Modelling of Turbulent Reacting Flows with Polydispersed Particle Formation

by

Fabian Sewerin
00788523

PhD thesis

submitted for the degree of

Doctor of Philosophy

in Mechanical Engineering

at Imperial College London,
Department of Mechanical Engineering
(Division of Thermofluids)

Supervisor: Dr Stelios Rigopoulos

September 4, 2017

Abstract

The formation of particles in a turbulent carrier flow is central to many environmental processes and engineering applications. Specific examples include the precipitation of crystals from aqueous solutions, the condensation of droplets in the context of cloud formation, flame synthesis of metal oxide particles or the formation of soot in hydrocarbon combustion devices. Frequently, the particles can be characterized by intrinsic properties such as the particle size, shape or charge and the distribution of values for these properties amongst a particle population is taken as an indicator for the quality, toxicity or environmental impact of the particulate product.

In the present work, we consider particle size as a representative property and develop a comprehensive model and numerical solution scheme for predicting the evolution of the size distribution associated with a particulate phase forming in a turbulent carrier flow. Physically, the evolution of the particle size distribution can be described by the population balance equation (PBE) which we incorporate into a large eddy simulation (LES) framework for turbulent reacting flows. In order to resolve the influence of turbulence on chemical reactions and particle formation, a formulation based on an evolution equation for the LES-filtered one-point, one-time probability density function (*pdf*) associated with the instantaneous fluid composition and particle number density distribution is developed. This forms the basis of our LES-PBE-PDF approach; its main advantage is that the LES-filtered particle size distribution can be predicted at each spatial location in the flow domain and every time instant without any restriction on the chemical or particle formation kinetics.

In view of a numerical solution scheme, we present a formulation in terms of Eulerian stochastic fields whose evolution statistically reproduces that of the joint scalar-number density *pdf.* For the discretization of the particle number density stochastic field equation, we develop a novel explicit adaptive grid technique which is able to accurately resolve sharp and moving features of the LES-filtered particle size distribution. This scheme is based on a space and time dependent coordinate transformation on particle size space which is explicitly marched in time. One innovative feature is an adjustment scheme for the distribution of grid points in particle size space which allows us to accommodate nucleation source terms and control the grid stretching. Our analysis demonstrates that the explicit adaptive grid method requires over an order of magnitude fewer grid points in particle size space to obtain a similar accuracy as a comparable fixed grid discretization scheme.

In a final investigation, we explore accelerating the time consuming chemical kinetics integration by implementing a high order implicit integration scheme for execution on a graphics card (GPU). This GPU implementation can be operated in conjunction with conventional solver implementations on central processing units (CPUs), yielding a notable performance benefit on desktop computer systems.

The combined LES-PBE-PDF approach is applied to model the precipitation of $BaSO_4$ particles in a coaxial pipe mixer, the condensation of an aerosol in a developed turbulent mixing layer and the formation of soot in a turbulent, non-premixed methane-air flame. Here, predictions of the particle size distribution or its moments are compared with experimental measurements and solutions from direct numerical simulations (DNS). Our analyses and findings not only indicate the predictive capabilities of the LES-PBE-PDF approach, but also demonstrate the computational efficiency and accuracy of the numerical solution scheme.

Acknowledgements

Throughout my time as a PhD student at Imperial College London I have been financially supported by the Imperial College PhD Scholarship Scheme. I am grateful for having had the opportunity to pursue a PhD at Imperial and to be part of the scientific community here. For the computationally intensive test cases reported in this work I used the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk) whose support I gratefully acknowledge. The DNS data of Reference [213] were provided to me by Fabrizio Bisetti, Amjad Alshaarawi and Kun Zhou who also shared their implementation of the DBP droplet formation kinetics with me.

I wish to thank my supervisor Dr Stelios Rigopoulos for his support, encouragement and trust and, above all, for giving me the liberty to explore and develop my ideas. Our discussions and his critical advice have been invaluable to me and I am grateful for his patience and positive spirits during the set backs I experienced. I also wish to express my sincerest gratitude to Dr Salvador Navarro-Martinez for his advice and comments on the practical problems I encountered with LES-BOFFIN.

My colleagues at Imperial have been invaluable to the creative, supportive and harmonious atmosphere in our offices and I am very grateful for their friendship and for rendering my past years in London so extraordinary and memorable: Vasilis, Petros, Teddy, Benoit, Panos, Lucas, Larry, Yuri and Sun (in the order in which we met) and my table tennis partners Nuri, Kris and Sam. I wish them all the good fortune they deserve for embarking on the careers they have in mind.

I am thankful to Kate for her presence, her love and her rationality – and for our many common journeys.

My parents and my brother Sebastian have contributed to my PhD and my personal endeavours in so many ways that I feel words are too little to do them justice. I am indebted to Mum for her patience, positivity, care, her love and her encouragement in difficult times; to Pap for all his love, advice, experience, efforts and empathy; and to Sebastian for the times we spent together, his humour, advice and for patiently reviewing my proof in Appendix A.1.

To Mum and Pap

Er sprach leise, doch beschwingt; die klaren, harten, logischen, lateinischen Sätze kamen aus seinem Mund wie griechische Verse. „Macht", träumte er, „[...] das ist die Idee, wie sie aus dem Kopf herausspringt, Tat wird, die plumpe Wirklichkeit überwältigt. [...]"

Varro in „Der falsche Nero"
Lion Feuchtwanger

Contents

List of Figures

List of Tables

Nomenclature

Operators and symbols

$(\cdot)^{-1}$	Inverse
$(\cdot)^T$	Transposition
$\dot{(\cdot)}$	Time derivative
$\langle(\cdot)\rangle_{\Omega_j}$	Average over cell Ω_j
$\leftarrow$	Substitution operator
$\overline{(\cdot)}$	LES-operator
$\partial(\cdot)/\partial t$	Partial derivative with respect to t
$\widetilde{(\cdot)}$	Favre-filter
$D(\cdot)/Dt$	Material time derivative
$d(\cdot)/dt$	Derivative with respect to t

Sets and spaces

$[0, L]$	Particle property space
$\mathcal{L}^\star$	An interval about $l^\star$
$\mathcal{L}^+$	An interval about l^+
Ω	Flow domain
$\mathbb{R}_0^+$	Set of non-negative real numbers

Functions

arg	Argument function
$\delta(\cdot)$	Dirac's delta distribution
erf	Error function

exp	Exponential function
floor	Returns the largest integer that is smaller than or equal to the argument value
ln	Natural logarithm
max	Maximum of the argument values
mod	Modulo operation
id	Identity

Physical quantities

$\bar{t}(x)$	Time coordinate associated with a batch reactor
$\bar{x}(t)$	Spatial coordinate associated with a steady-state plug flow reactor
$\chi(z)$	Scalar dissipation rate
$\dot{\boldsymbol{\omega}}(\mathbf{Y}, N)$	Scalar production/destruction rates
$\dot{s}(l, \mathbf{Y}, N)$	Particle formation rate
$\hat{\rho}(\mathbf{Y})$	Mixture density computed in terms of the reactive scalars
$\mathbf{n}(\mathbf{x})$	Outward unit normal vector
$\nu(\mathbf{x}, t)$	Kinematic viscosity
$\rho(\mathbf{x}, t)$	Mixture density
$\tau_{ij}(\mathbf{x}, t)$	Viscous stress tensor
$\mathbf{u}(\mathbf{x}, t)$	Ambient velocity field
$\mathbf{x}$	A location in physical space
$\mathbf{Y}(\mathbf{x}, t)$	Reactive scalars (fluid phase composition)
$\mathbf{Y}_0(\mathbf{x})$	Initial reactive scalars
$\mathbf{Y}_F$	Fuel composition
$\mathbf{Y}_O$	Oxidizer composition
$D(\mathbf{x}, t)$	Diffusivity of the reactive scalars
$D_p(\mathbf{x}, t)$	Particle diffusivity

$f(\tau, \mathbf{x}, t)$	Transformed particle number density
$G(l, \mathbf{Y})$	Particle growth/shrinkage rate
$J_{ij}(\mathbf{x}, t)$	Diffusive flux of scalar i in the jth coordinate direction
$K_j(\mathbf{x}, t)$	Diffusive flux of number density in the jth coordinate direction
l	A particle property (e.g., particle size)
$M_k(N)$	kth moment of $N(l, \mathbf{x}, t)$
$N(l, \mathbf{x}, t)$	Particle number density
$N_0(l, \mathbf{x})$	Initial particle number density (Chapter 3)
$N_\Omega(l, \mathbf{x})$	Initial particle number density (Chapter 2)
$N_{\rho,0}(l, \mathbf{x})$	Initial mass-based particle number density
$N_\rho(l, \mathbf{x}, t)$	Mass-based particle number density
n_s	Number of reactive scalars
$p(\mathbf{x}, t)$	Pressure
s	Flamelet strain rate (Chapter 5)
t	Time
$T(\mathbf{Y})$	Temperature
t_0	Initial time
v	Particle volume
$V(\mathbf{x}, t)$	Volume density of the particulate phase
x	Axial coordinate
z	Mixture fraction

LES and PDF-formalism

Δ	Local LES mesh size
$\epsilon(\cdot, \mathbf{x}, t)$	Monte Carlo error
$\Gamma(\mathbf{x}, t)$	Eddy viscosity
$\kappa(\mathbf{x}, t)$	Turbulent mixing frequency

Nomenclature

$\mathbf{m}(\mathbf{x}, t, \mathbf{z})$	Micromixing model
$\mathcal{M}_i, \mathcal{M}_p$	Micromixing operators
$\Phi(\mathbf{u}, p, \mathbf{Y}, N)$	A scalar function(al) of $\mathbf{u}(\mathbf{x}, t)$, $p(\mathbf{x}, t)$, $\mathbf{Y}(\mathbf{x}, t)$ and $N(\cdot, \mathbf{x}, t)$
$\phi(\mathbf{x}, t)$	A scalar function
$\boldsymbol{\phi}(t; \tau, \mathbf{x})$	Transformed stochastic fields
$\boldsymbol{\phi}^{(i)}(t; \tau, \mathbf{x})$	ith realization of the transformed stochastic fields
ψ	Sample space variable associated with $\phi(\mathbf{x}, t)$
$\mathbf{s}(\cdot, \mathbf{y}, n(\cdot))$	Joint scalar-number density source term
$\sigma(\cdot, \mathbf{x}, t)$	Standard deviation of an observable $F(\mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t))$
$\tau^*_{ij}(\mathbf{x}, t)$	Residual stress tensor
$\boldsymbol{\theta}(t; l, \mathbf{x})$	Stochastic fields
$\boldsymbol{\theta}^{(i)}(t; l, \mathbf{x})$	ith realization of the stochastic fields
$\mathbf{v}$	Sample space variable associated with $\mathbf{u}(\mathbf{x}, t)$
$\mathbf{y}$	Sample space variable associated with $\mathbf{Y}(\mathbf{x}, t)$
$\mathbf{z}$	Joint sample space variable associated with $(\mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t))$ or $(\mathbf{Y}(\mathbf{x}, t), N_\rho(\cdot, \mathbf{x}, t))$
a	A constant field
C_κ	Micromixing constant
$F(\mathbf{y}, n(\cdot))$	A scalar function(al) of $(\mathbf{y}, n(\cdot))$
$f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$	Joint scalar-number density *pdf*
$f_\phi(\psi; \mathbf{x}, t)$	Filtered *pdf* associated with a single realization of $\phi(\mathbf{x}, t)$
$f_{\mathbf{Y}}(\mathbf{y}; \mathbf{x}, t)$	Joint scalar *pdf*
$G(\mathbf{x}, \mathbf{x}')$	LES filter kernel
$g(\mathbf{y}, n(\cdot); \mathbf{x}, t)$	Fine-grained density associated with $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$
$h'(\mathbf{z}, t; \mathbf{x})$	Fine-grained density associated with $h(\mathbf{z}, t; \mathbf{x})$
$h(\mathbf{z}, t; \mathbf{x})$	*pdf* associated with $\boldsymbol{\theta}(t; \mathbf{x})$
$n(\cdot)$	Sample space function associated with $N(\cdot, \mathbf{x}, t)$ or $N_\rho(\cdot, \mathbf{x}, t)$

n_f	Number of stochastic fields
n_ϕ	Number of reactive scalars plus one
q	Sample space variable associated with $p(\mathbf{x}, t)$
$W_j(t)$	jth Wiener process

Coordinate transformation

$\bar{\tau}(l, \mathbf{x}, t)$	Inverse coordinate transformation associated with $\bar{l}(\tau, \mathbf{x}, t)$
$\bar{\tau}_0(l)$	Inverse coordinate transformation associated with $\bar{l}_0(\tau)$
$\bar{\tau}_\infty(l)$	Inverse coordinate transformation associated with $\bar{l}_\infty(\tau)$
$\bar{l}(\tau, \mathbf{x}, t)$	An $(\mathbf{x}, t)$-dependent coordinate transformation on $[0, L]$
$\bar{l}_0(\tau)$	Equidistributing coordinate transformation on $[0, L]$
$\bar{l}_\infty(\tau)$	Adjusted equidistributing coordinate transformation on $[0, L]$
$\dot{R}(l, s)$	Rate of change for the node density constraints violation
κ	Relaxation time constant
λ	Scaling parameter
$\rho(l, s)$	Node density distribution
$\rho_0(l)$	Node density distribution associated with $\bar{\tau}_0(l)$
$\rho_\infty(l)$	Node density distribution associated with $\bar{\tau}_\infty(l)$
ρ_τ	Uniform node density in τ-space
$\rho_{\min}(l)$	Minimum admissible node density in l-space
b	Regularization constant
$b(\tau, \mathbf{x}, t)$	A term involving second order derivatives of $\bar{l}(\tau, \mathbf{x}, t)$
C	A constant
c	Normalization constant
C'	A positive constant
d	Equidistribution constant associated with a node density distribution
$g(\tau, \mathbf{x}, t)$	Contribution of $\bar{l}(\tau, \mathbf{x}, t)$ to the cumulative particle growth rate

$G_{\rho_0}(l, l')$	A hyperbolic function with a tip at $(l', \rho_0(l'))$
L	Maximum representable value for the particle property
l^+	A particular l-value
$l^\star$	A particular l-value
l_l	Minimum representable value for the particle property
$M_0(N_0)$	Zeroth moment of $N_0(l)$
$m_{N_0}(l)$	Monitor function
$N_0(l)$	Reference particle property distribution
$P_{\rho_0}(l)$	Padding associated with $\rho_0(l)$
$P_{\rho_\infty}(l)$	Padding associated with $\rho_\infty(l)$
$P_\rho(l, s)$	Padding associated with $\rho(l, s)$
r	Maximum admissible grid stretching
s	A measure of progress or time
s_∞	Time at which $\rho(l, s)$ reaches a steady-state
T	Time constant
$w(\tau, \mathbf{x}, t)$	Jacobian associated with $\bar{l}(\tau, \mathbf{x}, t)$

Discrete representation

$\bar{\tau}_j$	Coordinate transformation at time s_j
Δs	Constant time step
Δt	Constant time step
Δt_k	Time step from t_k to t_{k+1}
Δx	Constant grid spacing along the x-coordinate
$\Delta \tau$	Uniform grid spacing in τ-space
tol	Absolute/relative convergence tolerances
Ω_j	Finite volume subdomain in physical space
$\partial\Omega_j$	Boundary of Ω_j

$\phi(\tau_i, \mathbf{x}, t)$	Flux limiter at node τ_i
τ_i	A grid node in τ-space
$\mathbf{x}_j$	Midpoint of cell Ω_j
$D'(\mathbf{x}, t)$	Effective diffusivity
$f_i(\mathbf{x}, t)$	Average transformed particle number density on cell $[\tau_i, \tau_{i+1}]$
$F_{i,j}(t)$	Average discrete number density on $[\tau_i, \tau_{i+1}] \times \Omega_j$
h	Constant time step
l_i	A grid node in l-space
n_c	Number of finite volume cells in physical space
n_p	Number of grid points in τ-space
s_j	A point in time
t_k	A point in time
$u(\tau_i)$	Index of the cell upstream of node τ_i
$uu(\tau_i)$	Index of the second cell upstream of node τ_i
V_j	Volume of cell Ω_j
$w_i(\mathbf{x}, t)$	Jacobian of $\bar{l}(\tau, \mathbf{x}, t)$ on $[\tau_i, \tau_{i+1}]$
z_i	A grid point in mixture fraction space

Numerical examples and kinetics

$[(\cdot)]$	Molar concentration of species $(\cdot)$
$\bar{C}_p$	Mixture molar specific heat at constant pressure
χ	A fraction of surface sites
$\dot{\omega}^{\star}_{C_2H_2}(\mathbf{Y}, N)$	Consumption/release rate of C_2H_2 due to soot formation
$\dot{\omega}^{\star}_{H_2}(\mathbf{Y}, N)$	Consumption/release rate of H_2 due to soot formation
Γ	Kinematic diffusivity
$\gamma_{\pm}(\mathbf{Y})$	Mean activity coefficient
$\hat{M}_2(\mathbf{x}, t)$	Second moment computed for a lognormal distribution from $M_0(\mathbf{x}, t)$, $M_1(\mathbf{x}, t)$ and $M_3(\mathbf{x}, t)$

$\lambda(\mathbf{x},t)$	Coefficient for a lognormal droplet size distribution at $(\mathbf{x},t)$
ν_{ij}	Stoichiometric coefficients
ϕ	Particle sphericity
ρ_s	Density of solid soot
ρ_{BaSO_4}	Density of BaSO_4
σ	Stefan-Boltzmann constant
$\sigma(\mathbf{x},t)$	Coefficient for a lognormal droplet size distribution at $(\mathbf{x},t)$
$C_{\min}$	Number of C-atoms per soot nucleus
C_s	Radiation coefficient
d	Jet diameter
$e(t)$	Average deviation from reference solution
$f_a(N)$	Soot surface fraction
$f_v(N)$	Soot particle volume fraction
g	Constant advection velocity
$H(\mathbf{Y},N)$	Loss in enthalpy due to gas phase and soot radiation
$H_i(T)$	Molar enthalpy of species i
$k_1(T),k_2(T)$	Arrhenius rate expressions
$K_A(T),K_B(T)$	T-dependent parameters due to Nagle and Strickland-Constable [134]
k_B	Boltzmann's constant
k_D	Mass transfer coefficient
k_g	Surface integration coefficient
k_S	Solubility product
$K_T(T),K_z(T)$	T-dependent parameters due to Nagle and Strickland-Constable [134]
k_v	Volume shape factor
l_1,l_2,l_3	Length of the computational domain in the x_1, x_2 and x_3-directions
l_{nuc}	Mean nuclei size

MW_i	Molecular weight of species i
MW_s	Molecular weight of solid soot
N_A	Avogadro's number
p_i	Partial pressure of species i
q_j	Progress variable associated with reaction j
$R_N(\mathbf{Y})$	Nucleation rate
$S(\mathbf{Y})$	Supersaturation
$s_N(\mathbf{Y})$	Soot nucleation rate
$s_{\mathrm{C_2H_2}}(\mathbf{Y})$	Specific surface growth rate
$s_{\mathrm{O_2}}(\mathbf{Y})$	Specific surface shrinkage rate due to oxidation by O_2
$s_{\mathrm{OH}}(\mathbf{Y})$	Specific surface shrinkage rate due to oxidation by OH
T_b	Ambient background temperature
u	Bulk velocity

Time integration

α_l	Quadrature mid-points
β_{lk}	Quadrature weights
γ_l	Quadrature weights
$\mathbf{k}_l$	Intermediate solutions
$\tilde{\alpha}$	A particular quadrature mid-point
$\tilde{\gamma}_l$	Quadrature weights
m	Number of stages of an implicit Runge-Kutta method
t'	Normalized time

GPU computing

$\bar{t}_{hd}$	Overhead for submitting a read/write command to the device
$\dot{n}$	Number of ODE systems that are solved per unit of time
$\mathbf{P}$	Property matrix

Nomenclature

$\mathbf{r}_i$	Row i of property matrix $\mathbf{P}$
$\tilde{\mathbf{P}}$	Permuted property matrix
$\mathbf{x}$	Permutation vector
a_0, a_{hd}, b_{hd}	Coefficients
K_j	Number of reactions assigned to work item j
l	OpenCL work group size
m_{fo}	Maximum number of fall-off parameters per reaction
m_{sp}	Maximum number of species per reaction
m_{tb}	Maximum number of third bodies per reaction
n	Number of ODE systems
n_0	Number of ODE systems below which the runtime remains constant
n_1	Number of ODE systems solved in a single t-cycle
n_2	Number of ODE systems solved in a single p-cycle
n_b	Memory buffer size required by a single ODE system
n_c	Number of reactive scalars
n_{fo}	Number of fall-off reactions
n_{hd}	Number of ODE systems beyond which the data transfer time increases linearly with n
n_{max}	Maximum number of ODE systems per kernel invocation
n_p	Number of ODE systems per t/p-cycle
n_r	Number of reactions
n_{sp}	Number of species
n_{tb}	Number of reactions involving third body efficiencies
p_1, p_2, p_3	Properties associated with chemical reactions
r	Number of t/p-cycles
r_b	Maximum allocatable buffer size

r_m	Remaining memory available on a device
s_1, s_2	Number of p-cycles
$t(r)$	Total runtime as a function of the number of t/p-cycles
t_{dd}	Time for executing a kernel on a device
t_{dh}	Time for device-to-host memory transfer
t_{hd}	Time for host-to-device memory transfer

Abbreviations

fdf	A filtered density function
pdf	A probability density function
API	Application programming interface
BC	Boundary condition
CPU	Central processing unit
DBP	Dibutyl phtalate ($C_{16}H_{22}O_4$)
DNS	Direct numerical simulation
DPB	Discretized population balance
DQMOM	Direct quadrature method of moments
DRAM	Dynamic random access memory
EAGM	Explicit adaptive grid method
FDM	Finite difference method
FEM	Finite element method
FVM	Finite volume method
GPU	Graphics processing unit
HMOM	Hybrid method of moments
IEM	Interaction by exchange with the mean
IVP	Initial value problem
LES	Large eddy simulation

MMPDE	Moving mesh partial differential equation
MOM	Method of moments
MOMIC	Method of moments with interpolative closure
MPI	Message passing interface
ODE	Ordinary differential equation
OpenCL	Open Computing Language
PBE	Population balance equation
PCIe	Peripheral component interconnect express
PDF	Probability density function (modelling approach)
QMOM	Quadrature method of moments
RANS	Reynolds averaged Navier-Stokes equations
rms	Root mean square
SIMD	Single instruction stream, multiple data streams
TVD	Total variation diminishing
VSD	Volumetric size distribution
XMOMY	A variant of the method of moments, e.g., MOMIC, QMOM, DQMOM, HMOM

Chapter 1

Introduction

1.1 Turbulent flows with particle formation

Turbulent reacting flows with particle formation appear in many environmental and engineering processes. In some applications such as precipitation, crystallization or cell growth experiments, the particulate phase constitutes the desired product of the unit operation, while in others particles appear as by-products, possibly altering the process characteristics or affecting its outcomes. This may occur, for instance, in hydrocarbon combustion devices, where under certain conditions soot particles form, potentially polluting the combustor or being released into the environment. The prediction and analysis of particle formation processes play an important role, for instance, in the reduction of pollutant emissions, the control of aerosols or the design of process conditions in chemical reactors.

Depending on the particular application, the individual particles can be characterized by different intrinsic properties, representing, for instance, the particle size, shape, charge or velocity. As the particles interact with the ambient fluid phase and/or with each other, these intrinsic properties change and, by consequence, the particles evolve both in particle property space and in physical space. If a large number of particles is present, it is common to adopt a continuum viewpoint and to monitor the number density of particles whose intrinsic properties take on particular values at a given point in the flow domain and instant of time. This naturally leads to the particle property distribution which describes the change in particle number density as particle property space is traversed. The Eulerian evolution of the particle property distribution throughout the flow domain is physically governed by the population balance equation [77].

In the applications mentioned above, the particles are so small that momentum exchange between the particles and the carrier fluid is instantaneous. As a result, the particles react immediately to changes in the ambient flow field and do not experience lift or drag forces exerted by the ambient flow. Such particles are termed non-inertial particles and we confine the attention to these particles in the present work. The formulation of a population balance based model to account for particle inertia is outlined by Rigopoulos [173], for instance. Furthermore, our focus lies on applications in which particles are polydispersed with respect to a single particle property, for example, particle size. The particle population is then characterized by its particle size distribution and other relevant properties are inherited from the ambient fluid.

Frequently, the carrier fluid and the immersed particulate phase flow turbulently. While turbulence has been addressed in many scientific works, it seems to have evaded a precise definition. Rather, our understanding of turbulence is more phenomenological in nature. The instantaneous flow field appears to vary randomly and chaotically, forming a multitude of eddies on different length and time scales that pervade the fluid motion. In this way, mixing is greatly enhanced and the transport of momentum, fluid constituents or particles appears to be more homogenizing than in a laminar flow. In the Reynolds-averaged approach to turbulence modelling, the governing fields describing the flow and the flowing medium are considered as random fields; the objective then consists in computing expectations, variances or, more generally, low order statistical moments of the random fields. This is met with a paradox, however: The physical laws governing the time and space evolution of the governing fields are deterministic. Commonly, this paradox is resolved by arguing that randomness enters our physical description through initial and boundary conditions. In practice, randomness can be triggered, for instance, by surface roughness in wall-bounded flows, changes in flow rates, pollutants, dust or many other factors which are difficult to control in the realization and repetition of a particular experiment. While the instantaneous flow field displays the characteristics we associate with turbulence, it is found that the ensemble average of several realizations behaves deterministically – reflecting the statistical expectation mentioned above.

A central aspect of the Reynolds-averaged (RANS) description is that the influence of the remaining statistical moments onto the moments which are solved for requires closure by physical insights into the processes through which the moments interact. Although these closures have reached a mature state by now and are widely deployed in industrial applications, there does not exist as of yet a universal closure model which is suitable for all flow configurations and performs equally well in wall-bounded flows as in free jets, for instance. This has led to the question whether a formal operation other than the statistical moment transformation can be found for which the transformed fields are less sensitive to the particular turbulence closure employed. Such an operation forms the foundation of large eddy simulation (LES). Originally, the LES concept has been developed by researchers in the atmospheric sciences community, but within the past two decades LES has attracted increasing interest from fluid mechanicians with other applications in mind. Indeed, current efforts of the combustion community are concerned with the construction of LES closures that are suitable for modelling reacting flows, possibly including particle formation. A part of these efforts are the developments which we report in this work.

Specifically, our objectives are, first, to develop an LES-based model for predicting the change in fluid composition and particle size distribution throughout a turbulent carrier flow with particle formation and, second, to devise a computationally efficient and accurate numerical solution scheme. In view of the second point, we invoke two complementary strategies. In the first one, a novel numerical solution scheme with an advantageous accuracy over computational cost ratio is developed, while in the second strategy a given numerical solution scheme is reimplemented for execution on a graphics processing unit (GPU). In the following section, the scientific contributions of our work are succinctly summarized.

1.2 Specific contributions

In view of the objectives set out in the previous section, we report on four main contributions in this work:

A grid-adaptive discretization scheme for the PBE. In Chapter 2, we present a novel explicit solution-adaptive numerical scheme for discretizing the spatially inhomogeneous and unsteady PBE in particle size space. This scheme is based on a space and time dependent coordinate transformation which redistributes resolution in particle size space according to the shapes of recent solutions for the particle size distribution. In particular, the coordinate transformation is marched in time explicitly.

By design, our adaptive grid technique is able to accurately capture sharp features such as peaks or near-discontinuities, while maintaining the semi-discrete system size and adhering to a uniform fixed grid discretization in transformed particle property space. This is particularly advantageous if the PBE is combined with a spatially and temporally fully resolved flow model and a standard Eulerian solution scheme is applied in physical space. In order to accommodate localized source terms and to control the grid stretching, we develop a robust scheme for modifying the coordinate transformation such that constraints on the resolution in physical particle property space are obeyed.

As an example, we consider the precipitation of $BaSO_4$ particles from an aqueous solution in a plug flow reactor. Our findings demonstrate that for a given accuracy of the numerical solution the explicit adaptive grid technique requires over an order of magnitude fewer grid points than a comparable fixed grid discretization scheme.

An LES-PBE-PDF approach for modelling particle formation in turbulent constant density fluids. In Chapter 3, we present a comprehensive model and stochastic numerical solution scheme for predicting the evolution of a particle property distribution in a turbulent constant density flow. Based on the concept of LES, the existing LES-transported *pdf* approach for fluid phase scalars is augmented by the particle number density and a modelled evolution equation for the filtered probability density function (*pdf*) associated with the instantaneous fluid composition and particle property distribution is obtained. This LES-PBE-PDF approach allows us to predict the LES-filtered particle property distribution at each spatial location and instant in time without any restriction on the chemical or particle formation kinetics. The numerical solution scheme is based on a reformulation of the joint scalar-number density *pdf* transport equation in terms of a statistically equivalent system of Eulerian stochastic fields.

As test cases, we consider the precipitation of $BaSO_4$ crystals in a coaxial pipe mixer as well as the condensation of an aerosol in a developed turbulent mixing layer. Our investigations in this chapter not only demonstrate the predictive capabilities of the LES-PBE-PDF model, but also indicate the computational efficiency of the numerical solution scheme.

An LES-PBE-PDF approach for modelling soot formation. Subsequently, in Chapter 4, the LES-PBE-PDF model and the stochastic field formulation are generalized to variable density flows at low Mach number and applied to investigate soot formation in the turbulent non-premixed Delft III flame. Here, the soot kinetics encompass acetylene-based rate expressions for nucleation and growth that have previously been validated in the context of laminar diffusion flames. In addition, both species consumption by soot formation and radiation based on the assumption of optical thinness are accounted for. While the agreement of our model predictions with experimental measurements is not perfect, we indicate the benefits of the combined LES-PBE-PDF model and demonstrate its computational viability.

A GPU-based implicit solver for the reaction fractional step. Since much of the computational expense of current reacting flow solvers is incurred during the chemical kinetics integration, we explore in Chapter 5 the question whether such integration schemes can be accelerated by execution on a modern GPU. Frequently, chemical reaction mechanisms exhibit severe stiffness such that, in practice, implicit integration schemes, typically of high order are applied.

In this light, we have carefully reimplemented in OpenCL C the Fortran 77 program of the 5th order accurate implicit Runge-Kutta method Radau5 by Hairer and Wanner [67] and tested it extensively in the context of a transient equilibrium scheme for the flamelet model. Our implementation can easily be integrated with any existing reacting flow software in order to solve the reaction fractional step on an OpenCL-enabled GPU. Moreover, it is suited for any Chemkin-format reaction mechanism with $\lesssim 200$ species without incurring a loss in GPU occupancy and it reaches its limit speedup (which is largely independent of the mechanism size) at a small problem size of ≈ 500 ODE systems. In view of memory constraints, we include an optimized scheme for splitting the solver call across several GPU invocations and overlapping the solver execution with data transfers. An in-depth evaluation is based on runtime measurements of the original CPU implementation and its GPU-based counterpart on a user level and a high-end CPU/GPU for an increasing number of grid points, reduced and detailed reaction mechanisms and time step sizes.

Solving the reaction fractional step on a CPU/GPU pair. In Chapter 5, we find that, for an implicit time integration scheme of high order, a GPU-based solver of the reaction fractional step runs at best approximately two times slower than a conventional CPU implementation. However, since most desktop computers encompass both a CPU and a GPU, it may be beneficial in practice to combine the CPU and GPU implementations and parallelize the reaction fractional step across both processors. In Chapter 6, we hence devise different strategies for parallelizing the reaction fractional step across a multi-core CPU and a GPU and assess the performance gain in the context of an LES of the Sandia D flame. The implementation here is modularized and may be readily incorporated into existing reacting flow solvers.

The developments in Chapters 2 through 6 have recently been reported in our publications [185–188] and the conference article [184]. The main content of these references (text, figures, tables) is reproduced here with minor modifications and amendments, while we have taken care to ensure that no input from the publishers (Elsevier, AIP Publishing) has been included.

As part of our research efforts, a second conference article has been published [58] recently. This conference article originated from a collaboration with C. E. Garcia-Gonzalez, A. Liu, S. Rigopoulos and B. A. O. Williams from the Department of Mechanical Engineering, Imperial College London, and the Department of Engineering Science, University of Oxford, respectively. Here, our objective was to obtain predictions of laser diagnostic signals from computed gas phase compositions and primary soot particle size distributions in a laminar co-flow diffusion flame [33] and to assess the influence of soot polydispersity on the recorded signals. It was found that the light scattered elastically from soot is not sensitive to the primary particle size distribution, supporting the common practice of considering soot particles as monodisperse for the interpretation of elastic light scattering signals.

Chapter 2

An explicit adaptive grid approach for solving the PBE

2.1 Introduction

From a Eulerian perspective, the evolution of the property distribution associated with a particulate phase that is immersed in a carrier flow can be described by the PBE. In the present chapter, we confine the attention to discretization-based methods for numerically solving the spatially inhomogeneous and unsteady PBE. Alternative approaches such as the method of moments and stochastic solution schemes have been reviewed by Ramkrishna [169] and Rigopoulos [173], for instance.

In the context of discretization-based methods, the PBE is commonly discretized on a *fixed* grid in particle property space and the resulting semi-discrete equations are solved by applying a standard Eulerian solution scheme [209]. Here, the semi-discrete system consists of scalar transport equations for so-called discrete number densities. Since these transport equations are formally identical to those of the reactive scalars which characterize the fluid phase, the PBE can be naturally incorporated into models for the laminar or turbulent carrier flow [39, 40, 173, 174].

While fixed grid discretization schemes are very mature, they frequently require an extremely fine grid throughout particle property space. This is particularly acute if the particle property distribution evolves over several orders of magnitude, potentially developing peaks or near-discontinuities. In commercial crystallizers, for instance, the particle size can span up to five orders of magnitude, ranging from the nucleation size $\sim 1\,\text{nm}$ to the final crystal size $\sim 100\,\mu\text{m}$. By consequence, the computational effort may become immense, in particular, if the particle property distribution evolves in a spatially inhomogeneous flow field.

Alternatively, the PBE can be discretized in particle property space by a moving or adaptive grid approach in which the local grid resolution varies dynamically. Here, the grid node locations in particle property space appear as additional dependent variables which are governed by a dynamical system that needs to be solved in conjunction with the PBE. Frequently, this dynamical system can be deduced from an evolution equation for a coordinate transformation which maps physical particle property space onto a transformed particle property space and is kinetically driven by the particle formation kinetics (moving grid approaches) or the current particle property distribution (adaptive grid approaches). In Section 2.2, we specifically review those moving and adaptive grid approaches which have been applied to the numerical solution of the spatially homogeneous PBE. The main persisting challenges associated with these solution schemes are that the size of the semi-discrete system significantly increases and that fixed grid source terms cannot be accommodated in a standard fashion.

In this chapter, we present an explicit adaptive grid technique which seeks to resolve these issues. The main idea is to prescribe the future time and space evolution of the coordinate transformation on particle property space explicitly in terms of recent solutions for the particle property distribution. Based on the prescribed coordinate transformation, the PBE can be reformulated in transformed particle property space and discretized there on a uniform reference grid. Formally, the coordinate transformation is constructed such that, at each spatial location, the current resolution in physical particle property space varies with, for instance, the gradient or curvature of a recent particle property distribution. In practice, this measure of variation is specified by a so-called monitor function.

As indicated above, one of the main challenges associated with grid-adaptive discretizations is related to the resolution of source terms which are localized in particle property space. If grid nodes move away from the nucleation size in a particle formation process, for instance, then the nucleation sources may be underresolved or become invisible to the adaptive grid. In addition, the accuracy and convergence of an adaptive grid scheme often rely on the grid spacings to change gently in particle property space. In order to address both of these points, we adopt the notion of a node density in physical particle property space [43] and impose conditions on the minimum admissible node density and the local change of node density [88]. In the context of particle formation, these conditions may be expressed in terms of two physical grid parameters, the minimum node density inside the nucleation interval and the maximum admissible grid stretching. Both of these values can be chosen based on experience, much like in the conventional generation of a fixed grid.

The contribution of this chapter is twofold: First, we develop an explicit adaptive grid technique for discretizing the spatially inhomogeneous and unsteady PBE in particle property space. Since here the motion of nodes in particle property space is prescribed explicitly, the system size of the semi-discrete PBE can be maintained. Second, we present a robust numerical scheme for accommodating localized source terms and preventing grid distortion by imposing conditions on the node density distribution in physical particle property space.

This chapter is organized as follows: In Sections 2.2 and 2.3, we review existing moving and adaptive grid schemes for discretizing the PBE in particle property space and

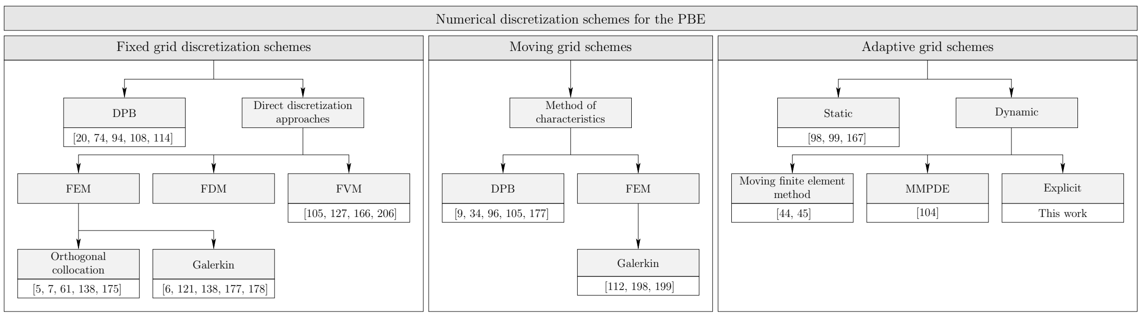


Figure 2.1 An overview of numerical schemes for discretizing the PBE in particle property space (DPB: Discretized Population Balance, FEM: Finite Element Method, FDM: Finite Difference Method, FVM: Finite Volume Method, MMPDE: Moving Mesh Partial Differential Equation).

contextualize the PBE as a model for particle formation in reacting carrier flows. Subsequently, in Section 2.4, a space and time dependent coordinate transformation on particle property space is introduced and the transformed PBE is presented. In Section 2.5, we formally construct the coordinate transformation and develop a scheme for adjusting the node density in physical particle property space. This is followed by Section 2.6 in which both the adaptive grid technique and its fixed grid counterparts are applied to a step advection example and a kinetically realistic test case for the precipitation of $BaSO_4$ in a plug flow reactor. Here, the accuracy, convergence behavior and computational efficiency of the adaptive grid method are critically assessed. Finally, we draw conclusions in Section 2.7 and embed our findings into the line of future work.

2.2 Review of moving and adaptive grid discretization schemes for the PBE

In the present section, we briefly review existing moving and adaptive[1] grid approaches which have been employed to discretize the spatially homogeneous PBE in particle property space. As an aid to the reader, these schemes are summarized and categorized in Figure 2.1, where, for completeness, we also include fixed grid discretization schemes.

In the context of the PBE, most moving grid approaches are based on a discretized population balance combined with the method of characteristics (or a slight variation thereof) which endorses the governing equations with a Lagrangian character. Within the scope of a discretized population balance, particle property space is partitioned into a finite number of bins and the particles in each bin are associated with a pivot property. The discrete equations then describe how the particles associated with one pivot collectively interact with the particles at other pivots and are often designed so as to exactly reproduce

[1]For clarity we term the combination of any fixed grid discretization approach with the method of characteristics a *moving* grid method, while schemes in which the node positions in particle property space are determined based on the current solution or its history are referred to as *adaptive* grid approaches.

particular moments of the particle property distribution.

One of the first numerical schemes which combined a discretized population balance with the method of characteristics is due to Kumar and Ramkrishna [96]. Here, the pivots propagate through particle size space at the local growth rate. Since pivots may naturally move away from the nuclei size, Kumar and Ramkrishna [96] proposed to frequently add a bin whose pivot is located at the nuclei size to the left of the leftmost bin. Roussos et al. [177], however, pointed out that the accuracy of the numerical scheme strongly depends on the bin addition frequency and that, for large frequencies, the number of bins quickly increases. As a counter measure, the grid may be coarsened at intervals [96] or, alternatively, for each bin added to the left, the rightmost bin may be deleted [105]. In this second technique, however, the conservation of particle mass cannot be guaranteed.

Considering a latex emulsion polymerization process, Crowley et al. [34] combined two discretized population balances for distinct particulate phases with a prescription for the motion of the pivots. Here, the velocity at which the largest pivots of the two discretized populations moved was given by the growth rate associated with one of the two pivots, while the velocity of the remaining pivots followed linearly. Physically, this approach is based on the rationale that particle nucleation would cease to be important as particles grow into larger size ranges and the grid expands.

Tsang and Brock [198] and Tsang and Rao [199] incorporated the method of characteristics into a combined particle property space and time Galerkin finite element formulation in which the grid nodes were confined to move along a first order approximation (in particle property space and time) of the local characteristics. This approach is based on a method proposed by Varoglu and Finn [201] who pointed out that the total number of nodes in particle property space may vary if the boundary conditions are such that more nodes are convected out of/into one boundary than enter/leave the opposite boundary.

By construction, the moving grid approaches reviewed above involve a dynamical system in which the pivot locations of the discretized population balance are kinetically driven by the local particle growth rate. In the absence of a growth mechanism, Kumar and Ramkrishna [95] showed that a similar system can also be constructed for pure aggregation[2] and breakage kinetics. Here, the pivots are allowed to move within the bin which they characterize, giving an indication of how the number density distribution varies across the bin as a result of aggregation and breakage events. As with moving grid approaches for growth dominated problems, however, the grid adaptivity here targets the accurate representation of one physical mechanism and may not be as well suited for other particulate processes. A more general approach in which the particle phase kinetics do not

[2]In line with most authors in the chemical engineering community, we refer to aggregation as an event in which two particles merge, yielding a particle with the same shape as the parent particles. In the literature on soot formation, by contrast, this process is more commonly referred to as coagulation and aggregation is reserved for a process in which the parent particles remain intact, while combining into a daughter particle with a new shape. (This second concept is adopted in Chapter 4.)

appear explicitly is that of a solution-adaptive grid to which we turn now.

The main idea underlying a solution-adaptive grid approach is to adjust the number or spatial distribution of nodes based on information about the *shape* of the current or a recent particle property distribution. One of the first forays in this direction is due to Kumar and Ramkrishna [94] who suggested to selectively refine/split those bins of a discretized population balance over which the number density varies significantly and to, conversely, remove/coalesce those bins over which the number density changes only slightly. If a bin is deleted, then its number density is distributed across the adjacent bins such that two particular moments are preserved exactly. This approach was adopted, for instance, by Lee et al. [99] who proposed a criterion for eliminating or refining bins based on the arc length monitor function. Following a slightly different approach, Lee et al. [98] enhanced a fixed grid finite difference discretization of the PBE by inserting or deleting nodes according to the local first and second derivatives of the particle property distribution.

If the number of grid points is kept constant, then the insertion/deletion schemes involve periodically redistributing the grid nodes and subsequently interpolating the solution from the old grid onto the new one. Such a scheme is termed a static adaptive grid method. An example is the two-step adaptive grid technique developed by Tang and Tang [197] and applied in the context of the PBE by Qamar et al. [167]. Here, the PBE is first evolved over a time step on a fixed grid using a high resolution finite volume scheme and a new grid is computed such that the nodes are equidistributed along a measure of variation of the particle property distribution. Subsequently, the discrete number densities associated with the new grid are obtained from the old ones by an interpolation scheme which conserves the first moment and prevents the creation of unphysical oscillations.

In dynamic adaptive grid schemes, on the other hand, node locations persist as dependent variables and are determined by a dynamical system which contains the particle property distribution as a kinetic variable. Frequently, this dynamical system is obtained as the semi-discrete counterpart of a so-called moving mesh partial differential equation (MMPDE) [76]. MMPDEs are formulated with respect to a transformed particle property space (sometimes referred to as computational domain) and govern the time evolution of a coordinate transformation which maps fixed grid points in transformed particle property space onto variable node locations in physical particle property space. In the context of the PBE, an MMPDE-based approach was adopted, for instance, by Lim et al. [104] who investigated the crystallization of potassium sulphate in a batch reactor.

Conceptually different from the dynamic adaptive grid methods based on an MMPDE is the moving finite element method developed by Miller and Miller [125] and applied to the numerical solution of the PBE by Duarte and Baptista [44, 45]. Here, both the nodal number densities and the node locations are considered as time dependent parameters defining a sufficiently smooth discrete particle property distribution. The time derivatives of these parameters are then determined such that the square norm of the PBE residual is minimized. In order to avoid both degeneracies in the parameterization and grid tangling the formulation is augmented by a penalty term which can be viewed as a source for viscous forces constraining the relative node movement.

Contrary to the moving and dynamic adaptive grid approaches reviewed above, we consider, in this chapter, an explicit relation between the coordinate transformation on particle property space and past solutions for the particle property distribution [37, 48, 49]. Here, the size of the semi-discrete system is maintained and the coordinate transformation can be modified quasi-statically, for instance, in order to accommodate fixed grid source terms or to prevent grid distortion.

2.3 The population balance equation

The description of a particulate phase as a continuous medium is often based on the number density of particles $N(\mathbf{l}, \mathbf{x}, t)$ per unit of mixture volume and per unit of volume in particle property space ($\mathbf{l}$-space) at a particular location $\mathbf{x}$ in the flow domain Ω and at time $t \geq 0$. From a Eulerian perspective, the evolution of the particle number density $N(\mathbf{l}, \mathbf{x}, t)$ is governed by the so-called population balance equation (PBE) which is sometimes also referred to as the general dynamic equation (GDE). Following Hulburt and Katz [77], the PBE can be derived from the Lagrangian laws which govern the motion of an individual particle in a way that is similar to the derivation of scalar conservation laws. In the present work, we confine the attention to a particle population which is characterized by the scalar property $\mathbf{l} \equiv l$. Without loss of generality, l can be conceived as a measure of particle size (characteristic length) such that $l \in [0, \infty)$.

If the particulate phase is immersed in a fluid with composition $\mathbf{Y}(\mathbf{x}, t)$ and the fluid flow is described by the velocity field $\mathbf{u}(\mathbf{x}, t)$, then the PBE is given by

$$\begin{aligned} \frac{\partial N(l,\mathbf{x},t)}{\partial t} &+ \sum_{j=1}^{3} \frac{\partial \left(u_j(\mathbf{x},t) N(l,\mathbf{x},t)\right)}{\partial x_j} + \frac{\partial \left(G(l,\mathbf{Y}(\mathbf{x},t)) N(l,\mathbf{x},t)\right)}{\partial l} \\ &= -\sum_{j=1}^{3} \frac{\partial K_j}{\partial x_j} + \dot{s}(l, \mathbf{Y}(\mathbf{x},t), N(\cdot,\mathbf{x},t)), \end{aligned} \tag{2.1}$$

where $K_j(\mathbf{x}, t)$ represents the diffusive flux of number density along the jth coordinate direction in physical space,

$$K_j(\mathbf{x},t) = -D_p(\mathbf{x},t) \frac{\partial N(l,\mathbf{x},t)}{\partial x_j}, \tag{2.2}$$

$G(l, \mathbf{Y}(\mathbf{x}, t))$ denotes the particle growth or shrinkage rate, $D_p(\mathbf{x}, t)$ is the common particle diffusivity and $\dot{s}(l, \mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t))$ represents a source/sink term which accounts for the processes of particle nucleation, aggregation and breakage. Note that, in general, $\dot{s}(l, \mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t))$ is a functional in the local particle property distribution $N(\cdot, \mathbf{x}, t)$.

Physically, the first term on the left hand side of Eq. (2.1) describes particle accumulation at $(l, \mathbf{x}, t)$, while the second and third terms correspond to particle advection in physical and particle property space and the first term on the right hand side represents diffusion of particles in physical space.

Eq. (2.1) is complemented by the initial condition

$$N(l, \mathbf{x}, 0) = N_\Omega(l, \mathbf{x}), \tag{2.3}$$

where $N_\Omega(l, \mathbf{x})$ represents a prescription of the particle property distribution at each location $\mathbf{x}$ at time $t = 0$. Along the spatial boundary of the domain, standard Dirichlet and/or Neumann boundary conditions (BCs) apply, while a homogeneous Dirichlet boundary condition holds at $l = 0$ [175],

$$N(0, \mathbf{x}, t) = 0. \tag{2.4}$$

2.4 A space and time dependent coordinate transformation on particle property space

In the present section, we formally introduce a space and time dependent coordinate transformation on particle property space and obtain transport equations for the transformed particle number density within the scope of a fractional steps scheme. For the time being, we consider the coordinate transformation as given; its explicit construction is addressed in detail in Section 2.5.

2.4.1 The transformed PBE

In view of the following developments, we restrict the semi-infinite particle property space $[0, \infty)$ to the interval $[0, L]$, where L denotes the maximum attainable value of the particle property. In particular, we suppose that L is chosen sufficiently large such that the particle property distribution remains contained in $[0, L]$. In applications involving aggregation and breakage, a so-called finite domain error is incurred if the particle property distribution leaves $[0, L]$, see, for instance, Gelbard and Seinfeld [61] and Attarakih et al. [9].

We begin by introducing a four parameter family of C^1-continuous coordinate transformations $l = \bar{l}(\tau, \mathbf{x}, t)$, $\bar{l} : [0, L] \times \Omega \times \mathbb{R}_0^+ \to [0, L]$ which map, at each physical location $\mathbf{x}$ and point in time t, a transformed particle size τ onto the physical particle size l and satisfy

$$\frac{\partial \bar{l}(\tau, \mathbf{x}, t)}{\partial \tau} \equiv w(\tau, \mathbf{x}, t) > 0 \quad \forall \mathbf{x} \in \Omega, t \geq 0 \tag{2.5}$$

as well as

$$\bar{l}(0, \mathbf{x}, t) = 0, \tag{2.6}$$

$$\bar{l}(L, \mathbf{x}, t) = \int_0^L w(s, \mathbf{x}, t)\, ds = L \quad \forall \mathbf{x} \in \Omega, t \geq 0. \tag{2.7}$$

Jointly, Eqs. (2.5) through (2.7) ensure that $\bar{l}(\cdot, \mathbf{x}, t)$ is one-to-one and onto. Associated with $\bar{l}$ is an inverse C^1-function $\bar{\tau} \equiv \bar{l}^{-1}$ which may be defined point-wise according to

$$\bar{\tau}(\bar{l}(\tau, \mathbf{x}, t), \mathbf{x}, t) = \tau \quad \forall \tau \in [0, L], \mathbf{x} \in \Omega, t \geq 0. \tag{2.8}$$

For future reference, we record the identities[3]

$$\left.\frac{\partial \bar{\tau}}{\partial l}\right|_{\bar{l}} = \frac{1}{w}, \tag{2.9}$$

$$\left.\frac{\partial \bar{\tau}}{\partial t}\right|_{\bar{l}} = -\frac{1}{w}\frac{\partial \bar{l}}{\partial t}, \tag{2.10}$$

$$\left.\frac{\partial \bar{\tau}}{\partial x_j}\right|_{\bar{l}} = -\frac{1}{w}\frac{\partial \bar{l}}{\partial x_j} \tag{2.11}$$

and

$$\left.\frac{\partial^2 \bar{\tau}}{\partial x_j^2}\right|_{\bar{l}} = \frac{2}{w^2}\frac{\partial^2 \bar{l}}{\partial x_j \partial \tau}\frac{\partial \bar{l}}{\partial x_j} - \frac{1}{w}\frac{\partial^2 \bar{l}}{\partial x_j^2} - \frac{1}{w^3}\frac{\partial^2 \bar{l}}{\partial \tau^2}\left(\frac{\partial \bar{l}}{\partial x_j}\right)^2 \tag{2.12}$$

with $j = 1, \ldots, 3$ which are readily obtained from Eqs. (2.5) and (2.8).

Defining the particle property distribution $f(\tau, \mathbf{x}, t)$ in transformed particle property space by

$$N(l, \mathbf{x}, t) \equiv f(\bar{\tau}(l, \mathbf{x}, t), \mathbf{x}, t) \tag{2.13}$$

and introducing Eq. (2.13) into Eq. (2.1), we obtain on account of the chain rule and after evaluation at $l = \bar{l}(\tau, \mathbf{x}, t)$

$$\begin{aligned} &\frac{\partial f}{\partial t} + \sum_{j=1}^{3} \frac{\partial (u_j f)}{\partial x_j} + f \left.\frac{\partial G}{\partial l}\right|_{\bar{l}} + \frac{\partial f}{\partial \tau}\Bigg(G \left.\frac{\partial \bar{\tau}}{\partial l}\right|_{\bar{l}} + \left.\frac{\partial \bar{\tau}}{\partial t}\right|_{\bar{l}} \\ &\quad + \sum_{j=1}^{3}\left(u_j - \frac{\partial D_p}{\partial x_j}\right)\left.\frac{\partial \bar{\tau}}{\partial x_j}\right|_{\bar{l}} - \sum_{j=1}^{3} D_p \left.\frac{\partial^2 \bar{\tau}}{\partial x_j^2}\right|_{\bar{l}} \Bigg) \\ &= \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(D_p \frac{\partial f}{\partial x_j}\right) + D_p \sum_{j=1}^{3}\left(\frac{\partial^2 f}{\partial \tau^2}\left.\frac{\partial \bar{\tau}}{\partial x_j}\right|_{\bar{l}}^2 + 2\frac{\partial^2 f}{\partial \tau \partial x_j}\left.\frac{\partial \bar{\tau}}{\partial x_j}\right|_{\bar{l}}\right) + \dot{s}(\bar{l}, \mathbf{Y}, f). \end{aligned} \tag{2.14}$$

In the reacting fluid flow community, it is common to apply the method of fractional steps in order to isolate the different physical phenomena which affect the evolution of the transported fields [211]. One advantage of this approach is that different solution schemes can be applied which are tailored to meet the physical characteristics of each fractional

[3]Throughout this work, we employ the long vertical line succeeding a function as shorthand notation for “evaluated at”.

step. The particular operator splitting approach applied in the following implements a first order approximation in time [161].

By collecting the terms with leading spatial derivatives of $f(\tau, \mathbf{x}, t)$ (the second term on the left hand side of Eq. (2.14) and the first term on its right hand side) and combining them with the accumulation term (the first term on the left hand side of Eq. (2.14)), the prescription for the convection-diffusion fractional step is obtained

$$\frac{\partial f}{\partial t} + \sum_{j=1}^{3} \frac{\partial (u_j f)}{\partial x_j} = \sum_{j=1}^{3} \frac{\partial}{\partial x_j} \left(D_p \frac{\partial f}{\partial x_j} \right). \tag{2.15}$$

The remaining terms in Eq. (2.14) combine with the accumulation term to yield the equation governing the PBE fractional step. Taking into account Eqs. (2.9) through (2.12), we obtain

$$\begin{aligned} &\frac{\partial f}{\partial t} + f \left. \frac{\partial G}{\partial l} \right|_{\bar{l}} + \frac{1}{w} \frac{\partial f}{\partial \tau} \left(G|_{\bar{l}} - \frac{\partial \bar{l}}{\partial t} - \sum_{j=1}^{3} \left(u_j - \frac{\partial D_p}{\partial x_j} \right) \frac{\partial \bar{l}}{\partial x_j} + D_p b \right) \\ &= \frac{D_p}{w^2} \frac{\partial}{\partial \tau} \left(\frac{\partial f}{\partial \tau} \sum_{j=1}^{3} \left(\frac{\partial \bar{l}}{\partial x_j} \right)^2 \right) - \frac{2 D_p}{w} \sum_{j=1}^{3} \left(\frac{\partial^2 f}{\partial \tau \partial x_j} \frac{\partial \bar{l}}{\partial x_j} \right) + \dot{s}(\bar{l}, \mathbf{Y}, f), \end{aligned} \tag{2.16}$$

where $b(\tau, \mathbf{x}, t)$ collects the second order derivatives of $\bar{l}(\tau, \mathbf{x}, t)$,

$$b(\tau, \mathbf{x}, t) = \sum_{j=1}^{3} \frac{\partial^2 \bar{l}}{\partial x_j^2} + \frac{1}{w^2} \frac{\partial^2 \bar{l}}{\partial \tau^2} \sum_{j=1}^{3} \left(\frac{\partial \bar{l}}{\partial x_j} \right)^2. \tag{2.17}$$

Physically, the terms involving spatial derivatives of $\bar{l}(\tau, \mathbf{x}, t)$ in Eq. (2.16) induce a redistribution of number density in τ-space which compensates for the transport of number density in physical space along lines of constant $\tau \in [0, L]$ (Eq. (2.15)) as opposed to lines of constant $l \in [0, L]$ (as in Eq. (2.1)). In this vain, we observe that diffusion in physical space naturally gives rise to diffusion in particle property space.

By virtue of the coordinate transformation, particle property distributions at different points in space and time can be endowed with different parameterizations ($l = \bar{l}(\tau, \mathbf{x}, t)$) and the individual parameterizations referred to a unique reference coordinate (τ) which remains consistent across the flow domain. This approach differs from the solution procedure of Campos and Lage [30], for instance, who applied a moving grid technique to the PBE fractional step and, subsequently, mapped the particle property distributions at every spatial grid point onto a common grid in physical particle property space such that spatial consistency for the convection/diffusion fractional step was restored. Here, the mapping operation was designed so as to preserve the total particle number and volume densities.

2.4.2 Semi-discrete representation

In this section, we present the time-continuous equations obtained by discretizing Eqs. (2.15) and (2.16) both in transformed particle property space ($\tau \in [0, L]$) and in physical space ($\mathbf{x} \in \Omega$). In view of the numerical experiments in Section 2.6, we specifically apply a high resolution finite volume scheme in transformed particle property space. Moreover, since many scientific and commercial fluid flow solvers are based on a finite volume discretization in physical space, the **x**-discrete formulation is presented for this type of discretization. Conceptually, however, our explicit adaptive grid approach is independent of the specific $(\tau, \mathbf{x})$ discretization scheme and can be combined with other discretization schemes than the ones considered here.

As a first step, we introduce a fixed uniform grid in τ-space,

$$\tau_i = (i-1)\Delta\tau, \quad \Delta\tau = \frac{L}{n_p}, \quad i = 1, \dots, n_p + 1, \tag{2.18}$$

where $n_p + 1$ denotes the number of nodes and $\Delta\tau$ represents the uniform grid spacing, and let

$$\bigcup_{j=1}^{n_c} \Omega_j = \Omega \tag{2.19}$$

denote a partitioning of Ω into n_c subdomains Ω_j with volume V_j and midpoints $\mathbf{x}_j$.

Since our objective is to devise a robust and computationally efficient numerical scheme, we focus attention on the case in which the coordinate transformation $\bar{l}(\cdot, \mathbf{x}, t)$ varies piecewise linearly on the τ-grid in Eq. (2.18).[4] In general, however, $\bar{l}(\cdot, \mathbf{x}, t)$ can also take on the form of a higher order polynomial. This may be beneficial if the τ-discretization scheme is based on a higher order polynomial approximation of $f(\cdot, \mathbf{x}, t)$.

In the context of a finite volume discretization in τ-space, we consider the average particle number density $f_i(\mathbf{x}, t)$ on $[\tau_i, \tau_{i+1}]$ at $(\mathbf{x}, t)$,

$$f_i(\mathbf{x}, t) = \frac{1}{\Delta\tau} \int_{\tau_i}^{\tau_{i+1}} f(\tau, \mathbf{x}, t)\, d\tau, \quad i = 1, \dots, n_p, \tag{2.20}$$

[4]For many direct discretization schemes, the τ-discrete representation of Eq. (2.16) does not require C^1-continuity of $\bar{l}(\cdot, \mathbf{x}, t)$ at the nodes τ_i.

and deduce by integrating Eq. (2.16) over $[\tau_i, \tau_{i+1}]$

$$\begin{aligned}
&\frac{\partial f_i}{\partial t} + \frac{1}{w_i \Delta\tau} \left[f \left(G(\bar{l}, \mathbf{Y}) - g \right) \right]_{\tau_i}^{\tau_{i+1}} \\
&= \frac{D_p}{w_i^2 \Delta\tau^2} \sum_{k=1}^{3} \left((f_{i+1} - f_i) \left(\frac{\partial \bar{l}}{\partial x_k} \right)^2 \Bigg|_{\tau_{i+1}} - (f_i - f_{i-1}) \left(\frac{\partial \bar{l}}{\partial x_k} \right)^2 \Bigg|_{\tau_i} \right) \\
&\quad - \frac{D_p}{w_i \Delta\tau} \sum_{k=1}^{3} \left(\frac{\partial (f_{i+1} - f_i)}{\partial x_k} \frac{\partial \bar{l}}{\partial x_k} \Bigg|_{\tau_{i+1}} + \frac{\partial (f_i - f_{i-1})}{\partial x_k} \frac{\partial \bar{l}}{\partial x_k} \Bigg|_{\tau_i} \right) \\
&\quad - \frac{f_i}{w_i \Delta\tau} \left[g|_{\tau_{i+1}} - g|_{\tau_i} \right] + \frac{1}{\Delta\tau} \int_{\tau_i}^{\tau_{i+1}} \dot{s}|_{\bar{l}} \, d\tau,
\end{aligned} \tag{2.21}$$

where $g(\tau, \mathbf{x}, t)$ encompasses the contribution of $\bar{l}(\tau, \mathbf{x}, t)$ to the cumulative growth rate,

$$g(\tau, \mathbf{x}, t) = \frac{\partial \bar{l}}{\partial t} + \sum_{k=1}^{3} \left(u_k - \frac{\partial D_p}{\partial x_k} \right) \frac{\partial \bar{l}}{\partial x_k} - D_p b, \tag{2.22}$$

and $w_i(\mathbf{x}, t)$ denotes the constant value of $w(\tau, \mathbf{x}, t)$ on $[\tau_i, \tau_{i+1}]$. The inter-cell fluxes $f(G(\bar{l}, \mathbf{Y}) - g)|_{\tau_{i+1}}$ can, for instance, be computed according to the flux-limited method of Koren [92]

$$f \left(G(\bar{l}, \mathbf{Y}) - g \right) \big|_{\tau_i} = \left(G(\bar{l}, \mathbf{Y}) - g \right) \big|_{\tau_i} \left(f_{u(\tau_i)} + \frac{\phi(\tau_i, \mathbf{x}, t)}{2} \left(f_{u(\tau_i)} - f_{uu(\tau_i)} \right) \right), \tag{2.23}$$

where $u(\tau_i)$ and $uu(\tau_i)$ indicate the indices of the two cells upstream of node τ_i and $\phi(\tau_i, \mathbf{x}, t)$ denotes the flux limiter at cell face τ_i [166].

In order to discretize Eq. (2.21) in physical space, we adopt a finite volume discretization scheme based on the domain partitioning given in Eq. (2.19). Here, the average discrete number density $F_{i,j}(t)$ on cell Ω_j is given by

$$F_{i,j}(t) \equiv \langle f_i(\mathbf{x}, t) \rangle_{\Omega_j} \equiv \frac{1}{V_j} \int_{\Omega_j} f_i(\mathbf{x}, t) \, d\mathbf{x}, \tag{2.24}$$

where the angled brackets indicate averaging over cell Ω_j. By integrating Eq. (2.21) over Ω_j and approximating $f_i(\mathbf{x}, t)$ on Ω_j by its cell-average $F_{i,j}(t)$, we obtain the semi-discrete representation

$$\begin{aligned}
&\langle w_i\rangle_{\Omega_j}\frac{dF_{i,j}}{dt}+\frac{1}{\Delta\tau}\left[\left\langle G(\bar{l},\mathbf{Y})-g\right\rangle_{\Omega_j}\left(F_{u(\tau),j}+\frac{\phi(\tau,\mathbf{x},t)}{2}\left(F_{u(\tau),j}-F_{uu(\tau),j}\right)\right)\right]_{\tau_i}^{\tau_{i+1}}\\
&\quad=\frac{1}{\Delta\tau}\Big[\left(F_{i+1,j}-F_{i,j}\right)\left(a_{i,j}+b_{i,j}\right)\big|_{\tau_{i+1}}\\
&\qquad\qquad-\left(F_{i,j}-F_{i-1,j}\right)\left(a_{i,j}-b_{i,j}\right)\big|_{\tau_i}\Big]\\
&\quad-\frac{1}{\Delta\tau V_j}\int_{\partial\Omega_j}\sum_{k=1}^{3}\left[\left(f_{i+1}-f_i\right)D_p\left.\frac{\partial\bar{l}}{\partial x_k}\right|_{\tau_{i+1}}\right.\\
&\qquad\qquad\left.+\left(f_i-f_{i-1}\right)D_p\left.\frac{\partial\bar{l}}{\partial x_k}\right|_{\tau_i}\right]n_k\,d\mathbf{x}\\
&\quad-\frac{F_{i,j}}{\Delta\tau}\left[\langle g\rangle_{\Omega_j}\Big|_{\tau_{i+1}}-\langle g\rangle_{\Omega_j}\Big|_{\tau_i}\right]+\frac{1}{\Delta\tau}\left\langle w_i\int_{\tau_i}^{\tau_{i+1}}\dot{s}|_{\bar{l}}\,d\tau\right\rangle_{\Omega_j}
\end{aligned}\tag{2.25}$$

for $i=1,\dots,n_p$ and $j=1,\dots,n_c$. In Eq. (2.25), $\mathbf{n}(\mathbf{x})$ denotes the outward unit normal vector on the cell surface $\partial\Omega_j$ and $a_{i,j}$ and $b_{i,j}$ are, respectively, given by

$$a_{i,j}|_{\tau_i}=\sum_{k=1}^{3}\frac{1}{\Delta\tau}\left\langle\frac{D_p}{w_i}\left.\left(\frac{\partial\bar{l}}{\partial x_k}\right)^2\right|_{\tau_i}\right\rangle_{\Omega_j},\tag{2.26}$$

$$b_{i,j}|_{\tau_i}=\sum_{k=1}^{3}\left\langle\frac{\partial}{\partial x_k}\left(D_p\left.\frac{\partial\bar{l}}{\partial x_k}\right|_{\tau_i}\right)\right\rangle_{\Omega_j}.\tag{2.27}$$

For a particular cell geometry, the surface integral in Eq. (2.25) can be expressed in terms of $F_{i-1,j}$, $F_{i,j}$, $F_{i+1,j}$ and the neighboring cell values. Eq. (2.25) then constitutes a possibly non-linear system of ODEs for the time evolution of the cell averages $F_{i,j}(t)$. If the surface integral term is evaluated at the most recent point in time for which the $F_{i,j}(t)$ have been computed, then Eq. (2.25) can be solved independently for each finite volume cell j. Formally, this corresponds to applying a forward Euler step in time to the surface integral term.

For completeness, we also record the semi-discrete formulation of the convection-diffusion fractional step in Eq. (2.15),

$$\frac{dF_{i,j}}{dt}+\frac{1}{V_j}\int_{\partial\Omega_j}\sum_{k=1}^{3}\left(u_kf_i-D_p\frac{\partial f_i}{\partial x_k}\right)n_k\,d\mathbf{x}=0.\tag{2.28}$$

Figure 2.2 illustrates the main algorithmic workflow of the semi-discrete fractional steps scheme detailed above.

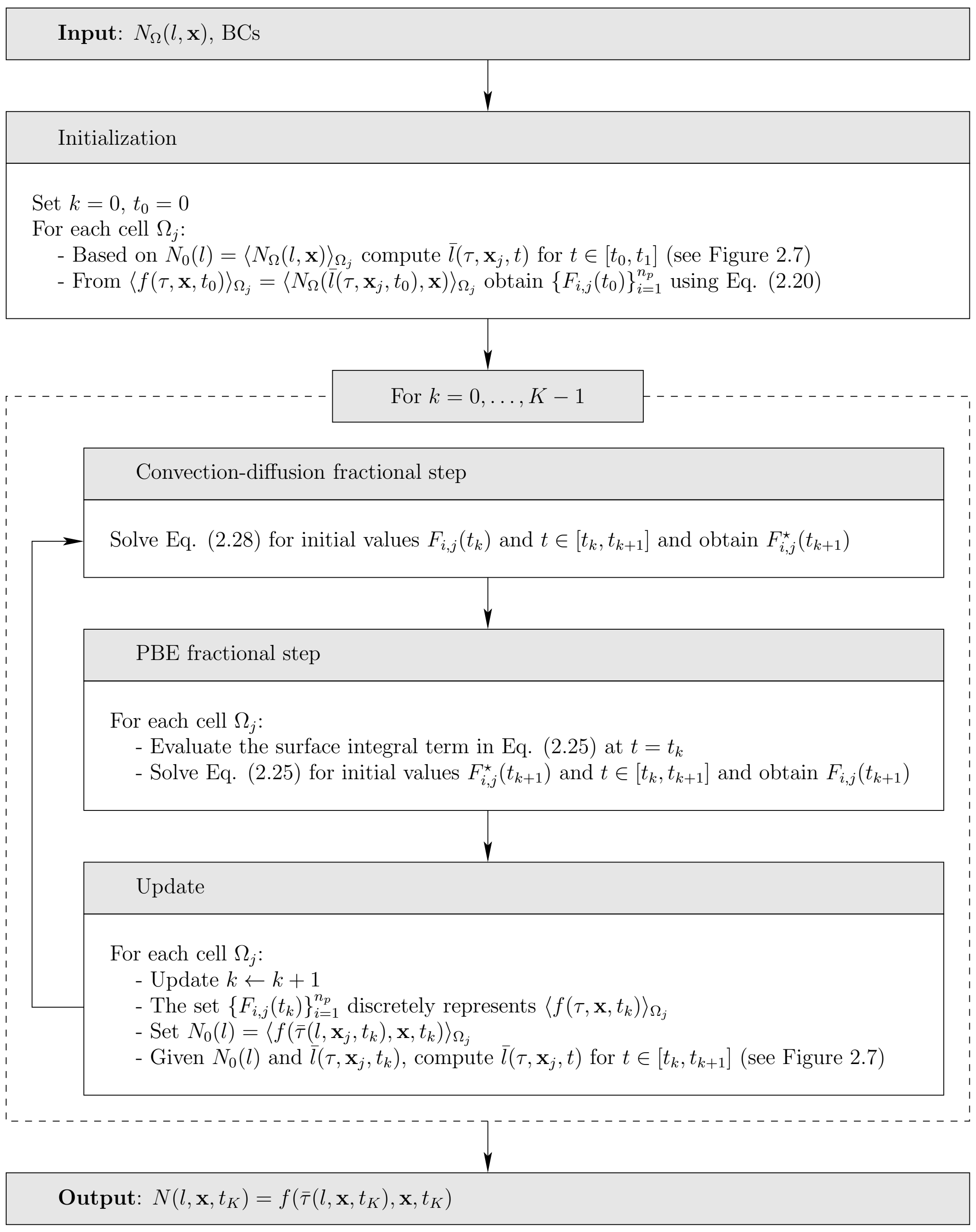

Input: $N_\Omega(l,\mathbf{x})$, BCs
Initialization
Set $k=0$, $t_0=0$
For each cell Ω_j:
- Based on $N_0(l)=\langle N_\Omega(l,\mathbf{x})\rangle_{\Omega_j}$ compute $\bar{l}(\tau,\mathbf{x}_j,t)$ for $t\in[t_0,t_1]$ (see Figure 2.7)
- From $\langle f(\tau,\mathbf{x},t_0)\rangle_{\Omega_j}=\langle N_\Omega(\bar{l}(\tau,\mathbf{x}_j,t_0),\mathbf{x})\rangle_{\Omega_j}$ obtain $\{F_{i,j}(t_0)\}_{i=1}^{n_p}$ using Eq. (2.20)
For $k=0,\ldots,K-1$
Convection-diffusion fractional step
Solve Eq. (2.28) for initial values $F_{i,j}(t_k)$ and $t\in[t_k,t_{k+1}]$ and obtain $F^\star_{i,j}(t_{k+1})$
PBE fractional step
For each cell Ω_j:
- Evaluate the surface integral term in Eq. (2.25) at $t=t_k$
- Solve Eq. (2.25) for initial values $F^\star_{i,j}(t_{k+1})$ and $t\in[t_k,t_{k+1}]$ and obtain $F_{i,j}(t_{k+1})$
Update
For each cell Ω_j:
- Update $k\leftarrow k+1$
- The set $\{F_{i,j}(t_k)\}_{i=1}^{n_p}$ discretely represents $\langle f(\tau,\mathbf{x},t_k)\rangle_{\Omega_j}$
- Set $N_0(l)=\langle f(\bar{\tau}(l,\mathbf{x}_j,t_k),\mathbf{x},t_k)\rangle_{\Omega_j}$
- Given $N_0(l)$ and $\bar{l}(\tau,\mathbf{x}_j,t_k)$, compute $\bar{l}(\tau,\mathbf{x}_j,t)$ for $t\in[t_k,t_{k+1}]$ (see Figure 2.7)
Output: $N(l,\mathbf{x},t_K)=f(\bar{\tau}(l,\mathbf{x},t_K),\mathbf{x},t_K)$

Figure 2.2 Schematic illustration of the algorithmic steps for numerically solving the spatially inhomogeneous PBE using the explicit adaptive grid approach (EAGM). The semi-discrete equations referenced here are detailed in Section 2.4.2.

