M. Paris, M. Pigni, A. Plompen, B. Pritychenko, V. Pronyaev, D. Roubtsov, D. Rochman, P. Romano, P. Schillebeeckx, S. Simakov, M. Sin, I. Sirakov, B. Sleaford, V. Sobes, E. Soukhovitskii, I. Stetcu, P. Talou, I. Thompson, S. van der Marck, L. Welser-Sherill, D. Wiarda, M. White, J. Wormald, R. Wright, M. Zerkle, G. Žerovnik, Y. Zhu, [ENDF/B-VIII.0: The 8 th Major Release of the Nuclear Reaction Data Library with CIELO-project Cross Sections, New Standards and Thermal Scattering Data](#), *Nuclear Data Sheets* 148 (2018) 1–142. doi:10.1016/j.nds.2018.02.001.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0090375218300206>
- [46] L. Zheng, S. Huang, K. Wang, [Criticality benchmarking of ENDF/B-VIII.0 and JEFF-3.3 neutron data libraries with RMC code](#), *Nuclear Engineering and Technology* 52 (9) (2020) 1917–1925. doi:10.1016/j.net.2020.02.022.
URL <https://linkinghub.elsevier.com/retrieve/pii/S1738573319308861>

- [47] P. Tamagno, W. van Rooijen, [Uncertainty analysis of the prototype FBR Monju with the JENDL-4.0 nuclear data set](#), *Annals of Nuclear Energy* 51 (2013) 257–273. doi:[10.1016/j.anucene.2012.06.039](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454912003234>
- [48] A. V. Ignatyuk, B. I. Fursov, [The latest BROND-3 developments](#), in: ND2007, EDP Sciences, Les Ulis, France, 2007. doi:[10.1051/ndata:07641](#).
URL <http://nd2007.edpsciences.org/10.1051/ndata:07641>
- [49] W. Shu, T. Zu, L. Cao, H. Wu, [Performance of CENDL-3.2 evaluated nuclear data library for the shielding benchmarks](#), *Progress in Nuclear Energy* 136 (2021) 103727. doi:[10.1016/j.pnucene.2021.103727](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197021000949>
- [50] A. Koning, D. Rochman, J.-C. Sublet, N. Dzysiuk, M. Fleming, S. van der Marck, [TENDL: Complete Nuclear Data Library for Innovative Nuclear Science and Technology](#), *Nuclear Data Sheets* 155 (2019) 1–55. doi:[10.1016/j.nds.2019.01.002](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S009037521930002X>
- [51] M. Chadwick, R. Capote, A. Trkov, M. Herman, D. Brown, G. Hale, A. Kahler, P. Talou, A. Plompen, P. Schillebeeckx, M. Pigni, L. Leal, Y. Danon, A. Carlson, P. Romain, B. Morillon, E. Bauge, F.-J. Hambsch, S. Kopecky, G. Giorginis, T. Kawano, J. Lestone, D. Neudecker, M. Rising, M. Paris, G. Nobre, R. Arcilla, O. Cabellos, I. Hill, E. Dupont, A. Koning, D. Cano-Ott, E. Mendoza, J. Balibrea, C. Paradela, I. Durán, J. Qian, Z. Ge, T. Liu, L. Hanlin, X. Ruan, W. Haicheng, M. Sin, G. Noguere, D. Bernard, R. Jacqmin, O. Bouland, C. De Saint Jean, V. Pronyaev, A. Ignatyuk, K. Yokoyama, M. Ishikawa, T. Fukahori, N. Iwamoto, O. Iwamoto, S. Kunieda, C. Lubitz, M. Salvatores, G. Palmiotti, I. Kodeli, B. Kiedrowski, D. Roubtsov, I. Thompson, S. Quaglioni, H. Kim, Y. Lee, U. Fischer, S. Simakov, M. Dunn, K. Guber, J. Márquez Damián, F. Cantargi, I. Sirakov, N. Otuka, A. Daskalakis, B. McDermott, S. van der Marck, [CIELO Collaboration Summary Results: International Evaluations of Neutron Reactions on Uranium, Plutonium, Iron, Oxygen and Hydrogen](#), *Nuclear Data Sheets* 148 (2018) 189–213. doi:[10.1016/j.nds.2018.02.003](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S009037521830022X>
- [52] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, T. Kaltiaisenaho, [The Serpent Monte Carlo code: Status, development and applications in 2013](#), *Annals of Nuclear Energy* 82 (2015) 142–150. doi:[10.1016/j.anucene.2014.08.024](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454914004095>
- [53] T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. Cox, J. Durkee, J. Elson, M. Fensin, R. Forster, J. Hendricks, H. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, T. Zukaitis, [Features of MCNP6](#), *Annals of Nuclear Energy* 87 (2016) 772–783. doi:

- 10.1016/j.anucene.2015.02.020.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454915000900>
- [54] P. K. Romano, B. Forget, [The OpenMC Monte Carlo particle transport code](#), *Annals of Nuclear Energy* 51 (2013) 274–281. doi:10.1016/j.anucene.2012.06.040.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454912003283>
- [55] R. MacFarlane, A. Kahler, [Methods for Processing ENDF/B-VII with NJOY](#), *Nuclear Data Sheets* 111 (12) (2010) 2739–2890. doi:10.1016/j.nds.2010.11.001.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0090375210001006>
- [56] A. Yamamoto, N. Sugimura, [Improvement on multi-group scattering matrix in thermal energy range generated by NJOY](#), *Annals of Nuclear Energy* 33 (6) (2006) 555–559. doi:10.1016/j.anucene.2006.01.005.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454906000089>
- [57] K. S. Smith, [Nodal diffusion methods and lattice physics data in LWR analyses: Understanding numerous subtle details](#), *Progress in Nuclear Energy* 101 (2017) 360–369. doi:10.1016/j.pnucene.2017.06.013.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197017301609>
- [58] B. Lindley, J. Hosking, P. Smith, D. Powney, B. Tollit, T. Newton, R. Perry, T. Ware, P. Smith, [Current status of the reactor physics code WIMS and recent developments](#), *Annals of Nuclear Energy* 102 (2017) 148–157. doi:10.1016/j.anucene.2016.09.013.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454916303929>
- [59] M. Pusa, [Incorporating sensitivity and uncertainty analysis to a lattice physics code with application to CASMO-4](#), *Annals of Nuclear Energy* 40 (1) (2012) 153–162. doi:10.1016/j.anucene.2011.10.013.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454911004117>
- [60] B. D. Ivanov, K. N. Ivanov, R. J. Stamm’ler, [Helios, current coupling collision probability method, applied for solving the NEA C5G7 MOX benchmark](#), *Progress in Nuclear Energy* 45 (2-4) (2004) 119–124. doi:10.1016/j.pnucene.2004.09.004.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197003800136>
- [61] E. Varin, R. Roy, R. Baril, G. Hotte, [CANDU-6 operation post-simulations using the reactor physics codes DRAGON/DONJON](#), *Annals of Nuclear Energy* 31 (18) (2004) 2139–2155. doi:10.1016/j.anucene.2004.06.004.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454904001203>
- [62] S. M. Bowman, [SCALE 6: Comprehensive Nuclear Safety Analysis Code System](#), *Nuclear Technology* 174 (2) (2011) 126–148. doi:10.13182/NT10-163.
URL <https://www.tandfonline.com/doi/full/10.13182/NT10-163>

- [63] R. Sanchez, J. Mondot, Ž. Stankovski, A. Cossic, I. Zmijarevic, [APOLLO II: A User-Oriented, Portable, Modular Code for Multigroup Transport Assembly Calculations](#), Nuclear Science and Engineering 100 (3) (1988) 352–362. doi:10.13182/NSE88-3. URL <https://www.tandfonline.com/doi/full/10.13182/NSE88-3>
- [64] S. Choi, W. Kim, J. Choe, W. Lee, H. Kim, B. Ebiwonjumi, E. Jeong, K. Kim, D. Yun, H. Lee, D. Lee, [Development of high-fidelity neutron transport code STREAM](#), Computer Physics Communications 264 (2021) 107915. doi:10.1016/j.cpc.2021.107915. URL <https://linkinghub.elsevier.com/retrieve/pii/S0010465521000515>
- [65] D. Knott, E. Wehlage, [Description of the LANCER02 Lattice Physics Code for Single-Assembly and Multibundle Analysis](#), Nuclear Science and Engineering 155 (3) (2007) 331–354. doi:10.13182/NSE155-331. URL <https://www.tandfonline.com/doi/full/10.13182/NSE155-331>
- [66] R. Lawrence, [Progress in nodal methods for the solution of the neutron diffusion and transport equations](#), Progress in Nuclear Energy 17 (3) (1986) 271–301. doi:10.1016/0149-1970(86)90034-X. URL <https://linkinghub.elsevier.com/retrieve/pii/014919708690034X>
- [67] D. Cuervo, M. Avramova, K. Ivanov, R. Miró, [Evaluation and enhancement of COBRA-TF efficiency for LWR calculations](#), Annals of Nuclear Energy 33 (9) (2006) 837–847. doi:10.1016/j.anucene.2006.03.011. URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454906000648>
- [68] A. Moorthi, A. Kumar Sharma, K. Velusamy, [A review of sub-channel thermal hydraulic codes for nuclear reactor core and future directions](#), Nuclear Engineering and Design 332 (2018) 329–344. doi:10.1016/j.nucengdes.2018.03.012. URL <https://linkinghub.elsevier.com/retrieve/pii/S0029549318302681>
- [69] D. Aumiller, E. Tomlinson, R. Bauer, [A coupled RELAP5-3D/CFD methodology with a proof-of-principle calculation](#), Nuclear Engineering and Design 205 (1-2) (2001) 83–90. doi:10.1016/S0029-5493(00)00370-8. URL <https://linkinghub.elsevier.com/retrieve/pii/S0029549300003708>
- [70] K. Zhang, A. C. Muñoz, V. H. Sanchez-Espinoza, [Development and verification of the coupled thermal-hydraulic code – TRACE/SCF based on the ICoCo interface and the SA-LOME platform](#), Annals of Nuclear Energy 155 (2021) 108169. doi:10.1016/j.anucene.2021.108169. URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454921000451>
- [71] D. Gaston, C. Newman, G. Hansen, D. Lebrun-Grandié, [Moose: A parallel computational framework for coupled systems of nonlinear equations](#), Nuclear Engineering and Design 239 (10) (2009) 1768–1778. doi:<https://doi.org/10.1016/j.nucengdes.2009>.