2.5 An explicit adaptive grid approach

In the present section, we develop a semi-analytical expression for the coordinate transformation $\bar{l}(\tau, \mathbf{x}, t)$ on particle property space which is marched in time. To this end, we first consider the task of computing a space and time independent coordinate transformation $\bar{l}_0(\tau)$ or $\bar{l}_\infty(\tau)$, respectively, from a given reference particle property distribution $N_0(l)$ in isolation (Sections 2.5.1 and 2.5.2). Based on the formalism developed here, the future time evolution of $\bar{l}(\tau, \mathbf{x}, t)$ for $t \in [t_k, t_{k+1}]$ can then be prescribed in terms of the current solution for the transformed number density $f(\tau, \mathbf{x}, t_k)$ and the current coordinate transformation $\bar{l}(\tau, \mathbf{x}, t_k)$ (Section 2.5.3).

2.5.1 The equidistribution principle

In this section, we consider the coordinate transformation $\bar{l}_0(\tau)$ for a steady and spatially homogeneous reactor and develop a kinetic expression for the coordinate transformation which is based on the equidistribution principle for one-dimensional adaptive grid methods. The main idea is to vary the parameterization of particle property space in accordance with the local information content of a given reference particle property distribution $N_0(l)$ as the controlling kinetic variable. To this end, the Jacobian of $\bar{l}_0(\tau)$ is linked to a monitor function $m_{N_0}(l)$,

$$\frac{d\bar{l}_0(\tau)}{d\tau} \equiv \frac{c}{m_{N_0}(\bar{l}_0(\tau))}, \tag{2.29}$$

where $c > 0$ represents a normalization constant and $m_{N_0}(l) > 0$ is a functional which quantifies the information content of $N_0(l)$ at $l \in [0, L]$.

By setting $\tau = \bar{\tau}_0(l)$ in Eq. (2.29) and taking into account Eqs. (2.6), (2.8) and (2.9), we obtain upon integration from 0 to l

$$\bar{\tau}_0(l) = \frac{1}{c} \int_0^l m_{N_0}(u)\, du. \tag{2.30}$$

Introducing the boundary condition $\bar{\tau}_0(L) = L$ into Eq. (2.30) now leads to the following equation for the normalization constant c

$$c = \frac{1}{L} \int_0^L m_{N_0}(u)\, du. \tag{2.31}$$

Eq. (2.30) is frequently referred to as the *equidistribution principle* which is attributed to Boor [25]. It ensures that a scalar measure $m_{N_0}(l)$ over l-space is uniformly distributed in τ-space. As an example, we consider the arc length monitor function

$$m_{N_0}(l) = \sqrt{1 + \left(\frac{dN_0(l)}{dl}\right)^2} \tag{2.32}$$

for which the differential element $c\,d\bar{\tau}_0(l) = m_{N_0}(l)\,dl$ corresponds to ds, the local arc length increment along the curve $(l, N_0(l))$. For a representative reference particle property distribution $N_0(l)$, Figure 2.3 illustrates a l-grid that equidistributes arc length in τ-space. One disadvantage associated with the arc length monitor is that transition regions between sharp and smooth features may be underresolved [24]. As an alternative, we hence propose the following one-parameter family of monitor functions

$$m_{N_0}(l) = \sqrt{1 + b\left|\frac{1}{M_0(N_0)}\frac{dN_0(l)}{dl}\right|}, \tag{2.33}$$

where $b > 0$ is a regularization constant and $M_0(N_0)$ represents a normalization factor,

$$M_0(N_0) = \begin{cases} \int_0^L N_0(u)\,du & \text{if } \int_0^L N_0(u)\,du \neq 0 \\ 1 & \text{otherwise} \end{cases} . \tag{2.34}$$

The magnitude of b in Eq. (2.33) controls the extent to which regions in l-space over which $dN_0(l)/dl$ vanishes are compressed on a linear scale, while $M_0(N_0)$ ensures that the monitor function only depends on the *shape* of the reference particle property distribution $N_0(l)$ and is independent of its *scale*.

In comparison to the arc length monitor function, Eq. (2.33) places more emphasis on low gradient regions which frequently preceed the transition to a sharp feature. While the arc length monitor function varies approximately linearly in the local gradient, Eq. (2.33) increases as the square root of the local gradient. This implies that the larger a local gradient is, the smaller a relative increase of information content is registered. In a way, Eq. (2.33) aims at imitating the curvature monitor function which is based on second derivatives of the reference particle property distribution $N_0(l)$ [24]. However, if derivatives of order two or higher are included in a monitor function, then this naturally leads to the question with which accuracy the discrete representation of $N_0(l)$ allows these higher derivatives to be estimated. For example, if $N_0(l)$ varies piecewise linearly in l, then the accuracy with which second derivatives, as in the curvature monitor function, can be estimated is limited.

Dorfi and Drury [43] argue that, in general, the monitor function ought to be chosen in such a way that the discretization error incurred in l-space is uniformly distributed in τ-space. In particular, if $N_0(l)$ is represented in discrete terms by an interpolating piecewise Lagrange polynomial of order n, then the approximation error scales as [36, Section 8.2.5, 43]

$$h^{n+1}\frac{d^{n+1}N_0(l)}{dl^{n+1}}, \tag{2.35}$$

where h represents the local grid spacing. By consequence, monitor functions which equidistribute a local approximation error generally incur the challenge of approximating $(n + 1)$th order derivatives based on a piecewise nth order polynomial. The monitor

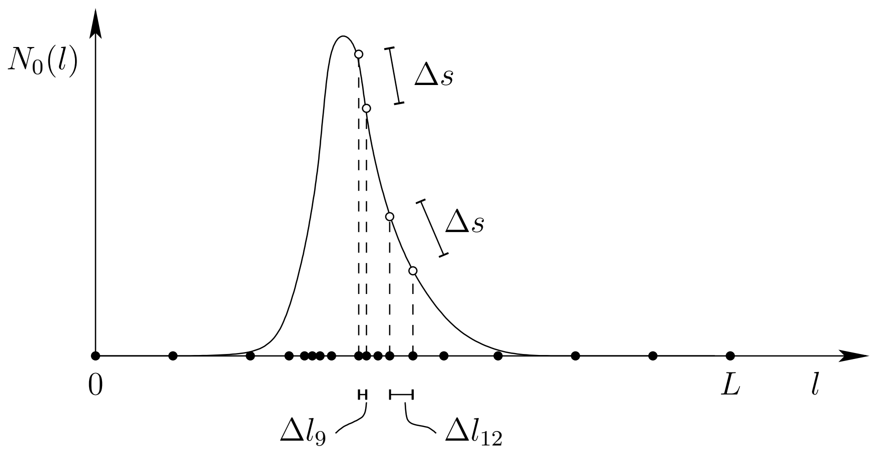


Figure 2.3 Schematic illustration of an arc length equidistributing grid. Here, Δs represents a uniform arc length increment along the graph of $N_0(l)$ and $\Delta l_i \equiv \bar{l}(\tau_{i+1}) - \bar{l}(\tau_i)$, $i = 1, \ldots, 17$, is the variable grid spacing.

function which we propose in Eq. (2.33) represents a compromise that, rather heuristically, aims at circumventing the shortcomings of the arc length monitor by mimicking the curvature monitor function, while avoiding the introduction of second order derivatives of $N_0(l)$.

2.5.2 An adjustment scheme for the node density

In many physical processes, the inception of new particles from the carrier fluid phase is modeled as a localized source term in physical particle property space. In the context of adaptive grid approaches, the distribution of nodes in particle property space varies and, by consequence, the nucleation interval in particle property space may not always contain enough nodes to accurately resolve the source term. Currently, the main strategy to prevent underresolution of the nucleation source term consists in manually adding nodes in the nucleation interval [96]. However, if at the same time nodes are removed elsewhere, then the conservation properties of the adaptive grid scheme may be impaired [105, 177].

In addition to the node distribution in the nucleation interval, the accuracy and convergence properties of an adaptive grid method are also influenced by the spatial smoothness of the grid in l-space [88]. Frequently, the spatial smoothness of a grid $0 = l_1 < \ldots < l_{n_p+1} = L$ is expressed in terms of the condition of local boundedness [43, 88],

$$\frac{1}{r} \leq \frac{l_{i+1} - l_i}{l_i - l_{i-1}} \leq r, \quad i = 2, \ldots, n_p, \tag{2.36}$$

where $r > 0$ represents the maximum admissible grid stretching in l-space (for example, ≈ 2). In the present section, we develop a scheme by which the equidistributing coordinate transformation $\bar{\tau}_0(l)$ of Section 2.5.1 can be modified such that the density of grid nodes in l-space does not fall below a prescribed minimum node density $\rho_{\min}(l)$ at any $l \in [0, L]$

and the l-grid nodes obey Eq. (2.36).

Since the τ-grid in Eq. (2.18) is uniform, we consider the node density $\rho_0(l)$ in l-space as the image of a uniform node density ρ_τ in τ-space under the action of $\bar{\tau}_0(l)$,

$$\rho_0(l) \equiv \frac{d\bar{\tau}_0(l)}{dl}\rho_\tau, \tag{2.37}$$

where

$$\rho_\tau \equiv \frac{n_p + 1}{L} \tag{2.38}$$

denotes the uniform node density in τ-space. Eq. (2.37) is a generalization of the discrete point concentration introduced by Dorfi and Drury [43]. Integrating Eq. (2.37) and taking into account the left boundary condition $\bar{\tau}_0(0) = 0$ yields

$$\int_0^l \rho_0(u)\, du = \bar{\tau}_0(l)\rho_\tau. \tag{2.39}$$

In order to accurately represent source terms which are localized in l-space, we let a minimum admissible node density $\rho_{\min}(l) \geq 0$ be prescribed throughout l-space and aim to find an *adjusted* coordinate transformation $\bar{\tau}_\infty(l)$ such that

$$\rho_\infty(l) = \frac{d\bar{\tau}_\infty(l)}{dl}\rho_\tau \geq \rho_{\min}(l) \quad \forall l \in [0, L] \tag{2.40}$$

and the main features of $\bar{\tau}_0(l)$ are preserved. Here, $\rho_{\min}$ is assumed to be at least piecewise continuous. Following Kautsky and Nichols [88], we additionally introduce the padding $P_{\rho_0}(l)$ of the node density $\rho_0(l)$

$$P_{\rho_0}(l) \equiv \max_{l' \in [0,L]} G_{\rho_0}(l, l'), \tag{2.41}$$

where, for a given l', $G_{\rho_0}(l, l')$ represents a function $\sim 1/(|l - l'|)$ with a tip at $(l', \rho_0(l'))$,

$$G_{\rho_0}(l, l') \equiv \frac{\rho_0(l')}{1 + \lambda|l - l'|\rho_0(l')}. \tag{2.42}$$

By Eq. (2.41), the padding $P_{\rho_0}(l)$ of $\rho_0(l)$ appears as the envelope of all functions $G_{\rho_0}(\cdot, l')$, $l' \in [0, L]$, whose tips trace the graph of the node density $\rho_0(l)$. This is illustrated in Figure 2.4, where a sample node density distribution is shown along with its padding and two instances of $G_{\rho_0}(\cdot, l')$. The constant λ in Eq. (2.42) can be related to the maximum grid stretching r according to

$$\lambda = \frac{\ln r}{d}, \tag{2.43}$$

where d denotes the equidistribution constant of $P_{\rho_0}(l)$,

$$d = \frac{1}{n_p}\int_0^L P_{\rho_0}(l)\, dl. \tag{2.44}$$

In practice, we evaluate d using a fixed-point iteration based on Eqs. (2.41) and (2.44) with initial estimate $d \approx \Delta\tau$. If the adjusted node density $\rho_\infty(l)$ coincides with its padding $P_{\rho_\infty}(l)$,

$$\rho_\infty(l) = P_{\rho_\infty}(l) \quad \forall l \in [0, L], \tag{2.45}$$

then, by Lemma 5 and Theorem 2 of Kautsky and Nichols [88], the grid $l_i = \bar{l}_\infty(\tau_i)$, $i = 1, \ldots, n_p + 1$, obtained from equidistributing $\rho_\infty(l)$,

$$\bar{\tau}_\infty(l) = \int_0^l \frac{\rho_\infty(u)}{\rho_\tau} \, du, \tag{2.46}$$

is locally bounded with ratio r (Eq. (2.36)).

For compatibility with ρ_τ, the padding of the minimum node density $P_{\rho_{\min}}(l)$ can distribute at most as much node density as ρ_τ is able to supply. In view of Eq. (2.39), we hence have the compatibility condition

$$\int_0^L P_{\rho_{\min}}(u) \, du \leq \rho_\tau L. \tag{2.47}$$

In order to evolve $\bar{\tau}_0(l)$ into the adjusted coordinate transformation $\bar{\tau}_\infty(l)$, we postulate the following initial value problem (IVP) for $l \in [0, L]$, $s \geq 0$

$$\frac{\partial \rho(l, s)}{\partial s} = \dot{R}(l, s) - \frac{\rho(l, s)}{\rho_\tau} \frac{1}{L} \int_0^L \dot{R}(u, s) \, du \tag{2.48}$$

subject to the initial condition

$$\rho(l, s = 0) = \rho_0(l) = \frac{d\bar{\tau}_0(l)}{dl} \rho_\tau \tag{2.49}$$

and consider $\rho_\infty(l) = \rho_\tau d\bar{\tau}_\infty(l)/dl \equiv \rho(l, s \to \infty)$ as the steady-state node density distribution. Here, the kinetic rate of change $\dot{R}(l, s)$,

$$\dot{R}(l, s) = \frac{1}{T} \max(\max(\rho_{\min}(l), P_\rho(l, s)) - \rho(l, s), 0), \tag{2.50}$$

quantifies the extent to which the current node density $\rho(l, s)$ violates the minimum node density requirement $\rho(l, s) \geq \rho_{\min}(l)$ or deviates from the padding $P_\rho(l, s) \equiv P_{\rho(\cdot, s)}(l)$ in relation to a time constant T.

From a physical perspective, the idea underlying Eq. (2.48) is to increase the node density locally commensurate with the node density deficit $\dot{R}(l, s)T$ and to reduce the node density elsewhere by an amount that is proportional to the node density already present. By integrating Eq. (2.48) over $[0, L]$ and taking into account Eq. (2.39), we obtain

$$\frac{\partial}{\partial s} \int_0^L \rho(u, s) \, du = \int_0^L \dot{R}(u, s) \, du \left(1 - \frac{1}{\rho_\tau L} \int_0^L \rho(u, s) \, du \right). \tag{2.51}$$

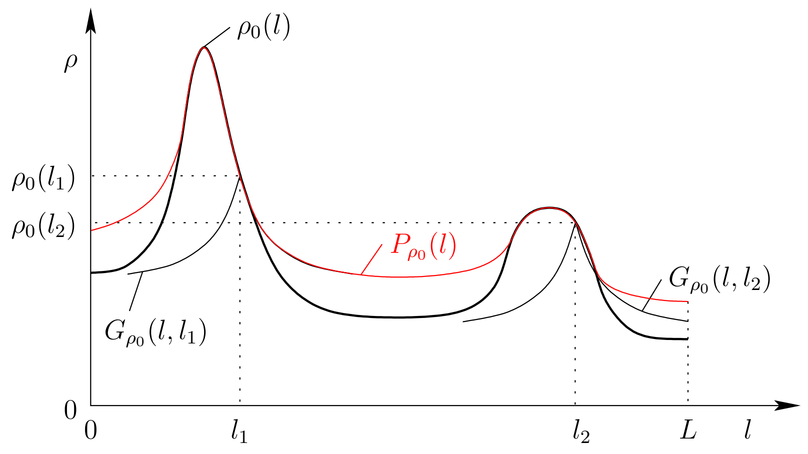


Figure 2.4 A schematic illustration of the node density $\rho_0(l)$ and its padding $P_{\rho_0}(l)$ [88]. Here, the padding is constructed as the envelope of all functions $G_{\rho_0}(l,l')$, $l' \in [0,L]$.

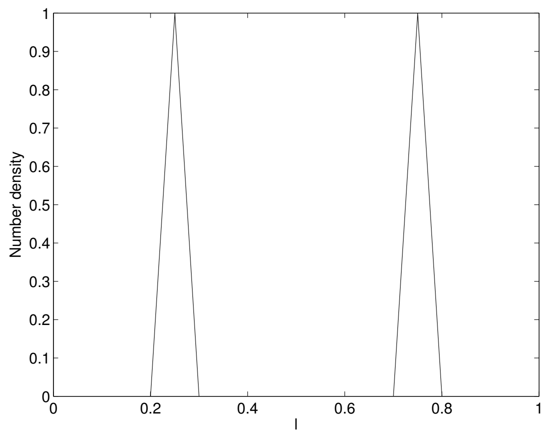
This shows that if $\int_0^L \rho(u,0)\,du = \rho_\tau L$ holds initially, then the source and sink terms on the right hand side of Eq. (2.48) balance such that the total amount of node density $\rho_\tau L$ is conserved for all times $s \geq 0$. In Appendix A.1, we proof that $\rho_\infty(l)$ satisfies Eqs. (2.40) and (2.45) if and only if $\rho_\infty(l) = \rho(l, s_\infty)$ is a steady-state solution of Eq. (2.48) for some $s_\infty \geq 0$.

Appendix A.2, moreover, details a numerical solution scheme for computing the steady-state solution $\rho_\infty(l)$ of Eqs. (2.48) and (2.49). From $\rho_\infty(l)$, we obtain the adjusted coordinate transformation $\bar{\tau}_\infty(l)$ by integration (Eq. (2.46)). Inverting $\bar{\tau}_\infty(l)$ then yields the associated inverse coordinate transformation $\bar{l}_\infty(\tau)$.

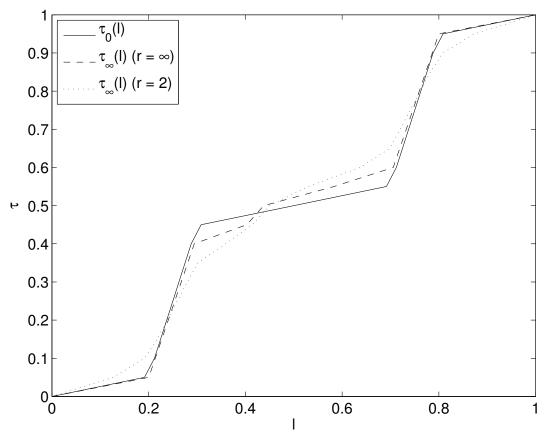
2.5.2.1 An example for the node density adjustment

In order to illustrate the node density adjustment scheme, we let $[0, L]$ be the unit interval $[0, 1]$ and consider the reference particle property distribution $N_0(l)$ depicted in Figure 2.5(a). Here, $N_0(l)$ varies piecewise linearly on a uniform grid in l-space with 21 nodes and possesses two sharp peaks at $l = 0.25$ and $l = 0.75$. By Eqs. (2.30), (2.31) and (2.33) with $b = 1$, $N_0(l)$ yields the equidistributing coordinate transformation $\bar{\tau}_0(l)$ depicted as a solid line in Figure 2.5(b) and, by Eq. (2.37), the l-space node density distribution shown in Figure 2.5(c). The inverse coordinate transformation $\bar{l}_0(\tau)$, moreover, maps a uniform τ-grid (see Eq. (2.18) with $n_p = 20$) onto the l-grid illustrated in the top row of Figure 2.5(d).

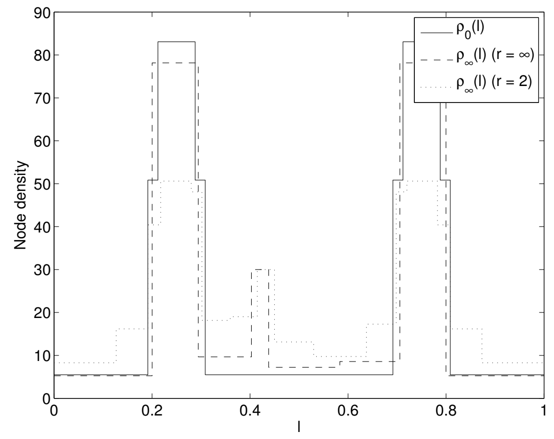
By way of example, we set the minimum admissible node density $\rho_{\min}(l)$ to 30 nodes per unit of l on $[0.4, 0.45]$ and to zero elsewhere and apply the node density adjustment scheme both with ($r = 2$) and without ($r = \infty$) the spatial smoothening. The resulting adjusted coordinate transformations $\bar{\tau}_\infty(l)$ and their corresponding node density distributions $\rho_\infty(l)$ are depicted as dotted and dashed lines in Figures 2.5(b) and 2.5(c), respectively, while



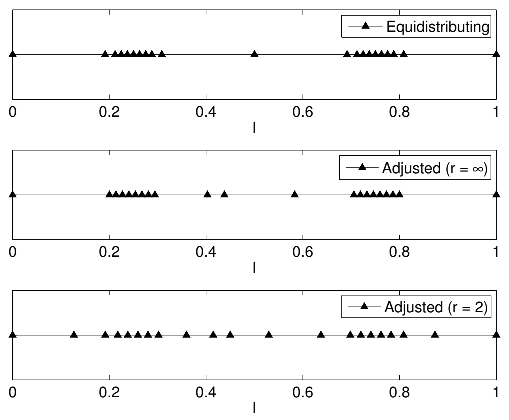
(a) Number density distribution



(b) Coordinate transformation



(c) Node density distribution



(d) l-images of a uniform τ-grid

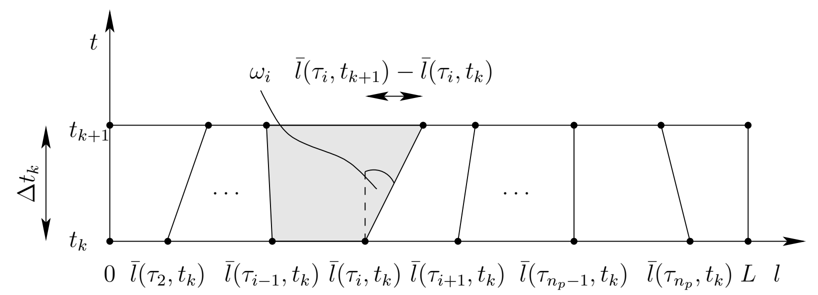
Figure 2.5 An example for the node density adjustment scheme. The solid lines in Figures (b) and (c) indicate the equidistributing coordinate transformation and node density for the reference particle property distribution shown in Figure (a). The dotted and dashed lines, on the other hand, refer to the adjusted coordinate transformation and node density both with ($r = 2$) and without ($r = \infty$) the spatial smoothening. Figure (d) illustrates the l-grids obtained by applying the equidistributing and adjusted coordinate transformations, respectively, to a uniform grid in τ-space.

the middle and bottom rows of Figure 2.5(d) illustrate the adjusted l-space images of the uniform τ-grid.

2.5.3 Prescribing the coordinate transformation explicitly

In this section, we develop an explicit prescription for the time evolution of the coordinate transformation $\bar{l}(\tau, \mathbf{x}, t)$ over the upcoming time interval $[t_k, t_{k+1}]$ based on the current transformed number density $f(\tau, \mathbf{x}, t_k)$ and the current coordinate transformation $\bar{l}(\tau, \mathbf{x}, t_k)$. Following Davis and Flaherty [37], we specifically let $\bar{l}(\tau, \mathbf{x}_j, t)$ vary linearly in time for $t \in [t_k, t_{k+1}]$ at each spatial grid point $\mathbf{x}_j$

$$\bar{l}(\tau, \mathbf{x}_j, t) = \bar{l}(\tau, \mathbf{x}_j, t_k) + \left(\bar{l}(\tau, \mathbf{x}_j, t_{k+1}) - \bar{l}(\tau, \mathbf{x}_j, t_k)\right) \frac{t - t_k}{\Delta t_k}, \tag{2.52}$$



t
ω_i
$\bar{l}(\tau_i, t_{k+1}) - \bar{l}(\tau_i, t_k)$
t_{k+1}
Δt_k
t_k
$\dots$
$\dots$
0
$\bar{l}(\tau_2, t_k)$
$\bar{l}(\tau_{i-1}, t_k)$
$\bar{l}(\tau_i, t_k)$
$\bar{l}(\tau_{i+1}, t_k)$
$\bar{l}(\tau_{n_p-1}, t_k)$
$\bar{l}(\tau_{n_p}, t_k)$
L
l

Figure 2.6 A schematic illustration of the l-t-mesh formed by the nodes $\bar{l}(\tau_i, \mathbf{x}_j, t_k)$ and $\bar{l}(\tau_i, \mathbf{x}_j, t_{k+1})$, $i = 1, \dots, n_p + 1$ [37]. For conciseness, we omit the space dependency of $\bar{l}$ in the figure.

where the new coordinate transformation $\bar{l}(\tau, \mathbf{x}_j, t_{k+1})$ is given by

$$\bar{l}(\tau, \mathbf{x}_j, t_{k+1}) = \left(1 - \exp\left(-\frac{\Delta t_k}{\kappa}\right)\right) \bar{l}_\infty(\tau) + \exp\left(-\frac{\Delta t_k}{\kappa}\right) \bar{l}(\tau, \mathbf{x}_j, t_k) \tag{2.53}$$

and the coordinate transformation $\bar{l}_\infty(\tau)$ is defined as in Sections 2.5.1 and 2.5.2 based on $N_0(l) = \langle N(l, \mathbf{x}, t_k) \rangle_{\Omega_j} = \langle f(\bar{\tau}(l, \mathbf{x}_j, t_k), \mathbf{x}, t_k) \rangle_{\Omega_j}$ for $k \geq 1$ and based on $N_0(l) = \langle N_\Omega(l, \mathbf{x}) \rangle_{\Omega_j}$ for $k = 0$. In Eqs. (2.52) and (2.53), $\Delta t_k = t_{k+1} - t_k$ represents the upcoming fractional time step. The initial coordinate transformation is set to

$$\bar{l}(\tau, \mathbf{x}_j, t_0 = 0) = \bar{l}_\infty(\tau) \tag{2.54}$$

such that $\bar{l}(\tau, \mathbf{x}_j, t_0) = \bar{l}(\tau, \mathbf{x}_j, t_1) = \bar{l}_\infty(\tau)$. For a piecewise linear coordinate transformation $\bar{l}(\cdot, \mathbf{x}, t)$ (see Section 2.4.2), Figure 2.6 schematically depicts the grid in the l-t-plane that is formed by the nodes $\bar{l}(\tau_i, \mathbf{x}_j, t_k)$ and $\bar{l}(\tau_i, \mathbf{x}_j, t_{k+1})$, $i = 1, \dots, n_p + 1$, at a given cell midpoint $\mathbf{x}_j$. As a brief summary, Figure 2.7, moreover, lists the algorithmic steps for constructing the coordinate transformation $\bar{l}(\tau, \mathbf{x}_j, t)$ at $\mathbf{x}_j$ over the upcoming time interval $[t_k, t_{k+1}]$ from the current coordinate transformation $\bar{l}(\tau, \mathbf{x}_j, t)$ and the local particle property distribution $N_0(l)$.

The exponential prefactors in Eq. (2.53) implement a first order lag element in time with a time constant of $\kappa > 0$. This temporal smoothening is quite common both in explicit [49] and in dynamic [24, 43] adaptive grid approaches and helps to prevent temporal oscillations in the evolution of the coordinate transformation if the monitor function is very sensitive to small perturbations in the defining particle property distribution. As an alternative, Davis and Flaherty [37] proposed to limit the angles ω_i formed by the lines $\bar{l}(\tau_i, \mathbf{x}_j, t)$, $i = 1, \dots, n_p + 1$, and the positive t-axis in order to prevent the l-t-quadrilaterals in Figure 2.6 from becoming severely distorted, but we found this criterion to be inadequate if the l-nodes are distributed over several orders of magnitude.

Further to Eqs. (2.52) through (2.54), $\bar{l}(\tau, \mathbf{x}, t)$ may be expressed in terms of the cell midpoint coordinate transformations $\bar{l}(\tau, \mathbf{x}_j, t)$ in a similar way to the formulation of $f(\tau, \mathbf{x}, t)$ in terms of $\langle f(\tau, \mathbf{x}, t) \rangle_{\Omega_j}$. Within a finite volume cell Ω_j, the grid $\{\bar{l}(\tau_i, \mathbf{x}_j, t)\}_{i=1}^{n_p+1}$

```
Input: $N_0(l)$, $\bar{l}(\tau, \mathbf{x}_j, t_k)$, $\Delta t_k$, $\kappa$
    1. Compute $m_{N_0}(l)$ from Eqs. (2.33) and (2.34)
    2. Compute the inverse equidistributing coordinate transformation $\bar{\tau}_0(l)$ according to Eqs. (2.30) and (2.31)
    3. Node density adjustment scheme:
        3.1 Compute the initial node density distribution $\rho_0(l)$ from Eqs. (2.37) and (2.38)
        3.2 Compute a steady-state solution $\rho_\infty(l)$ of the IVP in Eqs. (2.48) and (2.49) (also see Appendix A.2)
        3.3 From $\rho_\infty(l)$ obtain $\bar{\tau}_\infty(l)$ using Eq. (2.46)
        3.4 Invert $\bar{\tau}_\infty(l)$ to obtain $\bar{l}_\infty(\tau)$
    4. Obtain $\bar{l}(\tau, \mathbf{x}_j, t_{k+1})$ from Eq. (2.53)
    5. $\bar{l}(\tau, \mathbf{x}_j, t)$ varies piecewise linearly in $t$ on $[t_k, t_{k+1}]$ (Eq. (2.52) and Figure 2.6)
Output: $\bar{l}(\tau, \mathbf{x}_j, t)$ for $t \in [t_k, t_{k+1}]$
```

Figure 2.7 Summary of the algorithmic steps for explicitly constructing the coordinate transformation $\bar{l}(\tau, \mathbf{x}_j, t)$ over the upcoming time interval $[t_k, t_{k+1}]$ from the current particle property distribution $N_0(l)$ in cell Ω_j and the current coordinate transformation $\bar{l}(\tau, \mathbf{x}_j, t_k)$.

is thus matched with the cell-average particle property distribution $\{F_{i,j}(t)\}_{i=1}^{n_p}$. For consistency, we found it to be important that the spatial derivatives of $\bar{l}(\tau, \mathbf{x}, t)$ in the surface integral term of Eq. (2.25) are continuous across the cell faces. In the case of cuboidal finite volume cells, this can be ensured, for example, by letting $\bar{l}(\tau, \mathbf{x}, t)$ vary piecewise quadratically in each coordinate direction.

Finally, we note that, by slightly adjusting the developments presented above, the **x**-discrete formulation of $\bar{l}(\tau, \mathbf{x}, t)$ may also be obtained for spatial discretization schemes other than the finite volume discretization considered here.

2.6 Numerical experiments

In this section, we consider three common fixed grid discretization schemes for discretizing the transformed PBE in Eqs. (2.15) and (2.16) on a uniform grid in τ-space:

(1) The standard Galerkin finite element method (GFEM) using linear finite elements [138, 178].

(2) A fully upwinded orthogonal collocation finite element method (OCFEM) based on linear finite elements [175].

(3) The $\kappa = 1/3$ high resolution finite volume method (FVM) proposed by Koren [92] and implemented in the context of the PBE by Qamar et al. [166]. Contrary to these references, we treat source terms in a standard manner for computational efficiency.

In the numerical experiments, the time constant κ in Eq. (2.53) is set equal to the fractional time step $\Delta t = \Delta t_k$, $k \geq 0$, and the absolute and relative convergence tolerances for computing steady-state solutions of Eqs. (2.48) and (2.49) (see A.2) are chosen as 10^{-8} and 10^{-4}, respectively. Additionally, the semi-discrete systems of equations are integrated in time using the 5th order accurate explicit Runge-Kutta method Dopri5 [68] with absolute and relative convergence tolerances of 10^{-10} and 10^{-4}, respectively.

2.6.1 Advection of a unit step

The first example which we consider consists of a step-shaped profile moving at a constant velocity of $g = 0.05$ /s in the positive l-direction. Here, the step is represented by a narrow region over which the number density increases linearly from 0 to 1. Initially, the left edge of the steep gradient interval is located at $l = 0.15$ and the profile of $N(l, t = 0)$ is given by

$$h(l) \equiv N(l,0) = \begin{cases} 0 & \text{for } l < 0.15 \\ \frac{l-0.15}{0.05} & \text{for } l \in [0.15, 0.2] \\ 1 & \text{else} \end{cases} \tag{2.55}$$

for $l \in [0, 4]$. The analytical solution of the pure advection equation

$$\frac{\partial N(l,t)}{\partial t} + g\frac{\partial N(l,t)}{\partial l} = 0 \tag{2.56}$$

subject to the left boundary condition $N(0, t) = 0$ and the initial condition in Eq. (2.55) can be obtained by the method of characteristics,

$$N(l,t) = h(l - gt). \tag{2.57}$$

In the following, we compare the analytical solution in Eq. (2.57) with the numerical solutions obtained from different direct discretization schemes (OCFEM and FVM) both stand-alone and as fixed grid solvers within the explicit adaptive grid method (EAGM) developed in Sections 2.4 and 2.5. Here, the non-adaptive discretization schemes are based on a uniform grid in physical particle property space which has been slightly modified such that the initial particle property distribution $h(l)$ can be represented exactly on this grid.

In the EAGM, the time horizon for advancing the coordinate transformation is set to $\Delta t = 10^{-2}$ s and, with the exception of Figure 2.11, the value for the regularization constant b amounts to $b = 10^4$. The maximum grid stretching is set to $r = 2$ and, in the absence of source terms, the minimum node density vanishes identically ($\rho_{\min} = 0$). Additionally, we construct the initial coordinate transformation based on the initial profile in Eq. (2.55) represented on a modified uniform grid and, subsequently, modify the coordinate transformation such that two nodes are again located at 0.15 and 0.2, respectively.

Figure 2.8 depicts the analytical reference solution as well as the numerical solutions obtained from the FVM and OCFEM both with and without the EAGM using 20 finite volume cells/finite elements at different points in time. While both the FVM and the

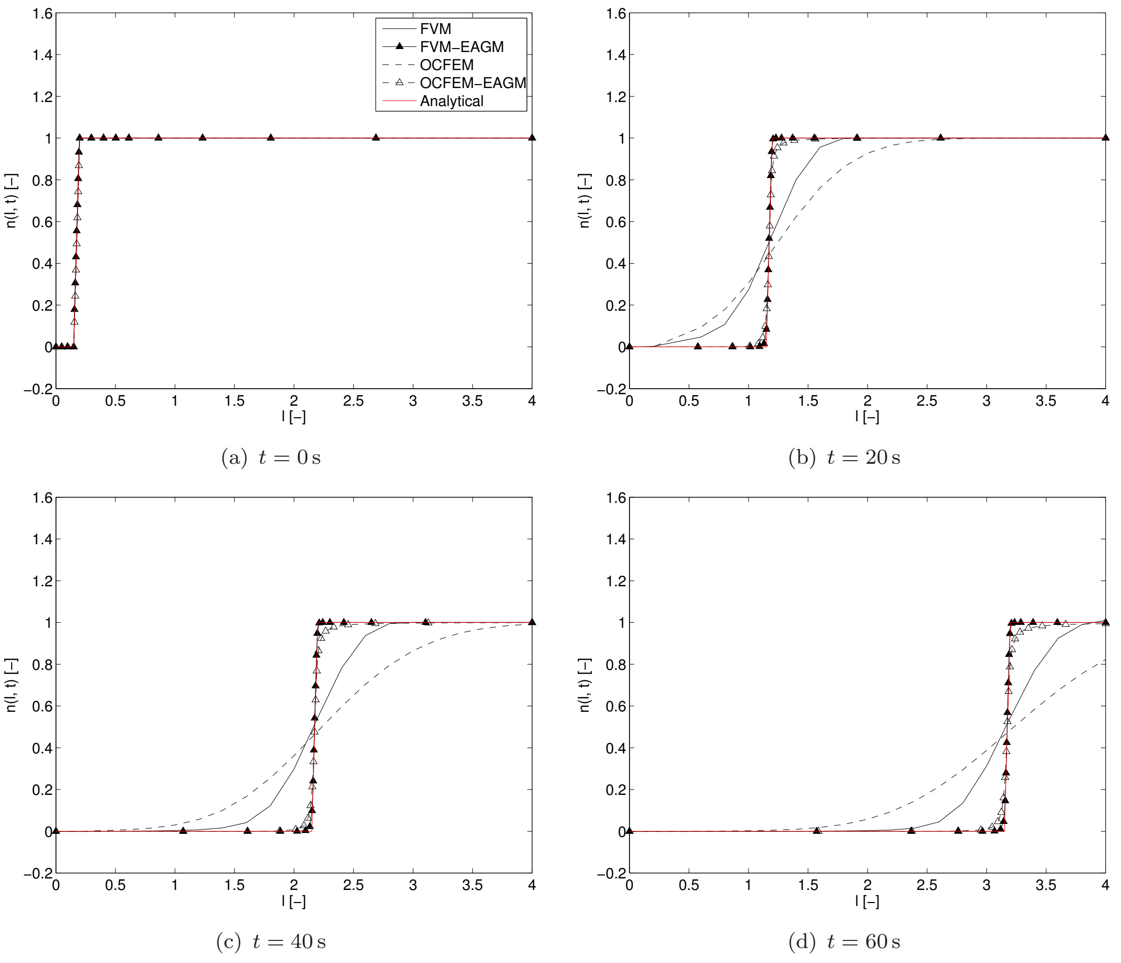


Figure 2.8 Comparing the analytical solution for the time evolution of a step-shaped profile with the numerical approximations computed from the FVM and OCFEM both with and without the EAGM. Here, the numerical solutions have been obtained using 20 finite volume cells/finite elements and the b-value in the EAGM monitor function is set to $b = 10^4$. The markers, moreover, indicate the cell face/node locations in physical particle property space.

OCFEM appear to be very diffusive, the combined FVM-EAGM and OCFEM-EAGM schemes yield results which agree very well with the analytical solution. Some signs of numerical diffusion remain (more so in the OCFEM-EAGM solutions than in the FVM-EAGM solutions), but the increase in accuracy is significant.

For the EAGM solutions, the triangular markers in Figure 2.8 indicate the positions in physical particle property space of the finite volume cell faces and the finite element nodes, respectively. Here, most of the finite volume cells/finite elements are located in the vicinity of the moving step to which the monitor function assigns a high measure of information content. Away from the steep gradient interval, by contrast, the slope of the profile flattens out and the node density decays. As the numerical solution evolves and the step moves to the right, the cells/elements readjust, thus tracking the step profile. In this regard, we emphasize that the solution adaptivity is independent of the local characteristics.

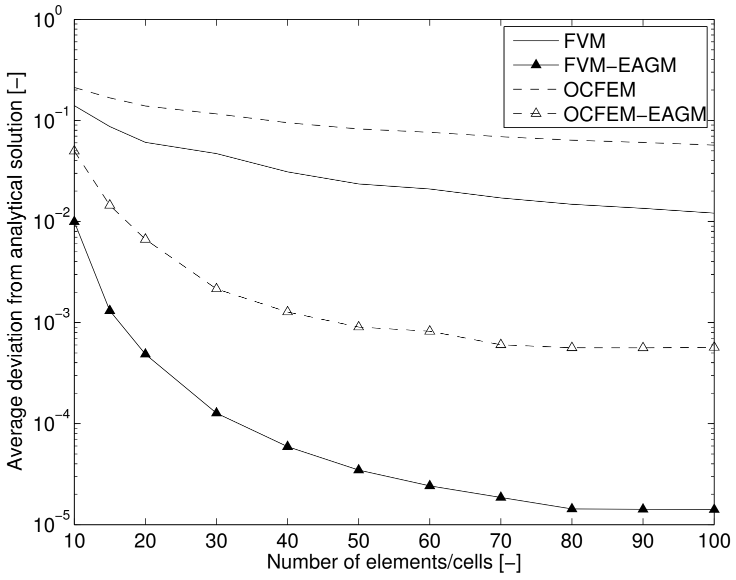


Figure 2.9 Convergence diagram in terms of the average deviation of the numerical solution from the analytical reference solution for the step advection example at time $t = 45\,\mathrm{s}$. Here, the FVM and OCFEM are applied both with and without the EAGM ($b = 10^4$).

For the four numerical solution schemes investigated in Figure 2.8, Figure 2.9 shows a convergence diagram in terms of the average deviation $e(t)$ of the numerical approximation $N(l,t)$ from the analytical reference solution $h(l-gt)$ at time $t = 45\,\mathrm{s}$,

$$e(t) \equiv \frac{1}{L}\int_0^L |N(l,t) - h(l-gt)|\; dv. \tag{2.58}$$

The results in Figure 2.9 indicate that the FVM solutions are more accurate than the OCFEM solutions and that incorporating the EAGM yields an increase in accuracy by two to three orders of magnitude. Conversely, in order to achieve a given accuracy, the EAGM-based solution schemes require over an order of magnitude fewer grid points than the stand-alone FVM and OCFEM solvers.

In order to assess the robustness of the EAGM, we quantify the influence of the time horizon Δt on the accuracy of the EAGM-based numerical solutions in Figure 2.10. Here, the average deviations of the EAGM-FVM and EAGM-OCFEM solutions from the analytical reference solution are plotted over the EAGM time horizon for 20 finite volume cells/finite elements. Up to $\Delta t \lesssim 5 \times 10^{-2}\,\mathrm{s}$, the average deviation remains constant at a minimum value, while it grows notably as Δt is increased beyond this threshold value. In practical configurations, particle formation processes are often found to evolve on time scales which are larger than the time scales governing the spatial transport of number density. In this case, the threshold value for the EAGM time horizon exceeds the time step for the fractional steps scheme which is typically chosen based on a CFL condition for convection or diffusion in physical space.

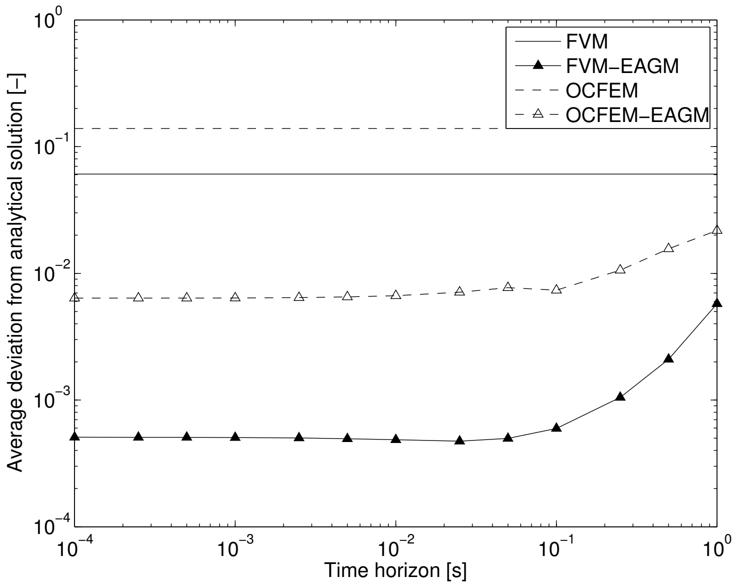


Figure 2.10 Analyzing the influence of the EAGM time horizon on the accuracy of the numerical solutions for the step advection example at time $t = 45\,\mathrm{s}$. Here, the EAGM ($b = 10^4$) is combined both with the FVM and OCFEM using 20 finite elements/finite volume cells. For reference, the horizontal lines indicate the average deviation of the numerical approximations without the EAGM from the analytical solution.

Finally, we analyze the influence of the regularization constant b in the monitor function (Eq. (2.33)) on the accuracy of the numerical approximations obtained from the combined FVM-EAGM and OCFEM-EAGM schemes for 20 finite volume cells/finite elements at time $t = 45\,\mathrm{s}$. In view of Section 2.5.1, b controls the spacing of nodes on a linear scale in regions of l-space over which the solution vanishes identically. Figure 2.11 indicates that, for very small b-values, the average deviation approaches a maximum value. Here, b is so small that solution adaptivity is lost since the monitor function has become insensitive to the first derivative of the solution. As b becomes very large, by contrast, the average deviation approaches a minimum limiting value which is one (OCFEM-EAGM) to two (FVM-EAGM) orders of magnitude smaller than the maximum average deviation at small b-values. For practical purposes, we thus recommend to choose b moderately large at the outset and to decrease b if the monitor function seems to loose regularity.

2.6.2 $BaSO_4$ precipitation in a plug flow reactor

In the present section, we consider the precipitation of barium sulphate ($BaSO_4$) particles from aqueous solutions of sodium sulphate (Na_2SO_4) and barium chloride ($BaCl_2$) in a plug flow reactor. In particular, our objective is to validate the accuracy and assess the convergence behavior and computational cost of the explicit adaptive grid method (EAGM) for realistic particle formation kinetics and inflow conditions. The kinetic relations and parameters correspond to those suggested by Bałdyga and Orciuch [18] and the inflow conditions are similar to the ones used in their high concentration experiments of

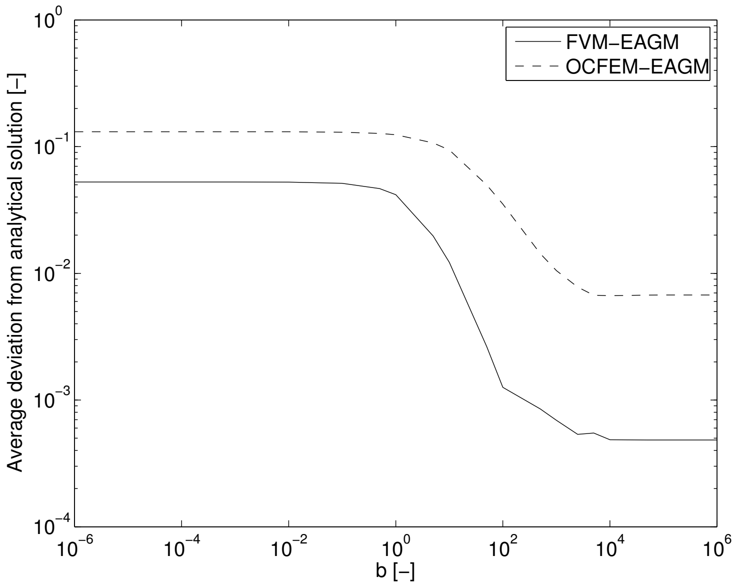


Figure 2.11 Investigating the influence of the regularization constant b in the monitor function (Eq. (2.33)) on the average deviation of the EAGM solutions from the analytical reference solution for the step advection example at time $t = 45\,\mathrm{s}$. Here, the EAGM is applied in combination with the FVM and OCFEM using 20 finite volume cells/finite elements.

precipitation in a coaxial pipe mixer at a Reynolds number of $Re = 3 \times 10^4$ and a unity jet/co-flow velocity ratio, $Ru = 1$. Since the plug flow model does not resolve the spatial and temporal flow structures in the mixer, we do not attempt to compare the computed particle size distributions with the experimental measurements of Bałdyga and Orciuch [18]. Rather, our focus lies on the performance of the numerical solution scheme.

The composition of the liquid phase in the reactor can be described by the mass fractions of H_2O and of the ionic species Ba^{2+}, ${SO_4}^{2-}$, Cl^- and Na^+,

$$\mathbf{Y}(\mathbf{x},t) = \begin{pmatrix} y_{\mathrm{H_2O}}(\mathbf{x},t) \\ y_{\mathrm{Ba}}(\mathbf{x},t) \\ y_{\mathrm{SO_4}}(\mathbf{x},t) \\ y_{\mathrm{Cl}}(\mathbf{x},t) \\ y_{\mathrm{Na}}(\mathbf{x},t) \end{pmatrix}. \tag{2.59}$$

At the plug flow inlet, the liquid phase composition is given by

$$\mathbf{Y}_0 = \begin{pmatrix} 9.969 \times 10^{-1} \\ 6.538 \times 10^{-4} \\ 1.436 \times 10^{-3} \\ 3.375 \times 10^{-4} \\ 6.875 \times 10^{-4} \end{pmatrix}. \tag{2.60}$$

This corresponds to a perfect mixture of the jet and co-flow compositions in the high

Phase	Description	Variable	Value	Units
Liquid	Fluid density	ρ	10^3	kg/m^3
	Kinematic viscosity	ν	10^{-6}	m^2/s
	Diffusivity	D	1.43×10^{-6}	m^2/s
Particles	$BaSO_4$ mass density	$\rho_{\mathrm{BaSO_4}}$	4480	kg/m^3
	$BaSO_4$ molecular weight	$MW_{\mathrm{BaSO_4}}$	233.4	kg/kmol
	$BaSO_4$ volume shape factor	k_v	58	$-$
	Average nuclei size	l_{nuc}	2×10^{-9}	m
	Maximum particle size	L	10^{-4}	m
	Diffusivity	D_p	1.43×10^{-6}	m^2/s
	Solubility product	k_S	1.1×10^{-10}	kmol^2/m^6
	Surface integration coefficient	k_g	4.0×10^{-11}	m/s
	Mass transfer coefficient	k_D	10^{-4}	m^4/kmol $-$ s

Table 2.1 Constitutive, kinetic and transport parameters for the precipitation of $BaSO_4$ in a plug flow reactor [18].

concentration experiments of Bałdyga and Orciuch [18] at $Re = 3 \times 10^4$ and $Ru = 1$. If solid $BaSO_4$ particles precipitate from the solution, then the liquid phase is locally depleted of both Ba^{2+} and ${SO_4}^{2-}$. In the transport equations for y_{Ba} and $y_{\mathrm{SO_4}}$, this effect is accounted for by the sink terms

$$\dot{s}_{\mathrm{Ba}}(N(l,\mathbf{x},t)) = \dot{s}_{\mathrm{SO_4}}(N(l,\mathbf{x},t)) = -\frac{\rho_{\mathrm{BaSO_4}}}{\rho MW_{\mathrm{BaSO_4}}} \frac{DV(\mathbf{x},t)}{Dt}, \tag{2.61}$$

where $\rho_{\mathrm{BaSO_4}}$ and $MW_{\mathrm{BaSO_4}}$ denote the mass density and molecular weight, respectively, of solid $BaSO_4$, ρ is the mixture density of the fluid phase and $DV(\mathbf{x},t)/Dt$ represents the material time derivative of the total volume density associated with the particulate phase. If l represents a characteristic particle size such that the volume v of one $BaSO_4$ particle is given by

$$v = k_v l^3, \tag{2.62}$$

then the total volume density of $BaSO_4$ particles can be computed as the third moment of the particle size distribution $N(l,\mathbf{x},t)$,

$$V(\mathbf{x},t) = k_v \int_0^L l^3 N(l,\mathbf{x},t)\, dl, \tag{2.63}$$

where k_v denotes the volume shape factor. For reference, Table 2.1 lists the values of all physical parameters introduced in this section.

2.6.2.1 $BaSO_4$ formation kinetics

Barium sulphate has been considered as a model substance for precipitation from aqueous solution by a number of experimental investigators, see, for instance, Aoun et al. [8] and references therein. One advantage of using $BaSO_4$ is that the suspension can be electrostatically stabilized against aggregation by adding an excess of Ba^{2+} ions [64]. This is particularly beneficial if the investigation focusses on nucleation and growth processes only. In addition, $BaSO_4$ particles are not toxic and their precipitation from an aqueous solution does not involve a liquid phase reaction. Moreover, since the combined solubility of Ba^{2+} and ${SO_4}^{2-}$ ions is very low, it is possible to realize high supersaturations (see below) which are a prerequisite for producing particles in the nanometer size range [183].

Independent of its benefits as a substance for the investigation of precipitation processes, $BaSO_4$ is also of practical importance: $BaSO_4$ particles are being used as additives in coatings or paints to improve brilliancy and as contrast agents in X-ray imaging, for instance. Following Schwarzer [183], moreover, $BaSO_4$ particles are frequently added to polymers in order to increase scratch resistency and included in the manufacturing of specialty inkjet paper to enhance the sharpness of plots.

The nucleation and growth rates of $BaSO_4$ are typically computed from semi-empirical models which include kinetic parameters that were determined from a set of experimental data [8]. For comparison purposes and for compatibility with the work of Di Veroli and Rigopoulos [40], we adopt, in this work, the nucleation and growth kinetics of Bałdyga and Orciuch [18].

Conceptually, the $BaSO_4$ kinetic rate expressions are based on supersaturation $S(\mathbf{Y}(\mathbf{x},t))$ as a measure for the deviation of an ionic aqueous solution from chemical equilibrium,

$$S(\mathbf{Y}) = \gamma_\pm(\mathbf{Y})\sqrt{\frac{[\mathrm{Ba}^{2+}][{\mathrm{SO}_4}^{2-}]}{k_S}}. \tag{2.64}$$

Here, the square brackets indicate the molar concentration of the argument species, k_S denotes the solubility product of $BaSO_4$ and $\gamma_\pm(\mathbf{Y})$ represents the mean activity coefficient of Ba^{2+} and ${SO_4}^{2-}$ in an ionic aqueous solution also containing Cl^- and Na^+. If the ionic strength of the solution is low ($\lesssim 6$) and $BaSO_4$ is considered a strong electrolyte, then $\gamma_\pm(\mathbf{Y})$ may be approximated by Bromley's relation [27]. This approach takes into account the interactions of Ba^{2+} and ${SO_4}^{2-}$ with unlike charged ions but neglects interactions between like charged ions and amongst ion triplets.

Following Nielsen [139], the critical radius r_{nuc} of a spherical $BaSO_4$ nucleus can be computed according to

$$r_{\mathrm{nuc}} = \frac{2\sigma_{\mathrm{BaSO}_4} v_{\mathrm{BaSO}_4}}{k_B T \ln S}, \tag{2.65}$$

where σ_{BaSO_4} denotes the $BaSO_4$ surface tension, k_B is Boltzmann's constant, T represents the ambient temperature and v_{BaSO_4} denotes the molecular volume

$$v_{\mathrm{BaSO}_4} = \frac{MW_{\mathrm{BaSO}_4}}{N_A \rho_{\mathrm{BaSO}_4}}. \tag{2.66}$$

Here, N_A is Avogadro's number. If $S = 326.96$ (the supersaturation associated with the initial composition in Eq. (2.60)) and $T = 298\,\mathrm{K}$ (the ambient temperature at which

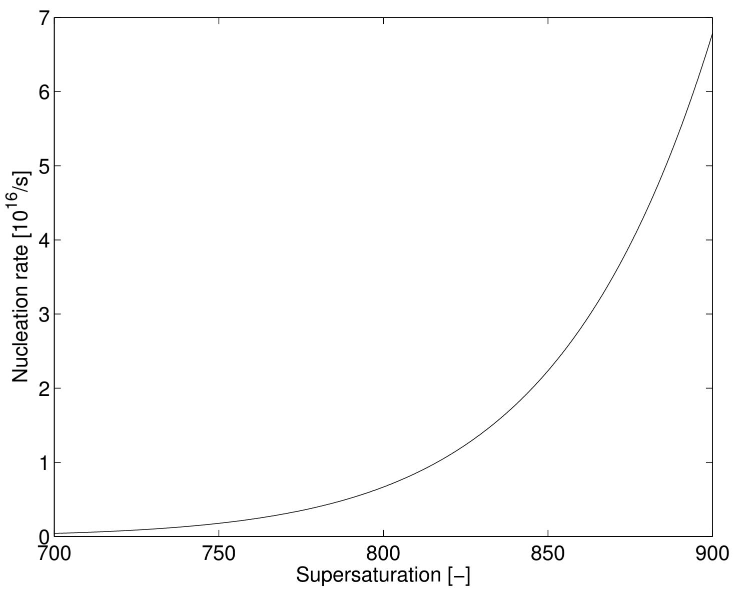


Figure 2.12 The nucleation rate of $BaSO_4$ as a function of supersaturation (Eq. (2.67)).

Bałdyga and Orciuch [18] conducted their experiments), then the critical radius is found to be $r_{\text{nuc}} = l_{\text{nuc}}/2 = 9.806 \times 10^{-10}\,\text{m} \approx 1\,\text{nm}$.

Following Bałdyga and Orciuch [18], the nucleation rate $R_N(\mathbf{Y}(\mathbf{x},t))$ is computed according to

$$R_N(\mathbf{Y}) = 1.06 \times 10^{12} \exp\left(-\frac{44.6}{(\ln S(\mathbf{Y}))^2}\right) + 1.50 \times 10^{45} \exp\left(-\frac{3020.0}{(\ln S(\mathbf{Y}))^2}\right), \tag{2.67}$$

while the size-independent growth rate $G(\mathbf{Y}(\mathbf{x},t))$ can be obtained from the two-step model

$$G(\mathbf{Y}) = k_D\left([\text{Ba}^{2+}] - [\text{Ba}^{2+}]_S\right) = k_D\left([\text{SO}_4{}^{2-}] - [\text{SO}_4{}^{2-}]_S\right) = k_g\left(S(\mathbf{Y}_S) - 1\right)^2, \tag{2.68}$$

where k_g denotes the surface integration coefficient, k_D is the diffusional mass transfer coefficient and $\mathbf{Y}_S$ represents the liquid composition at the crystal surface. Bałdyga and Orciuch [18] point out that the coefficients in Eq. (2.67) have been obtained neglecting the effect of ion pair complex formation, but that inclusion of this effect is expected to only slightly change the nucleation rate, if at all. Figure 2.12 depicts the change in nucleation rate R_N over the local supersaturation as given by Eq. (2.67). For the precipitation of $BaSO_4$ in a steady-state plugflow reactor with the inflow composition $\mathbf{Y}_0$ given in Eq. (2.60), Figure 2.13, moreover, shows the evolution of the growth rate G with supersaturation $S(\mathbf{Y})$ or, equivalently, with the volume density of the particulate phase.

Lastly, the PBE source term $\dot{s}(l, \mathbf{Y})$ is modeled as a hat function on $[0, 2l_{\text{nuc}}]$ with maximum value $R_N(\mathbf{Y})/l_{\text{nuc}}$ at the mean nuclei size l_{nuc}. In this regard, the minimum

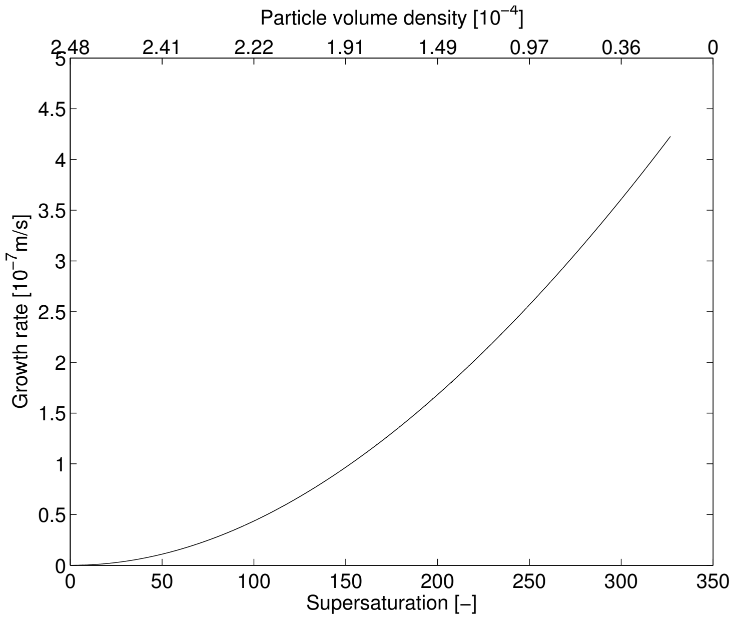


Figure 2.13 The $BaSO_4$ growth rate as a function of supersaturation and volume density of the particulate phase for the precipitation of $BaSO_4$ in a steady-state plug flow reactor with initial composition $\mathbf{Y}_0$ (Eq. (2.60)).

node density $\rho_{\min}(l)$ is chosen as

$$\rho_{\min}(l) = \begin{cases} \rho_{\text{nuc}} & \text{for } l \in [0, 2l_{\text{nuc}}] \\ 0 & \text{for } l > 2l_{\text{nuc}} \end{cases}, \tag{2.69}$$

where $\rho_{\text{nuc}} > 0$ denotes the constant minimum node density in the nucleation interval (for instance, 4 nodes/$2l_{\text{nuc}}$). By introducing Eq. (2.69) into Eq. (2.47) and taking into account Eqs. (2.41) through (2.43), we obtain the following implicit relation between the nucleation node density ρ_{nuc} and the maximum grid stretching $r = \exp(\lambda d)$

$$\frac{1}{\lambda} \ln\left(1 + \lambda(L - 2l_{\text{nuc}})\rho_{\text{nuc}}\right) + 2l_{\text{nuc}}\rho_{\text{nuc}} \leq \rho_{\mathcal{T}} L. \tag{2.70}$$

For a given value of r, the equality condition in Eq. (2.70) returns the maximum admissible value for ρ_{nuc}.

2.6.2.2 $BaSO_4$ precipitation in a steady-state plug flow reactor

In a steady-state plug flow reactor both the composition of the fluid phase and the particle size distribution are parameterized by a single spatial coordinate, the axial distance x, and evolve according to the governing equations

$$u\frac{d\mathbf{Y}(x)}{dx} = \dot{\mathbf{s}}(N(\cdot, x)) \tag{2.71}$$

and

$$u\frac{\partial N(l,x)}{\partial x}+\frac{\partial\left(G(\mathbf{Y}(x))N(l,x)\right)}{\partial l}=\dot{s}(l,\mathbf{Y}(x)) \tag{2.72}$$

subject to the boundary conditions

$$N(0,x)=0, \tag{2.73}$$

$$N(l,0)=0, \tag{2.74}$$

$$\mathbf{Y}(0)=\mathbf{Y}_0. \tag{2.75}$$

Here, $u=0.9375\,\mathrm{m/s}$ denotes the bulk velocity and $\dot{\mathbf{s}}(N(l,x))=(0,\dot{s}_{\mathrm{Ba}}(N(l,x)),\dot{s}_{\mathrm{SO}_4}(N(l,x)), 0,0)^T$ represents the composition source term (Eq. (2.61)).

By applying the coordinate transformation $x=\bar{x}(t)=ut$, $t=\bar{t}(x)=x/u$, Eqs. (2.71) and (2.72) can be transformed into the governing equations for a batch reactor. Within the scope of a fractional steps scheme, both equations are then solved sequentially in a time step.

In the following, we consider solutions for the particle size distribution $N(l,x)$ downstream of the plug flow inlet obtained from different direct discretization approaches (GFEM, OCFEM and FVM) both with and without the EAGM. In the non-adaptive case, the discretization is based on an l-grid that is uniformly spaced over the nucleation interval, encompassing $\max(0.1n_p,4)$ nodes in $[0,2l_{\mathrm{nuc}}]$, and exponentially spaced over $[2l_{\mathrm{nuc}},L]$. In the EAGM, the regularization constant b is chosen as unity and, for the node density adjustment scheme, we set $\rho_{\mathrm{nuc}}=\max(0.1n_p,4)/(2l_{\mathrm{nuc}})$ and $r=2.5$. The fractional time step, moreover, coincides with the EAGM time horizon and is set to $\Delta\bar{t}=10^{-4}\,\mathrm{s}$, corresponding to an x-step of $\Delta x=9.4\times10^{-5}\,\mathrm{m}$.

Figure 2.14 depicts the particle size distributions at four different cross-sections of the plug flow reactor computed using the FVM with 30 and 1000 finite volume cells, respectively, and the combined FVM-EAGM approach with 30 finite volume cells. In the absence of an analytical solution, we consider the results obtained from the FVM with 1000 cells as reference solution. (Figure 2.19(a) indicates that at this level of mesh refinement convergence in the 0th moment, the total particle number density, has been attained.) Close to the inlet, at $x=9.4\,\mathrm{mm}$, the particle size distribution consists of a sharp peak near the mean nuclei size, indicating that here nucleation is the dominant process for particle formation. Further downstream, at $x=0.47\,\mathrm{m}$, the right leg of the peak has moved towards bigger particle sizes forming a steep moving front which propagates at the local growth rate. In both cross-sections, the particle size distributions from the combined FVM-EAGM approach using 30 cells almost perfectly reproduce the reference results from the FVM using 1000 cells. In particular, the FVM-EAGM scheme is able to accurately resolve the moving near-discontinuity and to correctly predict its location. The FVM-only results for 30 finite volume cells, on the other hand, display some numerical diffusion which smears out the moving front over a few cells.

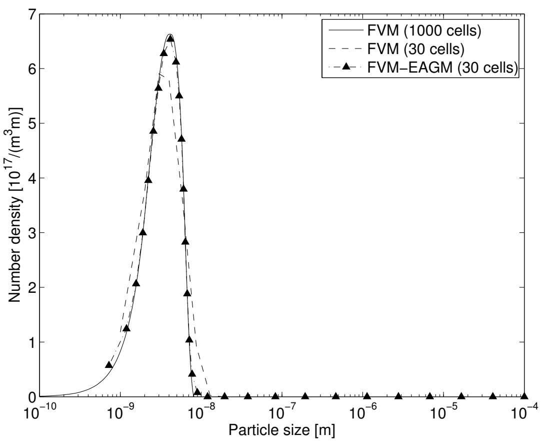
By $x = 7.0\,\mathrm{m}$, the moving front has advanced into the micrometer range, while nucleation has continued to supply new particles. Immediately to the right of the nucleation size range this leads to the formation of a heavy number density hump which smoothly decays towards the moving front. This large-scale structure is significantly over-predicted by the FVM with 30 cells, while it is reproduced well by the combined FVM-EAGM approach using 30 cells. Further downstream, at $x = 42.2\,\mathrm{m}$, the particle size distribution has reached a steady-state in x. Here, the agreement between the reference result and the FVM-EAGM solution for 30 cells remains very good, while the FVM-only scheme with 30 cells fails to resolve the steep front on the far right.

For the EAGM solutions, the filled markers in Figure 2.14 indicate the locations in physical particle size space of the finite volume cell faces. As in the step advection example of Section 2.6.1, the markers move towards locations at which the slope of the particle size distribution is large, automatically adjusting the local l-grid resolution. In addition, commensurate with the node density constraints, some nodes remain seemingly fixed in the nucleation interval, while others are placed so as to create a smoothly varying grid.

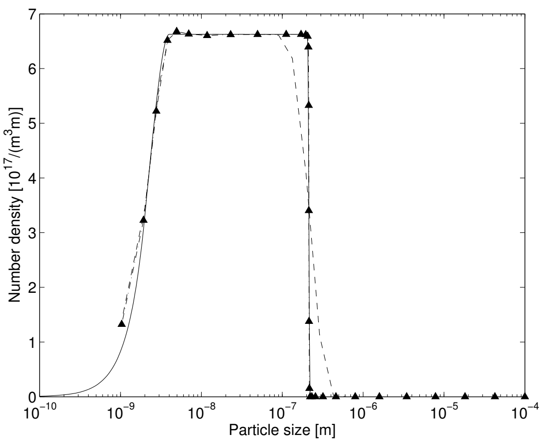
Complementary to Figure 2.14, Figures 2.15 and 2.16 show the reduction in supersaturation and the evolution of the total particle volume density along the axis of the plug flow reactor. Here, the combined FVM-EAGM scheme using 30 finite volume cells perfectly reproduces the reference results computed from the FVM using 1000 finite volume cells. In the FVM-only solution with 30 finite volume cells, by contrast, supersaturation reduction sets in too early. This is due to the diffusion of the leading moving front in l-space which leads to an overprediction of the total particle volume density and, by Eq. (2.61), to an increased consumption rate of Ba^{2+} and ${SO_4}^{2-}$. Additionally, we note that for 30 finite volume cells the FVM-only solution underestimates the total particle number density (not shown) in x-steady-state by 7.89 %, while the combined FVM-EAGM approach yields a value which is accurate to within 0.02 %.

Figure 2.17 depicts the particle size distribution at $x = 0.94\,\mathrm{m}$ obtained from the FVM-EAGM scheme with 30 finite volume cells for three different r-values (1.75, 2.5 and 5). Here, both $r = 2.5$ and $r = 5$ yield very similar results, although for $r = 5$ the node spacing is larger in l-regions in which the slope of the distribution is close to vanishing. If r is increased further, then more and more nodes leave the zero-gradient regions and move towards the steep number density fronts. On the other hand, if r is reduced, for instance to 1.75, then the moving front begins to widen, displaying signs of numerical diffusion. This indicates that the moving front is insufficiently resolved and that the node density constraints in Eqs. (2.40) and (2.45) have led to too many nodes moving away from the steep gradient region for the purpose of increasing the node density elsewhere.

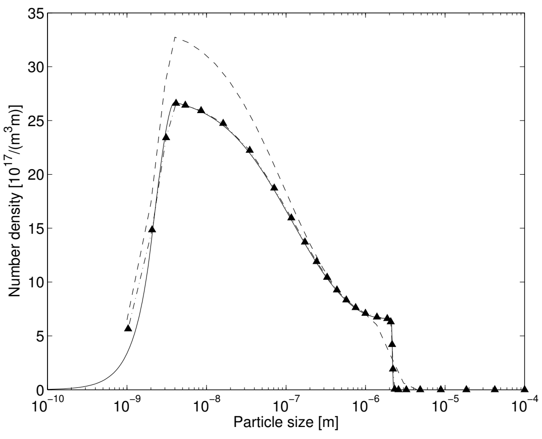
In Figure 2.18, we compare the $BaSO_4$ particle size distribution at $x = 0.19\,\mathrm{m}$ computed from the EAGM in combination with different direct discretization approaches in τ-space (GFEM, OCFEM and FVM). Here, the number of finite elements/finite volume cells is set to 30 for all three solution methods and, additionally, the source term in Eq. (2.71)



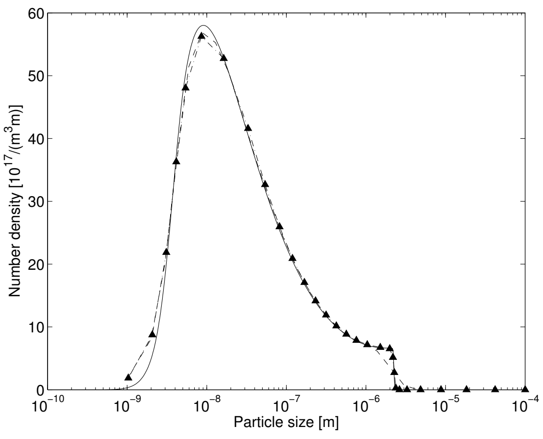
(a) $x = 9.4\,\mathrm{mm}$



(b) $x = 0.47\,\mathrm{m}$



(c) $x = 7.0\,\mathrm{m}$



(d) $x = 42.2\,\mathrm{m}$

Figure 2.14 The $BaSO_4$ particle size distributions at four cross-sections of the steady-state plug flow reactor computed from the FVM both with and without the EAGM.

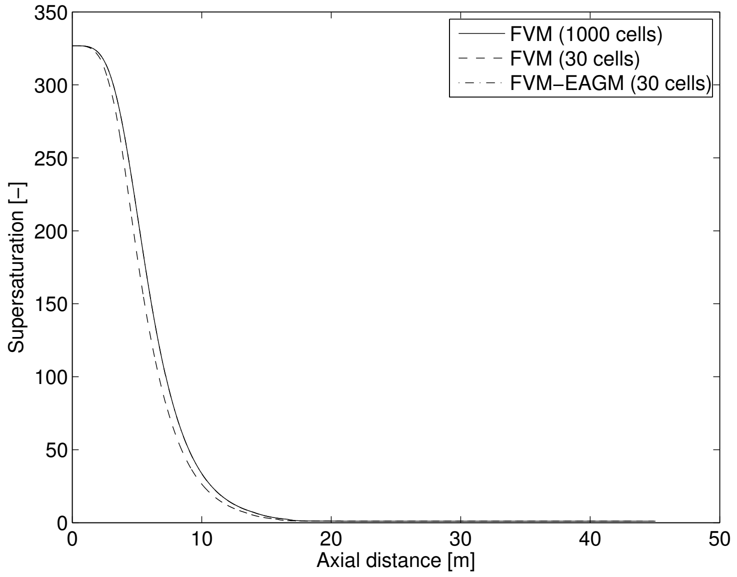


Figure 2.15 Reduction in supersaturation along the steady-state plug flow reactor for the numerical solutions depicted in Figure 2.14.

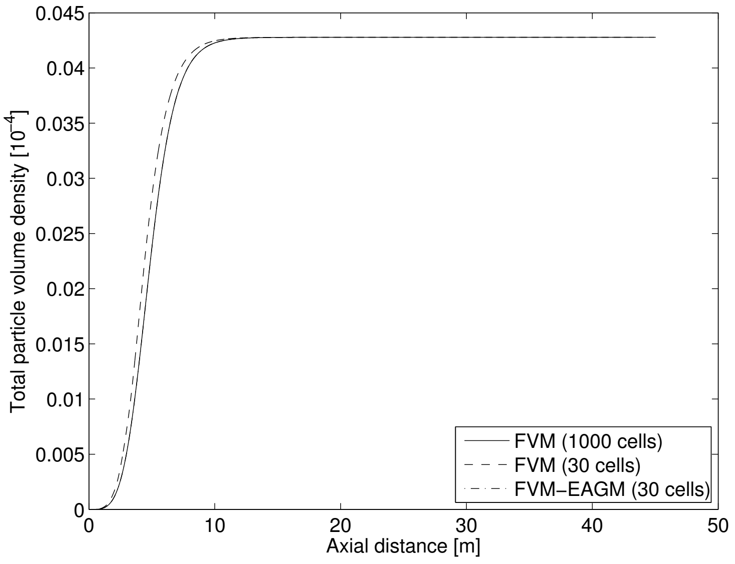


Figure 2.16 Evolution of the total particle volume density along the steady-state plug flow reactor for the numerical solutions shown in Figure 2.14.

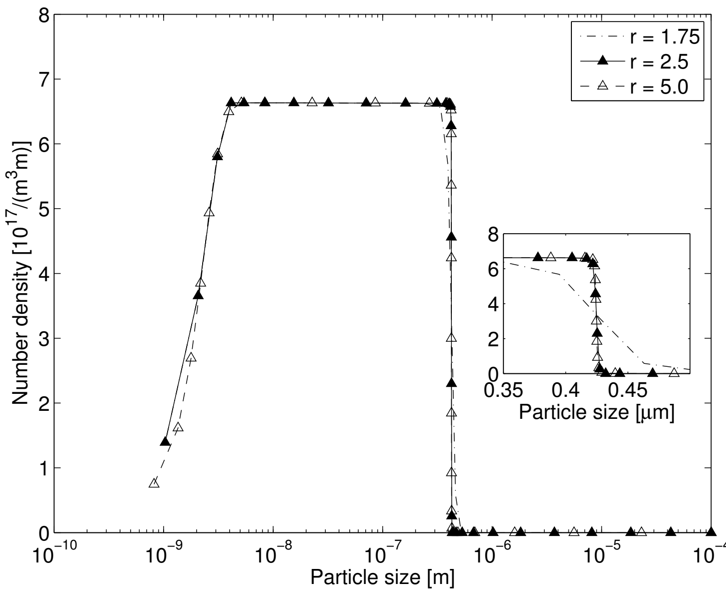


Figure 2.17 The $BaSO_4$ particle size distribution at $x = 0.94$ m of the steady-state plug flow reactor computed from the FVM-EAGM approach using 30 finite volume cells and three different values of the maximum grid stretching r.

is omitted, $\dot{\mathbf{s}}(N(\cdot, x)) \equiv \mathbf{0}$. For the GFEM, this prevents the high frequency oscillations which appear in the particle size distribution due to numerical dispersion to feed back into the fluid composition and, in particular, to affect the nucleation and growth rates.

Figure 2.18 indicates that the resulting particle size distributions mainly differ with regard to the resolution of the steep moving front which has developed at the right end of the distribution. The FVM maintains the near-discontinuity very well, while the GFEM results are severely compromised by spurious oscillations and the OCFEM solution displays some diffusion of number density in l-space. Despite these differences, Figure 2.18 demonstrates that the EAGM can be combined with any common direct discretization approach and that viable results are obtained without adjusting the EAGM parameters to a particular solution method. This corroborates our observation that the EAGM parameters either regularize the monitor function (b) or control the extent to which the mesh is allowed to deform (ρ_{nuc} and r).

One of the main factors controlling the computational expense for solving the PBE in conjunction with a spatially and temporally resolved flow model is the number of grid points that are required to achieve convergence in particle property space. In Figures 2.19 and 2.20, we hence compare the convergence behavior of the EAGM with that of its fixed grid counterparts as the number of finite volume cells/finite elements is increased.

Figure 2.19 shows convergence diagrams in terms of the x-steady-state total particle number density for the FVM and FVM-EAGM as well as the OCFEM and OCFEM-EAGM schemes. Here, the total particle number density was computed as the zeroth moment of the particle size distribution taken from a cross-section far downstream of the

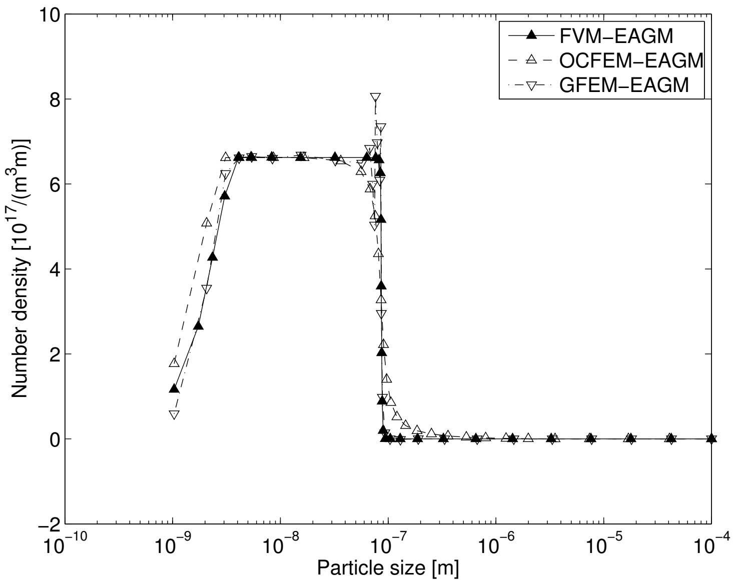
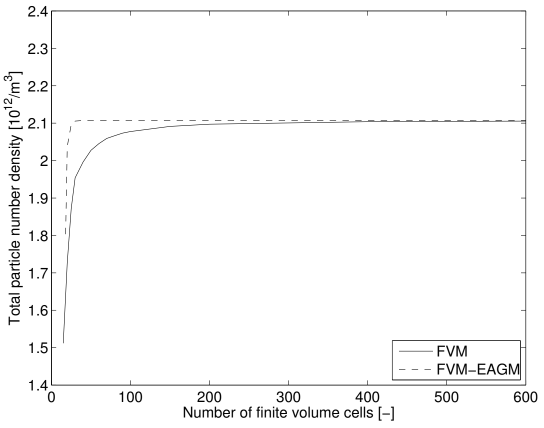


Figure 2.18 The $BaSO_4$ particle size distribution at $x = 0.19$ m of the steady-state plug flow reactor obtained from the EAGM combined with different direct discretization schemes (GFEM, OCFEM and FVM) using 30 finite elements/finite volume cells.

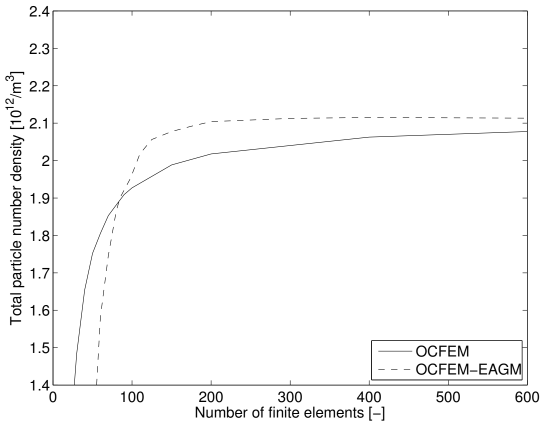
steady-state plug flow reactor in which the supersaturation has decreased below 1.01. Considering, for instance, a relative convergence tolerance of 0.05 %, the FVM-EAGM results converge at 30 to 40 finite volume cells, while the FVM-only scheme requires approximately 1000 finite volume cells to achieve the same accuracy. The combined OCFEM-EAGM scheme, on the other hand, reaches convergence at about 2500 finite elements, whereas the OCFEM-only solutions did not converge at a reasonable number of finite elements. In line with our conclusion in Section 2.6.1, Figure 2.19 thus indicates that, for a given accuracy in the zeroth moment, incorporating the EAGM reduces the required number of grid points by more than one order of magnitude.

Complementary to Figure 2.19(a), Figure 2.20 depicts the convergence of the volumetric particle size distribution in x-steady-state computed from the FVM (Figure 2.20(a)) and the combined FVM-EAGM approach (Figure 2.20(b)). For the FVM-EAGM scheme, the volumetric particle size distributions converge much quicker than for the FVM-only discretization as the number of cells is increased. For instance, the volumetric particle size distribution computed from the FVM-EAGM scheme using 60 finite volume cells almost perfectly reproduces the volumetric particle size distribution computed from the FVM-only approach with 1000 finite volume cells.

Table 2.2 lists the average runtime of a single integration step on an Intel Xeon E5-2687W processor for the FVM, OCFEM and GFEM discretization approaches both with and without the EAGM. The time measurements indicate that the FVM and FVM-EAGM schemes consume approximately the same amount of runtime as the OCFEM and OCFEM-EAGM methods, respectively, while the GFEM and GFEM-EAGM schemes are computa-

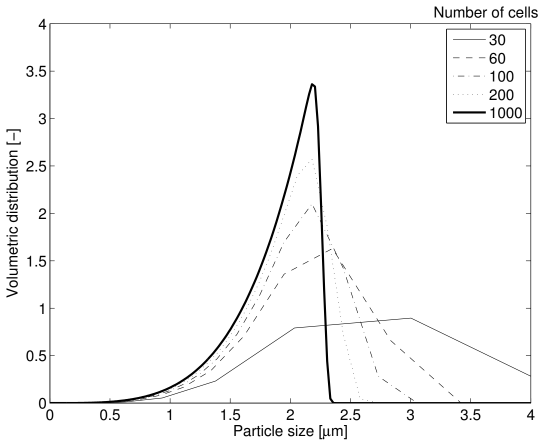


(a) $\kappa = 1/3$ finite volume method (FVM)

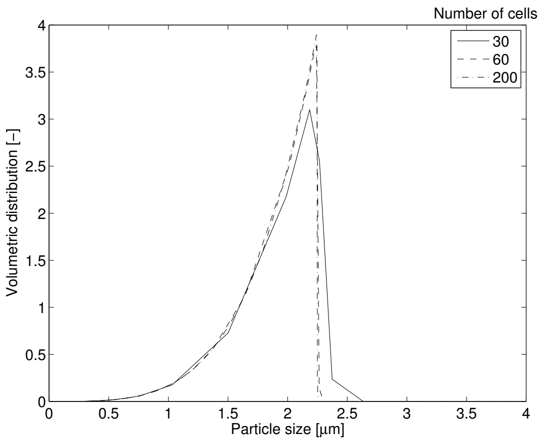


(b) Orthogonal collocation finite element method (OCFEM)

Figure 2.19 Convergence diagram for the zeroth moment of the fully developed $BaSO_4$ particle size distribution in the steady-state plug flow reactor. Here, the particle size distributions were obtained using the FVM and OCFEM both with and without the EAGM.



(a) FVM



(b) Combined FVM-EAGM scheme

Figure 2.20 Convergence diagram for the fully developed volumetric $BaSO_4$ particle size distribution in the steady-state plug flow reactor computed from the FVM both with and without the EAGM.

Plug flow	Method	Adaptivity	Elements/cells	Average runtime [s]
Steady-state	FVM		1000	6.88×10^{-4}
		EAGM	60	5.68×10^{-5}
			30	1.53×10^{-5}
		EAGM	30	3.65×10^{-5}
	OCFEM		6000	2.90×10^{-2}
		EAGM	600	4.59×10^{-4}
			30	1.37×10^{-5}
		EAGM	30	3.42×10^{-5}
	GFEM		30	2.05×10^{-5}
		EAGM	30	8.68×10^{-5}
Unsteady	FVM		1000	9.40×10^{-2}
			30	3.39×10^{-3}
		EAGM	30	7.74×10^{-3}

Table 2.2 Comparing the average runtime per time step ($\Delta t = 10^{-4}\,\mathrm{s}$) of the $\mathrm{BaSO_4}$ precipitation example for different direct discretization methods (GFEM, OCFEM, FVM) both with and without the EAGM.

tionally more expensive. Comparing the EAGM schemes with their fixed grid counterparts, we find that the FVM and OCFEM solvers slow down by a factor of about 2.5 when the EAGM is activated and that the GFEM solver runs about 4 times slower with the EAGM. This increase in runtime is mainly caused by the numerical scheme for computing the new coordinate transformation for the next time step. In case of the GFEM, the additional performance loss is due to the fact that the GFEM-EAGM scheme requires two Gauss points to exactly integrate the growth term matrix, while for the original GFEM one Gauss point suffices. In view of the convergence diagrams in Figures 2.19 and 2.20, moreover, we observe that by incorporating the EAGM the runtimes for computing *converged* solutions for the $\mathrm{BaSO_4}$ particle size distribution in x-steady-state decrease by over an order of magnitude.

2.6.2.3 $\mathrm{BaSO_4}$ precipitation in an unsteady plug flow reactor

In the present section, we consider the precipitation of $\mathrm{BaSO_4}$ particles in an unsteady plug flow reactor. In comparison to Section 2.6.2.2, this introduces the additional complication that the particle size distribution is parameterized both by a spatial coordinate $x \in [0, X]$, $X = 4.7\,\mathrm{cm}$, and time $t \geq 0$. The particle number density $N(l, x, t)$ obeys the transport equation

$$\begin{aligned}\frac{\partial N(l,x,t)}{\partial t} + u\frac{\partial N(l,x,t)}{\partial x} + \frac{\partial (G(\mathbf{Y}(x,t))N(l,x,t))}{\partial l}\\ = \frac{\partial}{\partial x}\left(D_p \frac{\partial N(l,x,t)}{\partial x}\right) + \dot{s}(l, \mathbf{Y}(x,t)),\end{aligned} \tag{2.76}$$

while the fluid phase composition $\mathbf{Y}(x,t)$ evolves according to

$$\frac{\partial \mathbf{Y}(x,t)}{\partial t} + u\frac{\partial \mathbf{Y}(x,t)}{\partial x} = \frac{\partial}{\partial x}\left(D\frac{\partial \mathbf{Y}(x,t)}{\partial x}\right) + \dot{\mathbf{s}}(N(\cdot,x,t)), \tag{2.77}$$

where D_p represents the particle diffusivity, D denotes the common diffusivity of the fluid's constituents into the mixture (see Table 2.1) and the sink term $\dot{\mathbf{s}}(N(\cdot,x,t))$ is defined similarly to Section 2.6.2.

Initially, the reactor contains pure water and no $BaSO_4$ particles such that $\mathbf{Y}(x,t=0) = (1,0,0,0,0)^T$ and $N(l,x,t=0) = 0$ identically for $x \in [0,X]$. The reactive mixture enters the domain at the left boundary $x=0$, where $\mathbf{Y}(0,t)$ coincides with the composition $\mathbf{Y}_0$ in Eq. (2.60) and $N(l,0,t)$ vanishes identically. At the right boundary $x = X$, on the other hand, a zero gradient outflow boundary condition is imposed on all scalar fields, $\partial\mathbf{Y}(X,t)/\partial x = 0$ and $\partial N(l,X,t)/\partial x = 0$.

For the unsteady plug flow reactor, the EAGM is configured in the default way of Section 2.6.2.2 and the fractional time step amounts to $\Delta t = 10^{-4}$ s. The convection-diffusion fractional step is solved using a high-resolution finite volume discretization for the convective terms and a standard finite volume scheme based on second-order central differences for the diffusion term [87]. Note that, for consistency, the molecular diffusivity D_p in the PBE fractional step (Eq. (2.16)) needs to be augmented by the artificial diffusivity introduced by the discretization of the spatial convection term in the PBE convection-diffusion step (Eq. (2.15)). At each spatial cell face $x_{j-\frac{1}{2}}$, this amounts to replacing D_p by an effective diffusivity $D_p'(x_{j-\frac{1}{2}},t)$,

$$D_p'(x_{j-\frac{1}{2}},t) = D_p + \frac{u}{2}\left(x_j - x_{j-1}\right)\Big(1 - r(x_{j-\frac{1}{2}},t)\Big), \tag{2.78}$$

where $r(x_{j-\frac{1}{2}},t)$ represents a common spatial flux limiter for all discrete number densities and x_{j-1} and x_j indicate the cell centers on either side of $x_{j-\frac{1}{2}}$. For the discretization in physical space, we employ a uniform grid with 100 finite volume cells, yielding a spatial CFL number of 0.2. The time discretization of the convection-diffusion step is based on the second order accurate Crank-Nicolson scheme.

For the FVM and the combined FVM-EAGM scheme, Figure 2.21 depicts the time evolution of the $BaSO_4$ particle size distribution, the local supersaturation as well as the total particle volume density along the plug flow reactor. In the beginning, at time $t = 5\times10^{-3}$ s, the precipitation process is dominated by particle nucleation near the inlet. Here, a moving number density front forms which slightly diffuses along the axial coordinate and, in this way, communicates the l-grid of the leading particle population to the flow domain further downstream. By consequence, the l-nodes ahead line up with the l-grid of the leading particle size distribution during the first few time steps.

At time $t = 2.5\times10^{-2}$ s, the first $BaSO_4$ particles have progressed halfway through the reactor. Figure 2.21(b) shows that the particle size distribution associated with the leading particles is slightly narrower and taller than the distributions characterizing the particles which were formed in their wake. This indicates that in the leading particle population nucleation takes place at a higher rate than in the succeeding ones, while the converse is true for particle growth. After $X/u = 5 \times 10^{-2}\,\mathrm{s}$, the leading particles have reached the reactor outlet and the number density peak in the l-x domain begins to disappear (Figure 2.21(c)). Figure 2.21(d), finally, corresponds to a steady-state in time of both the particle number density and the fluid composition.

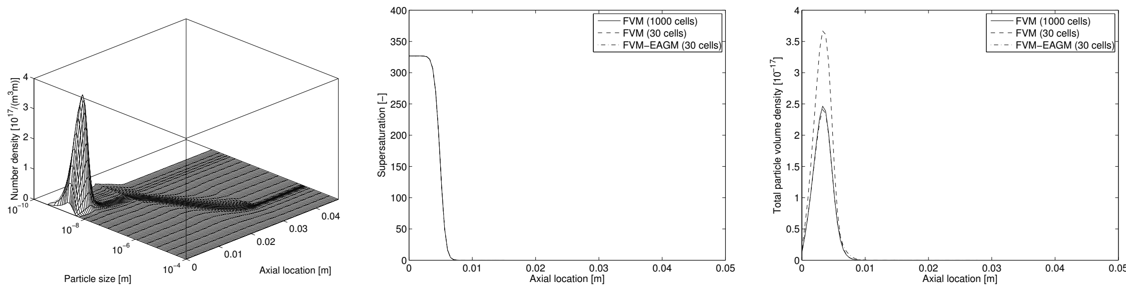
For comparison, the solid and dashed lines in columns two and three of Figure 2.21 indicate the results obtained from the FVM-only scheme with 1000 and 30 finite volume cells, respectively. For supersaturation, the solutions from the three numerical schemes match very well since, at this early stage, Ba^{2+} and ${SO_4}^{2-}$ consumption is almost negligible and the fluid composition is mainly governed by the flow dynamics. As in Figure 2.16, the total particle volume density, on the other hand, is significantly overestimated by the FVM-only scheme with 30 finite volume cells, while it is almost perfectly predicted by the combined FVM-EAGM approach using the same amount of cells.

The bottom part of Table 2.2 shows average runtimes for a single time step of the unsteady plug flow model. In line with our findings for the steady-state plug flow reactor, activating the EAGM leads to a runtime increase by a factor of approximately 2.5. However, if we compare the runtimes of the FVM-only implementation using 1000 finite volume cells with that of the combined FVM-EAGM approach using 30 cells, the latter shows a speedup of more than one order of magnitude for a comparable degree of accuracy.

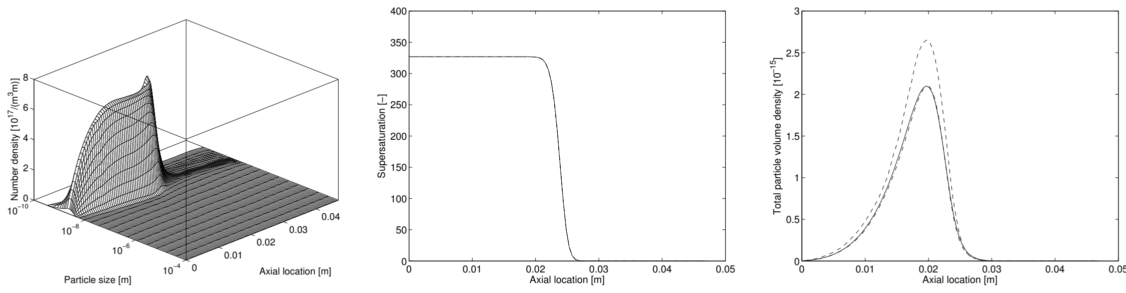
2.7 Chapter summary

In this chapter, we presented an explicit solution-adaptive technique for discretizing the spatially inhomogeneous and unsteady PBE along a one-dimensional particle property space. Our method is based on a space and time dependent coordinate transformation which maps physical particle property space onto a transformed particle property space and is controlled by the shapes of recent particle property distributions. A main feature of our approach is that the coordinate transformation can be marched in time explicitly since its evolution over the next time step is prescribed based on the current solution for the particle number density distribution and the current coordinate transformation. In comparison to many existing moving or adaptive grid approaches, this has the advantage that the node locations in physical particle property space do not appear as additional dependent variables and the size of the semi-discrete system is maintained. Under the coordinate transformation the PBE is reformulated in transformed particle property space and discretized there using a standard fixed grid discretization scheme.

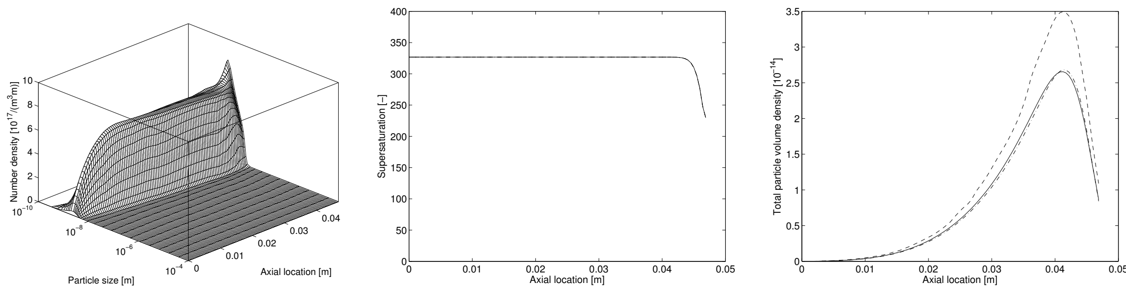
In order to accommodate nucleation source terms and to prevent grid distortion, we adopted the notion of a node density in physical particle property space and developed a robust numerical scheme by which the coordinate transformation can be adjusted such that



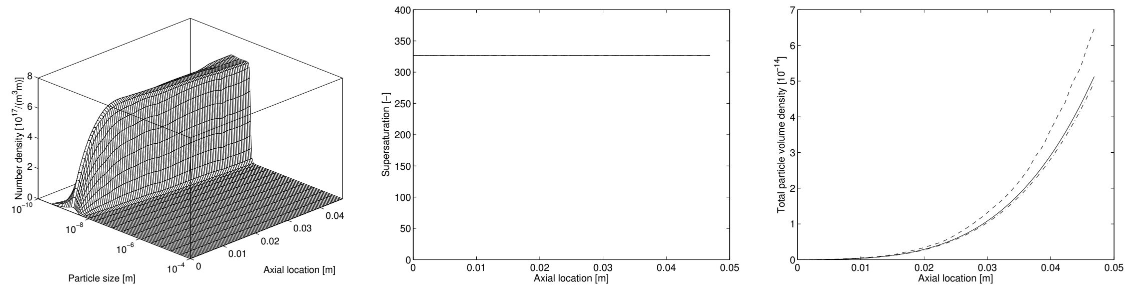
(a) Time $t = 5 \times 10^{-3}$ s



(b) Time $t = 2.5 \times 10^{-2}$ s



(c) Time $t = 5 \times 10^{-2}$ s $(= u/X)$



(d) Time $t = 10^{-1}$ s

Figure 2.21 Time evolution of the $BaSO_4$ particle size distributions (first column), the $BaSO_4$ supersaturation (second column) and the total particle volume density (third column) along the unsteady plug flow reactor computed from the FVM-EAGM scheme using 30 finite volume cells (dash-dotted lines). For reference, the solid and dashed lines in columns two and three indicate the results obtained from an FVM-only scheme with 1000 and 30 finite volume cells, respectively.

the node density at a particular particle property value does not fall below a prescribed minimum node density and the grid stretching does not exceed a given maximum value. In the $BaSO_4$ precipitation example that we investigated, the adjustment scheme was controlled by two grid parameters which can be interpreted as the minimum node density in the nucleation interval and the maximum admissible grid stretching.

In view of general applicability, an advantage of our adaptive solution technique is that it can be combined with any fixed grid discretization scheme in transformed particle property space and, similarly, with any time integrator and physical space discretization scheme. Moreover, owing to its simplicity, the adaptive solution technique is easy to implement and, in particular, to integrate into existing academic or commercial solution software.

As examples we considered the advection of a unit step in a constant flow field as well as the precipitation of $BaSO_4$ particles from an aqueous solution in a plug flow reactor. In order to demonstrate its flexibility, the explicit adaptive grid method was combined with three different direct discretization schemes in transformed particle property space: the Galerkin finite element method, a fully upwinded orthogonal collocation finite element method and a high resolution finite volume method. For the latter two methods, we demonstrated that incorporating the explicit adaptive grid approach reduces the number of grid points necessary to achieve a given accuracy by more than one order of magnitude. Additionally, runtime measurements confirmed that the explicit adaptive grid scheme is computationally very efficient.

The developments reported in this chapter are motivated by our objective to incorporate the PBE as a model for a polydispersed particulate phase into existing spatially and temporally resolved reacting flow models. Currently, the standard method to this end is to discretize the PBE on a fixed grid in particle property space and to augment the vector of fluid phase scalars by the discrete number density fields. The main disadvantage here is that the fixed grid in particle property space often needs to be very fine in order to represent all particle property distributions which may be present at any spatial location and any point in time with comparable accuracy. Our explicit adaptive grid scheme alleviates this requirement: The distribution of resolution in physical particle property space changes both across the flow domain and in time, while spatial consistency is maintained with respect to the transformed discrete number densities.

In future times, we intend to apply the explicit adaptive grid method to problems also involving particle aggregation and breakage. Furthermore, we wish to extend the formulation to a higher dimensional particle property space, incorporating, for instance, particle morphology or inertia (see Section 7.2).

Chapter 3

An LES-PBE-PDF approach for modelling turbulent precipitation and condensation

3.1 Introduction

While the PBE governs the particle property distribution associated with a polydispersed particulate phase, the carrier fluid is commonly described in terms of reactive scalars which are often taken as species mass fractions supplemented by a calorific mixture quantity. In this chapter, we present a comprehensive methodology for incorporating the PBE and the reactive scalars' transport equations into the large eddy simulation (LES) framework, while allowing for a computationally efficient numerical solution procedure. Our approach for resolving the interaction between turbulence and chemical reactions/particle formation is based on the formulation of an evolution equation for the filtered joint scalar-number density probability density function (*pdf*) associated with a single realization of the scalars and number density fields [174]. In view of a numerical solution method, we present a statistically equivalent reformulation of the joint scalar-number density *pdf* in terms of an ensemble of Eulerian stochastic fields [71, 179, 200]. The stochastic field equation which is physically associated with the particle number density is discretized in particle property space using the explicit adaptive grid scheme developed in Chapter 2.

This brief introduction reflects the three main constituents which a model for polydispersed particle formation in a turbulent reacting flow comprises [173]:

1. Representation of the particle property distribution
2. A turbulent flow model
3. Turbulence-chemistry and turbulence-particle formation interaction

In Table 3.1, the current approaches for modelling turbulent reacting flows with polydispersed particle formation are classified according to these three submodels. Here, each 'class' encompasses a few representative references. In the following, we briefly review these approaches, laying emphasis on the physical assumptions they involve as well as on the computational expense they entail.

The first submodel mentioned above is largely independent of the turbulent nature of the flow and also plays an important role in laminar flow configurations. Here, the main question is how the particle property distribution may be represented in terms of a finite number of 'particle phase' scalars. One strategy, in this regard, consists in replacing the PBE by evolution equations for a few integral properties of the particle property distribution. This leads to the method of moments which was introduced by Hulburt and Katz [77] and has, until now, been one of the most popular approaches for characterizing particulate phases in spatially inhomogeneous flows. Frequently, low order moments are important from an engineering perspective, for example, the total particle number density or volume fraction [168], or directly accessible by measurement techniques [56]. On the minus side, the moment transport equations are only closed for certain functional forms of the particle growth, coagulation and breakage rates [18, 46] and, in general, require an assumption on the shape of the particle property distribution to be closed. In the context of droplet condensation in a turbulent jet, Garmory and Mastorakos [59] assumed a log-normal droplet size distribution [164], for instance.

Marchisio et al. [117] applied an alternative closure scheme, the quadrature method of moments (QMOM) [120]. Here, integrals with respect to the particle property distribution are computed using a quadrature approximation whose weights and abscissas are expressed in terms of a finite number of moments. The direct quadrature method of moments (DQMOM), moreover, is a variant of the QMOM in which evolution equations for the quadrature weights and abscissas are solved in place of the moment equations [214]. One difficulty associated with quadrature-based moment methods is related to the conservation of moment realizability during spatial convection/diffusion and specially designed spatial discretization schemes may become necessary [202, 203]. Only in passing we mention the formulation of other moment-based methods such as MOMIC [55] and HMOM [130], also see Rigopoulos [173] and Marchisio and Fox [115]. Following Raman and Fox [168], moment-based methods are computationally very economical, in particular, if the particle property space comprises more than one dimension.

In recent years, researchers have also investigated approaches for directly discretizing the spatially inhomogeneous PBE in particle property space. Albeit more expensive, these schemes have, by now, become a computationally viable alternative to moment-based methods. Cheng et al. [31] applied the discretized population balance (DPB) scheme of Kumar and Ramkrishna [94] to investigate the precipitation of $BaSO_4$ crystals in a stirred tank. Within the scope of a DPB, particle property space is divided into bins and the particles in each bin collectively interact with the particles in other bins such that particular moments of the particle property distribution are conserved. Campos and Lage [30] pointed out that, strictly, DPBs are not function approximation methods and that the particle size distribution hence converges slower than its moments as the number of bins is increased. This is at variance with direct discretization approaches such as finite volume [209] or finite element methods [39, 40]. Here, however, both the particle property distribution and all of its moments are affected by a discretization error [175]. In the present work, we combine a high resolution finite volume method with the explicit adaptive grid approach detailed in Chapter 2 for discretizing the stochastic field counterpart (see below) of the inhomogeneous PBE in particle property space [186].

The second submodel indicated above refers to the physical concept which underlies the representation of the turbulent carrier flow. Most of the investigations listed in Table 3.1 employ a description based on the Reynolds averaging formalism (RANS). Recently, however, researchers have also invoked LES in order to analyze polydispersed particle formation in turbulent flows. Makowski et al. [113], for example, investigated $BaSO_4$ precipitation in an impingement T-mixer, while Neuber et al. [137] predicted the evolution of droplet size distributions in a turbulent jet. Although LES is considered in this work, we emphasize that our adaptive discretization scheme for the PBE as well as the PBE-PDF model for the turbulence-chemistry/particle formation interaction (see below) are not limited in validity to LES, but can similarly be applied in the context of RANS-based flow models.

From a physical perspective, the third submodel for the turbulence-chemistry and turbulence-particle formation interaction quantifies how spatial and temporal inhomogeneities that are not resolved by the turbulence model affect chemical reactions in the fluid phase along with particle formation. One of the simplest models for the turbulence-chemistry/particle formation interaction is the perfect micromixing hypothesis [31, 113, 117, 145, 208]. Physically, this model is based on the assumption that mixing of reactants on a molecular scale takes place instantaneously and that chemical reactions and particle formation are the rate-controlling steps. In the past, the perfect micromixing model has been applied also to fast precipitation reactions [31, 208], although it is difficult to justify its validity in this case.

Presumed *pdf* methods, on the other hand, are models for the turbulence-chemistry/-particle formation interaction which leverage physical insight and computational expense. In general, they may be categorized into presumed *pdf* methods based on 'tracking' scalars [154] and multi-environment/DQMOM-IEM approaches. In the first class, the reaction and particle formation kinetics are described in terms of a small number of scalars such as mixture fraction or a reaction progress variable whose joint *pdf* is assumed to take on a particular functional form. Commonly, the resulting presumed joint *pdf* is parameterized by a few statistical properties, for instance, mean and covariance, which obey modelled evolution equations. In the context of precipitation, Bałdyga and Orciuch [14] and Bałdyga and Orciuch [18] combined a presumed *pdf* method based on a β-*pdf* for mixture fraction with an interpolation scheme for relating the reactant concentrations and the moments of the particle size distribution to mixture fraction. Considering soot formation in a turbulent jet flame, Zucca et al. [214], applied a β-*pdf* approach based on mixture fraction [17] to the gas phase chemistry and, adopting a DQMOM approximation of the PBE, assumed that the weights and abscissas are statistically independent and perfectly micromixed. This idea was taken further by Mueller and Pitsch [128] who described the thermochemical state of a gas in terms of three variables and applied the HMOM in order to close the moment equations associated with a bivariate soot property distribution. Based on an assumption of statistical independence, these authors let the thermochemical variables and moments obey a marginal β-*pdf* or be perfectly micromixed.

Drawing on early Lagrangian micromixing models, multi-environment methods were first rationalized by Fox [51] who later developed the DQMOM-IEM method [50]. Here, the joint *pdf* of the fluid and particle phase scalars is represented as a linear combination of the *pdfs* associated with several model flow realizations (or environments) which may exchange likelihood (or volume fraction) with each other according to a micromixing model. In the DQMOM-IEM approach, this interaction is formalized by imposing the condition that the evolution equations for the mean scalars in each environment as well as the associated volume fractions exactly reproduce the evolution of the first few unmixed moments of the joint scalar *pdf*. In the context of $BaSO_4$ precipitation in a tubular reactor, Marchisio et al. [116] and Marchisio et al. [118] formulated a three-environment model in terms of mixture fraction, a reaction progress variable as well as the first few moments of the particle size distribution. Woo et al. [209] also considered a three-environment model, but combined it with a discrete finite volume-based representation of the particle size distribution in order to analyze an antisolvent crystallization process in a semibatch stirred reactor. The DQMOM-IEM method, on the other hand, was applied by Gavi et al. [60] in conjunction with the QMOM and by Akroyd et al. [4] in combination with the MOMIC.

Finally, transported *pdf* approaches rank amongst the most comprehensive, albeit computationally expensive closure schemes for the turbulence-chemistry interaction. These methods are based on a modelled evolution equation for the one-point, one-time joint scalar *pdf*. Since their formulation is independent of tracking variables, transported *pdf* methods may be applied to different flow configurations (premixed, non-premixed, partially premixed) without adjustment. Considering the precipitation of silica particles in a tubular reactor, Falk and Schaer [46] seem to have been the first to formulate a joint scalar-moment *pdf* transport equation. A similar approach was adopted by Garmory and Mastorakos [59] who investigated the nucleation and growth of an aerosol in a turbulent jet.

While joint scalar-moment transported *pdf* methods are well-established by now, incorporating the complete PBE into a transported *pdf* approach is not yet widely adopted. This idea was originally put forward by Rigopoulos [174] who showed how the discrete number densities resulting from a discretization of the PBE in particle property space may be accounted for in an evolution equation for the joint scalar-discrete number density *pdf*. The main advantage of the so-called PBE-PDF method is that it allows for the prediction of the particle property distribution and is able to accommodate any fluid or particle phase kinetics without approximation. The only remaining closure which the PBE-PDF methods requires is related to the two-point process of molecular diffusion. Di Veroli and Rigopoulos [38–40] first demonstrated the effectiveness of the PBE-PDF method in the context of RANS by validating PBE-PDF predictions for a finite element discretization of the particle size distribution in two experimental test cases. Additionally, they were able to show that a numerical solution of the joint scalar-discrete number density *pdf* transport equation is computationally viable and can be achieved in reasonable computing times by using a Lagrangian stochastic particle solver [161]. Recently, Neuber et al. [137] applied the PBE-PDF method based on a DPB in the context of LES and obtained predictions of droplet size distributions in a turbulent jet. Specifically, these authors combined a Lagrangian stochastic particle scheme based on the modified Curl micromixing model with a generalized multiple mapping conditioning (MMC) approach in order to localize micromixing in composition space. Upon deemphasizing the proximity of stochastic particles in physical space, this led to a so-called sparse particle MMC-LES approach which was found to not only achieve a similar accuracy as a dense stochastic particle solver without MMC localization, but also yield a reduction in runtime of one order of magnitude.

By design, the PBE-PDF model accounts for the influence of turbulence on particle growth and particle-particle interactions via the particle number density. In an early work, Warshaw [207] suspected that these effects are important in laboratory scale systems, but may be negligible in large-scale volumes with many particles such as clouds. Subsequently, only few authors have addressed the number density correlations which appear in a turbulent flow model. In the context of nucleation and growth of $BaSO_4$ particles, Di Veroli and Rigopoulos [38] demonstrated that omitting the influence of turbulence on the number density can lead to noticeable differences in the predicted particle mean diameter, in particular, if micromixing is the rate-controlling step. Interestingly, both Rigopoulos [174] and Di Veroli and Rigopoulos [38] observed that for some mixing conditions the modelling errors in kinetic rates computed using the perfect micromixing assumption and the errors due to omission of growth rate-number density correlations compensate each other.

The contribution of this chapter to our overall objective is threefold: First, we formulate an LES-based evolution equation for the filtered joint scalar-number density *pdf* that is independent of a particular particle property discretization. Second, a Eulerian stochastic field formulation is presented which, in a statistical sense, reproduces the evolution of the joint scalar-number density *pdf* and is compatible with both fixed and adaptive grid discretizations of particle property space. Finally, we consider the precipitation of $BaSO_4$ in an industrial precipitator and the condensation of an aerosol in a developed turbulent mixing layer as test cases and compare predictions of the LES-PBE-PDF model with experimental measurements or direct numerical simulation (DNS) results, respectively.

Representation of the particle property distribution		Turbulence-chemistry/particle formation interaction				
			Presumed *pdf*			
		Perfect micromixing	β-*pdf*	Multi-environment	DQMOM-IEM	Transported *pdf*
Moments	MOM	[145, 208] [113]*	[14, 18, 113]	[16, 116, 118]		[46, 59]
	XMOMY	[117]	[214] [128]*		[4, 60]	
Fixed grid	DPB	[31]				[137]*
	FVM/FEM			[209]		[38–40]
Adaptive grid	FVM					This work*

Table 3.1 An overview of different approaches for modelling polydispersed particle formation in turbulent reacting flows (DPB: Discretized Population Balance, FEM: Finite Element Method, FVM: Finite Volume Method, MOM: Method of Moments, XMOMY: Quadrature-based Method of Moments, for example, QMOM, DQMOM, HMOM, MOMIC). From a modelling perspective, the main challenges are concerned with (i) the representation of the particle property distribution (rows), (ii) the description of the turbulent carrier flow (* LES, otherwise RANS) and (iii) the interaction between turbulence and chemical reactions/particle formation (columns).

Here, it is demonstrated that, based on our numerical solution scheme, model solutions can be efficiently and accurately obtained on a modern computing system.

This chapter is structured as follows: In Section 3.2, we first record the governing equations and briefly review the LES concept. Subsequently, the evolution equation for the joint scalar-number density *pdf* is derived and the statistically equivalent stochastic field equations are developed. In Sections 3.3 and 3.4, we provide details on the implementation of the numerical solution scheme and assess the predictive quality and computational viability of the LES-PBE-PDF approach in two test cases. This is followed by our conclusions in Section 3.5.

3.2 The LES-PBE-PDF framework for turbulent reacting flows with particle formation

In this section, we present the LES-PBE-PDF approach for modelling the evolution of a particle property distribution in a turbulent carrier flow. Figure 3.1 schematically illustrates the main constituents of the model as well as the interconnecting rationale which leads to the final model formulation. The remaining part of this section can be seen as a guide through this diagram, leading from left to right.

3.2.1 Governing equations

Following the motivation in Section 2.3, we consider a particulate phase that is polydispersed with respect to a characteristic particle property l, $l \in [0, L]$. From a Eulerian perspective, the evolution of the number density $N(l, \mathbf{x}, t)$ associated with the particulate phase can be described by the PBE in Eq. (2.1). The carrier fluid mixture,

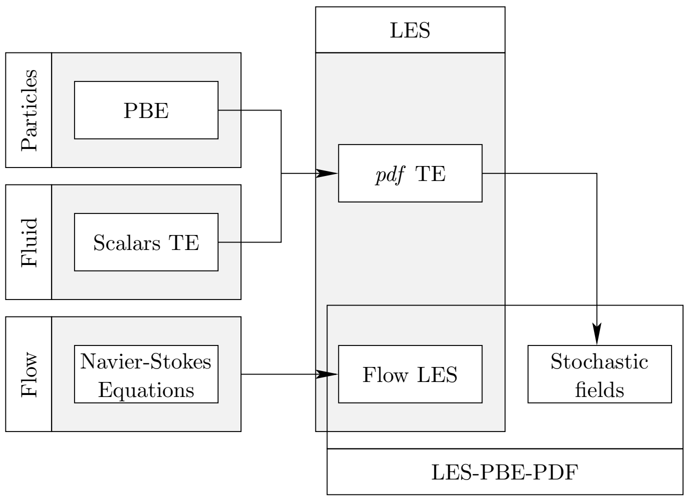


Figure 3.1 An overview of the LES-PBE-PDF framework for modelling turbulent reacting flows with particle formation (TE: Transport Equation).

on the other hand, is commonly characterized in terms of reactive scalars $\mathbf{Y}(\mathbf{x},t) = (Y_1(\mathbf{x},t),\ldots,Y_{n_s}(\mathbf{x},t))^T$ which may represent species mass fractions and temperature, for instance. If the fluid density $\rho(\mathbf{x},t)$ is constant, then the reactive scalars $Y_i(\mathbf{x},t)$, $i = 1,\ldots,n_s$, evolve according to

$$\begin{aligned}\frac{\partial Y_i(\mathbf{x},t)}{\partial t} &+ \sum_{j=1}^{3} \frac{\partial\left(u_j(\mathbf{x},t) Y_i(\mathbf{x},t)\right)}{\partial x_j} \\ &= \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D(\mathbf{x},t)\frac{\partial Y_i(\mathbf{x},t)}{\partial x_j}\right) + \dot{\omega}_i(\mathbf{Y}(\mathbf{x},t), N(\cdot,\mathbf{x},t))\end{aligned} \tag{3.1}$$

in the ambient velocity field $\mathbf{u}(\mathbf{x},t)$, where $\dot{\omega}_i(\mathbf{Y}(\mathbf{x},t), N(\cdot,\mathbf{x},t))$ represents the rate at which scalar i is produced or depleted, respectively. Note that, owing to the consumption of fluid phase species on account of particle formation, $\dot{\boldsymbol{\omega}}$ is, in general, a functional of the local particle property distribution $N(\cdot,\mathbf{x},t)$.

In Eq. (3.1), $D(\mathbf{x},t)$ denotes a mixture averaged diffusivity that is common to all scalars. Strictly, this presents an approximation of the more general case of fluid phase differential diffusion, whereby each scalar is characterized by a different mixture averaged diffusion coefficient. In turbulent flows, differential diffusion is frequently omitted, although it may affect the location of reaction zones in mixture fraction space. In part, the simplification of considering a single diffusion coefficient $D(\mathbf{x},t)$ may be owed to the fact that micromixing models based on the IEM-concept (Interaction by Exchange with the Mean; see Sections 3.2.4 and 4.3.4) have not yet been extended to include differential micromixing rates. In line with References [84, 85], we thus do not consider differential diffusion within the fluid phase in this work, although it is emphasized that our LES-PBE-PDF framework does not inherently prevent its inclusion.

For future reference, we also record the continuity equation for a constant density flow,

$$\sum_{i=1}^{3} \frac{\partial u_i(\mathbf{x},t)}{\partial x_i} = 0. \tag{3.2}$$

Initially, at time $t = t_0$, the fluid mixture in the flow domain possesses the composition $\mathbf{Y}_0(\mathbf{x})$, while the number density is given by $N_0(l, \mathbf{x})$,

$$\mathbf{Y}(\mathbf{x}, t_0) = \mathbf{Y}_0(\mathbf{x}), \tag{3.3}$$

$$N(l, \mathbf{x}, t_0) = N_0(l, \mathbf{x}). \tag{3.4}$$

Additionally, both the reactive scalars $\mathbf{Y}(\mathbf{x}, t)$ and the number density $N(l, \mathbf{x}, t)$ obey standard Dirichlet and/or Neumann boundary conditions along the boundary of the flow domain. If l represents a measure of particle size, we also have the boundary condition in Eq. 2.4 at $l = 0$.

3.2.2 Large eddy simulation

The main idea in LES is to resolve the large energy-containing scales of a single realization of the flow field and, in this way, to reduce the sensitivity of predictions with respect to the closure approximations for turbulent transport. Conceptually, this approach is very different from RANS-based turbulence models in which an expectation over a large number of realizations is computed. In LES, the fields describing a single realization are formally subjected to a linear operator which (in an abstract fashion) reduces the contained information without affecting the representation of large scale flow structures (eddies). Following Sagaut [180], we introduce the LES-operator $\bar{\cdot}$ as a linear operator on the set of admissible fields $\phi(\mathbf{x}, t)$ obeying two properties: First, a constant field $\phi(\mathbf{x}, t) = a \in \mathbb{R}$ is mapped onto itself,

$$\overline{\phi}(\mathbf{x}, t) = \overline{a} = a = \phi(\mathbf{x}, t). \tag{3.5}$$

Second, differentiation of a field $\phi(\mathbf{x}, t)$ with respect to $\mathbf{x}$ or t and application of the LES-operator commute,

$$\overline{\frac{\partial \phi(\mathbf{x}, t)}{\partial x_i}} = \frac{\partial \overline{\phi}(\mathbf{x}, t)}{\partial x_i}, \quad i = 1, \dots, 3, \tag{3.6}$$

$$\overline{\frac{\partial \phi(\mathbf{x}, t)}{\partial t}} = \frac{\partial \overline{\phi}(\mathbf{x}, t)}{\partial t}. \tag{3.7}$$

In addition, we assume that, formally, the LES-operation on $\phi(\mathbf{x}, t)$ can be expressed in terms of a scalar function $G(\mathbf{x}, \mathbf{x}') \geq 0$ in the following way

$$\overline{\phi}(\mathbf{x}, t) = \int_\Omega G(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x}', t)\, d\mathbf{x}'. \tag{3.8}$$

Since $G(\mathbf{x},\mathbf{x}')$ is non-negative and by the property in Eq. (3.5), $G(\mathbf{x},\mathbf{x}')$ possesses all properties of a *pdf*. Frequently, $G(\mathbf{x},\mathbf{x}')$ is referred to as a spatial filter kernel. For our developments, however, this interpretation is not necessary and, apart from the existence of a kernel $G(\mathbf{x},\mathbf{x}')$, its precise definition is immaterial at this point. Ultimately, the kernel $G(\mathbf{x},\mathbf{x}')$ is defined implicitly through the closure approximations for the turbulent transport term and the molecular mixing in Sections 3.2.3 and 3.2.4 [162, Section 13.4.3]. Introducing the identity

$$\phi(\mathbf{x},t) = \int \psi\delta(\psi - \phi(\mathbf{x},t))\,d\psi \tag{3.9}$$

into Eq. (3.8) leads to

$$\overline{\phi}(\mathbf{x},t) = \int \psi f_\phi(\psi;\mathbf{x},t)\,d\psi, \tag{3.10}$$

where ψ indicates the sample space variable associated with $\phi(\mathbf{x},t)$ and $f_\phi(\psi;\mathbf{x},t)$ defined by

$$f_\phi(\psi;\mathbf{x},t) \equiv \int_\Omega G(\mathbf{x},\mathbf{x}')\delta(\psi - \phi(\mathbf{x}',t))\,d\mathbf{x}' \tag{3.11}$$

represents the LES-filtered *pdf* characterizing the instantaneous field $\phi(\mathbf{x},t)$. Eq. (3.10) shows that the LES-operation on a scalar field can be computed as an expectation of this field with respect to its filtered instantaneous *pdf*.

3.2.3 The LES-PBE-PDF framework

In the present section, we conceive the governing fields as random variables and derive a transport equation for the joint filtered *pdf* $f(\mathbf{y},n(\cdot);\mathbf{x},t)$ of the reactive scalars $\mathbf{Y}(\mathbf{x},t)$ and the particle property distribution $N(\cdot,\mathbf{x},t)$. Here, $\mathbf{y} = (y_1,\ldots,y_{n_s})$ denotes the sample space vector associated with $\mathbf{Y}(\mathbf{x},t)$ and, similarly, $n(\cdot)$ represents the sample space function indicating an element of the set of all admissible particle property distributions $N(\cdot,\mathbf{x},t)$ for a given pair $(\mathbf{x},t)$. By construction, $f(\mathbf{y},n(\cdot);\mathbf{x},t)$ uniquely characterizes all multi-l statistics of $N(\cdot,\mathbf{x},t)$ including, for instance, the LES-filtered values of the moments of the particle property distribution. Although the evolution of the governing fields is described by deterministic transport equations, randomness may enter the physical description via the initial and boundary conditions or the material properties [162, Section 3.1].

In order to obtain a governing equation for $f(\mathbf{y},n(\cdot);\mathbf{x},t)$, we follow the standard practice of first considering the fine-grained density

$$g(\mathbf{y},n(\cdot);\mathbf{x},t) = \delta\left(\mathbf{y} - \mathbf{Y}(\mathbf{x},t)\right)\delta\left(n(\cdot) - N(\cdot,\mathbf{x},t)\right), \tag{3.12}$$

where δ represents Dirac's delta distribution, and, subsequently, computing its expectation with respect to $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$,

$$\begin{aligned} f(\mathbf{y}, n(\cdot); \mathbf{x}, t) &= \overline{g(\mathbf{y}, n(\cdot); \mathbf{x}, t)} \\ &= \int \delta\left(\mathbf{y}-\mathbf{y}'\right) \delta\left(n(\cdot)-n'(\cdot)\right) f(\mathbf{y}', n'(\cdot); \mathbf{x}, t)\, d\mathbf{y}' dn'(\cdot). \end{aligned} \tag{3.13}$$

This approach was first pioneered by Lundgren [111] in the context of RANS and later generalized to LES by Pope [159].

If we consider, as an auxiliary vehicle, a grid $l_i = iL/m,\ i = 0, \ldots, m$, in particle property space, then the Dirac delta expression involving the sample space function $n(\cdot)$ in Eq. (3.12) may be understood, formally, as the limit

$$\delta\left(n(\cdot)-N(\cdot, \mathbf{x}, t)\right) = \lim_{m\to\infty} \prod_{i=0}^{m} \delta\left(n(l_i)-N(l_i, \mathbf{x}, t)\right) \tag{3.14}$$

such that the fine grained density $g(\mathbf{y}, n(\cdot); \mathbf{x}, t)$ appears as a functional in $n(\cdot)$.

The derivatives of $g(\mathbf{y}, n(\cdot); \mathbf{x}, t)$ with respect to $\mathbf{x}$ and t are given by

$$\frac{\partial g}{\partial x_j} = -\sum_{i=1}^{n_s} \frac{\partial g}{\partial y_i}\frac{\partial Y_i}{\partial x_j} - \frac{\partial g}{\partial n}\frac{\partial N}{\partial x_j}, \quad j = 1, \ldots, 3, \tag{3.15}$$

$$\frac{\partial g}{\partial t} = -\sum_{i=1}^{n_s} \frac{\partial g}{\partial y_i}\frac{\partial Y_i}{\partial t} - \frac{\partial g}{\partial n}\frac{\partial N}{\partial t}, \tag{3.16}$$

respectively, where, for conciseness, the arguments $(\mathbf{y}, n(\cdot); \mathbf{x}, t)$, $(\mathbf{x}, t)$ and $(\cdot, \mathbf{x}, t)$ of g, $\mathbf{Y}$ and N have been omitted. By introducing Eqs. (2.1) and (3.1) into Eq. (3.16) and taking into account Eqs. (3.2) and (3.15), we obtain the following evolution equation for the fine grained density $g(\mathbf{y}, n(\cdot); \mathbf{x}, t)$

$$\begin{aligned} \frac{\partial g}{\partial t} + \sum_{j=1}^{3} u_j(\mathbf{x}, t)\frac{\partial g}{\partial x_j} &= -\sum_{i=1}^{n_s} \frac{\partial g}{\partial y_i}\left(\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(D(\mathbf{x}, t)\frac{\partial Y_i}{\partial x_j}\right) + \dot{\omega}_i(\mathbf{Y}, N(\cdot, \mathbf{x}, t))\right) \\ &\quad - \frac{\partial g}{\partial n}\left(\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(D_p(\mathbf{x}, t)\frac{\partial N}{\partial x_j}\right) - \frac{\partial\left(G(\cdot, \mathbf{Y})N(\cdot, \mathbf{x}, t)\right)}{\partial l} + \dot{s}(\cdot, \mathbf{Y}, N(\cdot, \mathbf{x}, t))\right). \end{aligned} \tag{3.17}$$

For any functional $F(\mathbf{Y}, N, \partial\mathbf{Y}/\partial x_j, \partial N/\partial x_j, \ldots)$ of $\mathbf{Y}(\mathbf{x}, t)$, $N(\cdot, \mathbf{x}, t)$ and their spatial derivatives, we have the identity

$$\begin{aligned} &\overline{gF\left(\mathbf{Y}, N, \frac{\partial \mathbf{Y}}{\partial x_j}, \frac{\partial N}{\partial x_j}, \ldots\right)} \\ &= f\,\overline{\left[F\left(\mathbf{Y}, N, \frac{\partial \mathbf{Y}}{\partial x_j}, \frac{\partial N}{\partial x_j}, \ldots\right)\middle| \mathbf{Y}(\mathbf{x}, t) = \mathbf{y}, N(\cdot, \mathbf{x}, t) = n(\cdot)\right]}, \end{aligned} \tag{3.18}$$

where the vertical bar under the LES-operator represents conditioning of the expectation on the entities specified to the right of the bar. As an aid to the reader, we include a proof of Eq. (3.18) in Appendix B.1. Applying the LES-operator to Eq. (3.17) and taking into account Eqs. (3.2), (3.6), (3.7) and (3.18) leads to an evolution equation for $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$,

$$\begin{aligned}\frac{\partial f}{\partial t} + \sum_{j=1}^{3} \overline{u}_j(\mathbf{x}, t)\frac{\partial f}{\partial x_j} &= -\sum_{i=1}^{n_s} \frac{\partial}{\partial y_i} f\dot{\omega}_i(\mathbf{y}, n(\cdot)) \\ &- \frac{\partial}{\partial n} f\left(\dot{s}(\cdot, \mathbf{y}, n(\cdot)) - \frac{\partial\left(G(\cdot, \mathbf{y})n(\cdot)\right)}{\partial l}\right) - \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(\overline{u_j(\mathbf{x}, t)g} - \overline{u}_j f\right) \\ &- \sum_{i=1}^{n_s} \frac{\partial}{\partial y_i} f\overline{\left[\sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D(\mathbf{x}, t)\frac{\partial Y_i}{\partial x_j}\right)\middle| \mathbf{y}, n(\cdot)\right]} \\ &- \frac{\partial}{\partial n} f\overline{\left[\sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D_p(\mathbf{x}, t)\frac{\partial N}{\partial x_j}\right)\middle| \mathbf{y}, n(\cdot)\right]}.\end{aligned} \tag{3.19}$$

Here, the third term on the right hand side corresponds to transport in physical space by the residual velocities. In analogy to the closure of the turbulent transport term in the momentum equation, we adopt an eddy viscosity model to close this term [57, 72],

$$\overline{u_j(\mathbf{x}, t)g} - \overline{u}_j(\mathbf{x}, t)f = -\Gamma(\mathbf{x}, t)\frac{\partial f}{\partial x_j}. \tag{3.20}$$

In Eq. (3.20), $\Gamma(\mathbf{x}, t) = \Gamma'(\mathbf{x}, t)/Sc$ denotes a scaled eddy viscosity which can be related to the eddy viscosity $\Gamma'(\mathbf{x}, t)$ obtained from the standard Smagorinsky model [103] via a turbulent Schmidt/Prandtl number, $Sc = 0.7$. For the turbulent pipe flow analyzed in Section 3.4.1, we specifically employ a dynamic Germano-type procedure for evaluating the Smagorinsky constant [62, 155]. In the context of the turbulent mixing layer which we discuss in Section 3.4.2, by contrast, the Smagorinsky constant remains fixed at 0.1.

Physically, the final two terms in Eq. (3.19) correspond to mixing on a molecular scale. The closure of these terms under the viewpoint of the one-point, one-time *pdf* $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$ is addressed in Section 3.2.4. For generality, we suppose at this point that the micromixing terms can be modelled by expressions of the form

$$f\overline{\left[\sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D(\mathbf{x}, t)\frac{\partial Y_i}{\partial x_j}\right)\middle| \mathbf{y}, n(\cdot)\right]} = \mathcal{M}_i f, \tag{3.21}$$

$$f\overline{\left[\sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D_p(\mathbf{x}, t)\frac{\partial N}{\partial x_j}\right)\middle| \mathbf{y}, n(\cdot)\right]} = \mathcal{M}_p f, \tag{3.22}$$

where $\mathcal{M}_i$, $i = 1, \ldots, n_s$, and $\mathcal{M}_p$ denote operators on $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$ [179]. By introducing Eqs. (3.20) through (3.22) into Eq. (3.19), we obtain the following modelled *pdf* transport equation

$$\begin{aligned}\frac{\partial f}{\partial t}+\sum_{j=1}^{3}\overline{u}_j(\mathbf{x},t)\frac{\partial f}{\partial x_j}=&\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\Gamma(\mathbf{x},t)\frac{\partial f}{\partial x_j}\right)-\sum_{i=1}^{n_s}\frac{\partial}{\partial y_i}\left(f\dot{\omega}_i(\mathbf{y},n(\cdot))+\mathcal{M}_i f\right)\\&-\frac{\partial}{\partial n}\left(f\dot{s}(\cdot,\mathbf{y},n(\cdot))-f\frac{\partial\left(G(\cdot,\mathbf{y})n(\cdot)\right)}{\partial l}+\mathcal{M}_p f\right).\end{aligned}\tag{3.23}$$

To slightly abbreviate the notation, we next define $n_\phi \equiv n_s+1$, $\mathbf{z}\equiv(\mathbf{y}^T,n(\cdot))^T$ and $\mathcal{M}_{n_\phi}\equiv\mathcal{M}_p$ as well as

$$\mathbf{s}(\cdot,\mathbf{z})\equiv\begin{pmatrix}\dot{\boldsymbol{\omega}}(\mathbf{y},n(\cdot))\\ \dot{s}(\cdot,\mathbf{y},n(\cdot))-\frac{\partial(G(\cdot,\mathbf{y})n(\cdot))}{\partial l}\end{pmatrix}\tag{3.24}$$

such that Eq. (3.23) reduces to

$$\frac{\partial f}{\partial t}+\sum_{j=1}^{3}\overline{u}_j(\mathbf{x},t)\frac{\partial f}{\partial x_j}=\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\Gamma(\mathbf{x},t)\frac{\partial f}{\partial x_j}\right)-\sum_{i=1}^{n_\phi}\frac{\partial}{\partial z_i}\left(fs_i(\cdot,\mathbf{z})+\mathcal{M}_i f\right).\tag{3.25}$$

In Eq. (3.25), the sample space function $n(\cdot)$ persists as an independent coordinate. This is at variance with the original PBE-PDF formulation by Rigopoulos [174] who discretized the PBE on a fixed grid in particle property space first and, subsequently, obtained an evolution equation for the joint *pdf* of the reactive scalars and the discrete number densities. However, we found that deferring the discretization in particle property space enables a distinct separation between physical modelling and numerical solution and, in particular, allows for both fixed and adaptive grid discretization schemes to be consistently applied in particle property space.

3.2.4 Mixing on the molecular scale

In this section, we address the closure of the molecular mixing term in the transport equation of the one-point, one-time joint scalar-number density *pdf* $f(\mathbf{z};\mathbf{x},t)$ (Eq. (3.25)),

$$-\sum_{i=1}^{n_\phi}\frac{\partial}{\partial z_i}\left(\mathcal{M}_i f\right),\tag{3.26}$$

where $\mathcal{M}_i f$, $i=1,\dots,n_\phi-1$, and $\mathcal{M}_{n_\phi}f=\mathcal{M}_p f$ are given by Eqs. (3.21) and (3.22), respectively. In an extension of the IEM-related model proposed by McDermott and Pope [119], we adopt the representation

$$\mathcal{M}_i f = f m_i(\mathbf{x},t,\mathbf{z}),\quad i=1,\dots,n_\phi,\tag{3.27}$$

where

$$m_i(\mathbf{x},t,\mathbf{z}) = \kappa(\mathbf{x},t)\left(\overline{Y}_i(\mathbf{x},t) - y_i\right) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D(\mathbf{x},t)\frac{\partial \overline{Y}_i(\mathbf{x},t)}{\partial x_j}\right) \tag{3.28}$$

for $i = 1, \ldots, n_\phi - 1$,

$$m_{n_\phi}(\mathbf{x},t,\mathbf{z}) = \kappa(\mathbf{x},t)\left(\overline{N}(\cdot,\mathbf{x},t) - n(\cdot)\right) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(D_p(\mathbf{x},t)\frac{\partial \overline{N}(\cdot,\mathbf{x},t)}{\partial x_j}\right) \tag{3.29}$$

and $\kappa(\mathbf{x},t)$ denotes the IEM mixing frequency. This frequency is often expressed in terms of the scaled eddy viscosity $\Gamma(\mathbf{x},t)$,

$$\kappa(\mathbf{x},t) = \frac{C_\kappa}{2}\frac{\Gamma(\mathbf{x},t)}{\Delta^2}, \tag{3.30}$$

where $C_\kappa = 2$ represents a micromixing constant and Δ is computed as the cubic root of the local cell volume in a finite volume partitioning of the flow domain [163]. In practice, it is common to replace the current LES-filtered fields $\overline{\mathbf{Y}}(\mathbf{x},t)$ and $\overline{N}(\cdot,\mathbf{x},t)$ in Eq. (3.29) by the LES-filtered fields at the previous point in time as this avoids introducing integrals of $f(\mathbf{z};\mathbf{x},t)$ in the *pdf* transport equation.

The final terms in Eq. (3.29) account for differential diffusion between the fluid and particulate phases; these may also be extended to account for differential diffusion among fluid phase scalars as well as to the case in which the diffusivities D and D_p are specified functions of $\mathbf{Y}(\mathbf{x},t)$ and $N(\cdot,\mathbf{x},t)$ [119]. By invoking an argument similar to the one of McDermott and Pope [119], it can be shown that a *pdf* formulation based on the micromixing model in Eq. (3.29) is consistent with the governing equations (Eqs. (2.1), (3.1) and (3.2)) in the DNS limit.

For the test cases which we investigate in Sections 3.4.1, 3.4.2 and 4.5, the particle diffusivity D_p is set to zero identically. Strictly, by Eq. (3.22), this implies $\mathcal{M}_{n_\phi} f = 0$ which is at variance with the modelled micromixing term $\mathcal{M}_{n_\phi} f = f m_{n_\phi} = \kappa(\mathbf{x},t)$ $\left(\overline{N}(\cdot,\mathbf{x},t) - n(\cdot)\right)$ obtained from Eqs. (3.27) and (3.29). Thus, even in the absence of molecular particle diffusion, the present micromixing model accounts for mixing of number density towards the mean at a rate that is proportional to the local scaled eddy viscosity $\Gamma(\mathbf{x},t)$. One possible approach to resolve this contradiction would be to choose different micromixing frequencies (variants of Eq. (3.30)) for the scalars and number density and to express the individual micromixing frequencies in terms of both $\Gamma(\mathbf{x},t)$ and the respective molecular diffusivity such that the right hand side of Eq. (3.29) consistently reduces to zero in the limit of vanishing molecular diffusion. This extension may also yield differential micromixing frequencies among the gas phase scalars. In view of the scope of this work, we defer such a model enhancement to future times, keeping, however, the limitations of the present micromixing closure in mind.

Introducing Eq. (3.27) into Eq. (3.25) leads to the following *modelled* joint scalar-number density *pdf* transport equation

$$\frac{\partial f}{\partial t} + \sum_{j=1}^{3} \overline{u}_j(\mathbf{x}, t) \frac{\partial f}{\partial x_j} = \sum_{j=1}^{3} \frac{\partial}{\partial x_j} \left(\Gamma(\mathbf{x}, t) \frac{\partial f}{\partial x_j} \right) - \sum_{i=1}^{n_\phi} \frac{\partial}{\partial z_i} \left(f s_i(\cdot, \mathbf{z}) + f m_i(\mathbf{x}, t, \mathbf{z}) \right). \quad (3.31)$$

3.2.5 The statistical significance of the *pdf* in the context of LES

The filtered *pdf* associated with a single realization of the governing fields which we introduced in Section 3.2.3 is sometimes referred to as a filtered density function (*fdf*).[5] Our label of *pdf* reflects that the *fdf* possesses, formally, all properties of a *pdf* and is in keeping with the terminology put forward by Jones [83]. At the same time, this is not to imply that the *pdf* also possesses the meaning of a *pdf* in the physical sense as in RANS, where the considered *pdfs* encompass all statistical properties of the governing random fields.

Instead, in the context of LES, the *pdf* remains a random quantity as it is naturally linked to one realization of the instantaneous governing fields. Concomitantly, also the statistics of the *pdf* are random variables [72], in particular, the filtered composition or particle property distribution. The unclosed terms which remain in the transport equations for the *pdf* and its statistics represent the influence of the residual fluctuations on the filtered quantities for one particular realization of the flow. Since there may be little generality in attempting to provide a closure based on a single realization, the closures which are commonly applied in LES, possess some statistical quality in the sense that they represent the expected influence of the residual scales conditioned on the current realization of the filtered fields [157]. At least in part, this is corroborated by the fact that many closures are adopted from RANS turbulence models. By consequence, as Pitsch [157] argues, we ought to consider instead a *pdf* associated with the governing random fields that is conditioned on one realization of the filtered fields (which are, themselves, random by nature).

These considerations seem to indicate that, thus far, the *pdf* evolution equations were commonly obtained in the sense of an *fdf* and endowed, somehow coincidentally, with some statistical quality by applying closures which actually pertained to a *pdf* of the governing fields that is conditioned on one realization of the filtered fields. If the evolution equation for the conditional *pdf* differed from the usual *pdf* evolution equations, then the present modelling would be, in the least, conceptually inconsistent. However, in response to this seeming disparity, we argue, in line with Pitsch [157], that if the closures which are commonly applied introduce statistical meaning into the *pdf* transport equation, then this meaning is also transferred to the *pdf* as the governing variable. In the same vain, the terms which would be different in a transport equation for a conditional *pdf* are then

[5]Note that the *pdf* associated with a single realization of the governing fields can be viewed as the *pdf* of the random governing fields (that is, in the sense of RANS) conditioned on a particular realization [72].

accounted for by the applied closure schemes, such that, ultimately, the original *pdf* is converted to its conditional counterpart.

It is in this sense, that the term *pdf* which we adopt throughout this work may extend beyond the purely formal meaning mentioned in the beginning of this section. While the *pdf* could be viewed in the strict sense of an *fdf*, reflecting the nature of its definition, it may also hint at a statistical quality in a sense that is particularized by the closures for turbulent transport and micromixing and may shift the physical meaning of the *pdf* to one that is conditioned on a specific realization of the filtered fields.

3.2.6 A stochastic field approach for solving the joint scalar-number density *pdf* transport equation

Owing to the large dimensionality of $f(\mathbf{z};\mathbf{x},t)$, the application of direct discretization schemes to Eq. (3.31) can lead to an enormous computational expense. From a physical viewpoint, however, our primary objective is not to accurately compute $f(\mathbf{z};\mathbf{x},t)$ but rather to estimate expectations with respect to $f(\mathbf{z};\mathbf{x},t)$. Within the combustion community, these observations have motivated the development of different stochastic solution approaches such as the Eulerian method of stochastic particles [158], its Lagrangian counterpart [161] or the method of Eulerian stochastic fields [71, 179, 200]. Here, the main idea is to construct an independent stochastic system whose transition *pdf* evolves according to the *pdf* transport equation in Eq. (3.31). If different realizations of this independent stochastic system are computed, then expectations with respect to $f(\mathbf{z};\mathbf{x},t)$ can be approximated by Monte Carlo estimates. One advantage of this strategy is that the computational effort is naturally channelled towards the estimation of low order statistics of $f(\mathbf{z};\mathbf{x},t)$ such as the LES-filtered reactive scalars $\overline{\mathbf{Y}}(\mathbf{x},t)$ or particle property distribution $\overline{N}(\cdot,\mathbf{x},t)$, while approximation errors are shifted onto higher order statistics.

In this work, we specifically apply the method of Eulerian stochastic fields which was pioneered by Valiño [200] and Hauke and Valiño [71] and later rationalized by Sabel'nikov and Soulard [179]. By design, this method preserves the Eulerian character of the *pdf* transport equation. In view of LES applications, its spatial resolution is independent of the characteristic flow structures which can be technically challenging to achieve with a Lagrangian stochastic particle solver if the flow pattern is complicated [72]. Furthermore, the evolution equations for one realization of the stochastic fields can be reduced quite naturally (with minor modifications) to the deterministic transport equations for $\overline{\mathbf{Y}}(\mathbf{x},t)$ and $\overline{N}(l,\mathbf{x},t)$ associated with the perfect micromixing hypothesis. This may be helpful for software development purposes, in order to verify that the implementation yields physically sensible results, or to assess the influence of finite-rate mixing effects with the same solution software.

In the stochastic field approach, we consider a vector-valued stochastic process $\boldsymbol{\theta}(t;l,\mathbf{x})$ which is smoothly parameterized by $(l,\mathbf{x})$ and constructed such that the transition *pdf*

$h(\mathbf{z}, t|\mathbf{z}_0(\cdot, \mathbf{x}), t_0; \mathbf{x})$ associated with $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$ obeys Eq. (3.31) subject to the initial condition

$$h(\mathbf{z}, t_0|\mathbf{z}_0(\cdot, \mathbf{x}), t_0; \mathbf{x}) = \delta(\mathbf{z} - \mathbf{z}_0(\cdot, \mathbf{x})). \tag{3.32}$$

Since $\mathbf{z}_0(\cdot, \mathbf{x}) = (\mathbf{Y}_0(\mathbf{x}), N_0(\cdot, \mathbf{x}))$ is deterministic and t_0 is given, we drop the conditioning on $\mathbf{z}_0(\cdot, \mathbf{x})$ and t_0 from the notation for brevity. In Appendix B.2.1 it is shown that if the stochastic process $\boldsymbol{\theta}(t; l, \mathbf{x})$ evolves according to the Itô stochastic differential equation (SDE)

$$\begin{aligned}\frac{\partial \theta_i}{\partial t} + \sum_{j=1}^{3} \overline{u}_j(\mathbf{x}, t)\frac{\partial \theta_i}{\partial x_j} = \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(\Gamma(\mathbf{x}, t)\frac{\partial \theta_i}{\partial x_j}\right) - \sum_{j=1}^{3} \sqrt{2\Gamma(\mathbf{x}, t)}\frac{\partial \theta_i}{\partial x_j}\dot{W}_j(t)\\ + s_i(l, \boldsymbol{\theta}) + m_i(\mathbf{x}, t, \boldsymbol{\theta}), \quad i = 1, \ldots, n_\phi,\end{aligned} \tag{3.33}$$

then the transition *pdf* $h(\mathbf{z}, t; \mathbf{x})$ satisfies Eq. (3.31) subject to the initial condition in Eq. (3.32). At this point, we recall that an SDE only holds in a time-integral sense [151, Section 3]; Eq. (3.33) is thus shorthand for

$$\theta_i(t; l, \mathbf{x}) = \theta_i(t_0; l, \mathbf{x}) - \sum_{j=1}^{3} \int_{t_0}^{t} \overline{u}_j \frac{\partial \theta_i}{\partial x_j}\, dt + \ldots - \sum_{j=1}^{3} \int_{t_0}^{t} \sqrt{2\Gamma(\mathbf{x}, t)}\frac{\partial \theta_i}{\partial x_j} dW_j(t) + \ldots, \tag{3.34}$$

where the stochastic integral is computed in Itô's sense with respect to Brownian motion $W_j(t)$.