05.021.

URL <https://www.sciencedirect.com/science/article/pii/S0029549309002635>

- [72] R. Schmidt, K. Belcourt, R. Hooper, R. Pawlowski, K. Clarno, S. Simunovic, S. Slattery, J. Turner, S. Palmtag, [An approach for coupled-code multiphysics core simulations from a common input](#), Annals of Nuclear Energy 84 (2015) 140–152. doi:10.1016/j.anucene.2014.11.015.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454914005957>
- [73] S. P. Hamilton, T. M. Evans, K. E. Royston, E. D. Biondo, [Domain decomposition in the GPU-accelerated Shift Monte Carlo code](#), Annals of Nuclear Energy 166 (2022) 108687. doi:10.1016/j.anucene.2021.108687.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454921005636>
- [74] K. S. Kim, M. L. Williams, D. Wiarda, K. T. Clarno, [Development of the multigroup cross section library for the CASL neutronics simulator MPACT: Method and procedure](#), Annals of Nuclear Energy 133 (2019) 46–58. doi:10.1016/j.anucene.2019.05.010.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454919302531>
- [75] Y. Wang, M. DeHart, V. Laboure, S. Schunert, J. Ortensi, F. Gleicher, [Multischeme transport with upwinding for rattlesnake code](#), Tech. Rep. INL/EXT-17-44238, Idaho National Lab (Jan 2018). doi:10.2172/1467003.
URL <https://www.osti.gov/biblio/1467003>
- [76] Z. Liu, H. Wu, L. Cao, Q. Chen, Y. Li, [A new three-dimensional method of characteristics for the neutron transport calculation](#), Annals of Nuclear Energy 38 (2-3) (2011) 447–454. doi:10.1016/j.anucene.2010.09.021.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454910003427>
- [77] S. Choi, D. Lee, [Three-dimensional method of characteristics/diamond-difference transport analysis method in STREAM for whole-core neutron transport calculation](#), Computer Physics Communications 260 (2021) 107332. doi:10.1016/j.cpc.2020.107332.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0010465520301430>
- [78] J.-Y. Cho, K.-S. Kim, C.-C. Lee, S.-Q. Zee, H.-G. Joo, [Axial \$SP_N\$ and Radial MOC Coupled Whole Core Transport Calculation](#), Journal of Nuclear Science and Technology 44 (9) (2007) 1156–1171. doi:10.1080/18811248.2007.9711359.
URL <http://www.tandfonline.com/doi/abs/10.1080/18811248.2007.9711359>
- [79] M. Eklund, M. Alamaniotis, H. Hernandez, T. Jevremovic, [Method of characteristics – A review with applications to science and nuclear engineering computation](#), Progress in Nuclear Energy 85 (2015) 548–567. doi:10.1016/j.pnucene.2015.05.002.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197015001080>

- [80] P. G. Ciarlet, [The Finite Element Method for Elliptic Problems](#), Society for Industrial and Applied Mathematics, 2002. doi:[10.1137/1.9780898719208](#).
URL <http://epubs.siam.org/doi/book/10.1137/1.9780898719208>
- [81] W. H. Reed, T. R. Hill, [Triangular mesh methods for the neutron transport equation: Report LA-UR-73-479](#), Tech. rep., Los Alamos Scientific Laboratory (1973).
URL <https://www.osti.gov/biblio/4491151-triangular-mesh-methods-neutron-transport->
- [82] P. Lasaint, P. Raviart, [On a Finite Element Method for Solving the Neutron Transport Equation](#), in: *Mathematical Aspects of Finite Elements in Partial Differential Equations*, Elsevier, 1974, pp. 89–123. doi:[10.1016/B978-0-12-208350-1.50008-X](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/B978012208350150008X>
- [83] L. A. Semenza, E. E. Lewis, E. C. Rossow, [The Application of the Finite Element Method to the Multigroup Neutron Diffusion Equation](#), *Nuclear Science and Engineering* 47 (3) (1972) 302–310. doi:[10.13182/NSE72-A22416](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE72-A22416>
- [84] C. M. Kang, K. F. Hansen, [Finite Element Methods for Reactor Analysis](#), *Nuclear Science and Engineering* 51 (4) (1973) 456–495. doi:[10.13182/NSE73-A23278](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE73-A23278>
- [85] T. S. Haut, B. S. Southworth, P. G. Maginot, V. Z. Tomov, [DSA Preconditioning for DG discretizations of \$S_N\$ transport and High-Order curved meshes](#) (Oct 2018). arXiv:[1810.11082](#).
URL <http://arxiv.org/abs/1810.11082>
- [86] T. S. Haut, P. G. Maginot, V. Z. Tomov, B. S. Southworth, T. A. Brunner, T. S. Bailey, [An Efficient Sweep-Based Solver for the \$S_N\$ Equations on High-Order Meshes](#), *Nuclear Science and Engineering* 193 (7) (2019) 746–759. doi:[10.1080/00295639.2018.1562778](#).
URL <https://www.tandfonline.com/doi/full/10.1080/00295639.2018.1562778>
- [87] L. Piegl, W. Tiller, [The NURBS Book](#), Monographs in Visual Communications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. doi:[10.1007/978-3-642-97385-7](#).
URL <http://link.springer.com/10.1007/978-3-642-97385-7>
- [88] R. Sevilla, S. Fernández-Méndez, A. Huerta, [NURBS-enhanced finite element method \(NEFEM\)](#), *International Journal for Numerical Methods in Engineering* 76 (1) (2008) 56–83. doi:[10.1002/nme.2311](#).
URL <https://onlinelibrary.wiley.com/doi/10.1002/nme.2311>
- [89] T. Hughes, J. Cottrell, Y. Bazilevs, [Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement](#), *Computer Methods in Applied Mechanics and Engineering* 194 (39–41) (2005) 4135–4195. doi:[10.1016/j.cma.2004.10.008](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782504005171>

- [90] L. B. da Veiga, F. Brezzi, A. Cangiani, G. Manzizni, L. D. Marini, A. Russo, Basic Principles of Virtual Element Methods, Mathematical Models and Methods in Applied Sciences 23 (01) (2013) 199–214. doi:[10.1142/s0218202512500492](https://doi.org/10.1142/s0218202512500492).
- [91] J. Ferguson, J. Kópházi, M. D. Eaton, Polygonal Virtual Element Spatial Discretisation Methods for the Neutron Diffusion Equation With Applications in Nuclear Reactor Physics, in: Volume 3: Nuclear Fuel and Material, Reactor Physics, and Transport Theory, American Society of Mechanical Engineers, 2018. doi:[10.1115/ICONE26-81317](https://doi.org/10.1115/ICONE26-81317).
URL <https://asmedigitalcollection.asme.org/ICONE/proceedings/ICONE26/51456/London,England/272707>
- [92] J. Ferguson, J. Kópházi, M. Eaton, Virtual element methods for the spatial discretisation of the multigroup neutron diffusion equation on polygonal meshes with applications to nuclear reactor physics, Annals of Nuclear Energy 151 (2021) 107884. doi:[10.1016/j.anucene.2020.107884](https://doi.org/10.1016/j.anucene.2020.107884).
URL <https://linkinghub.elsevier.com/retrieve/pii/S030645492030582X>
- [93] J. A. Ferguson, J. Kópházi, M. D. Eaton, NURBS Enhanced Virtual Element Methods for the Spatial Discretization of the Multigroup Neutron Diffusion Equation on Curvilinear Polygonal Meshes, Journal of Computational and Theoretical Transport 51 (4) (2022) 145–204. doi:[10.1080/23324309.2022.2103150](https://doi.org/10.1080/23324309.2022.2103150).
URL <https://www.tandfonline.com/doi/full/10.1080/23324309.2022.2103150>
- [94] L. Ornstein, G. Uhlenbeck, Some kinetic problems regarding the motion of neutrons through paraffine, Physica 4 (6) (1937) 478–486. doi:[10.1016/S0031-8914\(37\)80080-4](https://doi.org/10.1016/S0031-8914(37)80080-4).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0031891437800804>
- [95] E. Amaldi, U. Fano, L. V. Spencer, M. J. Berger, Neutrons and Related Gamma Ray Problems / Neutronen und Verwandte Gammastrahlprobleme, Vol. 8 / 38 / 2 of Encyclopedia of Physics / Handbuch der Physik, Springer Berlin Heidelberg, Berlin, Heidelberg, 1959. doi:[10.1007/978-3-642-45920-7](https://doi.org/10.1007/978-3-642-45920-7).
URL <http://link.springer.com/10.1007/978-3-642-45920-7>
- [96] L. Boltzmann, Further Studies on the Thermal Equilibrium of Gas Molecules, 2003, pp. 262–349. doi:[10.1142/9781848161337_0015](https://doi.org/10.1142/9781848161337_0015).
URL http://www.worldscientific.com/doi/abs/10.1142/9781848161337_0015
- [97] W. F. Miller, E. E. Lewis, Computational Methods of Neutron Transport, 1st Edition, American Nuclear Society, 1993.
URL <https://www.ans.org/store/item-350016/>
- [98] J. J. Duderstadt, W. R. Martin, Transport Theory, Wiley-Interscience, New York, New York, USA, 1979.
- [99] R. K. Osborn, S. Yip, The foundations of neutron transport theory, Monograph series on nuclear science and technology, Gordon and Breach, New York, 1966.