Apart from conceptual differences between LES and RANS, Eq. (3.33) differs from the joint scalar stochastic field equations presented by Hauke and Valiño [71, Eq. (9)] mainly in two respects. First, our stochastic process $\boldsymbol{\theta}(t; l, \mathbf{x})$ features the particle property l as an additional independent coordinate and, second, it encompasses a component (index n_ϕ) which physically represents the particle number density. Yet, since the stochastic processes $\theta_i(t; l, \mathbf{x})$, $i = 1, \ldots, n_\phi - 1$, corresponding to the fluid phase scalars are independent of l initially and since the corresponding source terms and initial/boundary conditions are independent of l, the scalars θ_i, $i = 1, \ldots, n_\phi - 1$, remain independent of l for all times $t \geq t_0$. This shows that, apart from a different choice of micromixing model, the SDEs for $\theta_i(t; l, \mathbf{x})$, $i = 1, \ldots, n_\phi - 1$, in Eq. (3.33) are consistent with the fluid phase SDE in Reference [71, Eq. (9)]. By contrast, the stochastic particle number density $\theta_{n_\phi}(t; l, \mathbf{x})$ in Eq. (3.33) is distributed in particle property space and may be created and removed, or convected along l at the local growth rate (Eq. (3.24)).

Commonly, evolution equations of the type of Eq. (3.33) are solved numerically using the method of fractional steps. While this introduces an approximation in time, it offers the advantage that the numerical solution procedures can be tailored to the physical characteristics of each fractional step. For a first-order accurate fractional time stepping

[161], the convection-diffusion fractional step encompasses the SDE

$$\frac{\partial\theta_i}{\partial t} + \sum_{j=1}^{3}\left(\overline{u}_j(\mathbf{x},t) + \sqrt{2\Gamma(\mathbf{x},t)}\dot{W}_j(t)\right)\frac{\partial\theta_i}{\partial x_j} = \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\Gamma(\mathbf{x},t)\frac{\partial\theta_i}{\partial x_j}\right), \quad i = 1,\ldots,n_\phi, \tag{3.35}$$

while the micromixing fractional step is based on

$$\frac{\partial\theta_i}{\partial t} = m_i(\mathbf{x},t,\boldsymbol{\theta}), \quad i = 1,\ldots,n_\phi, \tag{3.36}$$

and the fluid reaction and PBE fractional steps are, respectively, given by

$$\frac{\partial\theta_i}{\partial t} = \dot{\omega}_i(\boldsymbol{\theta}), \quad i = 1,\ldots,n_\phi - 1, \tag{3.37}$$

$$\frac{\partial\theta_{n_\phi}}{\partial t} + \frac{\partial\left(G(l,\boldsymbol{\theta})\theta_{n_\phi}\right)}{\partial l} = \dot{s}(l,\boldsymbol{\theta}), \tag{3.38}$$

where, with a slight abuse of notation, $G(l,\cdot)$ is evaluated at $\boldsymbol{\theta}$. In Eqs. (3.37) and (3.38), the source terms $s_i(l,\boldsymbol{\theta})$ have been recast in terms of $\dot{\omega}_i(\boldsymbol{\theta})$ and $\dot{s}(l,\boldsymbol{\theta})$ by introducing Eq. (3.24).

Considering n_f realizations $\boldsymbol{\theta}^{(i)}(t;l,\mathbf{x})$, $i = 1,\ldots,n_f$, of the stochastic process $\boldsymbol{\theta}(t;l,\mathbf{x})$, we approximate expectations with respect to $h(\mathbf{z},t;\mathbf{x})$ by Monte Carlo estimates,

$$\begin{aligned}\overline{H}(\mathbf{Y}(\mathbf{x},t),N(\cdot,\mathbf{x},t)) &= \int H(\mathbf{y},n(\cdot))f(\mathbf{y},n(\cdot);\mathbf{x},t)\,d\mathbf{y}dn(\cdot) = \int H(\mathbf{z})h(\mathbf{z},t;\mathbf{x})\,d\mathbf{z}\\ &\approx \frac{1}{n_f}\sum_{i=1}^{n_f} H(\boldsymbol{\theta}^{(i)}(t;\cdot,\mathbf{x})),\end{aligned} \tag{3.39}$$

where the observable $H(\mathbf{Y}(\mathbf{x},t),N(\cdot,\mathbf{x},t))$ is a function of the fluid phase composition and a functional of the particle property distribution. Formally, we require $H(\mathbf{Y}(\mathbf{x},t),N(\cdot,\mathbf{x},t))$ to be such that the integrals in Eq. (3.39) exist [161]. If H is given by the identity, then Eq. (3.39) yields the LES-filtered values $\overline{\mathbf{Y}}(\mathbf{x},t)$ and $\overline{N}(\cdot,\mathbf{x},t)$. By the central limit theorem, the distribution associated with the Monte Carlo error

$$\epsilon(\cdot,\mathbf{x},t) \equiv \left|\frac{1}{n_f}\sum_{i=1}^{n_f} H(\boldsymbol{\theta}^{(i)}(t;\cdot,\mathbf{x})) - \overline{H}(\mathbf{Y}(\mathbf{x},t),N(\cdot,\mathbf{x},t))\right| \tag{3.40}$$

converges as $n_f \to \infty$ to a normal distribution with mean 0 and standard deviation $\sigma(\cdot,\mathbf{x},t)/\sqrt{n_f}$, where $\sigma(\cdot,\mathbf{x},t)$ denotes the standard deviation of $H(\mathbf{Y}(\mathbf{x},t),N(\cdot,\mathbf{x},t))$ [151, 161].

In practice, we apply the explicit adaptive grid method (EAGM) developed in Chapter 2 to discretize the stochastic number density equation (Eq. (3.33) for $i = n_\phi$) in particle property space. This method is based on a time- and space-dependent coordinate transformation on particle property space which is explicitly advanced in time. Applying the coordinate transformation to Eq. (3.33) yields an SDE that is expressed in terms of so-called transformed stochastic fields. Since the fractional steps corresponding to the transformed stochastic field equations differ, in part, from Eqs. (3.35) through (3.38), we present a brief formal derivation in Appendix B.3.

3.3 Numerical solution scheme

The evolution equations for the stochastic fields (Eqs. (B.27) through (B.30) in Appendix B.3) are solved in conjunction with the momentum and continuity equations for the fluid flow in our in-house research software LES-BOFFIN [87]. Here, the spatial discretization is based on a finite volume scheme on a rectilinear grid. The convective derivatives in the momentum equation are evaluated using an energy-conserving discretization scheme, whereas a TVD scheme is applied to the convective derivatives in the stochastic scalars' convection-diffusion step (Eq. (B.27)). The remaining spatial derivatives are approximated by second order accurate central differences. In order to avoid the velocity-pressure decoupling, the velocity components are stored on a staggered grid. The flow solver, moreover, is based on the iterative SIMPLE scheme [149]. For the discretization in particle property space, we combine the EAGM of Section 2 with the $\kappa = 1/3$ high resolution finite volume scheme of Koren [92], also see Qamar et al. [166, 167].

Both the continuity/momentum equations and the deterministic terms in the convection-diffusion fractional step (Eq. (B.27)) are discretized in time using the second order accurate Crank-Nicolson scheme. On the other hand, for consistency with the Itô interpretation of the stochastic integral, we apply an Euler-Maruyama discretization to the stochastic terms in Eqs. (B.27) and (B.30) [151]. The micromixing fractional step (Eq. (B.28)) is integrated in time using a backward Euler scheme. For the deterministic terms in the PBE fractional step (Eq. (B.30)), we either invoke the 5th order accurate Runge-Kutta scheme Dopri5 [68] (Section 3.4.1) or combine the Crank-Nicolson method with a modified Newton-Raphson non-linear system solver (Section 3.4.2). If no reaction takes place in the fluid phase, then the fluid reaction fractional step (Eq. (B.29)) reduces to species consumption by particle formation and can be solved analytically.

In the stochastic solution scheme for the joint scalar-number density *pdf*, we invoke eight realizations of the stochastic field process [85, 86]. Considering a turbulent diffusion flame, Mustata et al. [133] computed results also using sixteen stochastic fields, but found that the changes in the temporal averages and the temporal root mean square (rms) deviations of the major scalars are very small. Similarly, Jones and Navarro-Martinez [84] modelled a lifted turbulent H_2-air diffusion flame and observed that eight and sixteen stochastic fields yielded very similar results for the time averaged scalars, while the predicted temporal rms results slightly improved by using sixteen stochastic fields. These authors also quantified the finite sampling error (about 11% for eight stochastic fields) for the instantaneous mixture fraction, but argued that this error significantly decreases if temporal statistics are considered.

The LES-filtered values which appear as coefficients in Eq. (3.33), for example, $\overline{\mathbf{Y}}(\mathbf{x}, t)$ and $\overline{N}(\cdot, \mathbf{x}, t)$, are computed from the stochastic fields based on the Monte Carlo estimator in Eq. (3.39). Since, for a finite number of stochastic fields, these estimates are not exact, but rather also influenced by randomness, the numerical solution of the stochastic field equations incurs an additional so-called bias error [72]. Pope [160] showed that this bias error decreases as n_f^{-1} and may thus be dominated by the statistical Monte Carlo error which reduces as $\sqrt{n_f}^{-1}$. For the stochastic field method, however, and, in particular, in view of the small number of fields that are commonly computed in practice, the relative importance of these two sources of error does not yet seem to have been assessed, at least to our awareness.

Contrary to the stochastic field method, numerical solution schemes based on stochastic particles usually invoke many more realizations of the underlying stochastic process, typically on the order of 100 stochastic particles per finite volume cell [189]. In this light, we may hence raise the question as to why such small a number of fields as 8 may be expected to yield sufficiently accurate estimates of statistics of the joint scalar-number density *pdf*. However, in response, recall that the main conceptual difference between solution schemes based on stochastic fields and particles is that, in the latter, the spatial coordinates are conceived as random variables exhibiting certain correlations with the stochastic variables corresponding to the composition and particle number density. Within the scope of the stochastic field method, by contrast, the random variables, now spatially distributed, are directly discretized in physical space, thus accounting for spatial correlations in a deterministic manner. While we cannot, with certainty and in the absence of a convergence analysis, affirm that this difference is powerful enough to reduce the number of stochastic entities so drastically, it seems conceivable that the superior convergence rate of a direct discretization approach over Monte Carlo schemes with respect to the spatial coordinates accounts for much of the difference in computed stochastic entities, at least as far as practical experience suggests.

3.4 Applications

3.4.1 $BaSO_4$ precipitation in a coaxial pipe mixer

As a first test case, we consider the coaxial pipe mixer of Bałdyga and Orciuch [18] for the precipitation of $BaSO_4$ from aqueous solutions of Na_2SO_4 and $BaCl_2$. The pipe mixer is schematically depicted in Figure 3.2 and consists of a long straight pipe with radius $d/2 = 16\,\text{mm}$ which contains a concentric injector tube with inner radius $r_i = 0.9\,\text{mm}$ and outer radius $r_o = 1.25\,\text{mm}$. Measured from the nozzle exit plane to the reactor outlet, the pipe mixer is $125d = 4\,\text{m}$ long [15]. Out of the series of experiments conducted by Bałdyga

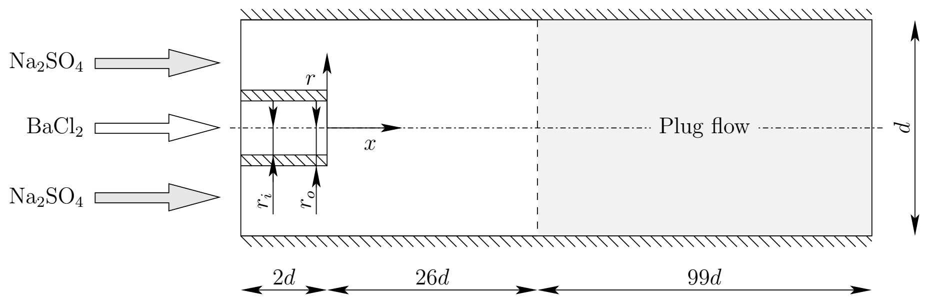


Figure 3.2 Schematic illustration of the coaxial pipe mixer considered by Bałdyga and Orciuch [18]. In the initial part of the pipe mixer (white) the full LES-PBE-PDF approach is applied, while the flow through the remaining part is modelled as a steady-state plug flow (grey).

and Orciuch [18], we specifically investigate the high concentration experiment with unity bulk velocity ratio between the central jet and the co-flow. Here, an aqueous solution with a $BaCl_2$ concentration of $[BaCl_2] = 1.5\,\mathrm{kmol/m^3}$ is introduced through the nozzle, while an aqueous Na_2SO_4 solution with $[Na_2SO_4] = 0.015\,\mathrm{kmol/m^3}$ flows through the main pipe. The co-flow and jet bulk velocities amount to $U = 0.9375\,\mathrm{m/s}$, corresponding to a Reynolds number based on the main pipe diameter of $Re = 3 \times 10^4$. At the outlet of the pipe mixer, Bałdyga and Orciuch [18] extracted samples of the product suspension and determined particle size distributions using a Coulter counter. Additionally, they confirmed by examining $BaSO_4$ crystals microscopically that no aggregation had taken place. This is due to the excess of Ba^{2+} ions which stabilizes the suspension against aggregation.

In the past, the experimental series of Bałdyga and Orciuch [18] has been investigated mainly using RANS-based approaches. Bałdyga and Orciuch [18] combined the method of moments with a *β-pdf* for mixture fraction and an interpolation scheme which related the fluid phase composition and particle phase moments to mixture fraction. By considering the statistically most likely distribution, they additionally reconstructed the product particle size distribution from the moments. The Sauter mean diameter was very well reproduced and the predicted particle size distributions agreed quite well with the measured ones. Later, Di Veroli and Rigopoulos [38] and Di Veroli and Rigopoulos [40] applied the RANS-PBE-PDF method in combination with a finite element discretization of the particle size distribution. In addition to a physical analysis of the precipitation process, they also quantified the impact of composition and composition-number density correlations on the course of the precipitation. Safe for small uncertainties in the $BaSO_4$ kinetics, Di Veroli and Rigopoulos [40] obtained very good predictions for the particle size distribution at the reactor outlet.

In this section, we present a similar comparison based on results obtained from the LES-PBE-PDF method and an adaptive finite volume discretization of the particle size distribution. The computational domain for the LES-PBE-PDF model commences $2d$ upstream of the nozzle exit plane and extends for $26d$ downstream. Subsequently, both the fluid composition and the particle size distribution are largely homogenized across the pipe cross-section and we continue, for computational efficiency, with a steady-state plug flow model. The spatial grid encompasses 360 cells in the axial direction, 64 in the radial direction and 36 cells circumferentially. Additionally, the grid is slightly contracted radially towards the nozzle rim. The global time step size is set to $\Delta t = 10^{-5}\,\mathrm{s}$. For the velocity inflow boundary conditions, we conducted a separate simulation of a developing turbulent flow in the main pipe upstream of the nozzle exit plane and sampled fully turbulent velocity profiles at the outlet of this domain. The EAGM is configured in a similar way as in Section 2.6.2 using $n_p = 30$ finite volume cells and setting the minimum number of nodes in the nucleation interval to 4 and the maximum grid stretching to 2.5.

As in Section 2.6.2, the carrier fluid composition encompasses the mass fractions of H_2O, Ba^{2+}, ${SO_4}^{2-}$, Cl^- and Na^+,

$$\mathbf{Y}(\mathbf{x},t) = \Big(Y_{H_2O}(\mathbf{x},t), Y_{Ba^{2+}}(\mathbf{x},t), Y_{{SO_4}^{2-}}(\mathbf{x},t), Y_{Cl^-}(\mathbf{x},t), Y_{Na^+}(\mathbf{x},t)\Big)^T . \tag{3.41}$$

The characteristic particle size l can be related to the particle volume v according to

$$v = k_v l^3, \tag{3.42}$$

where $k_v = 58$ denotes the experimentally determined particle volume shape factor. The kinetic expressions for the $BaSO_4$ supersaturation $S(\mathbf{Y}(\mathbf{x},t))$, the nucleation rate $R_N(\mathbf{Y}(\mathbf{x},t))$ and the size-independent surface growth rate $G(\mathbf{Y}(\mathbf{x},t))$ are given in Eqs. (2.64) through (2.68). Similar to the setup in Section 2.6.2, the PBE nucleation source term $s(l,\mathbf{Y}(\mathbf{x},t))$ is taken as a symmetric hat function on $[0, 2l_{\mathrm{nuc}}]$ with a maximum value of $R_N(\mathbf{Y}(\mathbf{x},t))/l_{\mathrm{nuc}}$ and a mean nuclei size $l_{\mathrm{nuc}} = 2 \times 10^{-9}\,\mathrm{m}$. The transport and constitutive/kinetic parameters take on the values listed in Table 2.1 except for the molecular particle diffusivity D_p which is set to zero identically.

Figure 3.3 shows instantaneous[6] contour plots of the main kinetic variables controlling the precipitation of $BaSO_4$. While particle nucleation and growth seem to commence simultaneously, the nucleation rate attains its maximum in the mixing layer near the injector, decreasing rapidly as we move downstream. By contrast, $BaSO_4$ growth persists throughout the reactor and slowly decays as supersaturation reduces due to species consumption and turbulent mixing. Both the instantaneous supersaturation and the nucleation/growth rates reflect the strong intermittency of the resolved scalar fields, displaying isolated pockets in which nucleation or growth are dominant.

Figure 3.4, moreover, depicts the time averaged supersaturation both along the centerline of the coaxial pipe mixer and along the radial coordinate at $x = d$, $10d$ and $25d$. Additionally, the shaded areas indicate the temporal rms deviation about the time aver-

[6]For brevity, we omit the term 'LES-filtered' when referring to an LES-filtered instantaneous field or its temporal statistics.

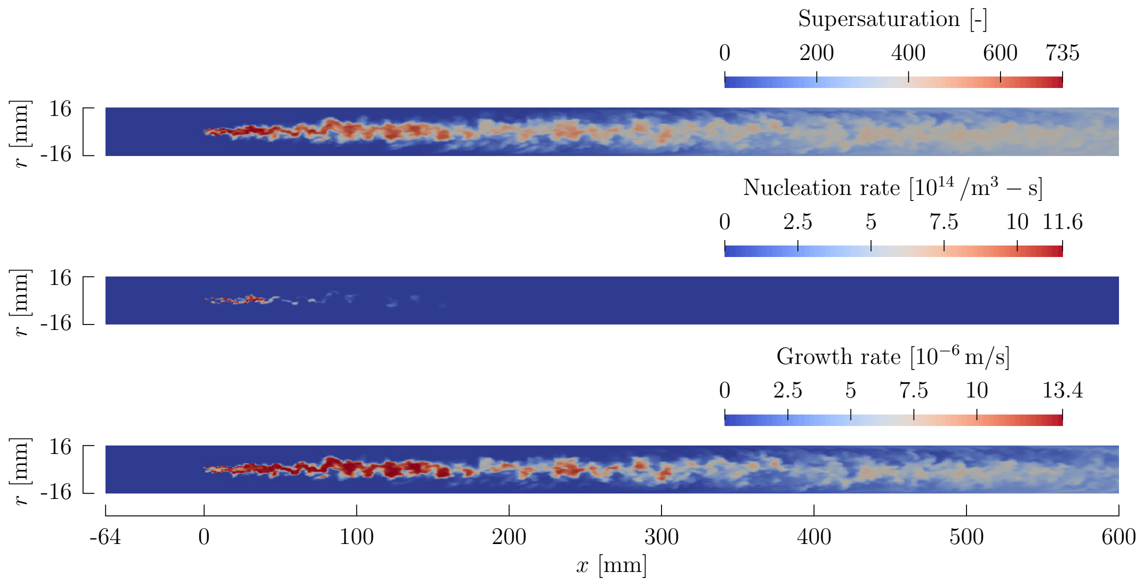
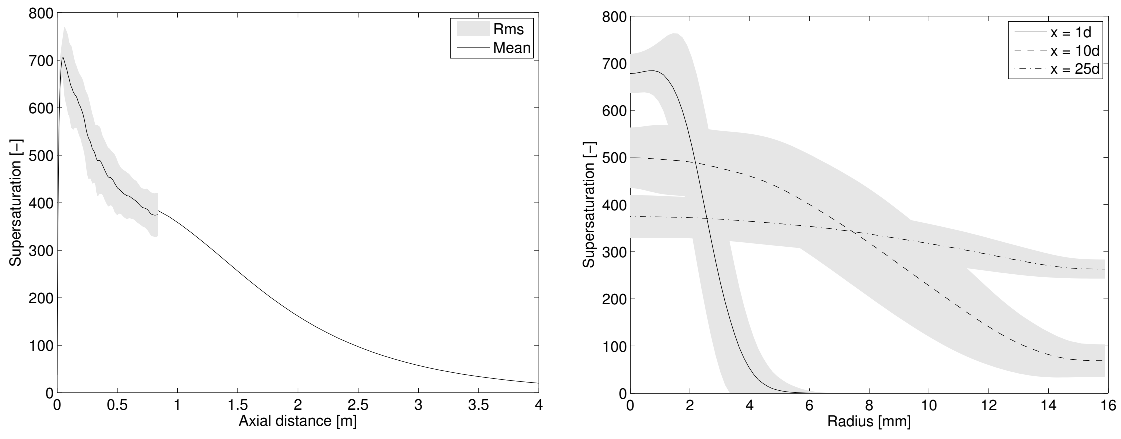


Figure 3.3 Contour plots of the instantaneous fields for $BaSO_4$ supersaturation (top), nucleation rate (center) and growth rate (bottom) in the initial part of the coaxial pipe mixer.

aged values. In the turbulent mixing layer near the nozzle exit plane, very high values of supersaturation are generated. Here, nucleation is dominant and the maximum nucleation rate occurs. Further downstream, supersaturation rapidly decays since the solution is diluted by turbulent mixing with the co-flow and since the fast nucleation and growth of $BaSO_4$ particles deplete Ba^{2+} and ${SO_4}^{2-}$ ions. The particle growth rate attains its maximum value on the centerline at $x = 0.1245\,\mathrm{m}$ and, similar to the supersaturation, slowly decays towards the reactor outlet.

For $x \gtrsim 25d$, the composition is largely homogenized over the cross-sections of the pipe mixer and turbulent mixing becomes much faster than particle nucleation or growth such that $\overline{G}(\mathbf{Y}(\mathbf{x},t)) \approx G(\overline{\mathbf{Y}}(\mathbf{x},t))$ and $\overline{R_N}(\mathbf{Y}(\mathbf{x},t)) \approx R_N(\overline{\mathbf{Y}}(\mathbf{x},t))$. Following Di Veroli and Rigopoulos [38, 40], this indicates that it may be permissible to continue with a steady-state plug flow model. The small gap in the time averaged supersaturation profile at the interface between the LES-PBE-PDF prediction and the steady-state plug flow result in Figure 3.4(a) measures the difference between the time average of the LES-filtered supersaturation computed from the stochastic fields and the supersaturation computed based on the time averaged LES-filtered composition. Since this difference is very small, perfect micromixing may be a valid assumption this far downstream of the reactor. In order to further check the validity of the steady-state plug flow model for $x \gtrsim 25d$, we have varied the axial location from which onwards the steady-state plug flow model is used and found the downstream results to be independent of the exact axial location at which the transition takes place.

Figure 3.5 shows the evolution of the instantaneous and time averaged particle size distribution along the centerline of the reactor. Near the inlet nozzle, many nuclei are



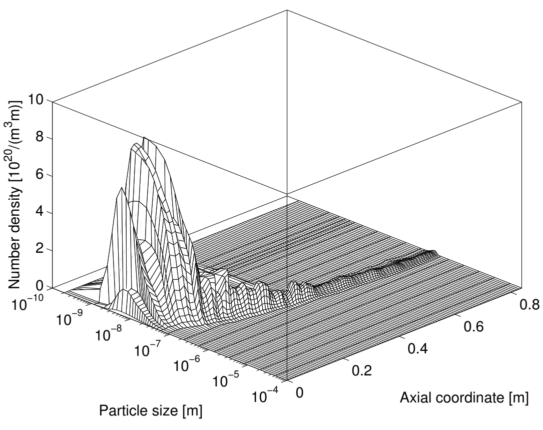
(a) Axial profile (b) Radial profiles

Figure 3.4 Axial and radial profiles of the time averaged supersaturation and its rms in the coaxial pipe mixer.

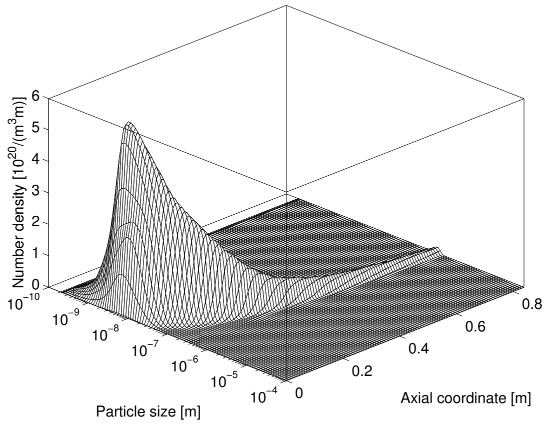
created in a confined region about the centerline. This leads to unimodal particle size distributions situated over the nucleation size range. As we move further downstream, the particle size distributions are convected towards larger particle sizes by the local growth rate. (In Figure 3.5, the particle size distributions seem to become narrower on account of the logarithmic particle size scale.) Simultaneously, turbulent motion distributes number density over the pipe cross-section such that the maximum of the particle size distribution on the centerline decays. By $x \approx 2.75\,\mathrm{m}$, nucleation has subsided and most of the nuclei have grown out of the nucleation size range.

The grid lines in Figure 3.5(a) illustrate the adaptivity of the grid in particle size space. Here, most of the nodes are located in the steep gradient regions, thus accurately resolving the legs of the unimodal particle size distribution. As the particle size distribution moves towards larger particle sizes, the grid nodes readjust and track the steep edges of the particle size distribution. The nodes which seem to remain fixed in particle size space obey the minimum node density conditions of Section 2.5.2 and prevent the grid stretching from exceeding the maximum admissible value r. For the time averages in Figure 3.5(b), by contrast, the instantaneous particle size distributions were interpolated piece-wise linearly onto a fixed exponential reference grid with 100 nodes.

Complementary to Figure 3.5, Figure 3.6 depicts the total particle number and volume densities computed as the zeroth and third moments, respectively, of the time averaged and rms particle size distributions along the centerline. Here, the temporal rms deviations from the time averages follow a similar profile as their corresponding time averages. For instance, the rms about the total particle number density initially increases sharply, peaks at the axial location of maximum average total number density and subsequently decays. The maximum occurs about 0.074 m downstream of the centerline maximum of the average supersaturation and the average nucleation rate. This separation between the two maxima



(a) Instantaneous



(b) Time average

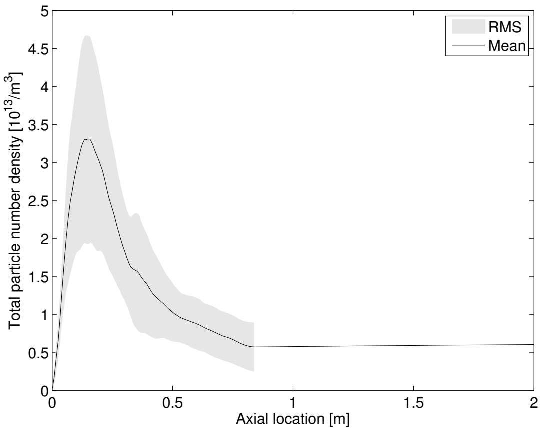
Figure 3.5 Evolution of the instantaneous and time averaged particle size distribution along the centerline of the reactor.

is a result of the competition between accumulation of nuclei along the centerline and the redistribution of number density over a cross-section by turbulent mixing. By contrast, both the time averaged particle volume density and its temporal rms increase along the centerline of the reactor.

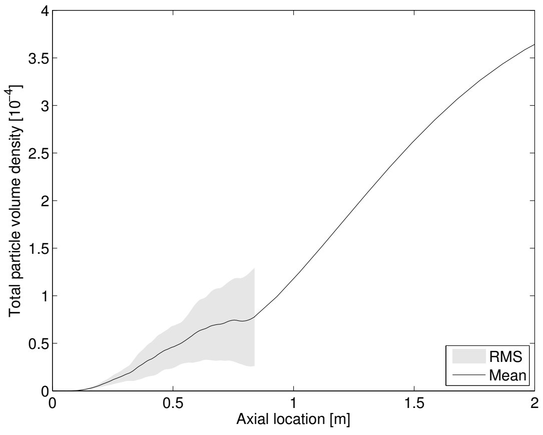
Figure 3.7 depicts the evolution of the reaction yield in terms of the radially integrated $BaSO_4$ volume fraction along the reactor. In the mixing layer region near the nozzle exit plane where nucleation is dominant, the precipitated particle volume remains very small, slowly increasing until $x \approx 0.4\,\text{m}$. At this point, the recently nucleated $BaSO_4$ particles have reached sufficiently large a size for the growth mechanism to contribute noticeably to the precipitated $BaSO_4$ volume. Interestingly, the reduction in supersaturation along the reactor interacts with the size independence of the growth rate in such a way that the time average of the precipitated $BaSO_4$ volume increases linearly in x for $x \gtrsim 0.4\,\text{m}$.

In Figure 3.8, we compare predictions for the time averaged volumetric particle size distribution sampled at the outlet of the reactor with predictions in References [18, 40] and the measured volumetric particle size distribution of Bałdyga and Orciuch [18]. Since the $BaSO_4$ formation kinetics are expressed in terms of the characteristic particle size l, the modelled volumetric distributions are adapted to the experimental size measure d by setting $l = \phi d$, where $\phi = \sqrt[3]{6\pi/k_v}$ denotes the particle sphericity.[7] Figure 3.8 indicates that the volumetric particle size distribution obtained from the LES-PBE-PDF model corresponds well in shape to the measured distribution and the RANS-PBE-PDF prediction of Di Veroli and Rigopoulos [40], but is notably shifted towards larger particle sizes.

[7]The characteristic particle size l represents an average of the particle extent in three orthogonal directions [118], while the particle size d determined by a Coulter counter is proportional to the largest dimension of the particle.



(a) Total particle number density



(b) Total particle volume density

Figure 3.6 Time average and rms of the total particle number and volume densities along the centerline of the coaxial pipe mixer.

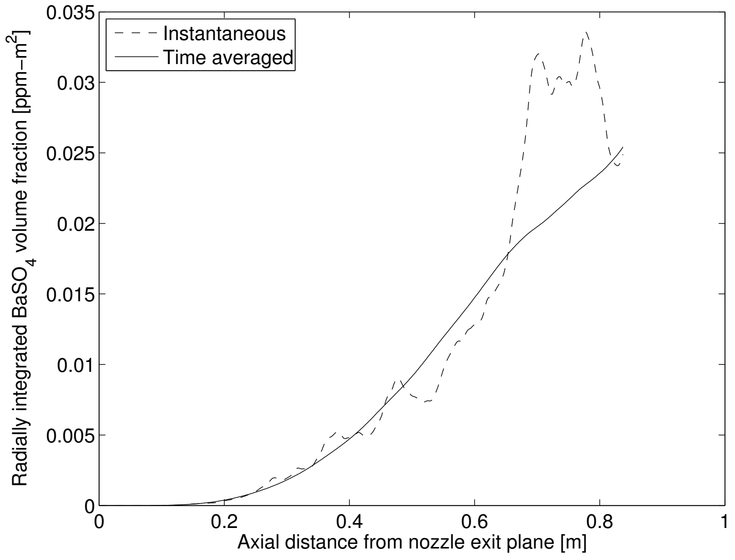


Figure 3.7 Instantaneous and time averaged radially integrated particle volume fraction along the reactor axis.

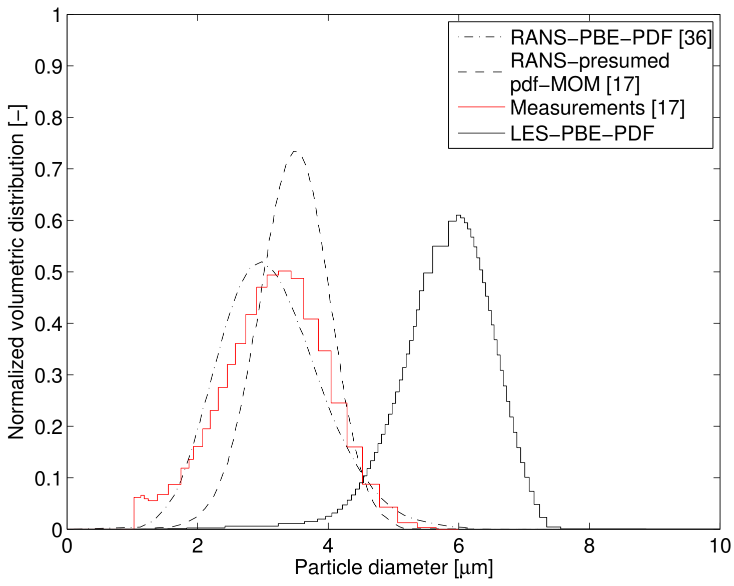


Figure 3.8 Comparing predictions for the mean volumetric particle size distribution at the outlet of the coaxial pipe mixer with measurements.

Quantitatively, this is corroborated by the values for integral properties of the predicted and measured volumetric particle size distributions listed in Table 3.2. Here, the standard deviation of the measured volumetric particle size distribution is accurately reproduced by the LES-PBE-PDF model, while the mean particle size is overpredicted by a factor of approximately 2. Our investigations into this misprediction did not allow for definite conclusions as to its main cause, but suggest an influence of the inflow boundary conditions combined with an insufficient spatial resolution of the mixing layer downstream of the nozzle exit plane. As a consequence, the produced eddy viscosity field may not correspond to fully developed turbulence.

Finally, we present runtime measurements of the numerical solution scheme in LES-BOFFIN for a time step of $\Delta t_k = 10^{-5}\,\mathrm{s}$ on a HP Z820 Workstation (two Intel Xeon CPU E5-2660 v2 processors) in Table 3.3. The runtime values have been averaged over 500 consecutive time steps and are listed separately for each fractional step (Eqs. (3.35) through (3.38)) and for the flow field solution scheme. Table 3.3 indicates that the PBE fractional step (encompassing particle formation, species consumption and PBE-grid adaptation) consumes slightly less time than the scalar convection/diffusion step and the flow solver take jointly. In comparison, the micromixing fractional step adds only little to the computational expense. We may hence conclude that in combination with an adaptive grid discretization along particle size space our comprehensive LES-PBE-PDF model can be solved very efficiently even for an industrial-scale application on a desktop computer system.

		Measurements [18]	RANS-MOM-presumed *pdf* [18]	RANS-PBE-PDF [40]	This work
Sauter mean diameter	[μm]	3.301	3.472	3.14	5.799
Mean of VSD	[μm]	3.176	3.466	3.133	5.798
Standard deviation of VSD	[μm]	0.809	0.552	0.792	0.764
Mean crystal volume	[10^{-17} m^3]	–	1.75	–	7.631

Table 3.2 Comparing different model predictions for the integral properties of the particle size distribution and the volumetric size distribution (VSD, Figure 3.7) with measured values.

3.4.2 Droplet condensation in a turbulent mixing layer

In the present section, we consider the nucleation and growth of dibutyl phtalate (DBP) droplets in the fully developed region of a spatially developing turbulent mixing layer. The flow configuration is the same as the one that was recently investigated by Zhou et al. [213]. In the context of a flow DNS, these authors specifically considered a moment transformation of the PBE and attained closure by invoking the quadrature method of moments (QMOM). Contrary to Zhou et al. [213], we confine the attention to the downstream part of the flow domain in which turbulence is fully developed, retrieving inflow conditions from the DNS database. Although the DNS-QMOM results of Zhou et al. [213] serve as a reference for comparison with our LES-PBE-PDF predictions below, some reservations as to the significance of this comparison remain, in particular, owing to the QMOM approximation in the DNS and the choice of inflow boundary conditions on part of the LES. However, it is emphasized that our primary objective is less to provide a comprehensive validation of the LES-PBE-PDF approach (including the physical closure schemes on micromixing and turbulent transport), but rather to demonstrate the effectiveness of the LES-PBE-PDF method, to indicate its predictive capabilities and to show that our numerical solution scheme based on the method of Eulerian stochastic fields and an explicit adaptive grid discretization in droplet size space is computationally efficient.

In place of an experimental test case, we selected the DNS-QMOM investigation by Zhou et al. [213] as reference mainly since it allowed for the adoption of the same kinetic expressions for the droplet nucleation and growth rates in the LES-PBE-PDF model as Zhou et al. [213] had used for the DNS-QMOM analysis. In this way, any uncertainties regarding the droplet formation kinetics could be eliminated. Considering an experimental configuration of DBP condensation in a turbulent jet [101, 102], Garmory and Mastorakos [59] found predictions obtained from a transported joint scalar-moment *pdf* method to be much more sensitive to the kinetic expression for the DBP droplet surface tension than to the closure for the interaction of turbulence and particle formation. For the same experimental campaign, Di Veroli and Rigopoulos [39] argued that, in the course of the measurements, droplet formation may have continued in the sampling tube, accounting, in part, for the mismatch between measured droplet size distributions and RANS-PBE-PDF predictions. Recently, Neuber et al. [137] concluded that, in the presence of kinetic and

	Average runtime [s]
Flow field	3.34
Convection/diffusion	9.38
Micromixing	1.62
Fluid phase reaction	–
Particle phase reaction	11.94
One time step	30.26

Table 3.3 Average runtime per time step ($\Delta t_k = 10^{-5}$ s) for each fractional step on a HP Z820 Workstation using 20 MPI processes.

experimental uncertainties, a quantitative model validation may instead be based on a series of experiments covering different operating conditions. In this light, our comparison with reference DNS-QMOM results has the advantage that both kinetic and measurement uncertainties are excluded and that, although definitive conclusions regarding the primary source of a discrepancy may be difficult to draw, potential sources of such discrepancies either on part of the LES-PBE-PDF method or the DNS-QMOM are well-defined.

The flow configuration is schematically illustrated in Figure 3.9. Here, a hot nitrogen stream at temperature 410 K that is laden with a DBP mass fraction of 5×10^{-3} (top) mixes with a cold stream of pure nitrogen at temperature 290 K (bottom). The flow domain for the LES-PBE-PDF calculation is cuboidal and extends for $l_1 = 6.43 \times 10^{-2}$ m, $l_2 = 10.29 \times 10^{-2}$ m and $l_3 = 4.30 \times 10^{-2}$ m in the streamwise, crosswise and spanwise directions. The inflow x_1-coordinate, moreover, is offset from the origin by l_1, indicating the streamwise distance over which the mixing layer transitions from a laminar to a fully developed turbulent flow regime. As a result of mixing with the cold bottom stream, DBP supersaturation occurs, driving droplet nucleation and growth. The free stream velocities of the top and bottom streams amount to $U_1 = 15$ m/s and $U_2 = 5$ m/s, respectively, such that the convective velocity $U_c = (U_1 + U_2)/2 = 10$ m/s and the velocity difference across the mixing layer $\Delta U = U_2 - U_1 = 10$ m/s are equal. In an *a posteriori* analysis, Zhou et al. [213] concluded that vapour consumption due to droplet formation only plays a minor role and that, owing to the small values of droplet number density, droplet coagulation is negligible. The kinetic expressions for the DBP nucleation and growth rates are listed in Table 3.4 [213].

As briefly indicated above, the LES-PBE-PDF model is restricted to the fully developed turbulent region of the mixing layer. In this regard, the inflow profiles for velocity, temperature and mass fractions of DBP and N_2 are obtained from the DNS results of Zhou et al. [213] by spatially averaging the DNS profiles sampled at $x_1 = l_1$ every 5×10^{-4} s over the cross-section of each finite volume cell in the inflow plane. From the cell averaged DNS-QMOM predictions of the zeroth, first and third moments of the droplet size

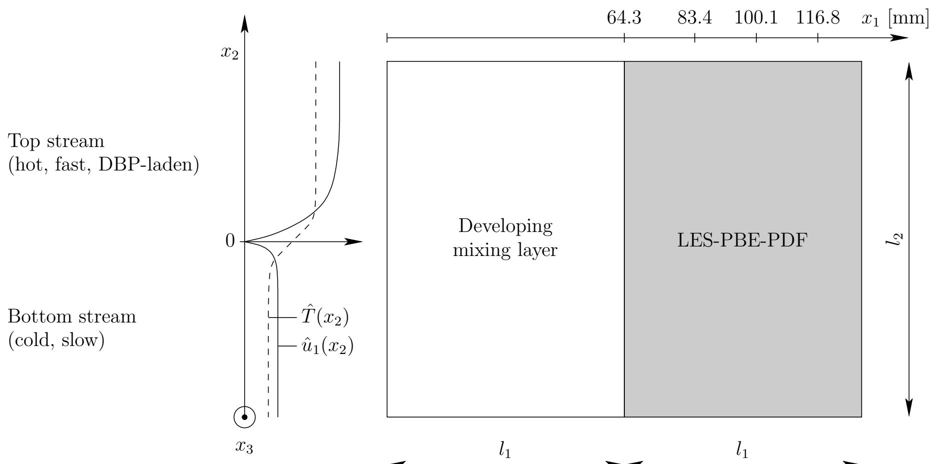


Figure 3.9 Schematic illustration of a spatially developing mixing layer with DBP droplet condensation. Here, $\hat{u}_1(x_2)$ and $\hat{T}(x_2)$ indicate the mean inflow profiles of axial velocity and temperature, respectively, which Zhou et al. [213] applied in the context of a DNS of the present mixing layer. The mean DBP mass fraction inflow profile is similar in shape to $\hat{T}(x_2)$. The LES which we consider in this section is offset by l_1 in the streamwise direction and commences at a cross-section beyond which Zhou et al. [213] observed fully developed turbulence.

distribution, a lognormal distribution [100]

$$\overline{N}(l, \mathbf{x}, t) = \frac{M_0(\mathbf{x}, t)}{3l\sqrt{2\pi}\ln\sigma(\mathbf{x}, t)} \exp\left(-\frac{\ln^2\left(\frac{l}{\lambda(\mathbf{x},t)}\right)}{18\ln^2\sigma(\mathbf{x}, t)}\right) \tag{3.43}$$

is analytically reconstructed for each cell in the inflow cross-section. The coefficients $\lambda(\mathbf{x}, t)$ and $\sigma(\mathbf{x}, t)$ can be computed from the cell-averaged DNS-QMOM results according to

$$\lambda(\mathbf{x}, t) = \frac{M_1^2(\mathbf{x}, t)}{\sqrt{M_0^3(\mathbf{x}, t)\hat{M}_2(\mathbf{x}, t)}}, \tag{3.44}$$

$$\ln^2\sigma(\mathbf{x}, t) = \frac{1}{9}\ln\left(\frac{M_0(\mathbf{x}, t)\hat{M}_2(\mathbf{x}, t)}{M_1^2(\mathbf{x}, t)}\right), \tag{3.45}$$

where the second moment $\hat{M}_2(\mathbf{x}, t)$ is given by

$$\hat{M}_2(\mathbf{x}, t) = M_1(\mathbf{x}, t)\sqrt[3]{\frac{M_3(\mathbf{x}, t)}{M_0(\mathbf{x}, t)}} \tag{3.46}$$

and $M_i(\mathbf{x}, t)$, $i = 0, \ldots, 3$, indicate the cell-averaged DNS-QMOM moments. During the reconstruction, we found the values for the second moments $\hat{M}_2(\mathbf{x}, t)$ computed according to Eq. (3.46) to deviate from the second moments $M_2(\mathbf{x}, t)$ obtained as cell-averages from the DNS-QMOM predictions by 2.8 % on average and by 99.9 % (in a single instance) at most.

At the outlet, the velocity is subjected to a convective outflow boundary condition [87],

Physical quantity	Symbol	Value or kinetic expression	Units
Gas mixture density	ρ	1	$\mathrm{kg/m^3}$
Gas diffusivity	D	1.8×10^{-5}	$\mathrm{m^2/s}$
Particle diffusivity	D_p	0	$\mathrm{m^2/s}$
Mass of a DBP molecule	m_{DBP}	4.62×10^{-25}	kg
Density of liquid DBP	ρ_{DBP}	$1063 - 0.826\,(T - 273.16)$	$\mathrm{kg/m^3}$
Volume of a DBP molecule	v_m	$\frac{m_{\mathrm{DBP}}}{\rho_{\mathrm{DBP}}}$	$\mathrm{m^3}$
Saturation pressure	p_{sat}	$133.1 \exp\left(16.27 - \frac{3836}{T} - \frac{1261126}{T^2}\right)$	$\mathrm{N/m^2}$
Surface tension	σ	$3.53 \times 10^{-2} - 8.63 \times 10^{-5}(T - 273.16)$	N/m
Saturation ratio	$S(\mathbf{Y})$	$\frac{pX_{\mathrm{DBP}}}{p_{\mathrm{sat}}}$	–
Nucleation rate	$R_N(\mathbf{Y})$	$\frac{Sp_{\mathrm{sat}}}{k_B T}\sqrt{\frac{2\sigma}{\pi m_{\mathrm{DBP}}}} \exp\left(-\frac{16}{3}\frac{\pi\sigma^3 v_m^2}{(k_B T)^3 \ln(S)^2}\right)$	$1/\mathrm{m^3} - \mathrm{s}$
Growth rate	$G(l, \mathbf{Y})$	$\left(G_1(\mathbf{Y})^{-1} + G_2(l, \mathbf{Y})^{-1}\right)^{-1}$	m/s
Free molecular regime	$G_1(\mathbf{Y})$	$(S-1)\,p_{\mathrm{sat}}\frac{2v_m}{\sqrt{2\pi m_{\mathrm{DBP}} k_B T}}$	m/s
Continuum regime	$G_2(l, \mathbf{Y})$	$(S-1)\,p_{\mathrm{sat}}\frac{4v_m D}{k_B T l}$	m/s
Consumption	$\dot{\omega}_{\mathrm{DBP}}(\mathbf{Y}, N(\cdot, \mathbf{x}, t))$	$-\frac{\pi}{6}\frac{\rho_{\mathrm{DBP}}}{\rho}\frac{D}{Dt}\int_0^V N(l, \mathbf{x}, t) l^3\, dl$	1/s

Table 3.4 Transport parameters, constitutive properties of liquid DBP and kinetic expressions for the DBP nucleation and growth rates [213]. In the bottom part of the table, X_{DBP} refers to the DBP mole fraction and D/Dt represents the material time derivative.

while the stochastic scalars obey a zero gradient boundary condition. For both the velocity and the stochastic scalars, free slip boundary conditions apply on the lateral domain faces in the cross-wise direction and periodic boundary conditions are imposed on the lateral boundaries in the spanwise direction.

In the LES, we employ a spatial grid of $96 \times 100 \times 64$ finite volume cells which is four times coarser in each coordinate direction than the reference DNS grid. Similar to the DNS grid, the LES grid is uniform in the streamwise and spanwise directions as well as in the cross-stream direction inside the core mixing layer for $|x_2| \leq 2.5 \times 10^{-2}$ m. Further outwards, the grid is slightly expanded by a factor of 1.05 towards the top and bottom boundaries. The fractional time step amounts to 10^{-5} s. Finally, the droplet size domain extends over $[0, L = 10^{-4}\,\mathrm{m}]$ and the mean nuclei size is given by $l_{\mathrm{nuc}} = 2.5 \times 10^{-9}$ m. For the EAGM, the total number of finite volume cells is set to 30 and the maximum grid stretching and the minimum number of nodes in the nucleation interval $[0, 2l_{\mathrm{nuc}}]$ are given by 2.0 and 4, respectively. Temporal statistics were collected over a time interval of 220 ms and, additionally, averaged along the spanwise direction.

Conceptually, the DNS investigation of Zhou et al. [213] differs from our LES-PBE-PDF approach in two ways: On the one hand, all turbulent length and time scales that are characteristic of the instantaneous flow field are resolved. On the other hand, the dispersed phase is described in terms of the first four moments of the droplet size distribution and closure is attained using the QMOM methodology. This second point complicates a comparison between the LES-PBE-PDF and the reference DNS-QMOM results since potential discrepancies may be due to either the turbulent transport model (Smagorinsky submodel) and the model for the interaction between turbulence and droplet formation

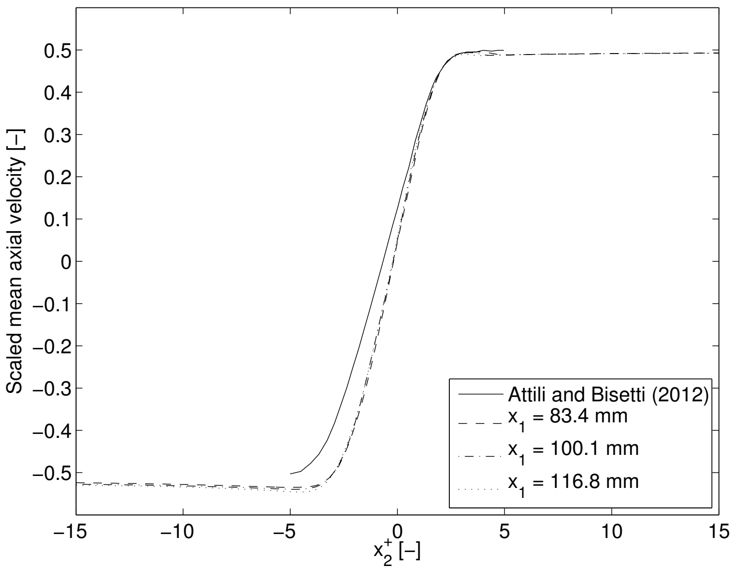


Figure 3.10 Crosswise profiles of the scaled mean axial velocity in the fully developed turbulent region of the mixing layer.

(micromixing closure) on part of the LES-PBE-PDF predictions or the QMOM scheme on part of the DNS-QMOM results. Although definitive conclusions are therefore difficult to draw, our following analyses on the temporal statistics of the flow field, the moments of the droplet size distribution and the nucleation and growth rates admit an indication of the effects of LES-PBE-PDF and DNS-QMOM closures. In order to assess the validity of an assumed shape of the droplet size distribution, we further analyze the change in shape of the size distribution as droplet populations evolve throughout the mixing layer.

Figure 3.10 shows the time average of the scaled axial velocity $(\overline{u}_1(\mathbf{x},t) - U_c)/\Delta U$ over the normalized crosswise coordinate $x_2^+ = x_2/\delta_\theta(x_1)$, where $\delta_\theta(x_1)$ represents the momentum thickness of the mixing layer [10] computed based on the DNS results of Zhou et al. [213]. In the fully developed region of a turbulent mixing layer, the momentum thickness increases linearly along the axial coordinate. For comparison, the solid line in Figure 3.10 indicates the limit mean scaled axial velocity profile reported by Attili and Bisetti [10]. Since the LES-PBE-PDF profiles in Figure 3.10 collapse onto a single curve, this indicates that the LES-flow reproduces well the self-similar nature of the mean axial velocity downstream [162].

In Figures 3.11 and 3.12, the crosswise profiles of the time averaged total droplet number density and volume fraction computed from the LES-PBE-PDF approach are compared with the DNS-QMOM results at two axial measurement stations in the turbulent mixing layer. The shaded areas in the figures indicate the temporal rms deviations about the mean for the LES-PBE-PDF predictions. At the first measurement station, the LES-PBE-PDF and DNS-QMOM profiles of mean total number density agree almost perfectly. Also, the droplet volume fraction profile obtained from the LES-PBE-PDF model reproduces

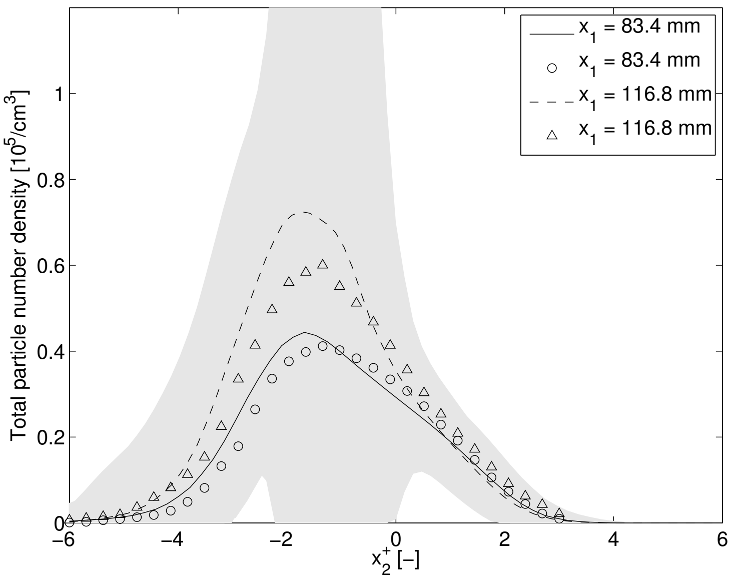


Figure 3.11 Comparing crosswise profiles of the time averaged total droplet number density for the LES-PBE-PDF approach (lines) and the DNS-QMOM reference results (symbols). The shaded areas, moreover, indicate the temporal rms about the mean droplet number density obtained from the LES-PBE-PDF model.

well the DNS-QMOM profile, although the maximum mean volume fraction is slightly underpredicted by about 12 %. Further downstream, at $x_1 = 116.8\,\text{mm}$, the agreement remains very good in terms of shape; however, the maximum value of the mean total number density is overpredicted by approximately 21 %, while the maximum mean droplet volume fraction is underpredicted by about 18 %. In the absence of coagulation, this adverse deviation indicates that nucleation in the LES-PBE-PDF model outweighs the DNS-QMOM nucleation, while the opposite may hold for droplet growth. Furthermore, Figures 3.11 and 3.12 indicate that, on average, the total droplet number density peaks on the cold side of the mixing layer, while the maximum volume fraction is attained closer to the center. Recall that, as the mixing layer progresses downstream, the center of the mixing layer shifts towards the low speed (cold) side linearly in x_1. Since, concomitantly, the momentum thickness in the self-similar region of a developed turbulent mixing layer also grows linearly [10], the shift in cross-wise position of the mixing layer center is absorbed in the normalization of $x_2^+ = x_2/\delta_\theta(x_1)$ such that the center is always located near $x_2^+ \approx -0.5$ [213].

Figures 3.13 and 3.14 additionally depict the crosswise profiles of the time averaged nucleation and growth rates at the same axial distances as in Figures 3.11 and 3.12. Here, the growth rates are computed in the limit of the free molecular regime. Nucleation mainly occurs on the cold side of the mixing layer and decays similarly towards the cold and hot sides. At both measurement stations, the wings of the mean nucleation rate profile are well reproduced by the LES-PBE-PDF predictions, although the maximum mean nucleation rate at $x_2^+ \approx -2$ is overpredicted by up to 66 %. However, similar to the DNS-

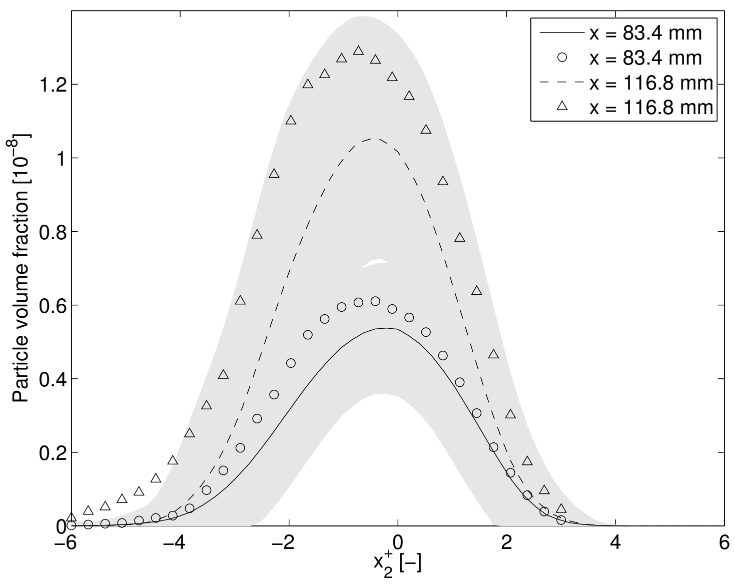


Figure 3.12 Comparing crosswise profiles of the time averaged droplet volume fraction for the LES-PBE-PDF approach (lines) and the DNS-QMOM reference results (symbols). Similar to Figure 3.11, the shaded areas indicate the temporal rms about the mean droplet volume fraction for the LES-PBE-PDF model.

QMOM reference results, the nucleation rate profile appears to be self-similar along the developed turbulent mixing layer. The slightly too vigorous nucleation is commensurate with our previous observation on the downstream deviation of the mean droplet number density. Since the volume fractions produced by droplet nucleation are very low and since supersaturation is not notably reduced by nucleation, the slight underprediction in droplet volume fraction does not seem to be related to the overprediction of nucleation. Indeed, if it were, then we would expect the opposite, an overprediction of droplet volume fraction, at least in the absence of significant vapour scavenging. Based on the DNS-QMOM results, Zhou et al. [213] compared the time averaged nucleation rates with the nucleation rates computed from the time averaged scalar fields and found that the latter exceed the true mean nucleation rate by up to one order of magnitude.

Contrary to the nucleation rate, the mean droplet growth rate peaks close to the center of the mixing layer, albeit on its warm side. Figure 3.14 shows that the cross-wise profiles of the mean growth rate obtained from the LES-PBE-PDF approach agree well with the DNS-QMOM reference results. A minor difference is that, on the side of the hot stream ($x_2^+ > 0$), the LES-PBE-PDF profiles are slightly wider than the DNS-QMOM ones.

Zhou et al. [213] pointed out that droplets nucleate on the cold side of the mixing layer and, subsequently, drift towards the warm side, where droplet growth takes place. Since the mean droplet growth rate in Figure 3.14 is well reproduced, the slight underprediction of mean droplet volume fraction in Figure 3.12 does not seem to be related to a deficiency in droplet growth, at least for droplets in the free molecular regime. Furthermore, as argued above, the deviations of the nucleation rate are also not accountable for the

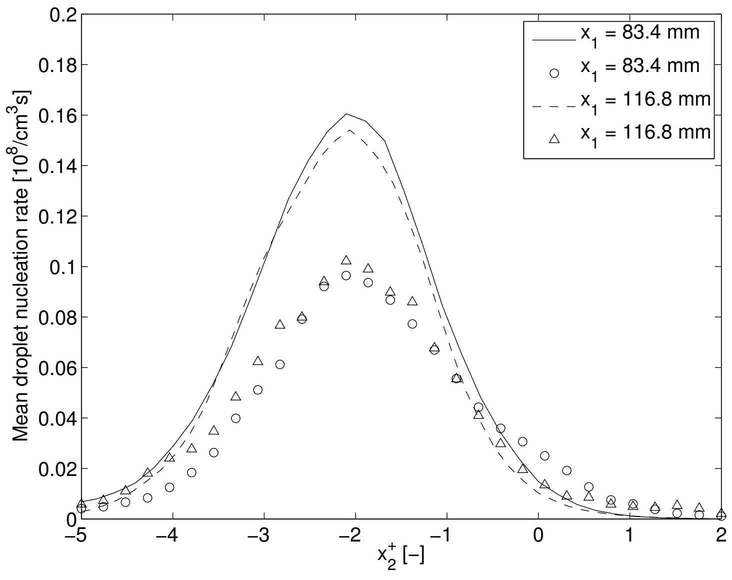


Figure 3.13 Comparing crosswise profiles of the time averaged droplet nucleation rate for the LES-PBE-PDF approach (lines) and the DNS-QMOM reference results (symbols).

deviations in droplet volume fraction. Instead, we suspect that the turbulent and molecular transport mechanisms which are responsible for the movement of recently nucleated droplets from the cold side of the mixing lower towards the warm side are less well developed or not as accurately represented in the current LES-PBE-PDF model. The molecular transport is accounted for by the micromixing model and influenced, to a large extend, by differential diffusion between droplets and fluid phase species [23, 213]. Although our micromixing model encompasses a contribution due to differential fluid-droplet diffusion, both the fluid and droplet phases mix according to the same micromixing rate at present. Turbulent transport, by contrast, is reflected in the eddy viscosity and, possibly owing to our procedure for constructing LES inflow profiles based on the DNS database, the eddy viscosity field in the LES may not be representative of the turbulence levels that persist in the DNS. At the same time, we cannot strictly exclude a contribution of the QMOM closure to the deviations in the maximum mean droplet volume fraction.

Figure 3.15(a) depicts the evolution of the instantaneous droplet size distribution along the center of the mixing layer at $x_2^+ = -0.5$ in the spanwise midplane. Spatially, the droplet size distributions vary largely in shape and magnitude and, thus, evidence the intermittency of droplet formation in a turbulent mixing layer. The grid lines emanating from the droplet size axis in Figure 3.15(a), reflect the adaptation of resolution in droplet size space to the shapes of the local droplet size distributions. In particular, nodes are clustered in the vicinity of the leading growth front, contributing to an accurate resolution of the steep number density gradient.

Complementary to Figure 3.15(a), the droplet size distributions at the inflow and outflow ends of the mixing layer are compared with the lognormal size distributions recon-

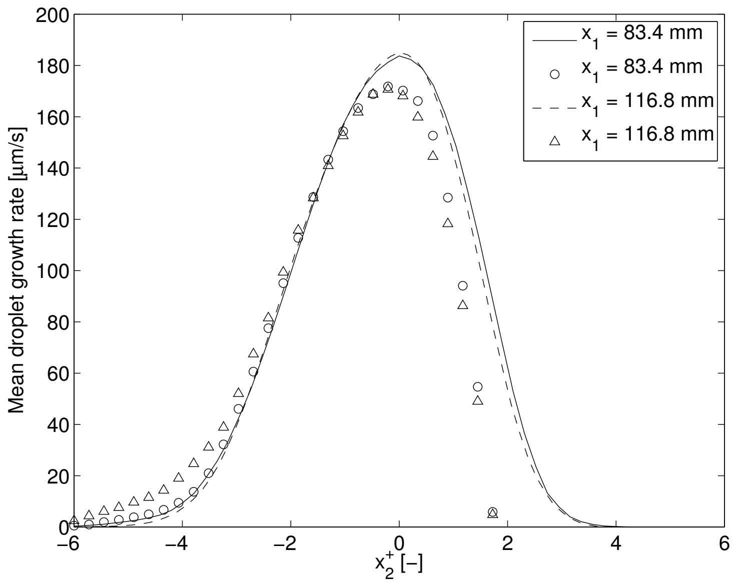
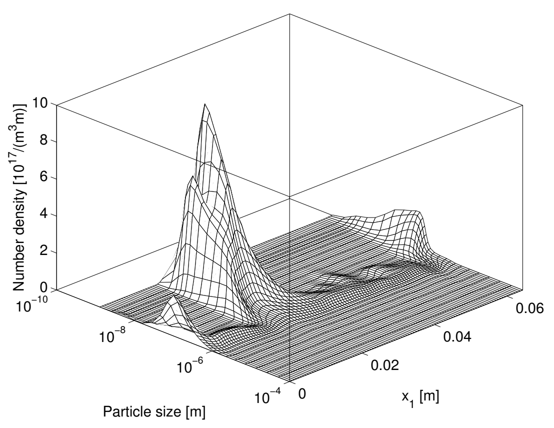


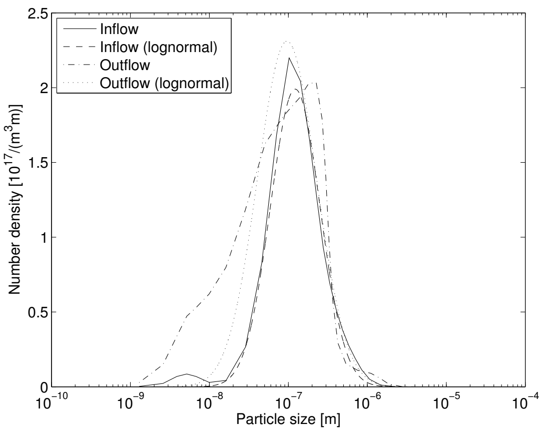
Figure 3.14 Comparing crosswise profiles of the time averaged droplet growth rate for the LES-PBE-PDF approach (lines) and the DNS-QMOM reference results (symbols).

structed from their respective zeroth, first and third moments in Figure 3.15(b). In the inflow cross-section, both droplet size distributions correspond well except for a local maximum in the predicted droplet size distribution which may be caused by a rapid local nucleation event or by transport of nuclei from a neighbouring location. By design, many presumed-shape approaches cannot account for such bimodality in the local droplet size distribution. At the domain outlet, by contrast, the shape of the droplet size distribution deviates quite significantly from a lognormal shape. Here, the current droplet size distribution is characterized by a sharp growth front that yields to a small population of larger droplets in the micrometer size range. Additionally, the maximum value of the droplet number density distribution is smaller than that of the corresponding lognormal distribution and the number density extends smoothly from the sharp front across all intermediate droplet sizes towards the nucleation interval. This indicates that, in a turbulent mixing layer, droplet size distributions may locally encompass droplets which originated from distinct nucleation events and underwent separate growth histories. As a result, the lognormal shape of a droplet size distribution is not conserved along the turbulent mixing layer in general, although some resemblance remains. Based on our observations, some doubt is thus cast on the practice of characterizing experimentally measured droplet size distributions by the parameters of a lognormal droplet size distribution [164].

In order to assess the efficiency of the numerical solution scheme, we examine average runtime measurements obtained on one node (two 12-core Intel Xeon E5-2697 v2 processors) of a Cray XC30 Supercomputer (ARCHER UK) in Table 3.5. The time measurements were averaged over 100 time steps of $\Delta t = 10^{-5}$ s and are listed separately for



(a) Instantaneous droplet size distributions along the center of the mixing layer ($x_2^+ = -0.5$, $x_3 = 0$)



(b) Droplet size distributions at the inflow and outflow ends of the mixing layer center

Figure 3.15 Evolution of the instantaneous DBP droplet size distribution along the center of the mixing layer at $x_2^+ = -0.5$ in the spanwise midplane (Figure (a)). In Figure (b), the instantaneous droplet size distributions at the inflow and outflow ends of the line $x_2^+ = -0.5$, $x_3 = 0$ are compared with lognormal distributions (Eq. (3.43)) reconstructed from the zeroth, first and third moments.

each fractional step. In particular, the computational expense mainly divides between the scalar convection-diffusion fractional step and the PBE fractional step, although the latter takes approximately 2.5 times longer to execute than the convection-diffusion step. Owing to grid adaptation, the PBE fractional step is executed for all grid points and we found that localized nucleation and growth combined with a comparatively large time step entail varying degrees of stiffness in the PBE fractional step across the mixing layer. By consequence, the computational effort is not well balanced across the MPI processes such that the measured runtime of the PBE step may rather be understood as an upper bound. This load imbalance can be remedied with the aid of an apt load balancing scheme which we intend to incorporate in future times. Although the PBE fractional step thus accounts for most of the runtime in Table 3.5, our time measurements indicate that a direct resolution of the droplet size distribution in a turbulent flow based on the PBE-PDF closure for turbulence-droplet formation interaction is feasible and even computationally efficient.

3.5 Chapter summary

Based on the concept of LES, we proposed a comprehensive framework for modelling turbulent reacting flows in which the formation of polydispersed particles takes place. Here, the immersed particulate phase is described in a Eulerian fashion by the PBE governing the spatial and temporal evolution of the particle property distribution. In regard to the turbulence-chemistry and turbulence-particle formation interaction, we augmented the existing joint scalar transported *pdf* approach by the particle number density and obtained a modelled joint scalar-number density *pdf* transport equation. Following common practice,

Fractional steps	Average runtime [s]
Flow field	0.42
Convection/diffusion	6.17
Micromixing	0.67
Fluid phase reaction	–
Particle phase reaction	15.12
One time step	24.11

Table 3.5 Average runtime per time step ($\Delta t = 10^{-5}$ s) for each fractional step on one nodes of a Cray XC30 Supercomputer (ARCHER UK) using 24 MPI processes.

transport by the residual velocities was closed using a gradient diffusion-type model, while micromixing was represented by a recent extension of the IEM model that accounts for differential diffusion effects.

Our physical model combines the LES conservation laws for fluid phase mass and momentum with the modelled joint scalar-number density *pdf* transport equation. In view of a numerical solution, we applied the method of Eulerian stochastic fields. Here, the stochastic field equation governing the particle number density was discretized in particle property space using an explicit adaptive grid scheme. As indicated in Chapter 2, this technique allows for the resolution in particle property space to change commensurate with the shape of the local particle property distribution; in particular, sharp peaks and steep moving fronts which often occur in nucleation-growth problems can be resolved with significantly fewer grid points in particle property space than are required by a fixed grid discretization scheme. For the discretization in physical space and time, on the other hand, standard Eulerian solution schemes were applied.

One advantage of the LES-PBE-PDF model is that it is able to predict the instantaneous LES-filtered composition along with the particle property distribution at any location in the flow domain and any point in time irrespective of the particular functional forms of the fluid and particle phase kinetics. Moreover, since our stochastic solution scheme for the *pdf* transport equation preserves the Eulerian nature of the model, it can be readily included into existing academic or commercial fluid dynamics software.

As an application, we considered the precipitation of $BaSO_4$ particles from an ionic aqueous solution of $BaCl_2$ and Na_2SO_4 in a coaxial pipe mixing device. Analysis of the model results provided insight into the competing influence of nucleation and growth on the shape of the particle size distribution and into the effects of species consumption by particle formation and turbulent dispersion. The predicted mean particle size distribution at the outlet of the reactor corresponded well in shape to the experimentally measured one, but was shifted towards larger particle sizes. While we were not able to ascertain the cause for this discrepancy, it may be related to the choice of inflow boundary conditions and the eddy viscosity field produced in the LES. With regard to computational efficiency, we found the PBE fractional step to consume approximately as much runtime as the convection-diffusion fractional step and the solution of the complete LES-PBE-PDF model to be feasible in reasonable computational times on a desktop computer system.

In a test case, we investigated the nucleation and condensation of aerosol droplets in a fully developed, turbulent mixing layer and compared predictions obtained from the LES-PBE-PDF model with reference DNS-QMOM results. Despite an overprediction of the mean nucleation rate, we registered good quantitative agreement for the temporal statistics of the total droplet number density and volume fraction upstream, while further downstream the number density was slightly overpredicted and the volume fraction underpredicted. Potentially, these deviations are due to limitations of the current micromixing model or may relate to the specific choice of inflow boundary conditions. Analyzing the change in shape of droplet size distributions, we observed that the shape assumed for reconstructing droplet size distributions from DNS-QMOM moments in the inflow cross-section was not preserved in general as the droplet populations underwent nucleation and growth throughout the turbulent mixing layer.

In conclusion, our LES-PBE-PDF model is well-suited for resolving particle property distributions in turbulent reacting flows, while allowing for arbitrary chemical and particle formation kinetics and supporting a computationally efficient numerical solution scheme.

Chapter 4

An LES-PBE-PDF approach for modelling soot formation

4.1 Introduction

Soot particles form in hydrocarbon combustion devices and are typically created in fuel-rich regions of high temperature. From an engineering perspective, soot particles contribute a heat sink through thermal radiation, but may also pollute a combustor if deposited on its walls. On the other hand, soot particles are known to be harmful to both the environment, acting as greenhouse agents, and to the human body as they may cause respiratory diseases [79] or act as carcinogens. By consequence, many of the recent modelling efforts in the combustion community have targeted the prediction of soot formation in hydrocarbon combustion, taking into account the effect of soot on the flame structure.

Typically, soot is considered as a particulate phase that is polydispersed with respect to particle size and behaves non-inertially, that is, the individual soot particles are assumed to be small enough such that they respond instantaneously to changes in the carrier flow field. From a Eulerian perspective, the soot phase can be described by the PBE governing the evolution of the soot particle size distribution throughout the flow domain.

In this chapter, we hence generalize the LES-PBE-PDF approach presented in Chapter 3 to variable density flows with polydispersed particle formation at low Mach numbers. Here, LES is combined with an evolution equation for the LES-filtered one-point, one-time joint probability density function (*pdf*) associated with a single realization of the reactive scalars characterizing the gas phase and the particle number density. Following the argument in Chapter 3, this formulation has the advantage that the physical processes related to chemical reactions, soot particle inception, soot growth and coagulation appear in closed form, while velocity and two-point correlations require modelling. For these, we adopt a standard gradient diffusion hypothesis as well as an IEM-based micromixing model. As in Chapter 3, the joint scalar-number density *pdf* transport equation is solved numerically by the method of Eulerian stochastic fields [71, 179, 200]. Moreover, for the discretization in particle size space of the stochastic field equation associated with the particle size distribution, we apply the explicit adaptive grid technique developed in Chapter 2.