- [100] L. Pál, [On the theory of stochastic processes in nuclear reactors](#), *Il Nuovo Cimento* 7 (S1) (1958) 25–42. doi:10.1007/BF02725086.
URL <http://link.springer.com/10.1007/BF02725086>
- [101] G. I. Bell, [On the Stochastic Theory of Neutron Transport](#), *Nuclear Science and Engineering* 21 (3) (1965) 390–401. doi:10.13182/NSE65-1.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE65-1>
- [102] J. J. Duderstadt, L. J. Hamilton, *Nuclear Reactor Analysis*, John Wiley and Sons, Inc., 1976.
- [103] A. Hebert, *Applied Reactor Physics*, 2nd Edition, Presses internationales Polytechnique, 2016.
- [104] I. Pázsit, [A Simple Derivation of the Neutron Transport Equation](#), *Nuclear Science and Engineering* 112 (4) (1992) 369–374. doi:10.13182/NSE92-A23985.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE92-A23985>
- [105] A. Buchan, C. Pain, M. Eaton, R. Smedley-Stevenson, A. Goddard, [Linear and quadratic octahedral wavelets on the sphere for angular discretisations of the Boltzmann transport equation](#), *Annals of Nuclear Energy* 32 (11) (2005) 1224–1273. doi:10.1016/j.anucene.2005.01.005.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454905000356>
- [106] S. Dargaville, M. Goffin, A. Buchan, C. Pain, R. Smedley-Stevenson, P. Smith, G. Gorman, [Solving the Boltzmann transport equation with multigrid and adaptive space/angle discretisations](#), *Annals of Nuclear Energy* 86 (2015) 99–107. doi:10.1016/j.anucene.2015.02.014.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454915000717>
- [107] S. Dargaville, A. Buchan, R. Smedley-Stevenson, P. Smith, C. Pain, [Scalable angular adaptivity for Boltzmann transport](#), *Journal of Computational Physics* 406 (2020) 109124. doi:10.1016/j.jcp.2019.109124.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999119308290>
- [108] W. R. Martin, J. J. Duderstadt, [Finite Element Solutions of the Neutron Transport Equation with Applications to Strong Heterogeneities](#), *Nuclear Science and Engineering* 62 (3) (1977) 371–390. doi:10.13182/NSE77-A26979.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE77-A26979>
- [109] W. R. Martin, C. E. Yehner, L. Lorence, J. J. Duderstadt, [Phase-space finite element methods applied to the first-order form of the transport equation](#), *Annals of Nuclear Energy* 8 (11-12) (1981) 633–646. doi:10.1016/0306-4549(81)90131-6.
URL <https://linkinghub.elsevier.com/retrieve/pii/0306454981901316>

- [110] G. Coppa, G. Lapenta, P. Ravetto, [Angular finite element techniques in neutron transport](#), Annals of Nuclear Energy 17 (7) (1990) 363–378. doi:10.1016/0306-4549(90)90074-N.
URL <https://linkinghub.elsevier.com/retrieve/pii/030645499090074N>
- [111] J. Kópházi, D. Lathouwers, [A space-angle DGFEM approach for the Boltzmann radiation transport equation with local angular refinement](#), Journal of Computational Physics 297 (2015) 637–668. doi:10.1016/j.jcp.2015.05.031.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999115003654>
- [112] J. H. Jeans, [The Equations of Radiative Transfer of Energy](#), Monthly Notices of the Royal Astronomical Society 78 (1) (1917) 28–36. doi:10.1093/mnras/78.1.28.
URL <https://academic.oup.com/mnras/article-lookup/doi/10.1093/mnras/78.1.28>
- [113] J. C. Mark, The spherical harmonics method, part i. Technical Report MT 92, Tech. rep., National Research Council of Canada (1944).
- [114] J. C. Mark, The spherical harmonics method, part ii. Technical Report MT 97, Tech. rep., National Research Council of Canada (1944).
- [115] B. Davison, J. B. Sykes, Neutron Transport Theory, Oxford University Press, Oxford, 1957.
- [116] S. Chandrasekhar, Radiative Transfer, Oxford University Press, Oxford, 1950.
- [117] B. Carlson, [Solution of the transport equation by \$s_n\$ approximations](#), Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (Feb 1955). doi:10.2172/4376236.
URL <https://www.osti.gov/servlets/purl/4376236/>
- [118] C. Lee, [THE DISCRETE \$S_n\$ APPROXIMATION TO TRANSPORT THEORY](#), Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (Jun 1961). doi:10.2172/4783088.
URL <https://www.osti.gov/servlets/purl/4783088/>
- [119] K. D. Lathrop, B. G. Carlson, [Discrete ordinates angular quadrature of the neutron transport equation](#), Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (Sep 1964). doi:10.2172/4666281.
URL <http://www.osti.gov/servlets/purl/4666281/>
- [120] B. G. Carlson, K. D. Lathrop, Transport Theory: The Method of Discrete Ordinates, Tech. rep., Los Alamos National Laboratory, Los Alamos, NM (United States) (1965).
- [121] J. Jenal, P. J. Erickson, W. A. Rhoades, M. Williams, The Generation of a Computer Library for Discrete Ordinates Quadrature Sets, Tech. rep., Oak Ridge National Laboratory, Oak Ridge, Tennessee, US (1977).

- [122] J. E. Morel, T. A. Wareing, R. B. Lowrie, D. K. Parsons, [Analysis of Ray-Effect Mitigation Techniques](#), Nuclear Science and Engineering 144 (1) (2003) 1–22. doi:[10.13182/NSE01-48](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE01-48>
- [123] K. D. Lathrop, [Remedies for Ray Effects](#), Nuclear Science and Engineering 45 (3) (1971) 255–268. doi:[10.13182/NSE45-03-255](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE45-03-255>
- [124] J. Jung, H. Chijiwa, K. Kobayashi, H. Nishihara, [Discrete Ordinate Neutron Transport Equation Equivalent to P L Approximation](#), Nuclear Science and Engineering 49 (1) (1972) 1–9. doi:[10.13182/NSE49-01-01](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE49-01-01>
- [125] W. H. Reed, [Spherical Harmonic Solutions of the Neutron Transport Equation from Discrete Ordinate Codes](#), Nuclear Science and Engineering 49 (1) (1972) 10–19. doi:[10.13182/NSE72-A22523](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE72-A22523>
- [126] E. W. Larsen, J. B. Keller, [Asymptotic solution of neutron transport problems for small mean free paths](#), Journal of Mathematical Physics 15 (1) (1974) 75–81. doi:[10.1063/1.1666510](#).
URL <http://aip.scitation.org/doi/10.1063/1.1666510>
- [127] E. W. Larsen, [The Asymptotic Diffusion Limit of Discretized Transport Problems](#), Nuclear Science and Engineering 112 (4) (1992) 336–346. doi:[10.13182/NSE92-A23982](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE92-A23982>
- [128] Y. Kato, T. Takeda, S. Takeda, [A Coarse-Mesh Correction of the Finite Difference Method for Neutron Diffusion Calculations](#), Nuclear Science and Engineering 61 (2) (1976) 127–141. doi:[10.13182/NSE76-A27347](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE76-A27347>
- [129] M. L. Adams, E. W. Larsen, [Fast iterative methods for discrete-ordinates particle transport calculations](#), Progress in Nuclear Energy 40 (1) (2002) 3–159. doi:[10.1016/S0149-1970\(01\)00023-3](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197001000233>
- [130] A. C. Kahler, R. MacFarlane, The NJOY Nuclear Data Processing System, Version 2012, Report LA-UR-12-27079, Tech. rep., Los Alamos National Laboratory, Los Alamos, NM (United States) (2012).
- [131] N. Soppera, M. Bossant, E. Dupont, [JANIS 4: An Improved Version of the NEA Java-based Nuclear Data Information System](#), Nuclear Data Sheets 120 (2014) 294–296. doi:[10.1016/j.nds.2014.07.071](#).

URL https://www.oecd-neo.org/jcms/pl_39910/janishttps://linkinghub.elsevier.com/retrieve/pii/S0090375214005237

- [132] H. Hurwitz Jr, R. Ehrlich, [Multigroup methods for neutron diffusion problems](#) (Oct 1954). URL <https://www.osti.gov/biblio/4418873>
- [133] M. Chadwick, M. Herman, P. Obložinský, M. Dunn, Y. Danon, A. Kahler, D. Smith, B. Pritychenko, G. Arbanas, R. Arcilla, R. Brewer, D. Brown, R. Capote, A. Carlson, Y. Cho, H. Derrien, K. Guber, G. Hale, S. Hoblit, S. Holloway, T. Johnson, T. Kawano, B. Kiedrowski, H. Kim, S. Kunieda, N. Larson, L. Leal, J. Lestone, R. Little, E. McCutchan, R. MacFarlane, M. MacInnes, C. Mattoon, R. McKnight, S. Mughabghab, G. Nobre, G. Palmiotti, A. Palumbo, M. Pigni, V. Pronyaev, R. Sayer, A. Sonzogni, N. Summers, P. Talou, I. Thompson, A. Trkov, R. Vogt, S. van der Marck, A. Wallner, M. White, D. Wiarda, P. Young, [ENDF/B-VII.1 Nuclear Data for Science and Technology: Cross Sections, Covariances, Fission Product Yields and Decay Data](#), Nuclear Data Sheets 112 (12) (2011) 2887–2996. doi:10.1016/j.nds.2011.11.002. URL <https://linkinghub.elsevier.com/retrieve/pii/S009037521100113X>
- [134] A. J. M. Plompen, O. Cabellos, C. De Saint Jean, M. Fleming, A. Algora, M. Angelone, P. Archier, E. Bauge, O. Bersillon, A. Blokhin, F. Cantargi, A. Chebboubi, C. Diez, H. Duarte, E. Dupont, J. Dyrda, B. Erasmus, L. Fiorito, U. Fischer, D. Flammini, D. Foligno, M. R. Gilbert, J. R. Granada, W. Haeck, F.-J. Hambsch, P. Helgesson, S. Hilaire, I. Hill, M. Hursin, R. Ichou, R. Jacqmin, B. Jansky, C. Jouanne, M. A. Kellett, D. H. Kim, H. I. Kim, I. Kodeli, A. J. Koning, A. Y. Konobeyev, S. Kopecky, B. Kos, A. Krása, L. C. Leal, N. Leclaire, P. Leconte, Y. O. Lee, H. Leeb, O. Litaize, M. Majerle, J. I. Márquez Damián, F. Michel-Sendis, R. W. Mills, B. Morillon, G. Noguère, M. Pecchia, S. Pelloni, P. Pereslavytsev, R. J. Perry, D. Rochman, A. Röhrmoser, P. Romain, P. Romojaro, D. Roubtsov, P. Sauvan, P. Schillebeeckx, K. H. Schmidt, O. Serot, S. Simakov, I. Sirakov, H. Sjöstrand, A. Stankovskiy, J. C. Sublet, P. Tamagno, A. Trkov, S. van der Marck, F. Álvarez-Velarde, R. Villari, T. C. Ware, K. Yokoyama, G. Žerovnik, [The joint evaluated fission and fusion nuclear data library, JEFF-3.3](#), The European Physical Journal A 56 (7) (2020) 181. doi:10.1140/epja/s10050-020-00141-9. URL <https://link.springer.com/10.1140/epja/s10050-020-00141-9>
- [135] D. G. Cacuci (Ed.), [Handbook of Nuclear Engineering](#), Springer US, Boston, MA, 2010. doi:10.1007/978-0-387-98149-9. URL <http://link.springer.com/10.1007/978-0-387-98149-9>
- [136] W. L. Briggs, V. E. Henson, S. F. McCormick, [A Multigrid Tutorial, Second Edition](#), Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719505. URL <http://epubs.siam.org/doi/book/10.1137/1.9780898719505>
- [137] K. Lathrop, [Spatial differencing of the transport equation: Positivity vs. accuracy](#), Journal of Computational Physics 4 (4) (1969) 475–498. doi:10.1016/0021-9991(69)

90015-1.

URL <https://linkinghub.elsevier.com/retrieve/pii/0021999169900151>

- [138] W. A. Rhoades, W. W. Engle, Jr, [New weighted-difference formulation for discrete-ordinates calculations](#) (Jan 1977).
URL <https://www.osti.gov/biblio/5365860>
- [139] A. Ern, J.-L. Guermond, [Theory and Practice of Finite Elements](#), Vol. 159 of Applied Mathematical Sciences, Springer New York, 2004. doi:10.1007/978-1-4757-4355-5.
URL <http://link.springer.com/10.1007/978-1-4757-4355-5>
- [140] D. Braess, [Finite Elements: Theory, Fast Solvers, and Applications in Elasticity Theory](#), 3rd Edition, Cambridge University Press, Cambridge, 2007.
- [141] S. C. Brenner, L. R. Scott, [The Mathematical Theory of Finite Element Methods](#), 3rd Edition, Vol. 15 of Texts in Applied Mathematics, Springer New York, 2008. doi:10.1007/978-0-387-75934-0.
URL <http://link.springer.com/10.1007/978-0-387-75934-0>
- [142] H. G. Kaper, G. K. Leaf, A. J. Lindeman, [A Timing Comparison Study for Some High Order Finite Element Approximation Procedures and a Low Order Finite Difference Approximation Procedure for the Numerical Solution of the Multigroup Neutron Diffusion Equation](#), Nuclear Science and Engineering 49 (1) (1972) 27–48. doi:10.13182/NSE72-A22525.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE72-A22525>
- [143] W. F. Miller, E. E. Lewis, E. C. Rossow, [The Application of Phase-Space Finite Elements to the One-Dimensional Neutron Transport Equation](#), Nuclear Science and Engineering 51 (2) (1973) 148–156. doi:10.13182/NSE73-A26590.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE73-A26590>
- [144] T. A. Wareing, J. M. McGhee, J. E. Morel, [Attila: A three-dimensional, unstructured tetrahedral mesh discrete ordinates transport code](#), Transactions of the American Nuclear Society 75 (Dec 1996).
URL <https://www.osti.gov/biblio/426394>
- [145] Y. Wang, W. Bangerth, J. Ragusa, [Three-dimensional h-adaptivity for the multigroup neutron diffusion equations](#), Progress in Nuclear Energy 51 (3) (2009) 543–555. doi:10.1016/j.pnucene.2008.11.005.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197008001406>
- [146] Y. Wang, J. Ragusa, [Application of hp Adaptivity to the Multigroup Diffusion Equations](#), Nuclear Science and Engineering 161 (1) (2009) 22–48. doi:10.13182/NSE161-22.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE161-22>

- [147] J. C. Ragusa, [Discontinuous finite element solution of the radiation diffusion equation on arbitrary polygonal meshes and locally adapted quadrilateral grids](#), Journal of Computational Physics 280 (2015) 195–213. doi:[10.1016/j.jcp.2014.09.013](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999114006494>
- [148] P. Ciarlet, L. Giret, E. Jamelot, F. Kpadonou, [Numerical analysis of the mixed finite element method for the neutron diffusion eigenproblem with heterogeneous coefficients](#), ESAIM: Mathematical Modelling and Numerical Analysis 52 (5) (2018) 2003–2035. doi:[10.1051/m2an/2018011](#).
URL <https://www.esaim-m2an.org/10.1051/m2an/2018011>
- [149] D. Lathouwers, [Goal-oriented spatial adaptivity for the SN equations on unstructured triangular meshes](#), Annals of Nuclear Energy 38 (6) (2011) 1373–1381. doi:[10.1016/j.anucene.2011.01.038](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454911000594>
- [150] D. Lathouwers, [Spatially adaptive eigenvalue estimation for the SN equations on unstructured triangular meshes](#), Annals of Nuclear Energy 38 (9) (2011) 1867–1876. doi:[10.1016/j.anucene.2011.05.013](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454911002039>
- [151] E. Hall, P. Houston, S. Murphy, [hp-Adaptive Discontinuous Galerkin Methods for Neutron Transport Criticality Problems](#), SIAM Journal on Scientific Computing 39 (5) (2017) B916–B942. doi:[10.1137/16M1079944](#).
URL <https://epubs.siam.org/doi/10.1137/16M1079944>
- [152] M. W. Hackemack, J. C. Ragusa, [Quadratic serendipity discontinuous finite element discretization for S transport on arbitrary polygonal grids](#), Journal of Computational Physics 374 (2018) 188–212. doi:[10.1016/j.jcp.2018.05.032](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999118303401>
- [153] R. Sevilla, S. Fernández-Méndez, A. Huerta, [Comparison of high-order curved finite elements](#), International Journal for Numerical Methods in Engineering 87 (8) (2011) 719–734. doi:[10.1002/nme.3129](#).
URL <https://onlinelibrary.wiley.com/doi/10.1002/nme.3129>
- [154] R. Sevilla, S. Fernández-Méndez, [Numerical integration over 2D NURBS-shaped domains with applications to NURBS-enhanced FEM](#), Finite Elements in Analysis and Design 47 (10) (2011) 1209–1220. doi:[10.1016/j.finel.2011.05.011](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0168874X1100117X>
- [155] R. Sevilla, S. Fernández-Méndez, A. Huerta, [3D NURBS-enhanced finite element method \(NEFEM\)](#), International Journal for Numerical Methods in Engineering 88 (2) (2011) 103–125. doi:[10.1002/nme.3164](#).
URL <https://onlinelibrary.wiley.com/doi/10.1002/nme.3164>

- [156] R. Sevilla, L. Rees, O. Hassan, [The generation of triangular meshes for NURBS-enhanced FEM](#), *International Journal for Numerical Methods in Engineering* 108 (8) (2016) 941–968. doi:10.1002/nme.5247.
URL <https://onlinelibrary.wiley.com/doi/10.1002/nme.5247>
- [157] C. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. Kirby, S. Sherwin, [Nektar++: An open-source spectral/hp element framework](#), *Computer Physics Communications* 192 (2015) 205–219. doi:10.1016/j.cpc.2015.02.008.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0010465515000533>
- [158] D. Moxey, C. D. Cantwell, Y. Bao, A. Cassinelli, G. Castiglioni, S. Chun, E. Juda, E. Kazemi, K. Lackhove, J. Marcon, G. Mengaldo, D. Serson, M. Turner, H. Xu, J. Peiró, R. M. Kirby, S. J. Sherwin, [Nektar++: Enhancing the capability and application of high-fidelity spectral/hp element methods](#), *Computer Physics Communications* 249 (2020) 107110. doi:10.1016/j.cpc.2019.107110.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0010465519304175>
- [159] M. Turner, J. Peiró, D. Moxey, [Curvilinear mesh generation using a variational framework](#), *Computer-Aided Design* 103 (2018) 73–91. doi:10.1016/j.cad.2017.10.004.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0010448517301744>
- [160] J. Marcon, D. A. Kopriva, S. J. Sherwin, J. Peiró, [A high resolution PDE approach to quadrilateral mesh generation](#), *Journal of Computational Physics* 399 (2019) 108918. doi:10.1016/j.jcp.2019.108918.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999119306230>
- [161] Y. Bazilevs, L. Beirão Da Veiga, J. A. Cottrell, T. J. R. Hughes, G. Sangalli, [ISOGEOMETRIC ANALYSIS: APPROXIMATION, STABILITY AND ERROR ESTIMATES FOR h-REFINED MESHES](#), *Mathematical Models and Methods in Applied Sciences* 16 (07) (2006) 1031–1090. doi:10.1142/S0218202506001455.
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202506001455>
- [162] T. Hughes, A. Reali, G. Sangalli, [Efficient quadrature for NURBS-based isogeometric analysis](#), *Computer Methods in Applied Mechanics and Engineering* 199 (5-8) (2010) 301–313. doi:10.1016/j.cma.2008.12.004.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782508004295>
- [163] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, T. Hughes, S. Lipton, M. Scott, T. Sederberg, [Isogeometric analysis using T-splines](#), *Computer Methods in Applied Mechanics and Engineering* 199 (5-8) (2010) 229–263. doi:10.1016/j.cma.2009.02.036.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782509000875>