By construction, the PBE-PDF approach achieves a direct resolution of the particle size distribution. This is different from moment-based approaches [168] in which the particle size distribution is described in terms of a small number of low order statistical moments such as the particle number or volume density. Moment-based methods are well-established by now and are not only computationally very economical, but also readily generalize to situations in which the particulate phase is characterized by more than one characteristic property. However, the main challenge associated with these formulations is that the moment equations are closed only for particular functional forms of the particle growth rate and the coagulation kernel. Common closure schemes such as the quadrature method of moments involve an assumption on the shape of the particle size distribution which allows for truncated moments to be expressed in terms of the first few resolved moments.

While moment-based approaches are very common in models for soot formation in turbulent flames (see Section 4.2), the direct resolution of soot particle size distributions has only been considered by few authors. In some approaches, the soot particle size distribution is calculated in a post-processing step [65, 136], omitting the influence of soot formation on the gas phase composition through species consumption and radiation. Recently, Akridis [1] and Akridis and Rigopoulos [2] combined the PBE-PDF concept with RANS [39, 40] in order to model the evolution of the soot particle size distribution in two turbulent diffusion flames.

Further to the developments in these references and in our Chapter 3, the present chapter details three main contributions: First, we develop an LES-PBE-PDF model for predicting the evolution of the soot particle size distribution in a turbulent combusting flow at low Mach number. In particular, this approach allows for the incorporation of arbitrary gas phase and soot kinetics without approximation. Second, we present a stochastic field formulation that reproduces, in a statistical sense, the evolution of the joint scalar-number density *pdf* and can be combined with both fixed and adaptive grid discretization schemes along the particle size coordinate. Finally, the computational viability and predictive capabilities of the combined LES-PBE-PDF approach are demonstrated in the context of the Delft III turbulent diffusion flame. In particular, we show that a detailed resolution of the soot particle size distribution hardly increases the computational cost and that the overall model is computationally feasible on modern computing devices. Furthermore, to our awareness, the present modelling effort constitutes the first attempt to directly predict soot particle size distributions within the scope of LES.

This chapter is organized as follows: In Section 4.2, we first review existing modelling strategies for predicting soot formation in turbulent non-premixed flames. Subsequently, in Section 4.3, the PBE and the LES concept are formally introduced and an evolution equation for the joint scalar-number density *pdf* is obtained. Here, we also discuss the micromixing closure and formulate the stochastic field equations. This is followed by Section 4.4, where the gas phase and soot kinetics and the radiation model are detailed. In Sections 4.5 and 4.5.3, we summarize details on the Delft III flame as well as the computer implementation used in this work. Model predictions are compared with experimental measurements from the Delft III database in Section 4.5.4 and discussed in the light of previous modelling attempts. Finally, we offer conclusions in Section 4.6 and provide a view towards further model enhancements.

4.2 Approaches for modelling soot formation in turbulent non-premixed flames

In the present section, we briefly review existing approaches for modelling soot formation in turbulent non-premixed flames. As an aid to the reader, the references discussed here are classified in Table 4.1.

From a modelling perspective, incorporating the formation of soot into models for turbulent reacting flows has been challenging for several main reasons. On the one hand, soot particles contribute significantly to radiative heat emission and, potentially, reabsorption, thus influencing the distribution of temperature and density in the carrier gas. On the other hand, the synthesis of soot precursors, the inception of soot and soot surface growth (both by PAH condensation and surface reactions) are characterized by much longer time scales than the mixing of reactants [11, 22, 212]. Attili et al. [11] argued that, as a consequence, the soot formation kinetics react only slowly to changes in the local turbulent mixing (scalar dissipation) rate. Furthermore, soot particles are characterized by a very low mass diffusivity and, contrary to light gas phase species, are mainly convected along by the ambient velocity field without significant dispersion [22].

Since RANS-based conserved scalar/presumed *pdf* approaches are computationally very economical, an early idea for accommodating the first challenge mentioned above was to introduce an indicator for radiative heat losses into such a model. Following Gore et al. [63], Young and Moss [212] augmented a steady-state flamelet representation of the gas composition by a heat loss parameter such that the reactive scalars could formally be parameterized by a mixture fraction, the scalar dissipation rate and a heat loss coefficient. As a criterion for determining the heat loss coefficient, Young and Moss [212] proposed the condition that, locally, the Favre-averaged enthalpy computed from the extended flamelet library and the presumed mixture fraction *pdf* coincides with the value obtained by solving a transport equation for the Favre-averaged mixture enthalpy with radiative heat losses. For the description of the soot particulate phase, Young and Moss [212] adopted the semi-empirical model of Moss et al. [126] based on two evolution equations for the Favre-averaged soot number density and volume fraction.

Conceptually, this approach is based on the premise that the carbon content of the gas phase (represented by the conserved mixture fraction) and the carbon represented by the soot-related scalars evolve independently. In heavily sooting flames, however, this may lead to an overprediction of the carbon content near soot pockets and, thus, to increased soot nucleation/growth rates. Furthermore, researchers have pointed out that the slow reaction rates associated with soot precursors entail a delayed response to changes in the scalar dissipation rate such that concentrations of soot precursors cannot be uniquely parameterized by mixture fraction and scalar dissipation rate as in a steady-state flamelet [128, 212]. Despite these concerns, the extended flamelet approach has been very influential in the past decades and motivated both model enhancement as well as application to several turbulent non-premixed flames. Bressloff et al. [26] incorporated the discrete transfer radiation model into the approach of Young and Moss [212] and assessed predictions of soot volume fraction in a confined turbulent methane-air diffusion flame. Bai et al. [13], on the other hand, adopted a one-equation semi-empirical soot model along with the hypothesis of optical thinness in order to investigate soot formation in a turbulent ethylene diffusion flame. Further to previous efforts, these authors considered the joint *pdf* of mixture fraction and scalar dissipation rate and proposed a presumed *pdf* model based on the product of a β-*pdf* for mixture fraction and a lognormal *pdf* for the scalar dissipation rate. The extended flamelet approach combined with a presumed β-*pdf* method for mixture fraction was also adopted by Reddy et al. [172] who included non-gray radiation effects and considered the semi-empirical model by Brookes and Moss [28] for describing the evolution of the instantaneous soot number density and volume fraction. In order to obtain Favre-averages of these two quantities, Reddy et al. [172] computed expectations with respect to an additional presumed *pdf* for temperature.

Formally, the evolution of the soot particle size (or, more generally, soot particle property) distribution through a flame can be described by the PBE which accounts for the physical processes by which soot particles interact both with the ambient gas phase (nucleation, surface growth, oxidation) and among each other (coagulation, aggregation). The semi-empirical one- and two-equation models referred to above are based on a moment reduction of the PBE for a monodisperse particle size distribution and include kinetic rate parameters that were estimated based on measurements in laminar flames [28, 212]. Conceptually, moment transformations of the PBE remain valid independent of the assumption of monodispersity, and for some functional forms of the soot growth/oxidation and coagulation rates, the moment equations are naturally closed at any order. For general kinetics, several approximate closure schemes have been developed [115, 168] and tailored to the characteristics of soot formation [130]. Physically, these closures represent an assumption on the shape of the particle size distribution, but maintain the kinetic details of the PBE and generalize to any number of moments.

Zucca et al. [214] applied the direct quadrature method of moments (DQMOM) and combined chemical equilibrium kinetics with a presumed β-*pdf* method for mixture fraction. Based on a moment reformulation of the bivariate PBE, Mueller and Pitsch [128] recently presented an LES-presumed *pdf* model which resolves the limitations of the original extended flamelet/presumed *pdf* approach, while maintaining its computational efficiency. Specifically, these authors augmented the evolution equations for the gas phase scalars by source terms for radiative heat losses and species/element absorption and accommodated the slow chemistry of soot precursors by adding a separate transport equation for a 'lumped' PAH mass fraction.

Considering the same representation of the gas composition and soot as Mueller and Pitsch [128], Donde et al. [42] explored a transported *pdf* approach based on the IEM micromixing model to resolve the turbulence-chemistry/particle formation interaction as compared to a presumed *pdf* closure. In a subsequent investigation, Xuan and Blanquart [210] replaced the lumped PAH evolution equation in the original LES-presumed *pdf* model of Mueller and Pitsch [128] by evolution equations for benzene and naphtalene and proposed a relaxation model to close the associated source terms under the LES viewpoint.

In transported *pdf* methods, the scalar source terms appear naturally closed, while turbulent transport and molecular mixing of the gas phase and soot scalars require phenomenological closures [83]. In the context of soot formation in turbulent flames, transported *pdf* methods seem to have first been introduced by Metternich et al. [124] who formulated a joint scalar-soot volume fraction *pdf* transport equation based on a constrained equilibrium model of the thermochemistry and a semi-empirical model for the evolution of soot volume fraction. Subsequently, this approach was taken further by Lindstedt and Louloudi [106] who developed a kinetically detailed model for soot formation based on a moment-reformulation of the PBE and solved a transport equation for the joint scalar-moment *pdf*.

Intermediate in computational expense between presumed and transported *pdf* methods for the turbulence-chemistry/soot formation interaction are multi-environment presumed *pdf* methods [50, 51]. Here, the joint scalar *pdf* is represented by a linear combination of the instantaneous *pdfs* associated with several model flow realizations which exchange likelihood/volume fraction according to a particular micromixing prescription. In the so-called DQMOM-IEM approach, this micromixing model is determined such that the evolution and interaction of the model flow representations preserve the first unmixed moments of the joint scalar *pdf*. In the context of RANS, Reddy and De [171] combined the DQMOM-IEM approach for modelling the interaction of turbulence and gas phase chemistry with the semi-empirical soot model of Brookes and Moss [28] and analyzed the influence of several radiation models on predictions of soot volume fraction in two turbulent diffusion flames. Following their previous investigations [170, 172], these authors introduced a separate presumed *pdf* for temperature and evaluated expectations of the temperature-dependent soot source terms with respect to this *pdf*.

In turbulent flames which do not experience extinction or reignition, the conditional moment closure approach (CMC) can be a computationally economical alternative to transported *pdf* methods. In the context of soot formation, Kronenburg et al. [93] seemed to have been the first to incorporate a semi-empirical soot model into a RANS-CMC approach. Commensurate with the third challenge mentioned above, these authors demonstrated the importance of accounting for differential diffusion between the gas- and soot-describing scalars. Combining the same soot model with an LES-CMC approach, Navarro-Martinez and Rigopoulos [135] similarly concluded that differential diffusion between the gas phase scalars and soot leads to more intermittent and locally intense soot volume fraction predictions.

Contrary to moment-based reductions of the PBE, only few researchers have attempted to resolve the soot particle size distribution in turbulent flames. An impending challenge, in this regard, is that mature primary soot particles tend to assemble in the form of chain-like aggregates. In order to describe these, the particle size coordinate needs to be complemented by an additional particle property such as a fractal dimension or the number of primary particles per aggregate. While a univariate description in terms of particle size is well-suited for tracing the evolution of primary soot particles or of aggregates with a fixed fractal shape, it is limited to shape-preserving particle formation and interaction processes.

In an *a posteriori* approach, soot particle size distributions are sometimes estimated by solving the PBE along specific paths through the flame, taking into account mean field information from a calculation without soot formation or with a moment-based soot model. Grosschmidt et al. [65], for instance, computed solutions to the PBE along streamlines of a reacting flow field obtained from a flamelet/presumed *pdf* model of the gas phase combustion. A slightly different approach was followed by Netzell et al. [136] who reformulated the PBE in mixture fraction space and solved a series of unsteady flamelet problems for scalar dissipation rates sampled from the flame field predictions of Bai et al. [13].

The PBE-PDF concept which we adopt here achieves a direct resolution of the particle size distribution within the scope of a transported *pdf* closure for the turbulence-chemistry and particle formation interaction [174]. In the context of RANS, this approach was recently applied to soot formation by Akridis and Rigopoulos [2] and Akridis [1] who investigated turbulent, non-premixed flames at atmospheric and elevated pressure. In the present chapter, the PBE-PDF paradigm is incorporated into an LES framework for combusting variable density flows at low Mach number. Our developments are guided by the objective to devise an efficient numerical solution scheme based on the stochastic field method and the explicit adaptive grid approach proposed in Chapter 2.

Representation of the number density distribution	Turbulence-chemistry/particle formation interaction				
		Presumed *pdf*			
	Perfect micromixing	β-/δ-/log-normal-*pdf*	DQMOM-IEM	CMC	Transported *pdf*
Semi-empirical		[13, 26, 170, 172, 212]	[171]	[93] [135]*	[124]
XMOMY		[214] [91, 128, 210]*			[106] [42]*
DPB	[65, 136]				
(Adaptive) FVM					[1, 2] This work*

Table 4.1 Overview of different approaches for modelling soot formation in turbulent non-premixed flames (DPB: Discretized Population Balance, FVM: Finite Volume Method, XMOMY: Quadrature based Method of Moments, e.g., MOMIC, DQMOM, HMOM). The references marked by a superscript * indicate LES-based modelling approaches, while unmarked references adopted the RANS turbulence model. (In the classification, we associate approaches in which the soot size distribution was computed in a post-processing step based on mean field information with the 'perfect micromixing' assumption.)

4.3 Physical model

4.3.1 Governing equations

In this section, we briefly review the conservation laws that are relevant to the continuum mechanical description of a fluid flow with an immersed particulate phase. In view of a Eulerian formulation, we consider the flow through a domain Ω and introduce the instantaneous velocity field $\mathbf{u}(\mathbf{x}, t)$, the pressure field $p(\mathbf{x}, t)$ and the fluid density $\rho(\mathbf{x}, t)$, where $\mathbf{x}$ represents a location in Ω and $t \geq t_0$ indicates time. In Cartesian coordinates, the continuity and momentum balance laws can be written in the following form

$$\frac{\partial \rho}{\partial t} + \sum_{j=1}^{3} \frac{\partial \rho u_j}{\partial x_j} = 0, \tag{4.1}$$

$$\frac{\partial \rho u_i}{\partial t} + \sum_{j=1}^{3} \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \sum_{j=1}^{3} \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i, \quad i = 1, \ldots, 3, \tag{4.2}$$

where τ_{ij} denotes the viscous stress tensor of a Newtonian fluid and $\mathbf{g}$ is the gravitational acceleration.

Commonly, the carrier fluid is described in terms of reactive scalars $\mathbf{Y}(\mathbf{x}, t)$ which evolve according to

$$\frac{\partial \rho Y_i}{\partial t} + \sum_{j=1}^{3} \frac{\partial \rho u_j Y_i}{\partial x_j} = -\sum_{j=1}^{3} \frac{\partial J_{ij}}{\partial x_j} + \rho \dot{\omega}_i(\mathbf{Y}, N), \quad i = 1, \ldots, n_s, \tag{4.3}$$

where $J_{ij}(\mathbf{x}, t)$ denotes the diffusive flux of scalar i along the jth coordinate direction,

$$J_{ij}(\mathbf{x}, t) = -\rho(\mathbf{x}, t) D(\mathbf{Y}(\mathbf{x}, t)) \frac{\partial Y_i(\mathbf{x}, t)}{\partial x_j}, \tag{4.4}$$

$D(\mathbf{Y}(\mathbf{x},t))$ is the kinematic diffusivity of any scalar into the mixture and $\dot{\omega}_i(\mathbf{Y}(\mathbf{x},t), N(\cdot,\mathbf{x},t))$ represent the scalar production/destruction terms. The diffusivity $D(\mathbf{Y}(\mathbf{x},t))$ can be related to the kinematic viscosity $\nu(\mathbf{Y}(\mathbf{x},t))$ via a molecular Schmidt/Prandtl number $Sc = 0.7$, $D(\mathbf{Y}(\mathbf{x},t)) = \nu(\mathbf{Y}(\mathbf{x},t))/Sc$ [84, 85]. Frequently, the reactive scalars are taken as species mass fractions complemented by a calorific variable such as enthalpy. In the argument list of $\dot{\omega}_i$, the dependency on $N(\cdot,\mathbf{x},t)$ indicates that the scalar source terms may be functionals of the particle size distribution (see below), for instance, owing to species consumption or release on account of particle formation.

In combusting flows at low Mach numbers, the mixture density $\rho(\mathbf{x},t)$ is often computed in terms of the reactive scalars $\mathbf{Y}(\mathbf{x},t)$,

$$\rho(\mathbf{x},t) = \hat{\rho}(\mathbf{Y}(\mathbf{x},t)). \tag{4.5}$$

Physically, this implies that the impact of local pressure deviations from the nominal ambient pressure on the density is comparably small. With regard to its constitutive relations, moreover, the carrier fluid is considered as a multicomponent ideal gas.

Adopting the viewpoint of Sections 2.3 and 3.2.1, soot can be described in terms of the particle number density $N(l,\mathbf{x},t)$, where $l \in [0,\infty)$ indicates a measure of particle size. Physically the particle number density is governed by the PBE in Eq. (2.1). With regard to the conclusions in References [11, 93, 135], we set the kinematic diffusivity of soot that appears in the PBE to zero, $D_p(\mathbf{x},t) \equiv 0$.

Similar to the developments in Section 3.2.1, the feed-back of particle formation on the continuity and momentum equations of the carrier fluid phase is neglected. On the one hand, the applications we consider feature rather low particle mass fractions ($\lesssim 10^{-5}\,\mathrm{kg/m^3}$ for soot formation in the Delft III flame) such that the cumulative density $\rho(\mathbf{x},t)$ of the carrier fluid is not notably affected by mass exchange with a particulate phase [173]. On the other hand, since the particles are very small, the exchange of momentum between the fluid and particulate phases can be well approximated as taking place on an instantaneous basis [173].

Frequently, the release or depletion of gas phase species briefly mentioned above is quantified with the aid of the moments of the particle size distribution $N(\cdot,\mathbf{x},t)$,

$$M_k(N(\cdot,\mathbf{x},t)) = \int_0^\infty l^k N(l,\mathbf{x},t)\,dl. \tag{4.6}$$

In view of subsequent developments, we restrict the semi-infinite particle size space $[0,\infty)$ to the finite domain $[l_l, L]$, where l_l and L represent, respectively, the minimum and maximum attainable particle diameters.

In variable density flows, it is advantageous to switch to a mass-based definition of the number density [11] and to consider the number of particles $N_\rho(l,\mathbf{x},t)$ per unit of mixture mass and unit of length in particle size space,

$$N_\rho(l,\mathbf{x},t) \equiv \frac{N(l,\mathbf{x},t)}{\rho(\mathbf{x},t)}. \tag{4.7}$$

For future reference, we record the initial conditions on the fluid composition and the mass-based particle number density,

$$\mathbf{Y}(\mathbf{x},t_0) = \mathbf{Y}_0(\mathbf{x}), \tag{4.8}$$

$$N_\rho(l,\mathbf{x},t_0) = N_{\rho,0}(l,\mathbf{x}). \tag{4.9}$$

4.3.2 Large eddy simulation

Frequently, the LES-operator is introduced as a spatial filter acting on the governing fields, see, for instance, Section 3.2.2. Since, for our developments, the precise definition (or construction) of such a filter is immaterial, we first recall that the LES-operator, $\bar{\cdot}$, implements an expectation operation and can thus be formulated in the following way [159]

$$\begin{aligned}&\overline{\Phi}(\mathbf{u}(\mathbf{x},t),p(\mathbf{x},t),\mathbf{Y}(\mathbf{x},t),N_\rho(\cdot,\mathbf{x},t))\\ &= \int \Phi(\mathbf{v},q,\mathbf{y},n(\cdot)) f_{\mathbf{u},p,\mathbf{Y},N_\rho}(\mathbf{v},q,\mathbf{y},n(\cdot);\mathbf{x},t)\, d\mathbf{v}dqd\mathbf{y}dn(\cdot),\end{aligned} \tag{4.10}$$

where $\Phi(\mathbf{u}(\mathbf{x},t),p(\mathbf{x},t),\mathbf{Y}(\mathbf{x},t),N_\rho(\cdot,\mathbf{x},t))$ denotes a general observable that appears, formally, as a function of the local velocity, pressure and fluid composition and as a functional of the particle size distribution $N_\rho(\cdot,\mathbf{x},t)$. $f_{\mathbf{u},p,\mathbf{Y},N_\rho}(\mathbf{v},q,\mathbf{y},n(\cdot);\mathbf{x},t)$ is termed the LES-filtered *pdf* associated with a single realization of the governing fields and may be related to a filter kernel $G(\mathbf{x},\mathbf{x}')$ according to

$$\begin{aligned}f_{\mathbf{u},p,\mathbf{Y},N_\rho}(\mathbf{v},q,\mathbf{y},n(\cdot);\mathbf{x},t) &= \overline{\delta\left(\mathbf{v}-\mathbf{u}(\mathbf{x},t)\right)\delta\left(q-p(\mathbf{x},t)\right)}\\ &\quad \overline{\times\delta\left(\mathbf{y}-\mathbf{Y}(\mathbf{x},t)\right)\delta\left(n(\cdot)-N_\rho(\cdot,\mathbf{x},t)\right)}\\ &= \int_\Omega G(\mathbf{x},\mathbf{x}')\delta\left(\mathbf{v}-\mathbf{u}(\mathbf{x}',t)\right)\delta\left(q-p(\mathbf{x}',t)\right)\\ &\quad\times\delta\left(\mathbf{y}-\mathbf{Y}(\mathbf{x}',t)\right)\delta\left(n(\cdot)-N_\rho(\cdot,\mathbf{x}',t)\right)\,d\mathbf{x}',\end{aligned} \tag{4.11}$$

where $\delta(\cdot)$ indicates the Dirac delta distribution and $G(\mathbf{x},\cdot) \geq 0$ integrates to unity. In Eqs. (4.10) and (4.11), $\mathbf{v}$, q and $\mathbf{y}$ represent the sample space variables related to $\mathbf{u}(\mathbf{x},t)$, $p(\mathbf{x},t)$ and $\mathbf{Y}(\mathbf{x},t)$, respectively, while $n(\cdot)$ indicates the sample space function associated with the particle size distribution $N_\rho(\cdot,\mathbf{x},t)$ at $(\mathbf{x},t)$. Specifically, $n(\cdot)$ indicates a particular function drawn from the space of admissible particle size distributions. For notational clarity and to distinguish the nature of $n(\cdot)$ from that of the remaining sample space variables, we maintain the parenthesis-notation $n(\cdot)$ in the following developments. The integral in Eq. (4.10) is written over the combined sample space $(\mathbf{v},q,\mathbf{y},n(\cdot))$.

Conceptually, the marginal number density *pdf* $f_{N_\rho}(n(\cdot);\mathbf{x},t)$ may be obtained as the limit functional

$$f_{N_\rho}(n(\cdot);\mathbf{x},t) = \overline{\lim_{m\to\infty}\prod_{i=0}^{m}\delta\left(n(l_i) - N_\rho(l_i,\mathbf{x},t)\right)}, \tag{4.12}$$

where $l_i = l_l + i(L-l_l)/m$, $i = 0,\dots,m$, represents an auxiliary grid in particle size space and $n(l_i)$ are auxiliary sample space variables associated with $N_\rho(l_i,\mathbf{x},t)$. Eq. (4.12) emphasizes that f_{N_ρ} can be viewed as a multi-size, albeit one-point, one-time *pdf* [174].

For subsequent developments, we record the following commutation property of the LES-operator

$$\overline{\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t}} = \frac{\partial \overline{\mathbf{u}}(\mathbf{x},t)}{\partial t}, \tag{4.13}$$

$$\overline{\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial x_i}} = \frac{\partial \overline{\mathbf{u}}(\mathbf{x},t)}{\partial x_i}, \quad i = 1,\dots,3, \tag{4.14}$$

and similarly for the remaining governing fields $p(\mathbf{x},t)$, $\mathbf{Y}(\mathbf{x},t)$ and $N_\rho(l,\mathbf{x},t)$.

Based on Eqs. (4.5) and (4.10), the LES-filtered density field is given by

$$\overline{\rho}(\mathbf{x},t) = \overline{\hat{\rho}(\mathbf{Y}(\mathbf{x},t))} = \int \hat{\rho}(\mathbf{y}) f_{\mathbf{Y}}(\mathbf{y};\mathbf{x},t)\,d\mathbf{y}, \tag{4.15}$$

where $f_{\mathbf{Y}}(\mathbf{y};\mathbf{x},t)$ represents the marginal filtered *pdf* associated with a single realization of the reactive scalar fields. With the aid of Eq. (4.15), we further introduce the density weighted Favre-filter

$$\begin{aligned}&\tilde{\Phi}(\mathbf{u}(\mathbf{x},t), p(\mathbf{x},t), \mathbf{Y}(\mathbf{x},t), N_\rho(\cdot,\mathbf{x},t)) \\ &= \frac{\overline{\rho(\mathbf{x},t)\Phi(\mathbf{u}(\mathbf{x},t), p(\mathbf{x},t), \mathbf{Y}(\mathbf{x},t), N_\rho(\cdot,\mathbf{x},t))}}{\overline{\rho}(\mathbf{x},t)}.\end{aligned} \tag{4.16}$$

Applying the LES-operator $\bar{\cdot}$ to the continuity and momentum equations (Eqs. (3.2) and (4.2)) and taking into account Eq. (4.16) leads to the following governing LES equations [86]

$$\frac{\partial \overline{\rho}}{\partial t} + \sum_{j=1}^{3}\frac{\partial \overline{\rho}\tilde{u}_j}{\partial x_j} = 0, \tag{4.17}$$

$$\frac{\partial \overline{\rho}\tilde{u}_i}{\partial t} + \sum_{j=1}^{3}\frac{\partial \overline{\rho}\tilde{u}_i\tilde{u}_j}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \sum_{j=1}^{3}\frac{\partial(\tilde{\tau}_{ij} - \tau^*_{ij})}{\partial x_j} + \overline{\rho}g_i, \quad i = 1,\dots,3, \tag{4.18}$$

where $\tilde{\tau}_{ij}$ denotes the viscous stress tensor associated with the Favre-filtered velocity field $\tilde{\mathbf{u}}(\mathbf{x},t)$ and $\tau^*_{ij} = \overline{\rho}(\widetilde{u_iu_j} - \tilde{u}_i\tilde{u}_j)$ is the residual stress tensor. Following common practice, we adopt the standard Smagorinsky model for the deviatoric component of τ^*_{ij} [85], while its spherical component is absorbed into the LES-filtered pressure $\overline{p}(\mathbf{x},t)$.

4.3.3 The joint scalar-number density *pdf*

As a basis for modelling the evolution of the gas phase composition and the particle size distribution in the context of LES, we obtain, in this section, a transport equation for the joint scalar-number density *pdf*,[8]

$$f(\mathbf{y}, n(\cdot); \mathbf{x}, t) = \overline{g(\mathbf{y}, n(\cdot); \mathbf{x}, t)} = \overline{\delta\left(\mathbf{y} - \mathbf{Y}(\mathbf{x}, t)\right) \delta\left(n(\cdot) - N_\rho(\cdot, \mathbf{x}, t)\right)}, \tag{4.19}$$

in which the physical processes of chemical reactions and particle formation appear naturally closed. The analysis leading to this transport equation is analogous to the development in Section 3.2.3, albeit based on the governing equations for variable density flows at low Mach number (Section 4.3.1).

In Eq. (4.19), $g(\mathbf{y}, n(\cdot); \mathbf{x}, t) = \delta\left(\mathbf{y} - \mathbf{Y}(\mathbf{x}, t)\right) \delta\left(n(\cdot) - N_\rho(\cdot, \mathbf{x}, t)\right)$ represents the fine-grained density associated with $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$. For future reference, we also introduce the density weighted *pdf* $\tilde{f}(\mathbf{y}, n(\cdot); \mathbf{x}, t)$,

$$\overline{\rho}(\mathbf{x}, t)\tilde{f}(\mathbf{y}, n(\cdot); \mathbf{x}, t) \equiv \hat{\rho}(\mathbf{y}) f(\mathbf{y}, n(\cdot); \mathbf{x}, t). \tag{4.20}$$

By following a standard procedure, we obtain on account of Eqs. (2.1), (4.1), (4.3) and (4.7) the following evolution equation for the density weighted fine-grained *pdf* ρg

$$\begin{aligned} \frac{\partial \rho g}{\partial t} + \sum_{j=1}^{3} \frac{\partial \rho u_j g}{\partial x_j} = &- \sum_{i=1}^{n_s} \frac{\partial g}{\partial y_i} \left(\rho \dot{\omega}_i - \sum_{j=1}^{3} \frac{\partial J_{ij}}{\partial x_j} \right) \\ &- \frac{\partial g}{\partial n} \left(\dot{s} - \frac{\partial G(\cdot, \mathbf{Y}) \rho N_\rho}{\partial l} - \sum_{j=1}^{3} \frac{\partial K_j}{\partial x_j} \right), \end{aligned} \tag{4.21}$$

where arguments have been omitted for brevity. Recall, at this point, that any scalar functional $\Psi(\mathbf{u}, p, \mathbf{Y}, N_\rho, \partial\mathbf{Y}/\partial\mathbf{x}, \ldots)$ of $\mathbf{u}(\mathbf{x}, t)$, $p(\mathbf{x}, t)$, $\mathbf{Y}(\mathbf{x}, t)$, $N_\rho(\cdot, \mathbf{x}, t)$ and their spatial derivatives obeys the identity (also see Eq. (3.18))

$$\overline{g\Psi\left(\mathbf{u}, p, \mathbf{Y}, N_\rho, \frac{\partial \mathbf{Y}}{\partial \mathbf{x}}, \ldots\right)} = f\overline{\left(\Psi\left(\mathbf{u}, p, \mathbf{Y}, N_\rho, \frac{\partial \mathbf{Y}}{\partial \mathbf{x}}, \ldots\right) \middle| \mathbf{y}, n(\cdot) \right)}, \tag{4.22}$$

where the vertical bar indicates conditioning on the events $\mathbf{Y}(\mathbf{x}, t) = \mathbf{y}$ and $N_\rho(\cdot, \mathbf{x}, t) = n(\cdot)$.

With the aid of Eq. (4.22), the LES-filtered value of $\overline{(\rho u_j g)}$ can be reformulated ac-

[8]If instead we based the definition of $f(\mathbf{y}, n(\cdot); \mathbf{x}, t)$ on the volumetric number density $N(\cdot, \mathbf{x}, t)$, then the transport equation for $\tilde{f}$ would include an additional unclosed term involving $\overline{\left(\sum_{j=1}^{3} \frac{\partial u_j}{\partial x_j} \middle| \mathbf{y}, n(\cdot) \right)}$.

cording to

$$\overline{(\rho u_j g)} = \hat{\rho}(\mathbf{y})\overline{(u_j|\,\mathbf{y}, n(\cdot))}f = \hat{\rho}(\mathbf{y})\tilde{u}_j f - \hat{\rho}(\mathbf{y})\left(\tilde{u}_j - \overline{(u_j|\mathbf{y}, n(\cdot))}\right)f. \tag{4.23}$$

The turbulent transport term in Eq. (4.23) is commonly modelled by adopting a gradient diffusion hypothesis

$$\hat{\rho}(\mathbf{y})\left(\tilde{u}_j - \overline{(u_j|\mathbf{y}, n(\cdot))}\right)f = \overline{\rho}(\mathbf{x}, t)\Gamma(\mathbf{x}, t)\frac{\partial \tilde{f}}{\partial x_j}. \tag{4.24}$$

Here, $\Gamma(\mathbf{x}, t) = \Gamma'(\mathbf{x}, t)/Sc'$ represents a scaled eddy viscosity, $\Gamma'(\mathbf{x}, t)$ denotes the eddy viscosity computed from the standard Smagorinsky model [103] and $Sc' = 0.7$ is a constant turbulent Schmidt/Prandtl number.

By applying the LES operator to Eq. (4.21) and taking into account Eqs. (4.17), (4.20), (4.22) and (4.24) as well as the commutation property in Eqs. (4.13) and (4.14), we arrive at the following transport equation for the joint scalar-number density *pdf*

$$\overline{\rho}\frac{\partial \tilde{f}}{\partial t} + \sum_{j=1}^{3}\overline{\rho}\tilde{u}_j\frac{\partial \tilde{f}}{\partial x_j} = \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\overline{\rho}\Gamma\frac{\partial \tilde{f}}{\partial x_j}\right) - \sum_{i=1}^{n_\phi}\frac{\partial}{\partial z_i}\left(\overline{\rho}s_i(\cdot, \mathbf{z})\tilde{f} + \overline{\rho}\mathcal{M}_i\tilde{f}\right), \tag{4.25}$$

where $n_\phi = n_s + 1$, $\mathbf{z} = (\mathbf{y}^T, n(\cdot))^T$ represents the joint scalar-number density sample space vector, $\mathcal{M}_i\tilde{f}$ encompasses the (unclosed) micromixing contribution

$$\mathcal{M}_i\tilde{f} = -\frac{\tilde{f}}{\hat{\rho}(\mathbf{y})}\begin{cases} \sum_{j=1}^{3}\overline{\left(\frac{\partial J_{ij}}{\partial x_j}\middle|\,\mathbf{y}, n(\cdot)\right)} & \text{for } i = 1, \ldots, n_\phi - 1 \\ \sum_{j=1}^{3}\overline{\left(\frac{\partial K_j}{\partial x_j}\middle|\,\mathbf{y}, n(\cdot)\right)} & \text{otherwise}\end{cases}, \tag{4.26}$$

and the vector-valued source term $\mathbf{s}(\cdot, \mathbf{z})$ is given by

$$\mathbf{s}(\cdot, \mathbf{z}) = \begin{pmatrix} \dot{\boldsymbol{\omega}}(\mathbf{y}, \hat{\rho}(\mathbf{y})n(\cdot)) \\ \frac{1}{\hat{\rho}(\mathbf{y})}\left(\dot{s}(\cdot, \mathbf{y}, \hat{\rho}(\mathbf{y})n(\cdot)) - \frac{\partial(G(\cdot,\mathbf{y})\hat{\rho}(\mathbf{y})n(\cdot))}{\partial l}\right)\end{pmatrix}. \tag{4.27}$$

4.3.4 Micromixing model

As in Section 3.2.4, we adopt the model proposed by McDermott and Pope [119] in order to close the molecular mixing term in Eq. $(4.26)_1$. Specifically, these authors augmented the IEM micromixing model for relaxing a scalar $Y_i(\mathbf{x}, t)$, $i = 1, \ldots, n_\phi - 1$, towards its Favre-filtered counterpart $\tilde{Y}_i(\mathbf{x}, t)$ by a spatially diffusive transport of $\tilde{Y}_i(\mathbf{x}, t)$,

$$\mathcal{M}_i\tilde{f} \equiv m_i(\mathbf{x}, t, \mathbf{z})\tilde{f} = \left[\kappa(\mathbf{x}, t)\left(\tilde{Y}_i - y_i\right) + \frac{1}{\overline{\rho}}\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{D}\frac{\partial \tilde{Y}_i}{\partial x_j}\right)\right]\tilde{f}, \tag{4.28}$$

where $\kappa(\mathbf{x},t)$ represents a micromixing frequency common to all scalars,

$$\kappa(\mathbf{x},t) = \frac{C_\kappa}{2}\frac{\Gamma(\mathbf{x},t)}{\Delta^2}, \tag{4.29}$$

and Δ is obtained as the cubic root of the local cell volume for a finite volume based spatial discretization scheme. The micromixing model in Eq. (4.28) may similarly be formulated for differential diffusion among gas phase scalars and consistently reduces to a pure diffusion term in the limit as the LES operator approaches the identity and a direct numerical simulation (DNS) is recovered. By adapting Eq. (4.28) to the number density micromixing term in Eq. $(4.26)_2$, we further obtain the following expression for $\mathcal{M}_{n_\phi}\tilde{f}$

$$\mathcal{M}_{n_\phi}\tilde{f} \equiv m_{n_\phi}(\mathbf{x},t,\mathbf{z})\tilde{f} = \left[\kappa(\mathbf{x},t)\left(\tilde{N}_\rho - n(\cdot)\right) + \frac{1}{\overline{\rho}}\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\tilde{D}_p\frac{\partial\overline{\rho}\tilde{N}_\rho}{\partial x_j}\right)\right]\tilde{f}. \tag{4.30}$$

Jointly, the final terms in Eqs. (4.28) and (4.30) account for differential diffusion between the gas and particulate phases. Concomitantly, the micromixing frequency $\kappa(\mathbf{x},t)$ (or the micromixing constant C_κ in Eq. (4.29)) may be adjusted separately for each scalar i, $i = 1,\ldots,n_\phi$, but we do not explore this model enhancement here (see Section 3.2.4). For the turbulent, non-premixed flame analyzed in Sections 4.5 through 4.5.4, the particle diffusivity is specifically set zero, $D_p(\mathbf{x},t) = 0$.

4.3.5 The stochastic field equations

In Sections 4.3.3 and 4.3.4, we obtained the joint scalar-number density *pdf* transport equation (Eq. (4.25) and Eqs. (4.28), (4.30)) as a model for the evolution of a fluid and an immersed particulate phase in a given flow field $\tilde{\mathbf{u}}(\mathbf{x},t)$. In view of the motivation in Section 3.2.6, we develop, in the present section, a stochastic numerical solution approach based on a stochastic process $\boldsymbol{\theta}(t;l,\mathbf{x})$ which is constructed in such a way that the transition *pdf* $h(\mathbf{z},t|\mathbf{Y}_0(\mathbf{x}),N_{\rho,0}(\cdot,\mathbf{x}),t_0;\mathbf{x})$ associated with $\boldsymbol{\theta}(t;\cdot,\mathbf{x})$ obeys Eq. (4.25) subject to the initial condition

$$h(\mathbf{z},t_0|\mathbf{Y}_0(\mathbf{x}),N_{\rho,0}(\cdot,\mathbf{x}),t_0;\mathbf{x}) = \delta(\mathbf{y}-\mathbf{Y}_0(\mathbf{x}))\delta(n(\cdot)-N_{\rho,0}(\cdot,\mathbf{x})). \tag{4.31}$$

Since both the initial time t_0 and the initial fields $\mathbf{Y}_0(\mathbf{x})$ and $N_{\rho,0}(\cdot,\mathbf{x})$ are deterministic, we drop the conditioning on $(\mathbf{Y}_0(\mathbf{x}),N_{\rho,0}(\cdot,\mathbf{x}),t_0)$ from the argument list of h for clarity and write $h = h(\mathbf{z},t;\mathbf{x})$. In a Monte Carlo-type solution method, several realizations (samples) $\boldsymbol{\theta}^{(1)}(t;l,\mathbf{x}),\ldots,\boldsymbol{\theta}^{(n_f)}(t;l,\mathbf{x})$ of such a stochastic process $\boldsymbol{\theta}(t;l,\mathbf{x})$ are computed and expectations are approximated by Monte Carlo estimates,

$$\tilde{F}(\mathbf{Y}(\mathbf{x},t),N_\rho(\cdot,\mathbf{x},t)) \approx \frac{1}{n_f}\sum_{i=1}^{n_f}F(\boldsymbol{\theta}^{(i)}(t;\cdot,\mathbf{x})), \tag{4.32}$$

where $F(\mathbf{Y}, N_\rho)$ represents a sufficiently smooth observable expressed in terms of $\mathbf{Y}(\mathbf{x}, t)$ and $N_\rho(\cdot, \mathbf{x}, t)$. This approach channels the computational effort towards the accurate computation of low order moments of $h(\mathbf{z}, t; \mathbf{x}) = \tilde{f}(\mathbf{z}; \mathbf{x}, t)$, while the statistical error is accumulated on higher order moments.

As in Section 3.2, we specifically invoke the method of stochastic fields to construct the stochastic process $\boldsymbol{\theta}(t; l, \mathbf{x})$. This approach preserves the Eulerian nature of the physical model; it was originally developed by Valiño [200] and Hauke and Valiño [71] and later rationalized by Sabel'nikov and Soulard [179]. Formally, the stochastic field equations are given by

$$\begin{aligned}\overline{\rho}\frac{\partial \theta_i}{\partial t} + \sum_{j=1}^{3} \overline{\rho}\tilde{u}_j \frac{\partial \theta_i}{\partial x_j} = \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(\overline{\rho}\Gamma(\mathbf{x}, t)\frac{\partial \theta_i}{\partial x_j}\right) - \overline{\rho}\sqrt{2\Gamma(\mathbf{x}, t)}\sum_{j=1}^{3} \dot{W}_j(t)\frac{\partial \theta_i}{\partial x_j} \\ + \overline{\rho}\left(s_i(l, \boldsymbol{\theta}) + m_i(\mathbf{x}, t, \boldsymbol{\theta})\right), \quad i = 1, \dots, n_\phi,\end{aligned} \tag{4.33}$$

and correspond to continuous-time stochastic processes which are smoothly parameterized by $(l, \mathbf{x})$. In Eq. (4.33), $\mathbf{W}(t)$ denotes a three-dimensional temporal Wiener process with (formal) time derivative $\dot{\mathbf{W}}(t)$. Strictly, Eq. (4.33) is only formally valid, holding in a time-integral sense

$$\begin{aligned}\int_{t_0}^{t} d\theta_i(t; l, \mathbf{x}) &= \int_{t_0}^{t} \frac{\partial \theta_i}{\partial t}\, dt \\ &= -\sum_{j=1}^{3} \int_{t_0}^{t} \tilde{u}_j \frac{\partial \theta_i}{\partial x_j}\, dt + \dots - \sum_{j=1}^{3} \int_{t_0}^{t} \sqrt{2\Gamma(\mathbf{x}, t)}\frac{\partial \theta_i}{\partial x_j}\, dW_j(t) + \dots,\end{aligned} \tag{4.34}$$

where the stochastic integral with respect to $\mathbf{W}(t)$ is interpreted in Itô's sense.

In Appendix B.2.2 we show that if the stochastic fields $\boldsymbol{\theta}(t; l, \mathbf{x})$ evolve according to Eq. (4.33) subject to the deterministic initial condition $\boldsymbol{\theta}(t_0; l, \mathbf{x}) = (\mathbf{Y}_0(\mathbf{x}), N_{\rho,0}(l, \mathbf{x}))$, then the transition *pdf* $h(\mathbf{z}, t; \mathbf{x})$ associated with the stochastic fields evolves according to Eq. (4.25). (The proof here generalizes the derivation in Appendix B.2.1 to variable density flows at low Mach number.)

4.4 Gas phase and soot kinetics

4.4.1 Gas phase kinetics and radiation

The gas phase chemical kinetics for methane combustion are based on the GRI 1.2 reaction mechanism [53, 54]. In order to model radiation, we adopt the hypothesis of optical thinness. Following Lindstedt and Louloudi [106], the loss in enthalpy $H(\mathbf{Y}, N)$ due to radiation from gas phase H_2O, CO_2, CH_4 and CO as well as from soot can be computed according to

$$\begin{aligned}\dot{\omega}_H(\mathbf{Y},N) = &-\frac{4\sigma}{\hat{\rho}(\mathbf{Y})}\left(T(\mathbf{Y})^4 - T_b^4\right)\sum_{i=1}^{4} a_{p,i}(T(\mathbf{Y}))p_i(\mathbf{Y}) \\ &-\frac{4\sigma}{\hat{\rho}(\mathbf{Y})}C_s f_v(N)\left(T(\mathbf{Y})^5 - T_b^5\right),\end{aligned} \tag{4.35}$$

where σ denotes the Stefan-Boltzmann constant, $T(\mathbf{Y})$ indicates temperature, $i = H_2O, CO_2, CH_4, CO$ represents an index running through the major radiating species, $p_i(\mathbf{Y})$ is the partial pressure of species i (in [atm]) and $a_{p,i}(T)$ denotes its corresponding Planck mean absorption coefficient. The latter is computed in terms of temperature from polynomial curve fit expressions of the RADCAL model [66, 75]. Furthermore, $T_b \equiv 295\,\mathrm{K}$ denotes the ambient background temperature, $C_s \equiv 1307\,/\mathrm{m}-\mathrm{K}$ [106] and $f_v(N) = \pi/6M_3(N)$ is the soot particle volume fraction.

One implication of the optical thinness hypothesis is that no reabsorption of thermal radiation occurs within the flame. In heavily sooting flames this may lead to an overprediction of the radiative heat loss and, by consequence, to a local underprediction of temperature [194]. The Delft III flame, however, is only lightly sooting with measured soot volume fractions below 2.5 ppb.

4.4.2 Soot kinetics

In the present work, we focus on the kinetic processes of soot nucleation, growth and oxidation and invoke kinetic rate expressions which have previously been employed in the context of laminar methane and ethylene diffusion flames [3, 69, 109, 193]. Since the coalescent growth of a soot particle from two parent particles may only be important in the initial stages of particle creation and since aggregation preserves the number density of primary soot particles [109], we omit both coalescence and aggregation and interpret $N(l,\mathbf{x},t)$ as the primary soot particle number density. Formally, however, the univariate PBE is able to accommodate coagulation.

Following Liu et al. [109], the nucleation rate is controlled by the molar concentration of acetylene and can be computed from

$$s_N(\mathbf{Y}) = \frac{2N_A}{C_{\min}}[C_2H_2]k_1(T), \tag{4.36}$$

where N_A denotes Avogadro's number, $C_{\min} = 700$ is the number of carbon atoms in a soot nucleus, $k_1(T) = 1.7\exp(-7548\,\mathrm{K}/T)(1/s)$, and square brackets indicate molar concentrations. The specific surface growth rate (in $[\mathrm{kg/m^2}-\mathrm{s}]$) of primary soot particles, on the other hand, is given by

$$s_{C_2H_2}(\mathbf{Y}) = 2p_{C_2H_2}(\mathbf{Y})k_2(T), \tag{4.37}$$

where $p_{\mathrm{C_2H_2}}(\mathbf{Y})$ represents the acetylene partial pressure (in [atm]) and $k_2(T) = 470 \exp(-16\,004\,\mathrm{K}/T)(s/m)$ [69, 193]. The leading factor of 2 in Eq. (4.37) is purely empirical; it has been introduced by Smooke et al. [193] in order to bring the growth rate which has originally been determined based on measurements in a laminar premixed ethylene flame [70] closer to measurements taken in laminar diffusion flames. The mean soot nuclei size amounts to $l_{\mathrm{nuc}} = 2.5 \times 10^{-9}\,\mathrm{m}$ and the minimum and maximum attainable particle sizes are set to $l_l = 2.5 \times 10^{-10}\,\mathrm{m}$ and $L = 10^{-5}\,\mathrm{m}$, respectively.

Additionally, primary soot particles may shrink on account of oxidative surface reactions with hydroxyl or molecular oxygen [69, 134]. Expressed in terms of the primary particle mass, the respective specific shrinkage rates (in $[\mathrm{kg/m^2-s}]$) are computed according to

$$s_{\mathrm{OH}}(\mathbf{Y}) = 167\frac{p_{\mathrm{OH}}(\mathbf{Y})}{\sqrt{T}}, \tag{4.38}$$

$$s_{\mathrm{O_2}}(\mathbf{Y}) = 1200 p_{\mathrm{O_2}}(\mathbf{Y})\left(\frac{K_A(T)\chi}{1+K_z(T)p_{\mathrm{O_2}}} + K_B(T)\chi'\right), \tag{4.39}$$

where $p_{\mathrm{OH}}(\mathbf{Y})$ and $p_{\mathrm{O_2}}(\mathbf{Y})$ denote the partial pressures (in [atm]) of hydroxyl and molecular oxygen, respectively, χ and χ' represent fractions defined by

$$\chi = \left(1 + \frac{K_T(T)}{K_B(T)p_{\mathrm{O_2}}}\right)^{-1}, \tag{4.40}$$

$$\chi' = 1 - \chi, \tag{4.41}$$

and $K_A(T)$, $K_B(T)$, $K_T(T)$ and $K_z(T)$ are the temperature dependent parameters introduced by Nagle and Strickland-Constable [134, p. 162]. Finally, the cumulative primary particle growth/oxidation rate $G(\mathbf{Y})$ can be computed by summing Eqs. (4.37) through (4.39) and converting from a mass-based to a diameter-based rate expression

$$G(\mathbf{Y}) = \frac{2}{\rho_s}\left(s_{\mathrm{C_2H_2}}(\mathbf{Y}) - s_{\mathrm{OH}}(\mathbf{Y}) - s_{\mathrm{O_2}}(\mathbf{Y})\right), \tag{4.42}$$

where $\rho_s = 1900\,\mathrm{kg/m^3}$ denotes the density of soot [109].

If primary soot particles are created and grow or, conversely, shrink due to oxidation, then gas phase species are consumed or released. This can be taken into account by augmenting the source terms due to chemical reactions by rate expressions based on the soot formation and oxidation stoichiometry as well as Eqs. (4.36) through (4.39). For acetylene and molecular hydrogen, for instance, we have

$$\frac{1}{MW_{\mathrm{C_2H_2}}}\dot{\omega}^{\star}_{\mathrm{C_2H_2}}(\mathbf{Y},N) = -\frac{1}{MW_{\mathrm{H_2}}}\dot{\omega}^{\star}_{\mathrm{H_2}}(\mathbf{Y},N) = -\frac{s_{\mathrm{C_2H_2}}(\mathbf{Y})f_a(N)}{2\hat{\rho}(\mathbf{Y})MW_s} - \frac{s_N(\mathbf{Y})C_{\min}}{2\hat{\rho}(\mathbf{Y})N_A}, \tag{4.43}$$

where the superscript $\star$ indicates that the respective scalar sink/source terms exclude contributions from the gas phase reaction mechanism. MW_i denotes the molecular weight of species i, $MW_s \equiv 12.011\,\mathrm{kg/kmol}$ represents the molecular weight of solid soot and $f_a(N) = \pi M_2(N)$ denotes the local soot particle surface fraction.

In view of the findings of References [11, 93, 135], we set the kinematic diffusivity of soot in Eq. (4.30) to zero, $D_p(\mathbf{x}, t) \equiv 0$ (also see Section 4.3.4).

4.5 Delft III flame

4.5.1 Flame configuration

The Delft III flame consists of a central fuel jet (Dutch natural gas) surrounded by two concentric co-flows of air at atmospheric pressure and an ambient temperature of 295 K [152]. The nozzle encompasses a wide cylindrical ring with inner diameter $d = 6\,\mathrm{mm}$ and outer diameter 15 mm featuring twelve equidistant holes of diameter 0.5 mm on a circle with diameter 7 mm from which the pilot flames emanate. The nozzle is embraced by an annulus of outer diameter 45 mm for the primary air co-flow which is, in turn, surrounded by a secondary air co-flow. The jet flows at a bulk velocity of 21.9 m/s ($Re \approx 8370$), while the bulk velocities of the primary and secondary air co-flows amount to 4.4 m/s and 0.3 m/s, respectively. The Delft III flame is characterized by strong extinction and reignition in the nearfield and possesses a lightly sooting flame head. Figure 4.1 schematically illustrates the geometry of the Delft burner tip as well as our computational domain.

Following Merci et al. [122, 123], we take the fuel composition as a mixture of methane and nitrogen with the same calorific value as Dutch natural gas (85.3 % CH_4 and 14.7 % N_2 by volume). The pilot flames, moreover, burn a mixture of hydrogen, acetylene and air with a H to C ratio of 4 and an equivalence ratio of 1.4 (16.936 % H_2, 5.682 % C_2H_2, 16.258 % O_2 and 61.124 % N_2 by volume). Here, air is assumed to consist of 21 % O_2 and 79 % N_2 by volume. For simplicity, the twelve pilot flames are replaced by a concentric inflow through an annulus in the nozzle rim with inner diameter 8 mm and outer diameter 9 mm at the experimental pilot flame mass flow rate of $2.3 \times 10^{-5}\,\mathrm{kg/s}$. Note that contrary to Ayache and Mastorakos [12] and Dodoulas and Navarro-Martinez [41] the pilot stream is separated from the fuel jet by a wall of 1 mm thickness in order to avoid the pilot gases from diffusing into the fuel mixture upstream of the nozzle exit plane. Although this representation of the pilot underestimates the experimental pilot flames' momentum flow rate, we found it to be sufficient to ignite the flame and lead to flame attachment at the nozzle rim (as observed in the experiment). The pilot inflow composition is taken as the chemical equilibrium composition corresponding to the unburned pilot pre-mixture at 295 K and the pilot inflow temperature is set to 1900 K.

Experimentally, the Delft III flame was first investigated by Peeters et al. [152] who reported measurements of velocity statistics, mean temperature as well as concentrations of

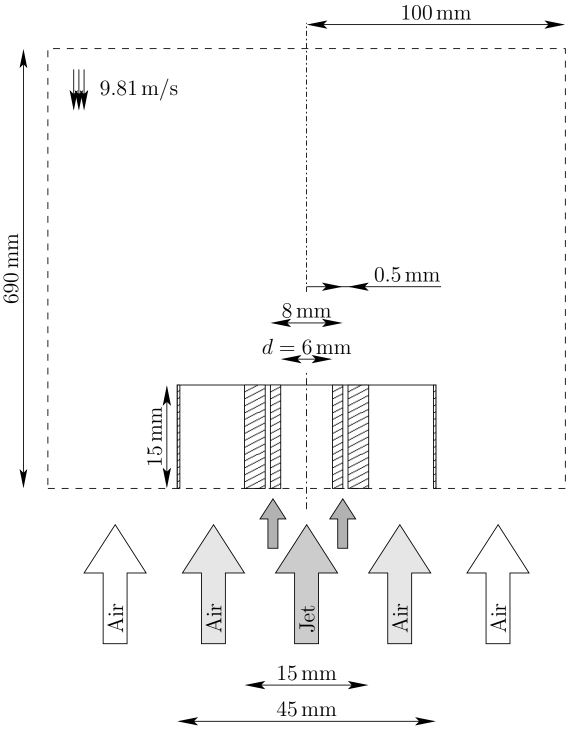


Figure 4.1 Schematic illustration of the Delft burner nozzle and the computational domain for the Delft III flame calculation.

OH and a passive scalar. Subsequently, additional measurements of the velocity field, concentrations of the major species and temperature statistics were obtained, see Nooren et al. [143] and references therein. Recently, Qamar et al. [165] augmented the experimental database of the Delft III flame by measurements of mean soot volume fraction, intermittency and centerline *pdfs* of instantaneous soot volume fraction. These measurements were obtained in the downstream region of the flame ($x/d \gtrsim 50$), where unfortunately no velocity, temperature or scalar measurements are available. As Mueller and Raman [129] pointed out, this may render drawing definitive conclusions from a comparison of model predictions with measured soot-related quantities difficult.

4.5.2 Previous investigations of the nearfield

One of the first modelling investigations of the Delft III flame is due to Peeters et al. [152] who combined a k-ϵ turbulence model including a round jet correction with a constrained equilibrium chemistry model and a four flux radiation model. Here, the reactive scalars were uniquely related to the mixture fraction whose one-point, one-time *pdf* was taken as a β-function *pdf* parameterized by mixture fraction mean and variance. Peeters et al. [152] achieved very good quantitative predictions of the mean velocity and temperature fields and qualitatively accurate representations of turbulence quantities, but concluded that, at least for radical predictions, the conserved scalars/constrained equilibrium model was invalid. Subsequently, Nooren et al. [142] reconsidered the Delft III flame in the context of a round jet corrected k-ϵ model matched with a two dimensional stochastic particle based Monte Carlo solver for the joint velocity-scalar *pdf*. These authors employed both the constrained equilibrium chemistry model of Peeters et al. [152] and a reduced chemical kinetics ILDM model in which all reactive scalars were expressed in terms of the mixture fraction as well as the H_2O and CO_2 mass fractions as the kinematically controlling variables. Their results indicated a strong dependency of temperature standard deviation on the micromixing model. For the ILDM chemistry, moreover, Nooren et al. [142] achieved significantly improved OH predictions.

Revisiting the results of Peeters et al. [152], Merci et al. [122] compared predictions obtained from a nonlinear k-ϵ model combined with an assumed β-function *pdf* for mixture fraction and a laminar flamelet or equilibrium chemistry model against an updated experimental database. In a second approach [122, 123], they combined the nonlinear k-ϵ model with a transported joint scalar *pdf* method and a C_1 skeletal reaction mechanism. Like Nooren et al. [142], Merci et al. [123] observed a strong dependency of flame ignition and stabilization on the micromixing model. In fact, their predictions of the flow, mixing and temperature fields as well as of the CO mass fractions conditioned on mixture fraction were compared to those of Nooren et al. [142] by Roekaerts et al. [176]. In line with the findings of Merci et al. [123] and Nooren et al. [142], Roekaerts et al. [176] emphasized the importance of the pilot flame model and, focussing on the C/D micromixing model, the choice of model constant. Also, they noted that CO was unsatisfactorily predicted by both models, potentially due to the pilot flame models and/or inherent limitations of the chemical reaction mechanisms.

In the context of LES, the Delft III flame has recently been considered as a test case for capturing and analyzing local extinction and reignition events. Ayache and Mastorakos [12], for instance, validated an LES-CMC approach combined with the GRI 3.0 reaction mechanism against the Delft III database. They obtained very good predictions for the temporal statistics of velocity, mixture fraction, temperature and CO mass fraction (with a slight deterioration of the predictions towards the domain outlet). NO, on the other hand, was overpredicted which might have been due to the omission of radiation (leading to a slight temperature overprediction) or to inherent limitations of the GRI 3.0 mechanism to capture NO formation. Dodoulas and Navarro-Martinez [41], moreover, analyzed the near-field of the Delft III flame in order to identify the flame structure and gain insight into the formation of extinction pockets by postprocessing results obtained from an LES augmented by a transport equation for the joint scalar *pdf*. Here, the modelled *pdf* transport equation (IEM micromixing model) was solved using the method of stochastic fields and the chemical kinetics were obtained from a 15 step augmented reduced mechanism derived from GRI 3.0. The time averages of the flow and mixing fields as well as of temperature and the major species were well predicted, while NO was overpredicted downstream. Contrary to the RANS-based modelling approaches reviewed above, the representation of the pilot flames seemed to be much less crucial for the fully Eulerian LES-based models.

4.5.3 Numerical solution scheme and implementational aspects

Following our developments in the present chapter, the implementation of a stochastic field based numerical solution scheme in LES-BOFFIN (see Section 3.3) is generalized to variable density flows at low Mach numbers. Similar to Section 3.3, the stochastic field equation associated with the particle size distribution is discretized in particle size space using the explicit adaptive grid method of Chapter 2 in combination with a high resolution finite volume method [92], also see Appendix B.3. Here, the total number of nodes, the minimum node density in the nucleation interval and the maximum grid stretching are set to 30, 4 nodes/4.75 nm and 2, respectively. Furthermore, in line with Section 3.3, we invoke eight realizations of the stochastic fields.

For efficiency, the gas-phase reaction fractional step is only executed for fluid cells whose temperature exceeds 800 K, while the particle reaction step is solved for all fluid cells. In order to further accelerate the solver for the reaction fractional step, we hard-coded the instructions for evaluating the reaction rates of the GRI 1.2 reaction mechanism, implemented a facility for computing the temperature-dependent kinetic coefficients only once per time step and grid point, and adopted a modified Newton-Raphson scheme for solving the backward Euler non-linear system. Cumulatively, these measures yield a reduction in runtime of the reaction step by about one order of magnitude.

The computational domain is cylindrical in shape and spans 16.67 d in the radial direction and 115 d in the axial direction, where $d = 6\,\mathrm{mm}$ denotes the nozzle diameter (Figure 4.1). Since the wide nozzle rim of the Delft III burner head acts as a bluff body enhancing the mixing and reinforcing the shear layer in the near-field, we include a representation of the burner nozzle which extends by 2.5 d into the domain. The finite volume grid encompasses 672, 70 and 36 cells in the axial, radial and circumferential directions, respectively. Axially, the grid is stretched by a factor of 1.002 and, radially, cells are thinner near the inner and outer nozzle diameter and stretch by a factor of 1.04 towards the lateral domain boundaries. In the circumferential direction, by contrast, a uniform grid is employed.

The mean axial velocity inflow profiles of both the fuel jet and the primary air co-flow are taken as power law profiles with exponents 1/6 and 1/2, respectively. The secondary air co-flow, on the other hand, features a constant mean axial velocity inflow profile. The velocity turbulence intensities for the jet and primary air co-flow, moreover, are set to 10 %, while the turbulence intensity in the secondary co-flow amounts to 1 %. The axial and radial root mean square (rms) inflow velocities vary quadratically within the jet (at the nozzle rim, they exceed the nominal centerline value by a factor of four) and reduce to constant profiles in the primary and secondary air co-flows, respectively. Along the lateral boundaries of the domain, both the stochastic scalars and the velocity field are subject to Dirichlet boundary conditions based on nominal values in the secondary co-flow. Finally, at the domain outlet, zero-gradient and convective outflow boundary conditions apply, respectively, to the stochastic scalars and the velocity field.

In order to assess the validity of our inflow boundary conditions, the mean and rms profiles of axial velocity were compared with the experimentally measured profiles at 3 mm above the nozzle exit plane (not shown). Except for a slight overprediction on the centerline, our choice of inflow boundary conditions approximates well the experimental mean axial velocity profile, including the weak recirculation zone above the nozzle rim. The rms of axial velocity are also well reproduced in the jet, except near the burner rim, but fall below the measured values in the primary co-flow, very similar to the rms profile obtained by Ayache and Mastorakos [12].

Temporal statistics were computed over a time period of approximately 250×10^{-3} s and complemented by circumferential averaging. The time measurements which we provide were obtained on 4 nodes of a Cray XC30 Supercomputer (ARCHER UK) and averaged over 100 time steps of 1.2×10^{-6} s at a point in time at which the temporal statistics of temperature had become time invariant.

4.5.4 Results and discussion

In Figure 4.2 the radial profiles of the time averaged (mean) axial velocity and temperature[9] at 50 mm, 150 mm and 250 mm above the nozzle exit plane are compared with measurements from the Delft III database [143, 152]. At the first two measurement stations, the mean axial velocity agrees well with the measured values, while it is slightly underpredicted further downstream, indicating that the jet spreads rather too rapidly. The mean temperature, moreover, is slightly overpredicted on the lean side of the reaction zone and the maximum mean temperature exceeds the measured maximum value by approximately 300 K. Since this overprediction is passed downstream, its main cause seems to persist in the near-nozzle region and may be related to the spatial resolution of our LES in the nearfield as well as the effectiveness or accuracy of the eddy viscosity and micromixing closures. Indeed, other LES-based investigations of the Delft III flame [12, 41, 128] reported much better agreement for the nearfield predictions using grids whose radial mesh spacing was smaller than the one in our grid by up to a factor of three.

For the same axial measurement stations as in Figure 4.2, Figure 4.3 depicts both predicted and measured radial profiles of the mean mass fractions of CO_2, H_2O, H_2 and CO. The agreement here reflects the discrepancy which we observed above for the temperature profiles: At the first measurement station, the gas composition is well reproduced on the rich side, while the maximum mass fractions and the values on the lean side are slightly overpredicted. The centerline value of CO_2 is slightly underpredicted at $x = 150$ mm, while the CO mass fraction is overpredicted near the centerline, the overprediction per-

[9]For conciseness, we omit the term 'Favre-filtered' when referring to both instantaneous and time averaged Favre-filtered variables.

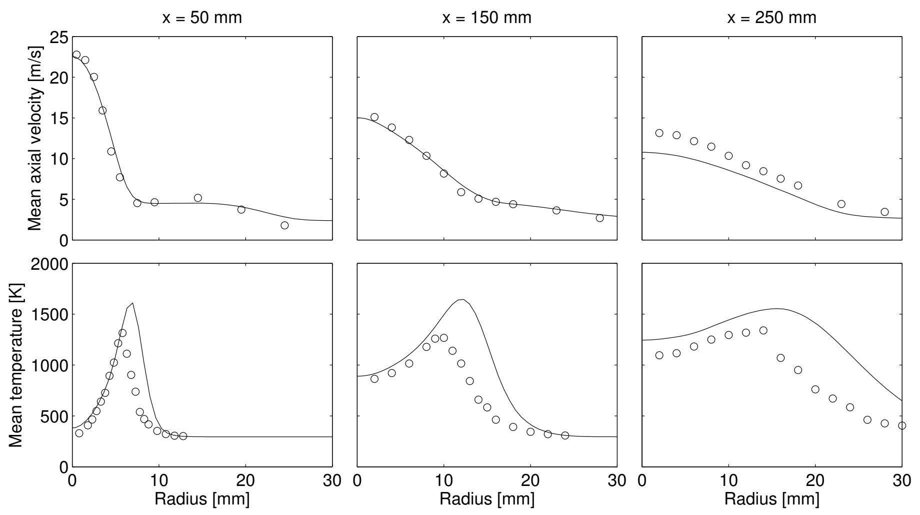


Figure 4.2 Comparing radial profiles of the mean axial velocity and temperature (lines) with experimental measurements (symbols) at 50 mm, 150 mm and 250 mm above the nozzle exit plane.

sisting throughout the radial profile. In general, the species profiles seem to be shifted radially outwards, reflecting the slightly excessive jet spreading. This is even more severe at the furthest measurement station, where the species profiles appear to be very diffusive. Here, the mass fractions of H_2O and CO are overpredicted throughout and the H_2 mass fraction is notably underpredicted in the jet core.

Figure 4.4 depicts contour plots of the mean temperature, soot number density and soot volume fraction. The superimposed contour in the left panel indicates the stoichiometric mixture fraction iso-line. For the center and right panels, the soot number and volume densities were computed as the zeroth and third moments, respectively, of the LES-filtered soot size distributions. At least qualitatively, the contours of the mean soot number density and volume fraction are in line with those reported by Mueller and Pitsch [128]. Both fields attain their maximum values on the centerline and the soot number density peaks slightly earlier, at $x \approx 350\,\mathrm{mm}$, than the soot volume density ($x \approx 475\,\mathrm{mm}$). These observations indicate that nucleation is more vigorous at distances closer to the nozzle, yielding to soot surface growth further downstream. Furthermore, soot oxidation appears to act more effectively on the number density, while soot volume fraction shows a slightly delayed response. In view of the soot particle size distributions analyzed below (Figure 4.7), this may be the case because most soot particles persist in the nuclei size range and are, hence, rapidly removed by oxidation. By contrast, the larger soot particles which contribute significantly to the soot volume fraction can resist an oxidative gas composition for longer. Beyond the stoichiometric mixture fraction contour most soot has been oxidized and both the soot number density and volume fraction are close to vanishing.

In Figure 4.5, the centerline profiles of the predicted and experimentally measured [165] mean soot volume fraction are compared. Here, the LES-PBE-PDF results indicate that soot formation commences much further upstream than was observed experimentally and, quantitatively, the mean soot volume fraction is underpredicted by one order of magnitude. A similar early onset and decay of soot formation was observed by Mueller and Pitsch [128] and Donde et al. [42], albeit for a soot model based on the hybrid method of moments and gas phase/soot kinetics which differed from the ones we employ here. These authors attributed the early onset to uncertainties in the soot formation kinetics.

In the context of a RANS-presumed *pdf* approach based on the semi-empirical soot model of Brookes and Moss [28], Reddy et al. [172] found that the upstream shift of the centerline soot volume fraction profile can be remedied by computing the OH and O radical concentrations from equilibrium and partial equilibrium relations independent of the governing gas phase reaction mechanism and the turbulence-chemistry interaction model. Since the semi-empirical model of Brookes and Moss [28] is based on C_2H_2, this is, possibly, related to the strong dependency of the C_2H_2 yield on the concentrations of OH and O which may be increased by the equilibrium and partial equilibrium chemistry in the nearfield. Reddy and De [171] and Reddy et al. [170] and Reddy et al. [172] further assessed the influence of different radiation models, but found that, while the peak soot volume fraction on the centerline can vary significantly, its location changes only slightly.

If we set aside the assumption of (partial) equilibrium OH and O concentrations, then both the present and previous modelling attempts of the Delft III flame predict an early onset and termination of soot formation. Since the modification employed by Reddy and De [171] and Reddy et al. [172] affects both the gas phase chemistry and overrides the turbulence-chemistry submodel, this indicates that either aspect may be held accountable. On the other hand, comparing our prediction and previous results for the soot volume fraction indicates that closures of the turbulence-soot formation interaction and the soot kinetics do not have as large an influence on the location of maximum soot volume fraction, albeit being important for quantitative differences. Consequently, it seems possible that there is a physical effect which is relevant for soot formation in methane-air combustion at moderate Reynolds numbers, but which both our attempt and previous investigations of soot formation in the Delft III flame omitted. Such an effect could be related to differential diffusion within the gas phase or differential micromixing between gas phase scalars and soot. For sooting ethylene flames at larger Reynolds numbers, Xuan and Blanquart [210] and Koo et al. [91] obtained very good predictions of soot volume fraction using LES-models similar to the one proposed by Mueller and Pitsch [128]. In this light the ‘missing’ physical effect, may turn less important under these conditions.

Apart from the upstream shift of the maximum soot volume fraction, our results in Figure 4.5 are characterized by an almost immediate onset of soot formation in the nearfield and a rather abrupt termination about 500 mm above the nozzle. The early onset of soot formation reflects the abundance of C_2H_2 and the large underprediction of soot volume fraction as well as the rapid decline in the vicinity of stoichiometric conditions indicates that soot oxidation is rather vigorous, potentially catalyzed by the temperature overprediction. Attili et al. [11] pointed out that soot surface growth takes place on much larger time scales than soot oxidation; in this light, it is possible that the residence time of soot pockets in flame regions which favour surface growth is too short, perhaps owing to limitations in the spatial LES resolution.

In order to elucidate the preferential location of soot relative to the flame front, we analyze the conditional soot number density and volume fraction in mixture fraction space for three different flame cross-sections in Figure 4.6. Here, the scatter represents instantaneous values sampled at several different time points, while the solid lines indicate time averages. The vertical dashed lines, furthermore, indicate the stoichiometric mixture fraction value of 0.0705. In terms of both instantaneous scatter and mean values the soot number density and volume fraction correlate well in mixture fraction space, reflecting the dominance of nucleation mentioned above. The instantaneous scatter covers a wide range of number density and volume fraction values on the rich side of the flame, emphasizing that soot number density and soot volume fraction may not be uniquely related to mixture fraction. On the lean side, by contrast, some scatter remains, but the number density and volume fraction values here are significantly reduced. Possibly, oxidation is enhanced by the micromixing model whose rate applies, at the moment, equally to the gas phase scalars and to soot. Further downstream, both soot number density and volume fraction shift towards leaner mixture fraction values as the flame tip is approached. The soot that persists here at low mixture fraction values is completely oxidized away by $x = 550\,\mathrm{mm}$.

Figure 4.7 depicts the instantaneous soot particle size distributions along the centerline of the flame and across the flame at the axial distance of maximum soot number density ($x = 350\,\mathrm{mm}$). The grid lines which emanate from the particle size axis illustrate the grid adaptivity in particle size space. While the grid nodes are rather evenly spaced on a logarithmic scale at large particle sizes, the majority of nodes have been drawn into the vicinity of the mean nuclei size at 2.5 nm and maintain an accurate resolution of the sharp rise and decline in particle number density. Both along and across the flame, the soot particle size distributions do not vary significantly in shape such that the grid nodes in particle size space display only very little spatial variability here. Throughout, the soot particle size distribution remains unimodal and, at the largest values of total soot number density and volume fraction, remains dominated by nucleation. In line with our observations for Figure 4.4, the particle number density per unit of length in particle size space takes on maximum values near the centerline and rapidly decreases further outwards.

In Table 4.2, we compare the average runtimes of the fractional steps for the stochastic scalars and the flow solver on 4 nodes of a Cray XC30 Supercomputer (ARCHER UK). Here, the reaction fractional step for approximately 35.0 % of reacting fluid cells takes slightly longer to execute than the PBE fractional step (including radiation, species consumption/release and particle size grid adaptation) which is called on all finite volume cells due to grid adaptivity. The PBE fractional step thus consumes about 27.2 % of the overall runtime per time step, approximately matching the time fraction of 26.3 % jointly

Physical process	Average runtime [s]
Flow field	1.94
Scalar convection/diffusion	6.89
Mixing	1.017
Gas phase reaction	10.85
Particle phase reaction	9.11
All processes	33.54

Table 4.2 Average runtimes for advancing the LES-PBE-PDF model by one time step ($\Delta t = 1.2 \times 10^{-6}$ s) on 4 nodes (96 MPI processes) of a Cray XC30 Supercomputer (ARCHER UK).

required by scalar convection/diffusion and the flow solver.

The time measurements in Table 4.2 indicate that the combined LES-PBE-PDF approach is computationally feasible on modest resources of a modern computing system and that, compared to the gas phase reaction step, the PBE step does not significantly increase the overall runtime. Our observations thus demonstrate that a detailed resolution of particle size space within each fluid cell is viable and even computationally efficient. By contrast, the majority of studies investigating soot formation in turbulent flames have favoured moment based methods, at least in part, due to the concern that a detailed PBE model would incur excessive computational costs.