- [164] R. Schmidt, R. Wüchner, K.-U. Bletzinger, [Isogeometric analysis of trimmed NURBS geometries](#), Computer Methods in Applied Mechanics and Engineering 241-244 (2012) 93–111. doi:10.1016/j.cma.2012.05.021.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782512001843>
- [165] S. K. Hall, M. D. Eaton, M. M. R. Williams, The application of isogeometric analysis to the neutron diffusion equation for a pincell problem with an analytic benchmark, Annals of Nuclear Energy 49 (2012) 160–169. doi:10.1016/j.anucene.2012.05.030.
- [166] J. Welch, J. Kópházi, A. Owens, M. D. Eaton, [Isogeometric analysis for the multigroup neutron diffusion equation with applications in reactor physics](#), Annals of Nuclear Energy 101 (2017) 465–480. doi:10.1016/j.anucene.2016.11.015.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454916310362>
- [167] J. Welch, J. Kópházi, A. Owens, M. Eaton, [A geometry preserving, conservative, mesh-to-mesh isogeometric interpolation algorithm for spatial adaptivity of the multigroup, second-order even-parity form of the neutron transport equation](#), Journal of Computational Physics 347 (2017) 129–146. doi:10.1016/j.jcp.2017.06.015.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999117304606>
- [168] A. Owens, J. Welch, J. Kópházi, M. Eaton, [Discontinuous isogeometric analysis methods for the first-order form of the neutron transport equation with discrete ordinate \(\$S_N\$ \) angular discretisation](#), Journal of Computational Physics 315 (2016) 501–535. doi:10.1016/j.jcp.2016.03.060.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999116300250>
- [169] A. Owens, J. Welch, J. Kópházi, M. Eaton, [An adaptive, hanging-node, discontinuous isogeometric analysis method for the first-order form of the neutron transport equation with discrete ordinate \(\$S_N\$ \) Angular Discretisation](#), Computer Methods in Applied Mechanics and Engineering 318 (2017) 215–241. doi:10.1016/j.cma.2017.01.036.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782516304145>
- [170] A. Owens, J. Kópházi, J. Welch, M. Eaton, [Energy dependent mesh adaptivity of discontinuous isogeometric discrete ordinate methods with dual weighted residual error estimators](#), Journal of Computational Physics 335 (2017) 352–386. doi:10.1016/j.jcp.2017.01.035.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999117300517>
- [171] C. Latimer, J. Kópházi, M. D. Eaton, R. G. McClarren, [A geometry conforming isogeometric method for the self-adjoint angular flux \(SAAF\) form of the neutron transport equation with a discrete ordinate \(\$S_N\$ \) angular discretisation](#), Annals of Nuclear Energy 136 (2020) 107049. doi:10.1016/j.anucene.2019.107049.
- [172] C. Latimer, J. Kópházi, M. D. Eaton, R. G. McClarren, [A geometry conforming, isogeometric, weighted least squares \(WLS\) method for the neutron transport equation](#)

- with discrete ordinate (S_N) angular discretisation, Progress in Nuclear Energy 121 (2020) 103238. doi:10.1016/j.pnucene.2019.103238.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0149197019303439>
- [173] C. Latimer, J. Kópházi, M. Eaton, R. McClarren, Spatial adaptivity of the SAAF and Weighted Least Squares (WLS) forms of the neutron transport equation using constraint based, locally refined, isogeometric analysis (IGA) with dual weighted residual (DWR) error measures, Journal of Computational Physics 426 (2021) 109941. doi:10.1016/j.jcp.2020.109941.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999120307154>
- [174] E. Kreyszig, Introductory Functional Analysis with Applications, John Wiley & Sons, Ltd, 1989.
- [175] J. N. Reddy, Applied Functional Analysis and Variational Methods in Engineering, McGraw-Hill, New York, NY, 1986.
- [176] R. Adams, J. Fournier, Sobolev Spaces, Academic Press, 2003.
- [177] D. A. Di Pietro, A. Ern, Mathematical Aspects of Discontinuous Galerkin Methods, Vol. 69 of Mathématiques et Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-22980-0.
URL <http://link.springer.com/10.1007/978-3-642-22980-0>
- [178] A. Cangiani, E. H. Georgoulis, P. Houston, hp-Version discontinuous Galerkin methods on polygonal and polyhedral meshes, Mathematical Models and Methods in Applied Sciences 24 (10) (2014) 2009–2041. doi:10.1142/S0218202514500146.
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202514500146>
- [179] A. Cangiani, Z. Dong, E. H. Georgoulis, P. Houston, hp-Version Discontinuous Galerkin Methods on Polygonal and Polyhedral Meshes, SpringerBriefs in Mathematics, Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-67673-9.
URL <http://link.springer.com/10.1007/978-3-319-67673-9>
- [180] E. Wachspress, A Rational Finite Element Basis, Academic Press, 1975.
- [181] N. Sukumar, E. A. Malsch, Recent advances in the construction of polygonal finite element interpolants, Archives of Computational Methods in Engineering 13 (1) (2006) 129–163. doi:10.1007/BF02905933.
URL <https://link.springer.com/article/10.1007/BF02905933>
- [182] C. Talischi, A. Pereira, G. H. Paulino, I. F. Menezes, M. S. Carvalho, Polygonal finite elements for incompressible fluid flow, International Journal for Numerical Methods in Fluids 74 (2) (2014) 134–151. doi:10.1002/fld.3843.
URL <http://doi.wiley.com/10.1002/fld.3843>

- [183] N. Sukumar, A. Tabarraei, [Conforming polygonal finite elements](#), International Journal for Numerical Methods in Engineering 61 (12) (2004) 2045–2066. doi:10.1002/nme.1141.
URL <http://doi.wiley.com/10.1002/nme.1141>
- [184] I. Babuška, J. E. Osborn, [Generalized Finite Element Methods: Their Performance and Their Relation to Mixed Methods](#), SIAM Journal on Numerical Analysis 20 (3) (1983) 510–536. doi:10.1137/0720034.
URL <http://epubs.siam.org/doi/10.1137/0720034>
- [185] I. Babuška, U. Banerjee, [Stable Generalized Finite Element Method \(SGFEM\)](#), Computer Methods in Applied Mechanics and Engineering 201–204 (2012) 91–111. doi:10.1016/j.cma.2011.09.012.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782511003082>
- [186] T. Strouboulis, I. Babuška, K. Copps, [The design and analysis of the Generalized Finite Element Method](#), Computer Methods in Applied Mechanics and Engineering 181 (1–3) (2000) 43–69. doi:10.1016/S0045-7825(99)00072-9.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782599000729>
- [187] T.-P. Fries, T. Belytschko, [The extended/generalized finite element method: An overview of the method and its applications](#), International Journal for Numerical Methods in Engineering 84 (3) (2010) 253–304. doi:10.1002/nme.2914.
URL <https://onlinelibrary.wiley.com/doi/10.1002/nme.2914>
- [188] A. Tabarraei, N. Sukumar, [Extended finite element method on polygonal and quadtree meshes](#), Computer Methods in Applied Mechanics and Engineering 197 (5) (2008) 425–438. doi:10.1016/j.cma.2007.08.013.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782507003374>
- [189] L. B. da Veiga, K. Lipnikov, G. Manzini, The Mimetic Finite Difference Method for Elliptic Problems, Springer International Publishing, 2014. doi:10.1007/978-3-319-02663-3.
- [190] K. Lipnikov, G. Manzini, M. Shashkov, Mimetic finite difference method, Journal of Computational Physics 257 Part B (2014) 1163–1227. doi:10.1016/j.jcp.2013.07.031.
- [191] G. F. M. Jose E. Castillo, [Mimetic Discretization Methods](#), Taylor & Francis Ltd., 2013.
URL https://www.ebook.de/de/product/21182811/jose_e_castillo_guillermo_f_miranda_mimetic_discretization_methods.html
- [192] L. B. da Veiga, K. Lipnikov, G. Manzini, Arbitrary-Order Nodal Mimetic Discretizations of Elliptic Problems on Polygonal Meshes, SIAM Journal on Numerical Analysis 49 (5) (2011) 1737–1760. doi:10.1137/100807764.

- [193] L. B. da Veiga, G. Manzini, A Higher-Order Formulation of the Mimetic Finite Difference Method, *SIAM Journal on Scientific Computing* 31 (1) (2008) 732–760. doi:[10.1137/080717894](https://doi.org/10.1137/080717894).
- [194] F. Brezzi, L. D. Marini, [Virtual Element Methods for plate bending problems](#), *Computer Methods in Applied Mechanics and Engineering* 253 (2013) 455–462. doi:[10.1016/j.cma.2012.09.012](https://doi.org/10.1016/j.cma.2012.09.012).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782512002940>
- [195] L. da Veiga, G. Manzini, [A virtual element method with arbitrary regularity](#), *IMA Journal of Numerical Analysis* 34 (2) (2014) 759–781. doi:[10.1093/imanum/drt018](https://doi.org/10.1093/imanum/drt018).
URL <https://academic.oup.com/imajna/article-lookup/doi/10.1093/imanum/drt018>
- [196] G. Manzini, A. Cangiani, O. Sutton, [The Conforming Virtual Element Method for the convection-diffusion-reaction equation with variable coefficients.](#), Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (Oct 2014). doi:[10.2172/1159207](https://doi.org/10.2172/1159207).
URL <http://www.osti.gov/servlets/purl/1159207/>
- [197] L. B. da Veiga, F. Brezzi, L. D. Marini, A. Russo, Virtual Element Implementation for General Elliptic Equations, in: *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*, Springer International Publishing, 2016, pp. 39–71. doi:[10.1007/978-3-319-41640-3_2](https://doi.org/10.1007/978-3-319-41640-3_2).
- [198] B. Ahmad, A. Alsaedi, F. Brezzi, L. Marini, A. Russo, [Equivalent projectors for virtual element methods](#), *Computers & Mathematics with Applications* 66 (3) (2013) 376–391. doi:[10.1016/j.camwa.2013.05.015](https://doi.org/10.1016/j.camwa.2013.05.015).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0898122113003179>
- [199] L. Beirão da Veiga, F. Brezzi, L. Marini, A. Russo, [Serendipity Nodal VEM spaces](#), *Computers & Fluids* 141 (2016) 2–12. doi:[10.1016/j.compfluid.2016.02.015](https://doi.org/10.1016/j.compfluid.2016.02.015).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045793016300391>
- [200] A. Russo, [On the choice of the internal degrees of freedom for the nodal Virtual Element Method in two dimensions](#), *Computers & Mathematics with Applications* 72 (8) (2016) 1968–1976. doi:[10.1016/j.camwa.2016.03.016](https://doi.org/10.1016/j.camwa.2016.03.016).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0898122116301328>
- [201] P. F. Antonietti, G. Manzini, M. Verani, The fully nonconforming virtual element method for biharmonic problems, *Mathematical Models and Methods in Applied Sciences* 28 (02) (2017) 387–407. doi:[10.1142/s0218202518500100](https://doi.org/10.1142/s0218202518500100).
- [202] A. Cangiani, G. Manzini, O. J. Sutton, Conforming and nonconforming virtual element methods for elliptic problems, *IMA Journal of Numerical Analysis* 37 (3) (2016) 1317–1354. doi:[10.1093/imanum/drw036](https://doi.org/10.1093/imanum/drw036).