4.6 Chapter summary

In the present chapter, we incorporated the PBE as a Eulerian description for the evolution of a polydispersed particulate phase into an LES model of a turbulent reacting flow with variable density. An important application of our approach is the prediction of soot particle size distributions in turbulent hydrocarbon flames. In order to resolve the turbulence-chemistry/particle formation interaction, we obtained an evolution equation for the filtered one-point, one-time joint *pdf* of the instantaneous reactive scalars and particle number density. Here, turbulent transport was closed by a gradient diffusion hypothesis, while two-point correlations of the reactive scalars and number density were replaced by a micromixing model that accounts for differential diffusion between the gas and particle phases.

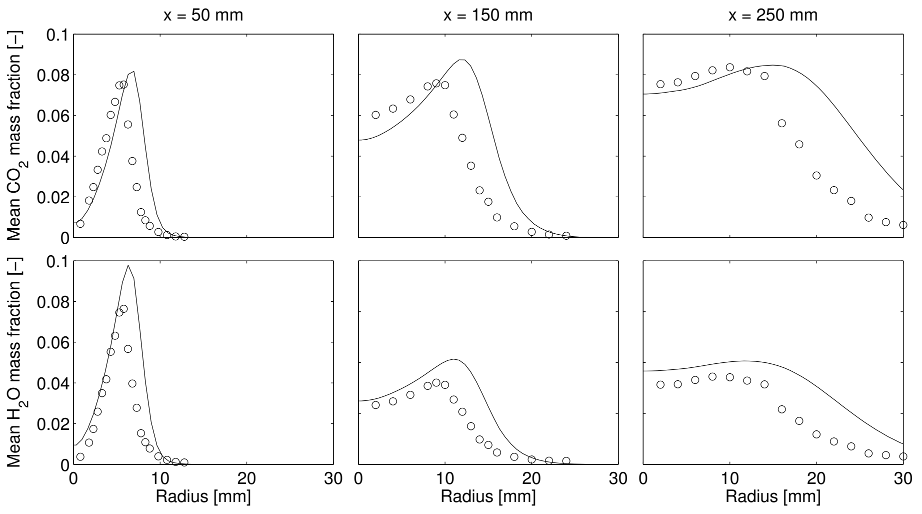
Numerically, the joint scalar-number density *pdf* transport equation was solved using the method of Eulerian stochastic fields. In the context of LES, this approach has the advantage that its spatial resolution is independent of the large-scale flow structures – as opposed to stochastic particle based solvers, for instance. The stochastic process governing the stochastic fields is constructed such that, in a statistical sense, the evolution of the joint scalar-number density *pdf* is reproduced. An important feature of our numerical solution scheme is the adaptive grid discretization of particle size space (see Chapter 2). This technique allows particle size distributions which vary largely in shape and width across the flow domain, potentially including sharp peaks or near-discontinuities, to be represented with similar accuracy, while keeping the number of grid points low.

The combined LES-PBE-PDF model was applied to model soot formation in the Delft III diffusion flame. Here, kinetic rate expressions for soot nucleation and growth were adopted from previous laminar diffusion flame calculations and both species depletion as well as radiation based on the hypothesis of optical thinness were included. At present, the formation of chain-like soot aggregates is not considered such that our soot phase is described solely in terms of spherical primary particles. Similarly, the coagulation of nascent soot particles is omitted, although we intend to incorporate coagulation in future times. The gas phase kinetics were represented by the detailed GRI 1.2 reaction mechanism.

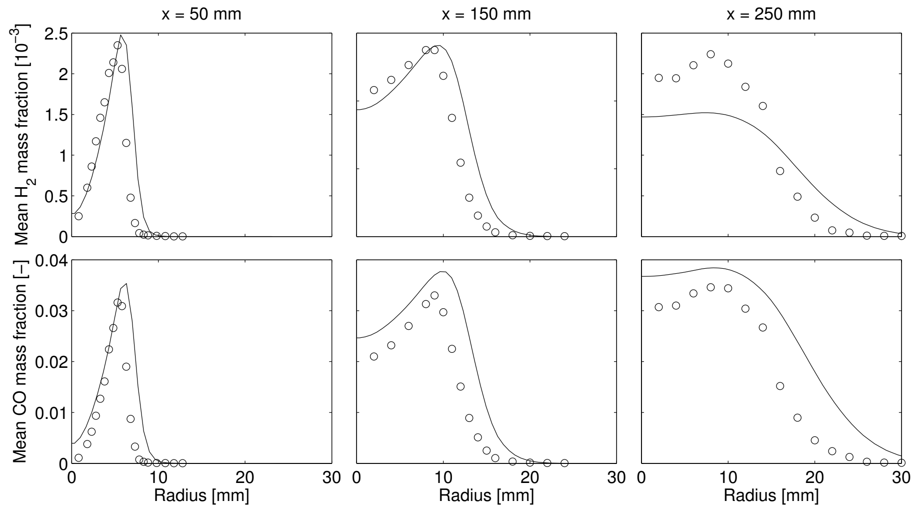
In the nearfield of the Delft III flame both the mean velocity and gas phase scalars agree reasonably well with measurements. The overprediction on the lean side here seems to be due to limitations of the spatial resolution which may render the turbulence closures less accurate. Similar to previous LES investigations of the Delft III flame, the soot volume fraction peaks further upstream than in the experimental observations. Additionally, soot volume fractions are notably underpredicted. However, except for the upstream shift and a slightly early onset of soot formation, the centerline soot volume fraction profile compares well qualitatively with the experimental data. Predictions of the soot particle size distribution reflect the dominance of soot nucleation and oxidation and their influence on the shape of the local particle size distribution. In terms of performance, we found the solver for the PBE fractional step to only consume a modest fraction of the average runtime per time step.

Since most of the computational cost remains concentrated in the gas phase reaction step, we aim at incorporating an apt tabulation technique in the near future [52]. Additionally, the more recently proposed soot kinetics are based on PAH chemistry and, hence, require a more comprehensive gas phase reaction mechanism. By applying a mechanism reduction technique, see, for example, Reference [90], the number of gas phase scalars may be manageably reduced. On part of the soot formation processes, the present LES-PBE-PDF framework is also suited for accommodating coagulation of incipient soot particles. As a final model enhancement we mention the inclusion of gas phase differential diffusion as well as the prescription of a different micromixing frequency for each gas phase scalar and for soot.

In conclusion, our investigation has demonstrated that modelling soot formation in a turbulent flame based on a detailed PBE-PDF approach is not only advantageous seeing as any soot kinetics can be accommodated without approximation and the entire particle size distribution is predicted, but also computationally efficient.



(a) CO_2 and H_2O



(b) H_2 and CO

Figure 4.3 Comparing radial profiles of selected mean species mass fractions (lines) with experimental measurements (symbols) at 50 mm, 150 mm and 250 mm above the nozzle exit plane.

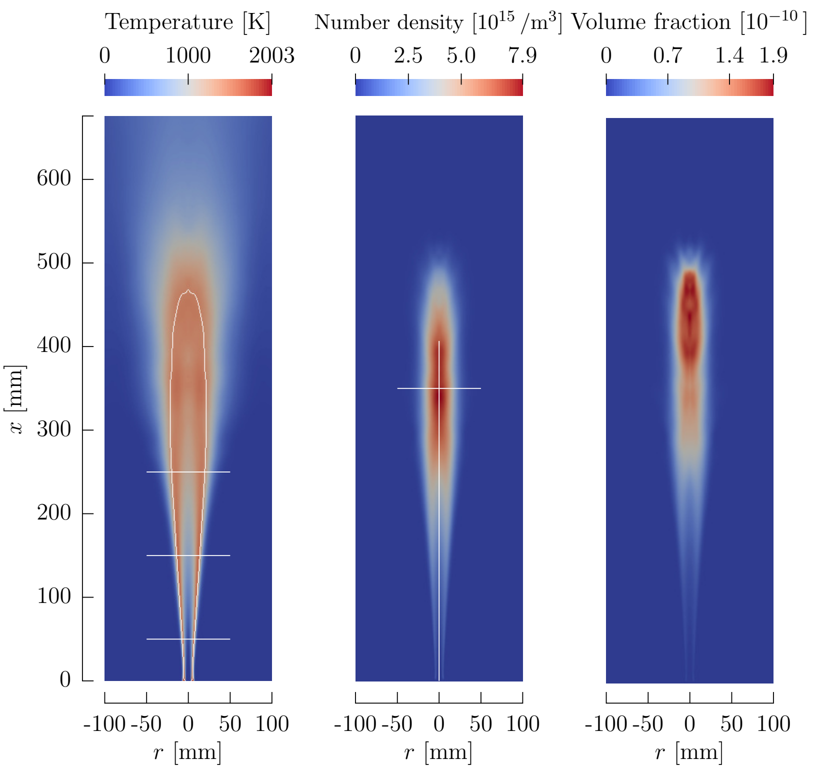


Figure 4.4 Contour plots of the mean temperature (left), soot number density (center) and soot volume fraction (right) computed from the LES-PBE-PDF model. The white contour in the left panel indicates stoichiometric mixture fraction values and the horizontal white lines show the near-field measurement stations at 50 mm, 150 mm and 250 mm above the nozzle. In the center panel, the white horizontal and vertical lines indicate the locations for which the instantaneous soot size distributions are shown in Figure 4.6.

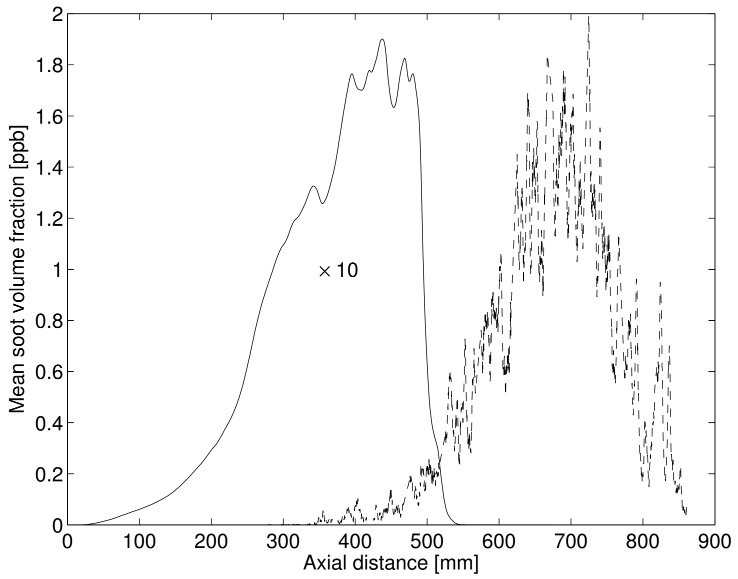


Figure 4.5 Comparing the mean soot volume fraction along the centerline (solid line) with the experimentally determined values (dashed line). Here, the LES-PBE-PDF predictions are scaled by a factor of 10.

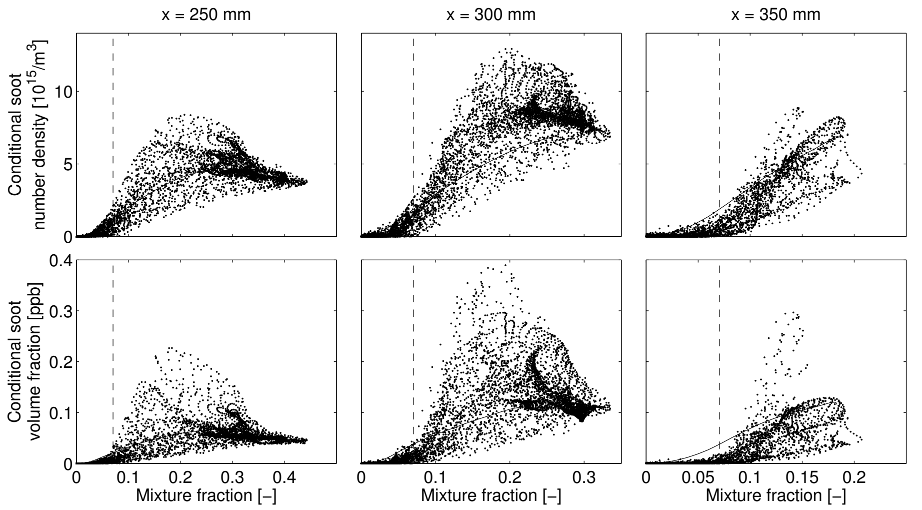
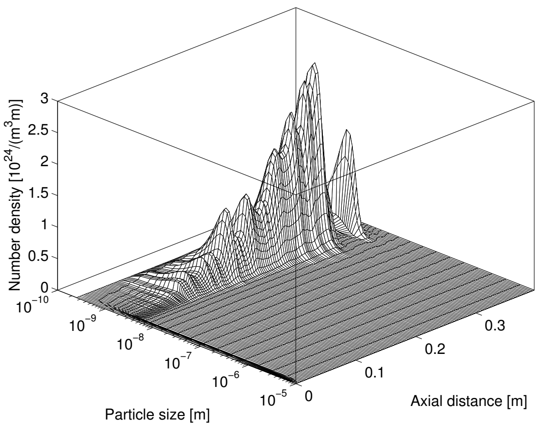
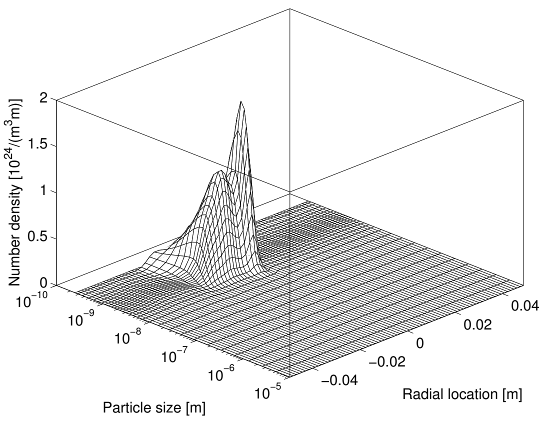


Figure 4.6 Instantaneous (scatter) and mean (solid lines) values of soot number density (top) and soot volume fraction (bottom) conditioned on mixture fraction at 250 mm, 300 mm and 350 mm above the nozzle. The vertical dashed lines indicate the stoichiometric mixture fraction of 0.0705.



(a) Along the flame centerline



(b) Across the flame at $x = 350$ mm

Figure 4.7 Instantaneous soot particle size distributions both along the flame centerline and across the flame at the axial distance of maximum mean soot number density. The spatial coordinate in the left panel runs along the vertical white line depicted in the central panel of Figure 4.4, while the spatial coordinate in the right panel corresponds to the horizontal white line.

Chapter 5

A methodology for the integration of stiff chemical kinetics on GPUs

5.1 Introduction

Graphics processing units (GPUs) were originally designed for quickly rendering images in special-purpose hardware. However, by now they have been developed into fully programmable compute devices for applications in which both data- and task-parallelism are nested. In recent times, researchers have hence begun to accelerate scientific programs by reimplementing apt subroutines for the execution on a GPU. In the context of non-reactive flows, a number of GPU-based fluid mechanics solvers have thus been developed and expertise with regard to efficient GPU programming techniques has been gained [141]. In this chapter, we focus on the related models for reactive flows and explore the capabilities of a GPU as an accelerator for including source terms and chemical kinetics.

Modern solution schemes for reactive flows often invoke an operator splitting technique in order to isolate the chemical kinetics model from diffusion/convection phenomena. Here, the main part of the computational effort is concentrated in the so-called reaction fractional step which requires the solution of one system of ordinary differential equations (ODEs) for each spatial grid point. Physically, these ODE systems describe the temporal evolution of the local composition and a calorific quantity like enthalpy in a constant pressure system. In view of the law of mass action, the species production and destruction rates are governed by the local concentrations of the reactants and products as well as the local thermodynamic state of the mixture and, thus, the ODE systems are spatially independent.

The problem of solving such a collection of independent ODE systems is sometimes termed embarrassingly parallel. Indeed, without the need for communication, each ODE system may be placed on a different processing thread and scheduled individually. On the other hand, each ODE system also features an inherent low level of parallelism since the reaction progress variables, the species production rates and the species' thermodynamic properties, for example, can be computed independently.

Although these two nested levels of parallelism correspond well with the concurrency exposed by modern GPU-computing APIs (Application Programming Interfaces),[10] initial efforts for accelerating the reaction fractional step concentrated on a single level of parallelism and outsourced the most time consuming operations such as the evaluation of the species production rates [195] or the finite difference approximation of the Jacobian matrix [21]. Following a similar approach as these two references, Shi et al. [191] invoked the GPU for evaluating the forward/reverse reaction rates and computing the LU decomposition of the Jacobian.

The main drawback of these approaches was the heavy data traffic between main memory and the GPU's memory which is limited by the bandwidth of the PCIe bus,[11] thus requiring large reaction mechanisms ($\gtrsim$ 250 species) for the GPU acceleration to take effect. On the plus side, conventional linear algebra subroutines could simply be replaced by their counterparts from GPU libraries such as CULA or CUBLAS [35, 78] and the software engineer did not need to be familiar with the GPU's hardware architecture or the API's programming model.

In order to reduce both the number of data transfers and the amount of data transferred between main memory and GPU memory, researchers have recently begun to reimplement the time stepping method and entire integration algorithms (including the step size adjustment scheme) for execution on the GPU. Here, the main focus lay on explicit integration algorithms of the Runge-Kutta type, while implicit integration schemes have received only little attention, see Table 5.1.

The first self-contained GPU implementation of a chemical kinetics integration scheme seems to be due to Niemeyer et al. [140] who implemented a 4th order explicit Runge-Kutta method in combination with the species rate expressions for a hydrogen reaction mechanism in CUDA C. Subsequently, Niemeyer and Sung [141] implemented an embedded 4th order Runge-Kutta-type method and a 2nd order stabilized Runge-Kutta-Chebyshev method in CUDA C and tested the implementation on a range of initial conditions sampled from solutions of a constant pressure homogeneous ignition problem. Similarly, Stone and Davis [196] included in their investigations an embedded 4th order Runge-Kutta-Fehlberg method which was also implemented using the CUDA API. As sample problem these authors considered a counter-flow linear eddy model.

Shi et al. [190], on the other hand, devised a hybrid explicit/implicit approach by pairing the implicit solver DVODE with a CUDA-based 2nd order accurate α-quasi steady state method. For this, each ODE system was assigned a degree of stiffness based upon the number of integration steps which the ODE system required in the previous global time

[10]Nvidia provides the proprietary CUDA API for developing general purpose software on Nvidia GPUs. OpenCL, on the other hand, is an API standard that targets different processor types and is jointly developed by the Khronos group, an industry consortium. Currently, an implementation of the OpenCL standard is available for most GPU brands (Nvidia, AMD, Intel).

[11]PCIe (Peripheral Component Interconnect Express) is a standardized design for connections between the main processor of a computer system and peripheral devices such as a GPU.

step. The ODE systems with a high number of integration steps were classified as stiff and integrated by DVODE on the CPU, while the less stiff ones were integrated using the explicit GPU solver.

In practice, however, explicit integration algorithms are rarely used on chemical kinetics ODE systems since the reactions may take place on time scales that differ by orders of magnitude, thus forcing prohibitively small time steps. Such stiffness proves difficult to estimate *a priori*; indeed, while explicit integrators may suffice in one situation (fuel, composition, temperature regime), they may fail in another.

In engineering applications, moreover, the flow is often turbulent and, therefore, the global time stepping is inherently linked with the size of the smallest resolved flow scales. A direct numerical simulation (DNS), for instance, attempts to resolve the turbulent fluctuations and, hence, time steps below the Kolmogorov time scale [32] are required, rendering explicit integration schemes for the reaction fractional step viable. In RANS and LES approaches, on the other hand, the fluctuating fields are subjected to a global averaging or filtering operation and the influence of the small scale fluctuations is modelled. Here, the time step sizes for the convection and diffusion fractional steps are typically on the order of 10^{-6} s. From the perspective of the mean or filtered flow fields, the reaction dynamics may thus appear severely stiff, warranting the application of implicit integration schemes.

In more general terms, stiffness can be encountered whenever the global time step size is controlled by a time scale that is bigger than the fastest reaction time. An example is the common case where the global time step size is governed by a CFL condition and, thus, by the characteristic convection/diffusion time scale. Here, the mixture may locally react on time scales which are much smaller than the global one, thus exhibiting severe stiffness.

By consequence, implicit integration schemes have shifted into the focus of GPU computing. Le et al. [97], for instance, included a first order accurate backward Euler scheme in a CUDA-based high order finite volume solver for the reactive Euler equations. As applications, these authors considered supersonic reactive flows such as detonations. Linford et al. [107], on the other hand, implemented an embedded 2nd order accurate 3-stage Rosenbrock method in CUDA C and assessed its performance in the context of chemical kinetics for the evolution of pollutants and trace gases in the earth's atmosphere. Rosenbrock algorithms are sometimes termed linearly implicit (or semi-implicit) methods and have been found to be very efficient for stiff ODE systems if the accuracy requirements are low [67, 181, 182].

To our awareness, a fully implicit, high order integration algorithm for chemical kinetics has only been implemented in a GPU API by Stone and Davis [196]. Specifically, these authors ported the 5th order accurate variable coefficient BDF-solver DVODE onto the CUDA framework and compared its performance to that of an explicit Runge-Kutta-Fehlberg method (see above) for the special case of a 19 species ethylene reaction mechanism.

In spite of the recent progress in the GPU acceleration of chemical kinetics integrators, certain questions which we believe are of paramount practical interest remain unanswered. Firstly, it is not yet clear whether the viability of a GPU-based implicit higher order integration scheme extends to common reaction mechanisms whose sizes range between 20 and about 200 species [141]. In this regard, it also remains to be investigated in which way the performance of such an implementation changes as the problem size is increased and as the time step size or the convergence tolerances are varied. Furthermore, we are not yet able to infer from the available GPU-CPU comparisons of implicit integration algorithms a recommendation on the most profitable acceleration strategies that do not inflict severe restrictions upon the mechanism size.

In view of these questions, we have carefully reimplemented the Fortran 77 program of the 5th order accurate implicit Runge-Kutta method Radau5 by Hairer and Wanner [67] in combination with the Chemkin III subroutines for evaluating the species production rates [89] in OpenCL C. Radau5 is suitable for our purposes since (i) it has been tested on a large number of reaction mechanisms within our group; (ii) it implements an A-stable method; and (iii) it is based upon a single step, multiple points integration algorithm for which the step size adjustment scheme is quite simple and, in particular, does not involve the interpolation of past solution vectors or additional right-hand-side evaluations.

As sample problem we consider a transient equilibrium scheme for the flamelet model based upon a uniform grid in mixture fraction space. Each chemical kinetics ODE system is thus associated with a different mixture fraction, reflecting the heterogeneous conditions in a real-life flow. The comparison between the GPU implementation and the original CPU version is based upon measurements of the runtime (and the data transfer time in the case of the GPU implementation) for an increasing number of ODE systems and mechanism sizes and for both a consumer-level and a high-end GPU/CPU. Here, the CPU version is equipped with an Open MPI parallelization which distributes the ODE systems across all threads available on the processor. In view of the cost-effectiveness, moreover, the number of ODE systems which can be solved on a given processor for a chosen reaction mechanism is determined per unit of time and per Pound Sterling invested into the device. Also, we quantify the influence of the time step size and of the convergence tolerances on the performance of the GPU implementation as compared to that of the CPU version.

This chapter is structured as follows: In Section 5.2, we briefly review the transient flamelet model and apply the method of fractional steps in order to isolate the chemical kinetics model from the diffusion in mixture fraction space. Subsequently, the class of implicit Runge-Kutta methods is formulated in Section 5.3 and the algorithmic structure of Radau5 is outlined. In Section 5.4, we present the architecture of a modern GPU from the perspective of the OpenCL execution and memory models. Section 5.5 then details aspects of the OpenCL reimplementation of Radau5. Among others, we discuss an asynchronous scheme for overlapping data transfers between main memory and the GPU's

Name	Algorithm	Implicit?	Order	Mechanisms (n_{sp}/n_r)	Maximum speedups	References
RK4	Runge-Kutta	No	4	9/38	$75/-\times$	[140]
RKCK	Runge-Kutta-Cash-Karp	No	4	9/38	$126/25\times$	[141]
RKC	Runge-Kutta-Chebyshev	No	2	13/27	$59/10\times$	[141]
				53/325	$69/13\times$	
				111/784	$-/18\times$	
RKF45	Runge-Kutta-Fehlberg	No	4	19/167	$20.2/10.7\times^*$	[196]
CHEMEQ2	α-Quasi-steady-state	No	2	39/131	$12.8/-\times$	[190]
				117/499	$13.2/-\times$	
–	Backward Euler	Yes	1	9/38	$\approx 42/-\times$	[97]
				36/308	$\approx 32/-\times$	
ROS3	3-stage Rosenbrock	Yes	2	61/156	$3.0/\mathbf{0.45}\times^\ddagger$	[107, 182]
DVODE	Variable step/order BDF	Yes	≤ 5	19/167	$\mathbf{7.3}/7.7\times^*$	[196]
Radau5	3-stage implicit Runge-Kutta (Radau IIA)	Yes	5	53/325	$\mathbf{4.8}/\mathbf{0.54}\times^\dagger$	This work

Table 5.1 GPU implementations of explicit/implicit integration schemes (double precision) for chemical kinetics ODE systems. Here, the speedups relate a one-thread GPU implementation to a single core/six cores CPU implementation, except when a *, † or ‡ is specified. In the case of *, the speedups are given for a one-block/one-thread GPU implementation relative to a single core CPU version, while † indicates a one-block GPU implementation that is compared with a single core/8 core CPU version (hyper-threading enabled). If a ‡ is specified, on the other hand, the speedup is computed for a one-thread GPU implementation relative to a single core/8 core CPU version (two 4 core CPUs). (For more details on the one-thread/one-block strategies we refer to Section 5.5.2.) Moreover, n_{sp} and n_r denote the numbers of species and reactions of the reaction mechanisms for which the speedups are given.

memory with multiple kernel[12] invocations and provide a careful analysis of the kernel's memory requirements. Runtime measurements are presented in Section 5.6, where the performance of the OpenCL-GPU implementation is compared with that of an MPI-CPU version on a per-processor basis. Finally, findings are summarized in Section 5.7 and future steps are set into the perspective of our conclusions.

5.2 Sample problem

As sample diffusion-reaction problem we consider a transient equilibrium scheme for the flamelet model

$$\frac{\partial \mathbf{Y}(z,t)}{\partial t} = \frac{\chi(z)}{2}\frac{\partial^2 \mathbf{Y}(z,t)}{\partial z^2} + \frac{\dot{\boldsymbol{\omega}}}{\rho}(\mathbf{Y}(z,t)), \tag{5.1}$$

$$\mathbf{Y}(z,0) = \mathbf{Y}_0(z), \tag{5.2}$$

$$\mathbf{Y}(0,t) = \mathbf{Y}_O, \tag{5.3}$$

$$\mathbf{Y}(1,t) = \mathbf{Y}_F, \tag{5.4}$$

where $z \in [0,1]$ denotes the mixture fraction, $t \geq 0$ represents time and the vector $\mathbf{Y}(z,t) = (Y_1,\ldots,Y_{n_s-1},T) \in \mathbb{R}^{n_s}$ contains the species mass fractions Y_i and temperature T. χ,

[12]In GPU terms, a kernel defines a function of which an instance executes for each processing thread launched on the GPU.

moreover, denotes the scalar dissipation rate which may be taken as a measure for the inhomogeneity of the convecting flow. It is often computed in terms of the strain rate $s > 0$ according to

$$\chi(z) = \frac{s}{\pi}\left(\exp(-2(\mathrm{erf}^{-1}(2z))^2)\right)^2. \tag{5.5}$$

The boundary conditions in Eqs. (5.3) and (5.4) are formulated in terms of the compositions $\mathbf{Y}_O$ and $\mathbf{Y}_F$ of the oxidizer and fuel, respectively. Constitutively, both reactant mixtures are assumed to behave as multicomponent ideal gases.

For the investigations in Section 5.6, we determined the initial conditions $\mathbf{Y}_0(z)$ in Eq. (5.2) from the chemical equilibrium composition and temperature which jointly minimize Gibbs' free energy in a constant enthalpy, constant pressure system.

The source term $\dot{\boldsymbol{\omega}}$ on the right-hand-side of Eq. (5.1) provides the link to the chemical kinetics model. Here, the species production rates are computed from the reaction progress variables q_j, $j = 1, \ldots, n_r$,

$$\dot{\omega}_i(\mathbf{Y}) = \sum_{j=1}^{n_r} \nu_{ij} q_j(\mathbf{Y}), \quad i = 1, \ldots, n_{sp}, \tag{5.6}$$

where $\boldsymbol{\nu} \in \mathbb{R}^{n_{sp} \times n_r}$ denotes the (sparse) matrix of stoichiometric coefficients and $n_{sp} = n_s - 1$ and n_r indicate the numbers of species and reactions, respectively. The reaction progress variables are expressive of the law of mass action and include Arrhenius-type expressions for the forward rate coefficients. While the reverse rate coefficients may also be computed from an Arrhenius-type expression, more than often the reverse Arrhenius parameters are not available. In this case, they can be determined from the equilibrium constant.

The final entry in $\dot{\boldsymbol{\omega}}$ corresponds to the source term in the calorific T-equation and is given by

$$\dot{\omega}_T(\mathbf{Y}) = -\frac{\sum_{i=1}^{n_{sp}} H_i(T)\dot{\omega}_i(\mathbf{Y})}{\bar{C}_p(\mathbf{Y})}, \tag{5.7}$$

where $H_i(T)$ denotes the molar enthalpy of species i at temperature T and $\bar{C}_p(\mathbf{Y})$ denotes the mixture's molar specific heat at constant pressure.

If a reaction requires the presence of inert species, then an enhanced equation similar to Eq. (5.6) applies which accounts for so-called third-body efficiencies. Furthermore, the reaction rates may be pressure-dependent in which case the Arrhenius expressions include pressure correction terms. These different reaction characteristics are accounted for by the Chemkin library, for instance. For conciseness, we omit further details on the complete chemical kinetics model and refer to the comprehensive Chemkin manual [89] instead.

By applying the method of fractional steps, Eq. (5.1) can be split into a passive diffusion problem ($\cdot^{(1)}$) and a pure reaction problem ($\cdot^{(2)}$) over a (sufficiently small) time interval

$[t_i, t_{i+1}]$

$$\frac{\partial \mathbf{Y}^{(1)}(z,t)}{\partial t} = \frac{\chi(z)}{2}\frac{\partial^2 \mathbf{Y}^{(1)}(z,t)}{\partial z^2}, \quad \mathbf{Y}^{(1)}(z,t_i) = \mathbf{Y}(z,t_i), \tag{5.8}$$

$$\frac{\partial \mathbf{Y}^{(2)}(z,t)}{\partial t} = \frac{\dot{\boldsymbol{\omega}}}{\rho}(\mathbf{Y}^{(2)}(z,t)), \qquad \mathbf{Y}^{(2)}(z,t_i) = \mathbf{Y}^{(1)}(z,t_{i+1}), \tag{5.9}$$

$$\mathbf{Y}(z,t_{i+1}) = \mathbf{Y}^{(2)}(z,t_{i+1}), \tag{5.10}$$

where $\mathbf{Y}(z,t_i)$ and $\mathbf{Y}(z,t_{i+1})$ indicate an (approximate) solution to Eq. (5.1) at the beginning and the end of a time step.

Eqs. (5.8) through (5.10) implement a first order approximation in time [161, 211]. Since the source terms $\dot{\omega}_i/\rho$ at a point z only depend upon the composition at this point, Eq. (5.9) can be solved for each point in mixture fraction space independently. In combination with a spatial discretization scheme encompassing $n+1 \geq 1$ intervals

$$[0,1] = \bigcup_{i=1}^{n+1} [z_{i-1}, z_i] \tag{5.11}$$

(for example, $z_i = i/(n+1)$), Eq. (5.9) leads to

$$\frac{d\mathbf{Y}_i^{(2)}(t)}{dt} = \frac{\dot{\boldsymbol{\omega}}}{\rho}(\mathbf{Y}_i^{(2)}(t)) \tag{5.12}$$

for $\mathbf{Y}_i^{(2)}(t) = \mathbf{Y}^{(2)}(z_i,t)$ and $i = 1, \ldots, n$ (excluding the boundaries $i = 0, n+1$ of mixture fraction space). This equation forms a so-called embarrassingly parallel problem. It is the main point of attack for most modern parallel implementations of the reaction fractional step and, indeed, justifies the leading order time approximation.

Although we have considered the flamelet Eq. (5.1) in the above, the reaction fractional step in Eq. (5.12) is not particular to this model, but can be constructed similarly for any scalar or vector-valued diffusion-convection-reaction transport problem. Specifically, the OpenCL-GPU solver for the reaction fractional step which we present is applicable to turbulent flow problems and can be readily incorporated into an existing RANS or LES reactive flow solver. In the present chapter, focus is laid on the flamelet model because it constitutes a lightweight framework for generating a range of initial conditions with which the GPU-based integration scheme may be tested.

For future reference, the numbers characterizing the problem and reaction mechanism sizes (such as n_{sp}, n_s and n_r) are summarized in Table 5.2.

Variable	Explanation
n_{sp}	# Species
$n_s = n_{sp} + 1$	# Scalars
n_r	# Reactions
n	# ODE systems
n_p	# ODE systems per t/p-cycle
n_{tb}	# Reactions involving third body efficiencies
n_{fo}	# Reactions including pressure-dependence (fall-off reactions)
$m_{sp} = 6, 7$	Maximum # species per reaction
$m_{tb} = 10$	Maximum # third bodies per reaction
$m_{fo} = 10$	Maximum # fall-off parameters per reaction
l	OpenCL work group size (see Eq. (5.24))

Table 5.2 Numbers characterizing the spatial discretization and the reaction mechanisms.

5.3 Radau5 II

By integrating Eq. (5.12) over the time interval $[t_j, t_{j+1}]$ we obtain

$$\mathbf{Y}(t_{j+1}) = \mathbf{Y}(t_j) + \int_{t_j}^{t_{j+1}} \mathbf{f}(t, \mathbf{Y}(t))\, dt, \quad \mathbf{Y}(t_j) = \mathbf{Y}^j, \tag{5.13}$$

where $\mathbf{Y} \equiv \mathbf{Y}_i^{(2)}$ and $\mathbf{f}(\cdot, \mathbf{Y}(\cdot)) \equiv \dot{\boldsymbol{\omega}}/\rho(\mathbf{Y}(\cdot))$ have been set and $\mathbf{Y}^j$ denotes the vector of initial conditions.

In a Runge-Kutta method, the main idea for advancing the (exact) solution $\mathbf{Y}(t)$ of Eq. (5.13) by one time step $h = t_{j+1} - t_j$ consists in approximating the integral on the right-hand-side by a Gauss-type quadrature formula

$$\begin{aligned}\int_{t_j}^{t_{j+1}} \mathbf{f}(t, \mathbf{Y}(t))\, dt &= h \int_0^1 \mathbf{f}(t_j + ht', \mathbf{Y}(t_j + ht'))\, dt' \\ &\approx h \sum_{l=1}^{m} \gamma_l \mathbf{f}(t_j + \alpha_l h, \mathbf{Y}(t_j + \alpha_l h)),\end{aligned} \tag{5.14}$$

where

$$t' = \frac{t - t_j}{h} \tag{5.15}$$

and $\alpha_l \in [0, 1]$. Since, here, $\mathbf{f}(\cdot, \mathbf{Y}(\cdot))$ is evaluated at times $t_j + \alpha_l h \in [t_j, t_{j+1}]$, Runge-Kutta methods are classified as single step, multiple points integration schemes.

For the Radau II integration rule, the weights γ_l and mid-points α_l in Eq. (5.14) are chosen such that the quadrature formula includes the right boundary ($\alpha_m = 1$) and is exact for polynomials of order $2m - 2$ ($m \geq 1$),

$$\mathbf{f}(t_j + ht', \mathbf{Y}(t_j + ht')) = \sum_{k=0}^{2m-2} \mathbf{g}_k {t'}^k. \tag{5.16}$$

Here, $t' \in [0,1]$ and $\mathbf{g}_k \in \mathbb{R}^n$ are $2m-1$ linearly independent vectors.

Combining Eqs. (5.13) and (5.14) leads to

$$\mathbf{Y}(t_j + 1h) \approx \mathbf{Y}^j + h \sum_{l=1}^{m} \gamma_l \mathbf{f}(t_j + \alpha_l h, \mathbf{Y}(t_j + \alpha_l h)). \tag{5.17}$$

If t_{j+1} in Eq. (5.14) is replaced by $\tilde{t}_{j+1} \in [t_j, t_{j+1}]$, then the upper bound 1 of the integral on the right-hand-side of Eq. (5.14) becomes $(\tilde{t}_{j+1} - t_j)/h = \tilde{\alpha}$. By keeping the mid-points α_l fixed, we may deduce from this substitution a slightly more general (but less accurate) version of Eq. (5.17),

$$\mathbf{Y}(t_j + \tilde{\alpha} h) \approx \mathbf{Y}^j + h \sum_{l=1}^{m} \tilde{\gamma}_l \mathbf{f}(t_j + \alpha_l h, \mathbf{Y}(t_j + \alpha_l h)). \tag{5.18}$$

For $\tilde{\alpha} = \alpha_1, \ldots, \alpha_{m-1}$ and $\alpha_m = 1$, Eq. (5.18) establishes a system of nm non-linear equations for the m unknown auxiliary points

$$\mathbf{k}_l = \mathbf{Y}(t_j + \alpha_l h), \quad l = 1, \ldots, m. \tag{5.19}$$

By substituting Eq. (5.19) into Eqs. (5.17) and (5.18) and taking $\tilde{\gamma}_l = \beta_{il}$ for a given $\tilde{\alpha} = \alpha_i$, we now obtain the following numerical scheme

$$\mathbf{k}_l = \mathbf{Y}^j + h \sum_{k=1}^{m} \beta_{lk} \mathbf{f}(t_j + \alpha_k h, \mathbf{k}_k), \quad l = 1, \ldots, m, \tag{5.20}$$

$$\mathbf{Y}^{j+1} = \mathbf{Y}^j + h \sum_{k=1}^{m} \gamma_k \mathbf{f}(t_j + \alpha_k h, \mathbf{k}_k). \tag{5.21}$$

Here, the $\approx$-signs have been replaced by $=$-signs, while the (exact) solution at t_{j+1} has been substituted by the approximation $\mathbf{Y}^{j+1} \approx \mathbf{Y}(t^{j+1})$.

If the weights γ_k and β_{lk}, $k, l = 1, \ldots, m$, and the mid-points α_j, $j = 1, \ldots, m-1$, in Eqs. (5.20) and (5.21) satisfy

$$\sum_{l=1}^{m} \gamma_l \alpha_l^{k-1} = \frac{1}{k}, \quad k = 1, \ldots, 2m-1, \tag{5.22}$$

$$\sum_{l=1}^{m} \beta_{il} \alpha_l^{k-1} = \frac{\alpha_i^k}{k}, \quad i, k = 1, \ldots, m, \tag{5.23}$$

then the Radau II approximations of the integrals in Eqs. (5.14) and (5.18) are exact for polynomials (see Eq. (5.16)) of order $2m-2$ and m, respectively. In this case, the (implicit) Runge-Kutta method is of order $2m-1$ [67, Theorem 5.3].

As basis for a GPU reimplementation we chose the m = 3-stage/5th order implicit Runge-Kutta algorithm Radau5 which has been developed by Hairer and Wanner [67].

Since these authors provide a thorough discussion of the implementational aspects (solving the non-linear system in Eq. (5.20)/step size adaptation/error control), we refer the interested reader to their monograph. However, an important aspect to note at this point is that the Newton-type solution scheme for Eq. (5.20) may be subjected to a special transformation and, in this way, the linear system matrix of size $3n$ split into a real and a complex linear system of size n.

Figures C.1 and C.2 in Appendix C.1 depict a flow chart of the Radau5 algorithm including the step size adjustment scheme.

5.4 GPU computing

In the present section, we review conceptual differences in the architectures of a CPU and a modern GPU (Section 5.4.1) and describe both the execution and memory models of a GPU from the viewpoint of the OpenCL standard (Section 5.4.2).[13] Moreover, the portability of OpenCL applications in relation to device-specific optimizations is addressed (Section 5.4.3). Finally, the notion of an SIMD unit is introduced (Section 5.4.4) as a basis for the discussion of GPU programming techniques in Section 5.5.

For a more comprehensive introduction into the architecture of a GPU and aspects of GPU computing which lie beyond the presentation in this section, we refer to the textbooks by Hennessy and Patterson [73, 150].

5.4.1 Characteristics of CPU and GPU architectures

Originally, GPUs have been developed for graphics applications which encompassed an instruction sequence that was applied to a large number of data sets. Since these data sets (vertices, fragments) could be manipulated independently of each other, the instructions were organized in a graphics pipeline in which different stages of the rendering process operated concurrently on distinct data sets and each stage could process several data sets simultaneously. The individual stages could be configured, but they were not programmable in a way that would make the GPU accessible to non-graphics applications [148].

By now, GPU vendors have lifted this restriction and GPU's have evolved into fully programmable processors, targeting massively data-parallel applications with a simple control flow. Here, a large number of processing elements (PEs) are orchestrated by a few instruction units which are reminiscent of the stages in the original graphics pipeline. A CPU, on the other hand, encompasses only few processing elements which operate independently and mainly targets task-parallel applications in which each task may contain an elaborate control flow.

[13]Although we adopt the perspective of the OpenCL API and its terminology in this work, this does not present a limitation to the validity of the concepts or programming techniques presented.

A major difference in the operation of both GPUs and CPUs is the way in which memory access latency is hidden. The CPU encompasses, to this end, an on-chip hierarchy of fast, small memory buffers (caches) through which data and instructions proceed to the processing elements. Modern GPUs, by contrast, instantly switch from processing instructions that are waiting for memory accesses to complete to invoking instructions which are ready to execute. This requires a large pool of independent (but possibly identical) instruction sequences between which the processing elements can arbitrate.

5.4.2 GPU architecture and the OpenCL standard

OpenCL is shorthand for Open Computing Language, a standardized API for developing software on heterogeneous systems that is administered by the Khronos Group. In recent times an OpenCL implementation has become available for most processors like Intel CPUs or Nvidia and AMD GPUs. The main idea which propelled the development of OpenCL was to be able to devise portable software that can explore the system on which it executes and invoke the processors that are available in the system for specific computational tasks. The scope of OpenCL is thus very modern, reflecting the heterogeneity of both personal computers and high-performance workstations.

An OpenCL application consists of a host program which dispatches both memory buffers and function evaluations on these buffers to a device. Functions which execute on the device are termed kernels. These are written in OpenCL C, an API-specific language extension to the C99 standard. Typically the host code is compiled for execution on a CPU. In this way, the OpenCL application can be launched in a manner similar to standard applications or the host code may be incorporated into an existing program. The kernel functions, by contrast, are compiled at runtime by a specific device compiler which is contacted through the OpenCL API.

From the perspective of a software engineer, an OpenCL application (i) first explores the platform on which it executes, (ii) chooses appropriate devices, (iii) builds the kernel source code for the selected devices, (iv) allocates memory buffers on the devices and (v) finally enqueues the kernels for execution. Typically, steps (i) through (iv) can be accomplished during an initialization phase. The communication (and synchronization) between the host and the device is channelled through a command queue which may execute in-order or out-of-order. In the latter case, the order of the commands can be structured using event objects which constitute a top-level synchronization mechanism.

As an aid to the reader we include a brief outline of the OpenCL memory and execution model. Further details can be found in the OpenCL specification [132] or the OpenCL Programming Guide [131].

The OpenCL execution model employs a rectilinear one, two or three-dimensional integer index grid as a basis for labelling execution paths. For each vertex in this index grid an instance of a kernel is executed. This instance is called a work item which executes on a processing element. Work items, moreover, can be clustered into work groups which execute on compute units. Typically, a GPU consists of several compute units (about 6 to 14) each of which comprises a large number of processing elements (32 on Nvidia GPUs), where, conceptually, a processing element may be compared with a single CPU core, see Figure 5.1. One difference between both is that a processing element on the GPU executes instructions in orchestration with the other processing elements within the same compute unit, while the CPU cores operate independently. Moreover, the processing elements cannot take advantage of a complete cache hierarchy (L3 through L1i/d caches in Figure 5.1) in the way that a CPU core does.

Work groups play an important role in the management of OpenCL local memory. While all work items can access global memory buffers, local memory is private to a work group and can only be read from or written to by the work items from this work group. Registers, moreover, are classified as private memory storage which cannot be shared among different work items. The last address space qualifier designates constant or read-only memory for which dedicated caches may exist on a device.

In the context of GPUs, global memory corresponds to DRAM (Dynamic Random Access Memory) which is a large off-chip memory. Local memory,[14] on the other hand, is located on-chip whence memory access latencies are much smaller than for global memory. On Nvidia GPUs, OpenCL local memory coincides with CUDA shared memory.

Finally, the work items inside a work group can be synchronized using barriers either on the local or the global memory level. By contrast, OpenCL does not supply a synchronization mechanism between work groups. Thus, communication between work items that belong to different work groups is not supported.

5.4.3 Portability

Although the OpenCL standard promotes software portability across a wide range of different computing devices, the hardware independence of an OpenCL program does not imply that one OpenCL kernel performs well on all OpenCL-enabled devices. Quite on the contrary, optimizations for improving the performance of an OpenCL kernel on a specific OpenCL device often require an understanding of the device's architecture and, in particular, of how this architecture maps onto the OpenCL execution and memory models. While some of these optimizations can be expressed in terms of kernel parameters (such as the number of work groups or the work group size) or memory space qualifiers (global vs. constant memory buffers, for example), other optimization techniques transcend through the entire kernel implementation and are not easily adapted for a different device. As an

[14]Although the names are identical, the OpenCL local memory space is different from CUDA's local memory which maps onto a portion of DRAM and, hence, is located off-chip. CUDA employs this memory space to accommodate variables or arrays that have been spilled out of the registers of a specific work item, but there is no OpenCL equivalent. In the following, we therefore do not reference CUDA local memory.

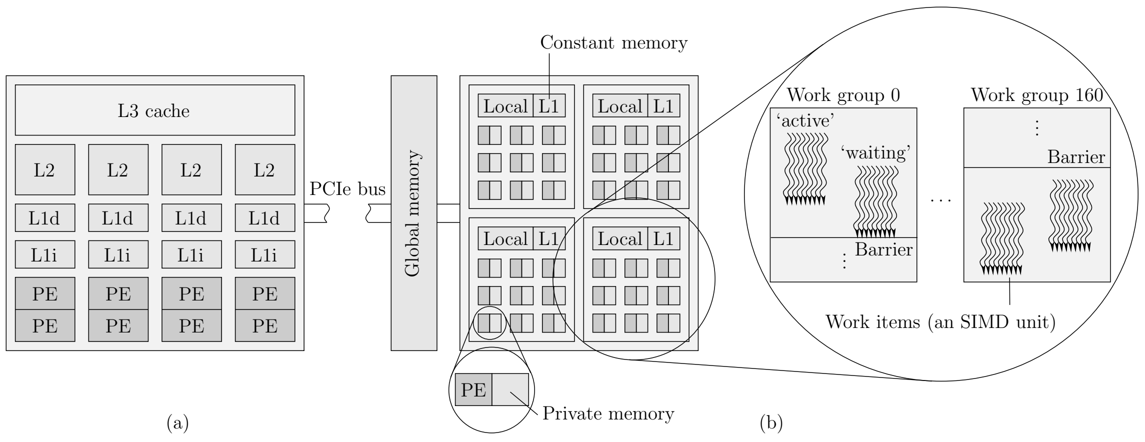


Figure 5.1 Architectures of a modern CPU (a) and a GPU (b). Here, the terminology refers to the OpenCL execution and memory models (except for the cache labels L1i/d through L3 and the term 'SIMD unit' which indicates a collection of work items that are being issued instructions simultaneously).

example, we may consider the memory access patterns which are reflected in the structure of parallel loops inside the kernel. The block access that is common for a shared memory CPU (each work item processes a consecutive block-subset of a data array) has a negative impact on the performance of the same kernel on a GPU since it inhibits the coalescence of global memory accesses, see Section 5.5.1. Thus, although an OpenCL kernel is portable across different OpenCL devices, its inherent low-level optimizations may not carry over to a different OpenCL device.

Currently, the main advantage of developing software using the OpenCL API is that OpenCL (unlike Nvidia's CUDA framework) is a non-proprietary industry standard for which most GPU vendors supply an implementation. Therefore and since modern personal computers encompass both a CPU and a GPU, the OpenCL chemical kinetics solver which we have developed can be invoked on any such system regardless of the GPU's brand.

5.4.4 SIMD architecture

The OpenCL execution model which has been reviewed in Section 5.4.2 implements both task- and data-parallelism on the level of work items. Since work items can be clustered into work groups, there is an optional superior level of task-parallelism. In the architecture of a GPU, by contrast, the capabilities for task- and data-parallelism are realized hierarchically such that a task is required to be inherent data-parallel and task-parallelism involves several such data-parallel tasks. Although this is not explicated by the OpenCL model, the OpenCL abstractions are transparent for the implications which the inherent data-parallelism of a GPU has on how the work items execute on processing elements. In the present section, we analyse these implications and deduce recommendations on the control flow within an OpenCL kernel.

As indicated in Section 5.4.2, a compute unit encompasses a number of processing elements. On GPUs, these processing elements are grouped into batches each of which is controlled by a single instruction unit. This unit issues instructions sequentially such that all processing elements within a batch are serviced the same instruction, possibly with different operands. The work items which are processed by a batch of processing elements are termed an SIMD (Single Instruction stream, Multiple Data streams) unit (or SIMD thread). In Nvidia CUDA terminology, an SIMD unit is called a warp and consists of 32 work items, while on AMD GPUs the SIMD unit is referred to as a wavefront and encompasses 64 work items.

If the work items within an SIMD unit follow diverging instruction paths (for example, on account of an if-else or switch statement), then the instruction unit proceeds through these instruction paths in a sequential manner. Here, the processing elements whose work items do not fulfill the condition for the current instruction path are masked off and remain idle until the instruction unit issues the instructions for their instruction path.

This implies that branch conditions for the work items within an SIMD unit lead to a serialization of the control flow. In other words, task-parallelism within a work group is supported only among SIMD units, but serialized within an SIMD unit. For high performance of an OpenCL kernel on a GPU, it is thus recommended to order the work items such that conditionals, if possible, evaluate identically for all work items within an SIMD unit and that the work group size equals a multiple of the number of SIMD units in order to avoid incomplete SIMD units. In keeping with the terminology used by other authors, we refer to the event that work items proceed along different instruction paths as thread divergence.

SIMD units also play an important role in the latency hiding strategy of a GPU. If the work items within an SIMD unit are waiting on a synchronization barrier or for memory accesses to complete, then the SIMD unit scheduler swaps the current SIMD unit for another SIMD unit in which the work items are ready to execute. In this way, the batch of processing elements on which the first SIMD unit was stalled remains busy.

A GPU thus implements a single level of data-parallelism which is nested inside two levels of task-parallelism: Multiple SIMD units (data-parallelism) execute on a batch of processing elements in an interleaved fashion (task-parallelism), while SIMD units which reside on different batches are processed simultaneously (task-parallelism).

5.5 GPU parallelization strategy

In the present section, we address the implementation in OpenCL of a GPU solver for the reaction fractional step based upon the implicit integration algorithm Radau5. In this regard, we first review specific programming techniques which exploit the architecture of a modern GPU (Section 5.5.1). In view of these techniques, strategies for mapping the reaction fractional step onto the OpenCL execution model are then presented and a scheme for sorting the chemical reactions in order to mitigate thread divergence is proposed (Section 5.5.2). Subsequently, the memory layout of the OpenCL implementation is analysed (Section 5.5.3) and an efficient implementation of the standard LU decomposition algorithm is detailed (Section 5.5.4). Moreover, we present a cyclic scheme for splitting the reaction fractional step across multiple kernel invocations which overlap with data transfers between the host CPU and the GPU (Section 5.5.5). Finally, the potential use of other implicit integration algorithms is discussed and it is shown how the concepts and OpenCL functions developed as part of the current implementation may be included in the implementation of an integration scheme that is different from Radau5 (Section 5.5.6).

5.5.1 Techniques for improving performance on a GPU

Since in this work an OpenCL kernel is designed for execution on a GPU, special programming techniques which take into account the architecture of a GPU can be applied in order to increase the kernel's performance. These strategies are in fact much different from their CPU counterparts where high-level, single-thread optimizations mainly concentrate on cache lines and instruction vectorization [80, 81]:

Coalesced memory accesses. If all work items in a work group access adjacent global memory addresses, then these accesses can be coalesced into a single (or a few) memory transaction(s). Since global memory accesses incur one of the highest latencies on a GPU, an apt data storage scheme and a compatible orchestration of the work items' read/write instructions may result in significant performance gains [141, Table 1]. On Nvidia GPUs, memory coalescing is performed on the level of a half warp.

Avoiding thread divergence. If two or more work items in an SIMD unit follow different instruction paths, then the instruction unit proceeds through these diverging instruction paths in a sequential manner. This leaves work items idle while their instruction paths have already or not yet been processed.

Using local memory. OpenCL local memory can be considered as a cache which the kernel function manages explicitly. A common programming pattern which exploits this cache consists in copying data from global memory to local memory (coalescing read instructions), operating on the data in local memory per work group and, finally, writing the results back to global memory in a coalesced fashion. Such a pattern embraces, for example, the implementation of a parallel reduction operation which has been devised in Reference [147]. On Nvidia GPUs, OpenCL local memory is organized in so-called banks which can service data to different work items simultaneously, but may serialize accesses to the same bank.

Avoiding register spilling. If a work item operates on large arrays or structures in private memory, then the OpenCL compiler may decide to expel these variables to OpenCL local or global memory, thus incurring a severe performance penalty. In practice, however, it is often very difficult to determine whether or when in particular this happens.

Occupancy. During the execution of a kernel, several work groups may reside on the same compute unit. Thus, whenever an SIMD unit is idle or waits for memory accesses to complete, the SIMD unit scheduler can switch to another SIMD unit which is ready to execute, possibly from a different work group. However, since local memory is particular to a compute unit and each work group's local memory maps to one portion of the compute unit's local memory, it may happen that the work groups allocate more local memory than there would be available if the maximum number of resident work groups per compute unit were present. In this case, the number of resident work groups is limited by the GPU's resources. Similarly, the registers on a compute unit are shared among work groups.

Host to device memory transfers. Before a kernel can be scheduled for execution on a GPU, the data on which the kernel operates needs to be transferred from the host CPU to the memory buffers that have been allocated on the GPU. Similarly, once the kernel terminates, the results of the computation are copied back to the host CPU's memory. The rate at which these data transfers happen is limited by the bandwidth of the PCIe bus that connects the host CPU and the GPU, and is often much smaller (by about an order of magnitude) than the bandwidths of GPU-internal global memory read/write instructions. Furthermore, there is an overhead associated with submitting a host to device memory transfer instruction to the OpenCL runtime. These invocation overheads may add up to a significant amount of time if many data transfers are scheduled repeatedly.

In order to mitigate the time consumed by memory transfers and invocation overheads, it is recommended that the overall computational task (the solution of the reaction fractional step) be split between the host CPU and the GPU such that the interface requires only a few small data transfers. Sometimes, in this regard, it is possible to store distinct data arrays in the same memory buffer and, hence, avoid repeated invocation overheads. Alternatively, if the computational scheme permits, the data transfers for the next kernel invocation can be overlapped with the execution of the current kernel. This leads to the overlapping compute-copy-compute cycles which we examine in detail in Section 5.5.5.

5.5.2 Mapping the reaction fractional step onto the GPU's execution model

The reaction fractional step which we formulated in Section 5.2, Eq. (5.12), encompasses two nested levels of concurrency: On the one hand, the ODE systems associated with the spatial grid points can be solved independently of each other. On the other hand, the innermost loops which appear both within the integration algorithm (vector updates, LU decompositions, forward/backward substitutions) and the evaluation of the chemical kinetics source terms (evaluating the species' thermodynamic properties, computing the reaction progress variables, Eq. (5.6)) involve independent iterations. In a computer implementation, these independent iterations translate into concurrent data streams to which a sequence of instructions is applied. If a loop contains conditionals, then the instruction sequence is split into branches which service subsets of data streams concurrently.

In a similar fashion, the OpenCL execution model also embraces two levels of concurrency: On the upper level, work groups operate independently, while, on the lower level, the work items within a work group collaborate. In this collaboration, the work items execute instructions concurrently, but can communicate by exchanging data through OpenCL global and local memory buffers and waiting at barrier synchronization points. In view of the reaction fractional step, the task for solving an ODE system can thus be mapped onto a work group such that the work items within this work group jointly integrate the ODE system over a global time step $t \in [t_j, t_{j+1}]$ (Eqs. (5.20) through (5.23)) and many work groups cover all ODE systems. This implementation strategy has first been advocated by Stone and Davis [196] who termed it the 'one-block' approach.

An alternative strategy is the so-called 'one-thread' approach in which one ODE-system is assigned to each work item. The main drawback, here, is that thread divergence is incurred whenever work items in the same work group disagree on the level of the step-size adjustment scheme and, therefore, take different numbers of step size refinements. The amount of thread divergence which thus occurs was quantified by Niemeyer and Sung [141] and Stone and Davis [196] who demonstrated a significant increase in performance when the work items in a work group operated on ODE systems which corresponded to adjacent grid points and, therefore, featured similar initial conditions and an identical step size adjustment path. Linford et al. [107], on the other hand, mitigated the impact of thread divergence by moving the top level conditionals within the integration scheme (time loop, integration step loop, error control) back onto the CPU host thread and invoking separate CUDA kernels for vector/matrix operations and right-hand-side evaluations. Despite the repeated memory transfers (time, step sizes) and kernel invocation overheads, this yielded a speed up of 1.56 (double precision) relative to the single-kernel implementation for an atmospheric chemical kinetics problem.

A second shortcoming of the one-thread approach is related to the usage of OpenCL local memory. Since the local memory buffer accessible to a work group is shared among a large number of work items and local memory is limited in size, only a small portion of local memory is available for the integration of one ODE system. However, in GPU programming it is common to store arrays whose size is $\gtrsim n_s$ in local memory in order to accelerate scattered or repeated memory accesses. For the one-thread approach, this implies that one work group requires a local memory buffer of size ln_s which may be incompatible with the maximum allocatable local memory buffer size or may impact the compute unit's occupancy. Le et al. [97] discussed this issue in more detail and quantified limits on the reaction mechanism size for the case where the array that one work item stores in local memory is of the order of n_s or larger. On the plus side, implementing the one-thread approach does not require any change in the serial implementation of the integration scheme or the chemical kinetics source term evaluation.

Among the existing GPU implementations of chemical kinetics integration schemes (Table 5.1), the one-thread approach is the most common. A direct comparison with the one-block approach has been presented by Stone and Davis [196] who considered both an explicit 4th order Runge-Kutta-Fehlberg scheme and the implicit 5th order accurate DVODE algorithm in the context of a 19 species/167 reactions mechanism. For both algorithms, the one-block approach proved to be superior if the number of ODE systems n was smaller than about 10^4, while the one-thread approach produced a higher maximum speedup relative to a single thread DVODE-CPU implementation. The difference in the maximum speedups between the one-thread and the one-block approach was significant for the explicit integration scheme, but only marginal for the implicit one.

In view of the shortcomings of the one-thread approach, we adopted the one-block strategy in this work. Specifically, the work group size l is related to the mechanism size n_s,

$$l = 32 \left\{ \text{floor}\left(\frac{n_s}{32}\right) + 1 \right\}, \tag{5.24}$$

where 32 denotes the warp size on Nvidia GPUs and floor$(\cdot)$ returns the biggest integer that is smaller than or equal to the argument value.

Within the scope of the one-block approach all work items within a work group jointly operate on one ODE system and, hence, proceed through the time stepping scheme (Figures C.1 and C.2 in Appendix C.1) collectively. Here, the parallelization takes place on the level of the innermost loops whose numbers of iterations are of the order of the work group size. If the iterations of these loops are independent, then they are assigned to the individual work items in a strided way. As an example, we consider the computation of the reaction progress variables which encompasses n_r independent iterations. In this case, the first work item computes the first, the $(l+1)$th, the $(2l+1)$th, …, the $(K_1 l+1)$th reaction progress variables, where K_1 is the biggest integer such that $(K_1 l + 1) \leq n_r$. Similarly, the second work item computes the second, the $(l+2)$th, …, reaction progress variables. If the reaction progress variables are written in-order to a global memory array, then the memory accesses are fully coalesced since all work items access consecutive memory addresses $(0, \ldots, l-1;\ l, \ldots, 2l-1;\ \ldots)$ during a single iteration k ($k = 1, \ldots, K_j$ for work item j).

This fine-grained parallelism appears throughout the OpenCL kernel and, in particular, in the following computations:

(1) Evaluation of the species' thermodynamic properties (n_s)

(2) Evaluation of the reaction progress variables (n_r)

(3) Sparse matrix-vector multiplication to compute the species production/destruction rates, see Eq. (5.6) (row-wise parallel reduction, $2n_s$)

(4) LU-decompositions (L-update/pivot search, $1, \ldots, n_s$; elimination submatrix update, $1, \ldots, n_s^2$)

(5) Forward/backward substitutions ($1, \ldots, n_s$)

(6) Computing the density or $\dot{\omega}_T$, see Eq. (5.7) (parallel reduction, $1, \ldots, n_s$)

Here, the numbers in the trailing brackets indicate the number of iterations in the corresponding loops.

In the current one-block implementation, thread divergence may occur on account of work item conditionals inside a loop that is parallelized across all work items. Thus, if the loop body (or part of it) differs for work items of the same SIMD unit (for example, the first 32 work items on an Nvidia GPU), then the instruction unit proceeds through these differing instruction paths sequentially. At present, this effect only arises within the loop for the evaluation of the species production rates, where reactions may feature different characteristics. In the context of the Chemkin library, for example, a reaction may involve third body efficiencies (or not); its reaction rates may be pressure dependent (or not) and the approximation scheme for the pressure dependence in the fall-off region between the low-pressure and the high-pressure limit may differ from one reaction to another. Sometimes, moreover, Arrhenius parameters for the reverse reaction rates are available, while usually the reverse reaction rates are computed from the equilibrium constant. If the reaction is irreversible, on the other hand, then its reverse reaction rate vanishes.

Since thread divergence is limited to the level of an SIMD unit, we are able to reduce its negative performance impact by sorting the reactions according to their characteristics. To illustrate this point, we consider the three properties of the previous paragraph (third bodies, pressure dependence, reversibility), where each property may attain a number of values. Formally, the three properties are labelled p_1, p_2 and p_3, respectively, and their values are numbered consecutively beginning at 0. If a reaction is pressure dependent ($p_2 > 0$), for example, the fall-off region may be approximated using the Lindemann form ($p_2 = 1$), the SRI form ($p_2 = 2$) or the TROE scheme with 6 ($p_2 = 3$) or 7 ($p_2 = 4$) parameters. Similarly, the characteristics of the third body and reversibility properties are associated with p_1 and p_3, say $p_1 \in \{0, 1\}$ and $p_3 \in \{0, 1, 2\}$.

As a specific example, we consider a reaction mechanism with four reactions and, based upon the property indexing scheme, establish a reaction property matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \\ \mathbf{r}_4^T \end{pmatrix} = \begin{pmatrix} 0 & 3 & 0 \\ 1 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 3 & 2 \end{pmatrix}. \tag{5.25}$$

Here, each row $\mathbf{r}_i^T$, $i = 1, \ldots, 4$, is associated with a reaction, while the columns correspond to the properties p_1 through p_3. By sorting the reaction matrix column-wise, we obtain

$$\tilde{\mathbf{P}} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_3^T \\ \mathbf{r}_4^T \\ \mathbf{r}_2^T \end{pmatrix} = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 3 & 2 \\ 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \end{pmatrix}, \tag{5.26}$$

where $\mathbf{x}$ denotes the permutation vector. If, in this example, the SIMD unit size was two, then the number of divergence roots in the first SIMD unit would be reduced from two to one, while in the second SIMD unit it would increase from one to three. In practice, however, there exist many reactions with identical properties and we found that grouping these together by the above sorting scheme proves beneficial, see Section 5.6.3.

5.5.3 Memory layout

During the reimplementation of Radau5 in OpenCL C we have taken care in order to ensure that global memory accesses are coalesced. Scattered memory accesses, on the other hand, are deferred to temporary arrays in OpenCL local memory. In this regard, Table 5.3 summarizes the memory requirements in bytes (B) of the OpenCL kernel.

In a preliminary study, we tested storing the integration control variables (8 integers, 12 doubles) in private memory for register-speed access, but found this to induce immense redundancy since each work item keeps private copies of values that all work items share. Consequently, the control variables have been moved to local memory and are being updated by the first work item in each work group.

Furthermore, each work group allocates two double arrays of length n_s and one integer array of length l in local memory. This local memory layout is motivated by two reasons. First, since Radau5 requires solving a complex linear system in each Newton iteration, one column of this linear system or the complex right hand side vector (which is being successively overwritten by the complex solution vector in the forward/backward substitution algorithm) can be stored in local memory. Also, the two double local memory arrays are employed during the chemical kinetics source term evaluation in order to accelerate reduction operations as well as scattered memory accesses. Such accesses occur, for instance, when the work items read the species concentrations which appear in the law of mass action. The integer local memory array, on the other hand, is used to store indices into the current column in the parallel pivot search which is part of the LU decompositions.

Second, the above choice of local memory buffers entails only moderate restrictions on n_s without inflicting upon the occupancy of a compute unit. To elaborate on this point, we consider the properties of the Nvidia Quadro 6000 which is one of the candidate GPUs in Section 5.6. On this device, each compute unit supplies 48 kB local memory, while admitting at most 8 work groups with $6 \cdot 32$ work items each to a compute unit (compute capability 2.0). If we assume that the kernel's register requirements are such that 8 work groups can be resident on the compute unit without violating register restrictions, then based upon the local memory usage we employ, n_s is limited above by 327. Thus, for a reaction mechanism with more than 326 species, fewer than the maximum number of resident work groups may be admitted to the compute unit.

A second memory limitation is due to the OpenCL constant memory space, where the maximum size of a single buffer can be device-constrained. Currently, the implementation allocates only one constant memory buffer of size

$$8 \cdot 14 \cdot (n_s - 1) \text{ B}. \tag{5.27}$$

On the Nvidia Quadro 6000, this memory buffer is limited to a maximum size of 64 kB, whence Eq. (5.27) implies $n_s \leq 506$. Contrary to the first limitation on n_s above, this restriction is strong, that is, the buffer cannot be created if n_s exceeds the limit. In practice, however, reaction mechanisms encompassing more than 505 species appear very rarely and, in particular, in the context of reactive flow simulations are far beyond the largest reaction mechanisms currently employed. For a different GPU, on the other hand, it may happen that the constant memory buffers are restricted to a smaller size than 64 kB and that the limitation on n_s is hence stricter. In this case, the constant buffer size limitation on n_s can be removed by declaring the respective constant memory buffer as global. As shown in Section 5.6.4, this has only a marginal (if any) effect on the performance of the OpenCL-GPU implementation.

Finally, there may be restrictions on the total size of global GPU memory that is accessible to the host program[15] and the maximum size of a global memory buffer. However, since the sizes of the biggest global memory buffers which Radau5 employs for temporary storage grow with the number of ODE systems per kernel invocation n_p (see Table 5.3), the global memory limitations can be mitigated by the overlapping kernel invocation scheme which is presented in the following section.

For the practical reaction mechanisms which we have tested as part of this work ($n_s \leq$

[15]The OpenCL implementation of the Nvidia GPU Computing Toolkit 5.5 does not seem to allow a single host thread to access all of global memory whose size can be retrieved by querying CL_DEVICE_GLOBAL_MEM_SIZE. In order to estimate the size of the accessible portion of global memory, we perform a binary search on the size of allocatable memory buffers modulo a memory allocation granularity and a resolution (256 B, for example).

Resource	Requirement	Nvidia Quadro 600(0)
Global memory	$8B\ \{2n_s + n_p[2 + n_s(13 + 4n_s) + n_r m_{sp}]$	1024 MB
	$+n_{fo}m_{fo} + n_{tb}m_{tb}\}+$	(6143 MB)
	$4B\ \{2n_p n_s + n_{fo} + n_{tb}m_{tb}\}$	
Constant memory	$8B\ \{14n_{sp} + 6n_r\}+$	64 kB
	$4B\ \{n_r(5 + 4m_{sp}) + n_{sp}\}$	
Local memory	$8B\ \{12 + 2n_s\}+$	48 kB
	$4B\ \{8 + l\}$	
Work group size l	≥ 1	≤ 1024

Table 5.3 Resources (in bytes, B) required by the OpenCL implementation. The final column indicates the limitations that are present when executing the kernel on two different GPUs.

118), the memory management scheme did not encounter the restrictions on constant or local memory buffer sizes mentioned above. The very detailed Jetsurf 2.0 mechanism ($n_s = 349$, $n_r = 2163$), by contrast, surpassed both the limitation on the amount of local memory per work group (for 8 resident work groups on a compute unit) and the GPU's limit on the maximum number of resident warps (for $l = 11 \cdot 32$ by Eq. (5.24) and 8 resident work groups). In this case, the number of resident work groups automatically reduces such that the constraints on the total number of resident warps per compute unit and the local memory and register limitations are obeyed. On the other hand, we could attempt to reduce the number of warps per compute unit, reckoning that this may increase the number of resident work groups, subject to the local memory and register constraints. We return to this point in Section 5.6.4, where the impact of the GPU's constraints on the integration of huge reaction mechanisms (such as the Jetsurf 2.0 mechanism) is assessed.

5.5.4 Solving the linear systems

In an implicit integration scheme, the computational effort is mainly concentrated in the subroutines which construct the Jacobian and solve the (dense) linear systems. Within the scope of Radau5, the Jacobian is approximated by forward differences and, thus, involves n_s evaluations of the chemical kinetics source terms.

The dense linear systems (one real and one complex system) are solved by standard LU decomposition with partial pivoting and subsequent forward/backward substitutions. As mentioned in Section 5.5.2, the loops which appear herein are parallelized across the work items in a work group. This made the introduction of synchronization points within the parallel pivot search, after the row permutation as well as after the L-column update and the elimination submatrix update necessary. In the course of this work, we have tested different schemes for computing the L- and elimination submatrix updates and found the best scheme/loop layout to be the following: The current column is stored in OpenCL local memory and, after the pivot search and row permutation, jointly updated by the active work items to obtain the new L-entries. Subsequently, all work items collectively update the remaining elimination submatrix in a column-wise manner, reading the L-entry corresponding to the current row from local memory and the pivot row element corresponding to the current column from global memory. If we examplarily consider the kth elimination step, then the L-update corresponds to a parallel $(n_s - k)$-loop, while the submatrix update involves a parallel $(n_s - k)^2$-loop without necessitating intermediate synchronization points. In this way, it is possible to directly map the $O((n_s - k)^2)$ operations of each submatrix update onto the GPU's lower level of concurrency. On an Nvidia GPU, for example, the number of instructions that are being issued sequentially during a submatrix update then reduces by one order of magnitude to $O((n_s - k)^2/32)$.

Furthermore, the left-hand-side matrices and the Jacobian are stored as one-dimensional arrays in a major column format. This enables the efficient coalescence of memory accesses since almost all of the parallel loops operating on these matrices are column-based. The only exception, here, is the pivot row permutation for which the memory accesses are row-based.

Finally, we point out that the above approach is at variance with the shared memory (reduced storage pattern) scheme tested by Le et al. [97]. In the context of a one-block approach, these authors suggested to store the pivot row and the row which is currently being updated in CUDA shared memory. Here, the update of the first element in the current row (encompassing columns k through n in the kth elimination step) corresponds to the evaluation of the L-entry, while the update of the remaining entries corresponds to updating one row of the elimination submatrix. As pointed out by the authors, the main shortcoming of this scheme is that there are two memory transfers from global to shared memory and vice versa for updating the jth row ($j = 1, \dots, n - k$) within the kth elimination step, requiring three synchronization points inside an $(n - k)$-loop. Le et al. [97] found that this shared memory-based approach is only effective for large reaction mechanisms with $\gtrsim 100$ species and, hence, reverted to a one-thread implementation in which the LU decomposition only operates on global memory buffers.

5.5.5 Overlapping kernel invocation and data transfer

If the number of ODE systems which can be integrated on the GPU is limited by a device resource (such as the available amount of global memory or the maximum allocatable buffer size) or if the data movement between host and device consumes a considerable amount of time, then the ODEs may be distributed across several kernel invocations. Although Nvidia GPUs are currently restricted to a single resident kernel at a time and the kernel invocations are therefore serialized, most professional GPUs (Nvidia Tesla and Quadro series) encompass so-called dual copy engines which enable sending data to the GPU and reading data from the GPU simultaneously or transferring data while a kernel executes. Following Reference [144], this allows the design of overlapping copy-compute-copy cycles.