- [203] S. Berrone, A. Borio, G. Manzini, [SUPG stabilization for the nonconforming virtual element method for advection–diffusion–reaction equations](#), *Computer Methods in Applied Mechanics and Engineering* 340 (2018) 500–529. doi:[10.1016/j.cma.2018.05.027](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782517302918>
- [204] B. Ayuso de Dios, K. Lipnikov, G. Manzini, [The nonconforming virtual element method](#), *ESAIM: Mathematical Modelling and Numerical Analysis* 50 (3) (2016) 879–904. doi:[10.1051/m2an/2015090](#).
URL <http://www.esaim-m2an.org/10.1051/m2an/2015090>
- [205] X. Liu, J. Li, Z. Chen, A nonconforming virtual element method for the Stokes problem on general meshes, *Computer Methods in Applied Mechanics and Engineering* 320 (2017) 694–711. doi:[10.1016/j.cma.2017.03.027](#).
- [206] F. Dassi, L. Mascotto, [Exploring high-order three dimensional virtual elements: Bases and stabilizations](#), *Computers & Mathematics with Applications* 75 (9) (2018) 3379–3401. doi:[10.1016/j.camwa.2018.02.005](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0898122118300786>
- [207] L. Mascotto, Ill-conditioning in the virtual element method: stabilizations and bases, *Numerical Methods for Partial Differential Equations* 34 (4) (2018) 1258–1281. doi:[10.1002/num.22257](#).
- [208] S. Berrone, A. Borio, Orthogonal polynomials in badly shaped polygonal elements for the Virtual Element Method, *Finite Elements in Analysis and Design* 129 (2017) 14–31. doi:[10.1016/j.finel.2017.01.006](#).
- [209] S. C. Brenner, Q. Guan, L.-Y. Sung, [Some Estimates for Virtual Element Methods](#), *Computational Methods in Applied Mathematics* 17 (4) (2017) 553–574. doi:[10.1515/cmam-2017-0008](#).
URL <https://www.degruyter.com/view/journals/cmam/17/4/article-p553.xml>
- [210] S. C. Brenner, L.-Y. Sung, [Virtual element methods on meshes with small edges or faces](#), *Mathematical Models and Methods in Applied Sciences* 28 (07) (2018) 1291–1336. doi:[10.1142/S0218202518500355](#).
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202518500355>
- [211] L. B. da Veiga, C. Lovadina, A. Russo, Stability analysis for the virtual element method, *Mathematical Models and Methods in Applied Sciences* 27 (13) (2017) 2557–2594. doi:[10.1142/s021820251750052x](#).
- [212] F. Gardini, G. Vacca, Virtual element method for second-order elliptic eigenvalue problems, *IMA Journal of Numerical Analysis* 38 (4) (2017) 2026–2054. doi:[10.1093/imanum/drx063](#).

- [213] D. Mora, G. Rivera, R. Rodríguez, [A virtual element method for the Steklov eigenvalue problem](#), Mathematical Models and Methods in Applied Sciences 25 (08) (2015) 1421–1445. doi:[10.1142/S0218202515500372](#).
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202515500372>
- [214] D. Mora, G. Rivera, R. Rodríguez, A posteriori error estimates for a Virtual Element Method for the Steklov eigenvalue problem, Computers & Mathematics with Applications 74 (9) (2017) 2172–2190. doi:[10.1016/j.camwa.2017.05.016](#).
- [215] D. Mora, I. Velásquez, [A virtual element method for the transmission eigenvalue problem](#), Mathematical Models and Methods in Applied Sciences 28 (14) (2018) 2803–2831. doi:[10.1142/S0218202518500616](#).
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202518500616>
- [216] O. Čertík, F. Gardini, G. Manzini, G. Vacca, The virtual element method for eigenvalue problems with potential terms on polytopic meshes, Applications of Mathematics 63 (3) (2018) 333–365. doi:[10.21136/am.2018.0093-18](#).
- [217] F. Wang, H. Wei, Virtual element method for simplified friction problem, Applied Mathematics Letters 85 (2018) 125–131. doi:[10.1016/j.aml.2018.06.002](#).
- [218] P. Wriggers, W. T. Rust, B. D. Reddy, A virtual element method for contact, Computational Mechanics 58 (6) (2016) 1039–1050. doi:[10.1007/s00466-016-1331-x](#).
- [219] L. Beirão Da Veiga, F. Brezzi, F. Dassi, L. D. Marini, A. Russo, Virtual Element approximation of 2D magnetostatic problems, Computer Methods in Applied Mechanics and Engineering 327 (2017) 173–195. doi:[10.1016/j.cma.2017.08.013](#).
- [220] P. F. Antonietti, L. B. da Veiga, D. Mora, M. Verani, A Stream Virtual Element Formulation of the Stokes Problem on Polygonal Meshes, SIAM Journal on Numerical Analysis 52 (1) (2014) 386–404. doi:[10.1137/13091141x](#).
- [221] G. Vacca, Virtual Element Methods for hyperbolic problems on polygonal meshes, Computers & Mathematics with Applications 74 (5) (2017) 882–898. doi:[10.1016/j.camwa.2016.04.029](#).
- [222] G. Vacca, L. Beirão da Veiga, [Virtual element methods for parabolic problems on polygonal meshes](#), Numerical Methods for Partial Differential Equations 31 (6) (2015) 2110–2134. doi:[10.1002/num.21982](#).
URL <http://doi.wiley.com/10.1002/num.21982>
- [223] A. L. Gain, C. Talischi, G. H. Paulino, On the Virtual Element Method for three-dimensional linear elasticity problems on arbitrary polyhedral meshes, Computer Methods in Applied Mechanics and Engineering 282 (2014) 132–160. doi:[10.1016/j.cma.2014.05.005](#).

- [224] H. Chi, L. B. da Veiga, G. Paulino, [Some basic formulations of the virtual element method \(VEM\) for finite deformations](#), Computer Methods in Applied Mechanics and Engineering 318 (2017) 148–192. doi:[10.1016/j.cma.2016.12.020](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782516309094>
- [225] E. Artioli, S. De Miranda, C. Lovadina, L. Patruno, A stress/displacement Virtual Element method for plane elasticity problems, Computer Methods in Applied Mechanics and Engineering 325 (2017) 155–174. doi:[10.1016/j.cma.2017.06.036](#).
- [226] L. B. da Veiga, D. Mora, G. Rivera, Virtual elements for a shear-deflection formulation of Reissner–Mindlin plates, Mathematics of Computation 88 (315) (2018) 149–178. doi:[10.1090/mcom/3331](#).
- [227] E. Artioli, L. Beirão da Veiga, F. Dassi, [Curvilinear Virtual Elements for 2D solid mechanics applications](#), Computer Methods in Applied Mechanics and Engineering 359 (2020) 112667. doi:[10.1016/j.cma.2019.112667](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782519305523>
- [228] S. Bertoluzza, M. Pennacchio, D. Prada, [High order VEM on curved domains](#), Rendiconti Lincei - Matematica e Applicazioni 30 (2) (2019) 391–412. arXiv:[1811.04755](#), doi:[10.4171/RLM/853](#).
URL <https://www.ems-ph.org/doi/10.4171/RLM/853>
- [229] L. Beirão da Veiga, F. Brezzi, L. D. Marini, A. Russo, [Polynomial preserving virtual elements with curved edges](#), Mathematical Models and Methods in Applied Sciences 30 (08) (2020) 1555–1590. doi:[10.1142/S0218202520500311](#).
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202520500311>
- [230] L. Beirão da Veiga, F. Brezzi, L. D. Marini, A. Russo, [The Hitchhiker’s Guide to the Virtual Element Method](#), Mathematical Models and Methods in Applied Sciences 24 (08) (2014) 1541–1573. doi:[10.1142/S021820251440003X](#).
URL <https://www.worldscientific.com/doi/abs/10.1142/S021820251440003X>
- [231] L. Beirão da Veiga, F. Brezzi, L. D. Marini, A. Russo, [Virtual Element Method for general second-order elliptic problems on polygonal meshes](#), Mathematical Models and Methods in Applied Sciences 26 (04) (2016) 729–750. doi:[10.1142/S0218202516500160](#).
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202516500160>
- [232] D. A. Cox, J. Little, D. O’Shea, [Ideals, Varieties, and Algorithms](#), Undergraduate Texts in Mathematics, Springer International Publishing, Cham, 2015. doi:[10.1007/978-3-319-16721-3](#).
URL <http://link.springer.com/10.1007/978-3-319-16721-3>
- [233] J. Kelleher, B. O’Sullivan, [Generating All Partitions: A Comparison Of Two Encodings](#) (Sep 2009). arXiv:[0909.2331](#).
URL <http://arxiv.org/abs/0909.2331>