Schematically, such cycles are depicted in Figure 5.2. Here, one p-cycle encompasses two t-cycles and a t-cycle corresponds to the conventional copy-compute-copy sequence.

The top line represents the commands submitted to the first command queue, while the bottom line indicates those submitted to the second command queue (to the same device).

Based upon the observations in Section 5.6, we outline a scheme for splitting the total number of ODEs n into r groups of size $n_p = n/r, r \geq 0$ (or smaller, for the last group) each of which requires a single t-cycle. As noted above, this splitting may be triggered, for instance, by the event that n exceeds the maximum number of ODE systems n_{max} whose data can simultaneously be stored in device memory.

If we take the time for copying data from host to device memory (t_{hd}) and vice versa (t_{dh}) as being independent of the direction in which the data moves (host to device or device to host), then the total runtime t can be computed from

$$t(r) = \left[2t_{dd}\left(\frac{n}{r}\right) + t_{dh}\left(\frac{n}{r}\right)\right]\frac{r}{2} + \left(1 + \frac{\mathrm{mod}(r,2)}{2}\right)t_{hd}\left(\frac{n}{r}\right), \tag{5.28}$$

where $r \geq 1$ denotes the number of t-cycles in Figure 5.2, $\mathrm{mod}(\cdot,\cdot)$ represents the modulo-operator and t_{dd} and $t_{hd} = t_{dh}$ indicate the times for executing a kernel on a device and for copying data to/from the device as functions of the number of ODE systems per kernel invocation $n_p = n/r$, respectively.

Figures 5.3 through 5.5 of Section 5.6 indicate that there exists a threshold n_0 that is both device and mechanism dependent and beyond which the kernel execution time t_{dd} grows linearly with the number of ODE systems n_p. For $n_p \geq n_0$, moreover, the time for packaging and moving data to the device either remains constant ($n_p < n_{hd}$) or increases linearly as well ($n_p \geq n_{hd}$). In the first case, we have $t_{dd} = a_0(n/r), t_{hd} = b_{hd}$ and, hence,

$$t(r) = a_0 n + b_{hd}\left(1 + \frac{r + \mathrm{mod}(r,2)}{2}\right), \tag{5.29}$$

where a_0 and b_{hd} are positive constants.

The trailing term in Eq. (5.29) indicates that $t(r+1) = t(r)$ if r is odd. Consequently, we can restrict the consideration to even r and set $r = 2s$, $s \geq 1$. t in Eq. (5.29) then increases linearly with s and, hence, s ought to be chosen as small as possible but such that $n_p \in [n_0, \min(n_{hd}, n_{max})]$,

$$s_1 \equiv s = \max\left(1, \mathrm{ceil}\left(\frac{n}{2\min(n_{hd}, n_{max})}\right)\right), \tag{5.30}$$

where $\mathrm{ceil}(\cdot)$ returns the smallest integer that is greater than or equal to the argument value.

For $n_p \geq n_{hd}$ and $n_{max} \geq n_{hd}$, on the other hand, we may set $t_{dd} = a_0(n/r)$ and $t_{hd} = a_{hd}(n/r)$, where a_0 and $a_{hd} > 0$ are constants. Introducing these approximations into Eq. (5.28) yields

$$t(r) = n\left\{a_0 + \frac{a_{hd}}{2}\left[1 + \frac{2 + \mathrm{mod}(r,2)}{r}\right]\right\}. \tag{5.31}$$

If r is even, then the difference between $t(r)$ and the time for $r+1$ t-cycles is given by

$$t(r) - t(r+1) = n\frac{a_{hd}}{2}\frac{2-r}{r(r+1)}\begin{cases} = 0 & \text{if} \quad r = 2 \\ < 0 & \text{if} \quad r = 4, 6, \dots \end{cases} \tag{5.32}$$

and, hence, $t(r+1) \geq t(r)$. Therefore, we may again concentrate on the case where r is even and set $r = 2s, s \geq 1$. By consequence, the modulo-term in Eq. (5.31) drops out and the total time t decreases hyperbolically with the number of p-cycles. Contrary to the first case, this suggests that s be chosen as big as possible, yet such that $n_p = n/(2s) \geq n_{hd}$,

$$s_2 \equiv s = \max\left(1, \mathrm{ceil}\left(\frac{n}{2n_{hd}}\right)\right). \tag{5.33}$$

The pseudo-code for the scheme which determines s and hence n_p based upon the above developments is summarized in Appendix C.2, Figure C.3. Note that since, for $r > 1$ and $n \geq \min(n_{hd}, n_{max})$, there is no benefit from considering the smaller t-cycles or from allowing a sequence of p-cycles to be terminated by a t-cycle, this scheme only involves even numbers of splittings, except for the case where n is small and a single t-cycle is taken.

Apart from the total number of ODE systems which, in the context of a fluid dynamics analysis is given by the number of grid points/finite volume cells, the above scheme requires that n_{max} and n_{hd} be specified. Here, n_{max} denotes the limiting number of ODE systems which can be integrated during a single kernel invocation on the given device and may be computed from the maximum allocatable buffer size or the maximum remaining memory on the device, depending on which one is smaller.

The threshold n_{hd}, on the other hand, depends upon the amount of data to be copied to the device as well as the memory transfer rates. Taking the overhead for submitting a read/write command to the device as a constant $\bar{t}_{hd} = 10^{-3}$ s, an estimate for t_{hd} in terms of both n and the mechanism size n_s can be obtained from

$$t_{hd}(n) = \bar{t}_{hd} + \frac{n_g(n_s+2)8\,\mathrm{B}}{BW}, \tag{5.34}$$

where BW denotes the unidirectional effective bandwidth of the PCIe bus. The numerator in Eq. (5.34) indicates the size of a package comprising an $n \times (n_s+2)$ double matrix of which the first n_s columns contain the species mass fractions and enthalpy/temperature and the final two columns are reserved for density and error reporting, respectively. If n_{hd}

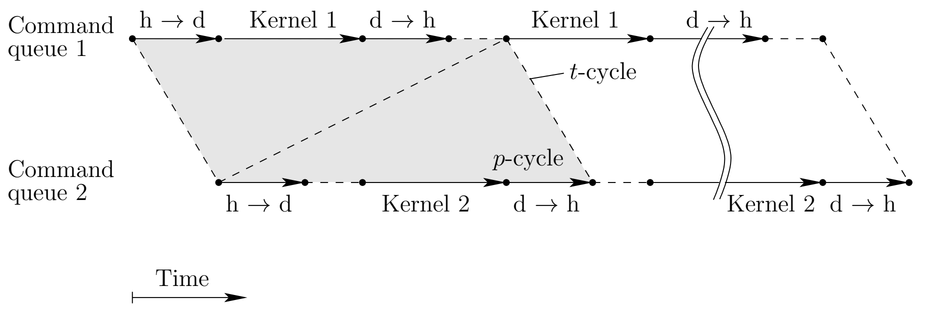


Figure 5.2 t- and p-cycles for executing multiple kernels successively and overlapping the computation with data transfers from the host (h) to the device (d) and vice versa [146].

is defined as $t_{hd}(n_{hd}) = 2\bar{t}_{hd}$, then Eq. (5.34) yields, for $BW = 8\,\mathrm{GB/s}$,

$$n_{hd} = \bar{t}_{hd}\frac{BW}{(n_s+2)8\,\mathrm{B}} = \begin{cases} 3.2\times10^4 & \text{(GRI 1.2)} \\ 1.9\times10^4 & \text{(GRI 3.0)} \\ 8.8\times10^3 & \text{(Curran)} \end{cases} \tag{5.35}$$

which agrees well with the measurements in Figures 5.3 through 5.5 in Section 5.6.

5.5.6 Employing a different integration scheme

The OpenCL reimplementation of Radau5 which we have developed in this work is modular in structure and encompasses a top-level function implementing the core integration scheme as well as a collection of auxiliary functions for the evaluation of the right hand side (species production/destruction rates and temperature source term), the real/complex LU decompositions, the real/complex forward/backward substitutions, the error estimation and parallel reductions. The one-block approach which has been introduced in Section 5.5.2 transcends through the hierarchy of function calls inside the OpenCL kernel; in particular, it entails a parallelization of the innermost loops across the work items within a work group.

In this modular setup, the integration algorithm can be changed at a reduced effort since many of the individual components of a complete integrator are shared among implicit integration schemes. In particular, only the top-level integration scheme and the function which returns the current error estimate have to be replaced, while the remaining auxiliary functions can be kept. This is further aided by the convention that only the first work item updates the integration control variables (Section 5.5.3), whence the top-level integration function remains serial in part.

5.6 Numerical experiments

In the present section, we assess the proposed OpenCL-GPU implementation in comparison with a standard MPI-CPU program. In particular, considerations of relative speedup and time complexity (problem and mechanism size) are complemented with an analysis of the influence of the time step size and the convergence tolerances.

As test-bed for generating initial conditions which cover a wide range of compositions and temperatures, we consider a transient equilibrium scheme for the flamelet model with a strain rate of $s = 4 \times 10^2\,\mathrm{s}^{-1}$, see Section 5.2. In this way, the results presented in the following may be viewed as representative of a real-world reactive flow analysis.

Both the average and the minimum/maximum values of the runtime and the data transfer time given below apply to a single time step of size $\Delta t = 10^{-6}\,\mathrm{s}$ (except for Figures 5.9 and 5.10, where Δt varies) and have been obtained from time measurements on the first 10 time steps. During a reaction fractional step one ODE system is solved for each node of a uniform spatial discretization in mixture fraction space (excluding the boundary points $z = 0, 1$). For the Radau5 integration scheme, the absolute/relative convergence tolerances for species mass fractions and temperature have been set to $10^{-8}/10^{-5}$ and $10^{-5}/10^{-5}$, respectively.

The OpenCL-GPU implementation is compared with a top-level MPI-parallelization of the original Radau5 CPU implementation. Here, one MPI process is invoked for each processing thread on the CPU (hyper-threading enabled) and the chemical kinetics ODE systems are assigned to the MPI processes in a strided fashion. The comparison between both implementations involves a consumer level GPU (Nvidia Quadro 600) and a high-end GPU (Nvidia Quadro 6000) alongside a standard CPU (Intel i5-520M) and a scientific workstation CPU (Intel Xeon E5-2687W). The MPI-CPU implementation was compiled by the Intel Fortran compiler 14.0.0 in combination with Open MPI 1.6.5, all optimizations enabled. With regard to the OpenCL-GPU implementation, the "-cl-fast-relaxed-math" optimization flag was passed to the OpenCL compiler (Nvidia CUDA Toolkit 5.5). Finally, both implementations employ double-precision for all floating point operations.

The size properties of the reaction mechanisms which we tested and the fuels for which the time measurements have been taken are detailed in Table 5.4.

The analyses in the following sections are based upon measurements of the runtime t of the reaction fractional step for the OpenCL-GPU and the MPI-CPU implementation on a given device. In general, t depends upon 5 input parameters

$$t = t(n, n_s, \Delta t, \mathbf{tol}, \{\mathbf{Y}_0^{(1)}, \ldots, \mathbf{Y}_0^{(n)}\}). \tag{5.36}$$

Here, **tol** symbolizes the absolute/relative convergence tolerances and $\{\mathbf{Y}_0^{(1)}, \ldots, \mathbf{Y}_0^{(n)}\}$ summarize the initial conditions for a collection of n ODE systems. As indicated above, we account for a whole range of initial conditions by uniformly distributing in mixture fraction

Name	# Species	# Reactions	# Fall-off reactions	# Third-body reactions	Fuel	References
GRI 1.2	31	175	20	27	CH4	[53, 54]
GRI 3.0	53	325	29	41	CH4	[192]
USC II	75	529	51	59	C2H4	[205]
Luche	89	680	42	47	C10H22	[110]
Curran	118	665	28	27	C3H8	[153]
Jetsurf 2.0	348	2163	387	393	CH4	[204]

Table 5.4 Size properties of the tested reaction mechanisms. Here, the second to last column lists the fuels for which the mechanisms are applied.

space the z-values that are associated with the ODE systems inside such a collection. In this way, the impact of stiffness on the runtime is reduced since the stiff ODE systems whose z-values lie in the vicinity of the stoichiometric mixture fraction value $\bar{z}$ are solved concurrently with a number of less stiff ODE systems whose associated z-values are located further away from $\bar{z}$ in mixture fraction space.

In Sections 5.6.1 through 5.6.4 we first quantify the dependence of t upon the problem size n and the mechanism size n_s for a given time step size Δt and convergence tolerances **tol**. Here, the values for Δt and **tol** chosen above are such that for the reaction mechanisms in Table 5.4 all ODE systems require one integration step independent of their associated mixture fractions. Subsequently, the influence of larger time step sizes for which the numbers of integration steps vary across mixture fraction space is examined in Section 5.6.5. The results presented in this section also apply to the influence of the convergence tolerances **tol** if the time step size Δt is held fixed. Finally, we collect concluding remarks in Section 5.6.6.

5.6.1 Varying the problem size

Figures 5.3 through 5.5 depict both the total runtime (including the time for assembling and moving data to and from the GPU) and the data transfer time over the total number of ODE systems for given reaction mechanisms. Here, the solid lines indicate the runtime averaged over 10 global time steps, while the top and bottom boundaries of the shaded areas show the minimum and maximum values, respectively. The broken lines, on the other hand, correspond to the data transfer time and have been obtained as the mean, for a single t-cycle, of the time consumed by allocating parcel memory on the CPU, assembling data and moving the parcel from the host to the device before the kernel is invoked and of the time for sending data back from the device to the host and distributing the results to host variables after the kernel terminated. As above, these time measurements are averaged over 10 consecutive global time steps.

If the total number of ODE systems n exceeds a small device-dependent threshold value n_0, then the kernel execution time increases linearly with n for the selected devices. In Figures 5.3 through 5.5, this is indicated by the fact that the slope of the log-log-lines equals 1 identically for $n \geq n_0$. Note that this is in contrast to the findings of Shi et al. [190] who reported a sub-linear scaling for the GPU implementation of an explicit solver.

For $n < n_0$, on the other hand, the computing time of the GPU implementation tends towards a constant value as n approaches 1 from above. In view of the GPU architecture, we can attribute this saturation effect to the following three points:

(1) Low occupancy. For a small number of ODE systems, there are only few resident SIMD units on each compute unit such that the time intervals during which SIMD units wait on a barrier or for memory accesses to complete cannot be completely hidden by switching to another SIMD unit which is ready to execute, possibly from a different work group (as there may not be one). (Since the number of SIMD units per work group is determined by the mechanism size n_s, the number of resident SIMD units on a compute unit increases with n_s for a given number of ODE systems n. This explains why n_0 in Figure 5.5 is slightly smaller than in Figures 5.3 and 5.4.)

(2) Idle compute units. If some compute units remain idle while a small number of ODE systems are being integrated, then these compute units appear as additional resources as n increases. Thus, the additional work does not add to the total kernel execution time.

(3) Kernel invocation overhead.

In terms of the overall runtime, we may infer from Figures 5.3 through 5.5 that for $n \gtrsim 20$ the professional level devices (Intel Xeon E5-2687W and Nvidia Quadro 6000) are more efficient than the user-end processors (Intel i5-520M and Nvidia Quadro 600) and that for both pairs of devices the MPI-CPU reference implementation outperforms the OpenCL-GPU implementation. For $n \lesssim 20$, on the other hand, the MPI-CPU solver performs better on both the professional and the user-end CPU than its OpenCL-GPU counterpart on either GPU.

As noted above, the dashed lines in Figures 5.3 through 5.5 indicate the average time for packing/unpacking and moving the grid point dependent data to/from the GPU before and after the kernel executes. Qualitatively, these graphs are very similar to those for the total runtime: Below a threshold n_{hd}, the time for packaging and moving data is governed by the overhead that is associated with issuing a read/write command to the GPU, while, for $n \geq n_{hd}$, the time for data packaging and movement is limited by the bandwidth of the PCIe bus. Indeed, if the data communication time were plotted over nn_s, the dashed lines in Figures 5.3 through 5.5 would coincide for each device and the slope of the linear part of the graphs would correspond to the PCIe bandwidth. In Section 5.5.5, this relation was used to estimate n_{hd} (Eq. (5.35)).

By comparing the solid and dashed lines in Figures 5.3 through 5.5, it can be seen that for $n \geq n_{hd}$ the computing time exceeds the time for packaging/moving data by about

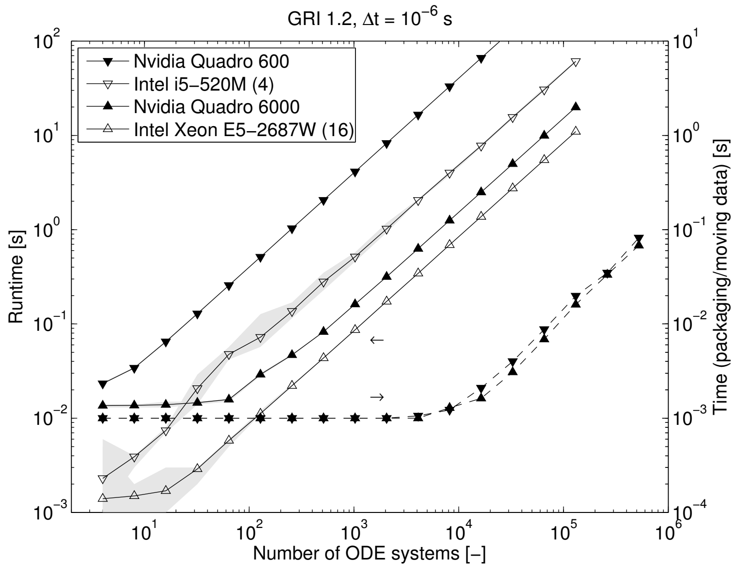


Figure 5.3 Comparing the runtime (measured on the host CPU) of the MPI-CPU and the OpenCL-GPU implementation for the GRI 1.2 mechanism and an increasing number of ODE systems. The bottom graphs, furthermore, depict the time consumed by packaging and transferring data between the host and the device in the case of the GPU implementation. The solid or dashed lines represent mean values, while the shaded areas indicate maximum/minimum values from a sample of 10 consecutive time steps.

three orders of magnitude for the selected reaction mechanisms. This suggests that there is only little practical benefit in scheduling the kernel invocations and memory transfers in the form of p-cycles as opposed to serially executing a t-cycle for each group of ODE systems.

5.6.2 Speedup compared to an MPI-CPU implementation

Complementary to Figures 5.3 through 5.5, Figure 5.6 depicts the relative speedup of the OpenCL-GPU implementation as compared to the MPI-CPU implementation for selected reaction mechanisms. Here, the high-end GPU performs at best 1.8 times slower than the workstation CPU (16 threads), while the consumer level GPU is at best 5.5 times slower than its CPU counterpart (4 threads). As above, this comparison is processor-based. (By contrast, the speedups reported in References [21, 97, 140, 190, 191, 195, 196] have been computed with respect to a serial CPU implementation.)

Moreover, Figure 5.6 indicates that the runtime of the OpenCL-GPU implementation relative to the MPI-CPU reference implementation is largely (although not completely) independent of the reaction mechanism size on the high-end devices. Here, the relative runtime varies between 1.8 for the GRI 1.2 reaction mechanism and 2.1 for the Curran mechanism. For the user-end devices, on the other hand, these values show a much larger spread, ranging from 8.2 for the GRI 1.2 reaction mechanism to 5.4 for the Curran mechanism. Also, we observe that the relative OpenCL-GPU/MPI-CPU performance on the user-end devices improves as the reaction mechanism size is increased, while the reverse

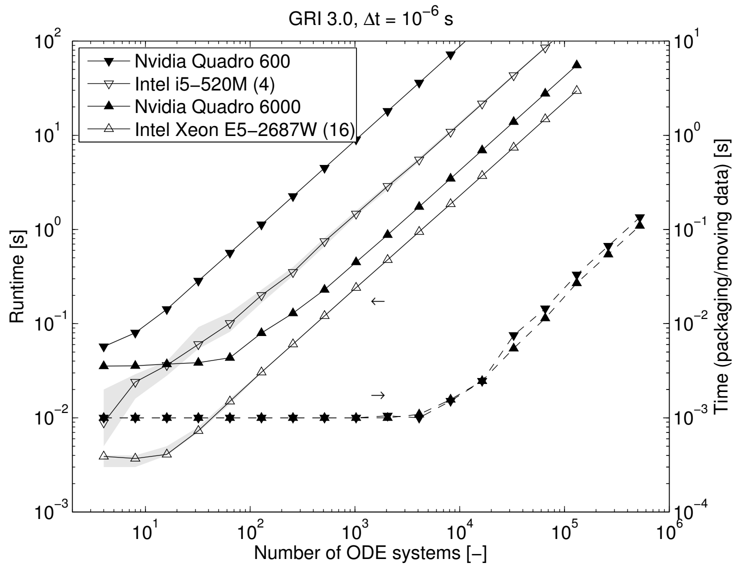


Figure 5.4 Comparing the runtime of the MPI-CPU and the OpenCL-GPU implementation for the GRI 3.0 mechanism and an increasing number of ODE systems. (Also see Figure 5.3.)

is true for the professional level devices. This may be compared with the results presented by Niemeyer and Sung [141] and Shi et al. [190] (also, see Table 5.1, rows $3-5$ and $7-8$) which suggest that the relative performance of explicit one-thread GPU integrators increase with growing reaction mechanism sizes.

Finally, we infer from Figure 5.6 that the OpenCL-GPU implementation attains its limit speedup at a small problem size of ≈ 500 ODE systems. This is in line with the conclusions of Stone and Davis [196] and constitutes a major advantage over existing explicit/implicit one-thread GPU implementations which have been reported to reach their limit speedups at large problem sizes on the order of 10^5 ODE systems.

5.6.3 A performance measure

Since the computing time varies linearly with the number of ODE systems n beyond a small threshold n_0, the t-n-dependence can be collapsed into a single number: the slope of the linear curves in Figures 5.3 through 5.5 measured in a linear-linear plot. By inverting this slope, we obtain $\dot{n}$, the number of ODE systems which the implementation on a device can solve per unit of time for a given reaction mechanism, a given time step size and fixed convergence tolerances. For the GRI 1.2, GRI 3.0 and Curran mechanisms, these data are summarized in Table 5.5. From a cost-effectiveness perspective, moreover, the bottom part of Table 5.5 shows the number of ODE systems per time unit normalized by the investment cost of the respective device. Here, the user-end CPU promises the highest performance per investment, while in terms of time only the MPI-CPU implementation on the workstation CPU is the most efficient.

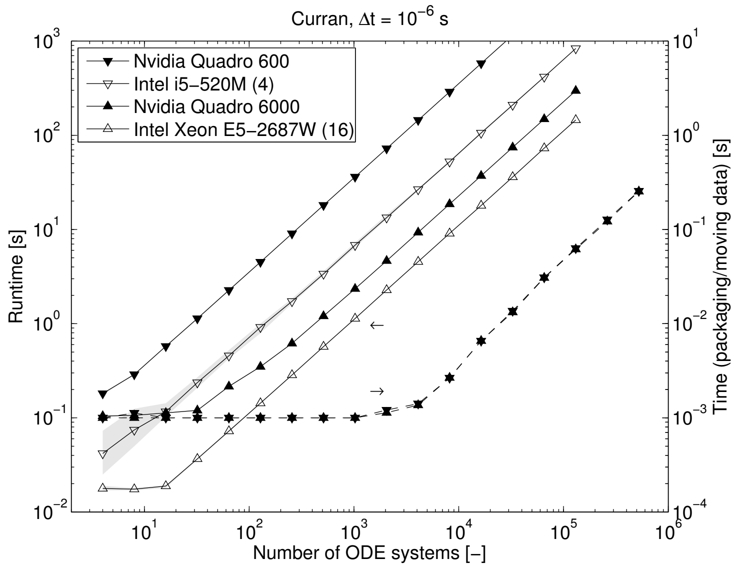


Figure 5.5 Comparing the runtime of the MPI-CPU and the OpenCL-GPU implementation for the Curran mechanism and an increasing number of ODE systems. (Also see Figure 5.3.)

If the sorting procedure we described in Section 5.4.2 is omitted, then the numbers of ODE systems per time unit in Table 5.5 drop by 10.2 % (GRI 1.2), 7.5 % (GRI 3.0) and 5.4 % (Curran), respectively, for the Nvidia Quadro 6000. These reductions suggest that the performance penalty due to thread divergence subsides inverse proportionally as the number of warps increases.

5.6.4 Varying the mechanism size

Figure 5.7 depicts the total runtime (including both the kernel execution time and the time for packaging/moving data to the device and back) over the mechanism size n_s for a given number of ODE systems. As above, the data lines represent the average elapsed time of a single reaction fractional step computed from 10 consecutive time steps and the top and bottom boundaries of the shaded areas indicate the minimum and maximum times, respectively. Here, the slopes of the graphs are approximately equal to 2.0 such that the computational cost grows quadratically as the reaction mechanism size increases. While, at first sight, this result seems to be at odds with the time complexity $O(n_s^3/3)$ of the LU decomposition, we may take it as an indication for the dominance of the forward/backward substitution algorithm which scales as $O(n_s^2/2)$. Indeed, the Jacobian and its LU decomposition are not recomputed during a single Newton-type integration step unless the scheme tends to diverge.

Figure 5.8, on the other hand, shows how the number of ODE systems which can be integrated on a given device per time unit varies as the reaction mechanism size increases. For moderately sized reaction mechanisms ($n_s \lesssim 200$), the results depicted here suggest

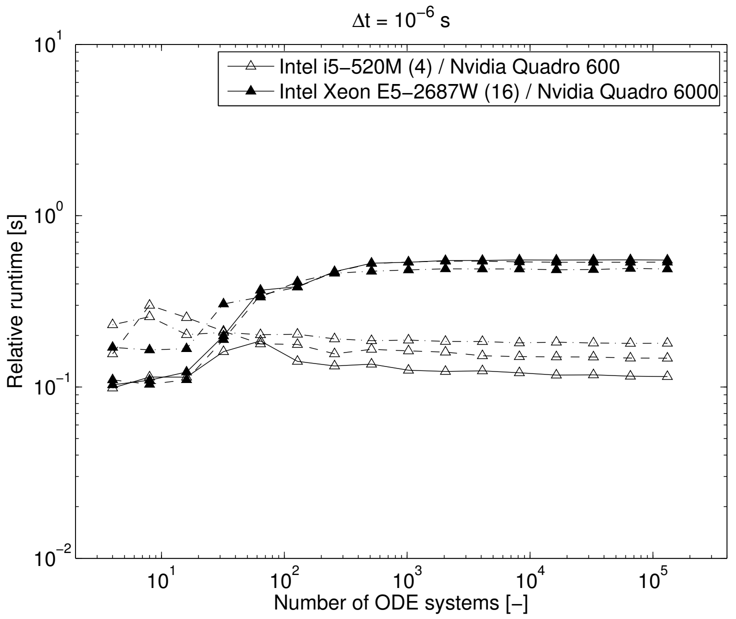


Figure 5.6 Total runtime of the OpenCL-GPU implementation on an Nvidia Quadro 600 and Nvidia Quadro 6000 relative to the reference runtime on an Intel i5-520M and an Intel Xeon E5-2687W processor, respectively, over the number of ODE systems. The solid lines indicate the results for the GRI 1.2 mechanism, the dashed lines those for the GRI 3.0 mechanism and the dash-dotted lines show the relative runtime for the Curran mechanism.

that the number of ODE systems per time unit decreases as $1/n_s^2$. This is indicated by the dotted regression lines in Figure 5.8. Also, the result $\dot{n} \sim 1/n_s^2$ for $n_s \lesssim 200$ is commensurate with the relation $t \sim n_s^2$ of Figure 5.7.

The far right hand side of Figure 5.8 includes data points for the Jetsurf 2.0 mechanism which encompasses 348 species and 2163 reactions. Although this mechanism is very detailed and may hence be computationally not feasible for practical reactive flow simulations, we include it here in order to assess the effects of local memory limitations and device-related constraints. In this respect, Figure 5.8 depicts a shaded threshold interval above which the occupancy of the Nvidia Quadro 600(0) significantly decreases on account of the resources that a single work group allocates. Here, the left interval boundary indicates the value for n_s at which the work group size times the maximum number of resident work groups reaches the total number of work items that can simultaneously reside on a compute unit. On the Nvidia Quadro 600(0), a compute unit can accommodate at most $48 \cdot 32$ work items being distributed across ≤ 8 work groups. In combination with Eq. (5.24), this leads to a threshold above which the work groups encompass so many work items that less than 8 work groups are admitted to the compute unit. The right boundary of the shaded interval, on the other hand, indicates the reaction mechanism size above which less than 8 work groups are admitted to a compute unit on account of OpenCL local memory restrictions. Here, Eq. (5.24) has been replaced by a constant work group size of $6 \cdot 32$ such that within the shaded interval 8 work groups may reside on a compute

		Intel i5-520M	Nvidia Quadro 600	Intel Xeon E5-2687W	Nvidia Quadro 6000
Price		£222	£170	£2681	≈ £3600
# ODE systems / s	GRI 1.2	2032	249	11 683	6538
	GRI 3.0	743	113	4334	2344
	Curran	152	28	902	439
# ODE systems / $s - £$	GRI 1.2	9.15	1.46	4.36	1.82
	GRI 3.0	3.35	0.67	1.62	0.65
	Curran	0.68	0.17	0.34	0.12

Table 5.5 Numbers of ODE systems per second of runtime and Pound Sterling invested into the device for selected reaction mechanisms ($\Delta t = 10^{-6}$).

unit.

This last point is closely related to the question of whether, for large reaction mechanisms, it is more efficient to increase the number of work items per work group according to Eq. (5.24) and let the number of resident work groups decrease or to limit the work group size so as to retain as many resident work groups as possible. For the Jetsurf 2.0 mechanism, we have tested both options and found that the second option leads to an increase in the number of ODE systems per time unit by 42.1% and 50.0% for the Nvidia Quadro 6000 and Quadro 600, respectively, as compared to the first policy. In Figure 5.8, the measurement point for the second option is shown.

In order to quantify the change in performance for large reaction mechanisms, the $\dot{n}$ value that is predicted by extrapolating the dotted regression line for the medium sized reaction mechanisms to $n_s = 349$ for the Jetsurf 2.0 mechanism can be compared with the measured values. For the second option above ($6 \cdot 32$ work items per work group), the measured number of Jetsurf 2.0 ODE systems that are being solved on the Nvidia Quadro 600 and Quadro 6000 per second are 25.0% and 37.0% smaller than the extrapolated values, respectively. This may be taken as an indication for the estimate that, beyond the shaded interval, the efficiency of the GPU integrator decreases by about one third.

The MPI-CPU implementation suffers from a similar deterioration in performance. Here, the measured $\dot{n}$ values for the Jetsurf 2.0 mechanism deviate from the extrapolated values by 27.7% and 27.2% on the Intel Xeon i5-520M and Intel E5-2687W CPUs, respectively. Thus, both implementations degrade significantly in efficiency when the Jetsurf 2.0 mechanism is applied in place of a medium sized reaction mechanism. In our opinion, this indicates that the reduction in occupancy of the GPU is paralleled by a counterpart-effect of resource limitations on the CPU. Although the deterioration in performance may be attributed entirely to these device-related constraints, we believe that, at least in part, it is also due to the $O(n_s^3/3)$ complexity of the LU-decomposition which may only take effect

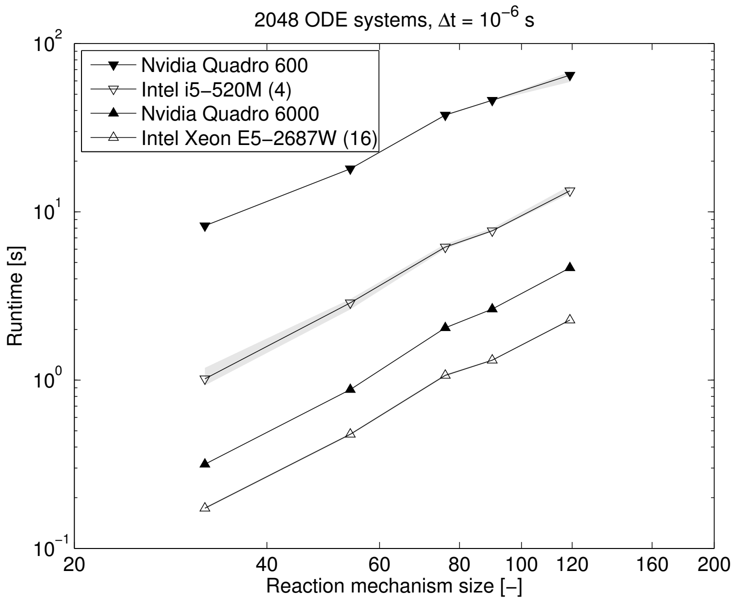


Figure 5.7 Comparing the runtime of the MPI-CPU and the OpenCL-GPU implementation for 2048 ODE systems and different reaction mechanism sizes (n_s).

for such large reaction mechanism sizes as $n_s \approx 350$.

The above analysis on occupancy does not take into account, however, that for a given work group size the number of resident work groups is constrained by the register availability. On a compute unit, registers are allocated by work groups and the number of registers which a work group requires is determined from the number of registers per work item, the work group size and the register allocation granularity. Here, the number of registers allocated by a work item of a particular kernel function is difficult to determine, in practice, unless a profiling software is available. Nevertheless we can assess the influence of the register limitation in a qualitative sense since, for a given work group size and a number of registers per thread, the register count constraint yields an upper limit on the number of resident work groups. In the above, this upper limit can be used to replace the 8 maximally resident work groups. As a result, the boundaries of the shaded interval slightly shift to the left, possibly by unequal amounts. Therefore, the limits in Figure 5.8 are rather approximative and subject to a left-shift which is smaller for a larger number of available registers.

Furthermore, we investigated the effect of declaring all OpenCL constant memory buffers as global. This might be necessary for GPUs which supply only a very limited amount of constant memory such that the constant memory buffer which the OpenCL implementation allocates would exceed the admissible size. For the user-end Nvidia Quadro 600, we found this redeclaration not to have any effect on the number of ODE systems that were solved per unit time, while, on the Nvidia Quadro 6000, $\dot{n}$ decreased very slightly (by less than 1%) for all reaction mechanisms.

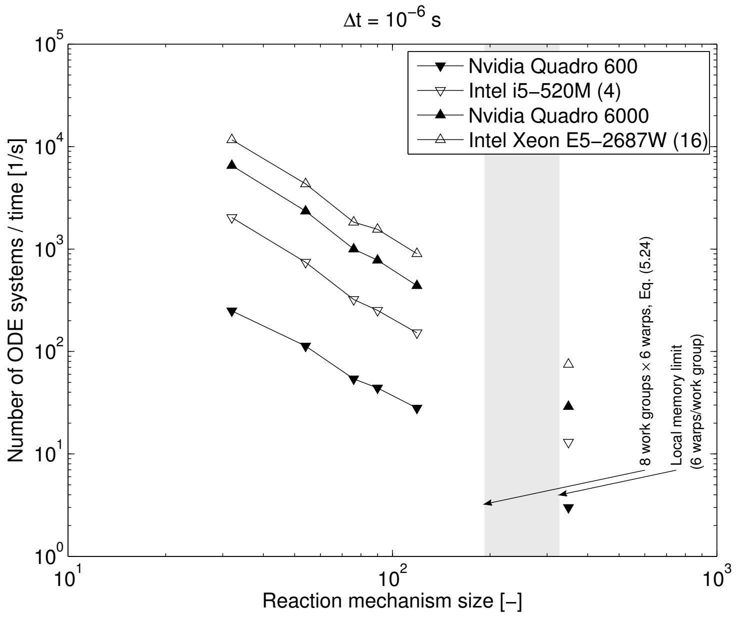


Figure 5.8 Number of ODE systems that are integrated on different devices per unit of time over the reaction mechanism size (n_s). Here, the dotted lines indicate linear regression approximations for $n_s \lesssim 200$. Furthermore, the shaded area indicates an estimate for the n_s interval above which the performance of the OpenCL-GPU implementation severely degrades on account of resource limitations.

5.6.5 Varying the time step size

Both the global time step size Δt and the absolute/relative convergence tolerances have a similar, albeit converse impact on the total runtime: If Δt is increased or the convergence tolerances are decreased, then the number of step size refinements (that is, the number of integration steps per global time step) increases for those ODE systems whose associated z-values are close to the stoichiometric value $\bar{z}$, while the number of integration steps remains small for those ODE systems which are far away from $\bar{z}$ in mixture fraction space. Thus, the number of integration steps is more inhomogeneously distributed in mixture fraction space and dynamic load balancing by the work group scheduler may not be able to hide the very stiff systems entirely. In order to quantify the stiffness inhomogeneity across mixture fraction space, we consider the ratio of the maximum number of integration steps which an ODE system requires per global time step and the minimum such number.

In Figures 5.9 and 5.10, this ratio (solid lines) is depicted along with the number of GRI 3.0 ODE systems which are solved on a given device per time unit (bars) for an increasing time step size. Here, the values for the stiffness inhomogeneity have been computed as averages across a number of problem sizes ranging from $n = 2^2$ to 2^{17}. In both Figures, the numbers on top of two adjacent bars compare the height of the left bar with that of the right one; they thus indicate the speedup of the MPI-CPU as compared to the OpenCL-GPU implementation. The results suggest that this speedup is invariant for the professional level devices, while, for the user-end devices, it shifts slightly towards the

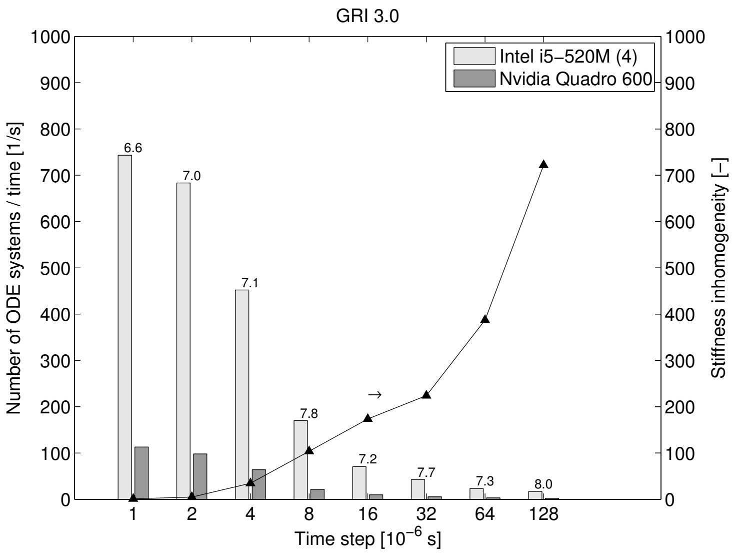


Figure 5.9 Number of ODE systems which can be solved per second on a user-end device (bars) and stiffness inhomogeneity (solid line) for different time step sizes. The numbers on top of two adjacent bars indicate the ratio of the bars' heights and correspond to the speedups of the baseline MPI-CPU implementation as compared to the OpenCL-GPU implementation for a number of ODE systems $n \geq n_0$.

CPU as the global time step size and, hence, the stiffness inhomogeneity increase.

5.6.6 Concluding remarks

In view of the benchmark Table 5.1, it may appear that the implementation which we developed in this work competes with the DVODE reimplementation by Stone and Davis [196] and, in this comparison, falls short. However, both implementations pursued different objectives: While we aimed at an implementation that is general by nature, allowing for the user to specify any Chemkin-format reaction mechanism, Stone and Davis [196] concentrated on a 19-species ethylene reaction mechanism and optimized their implementation accordingly. In particular, the authors were able to store, for each ODE system or thread block, the LU factors of the Newton iteration matrix, its pivot array, the current right-hand-side vector and temporary arrays in CUDA shared memory without inflicting upon the number of resident thread blocks. We believe that this strategy accounts for the main performance difference in comparison with the present OpenCL implementation in which local memory usage is restricted to two double n_s-arrays and one integer l-array (plus 128 B of control variables) per work group. Since the reaction mechanism data in Reference [196] is small, moreover, it may completely fit into the GPU's constant memory cache, allowing register-speed access.

At the same time, we can rule out that the performance difference is incurred by using the OpenCL API as opposed to the CUDA API (which has been employed by all references in Table 5.1), see Fang et al. [47].

Further to the local memory layout detailed in Section 5.5.3, we have tested comple-

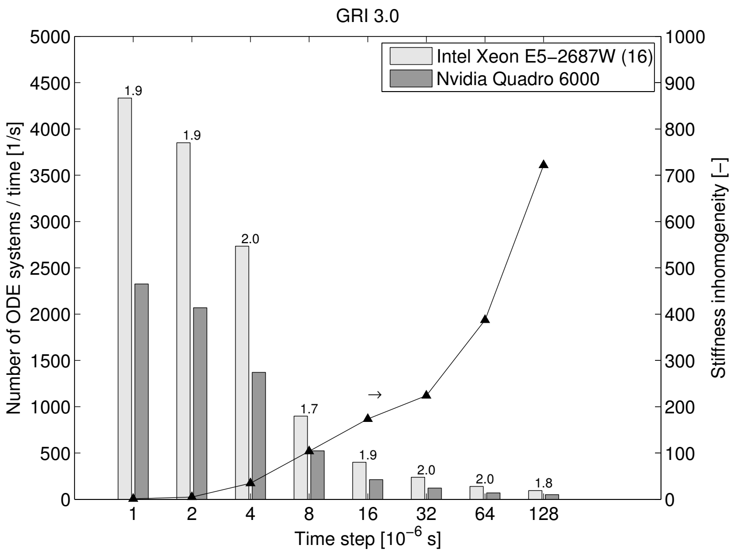


Figure 5.10 Number of ODE systems which can be solved per second on a professional level device (bars) and stiffness inhomogeneity (solid line) for different time step sizes. (Also see Figure 5.9.)

menting the two n_s-arrays by a third one. This proves advantageous in that both the real and the complex LU decompositions and forward/backward substitutions, respectively, may be merged into single subroutines, where either share the loop control schemes as well as the synchronization points. In addition, a third n_s-array can be incorporated into the source term evaluation subroutine in order to reduce the number of global memory accesses. For the reaction mechanisms in Table 5.4, the three n_s-local memory arrays implementation on an Nvidia Quadro 6000 GPU achieved a maximum speedup of 4.9 and 0.56 over a single and 16 thread CPU implementation, respectively. This slight increase in performance, however, trades against a stricter limit on the number of species ($n_{sp} \leq 213$, neglecting register limitations) which can be accommodated before occupancy reduces. This emphasizes that the implementation seeks a compromise between mechanism restrictions and performance through the way in which it manages the GPU resources.

Finally, we believe that the performance of the present implementation may benefit from an analytical subroutine for computing the source term Jacobian. In this way, the total number of global and constant memory accesses within the Jacobian computation may reduce considerably since the reaction mechanism data would only have to be read a few times. On the part of the CPU implementation, this might not have as large an impact since, for the finite difference approximation, the CPU already benefits from its large caches through which the mechanism data are being kept readily available.

On the minus side, the latter approach is hardly realizable if a reduced chemistry scheme replaces the current source term evaluation scheme. For this reason we have abstained from exploring it.

5.7 Chapter summary

Based upon the 5th order implicit integration scheme Radau5 [67], we have developed a novel OpenCL implementation for solving the reaction fractional step on a GPU.[16] The implementation follows the one-block approach advocated by Stone and Davis [196] and attains its limit speedup at a small problem size of ≈ 500 ODE systems. Also, the constant and local memory buffer layout is such that the implementation can accommodate Chemkin-format reaction mechanisms with $\lesssim 200$ species without encountering resource limitations.

In view of the architecture of a GPU, the implementation exploits GPU-specific utilities: Scattered or repeated array accesses are deferred to temporary arrays in OpenCL local memory; the number of registers per work item is reduced by storing the integration control variables in local memory; and (with a single exception) all global memory accesses are coalesced.

For large problems, moreover, the ODE systems are distributed across several kernel invocations which overlap with data transfers from the host CPU to the GPU and back. For this scheme, we developed a first model on the total runtime t and, based upon an analysis of the model, derived a strategy for determining the number of ODE systems per kernel invocation which minimizes t.

Finally, thread divergence within the source term evaluation subroutine is mitigated by sorting the reactions for properties on which they disagree. This yielded a performance benefit of up to 10.2 % for the Nvidia Quadro 6000 and the GRI 1.2 mechanism.

A thorough performance analysis was based upon runtime measurements of the reaction fractional step within a transient equilibrium scheme for the flamelet model. As reference implementation for a comparison we considered a top-level MPI-parallelization of the original Fortran 77 subroutine Radau5 by Hairer and Wanner [67]. Here, a range of initial conditions was accounted for by distributing the ODE systems uniformly in mixture fraction space. The analyses included performance assessments for varying problem sizes, reaction mechanisms, time steps and convergence tolerances. The latter two were demonstrated to exert a strong influence upon the homogeneity of the distribution of the number of step size refinements in mixture fraction space. In particular, we found that in the case of the user-end devices the performance ratio slightly shifts towards the CPU for increasing time step sizes or decreasing convergence tolerances.

Furthermore, the numerical tests demonstrated that the performance of the OpenCL-GPU implementation relative to the MPI-CPU reference implementation on high-end devices is largely independent of the reaction mechanism size. This is at variance with explicit GPU integrators based on the one-thread approach for which the performance on comparable devices depends upon the mechanism size [140, 190].

[16]If you wish to obtain the current OpenCL-GPU implementation, then please email us.

However, in comparison with the baseline MPI-CPU implementation we conclude that the present implementation falls short. For a comparison on a per-processor level, the OpenCL-GPU implementation performs at best 1.8 (professional level) and 5.5 (user-end) times slower than its MPI-CPU counterpart. If we compare the performance of the GPU against a single CPU thread, on the other hand, a maximum speedup of 4.8 can be demonstrated for the professional level devices. This result is in contrast to the speedup of 7.3 reported by Stone and Davis [196] for a CUDA DVODE reimplementation which has been specialized to a 19 species reaction mechanism. The mismatch between both implementations demonstrates the potential performance improvement for small reaction mechanisms at the sacrifice of general applicability.

Although the OpenCL-GPU implementation is thus outperformed by its MPI-CPU counterpart, both implementations may operate concurrently. In this regard, the ODE systems can be distributed across the CPU and the GPU such that the runtimes on both processors match. This idea is explored further in Chapter 6.

Finally, we point out that recent generations of Intel processors include a GPU on the same die as the CPU. Here, both the CPU and the accompanying GPU access the same main memory. This renders OpenCL's capability for mapping data arrays allocated in the host CPU's main memory onto OpenCL buffers very efficient [82]. Thus, the combined CPU-GPU architecture is able to mitigate the time consumed by host to device and device to host memory transfers via the PCIe bus which a standard CPU/external GPU pair requires. The line of development which these processor types outline favours a close cooperation between the CPU and the GPU in modern personal computers and, hence, corroborates both the importance and the potential of GPU acceleration.

Chapter 6

Chemical kinetics integration on a CPU-GPU pair

6.1 Introduction

In the previous chapter, we presented a GPU reimplementation of the high order implicit integration scheme Radau5, but found that, in a comparison on a per-processor basis, solving a representative reaction fractional step on the GPU is significantly slower than a conventional MPI parallelization of Radau5 on a CPU. However, since most desktop computer systems possess both a CPU and a GPU, there may be some benefit in combining the CPU and GPU implementations and parallelizing the reaction fractional step across both devices.

In the present chapter, we thus device such a combined CPU/GPU implementation for solving the reaction fractional step and quantitatively assess the gain in performance. The parallelization strategies which we explore involve an on-the-fly determination of the relative CPU/GPU performance and automatically balance the work loads such that idle times on either processor are mitigated. Furthermore, the implementation is modularized and can readily be incorporated into an established reactive flow solver. As a realistic test case, we consider the Sandia D flame and compare the performance of the combined CPU/GPU implementation with that of one GPU-only and two CPU-only implementations in order to identify the most profitable parallelization strategy.

This chapter is organized as follows: In Section 6.2 we briefly review the formulation of the reaction fractional step, before different strategies for parallelizing the reaction fractional step across a CPU-GPU pair are analyzed in Section 6.3. Subsequently, in Section 6.4, the performance of the CPU/GPU implementation is quantitatively assessed in terms of time measurements of the reaction step in an LES of the Sandia D flame and compared with both CPU and GPU-only implementations. Our findings and recommendations regarding the most profitable strategy are summarized in Section 6.5.

6.2 Reaction fractional step

Numerical solution schemes for reacting flows frequently involve an operator splitting technique which allows treating diffusion, convection and reaction phenomena sequentially. Typically most of the computational effort is concentrated in the reaction fractional step which involves solving one ODE system for each spatial grid point $\mathbf{x}_i$, $i = 1, \ldots, n$,

$$\frac{d\mathbf{Y}(\mathbf{x}_i, t)}{dt} = \frac{\dot{\boldsymbol{\omega}}}{\rho}(\mathbf{Y}(\mathbf{x}_i, t)), \quad \mathbf{Y}(\mathbf{x}_i, t_0) = \mathbf{Y}_0^{(i)}, \quad t \in [t_0, t_E]. \tag{6.1}$$

Here, $\mathbf{Y} = (Y_1, \ldots, Y_{n_{sp}}, T)^T$ denotes the vector of reactive scalars, Y_j are the species mass fractions, T represents temperature and ρ is the mixture density. The first n_{sp} entries of the source term vector on the right hand side of Eq. (6.1) represent the species conversion rates, while the final entry corresponds to the calorific temperature source term. For more details on the complete chemical kinetics model which is employed in this work, we refer to the comprehensive Chemkin manual [89].

The reaction fractional step in Eq. (6.1) is sometimes referred to as an ‘embarrassingly parallel’ problem since the individual ODE systems can be solved independently of each other. This partly justifies the application of the operator splitting technique which formally introduces an error proportional to the fractional time step $\Delta t = t_E - t_0$. A second merit of the fractional step formulation is that different solution techniques can be applied during each fractional step and, hence, the solution algorithms are tailored to meet the characteristics of the fractional step submodels.

In the context of RANS/LES schemes, the global time step Δt is commonly chosen according to a CFL condition which links Δt to a characteristics convection time scale of the mean or filtered flow field. Chemical reactions, however, often take place on time scales which are much smaller than Δt, thus introducing stiffness into the ODE systems in Eq. (6.1). In such circumstances, implicit integration schemes are preferred due to their stability properties. The order of the integration scheme, on the other hand, is mainly determined by the required accuracy. In engineering applications for reactive flows, schemes up to order 5 are commonly used.

In the present investigation, we combine an MPI parallelization of the original Radau5 CPU implementation by Hairer and Wanner [67] with the GPU reimplementation presented in Chapter 5. Here, the CPU constitutes the host for the block-structured fluid dynamics software and each MPI process is assigned one block of the spatially decomposed flow domain. During the reaction fractional step, each MPI process first determines the flammable cells in its subdomain based on a temperature criterion (e.g., $T > T_r = 800\,\mathrm{K}$). As standard CPU implementation we consider a program in which each MPI process solves the chemical kinetics ODE systems which are associated with the flammable cells in its own subdomain. Since this may lead to an imbalance in the number of ODE systems solved by each MPI process if the reacting parts of the flame do not occupy equal volumes in each

MPI process' subdomain, we consider different runtime-efficient strategies for distributing a number of ODE systems across the MPI processes in the following section.

6.3 Distribution strategies

The main questions which we encounter in the context of parallelizing the reaction fractional step across a CPU-GPU pair are, first, how many ODE systems ought to be assigned to either processor and, second, in which way the CPU's ODE systems should be distributed across its MPI processes. These questions are related to the task of load balancing, that is, the distribution of work across a number of workers such that the time span between the point in time at which the quickest worker finishes and the one at which the slowest worker finishes is minimized.

In the present section, such load balancing strategies are devised based on estimates for the amount of work associated with each ODE system. In a first approach, we assume that all ODE systems require the same amount of work. This is the basis for the strategies proposed in the next section. Subsequently, the developed approaches are generalized to the case, where each ODE system may involve a different amount of work and this amount of work can be reliably estimated.

6.3.1 ODE systems which behave similarly stiff

In general, the degree of stiffness of a chemical kinetics ODE system is difficult to predict *a priori* and to quantify in absolute terms. However, *a posteriori*, the number of step size refinements which an integration scheme required to solve an ODE system can be employed to construct a relative measure for stiffness. That is, by comparing the numbers of step size refinements for two ODE systems, one ODE system is said to behave stiffer than the other one if it required more step size refinements.

Although, step size refinements thus provide an accessible, relative measure for evaluating stiffness after the integration completed, its quality as an indicator for the amount of work required by an integration scheme varies from integration scheme to integration scheme. In particular, in an implicit integration algorithm, not all integration steps are identical since some involve a reevaluation of the Jacobian or of the left-hand-side matrices of the linear systems while other integration steps skip these tasks. Nevertheless, such tasks often reappear periodically and, thus, the amount of work increases monotonically with the number of step size refinements.

The developments in this section are based on the assumption that all ODE systems behave similarly stiff and, hence, require the same number of step size refinements. Although this assumption may not always hold, in particular for practical applications, it significantly facilitates the design and analysis of load balancing schemes. In the subsequent section, a mapping is introduced which generalizes the strategies developed here to the case in which ODE systems are expected to take varying numbers of step size refinements.

Ideally, the ODE systems are distributed across both the CPU and the GPU such that the runtimes on both processors balance and the resources on each processor are fully utilized. Thus, for a given number of ODE systems N, the task consists in determining $n \leq N$ ODE systems which are to be solved on the CPU such that

$$|t_{gpu}(N-n) - t_{cpu}(n)|^2 = \underset{n}{\text{Min!}}, \tag{6.2}$$

where $t_{gpu}(N-n)$ and $t_{cpu}(n)$ denote the times consumed by integrating $N-n$ ODE systems on the GPU and n ODE systems on the CPU, respectively.

In Eq. (6.2), both t_{gpu} and t_{cpu} need to be estimated. For the runtime on the GPU, we found in Sections 5.5.5 and 5.6.1 that t_{gpu} is well approximated by the following functional form

$$t_{gpu}(n) = \begin{cases} b_{gpu} & \text{for } n \in [0, n_0) \\ a_{gpu} n & \text{else} \end{cases} \tag{6.3}$$

if all ODE systems require the same number of step size refinements. In Eq. (6.3), b_{gpu} and a_{gpu} are two positive constants which represent the times that it takes to integrate one out of n ODE systems on the GPU for $n < n_0$ and $n \geq n_0$, respectively. For the OpenCL-GPU implementation, the threshold value n_0 slightly decreases as the reaction mechanism size is increased.

In practice, a_{gpu} can be determined from past runtime measurements of the GPU integrator on a collection of n ODE systems. Although, for $n \geq n_0$ one time measurement would be sufficient, we found that collecting up to 10 past time measurements with $n \geq n_0$ and computing a_{gpu} by a linear regression yields a more accurate and less fluctuating estimate for a_{gpu}. If at start up an estimate for n_0 is not available, then the implementation might have to wait until the slope predicted by a linear regression in $\log(t_{gpu})$-$\log(n)$-space of the two past sample measurements (or more) with the biggest n-values is sufficiently close to 1. On the other hand, if this slope is small, then n_0 can be updated for the smallest n-value of the stored t_{gpu}-n-pairs. Based on estimates for a_{gpu} and n_0, b_{gpu} is finally computed from $a_{gpu} n_0$.

On the GPU, the ODE systems are assigned automatically to the individual compute units. For a collection in which each ODE system takes the same number of step size refinements, this yields a well-balanced utilization of the compute units such that Eq. (6.3) holds. In the case of the CPU, on the other hand, the ODE systems are manually distributed across the MPI processes. Hence, the runtime on the CPU, t_{cpu}, is equal to the runtime of the MPI process which has been assigned the most ODE systems

$$t_{cpu}(n) = \max_{i=1,\ldots,n_p} t_{mpi}(n_i) = a_{mpi} \max_{i=1,\ldots,n_p} n_i, \tag{6.4}$$

where n_i indicates the number of ODE systems assigned to MPI process i, $n = \sum_{i=1}^{n_p} n_i$

and n_p denotes the number of MPI processes. In Eq. (6.4), t_{mpi} has been taken as a linear function of n_i with slope $a_{mpi} > 0$. This follows from the premise that the work associated with each ODE system is identical and the fact that, on an MPI process, the computation is serialized. Similar to the case of the GPU, a_{mpi} can be determined from runtime measurements.

In order to close Eqs. (6.2), (6.3) and (6.4), the values for n_i, $i = 1, \ldots, n_p$, in Eq. (6.4) need to be expressed in terms of n. This is tantamount to specifying a strategy in which the n ODE systems which have been assigned to the CPU are distributed across the MPI processes. The strategy which is ideal in terms of load balancing in the present context is to distribute the ODE systems uniformly by setting

$$n_i = \mathrm{floor}\left(\frac{n}{n_p}\right) + \begin{cases} 1 & \text{if} \quad \mathrm{mod}\,(n, n_p) \le i \\ 0 & \text{else} \end{cases}, \tag{6.5}$$

where $\mathrm{mod}\,(\cdot,\cdot)$ denotes the modulo-operator and floor$(\cdot)$ returns the biggest integer that is smaller than or equal to the argument value. However, in practice, this might not to be the best strategy since it potentially involves significant communication between the MPI processes on account of the redistribution. At the end of this section, an alternative strategy is proposed which requires less communication, but also attempts to balance the number of ODE systems that are integrated on each MPI process. Although the $n_i(n)$ relationship for this second strategy is recursive and not straightforward to explicate, Eq. (6.5) provides a good estimate for $n \lesssim N/2$. Hence, in this case, Eq. (6.5) can be adopted for both strategies. In the combined CPU/GPU implementation, however, the values for $\max_{i=1,\ldots,n_p} n_i(n)$ are computed exactly and the minimization problem for strategy 2 (Eq. (6.2)) is solved numerically.

By combining Eqs. (6.3), (6.4) and (6.5) with Eq. (6.2), we obtain for the optimal number of ODE systems n that are integrated on the CPU

$$n = \begin{cases} \mathrm{floor}\left(\frac{b_{gpu} n_p}{a_{mpi}}\right) & \text{for } N - n < n_0 \\ \mathrm{floor}\left(\frac{a_{gpu}}{a_{gpu} + a_{mpi}/n_p}\right) N & \text{else} \end{cases}. \tag{6.6}$$

Here, the condition $N - n < n_0$ is tested after one of the two formulas has been applied and if it is not satisfied, then the other formula is used.

Since during the first time steps, estimates for $a_{gpu}, b_{gpu}, a_{mpi}$ and n_0 are not yet available, the strategy for determining n is replaced by a non-adaptive one. For the numerical experiments, $n = \mathrm{floor}(N/2)$ has been set, for instance.

As noted above, there exist different strategies for distributing the CPU's n ODE systems across the MPI processes. Figure 6.1 depicts two such schemes which have been investigated as part of this work. In the first scheme (Figure 6.1(a)), all MPI processes send their ODE systems to the master process which dispatches $N - n$ ODE systems to the GPU and distributes the remaining n ODE systems equally across the MPI processes. After the integration is complete, the master retrieves the results from the GPU and the MPI processes and redistributes the results such that each MPI process receives the results for its own ODE systems. The main drawback of this strategy is that the master process gathers and distributes all ODE systems twice (once before and once after the integration). On the one hand, this involves transferring a large of amount of data repeatedly which may lead to a significant performance loss. On the other hand, the first processor must be able to supply a large enough memory buffer to store all ODE systems for the gather/scatter operations.

Figure 6.1(b) depicts an alternative strategy which seeks to overcome the shortcomings of the first one while maintaining a balanced split of the CPU's ODE systems across the MPI processes. Here, each MPI process only sends a subset of its ODE systems to a master process (MPI process 0) which forwards all the ODE systems that it receives to the GPU. In this strategy, the ODE systems which an MPI process keeps are determined in the following way: The master process marks n ODE systems which are owned by the MPI processes in a round-robin fashion, where an MPI process is skipped if all its ODE systems have been marked. The marked ODE systems then remain on the MPI processes while the unmarked ones are send to the master process.

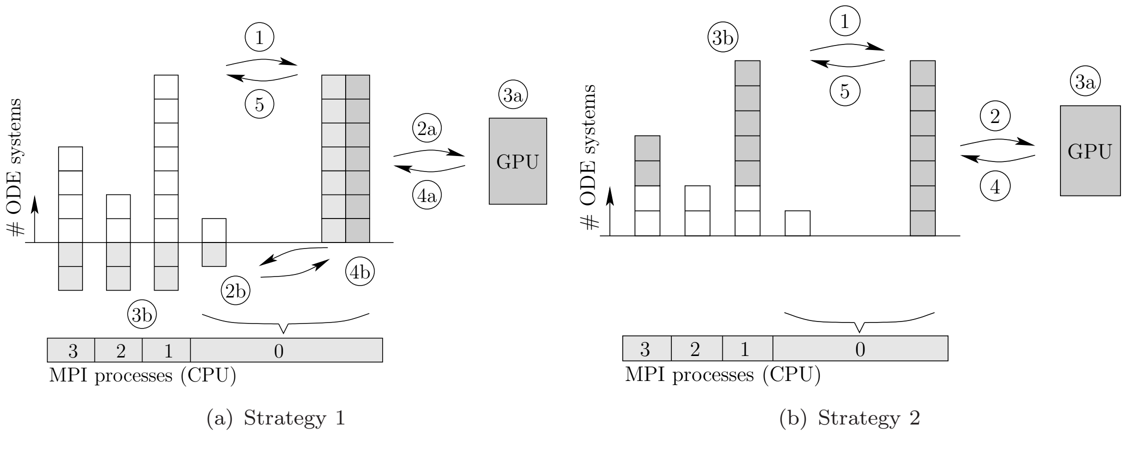
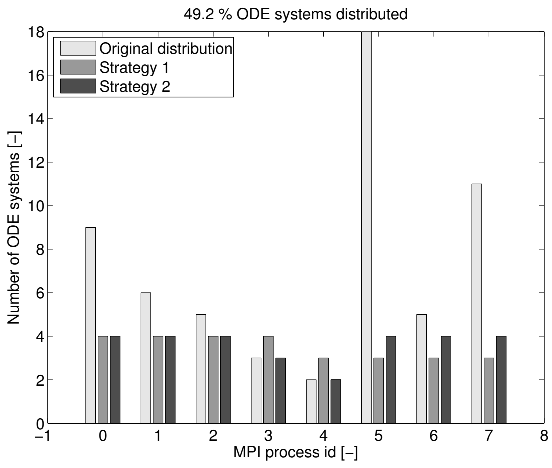
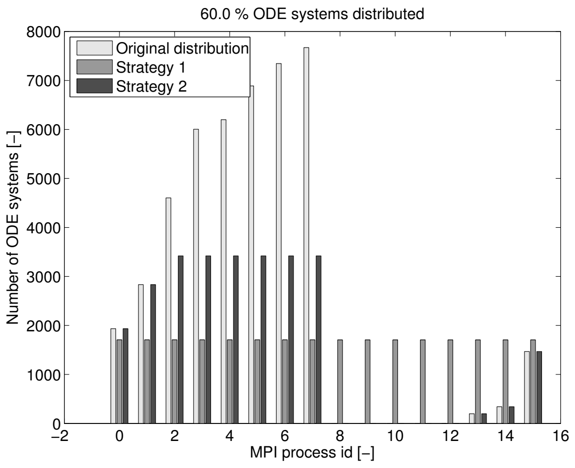


Figure 6.1 Strategies for distributing the chemical kinetics ODE systems across the GPU and the MPI processes. (1: Send data to process 0, 2/2a: Send data to GPU, 2b: Distribute data across MPI processes, 3a/b: Solve, 4/4a: Retrieve data from GPU, 4b: Send data to process 0, 5: Redistribute results)

Both of these distribution strategies are illustrated in Figure 6.2(a) for the case of 8 MPI processes and $n = \text{floor}(N/2)$. Here, the number of ODE systems which each MPI process owns is determined by sampling from a normal distribution with mean and variance 10. Although the initial distribution is thus very inhomogeneous, the second strategy achieves a homogenization of the ODE systems on the MPI processes which comes very close to the one resulting from the first strategy. Figure 6.2(b), on the other hand, depicts the situation for the Sandia D flame in a temporally statistically stationary state and for a domain decomposition involving 16 MPI processes. Here, the scheme for balancing the GPU/CPU runtimes leads to dispatching 40.0 % of the total number of ODE systems to the GPU. However, since some MPI processes do not own any flammable cells in the first place, they remain idle throughout the reaction fractional step.



(a) Normal distribution (mean and variance 10)



(b) Sandia D

Figure 6.2 Sample distributions of ODE systems across 8 and 16 MPI processes (light grey bars) and redistributions for CPU integration resulting from strategy 1 (medium gray bars) and strategy 2 (dark gray bars), respectively.

For the same initial distribution of ODE systems as in Figure 6.2(a), Figure 6.3 depicts the values of $\max_{i=1,\ldots,n_p} n_i$ (Eq. (6.4)) which result from either strategy over the ratio n/N. Since both graphs remain within a distance of ± 1 for $n/N \lesssim 1/2$, this lends some justification to the above analysis in which Eq. (6.5) has been adopted for both strategies.

Finally, we note that other strategies for distributing the CPU's ODE systems across the MPI processes can be conceived of. In particular, it is possible to apply the redistribution strategies to subsets of MPI processes and dispatch one set of ODE systems for GPU integration from each such subset. This would imply, however, that the GPU integration involves multiple kernel invocations on smaller numbers of ODE systems which may potentially result in a loss in occupancy and hence performance on the GPU.

Remark 6.1 (Implementation on Nvidia GPUs)**.** The current Nvidia implementation of the OpenCL function for reading data from the GPU's memory after a kernel invocation blocks until both the kernel execution and the read command completed. Hence, the master process is stalled until the GPU returns. For this reason, we have had to augment the above strategies by the convention that the master process sends all its ODE systems to the GPU whenever $N - n > 0$. Although the master thus remains idle, the impact on the overall performance may be small if the CPU accommodates many hardware threads.

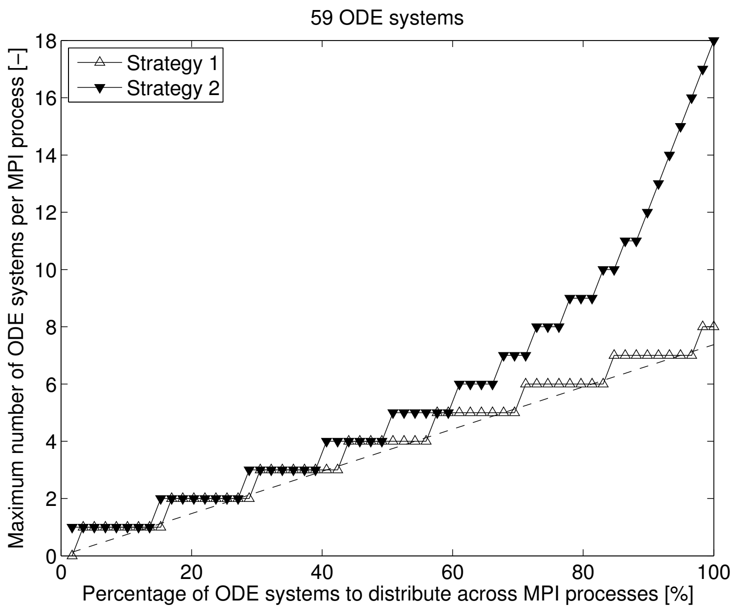


Figure 6.3 Maximum number of ODE systems on an MPI process, $\max_{i=1,\ldots,n_p} n_i$, over the fraction of ODE systems n/N, $n = \sum_{i=1}^{n_p} n_i$, that are integrated on the CPU for distribution strategies 1 and 2. Here, the dashed line corresponds to the relation n/n_p.

6.3.2 Accounting for stiffness

In practice, the initial conditions for the chemical kinetics ODE systems encompass a range of compositions which vary with regard to flammability and stiffness. As noted in the beginning of the previous section, a relative, *a posteriori* measure for stiffness is the number of step size refinements in a given integration algorithm. If an estimate for this number is available, then the amount of work that is associated with an ODE system (relative to a different one) can be predicted. In the present section, we show how this information can be employed within the scope of the distribution strategies introduced in the previous section.

The main idea consists in conceptually replacing one ODE system with k estimated step size refinements by k ODE systems with a single integration step. This replacement might not be accurate since individual integration steps may involve a varying amount of work, but it can be taken as a good estimate for integration schemes in which the work per integration step varies at least periodically. The distribution strategies above are then based upon numbers of integration steps rather than ODE systems. However, when integration steps are transferred between MPI processes or between the CPU and the GPU, these integration steps are moved in blocks, each block corresponding to one ODE system. This can be aided by sorting the ODE systems on each MPI process for the numbers of integration steps. For the first strategy, the sorted ODE systems are then assigned to the MPI processes in a round-robin fashion, while for the second strategy each MPI process transfers the ODE systems with the smallest numbers of step size refinements

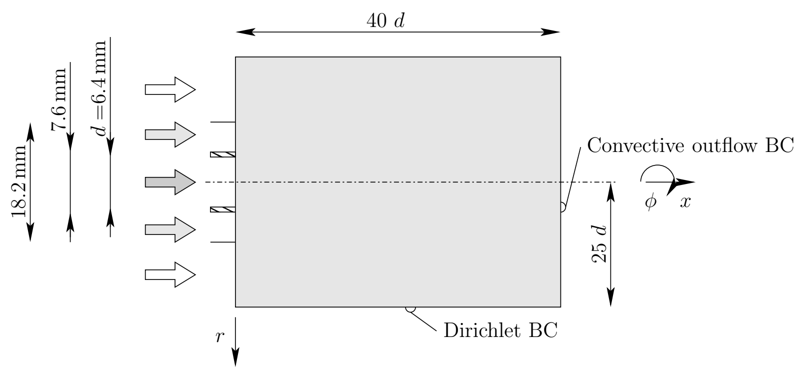


Figure 6.4 Schematic illustration of the computational domain and the boundary conditions for the Sandia D flame.

to the master process. In this way, it might be possible to obtain a closer match with the number of integration steps to be transferred since the size of the blocks in which the integration steps are grouped (i.e., the number of integration steps for the current ODE system) remains close to one, ideally.

One way to estimate the expected number of step size refinements is to store the number of step size refinements which an ODE system associated with a specific grid point required in the previous time step and to employ this number as an estimate for the current time step. This is akin to the criterion devised by Shi et al. [190] who assigned a degree of stiffness to the ODE systems based upon the number of step size refinements in the previous time step. However, the reliability of this estimate does not seem to have been assessed yet.

In this chapter, we focus on the CPU/GPU and CPU/MPI distribution strategies proposed in the previous section and quantify the performance benefits for a realistic test case. The extension of these strategies to account for varying degrees of stiffness is left for future work.

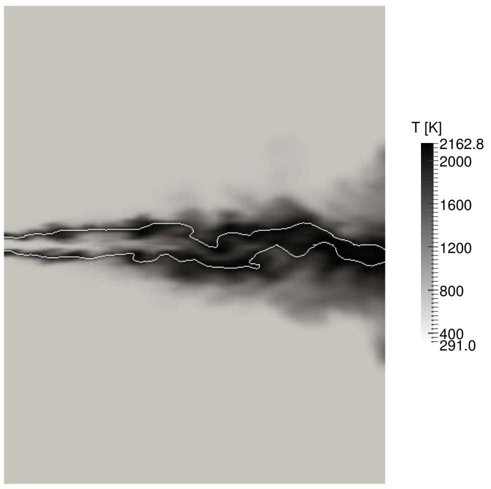
6.4 Numerical experiments

In the present section, the combined CPU/GPU implementation for solving the reaction fractional step is applied to an LES of the Sandia D flame. This flame has been considered by a number of research groups for model validation [85, 133, 156]. It encompasses a methane-air jet and a surrounding annular pilot flow which issue into co-flowing air. The composition of the pilot flame mixture is taken identical to that of a burnt laminar methane-air mixture at equivalence ratio 0.77 [19]. Schematically, the flow geometry and the initial and boundary conditions are depicted in Figure 6.4.

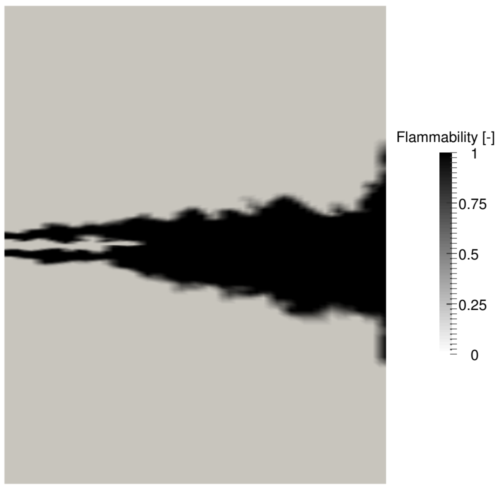
Since our focus lies on assessing the performance of different implementations of a solution scheme for the reaction fractional step, we adopt the simplifying assumption of perfect micromixing for the turbulence-chemistry interaction. The chemical reactions, moreover, are represented by the GRI 1.2 mechanism ($n_s = 32$).