- [234] F. Bassi, L. Botti, A. Colombo, D. Di Pietro, P. Tesini, [On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations](#), Journal of Computational Physics 231 (1) (2012) 45–65. doi:[10.1016/j.jcp.2011.08.018](#).
URL [https://linkinghub.elsevier.com/retrieve/pii/S0021999111005055](#)
- [235] C. Pozrikidis, [Introduction to Finite and Spectral Element Methods Using Matlab](#), 2nd Edition, CRC Press, Boca Raton, 2014.
URL [https://www.routledge.com/Introduction-to-Finite-and-Spectral-Element-Methods-Pozrikidis/p/book/9781482209150](#)
- [236] L. B. da Veiga, F. Brezzi, L. D. Marini, A. Russo, [H\(div\) and H\(curl\)-conforming virtual element methods](#), Numerische Mathematik 133 (2) (2016) 303–332. doi:[10.1007/s00211-015-0746-1](#).
URL [http://link.springer.com/10.1007/s00211-015-0746-1](#)
- [237] M. G. Cox, [The Numerical Evaluation of B-Splines](#), IMA Journal of Applied Mathematics 10 (2) (1972) 134–149. doi:[10.1093/imamat/10.2.134](#).
URL [https://academic.oup.com/imamat/article-lookup/doi/10.1093/imamat/10.2.134](#)
- [238] C. de Boor, [On calculating with B-splines](#), Journal of Approximation Theory 6 (1) (1972) 50–62. doi:[10.1016/0021-9045\(72\)90080-9](#).
URL [https://linkinghub.elsevier.com/retrieve/pii/0021904572900809](#)
- [239] S. A. Coons, MAC-TR-41: Surfaces for Computer-Aided Design of Space Forms, Tech. rep., Massachusetts Institute of Technology, Cambridge, Massachusetts (1967).
- [240] S. Coons, [Surface Patches and B-Spline Curves](#), in: Computer Aided Geometric Design, Elsevier, 1974, pp. 1–16. doi:[10.1016/B978-0-12-079050-0.50006-0](#).
URL [https://linkinghub.elsevier.com/retrieve/pii/B9780120790500500060](#)
- [241] F. Lin, W. Hewitt, [Expressing Coons-Gordon surfaces as nurbs](#), Computer-Aided Design 26 (2) (1994) 145–155. doi:[10.1016/0010-4485\(94\)90035-3](#).
URL [https://linkinghub.elsevier.com/retrieve/pii/0010448594900353](#)
- [242] J. B. Lasserre, Integration on a convex polytope, Proceedings of the American Mathematical Society (1998). doi:[10.1090/S0002-9939-98-04454-2](#).
- [243] A. Sommariva, M. Vianello, Product Gauss cubature over polygons based on Green’s integration formula, BIT Numerical Mathematics (2007). doi:[10.1007/s10543-007-0131-2](#).
- [244] M. W. Beall, M. S. Shephard, [A General Topology-Based Mesh Data Structure](#), International Journal for Numerical Methods in Engineering 40 (9) (1997) 1573–1596. doi:[10.1002/\(SICI\)1097-0207\(19970515\)40:9<1573::AID-NME128>3.0.CO;2-9](#).

- URL [https://onlinelibrary.wiley.com/doi/10.1002/\(SICI\)1097-0207\(19970515\)40:9%3C1573::AID-NME128%3E3.0.CO;2-9](https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-0207(19970515)40:9%3C1573::AID-NME128%3E3.0.CO;2-9)
- [245] R. Biswas, R. C. Strawn, *A new procedure for dynamic adaption of three-dimensional unstructured grids*, Applied Numerical Mathematics 13 (6) (1994) 437–452. doi:10.1016/0168-9274(94)90007-8.
URL <https://linkinghub.elsevier.com/retrieve/pii/0168927494900078>
- [246] Y. Kallinderis, P. Vijayan, *Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes*, AIAA Journal 31 (8) (1993) 1440–1447. doi:10.2514/3.11793.
URL <https://arc.aiaa.org/doi/10.2514/3.11793>
- [247] D. Muller, F. Preparata, *Finding the intersection of two convex polyhedra*, Theoretical Computer Science 7 (2) (1978) 217–236. doi:10.1016/0304-3975(78)90051-8.
URL <https://linkinghub.elsevier.com/retrieve/pii/0304397578900518>
- [248] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, *Computational Geometry*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-77974-2.
URL <http://link.springer.com/10.1007/978-3-540-77974-2>
- [249] L. Kettner, *CGAL 5.3 - Halfedge Data Structures* (2021).
URL https://doc.cgal.org/latest/HalfedgeDS/index.html#Chapter_Halfedge_Data_Structures
- [250] L. Kobbelt, *OpenMesh*.
URL <https://www.graphics.rwth-aachen.de/software/openmesh/>
- [251] M. Mäntylä, R. Sulonen, *GWB: A Solid Modeler with Euler Operators*, IEEE Computer Graphics and Applications 2 (7) (1982) 17–31. doi:10.1109/MCG.1982.1674396.
URL <http://ieeexplore.ieee.org/document/1674396/>
- [252] M. Mäntylä, *Topological analysis of polygon meshes*, Computer-Aided Design 15 (4) (1983) 228–234. doi:10.1016/0010-4485(83)90126-4.
URL <https://linkinghub.elsevier.com/retrieve/pii/0010448583901264>
- [253] M. Mäntylä, *A note on the modeling space of Euler operators*, Computer Vision, Graphics, and Image Processing 26 (1) (1984) 45–60. doi:10.1016/0734-189X(84)90129-4.
URL <https://linkinghub.elsevier.com/retrieve/pii/0734189X84901294>
- [254] C. Geuzaine, J.-F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331. doi:10.1002/nme.2579.
URL <https://onlinelibrary.wiley.com/doi/10.1002/nme.2579>
- [255] J. R. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, 1996, pp. 203–222. doi:10.1007/BFb0014497.
URL <http://link.springer.com/10.1007/BFb0014497>

- [256] H. Si, [TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator](#), ACM Transactions on Mathematical Software 41 (2) (2015) 1–36. doi:10.1145/2629697.
URL <https://dl.acm.org/doi/10.1145/2629697>
- [257] C. Talischi, G. H. Paulino, A. Pereira, I. F. M. Menezes, [PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab](#), Structural and Multidisciplinary Optimization 45 (3) (2012) 309–328. doi:10.1007/s00158-011-0706-z.
URL <http://link.springer.com/10.1007/s00158-011-0706-z>
- [258] D.-M. Yan, W. Wang, B. Lévy, Y. Liu, [Efficient computation of clipped Voronoi diagram for mesh generation](#), Computer-Aided Design 45 (4) (2013) 843–852. doi:10.1016/j.cad.2011.09.004.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0010448511002351>
- [259] S. G. Johnson, [The NLOpt nonlinear-optimization package](#).
URL <http://github.com/stevengj/nlopt>
- [260] F. Aurenhammer, R. Klein, D. T. Lee, [Voronoi diagrams and delaunay triangulations](#), World Scientific, 2013. doi:10.1142/8685.
URL <https://www.worldscientific.com/worldscibooks/10.1142/8685>
- [261] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, D. G. Kendall (Eds.), [Spatial Tessellations](#), Wiley Series in Probability and Statistics, John Wiley and Sons, Inc., Hoboken, NJ, USA, 2000. doi:10.1002/9780470317013.
URL <http://doi.wiley.com/10.1002/9780470317013>
- [262] F. Aurenhammer, [Voronoi diagrams—a survey of a fundamental geometric data structure](#), ACM Computing Surveys 23 (3) (1991) 345–405. doi:10.1145/116873.116880.
URL <https://dl.acm.org/doi/10.1145/116873.116880>
- [263] Q. Du, V. Faber, M. Gunzburger, [Centroidal Voronoi Tessellations: Applications and Algorithms](#), SIAM Review 41 (4) (1999) 637–676. doi:10.1137/S0036144599352836.
URL <http://epubs.siam.org/doi/10.1137/S0036144599352836>
- [264] Q. Du, M. Emelianenko, L. Ju, [Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations](#), SIAM Journal on Numerical Analysis 44 (1) (2006) 102–119. doi:10.1137/040617364.
URL <http://epubs.siam.org/doi/10.1137/040617364>
- [265] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, C. Yang, [On centroidal voronoi tessellation—energy smoothness and fast computation](#), ACM Transactions on Graphics 28 (4) (2009) 1–17. doi:10.1145/1559755.1559758.
URL <https://dl.acm.org/doi/10.1145/1559755.1559758>
- [266] J. C. Hateley, H. Wei, L. Chen, [Fast Methods for Computing Centroidal Voronoi Tessellations](#), Journal of Scientific Computing 63 (1) (2015) 185–212. doi:10.1007/

s10915-014-9894-1.

URL <http://link.springer.com/10.1007/s10915-014-9894-1>

- [267] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, [The quickhull algorithm for convex hulls](#), ACM Transactions on Mathematical Software (TOMS) 22 (4) (1996) 469–483. doi:[10.1145/235815.235821](https://doi.org/10.1145/235815.235821).
URL <http://dl.acm.org/doi/10.1145/235815.235821>
- [268] D. Sieger, P. Alliez, M. Botsch, [Optimizing Voronoi Diagrams for Polygonal Finite Element Computations](#), in: Proceedings of the 19th International Meshing Roundtable, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 335–350. doi:[10.1007/978-3-642-15414-0_20](https://doi.org/10.1007/978-3-642-15414-0_20).
URL http://link.springer.com/10.1007/978-3-642-15414-0_20
- [269] E. Brisson, [Representing geometric structures in \$d\$ dimensions: topology and order](#), in: Proceedings of the fifth annual symposium on Computational geometry - SCG '89, ACM Press, New York, New York, USA, 1989, pp. 218–227. doi:[10.1145/73833.73858](https://doi.org/10.1145/73833.73858).
URL <http://portal.acm.org/citation.cfm?doid=73833.73858>
- [270] T. J. Alumbaugh, X. Jiao, [Compact Array-Based Mesh Data Structures](#), in: Proceedings of the 14th International Meshing Roundtable, Springer-Verlag, Berlin/Heidelberg, pp. 485–503. doi:[10.1007/3-540-29090-7_29](https://doi.org/10.1007/3-540-29090-7_29).
URL http://link.springer.com/10.1007/3-540-29090-7_29
- [271] G. Damiani, [CGAL 5.3 - Combinatorial Maps](#) (2021).
URL https://doc.cgal.org/latest/Combinatorial_map/index.html#Chapter_Combinatorial_Maps
- [272] X. Feng, Y. Wang, Y. Weng, Y. Tong, [Compact combinatorial maps: A volume mesh data structure](#), Graphical Models 75 (3) (2013) 149–156. doi:[10.1016/j.gmod.2012.10.001](https://doi.org/10.1016/j.gmod.2012.10.001).
URL <https://linkinghub.elsevier.com/retrieve/pii/S1524070312000690>
- [273] C. Burstedde, L. C. Wilcox, O. Ghattas, [p4est : Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees](#), SIAM Journal on Scientific Computing 33 (3) (2011) 1103–1133. doi:[10.1137/100791634](https://doi.org/10.1137/100791634).
URL <http://epubs.siam.org/doi/10.1137/100791634>
- [274] E. Wachspress, [Iterative Solution of Elliptic Systems: and Applications to the Neutron Diffusion Equations of Reactor Physics](#), Prentice-Hall, 1966.
- [275] J. P. Aubin, [Behavior of the error of the approximate solutions of boundary value problems for linear elliptic operators by Galerkin's and finite difference methods](#), Annali della Scuola Normale Superiore di Pisa. Scienze Fisiche e Matematiche. III. Ser (1967).
URL <https://zbmath.org/?q=an:0276.65052>

- [276] I. Babuška, J. Osborn, [Eigenvalue problems](#), 1991, pp. 641–787. doi:[10.1016/S1570-8659\(05\)80042-0](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S1570865905800420>
- [277] S. Balay, S. Anhyankar, M. F. Adams, J. Brown, P. Brune, D. Buschleman, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, PETSc Users Manual, Tech. rep., Argonne National Laboratory (2020).
- [278] W. Gropp, E. Lusk, A. Skjellum, [Using MPI: Portable Parallel Programming with the Message-Passing Interface](#), 3rd Edition, The MIT Press, Cambridge, Massachusetts, 2014.
- [279] W. Gropp, T. Hoefler, R. Thakur, E. Lusk, [Using Advanced MPI: Modern Features of the Message-Passing Interface](#), 3rd Edition, The MIT Press, Cambridge, Massachusetts, 2014.
- [280] Y. Saad, M. H. Schultz, [GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems](#), SIAM Journal on Scientific and Statistical Computing 7 (3) (1986) 856–869. doi:[10.1137/0907058](#).
URL <http://epubs.siam.org/doi/10.1137/0907058>
- [281] Y. Saad, [Iterative Methods for Sparse Linear Systems](#), 2nd Edition, Society for Industrial and Applied Mathematics, 2003. doi:[10.1137/1.9780898718003](#).
URL <http://epubs.siam.org/doi/book/10.1137/1.9780898718003>
- [282] V. E. Henson, U. M. Yang, BoomerAMG: A parallel algebraic multigrid solver and preconditioner, in: [Applied Numerical Mathematics](#), 2002. doi:[10.1016/S0168-9274\(01\)00115-5](#).
- [283] G. Karypis, V. Kumar, [A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs](#), SIAM Journal on Scientific Computing 20 (1) (1998) 359–392. doi:[10.1137/S1064827595287997](#).
URL <http://epubs.siam.org/doi/10.1137/S1064827595287997>
- [284] E. Cuthill, J. McKee, [Reducing the bandwidth of sparse symmetric matrices](#), in: [Proceedings of the 1969 24th national conference on -](#), ACM Press, New York, New York, USA, 1969, pp. 157–172. doi:[10.1145/800195.805928](#).
URL <http://portal.acm.org/citation.cfm?doid=800195.805928>
- [285] J. Wood, M. M. R. Williams, [Recent progress in the application of the finite element method to the neutron transport equation](#), Progress in Nuclear Energy 14 (1) (1984) 21–40. doi:[10.1016/0149-1970\(84\)90010-6](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/0149197084900106>
- [286] A. Owens, J. Kópházi, M. D. Eaton, [Optimal trace inequality constants for interior penalty discontinuous Galerkin discretisations of elliptic operators using arbitrary ele-](#)

- ments with non-constant Jacobians, *Journal of Computational Physics* 350 (2017) 847–870. doi:10.1016/j.jcp.2017.09.020.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999117306794>
- [287] E. E. Lewis, M. A. Smith, N. Tsoulfanidis, G. Palmiotti, T. A. Taiwo, R. N. Blomquist, Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation: A 2-D/3-D MOX Fuel Assembly Benchmark, Tech. Rep. NEA/NSC/ DOC(2003)16, Nuclear Energy Agency (NEA), Organisation for Economic Co-operation and Development (2003).
- [288] F. Brezzi, L. D. Marini, E. Süli, *Discontinuous Galerkin Methods for First-Order Hyperbolic Problems*, *Mathematical Models and Methods in Applied Sciences* 14 (12) (2004) 1893–1903. doi:10.1142/S0218202504003866.
URL <https://www.worldscientific.com/doi/abs/10.1142/S0218202504003866>
- [289] R. Tarjan, *Depth-First Search and Linear Graph Algorithms*, *SIAM Journal on Computing* 1 (2) (1972) 146–160. doi:10.1137/0201010.
URL <http://epubs.siam.org/doi/10.1137/0201010>
- [290] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd Edition, MIT Press, Cambridge, Massachusetts, 2009.
- [291] A. H. Aho, J. E. Hopcroft, J. D. Ullman, *Data Structures and Algorithms*, 1st Edition, Addison-Wesley, London, 1983.
- [292] A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.
- [293] T.-W. Huang, D.-L. Lin, C.-X. Lin, Y. Lin, *Taskflow: A Lightweight Parallel and Heterogeneous Task Graph Computing System*, *IEEE Transactions on Parallel and Distributed Systems* 33 (6) (2022) 1303–1320. doi:10.1109/TPDS.2021.3104255.
URL <https://ieeexplore.ieee.org/document/9511796/>
- [294] B. Chapman, G. Jost, R. van der Pas, *Using OpenMP*, MIT Press, Cambridge, Massachusetts, 2007.
- [295] Intel, *Intel TBB Documentation* (2022).
URL <https://oneapi-src.github.io/oneTBB/index.html>
- [296] W. H. Reed, *New Difference Schemes for the Neutron Transport Equation*, *Nuclear Science and Engineering* 46 (2) (1971) 309–314. doi:10.13182/NSE46-309.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE46-309>
- [297] Y. Wang, J. C. Ragusa, *On the Convergence of DGFEM Applied to the Discrete Ordinates Transport Equation for Structured and Unstructured Triangular Meshes*, *Nuclear Science and Engineering* 163 (1) (2009) 56–72. doi:10.13182/NSE08-72.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE08-72>

- [298] Y. Wang, J. C. Ragusa, [A high-order discontinuous Galerkin method for the \$S_N\$ transport equations on 2D unstructured triangular meshes](#), *Annals of Nuclear Energy* 36 (7) (2009) 931–939. doi:[10.1016/j.anucene.2009.03.002](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0306454909000930>
- [299] Y. Wang, Adaptive Mesh Refinement Solution Techniques For The Multigroup S_N Transport Equation Using A Higher-Order Discontinuous Finite Element Method, Ph.D. thesis, Texas A&M University (2009).
- [300] Y. Wang, J. C. Ragusa, [Diffusion Synthetic Acceleration for High-Order Discontinuous Finite Element \$S_N\$ Transport Schemes and Application to Locally Refined Unstructured Meshes](#), *Nuclear Science and Engineering* 166 (2) (2010) 145–166. doi:[10.13182/NSE09-46](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE09-46>
- [301] J. S. Warsa, T. A. Wareing, J. E. Morel, [Krylov Iterative Methods and the Degraded Effectiveness of Diffusion Synthetic Acceleration for Multidimensional \$S_N\$ Calculations in Problems with Material Discontinuities](#), *Nuclear Science and Engineering* 147 (3) (2004) 218–248. doi:[10.13182/NSE02-14](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE02-14>
- [302] J. S. Warsa, T. A. Wareing, J. E. Morel, [Fully Consistent Diffusion Synthetic Acceleration of Linear Discontinuous \$S_N\$ Transport Discretizations on Unstructured Tetrahedral Meshes](#), *Nuclear Science and Engineering* 141 (3) (2002) 236–251. doi:[10.13182/NSE141-236](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE141-236>
- [303] E. W. Larsen, [Unconditionally Stable Diffusion-Synthetic Acceleration Methods for the Slab Geometry Discrete Ordinates Equations. Part I: Theory](#), *Nuclear Science and Engineering* 82 (1) (1982) 47–63. doi:[10.13182/NSE82-1](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE82-1>
- [304] S. D. Pautz, T. S. Bailey, [Parallel Deterministic Transport Sweeps of Structured and Unstructured Meshes with Overloaded Mesh Decompositions](#), *Nuclear Science and Engineering* 185 (1) (2017) 70–77. doi:[10.13182/NSE16-34](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE16-34>
- [305] S. D. Pautz, [An Algorithm for Parallel \$S_N\$ Sweeps on Unstructured Meshes](#), *Nuclear Science and Engineering* 140 (2) (2002) 111–136. doi:[10.13182/NSE02-1](#).
URL <https://www.tandfonline.com/doi/full/10.13182/NSE02-1>
- [306] J. I. Vermaak, J. C. Ragusa, M. L. Adams, J. E. Morel, [Massively parallel transport sweeps on meshes with cyclic dependencies](#), *Journal of Computational Physics* 425 (2021) 109892. doi:[10.1016/j.jcp.2020.109892](#).
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999120306665>

- [307] S. J. Plimpton, B. Hendrickson, S. P. Burns, W. McLendon, L. Rauchwerger, [Parallel \$S_N\$ Sweeps on Unstructured Grids: Algorithms for Prioritization, Grid Partitioning, and Cycle Detection](#), Nuclear Science and Engineering 150 (3) (2005) 267–283. doi:10.13182/NSE150-267.
URL <https://www.tandfonline.com/doi/full/10.13182/NSE150-267>
- [308] P. F. Antonietti, L. Mascotto, M. Verani, A multigrid algorithm for the p-version of the virtual element method, ESAIM: Mathematical Modelling and Numerical Analysis 52 (1) (2018) 337–364. doi:10.1051/m2an/2018007.
- [309] P. F. Antonietti, S. Berrone, M. Busetto, M. Verani, [Agglomeration-based geometric multigrid schemes for the Virtual Element Method](#) (Dec 2021). arXiv:2112.11080.
URL <http://arxiv.org/abs/2112.11080>
- [310] P. F. Antonietti, M. Sarti, M. Verani, [Multigrid Algorithms for \$hp\$ -Discontinuous Galerkin Discretizations of Elliptic Problems](#), SIAM Journal on Numerical Analysis 53 (1) (2015) 598–618. doi:10.1137/130947015.
URL <http://epubs.siam.org/doi/10.1137/130947015>
- [311] J. A. Sethian, [Fast Marching Methods](#), SIAM Review 41 (2) (1999) 199–235. doi:10.1137/S0036144598347059.
URL <http://epubs.siam.org/doi/10.1137/S0036144598347059>