The computations were performed in double precision on an Intel Xeon E5-2687W processor and an Nvidia Quadro 6000 GPU. This is the same pair of high-end devices which we already considered in Section 5.6. The Intel Xeon processor possesses 16 hardware threads (hyperthreading enabled) each of which hosts one MPI process. Accordingly, the flow domain ($56 \times 64 \times 36$ cells) is decomposed into 16 blocks of equal size (8/2 equidistant splits in the axial/radial direction), each block being assigned to one MPI process. The CPU source code was compiled by the Intel Fortran compiler 14.0.0 ("-O 3") in combination with Open MPI 1.6.5, while the OpenCL kernel functions were compiled by the OpenCL compiler that is part of the NVIDIA CUDA Toolkit 5.5 ("-cl-fast-relaxed-math").

Figure 6.5(a) depicts the instantaneous temperature field and the iso-contour of the stoichiometric mixture fraction at a point in time at which the temporal statistics of the flow fields have reached a steady state. For the same time instance, Figure 6.5(b), moreover, shows the instantaneous flammability indicator (= 0 for $T < T_r = 800\,\mathrm{K}$, = 1 otherwise).



(a) Instantaneous temperature field and contour of stoichiometric mixture fraction



(b) Instantaneous flammability indicator

Figure 6.5 Instantaneous temperature field (figure (a)) and flammability indicator (figure (b); 0 non-reacting, 1 reacting) for the Sandia D flame in a temporally statistically stationary state. The white line in figure (a) indicates the stoichiometric mixture fraction contour.

In Table 6.1, the average runtime of the reaction fractional step is compared for the reference CPU implementation and the combined CPU/GPU implementations based on the two load balancing strategies proposed in Section 6.3.1. Also, a CPU implementation in which the ODE systems are uniformly distributed across the MPI processes and a GPU-only implementation are included here. The final column of Table 6.1 indicates the speedup of an implementation relative to the reference CPU implementation in which each

Processor(s)	Strategy	Average runtime [s]	Speedup
CPU	–	8.890	1.000
GPU	–	6.607	1.346
CPU	Redistribute	3.401	2.614
CPU, GPU	Strategy 1	2.471	3.598
CPU, GPU	Strategy 2	4.277	2.078

Table 6.1 Comparing the runtimes of the reaction fractional step for CPU-only, GPU-only and combined CPU/GPU implementations. Here, the average runtime values were computed from time measurements on 10 consecutive time steps ($\Delta t = 10^{-6}$ s).

MPI process solves its own ODE systems (see Section 6.2).

The runtime measurements in Table 6.1 indicate that redistributing the ODE systems uniformly across the MPI processes is a profitable acceleration strategy. For the CPU-only implementation, this yields a speedup of 2.6, while for the combined CPU/GPU implementation a speedup of 3.6 is obtained. Regarding the second distribution strategy, we observe that although it requires significantly less MPI communication and memory allocation/deallocation, it is slightly less efficient than a CPU-only implementation in which the ODE systems are uniformly distributed.

6.5 Chapter summary

In this chapter, we presented a combined CPU/GPU implementation for solving the reaction fractional step using the 5th order accurate implicit Runge-Kutta algorithm Radau5 and assessed the performance of this implementation in the context of an LES of the Sandia D flame. The combined CPU/GPU implementation involves a scheme for measuring the runtimes on both the GPU and the CPU and adjusting the distribution of the chemical kinetics ODE systems across the GPU and the CPU's MPI processes such that the runtimes on both processors match.

In this regard, we investigated two distribution strategies. In the first one, all MPI processes communicate their proprietary ODE systems to a master process which sends a fraction to the GPU and distributes the remaining ODE systems equally across the MPI processes. This strategy has the disadvantage that the master process requires a large amount of memory in order to store the flammable ODE systems for the whole domain. In the case of the Sandia D flame, the number of reacting cells amounts to about 35 % of the total number of cells, such that for the GRI 1.2 reaction mechanism (32 scalars) and the finite volume mesh employed above (129,024 cells) the master process allocates 11.7 MB of memory. Furthermore, this strategy involves repeated all-to-one and one-to-all MPI communication. At present, we are therefore devising an implementation in which the communication among the MPI processes is pairwise, that is, given an average number of ODE systems per MPI process, surplus ODE systems are communicated either directly to another MPI process with free capacity or to the master for forwarding to the GPU.

In the second distribution strategy, the MPI processes solve a fraction (or all) of the ODE systems which they own and dispatch surplus ODE systems to the master process which sends them to the GPU. In this way, the disadvantages of the first strategy are mitigated, although a balanced distribution only results when the fraction of ODE systems which is dispatched to the GPU is large.

For a reaction step in a temporally statistically steady state, we found the first strategy to perform 1.7 times better than the second one. In comparison to a standard CPU implementation in which each MPI process solves its own ODE systems, the combined CPU/GPU implementations showed a speedup of 3.6 and 2.1 for strategies one and two, respectively. On the other hand, if the performance is compared with a CPU implementation in which all ODE systems are equally distributed across the MPI processes, then only the first strategy leads to a speedup (1.4), while the second strategy is 1.3 times slower.

In the above, we confined the attention to the case in which all ODE systems exhibit a similar degree of stiffness. However, the formalism on which the current distribution strategies are based can be extended to a collection of ODE systems which behave differently stiff. Although we have outlined the procedure for this, the implementation is left for future work.

Chapter 7

Conclusions

7.1 Summary

In this work, we presented a comprehensive modelling approach and an efficient numerical solution strategy for predicting the formation of a polydispersed particulate phase in a turbulent reacting carrier flow. For particles which are polydispersed with respect to a measure of particle size, our focus lay on resolving the change in the particle size distribution through the flow domain and over time due to the physical processes of spatial transport, turbulent mixing, nucleation and growth. The applications which we considered as validation test cases in the previous chapters ranged from turbulent precipitation and droplet condensation in a turbulent mixing layer to soot formation in a turbulent hydrocarbon flame. For these examples, we demonstrated the predictive capabilities of our model and assessed the accuracy and computational efficiency of the numerical solution scheme.

In line with recent progress in the development of LES, we adopted the LES concept as underlying model for a turbulent flow. The immersed particulate phase was described in terms of the particle number density whose Eulerian evolution in both physical and particle size space is governed by the PBE. The scalars characterizing the carrier fluid, moreover, obey convection/diffusion/reaction equations. Incorporating the PBE and the reactive scalars transport equations into an LES-framework engenders three main challenges which we addressed in this work. First, a difficulty in the numerical solution of the PBE by direct discretization approaches is caused by the observations that the particle size distribution frequently evolves over several orders of magnitude and may develop sharp features since there is no physical mechanism for diffusion in particle size space. Second, applying the LES operator to the PBE and the reactive scalars transport equations leads to unclosed number density-scalars correlations which, physically, represent the influence of turbulence on chemical reactions and particle formation. Finally, the chemical kinetics often exhibit severe stiffness, requiring computationally expensive implicit time integration schemes of higher order. In the following, we briefly summarize the methods that have been developed within the scope of the present work to address these issues.

With regard to the first challenge, we devised a novel explicit adaptive grid approach for solving the PBE. This technique is based on a space- and time-dependent coordinate transformation on particle size space that is explicitly marched in time. Here, the evolution of the coordinate transformation during the upcoming time step is controlled by the current coordinate transformation and the current solution for the number density distribution. The coordinate transformation can be viewed as a redistributor in physical particle size space of the available resolution (supplied via a uniform reference grid in transformed particle size space) such that, for example, sharp features of the particle size distribution are allocated a larger measure of resolution than smooth features. For the construction of the coordinate transformation, we augmented the established equidistribution principle by a novel approach for controlling the node density distribution in the presence of conditions on the minimum admissible resolution and a maximum grid stretching. The combined scheme allows for the generation of high quality adaptive grids, while adhering to grid parameters that similarly occur in the characterization of fixed grids.

The convergence, accuracy and computational efficiency of our adaptive grid scheme were thoroughly assessed in the context of a pure advection problem and a kinetically realistic example describing the nucleation and growth of $BaSO_4$ in a plug flow reactor. For a prescribed accuracy, we observed that the explicit adaptive grid approach requires significantly (more than an order of magnitude) fewer grid points than comparable fixed grid solution schemes. This also affected the solver runtime which we found to decrease by approximately one order of magnitude when grid adaptivity was active.

The second part of this work focussed on the LES-PBE-PDF framework for modelling turbulent reacting flows with polydispersed particle formation. Conceptually, the main idea consisted in obtaining from the PBE and the reactive scalars transport equations an evolution equation for the LES-filtered one-point, one-time *pdf* associated with a single realization of the fluid composition and particle number density distribution. Here, turbulent transport effects were closed by invoking a gradient diffusion hypothesis and, for molecular mixing, we adapted a recently proposed model to account for differential diffusion between the fluid and the particle phases. Contrary to many existing modelling approaches, the LES-PBE-PDF formulation is independent of tracking scalars such as a mixture fraction or a reaction progress variable which are often linked to particular flow configurations, and can thus be applied without modification to very general flow devices and reactors. Moreover, the LES-filtered particle size distribution is predicted at each location in the flow domain and every time instant and both chemical and particle formation kinetics are accommodated without approximation.

Based on the joint scalar-number density *pdf*, a statistically equivalent system of Eulerian stochastic fields was presented, forming the basis for a Monte-Carlo type numerical solution scheme. By this scheme, we were able to concentrate the computational effort towards the accurate estimation of low order moments of the *pdf* such as the LES-filtered fluid composition or particle size distribution, while accounting for the dynamic evolution in physical and particle size space of the entire *pdf*. The evolution equation for the stochastic number density field was discretized using the explicit adaptive grid scheme introduced above.

In Chapter 3, the LES-PBE-PDF approach and the corresponding stochastic field reformulation were first presented for incompressible, constant density flows. The examples we considered here encompassed the precipitation of $BaSO_4$ particles from ionic aqueous solutions in a coaxial pipe mixing device and the condensation of DBP droplets in the fully turbulent part of a spatially developing mixing layer. Subsequently, in Chapter 4, the LES-PBE-PDF approach was extended to low Mach number, variable density flows and applied to model soot formation in a turbulent hydrocarbon flame. Although our physical description was confined to nucleation and growth of primary soot particles, quantitatively reasonable predictions of soot volume fraction were obtained.

In the third part of this work, we addressed the question whether the chemical kinetics integration using a high order implicit Runge-Kutta scheme can be accelerated by implementing such an integration scheme for execution on a graphics card (GPU). In contrast to a CPU, a GPU consists of several blocks of processing elements which are orchestrated, on each block, by a single instruction unit; also, large memory buffers are located off-chip. By consequence, the execution and memory models of APIs for developing software on GPUs largely differ from those of CPUs, exposing to the software developer the GPU architecture. Based on several strategies for distributing tasks across a GPU, aligning memory accesses and efficiently communicating intermediate results, we developed a GPU reimplementation of the 5th order accurate implicit Runge-Kutta solver Radau5.

The performance of the GPU reimplementation was compared with that of an MPI parallelization of the original Fortran 77 Radau5 implementation on both a user-end GPU/CPU and a professional level GPU/CPU for initial conditions sampled from a transient equilibrium scheme for the flamelet model. In a comprehensive analysis we included several reaction mechanisms and considered a range of problem sizes (number of grid points), time steps and convergence tolerances. Our findings indicated that the GPU reimplementation performs well even for small problems and that, on a high-end GPU, the relative performance of GPU and MPI-CPU implementation is largely independent of the reaction mechanism size. However, at best, the GPU implementation took about two times longer to execute than its MPI-CPU counterpart. We believe that this is mainly due to the frequent communication and synchronizations on the GPU that occur in the direct solution of linear systems and in global reduction schemes.

Finally, we explored strategies for parallelizing the computationally intensive reaction fractional step across a CPU-GPU pair in a desktop workstation. Considering a turbulent, non-premixed jet flame, we found an implementation in which the chemical kinetics systems were equally distributed across the compute units of a CPU and the GPU, while minimizing idle times of either processor, to be the most efficient—despite the significant reallocation of memory and data migration. This implementation, however, can be further improved and work is in progress on restricting the repeated all-to-one and one-to-all communication to pair-wise data exchange.

In summary, we have presented, in this work, a comprehensive LES-based model for predicting the evolution of the size distribution associated with a polydispersed particulate phase in a turbulent carrier flow. Complementarily, an accurate grid-adaptive discretization scheme was developed and the question whether the chemical reaction step can be accelerated by deferral to a graphics card or by parallelization across a CPU-GPU pair was investigated. Our model and numerical solution scheme are general by nature and can be applied to investigate processes as different as precipitation, condensation, nano-particle synthesis, emulsification or soot formation. In the following section, some potential model enhancements are outlined for future consideration and suggestions for further validation are given.

7.2 Outlook

In this section, a few ideas for extending the LES-PBE-PDF approach towards flows in which the immersed particulate phase is characterized by more than one characteristic property and for validating the model more extensively are collected. For each point, we briefly summarize the main motivation, rationale and projected outcomes.

Turbulent precipitation. Our first validation test case in Chapter 3, the precipitation of $BaSO_4$ in a coaxial pipe mixing device [18], was chosen for two main reasons. First, the kinetics of $BaSO_4$ nucleation and growth as well as the thermodynamic properties of ionic aqueous solutions can be reliably described by (semi-)empirical or analytical expressions. Moreover, previous investigators [18, 40] have been able to predict the particle size distribution at the outlet of the coaxial pipe mixer very well in the context of RANS using different approaches for the turbulence-chemistry/particle formation interaction and the parameterization of the particle size distribution. Thus, we had some confidence in the reliability of the experimental measurements and adopted the first test case both for model validation and for verifying the implementation of the numerical solution scheme. In view of the excellent earlier RANS predictions, however, it may be argued that there is little reason to apply an LES-based model here. Adhering to precipitation, we hence propose to apply the LES-PBE-PDF method to precipitation devices with more intricate flow patterns such as impinging jet or cross-flow reactors [113].

Soot particle coagulation and aggregation. Nascent soot particles have been found to possess a liquid-like consistency until carbonization occurs [29]. Thus, young soot particles may coagulate similar to droplets or merge with carbonized primary particles [130], maintaining an approximately spherical shape. Although coagulation was not considered in our work, the coagulation source terms can be readily included in the present formulation. The process of soot particle aggregation, by contrast, is more challenging to incorporate. Here, mature carbonized soot particles assemble in chain like structures. In order to describe such aggregates, more than one particle property is needed, for instance, a primary particle size and a property for the fractal dimension of an aggregate or its number of primary particles. In this regard, the explicit adaptive grid discretization could be generalized to a higher dimensional particle property space. Alternatively, maintaining the information on the primary particle size distribution, one idea would be to apply a moment transformation along the additional particle property only or, if possible, to invoke an assumption on the shape of the additional property distribution conditioned on primary particle size based on experimental observations.

Inertial particles. In some applications of industrial relevance such as spray drying or liquid spray combustion, the immersed particles or droplets experience significant lift and drag forces and their trajectories, hence, do not coincide with pathlines of the carrier flow. Within the scope of a population balance modelling approach, this implies that the particulate phase is characterized both by a measure of particle size and a proprietary velocity vector, yielding a PBE with four internal coordinates. The momentum exchange between the particles and the ambient fluid is governed by the local fluid velocity 'seen' by the particles. For the PBE formulated in terms of a joint velocity-size property space, it may be possible to reduce the number of independent coordinates by applying moment methods along the velocity coordinates or by exploiting relations between particle size, velocity and momentum exchange terms. However, the validity and benefits of such an approach are not yet clear. In the context of turbulent flows, many modelling approaches are, at the moment, based on the Lagrangian equations of motion of individual particles [173].

Overlapping the reaction and PBE fractional steps. If the particle formation kinetics exhibit a weak dependence on the fluid phase composition, it may be possible to solve the PBE fractional step on the CPU while the chemical reaction step is in progress on the GPU. While a formal justification for this approach is yet missing, it would provide further opportunity for parallelizing and accelerating the solution scheme developed in this work.

Bibliography

[1] P. Akridis, "Coupled CFD-Population Balance Modelling of Soot Formation in Laminar and Turbulent Flames", PhD thesis, Imperial College London, 2016.

[2] P. Akridis and S. Rigopoulos, "Modelling of soot formation in a turbulent diffusion flame using a comprehensive CFD-PBE model with full chemistry", *Proceedings of the European Combustion Meeting*, Budapest, Hungary, 2015.

[3] P. Akridis and S. Rigopoulos, "Modelling of soot formation in laminar diffusion flames using a comprehensive CFD-PBE model with detailed gas-phase chemistry", *Combustion Theory and Modelling* 21.1 (2017), pp. 35–48.

[4] J. Akroyd, A. J. Smith, R. Shirley, L. R. McGlashan, and M. Kraft, "A coupled CFD-population balance approach for nanoparticle synthesis in turbulent reacting flows", *Chemical Engineering Science* 66.17 (2011), pp. 3792–3805.

[5] A. H. Alexopoulos, A. I. Roussos, and C. Kiparissides, "Part I: Dynamic evolution of the particle size distribution in particulate processes undergoing combined particle growth and aggregation", *Chemical Engineering Science* 59.24 (2004), pp. 5751–5769.

[6] A. H. Alexopoulos, A. Roussos, and C. Kiparissides, "Part V: Dynamic evolution of the multivariate particle size distribution undergoing combined particle growth and aggregation", *Chemical Engineering Science* 64.14 (2009), pp. 3260–3269.

[7] A. H. Alexopoulos and C. A. Kiparissides, "Part II: Dynamic evolution of the particle size distribution in particulate processes undergoing simultaneous particle nucleation, growth and aggregation", *Chemical Engineering Science* 60.15 (2005), pp. 4157–4169.

[8] M. Aoun, E. Plasari, R. David, and J. Villermaux, "A simultaneous determination of nucleation and growth rates from batch spontaneous precipitation", *Chemical Engineering Science* 54.9 (1999), pp. 1161–1180.

[9] M. M. Attarakih, H.-J. Bart, and N. M. Faqir, "Optimal moving and fixed grids for the solution of discretized population balances in batch and continuous systems: droplet breakage", *Chemical Engineering Science* 58.7 (2003), pp. 1251–1269.

[10] A. Attili and F. Bisetti, “Statistics and scaling of turbulence in a spatially developing mixing layer at $Re_\lambda = 250$”, *Physics of Fluids* 24.3 (2012), p. 035109.

[11] A. Attili, F. Bisetti, M. E. Mueller, and H. Pitsch, “Formation, growth, and transport of soot in a three-dimensional turbulent non-premixed jet flame”, *Combustion and Flame* 161.7 (2014), pp. 1849–1865.

[12] S. Ayache and E. Mastorakos, “Conditional moment closure/large eddy simulation of the Delft-III natural gas non-premixed jet flame”, *Flow, Turbulence and Combustion* 88.1-2 (2012), pp. 207–231.

[13] X. S. Bai, M. Balthasar, F. Mauss, and L. Fuchs, “Detailed soot modeling in turbulent jet diffusion flames”, *Symposium (International) on Combustion* 27.1 (1998), pp. 1623–1630.

[14] J. Bałdyga and W. Orciuch, “Closure problem for precipitation”, *Chemical Engineering Research and Design* 75.2 (1997), pp. 160–170.

[15] J. Bałdyga, M Jasińska, and W. Orciuch, “Barium sulphate agglomeration in a pipe – An experimental study and CFD modeling”, *Chemical Engineering & Technology* 26.3 (2003), pp. 334–340.

[16] J. Bałdyga, M. Jasińska, A. Krasiński, and A. Rożeń, “Effects of fine scale turbulent flow and mixing in agglomerative precipitation”, *Chemical Engineering & Technology* 27.3 (2004), pp. 315–323.

[17] J. Bałdyga, “A closure model for homogeneous chemical reactions”, *Chemical Engineering Science* 49.12 (1994), pp. 1985–2003.

[18] J. Bałdyga and W. Orciuch, “Barium sulphate precipitation in a pipe − an experimental study and CFD modelling”, *Chemical Engineering Science* 56.7 (2001), pp. 2435–2444.

[19] R. Barlow and J. Frank, *Piloted CH_4/Air Flames C, D, E, and F - Release 2.1*, 2.1, Sandia National Laboratories, Livermore, CA, USA, 2007.

[20] R. J. Batterham, J. S. Hall, and G. Barton, “Pelletizing kinetics and simulation of full scale balling circuits”, *Proceedings of the 3rd International Symposium on Agglomeration*, Nürnberg, W. Germany, 1981, A136–A150.

[21] F. Beenken, S. Ulmer, and F. Joos, “Parallel computing of chemical reactions on graphic processing units”, *Proceedings of the European Combustion Meeting*, Lund, Sweden, 2013.

[22] F. Bisetti, A. Attili, and H. Pitsch, “Advancing predictive models for particulate formation in turbulent flames via massively parallel direct numerical simulations”, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372.2022 (2014).

[23] F. Bisetti, G. Blanquart, M. E. Mueller, and H. Pitsch, “On the formation and early evolution of soot in turbulent nonpremixed flames”, *Combustion and Flame* 159.1 (2012), pp. 317–335.

[24] J. G. Blom and J. G. Verwer, *On the use of the arclength and curvature monitor in a moving-grid method which is based on the method of lines*, tech. rep. NM-N8902, Centre for Mathematics and Computer Science, 1989, pp. 1–15.

[25] C. de Boor, “Good approximation by splines with variable knots. II”, *Conference on the Numerical Solution of Differential Equations: Dundee 1973*, ed. by G. A. Watson, vol. 363, Lecture Notes in Mathematics, Berlin, Heidelberg: Springer-Verlag, 1974, pp. 12–20.

[26] N. W. Bressloff, J. B. Moss, and P. A. Rubini, “CFD prediction of coupled radiation heat transfer and soot production in turbulent flames”, *Symposium (International) on Combustion* 26.2 (1996), pp. 2379–2386.

[27] L. A. Bromley, “Thermodynamic properties of strong electrolytes in aqueous solutions”, *AIChE Journal* 19.2 (1973), pp. 313–320.

[28] S. J. Brookes and J. B. Moss, “Predictions of soot and thermal radiation properties in confined turbulent jet diffusion flames”, *Combustion and Flame* 116.4 (1999), pp. 486–503.

[29] J. Cain, A. Laskin, M. R. Kholghy, M. J. Thomson, and H. Wang, “Molecular characterization of organic content of soot along the centerline of a coflow diffusion flame”, *Physical Chemistry Chemical Physics* 16.47 (2014), pp. 25862–25875.

[30] F. B. Campos and P. L. C. Lage, “A numerical method for solving the transient multidimensional population balance equation using an Euler-Lagrange formulation”, *Chemical Engineering Science* 58.12 (2003), pp. 2725–2744.

[31] J. Cheng, C. Yang, and Z.-S. Mao, “CFD-PBE simulation of premixed continuous precipitation incorporating nucleation, growth and aggregation in a stirred tank with multi-class method”, *Chemical Engineering Science* 68.1 (2012), pp. 469–480.

[32] H. Choi and P. Moin, “Effects of the computational time step on numerical solutions of turbulent flow”, *Journal of Computational Physics* 113.1 (1994), pp. 1–4.

[33] B. C. Connelly, B. A. V. Bennett, M. D. Smooke, and M. B. Long, “A paradigm shift in the interaction of experiments and computations in combustion research”, *Proceedings of the Combustion Institute* 32.1 (2009), pp. 879–886.

[34] T. J. Crowley, E. S. Meadows, and F. J. Doyle III, “Numerical issues in solving population balance equations for particle size distribution control in emulsion polymerization”, *Proceedings of the American Control Conference*, vol. 2, San Diego, CA, USA, 1999, pp. 1138–1142.

[35] *CUBLAS Library User Guide*, NVIDIA Corporation, 2012.

[36] W. Dahmen and A. Reusken, *Numerik für Ingenieure und Naturwissenschaftler*, 2nd ed., Springer-Lehrbuch, Berlin: Springer-Verlag, 2008.

[37] S. F. Davis and J. E. Flaherty, “An adaptive finite element method for initial-boundary value problems for partial differential equations”, *SIAM Journal on Scientific and Statistical Computing* 3.1 (1982), pp. 6–27.

[38] G. Y. Di Veroli and S. Rigopoulos, “A study of turbulence-chemistry interaction in reactive precipitation via a Population Balance-transported PDF method”, *Proceedings of the sixth International Symposium on Turbulence, Heat and Mass Transfer*, ed. by K. Hanjalić, Y. Nagano, and S. Jakirlić, Rome, Italy: Begell House, Inc., 2009.

[39] G. Y. Di Veroli and S. Rigopoulos, “Modeling of aerosol formation in a turbulent jet with the transported population balance equation-probability density function approach”, *Physics of Fluids* 23.4, 043305 (2011), p. 043305.

[40] G. Y. Di Veroli and S. Rigopoulos, “Modeling of turbulent precipitation: A transported population balance-PDF method”, *AIChE Journal* 56.4 (2010), pp. 878–892.

[41] I. A. Dodoulas and S. Navarro-Martinez, “Analysis of extinction in a non-premixed turbulent flame using large eddy simulation and the chemical explosion mode analysis”, *Combustion Theory and Modelling* 19.1 (2015), pp. 107–129.

[42] P. Donde, V. Raman, M. E. Mueller, and H. Pitsch, “LES/PDF based modeling of soot-turbulence interactions in turbulent flames”, *Proceedings of the Combustion Institute* 34.1 (2013), pp. 1183–1192.

[43] E. A. Dorfi and L. O. Drury, “Simple adaptive grids for 1-D initial value problems”, *Journal of Computational Physics* 69.1 (1987), pp. 175–195.

[44] B. P. M. Duarte and C. M. S. G. Baptista, “Moving finite elements method applied to dynamic population balance equations”, *AIChE Journal* 54.3 (2008), pp. 673–692.

[45] B. P. M. Duarte and C. M. S. G. Baptista, “Using moving finite elements method to solve population balance equations comprising breakage terms”, *17th European Symposium on Computer Aided Process Engineering*, ed. by V. Pleşu and P. Ş. Agachi, vol. 24, Elsevier, 2007, pp. 255–260.

[46] L. Falk and E. Schaer, “A PDF modelling of precipitation reactors”, *Chemical Engineering Science* 56.7 (2001), pp. 2445–2457.

[47] J. Fang, A. L. Varbanescu, and H. Sips, “A comprehensive performance comparison of CUDA and OpenCL”, *International Conference on Parallel Processing (ICPP)*, Taipei, Taiwan, 2011, pp. 216–225.

[48] K. Farrell, "Explicit and adaptive grid methods for implicit conservative gas dynamics in one-dimension", *Irish Astronomical Journal* 23.2 (1996), pp. 165–170.

[49] K. Farrell and L. O. Drury, "An explicit, adaptive grid algorithm for one-dimensional initial value problems", *Applied Numerical Mathematics* 26.1-2 (1998), pp. 3–12.

[50] R. O. Fox, *Computational Models for Turbulent Reacting Flows*, Cambridge University Press, 2003.

[51] R. O. Fox, "On the relationship between Lagrangian micromixing models and computational fluid dynamics", *Chemical Engineering and Processing: Process Intensification* 37.6 (1998), pp. 521–535.

[52] L. L. C. Franke, A. K. Chatzopoulos, and S. Rigopoulos, "Tabulation of combustion chemistry via Artificial Neural Networks (ANNs): Methodology and application to LES-PDF simulation of Sydney flame L", *Combustion and Flame* 185 (2017), pp. 245–260.

[53] M. Frenklach, H. Wang, M. Goldenberg, G. P. Smith, D. Golden, C. T. Bowman, R. K. Hanson, W. C. Gardiner, and V. Lissianski, *GRI-Mech–An Optimized Detailed Chemical Reaction Mechanism for Methane Combustion*, Topical Report No. GRI-95/0058, Gas Research Institute, 1995.

[54] M. Frenklach et al., *GRI-Mech 1.2*, 1995, URL: `http://www.me.berkeley.edu/gri\_mech/`.

[55] M. Frenklach, "Method of moments with interpolative closure", *Chemical Engineering Science* 57.12 (2002), pp. 2229–2239.

[56] S. K. Friedlander, *Smoke, Dust, and Haze*, vol. 198, New York: Oxford University Press, 2000.

[57] F. Gao and E. E. O'Brien, "A large-eddy simulation scheme for turbulent reacting flows", *Physics of Fluids A: Fluid Dynamics* 5 (1993), pp. 1282–1284.

[58] C. E. Garcia-Gonzalez, F. Sewerin, A. Liu, S. Rigopoulos, and B. A. O. Williams, "Predicting and measuring soot formation and particle size distributions in a laminar diffusion flame", *European Combustion Meeting*, Dubrovnik, Croatia, 2017.

[59] A. Garmory and E. Mastorakos, "Aerosol nucleation and growth in a turbulent jet using the stochastic fields method", *Chemical Engineering Science* 63.16 (2008), pp. 4078–4089.

[60] E. Gavi, L. Rivautella, D. L. Marchisio, M. Vanni, A. A. Barresi, and G. Baldi, "CFD modelling of nano-particle precipitation in confined impinging jet reactors", *Chemical Engineering Research and Design* 85.5 (2007), pp. 735–744.

[61] F. Gelbard and J. H. Seinfeld, "Numerical solution of the dynamic equation for particulate systems", *Journal of Computational Physics* 28.3 (1978), pp. 357–375.

[62] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, “A dynamic subgridscale eddy viscosity model”, *Physics of Fluids A: Fluid Dynamics* 3.7 (1991), pp. 1760–1765.

[63] J. P. Gore, U.-S. Ip, and Y. R. Sivathanu, “Coupled structure and radiation analysis of acetylene/air flames”, *Journal of Heat Transfer* 114 (1992), pp. 487–493.

[64] J. Gradl, *Experimentelle und theoretische Untersuchungen der Bildungskinetik diffusions- sowie reaktionslimitierter Systeme am Beispiel der Nanopartikelfällung von Bariumsulfat und Zinkoxid*, Göttingen: Cuvillier Verlag, 2010.

[65] D. Grosschmidt, P. Habisreuther, and H. Bockhorn, “Calculation of the size distribution function of soot particles in turbulent diffusion flames”, *Proceedings of the Combustion Institute* 31.1 (2007), pp. 657–665.

[66] W. L. Grosshandler, *RADCAL: A Narrow-Band Model for Radiation Calculations in a Combustion Environment*, NIST Technical Note 1402, National Institute of Standards and Technology, 1993.

[67] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, vol. 8, Springer Series in Computational Mathematics, Berlin: Springer-Verlag, 1991.

[68] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, vol. 8, Springer Series in Computational Mathematics, Berlin, Heidelberg: Springer-Verlag, 1993.

[69] R. J. Hall, M. D. Smooke, and M. B. Colket, “Predictions of soot dynamics in opposed jet diffusion flames”, *Physical and Chemical Aspects of Combustion: A Tribute to Irvin Glassman*, ed. by R. F. Sawyer and F. L. Dryer, Langhorne, PA: Combustion Science and Technology Book Series, Gordon & Breach, 1997, pp. 189–230.

[70] S. J. Harris and A. M. Weiner, “Surface growth of soot particles in premixed ethylene/air flames”, *Combustion Science and Technology* 31.3-4 (1983), pp. 155–167.

[71] G. Hauke and L. Valiño, “Computing reactive flows with a field Monte Carlo formulation and multi-scale methods”, *Computer Methods in Applied Mechanics and Engineering* 193.15-16 (2004), pp. 1455–1470.

[72] D. C. Haworth, “Progress in probability density function methods for turbulent reacting flows”, *Progress in Energy and Combustion Science* 36.2 (2010), pp. 168–259.

[73] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th, Morgan Kaufmann/Elsevier, 2012.

[74] M. J. Hounslow, R. L. Ryall, and V. R. Marshall, “A discretized population balance for nucleation, growth, and aggregation”, *AIChE Journal* 34.11 (1988), pp. 1821–1832.

[75] *http://www.sandia.gov/TNF/radiation.html*, 2003.

[76] W. Huang, Y. Ren, and R. Russell, “Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle”, *SIAM Journal on Numerical Analysis* 31.3 (1994), pp. 709–730.

[77] H. M. Hulburt and S. Katz, “Some problems in particle technology: A statistical mechanical formulation”, *Chemical Engineering Science* 19.8 (1964), pp. 555–574.

[78] J. R. Humphrey, D. K. Price, K. E. Spagnoli, A. L. Paolini, and E. J. Kelmelis, “CULA: Hybrid GPU accelerated linear algebra routines”, *SPIE Defense and Security Symposium (DSS)*, Orlando, FL, USA, 2010.

[79] A. Hunt, J. L. Abraham, B. Judson, and C. L. Berry, “Toxicologic and epidemiologic clues from the characterization of the 1952 London smog fine particulate matter in archival autopsy lung tissues”, *Environmental Health Perspectives* 111.9 (2003), pp. 1209–1214.

[80] R. Hyde, *Write Great Code: Understanding the Machine*, San Francisco, CA, USA: No Starch Press, 2004.

[81] R. Hyde, *Write Great Code Vol. 2: Thinking Low-Level, Writing High-Level*, San Francisco, CA, USA: No Starch Press, 2006.

[82] *Intel SDK for OpenCL Applications 2013 - Optimization Guide*, Document Number: 326542-003US, Intel Corporation, 2013.

[83] W. P. Jones, “The joint scalar probability density function method”, *Closure Strategies for Turbulent and Transitional Flows*, ed. by B. E. Launder and N. Sandham, Cambridge University Press, 2002, chap. 20, pp. 582–625.

[84] W. P. Jones and S. Navarro-Martinez, “Large eddy simulation of autoignition with a subgrid probability density function method”, *Combustion and Flame* 150.3 (2007), pp. 170–187.

[85] W. P. Jones and V. N. Prasad, “Large eddy simulation of the Sandia Flame Series (D-F) using the Eulerian stochastic field method”, *Combustion and Flame* 157.9 (2010), pp. 1621–1636.

[86] W. P. Jones and V. N. Prasad, “LES-PDF simulation of a spark ignited turbulent methane jet”, *Proceedings of the Combustion Institute* 33.1 (2011), pp. 1355–1363.

[87] W. P. Jones, F. di Mare, and A. J. Marquis, *LES-BOFFIN: User’s Guide*, Imperial College London, Department of Mechanical Engineering, London, 2002.

[88] J. Kautsky and N. K. Nichols, "Equidistributing meshes with constraints", *SIAM Journal on Scientific and Statistical Computing* 1.4 (1980), pp. 499–511.

[89] R. J. Kee, F. M. Rupley, E. Meeks, and J. A. Miller, *CHEMKIN-III: A FORTRAN chemical kinetics package for the analysis of gas-phase chemical and plasma kinetics*, Sandia National Laboratories Livermore, CA, USA, 1996.

[90] P. Koniavitis, S. Rigopoulos, and W. P. Jones, "A methodology for derivation of RCCE-reduced mechanisms via CSP", *Combustion and Flame* 183 (2017), pp. 126–143.

[91] H. Koo, M. Hassanaly, V. Raman, M. E. Mueller, and K. P. Geigle, "Large-eddy simulation of soot formation in a model gas turbine combustor", *Journal of Engineering for Gas Turbines and Power* 139.3 (2016), p. 031503.

[92] B. Koren, "A robust upwind discretization method for advection, diffusion and source terms", *Numerical Methods for Advection-Diffusion Problems*, ed. by C. B. Vreugdenhil and B. Koren, vol. 45, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Vieweg Verlag, 1993, pp. 117–138.

[93] A. Kronenburg, R. Bilger, and J. H. Kent, "Modeling soot formation in turbulent methane-air jet diffusion flames", *Combustion and Flame* 121.1-2 (2000), pp. 24–40.

[94] S. Kumar and D. Ramkrishna, "On the solution of population balance equations by discretization – I. A fixed pivot technique", *Chemical Engineering Science* 51.8 (1996), pp. 1311–1332.

[95] S. Kumar and D. Ramkrishna, "On the solution of population balance equations by discretization – II. A moving pivot technique", *Chemical Engineering Science* 51.8 (1996), pp. 1333–1342.

[96] S. Kumar and D. Ramkrishna, "On the solution of population balance equations by discretization – III. Nucleation, growth and aggregation of particles", *Chemical Engineering Science* 52.24 (1997), pp. 4659–4679.

[97] H. P. Le, J.-L. Cambier, and L. K. Cole, "GPU-based flow simulation with detailed chemical kinetics", *Computer Physics Communications* 184.3 (2013), pp. 596–606.

[98] G. Lee, X. M. Meyer, B. Biscans, J. M. Le Lann, and E. S. Yoon, "Adaptive finite difference method for the simulation of batch crystallization", *Computers and Chemical Engineering Supplement* 23 (1999), S363–S366.

[99] G. Lee, E. S. Yoon, Y.-I. Lim, J. M. Le Lann, X.-M. Meyer, and X. Joulia, "Adaptive mesh method for the simulation of crystallization processes including agglomeration and breakage: the potassium sulfate system", *Industrial & Engineering Chemistry Research* 40.26 (2001), pp. 6228–6235.

[100] K. W. Lee, J. Chen, and J. A. Gieseke, "Log-normally preserving size distribution for Brownian coagulation in the free-molecule regime", *Aerosol Science and Technology* 3.1 (1984), pp. 53–62.

[101] T. K. Lesniewski and S. K. Friedlander, "Particle nucleation and growth in a free turbulent jet", *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 454.1977 (1998), pp. 2477–2504.

[102] T. K. Lesniewski, "Particle nucleation and growth in turbulent jets", PhD thesis, University of California, Los Angeles, 1997.

[103] D. K. Lilly, *The representation of small scale turbulence in numerical simulation experiments*, tech. rep. 281, National Center for Atmospheric Research (NCAR), 1967.

[104] Y. I. Lim, J. M. Le Lann, X. M. Meyer, and X. Joulia, "Dynamic simulation of batch crystallization process by using moving finite difference method", *11th European Symposium on Computer Aided Process Engineering*, ed. by R. Gani and S. B. Jørgensen, vol. 9, Elsevier, 2001, pp. 201–206.

[105] Y. I. Lim, J.-M. Le Lann, X. M. Meyer, X. Joulia, G. Lee, and E. S. Yoon, "On the solution of population balance equations (PBE) with accurate front tracking methods in practical crystallization processes", *Chemical Engineering Science* 57.17 (2002), pp. 3715–3732.

[106] R. P. Lindstedt and S. A. Louloudi, "Joint-scalar transported PDF modeling of soot formation and oxidation", *Proceedings of the Combustion Institute* 30.1 (2005), pp. 775–783.

[107] J. C. Linford, J. Michalakes, M. Vachharajani, and A. Sandu, "Multi-core acceleration of chemical kinetics for simulation and prediction", *Proceedings of the Conference on High Performance Computing, Networking, Storage and Analysis*, SC '09, Portland, OR, USA: ACM, 2009, 7:1–7:11.

[108] J. D. Litster, D. J. Smit, and M. J. Hounslow, "Adjustable discretized population balance for growth and aggregation", *AIChE Journal* 41.3 (1995), pp. 591–603.

[109] F. Liu, H. Guo, G. Smallwood, and Ö. Gülder, "Numerical modelling of soot formation and oxidation in laminar coflow non-smoking and smoking ethylene diffusion flames", *Combustion Theory and Modelling* 7.2 (2003), pp. 301–315.

[110] J. Luche, "Elaboration of reduced kinetic models of combustion - Application to a kerosene mechanism", PhD thesis, Université d'Orléans, 2003.

[111] T. S. Lundgren, "Distribution functions in the statistical theory of turbulence", *Physics of Fluids* 10.5 (1967), pp. 969–975.

[112] A. W. Mahoney and D. Ramkrishna, “Efficient solution of population balance equations with discontinuities by finite elements”, *Chemical Engineering Science* 57.7 (2002), pp. 1107–1119.

[113] Ł. Makowski, W. Orciuch, and J. Bałdyga, “Large eddy simulations of mixing effects on the course of precipitation process”, *Chemical Engineering Science* 77.0 (2012), pp. 85–94.

[114] P. Marchal, R. David, J. P. Klein, and J. Villermaux, “Crystallization and precipitation engineering – I. An efficient method for solving population balance in crystallization with agglomeration”, *Chemical Engineering Science* 43.1 (1988), pp. 59–67.

[115] D. L. Marchisio and R. O. Fox, *Computational Models for Polydisperse Particulate and Multiphase Systems*, Cambridge Series in Chemical Engineering, Cambridge, New York: Cambridge University Press, 2013.

[116] D. L. Marchisio, R. Fox, A. A. Barresi, M. Garbero, and G. Baldi, “On the simulation of turbulent precipitation in a tubular reactor via computational fluid dynamics (CFD)”, *Chemical Engineering Research and Design* 79.8 (2001), pp. 998–1004.

[117] D. L. Marchisio, R. Dennis Vigil, and R. O. Fox, “Implementation of the quadrature method of moments in CFD codes for aggregation-breakage problems”, *Chemical Engineering Science* 58.15 (2003), pp. 3337–3351.

[118] D. L. Marchisio, A. A. Barresi, and M. Garbero, “Nucleation, growth, and agglomeration in barium sulfate turbulent precipitation”, *AIChE Journal* 48.9 (2002), pp. 2039–2050.

[119] R. McDermott and S. B. Pope, “A particle formulation for treating differential diffusion in filtered density function methods”, *Journal of Computational Physics* 226.1 (2007), pp. 947–993.

[120] R. McGraw, “Description of aerosol dynamics by the quadrature method of moments”, *Aerosol Science and Technology* 27.2 (1997), pp. 255–265.

[121] D. Meimaroglou, A. I. Roussos, and C. Kiparissides, “Part IV: Dynamic evolution of the particle size distribution in particulate processes. A comparative study between Monte Carlo and the generalized method of moments”, *Chemical Engineering Science* 61.17 (2006), pp. 5620–5635.

[122] B. Merci, B. Naud, and D. Roekaerts, “Flow and mixing fields for transported scalar PDF simulations of a piloted jet diffusion flame (‘Delft flame III’)”, *Flow, Turbulence and Combustion* 74.3 (2005), pp. 239–272.

[123] B. Merci, D. Roekaerts, and B. Naud, “Study of the performance of three micromixing models in transported scalar PDF simulations of a piloted jet diffusion flame (“Delft flame III”)”, *Combustion and Flame* 144.3 (2006), pp. 476–493.

[124] M. Metternich, W. Kollmann, I. M. Kennedy, and J.-Y. Chen, “PDF prediction of sooting turbulent flames”, *29th Aerospace Sciences Meeting*, Reno, NV, USA, 1991.

[125] K. Miller and R. N. Miller, “Moving finite elements. I”, *SIAM Journal on Numerical Analysis* 18.6 (1981), pp. 1019–1032.

[126] J. B. Moss, C. D. Stewart, and K. J. Syed, “Flowfield modelling of soot formation at elevated pressure”, *Symposium (International) on Combustion* 22.1 (1989), pp. 413–423.

[127] S. Motz, A. Mitrović, and E.-D. Gilles, “Comparison of numerical methods for the simulation of dispersed phase systems”, *Chemical Engineering Science* 57.20 (2002), pp. 4329–4344.

[128] M. E. Mueller and H. Pitsch, “LES model for sooting turbulent nonpremixed flames”, *Combustion and Flame* 159.6 (2012), pp. 2166–2180.

[129] M. E. Mueller and V. Raman, “Effects of turbulent combustion modeling errors on soot evolution in a turbulent nonpremixed jet flame”, *Combustion and Flame* 161.7 (2014), pp. 1842–1848.

[130] M. E. Mueller, G. Blanquart, and H. Pitsch, “Hybrid method of moments for modeling soot formation and growth”, *Combustion and Flame* 156.6 (2009), pp. 1143–1155.

[131] A. Munshi, B. R. Gaster, and T. G. Mattson, *OpenCL Programming Guide*, ed. by M. Taub, Addison-Wesley Publishing Company Inc., 2011.

[132] A. Munshi, ed., *The OpenCL Specification, Version 1.1*, The Khronos Group Inc., 2011.

[133] R. Mustata, L. Valiño, C. Jiménez, W. P. Jones, and S. Bondi, “A probability density function Eulerian Monte Carlo field method for large eddy simulations: Application to a turbulent piloted methane/air diffusion flame (Sandia D)”, *Combustion and Flame* 145.1-2 (2006), pp. 88–104.

[134] J. Nagle and R. F. Strickland-Constable, “Oxidation of carbon between 1000-2000°C”, *Proceedings of the fifth Carbon Conference*, vol. 1, 1, New York: Pergamon, 1962, pp. 154–164.

[135] S. Navarro-Martinez and S. Rigopoulos, “Differential diffusion modelling in LES with RCCE-reduced chemistry”, *Flow, Turbulence and Combustion* 89.2 (2012), pp. 311–328.

[136] K. Netzell, H. Lehtiniemi, and F. Mauss, “Calculating the soot particle size distribution function in turbulent diffusion flames using a sectional method”, *Proceedings of the Combustion Institute* 31.1 (2007), pp. 667–674.

[137] G. Neuber, A. Kronenburg, O. T. Stein, and M. J. Cleary, “MMC-LES modelling of droplet nucleation and growth in turbulent jets”, *Chemical Engineering Science* 167 (2017), pp. 204–218.

[138] M. Nicmanis and M. J. Hounslow, “Finite-element methods for steady-state population balance equations”, *AIChE Journal* 44.10 (1998), pp. 2258–2272.

[139] A. E. Nielsen, “Nucleation and growth of crystals at high supersaturation”, *Kristall und Technik* 4.1 (1969), pp. 17–38.

[140] K. E. Niemeyer, C.-J. Sung, C. G. Fotache, and J. C. Lee, “Turbulence-chemistry closure method using graphics processing unit: a preliminary test”, *Seventh Fall Technical Meeting of the Eastern States Section of the Combustion Institute*, Storrs, CT, USA, 2011.

[141] K. E. Niemeyer and C.-J. Sung, “Accelerating moderately stiff chemical kinetics in reactive-flow simulations using GPUs”, *Journal of Computational Physics* 256 (2014), pp. 854–871.

[142] P. A. Nooren, H. A. Wouters, T. W. J. Peeters, D. Roekaerts, U. Maas, and D. Schmidt, “Monte Carlo PDF modelling of a turbulent natural-gas diffusion flame”, *Combustion Theory and Modelling* 1.1 (1997), pp. 79–96.

[143] P. A. Nooren, M. Versluis, T. H. van der Meer, R. S. Barlow, and J. H. Frank, “Raman-Rayleigh-LIF measurements of temperature and species concentrations in the Delft piloted turbulent jet diffusion flame”, *Applied Physics B* 71.1 (2000), pp. 95–111.

[144] *Nvidia Quadro Dual Copy Engines*, NVIDIA Corporation, 2010.

[145] A. A. Öncül, K. Sundmacher, A. Seidel-Morgenstern, and D. Thévenin, “Numerical and analytical investigation of barium sulphate crystallization”, *Chemical Engineering Science* 61.2 (2006), pp. 652–664.

[146] *OpenCL Best Practices Guide*, NVIDIA Corporation, 2011.

[147] *OpenCL Programming for the CUDA Architecture*, NVIDIA Corporation, 2012.

[148] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, “GPU Computing”, *Proceedings of the IEEE* 96.5 (2008), pp. 879–899.

[149] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, New York: Hemisphere Publishing Corporation, 1980.

[150] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th, Morgan Kaufmann/Elsevier, 2013.

[151] G. A. Pavliotis, *Stochastic Processes and Applications*, New York: Springer-Verlag, 2014.

[152] T. W. J. Peeters, P. P. J. Stroomer, J. E. de Vries, D. J. E. M. Roekaerts, and C. J. Hoogendoorn, “Comparative experimental and numerical investigation of a piloted turbulent natural-gas diffusion flame”, *Symposium (International) on Combustion* 25.1 (1994), pp. 1241–1248.

[153] E. L. Petersen, D. M. Kalitan, S. Simmons, G. Bourque, H. J. Curran, and J. M. Simmie, “Methane/propane oxidation at high pressures: Experimental and detailed chemical kinetic modeling”, *Proceedings of the Combustion Institute* 31.1 (2007), pp. 447–454.

[154] C. D. Pierce and P. Moin, “Progress-variable approach for large-eddy simulation of non-premixed turbulent combustion”, *Journal of Fluid Mechanics* 504 (2004), pp. 73–97.

[155] U. Piomelli and J. Liu, “Large-eddy simulation of rotating channel flows using a localized dynamic model”, *Physics of Fluids* 7.4 (1995), pp. 839–848.

[156] H. Pitsch and H. Steiner, “Large-eddy simulation of a turbulent piloted methane/air diffusion flame (Sandia flame D)”, *Physics of Fluids* 12.10 (2000), pp. 2541–2554.

[157] H. Pitsch, “Large-eddy simulation of turbulent combustion”, *Annual Review of Fluid Mechanics* 38.1 (2006), pp. 453–482.

[158] S. B. Pope, “A Monte Carlo method for the PDF equations of turbulent reactive flow”, *Combustion Science and Technology* 25.5-6 (1981), pp. 159–174.

[159] S. B. Pope, “Computations of turbulent combustion: Progress and challenges”, *Symposium (International) on Combustion* 23.1 (1991), pp. 591–612.

[160] S. B. Pope, “Particle method for turbulent flows: Integration of stochastic model equations”, *Journal of Computational Physics* 117.2 (1995), pp. 332–349.

[161] S. B. Pope, “PDF methods for turbulent reactive flows”, *Progress in Energy and Combustion Science* 11.2 (1985), pp. 119–192.

[162] S. B. Pope, *Turbulent Flows*, Cambridge University Press, 2000.

[163] V. N. Prasad, “Large eddy simulation of partially premixed turbulent combustion”, PhD thesis, Imperial College London, 2011.

[164] S. E. Pratsinis, “Simultaneous nucleation, condensation, and coagulation in aerosol reactors”, *Journal of Colloid and Interface Science* 124.2 (1988), pp. 416–427.

[165] N. H. Qamar, Z. T. Alwahabi, Q. N. Chan, G. J. Nathan, D. Roekaerts, and K. D. King, “Soot volume fraction in a piloted turbulent jet non-premixed flame of natural gas”, *Combustion and Flame* 156.7 (2009), pp. 1339–1347.

[166] S. Qamar, M. P. Elsner, I. A. Angelov, G. Warnecke, and A. Seidel-Morgenstern, “A comparative study of high resolution schemes for solving population balances in crystallization”, *Computers and Chemical Engineering* 30.6-7 (2006), pp. 1119–1131.

[167] S. Qamar, A. Ashfaq, G. Warnecke, I. Angelov, M. P. Elsner, and A. Seidel-Morgenstern, “Adaptive high-resolution schemes for multidimensional population balances in crystallization processes”, *Computers and Chemical Engineering* 31.10 (2007), pp. 1296–1311.

[168] V. Raman and R. O. Fox, “Modeling of fine-particle formation in turbulent flames”, *Annual Review of Fluid Mechanics* 48 (2016), pp. 159–190.

[169] D. Ramkrishna, *Population Balances: Theory and Applications to Particulate Systems in Engineering*, Elsevier Science, 2000.

[170] B. M. Reddy, A. De, and R. Yadav, “Numerical investigation of soot formation in turbulent diffusion flame with strong turbulence-chemistry interaction”, *Journal of Thermal Science and Engineering Applications* 8.1 (2016), p. 011001.

[171] M. Reddy and A. De, “Numerical investigation of soot formation in turbulent diffusion flames using Moss-Brookes model”, *ASME 2014 Gas Turbine India Conference, GTINDIA 2014*, New Delhi, India, 2014.

[172] M. Reddy, A. De, and R. Yadav, “Effect of precursors and radiation on soot formation in turbulent diffusion flame”, *Fuel* 148 (2015), pp. 58–72.

[173] S. Rigopoulos, “Population balance modelling of polydispersed particles in reactive flows”, *Progress in Energy and Combustion Science* 36.4 (2010), pp. 412–443.

[174] S. Rigopoulos, “PDF method for population balance in turbulent reactive flow”, *Chemical Engineering Science* 62.23 (2007), pp. 6865–6878.

[175] S. Rigopoulos and A. G. Jones, “Finite-element scheme for solution of the dynamic population balance equation”, *AIChE Journal* 49.5 (2003), pp. 1127–1139.

[176] D. Roekaerts, B. Merci, and B. Naud, “Comparison of transported scalar PDF and velocity-scalar PDF approaches to ‘Delft flame III’”, *Comptes Rendus Mécanique* 334.8-9 (2006), pp. 507–516.

[177] A. I. Roussos, A. H. Alexopoulos, and C. Kiparissides, “Dynamic evolution of PSD in continuous flow processes: A comparative study of fixed and moving grid numerical techniques”, *Chemical Engineering Science* 61.1 (2006), pp. 124–134.

[178] A. I. Roussos, A. H. Alexopoulos, and C. Kiparissides, “Part III: Dynamic evolution of the particle size distribution in batch and continuous particulate processes: A Galerkin on finite elements approach”, *Chemical Engineering Science* 60.24 (2005), pp. 6998–7010.

[179] V. Sabel’nikov and O. Soulard, “Rapidly decorrelating velocity-field model as a tool for solving one-point Fokker-Planck equations for probability density functions of turbulent reactive scalars”, *Physical Review E* 72.1 (2005), p. 016301.

[180] P. Sagaut, *Large Eddy Simulation for Incompressible Flows: An Introduction*, 3rd, Berlin: Springer, 2006.

[181] A. Sandu, J. Verwer, M. Van Loon, G. Carmichael, F. Potra, D. Dabdub, and J. Seinfeld, “Benchmarking stiff ODE solvers for atmospheric chemistry problems–I. Implicit vs explicit”, *Atmospheric Environment* 31.19 (1997), pp. 3151–3166.

[182] A. Sandu, J. Verwer, J. Blom, E. Spee, G. Carmichael, and F. Potra, “Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock solvers”, *Atmospheric Environment* 31.20 (1997), pp. 3459–3472.

[183] H.-C. Schwarzer, *Nanoparticle precipitation – An experimental and numerical investigation including mixing*, Berlin: Logos-Verlag, 2005.

[184] F. Sewerin and S. Rigopoulos, “Integration of stiff chemical kinetics on a CPU-GPU pair – Application to a turbulent, non-premixed flame”, *Ninth Mediterranean Combustion Symposium*, Rhodes, Greece, 2015.

[185] F. Sewerin and S. Rigopoulos, “A methodology for the integration of stiff chemical kinetics on GPUs”, *Combustion and Flame* 162.4 (2015), pp. 1375–1394.

[186] F. Sewerin and S. Rigopoulos, “An explicit adaptive grid approach for the numerical solution of the population balance equation”, *Chemical Engineering Science* 168 (2017), pp. 250–270.

[187] F. Sewerin and S. Rigopoulos, “An LES-PBE-PDF approach for modeling particle formation in turbulent reacting flows”, *Physics of Fluids* 29.10 (2017), p. 105105.

[188] F. Sewerin and S. Rigopoulos, “An LES-PBE-PDF approach for predicting the soot particle size distribution in turbulent flames”, *Combustion and Flame* 189 (2018), pp. 62–76.

[189] M. R. H. Sheikhi, T. G. Drozda, P. Givi, and S. B. Pope, “Velocity-scalar filtered density function for large eddy simulation of turbulent flows”, *Physics of Fluids* 15.8 (2003), pp. 2321–2337.

[190] Y. Shi, W. H. Green, H.-W. Wong, and O. O. Oluwole, “Accelerating multi-dimensional combustion simulations using GPU and hybrid explicit/implicit ODE integration”, *Combustion and Flame* 159.7 (2012), pp. 2388–2397.

[191] Y. Shi, W. H. Green Jr., H.-W. Wong, and O. O. Oluwole, “Redesigning combustion modeling algorithms for the Graphics Processing Unit (GPU): Chemical kinetic rate evaluation and ordinary differential equation integration”, *Combustion and Flame* 158.5 (2011), pp. 836–847.

[192] G. P. Smith et al., *GRI-Mech 3.0*, 2000, URL: `http://www.me.berkeley.edu/gri\_mech/`.

[193] M. D. Smooke, C. S. McEnally, L. D. Pfefferle, R. J. Hall, and M. B. Colket, “Computational and experimental study of soot formation in a coflow, laminar diffusion flame”, *Combustion and Flame* 117.1-2 (1999), pp. 117–139.

[194] M. D. Smooke, M. B. Long, B. C. Connelly, M. B. Colket, and R. J. Hall, "Soot formation in laminar diffusion flames", *Combustion and Flame* 143.4 (2005), pp. 613–628.

[195] K. Spafford, J. Meredith, J. Vetter, J. Chen, R. Grout, and R. Sankaran, "Accelerating S3D: A GPGPU case study", *Euro-Par 2009 Workshops, LNCS 6043*, ed. by H. X. Lin, M. Alexander, M. Forsell, A. Knüpfer, R. Prodan, L. Sousa, and A. Streit, Berlin, Heidelberg: Springer-Verlag, 2010, pp. 122–131.

[196] C. Stone and R. Davis, "Techniques for solving stiff chemical kinetics on GPUs", *51st AIAA Aerospace Sciences Meetings*, Grapevine, TX, USA: American Institute of Aeronautics and Astronautics, 2013.

[197] H. Tang and T. Tang, "Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws", *SIAM Journal on Numerical Analysis* 41.2 (2003), pp. 487–515.

[198] T. H. Tsang and J. R. Brock, "Simulation of condensation aerosol growth by condensation and evaporation", *Aerosol Science and Technology* 2.3 (1983), pp. 311–320.

[199] T. H. Tsang and A. Rao, "A moving finite element method for the population balance equation", *International Journal for Numerical Methods in Fluids* 10.7 (1990), pp. 753–769.

[200] L. Valiño, "A field Monte Carlo formulation for calculating the probability density function of a single scalar in a turbulent flow", *Flow, Turbulence and Combustion* 60.2 (1998), pp. 157–172.

[201] E. Varoglu and W. D. L. Finn, "Finite elements incorporating characteristics for one-dimensional diffusion-convection equation", *Journal of Computational Physics* 34.3 (1980), pp. 371–389.

[202] V. Vikas, Z. J. Wang, A. Passalacqua, and R. O. Fox, "Realizable high-order finite-volume schemes for quadrature-based moment methods", *Journal of Computational Physics* 230.13 (2011), pp. 5328–5352.

[203] V. Vikas, Z. J. Wang, and R. O. Fox, "Realizable high-order finite-volume schemes for quadrature-based moment methods applied to diffusion population balance equations", *Journal of Computational Physics* 249 (2013), pp. 162–179.

[204] H. Wang et al., *A high-temperature chemical kinetic model of n-alkane (up to n-dodecane), cyclohexane, and methyl-, ethyl-, n-propyl and n-butyl-cyclohexane oxidation at high temperatures, JetSurF version 2.0*, 2010, URL: `http://melchior.usc.edu/JetSurF/JetSurF2.0`.

[205] H. Wang, X. You, A. V. Joshi, S. G. Davis, A. Laskin, F. Egolfopoulos, and C. K. Law, *USC Mech Version II. High-temperature combustion reaction model of H2/CO/C1-C4 compounds*, University of Southern California, 2007, URL: `http://ignis.usc.edu/USC\_Mech\_II.htm`.

[206] J. D. Ward and C.-C. Yu, "Population balance modeling in Simulink: PCSS", *Computers and Chemical Engineering* 32.10 (2008), pp. 2233–2242.

[207] M. Warshaw, "Cloud droplet coalescence: Statistical foundations and a one-dimensional sedimentation model", *Journal of the Atmospheric Sciences* 24.3 (1967), pp. 278–286.

[208] H. Wei and J. Garside, "Application of CFD Modelling to Precipitation Systems", *Chemical Engineering Research and Design* 75.2 (1997), pp. 219–227.

[209] X. Y. Woo, R. B. H. Tan, P. S. Chow, and R. D. Braatz, "Simulation of mixing effects in antisolvent crystallization using a coupled CFD-PDF-PBE approach", *Crystal Growth & Design* 6.6 (2006), pp. 1291–1303.

[210] Y. Xuan and G. Blanquart, "Effects of aromatic chemistry-turbulence interactions on soot formation in a turbulent non-premixed flame", *Proceedings of the Combustion Institute* 35.2 (2015), pp. 1911–1919.

[211] N. N. Yanenko, *The Method of Fractional Steps: The Solution of Problems of Mathematical Physics in Several Variables*, ed. by M. Holt, Berlin: Springer-Verlag, 1971.

[212] K. J. Young and J. B. Moss, "Modelling sooting turbulent jet flames using an extended flamelet technique", *Combustion Science and Technology* 105.1-3 (1995), pp. 33–53.

[213] K. Zhou, A. Attili, A. Alshaarawi, and F. Bisetti, "Simulation of aerosol nucleation and growth in a turbulent mixing layer", *Physics of Fluids* 26.6 (2014), p. 065106.

[214] A. Zucca, D. L. Marchisio, A. A. Barresi, and R. O. Fox, "Implementation of the population balance equation in CFD codes for modelling soot formation in turbulent flames", *Chemical Engineering Science* 61.1 (2006), pp. 87–95.

Appendices

Appendix A

Node density distribution

A.1 Steady-state solutions of Eq. (2.48)

In this section, we show that Eqs. (2.40) and (2.45) are both sufficient (a) and necessary (b) conditions for $\rho_\infty = \rho(\cdot, s_\infty)$ to be a steady-state solution of Eq. (2.48) for some $s_\infty \geq 0$. In view of Eqs. (2.41) and (2.42), we have

$$P_\rho(l,s) \geq \rho(l,s) \quad \forall l \in [0,L] \tag{A.1}$$

such that, by Eq. (2.50), Eqs. (2.40) and (2.45) are equivalent to

$$\dot{R}(l,s) = 0 \quad \forall l \in [0,L]. \tag{A.2}$$

(a) If $\dot{R}(\cdot, s_\infty)$ vanishes identically, then, from Eq. (2.48), we immediately obtain $\partial\rho(l,s_\infty)/\partial s = 0$ for all $l \in [0,L]$, indicating that $\rho(\cdot, s_\infty)$ is a steady-state solution of Eq. (2.48).

(b) In order to show that Eq. (A.2) is also a necessary condition, we consider $\rho(\cdot, s_\infty)$ as a steady-state solution of Eq. (2.48) and proceed by contradiction, assuming that Eq. (A.2) does not hold. This implies that $\dot{R}(l^+, s_\infty) > 0$ for at least one point $l^+ \in [0,L]$ and, since $\dot{R}(\cdot, s_\infty)$ is piecewise continuous, that there exists a half-open interval $\mathcal{L}^+$ about l^+ on which $\dot{R}(\cdot, s_\infty) > 0$. In particular, $\mathcal{L}^+$ has a non-zero measure such that

$$\frac{1}{L}\int_0^L \dot{R}(u,s)\,du > 0. \tag{A.3}$$

Since, by assumption, $\rho(\cdot, s_\infty)$ is a steady-state solution of Eq. (2.48), we have the relation

$$\dot{R}(l,s_\infty) = \frac{\rho(l,s_\infty)}{\rho_\tau L}\int_0^L \dot{R}(u,s_\infty)\,du \equiv C'\rho(l,s_\infty) > 0 \quad \forall l \in [0,L] \tag{A.4}$$

with $C' > 0$ due to Eqs. (2.37), (2.38) and (A.3). Introducing Eq. (2.50) into Eq. (A.4) leads to the condition

$$\max(\rho_{\min}, P_\rho(\cdot, s_\infty)) = C\rho(\cdot, s_\infty) \tag{A.5}$$

with $C = 1 + C'T > 1$. If $P_\rho(\cdot, s_\infty) = \rho(\cdot, s_\infty)$ identically, then Eq. (A.5) implies

$$\rho_{\min} = C\rho(\cdot, s_\infty) \tag{A.6}$$

which contradicts Eq. (2.47). In view of Eq. (A.1), we must hence have $P_\rho(l^\star, s_\infty) > \rho(l^\star, s_\infty)$ for at least one point $l^\star \in [0, L]$. By continuity, it follows that $P_\rho(\cdot, s_\infty) > \rho(\cdot, s_\infty)$ on an entire open (or half-open if $l^\star$ is a boundary point) interval $\mathcal{L}^\star$ about $l^\star$. However, Eqs. (2.41) and (2.42) imply $P_\rho(\cdot, s_\infty) \neq C\rho(\cdot, s_\infty)$ on $\mathcal{L}^\star$ such that, for Eq. (A.5) to be valid on $\mathcal{L}^\star$, $\rho_{\min} \geq P_\rho(\cdot, s_\infty)$ here. On the other hand, $\rho(\cdot, s_\infty)$ and its padding $P_\rho(\cdot, s_\infty)$ coincide at all those points in $[0, L]$, at which $P_\rho(\cdot, s_\infty) \not> \rho(\cdot, s_\infty)$. In view of Eq. (A.5), we thus have $\rho_{\min} > P_\rho(\cdot, s_\infty)$ at these points. By consequence, $\rho_{\min}(l) \geq P_\rho(l, s_\infty)$ for all $l \in [0, L]$. In combination with Eq. (A.5), this again implies Eq. (A.6). Since Eq. (A.6) contradicts Eq. (2.47), we conclude by reversing the above rationale that Eq. (A.2) must hold.

A.2 An iterative solution scheme for the steady-state node density distribution

In view of a numerical solution scheme for Eqs. (2.48) and (2.49), we let $s_j = j\Delta s$ for $j \geq 0$, set $T = \Delta s = 1$ and for

$$\rho(l, s_j) \equiv \rho_j(l) = \rho_\tau \frac{d\bar{\tau}_j(l)}{dl} \tag{A.7}$$

consider the following semi-implicit integration scheme

$$\rho_{j+1}(l) - \rho_j(l) = \Delta s \dot{R}(l, s_j) - \frac{\rho_{j+1}(l)}{\rho_\tau} \frac{\Delta s}{L} \int_0^L \dot{R}(u, s_j)\, du. \tag{A.8}$$

Taking into account Eqs. (2.50) and (A.7) and slightly rearranging Eq. (A.8) now leads to the following two-step scheme for $j \geq 0$ and $l \in [0, L]$

$$\frac{d\bar{\tau}_{j+\frac{1}{2}}(l)}{dl} \equiv \frac{d\bar{\tau}_j(l)}{dl} + \max\left(\frac{\max\left(\rho_{\min}(l), P_{\rho_j}(l)\right)}{\rho_\tau} - \frac{d\bar{\tau}_j(l)}{dl}, 0\right), \tag{A.9}$$

$$\frac{d\bar{\tau}_{j+1}(l)}{dl} = \frac{d\bar{\tau}_{j+\frac{1}{2}}(l)}{dl} \left(\frac{1}{L} \int_0^L \frac{d\bar{\tau}_{j+\frac{1}{2}}(u)}{du}\, du\right)^{-1}. \tag{A.10}$$

Here, the first step implements an update rule, while the second step renormalizes the updated coordinate transformation $\bar{\tau}_{j+\frac{1}{2}}(l)$ such that the boundary conditions in Eqs. (2.6) and (2.7) are obeyed. Note that if $\bar{\tau}_0(l)$ is given in a discrete representation, then

Eqs. (A.9) and (A.10) can inherit this discretization to yield a l-discrete scheme. In the context of a finite volume discretization, for example, Eqs. (A.9) and (A.10) hold in a cell-average sense. The numerical scheme terminates if $d\bar{\tau}_j(l)/dl$ and $d\bar{\tau}_{j+1}(l)/dl$ are identical to within a given convergence tolerance. The converged Jacobians then yield the adjusted equidistributing coordinate transformation $\bar{\tau}_\infty(l)$ with inverse $\bar{l}_\infty(\tau)$.

Appendix B

The LES-PBE-PDF method

B.1 Identity in Eq. (3.18)

In order to show Eq. (3.18), we first note that the LES-operator $\bar{\cdot}$ represents the expectation of the argument fields $\cdot$ with respect to the LES-filtered joint *pdf* $f_{\mathbf{Y},N,\frac{\partial \mathbf{Y}}{\partial x_j},\dots}(\mathbf{y}, n(\cdot), \boldsymbol{\psi}, \dots; \mathbf{x}, t)$ corresponding to a single realization of $\mathbf{Y}(\mathbf{x}, t)$, $N(\cdot, \mathbf{x}, t)$ and the derivatives $\partial \mathbf{Y}(\mathbf{x}, t)/\partial x_j$, ..., where $\mathbf{y}$, $n(\cdot)$, $\boldsymbol{\psi}$, ... denote the sample space variables associated, respectively, with the random fields $\mathbf{Y}(\mathbf{x}, t)$, $N(\cdot, \mathbf{x}, t)$, $\partial \mathbf{Y}(\mathbf{x}, t)/\partial x_j$, ... and the lower dots are place holders for the remaining spatial derivatives not listed explicitly. Expanding the left hand side of Eq. (3.18) yields

$$\begin{aligned}
&\overline{g(\mathbf{y}, n(\cdot); \mathbf{x}, t) F\left(\mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t), \frac{\partial \mathbf{Y}(\mathbf{x}, t)}{\partial x_j}, \dots\right)} \\
&= \int \delta(\mathbf{y} - \mathbf{y}')\delta(n(\cdot) - n'(\cdot)) F(\mathbf{y}', n'(\cdot), \boldsymbol{\psi}, \dots) \\
&\quad \times f_{\mathbf{Y},N,\frac{\partial \mathbf{Y}}{\partial x_j},\dots}(\mathbf{y}', n'(\cdot), \boldsymbol{\psi}, \dots; \mathbf{x}, t)\, d\boldsymbol{\psi} \cdots dn'(\cdot)\, d\mathbf{y}'.
\end{aligned} \tag{B.1}$$

By introducing the joint *pdf* of the derivatives $\partial \mathbf{Y}(\mathbf{x}, t)/\partial x_j$, ... conditioned on $\mathbf{Y}(\mathbf{x}, t)$ and $N(\cdot, \mathbf{x}, t)$,

$$f_{\mathbf{Y},N,\frac{\partial \mathbf{Y}}{\partial x_j},\dots}(\mathbf{y}', n'(\cdot), \boldsymbol{\psi}, \dots; \mathbf{x}, t) = f_{\frac{\partial \mathbf{Y}}{\partial x_j},\dots|\mathbf{Y},N}(\boldsymbol{\psi}, \dots | \mathbf{y}', n'(\cdot); \mathbf{x}, t) f(\mathbf{y}', n'(\cdot); \mathbf{x}, t), \tag{B.2}$$

and invoking the sifting property of the Dirac δ-distribution [162, Appendix C], we obtain from Eq. (B.1)

$$\begin{aligned}
&\overline{g(\mathbf{y}, n(\cdot); \mathbf{x}, t) F\left(\mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t), \frac{\partial \mathbf{Y}(\mathbf{x}, t)}{\partial x_j}, \ldots\right)} \\
&= \int \delta(\mathbf{y} - \mathbf{y}')\delta(n(\cdot) - n'(\cdot)) f(\mathbf{y}', n'(\cdot); \mathbf{x}, t) \\
&\quad \times \left(\int F(\mathbf{y}', n'(\cdot), \boldsymbol{\psi}, \ldots) f_{\frac{\partial \mathbf{Y}}{\partial x_j}, \ldots | \mathbf{Y}, N}(\boldsymbol{\psi}, \ldots | \mathbf{y}', n'(\cdot); \mathbf{x}, t)\, d\boldsymbol{\psi} \cdots \right) dn'(\cdot)\, d\mathbf{y}' \\
&= f(\mathbf{y}, n(\cdot); \mathbf{x}, t) \left(\int F(\mathbf{y}, n(\cdot), \boldsymbol{\psi}, \ldots) f_{\frac{\partial \mathbf{Y}}{\partial x_j}, \ldots | \mathbf{Y}, N}(\boldsymbol{\psi}, \ldots | \mathbf{y}, n(\cdot); \mathbf{x}, t)\, d\boldsymbol{\psi} \cdots \right) \\
&= f(\mathbf{y}, n(\cdot); \mathbf{x}, t) \overline{\left[F\left(\mathbf{Y}(\mathbf{x}, t), N(\cdot, \mathbf{x}, t), \frac{\partial \mathbf{Y}(\mathbf{x}, t)}{\partial x_j}, \ldots\right) \middle| \mathbf{y}, n(\cdot) \right]},
\end{aligned} \tag{B.3}$$

where the final equality follows from the definition of the conditional expectation in terms of $f_{\frac{\partial \mathbf{Y}}{\partial x_j}, \ldots | \mathbf{Y}, N}$. This completes the proof of Eq. (3.18).

B.2 An evolution equation for the transition *pdf* associated with the stochastic fields

B.2.1 Incompressible, constant density flows

In the present section, we show that the transition *pdf* $h(\mathbf{z}, t; \mathbf{x})$ associated with the stochastic process $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$ in Eq. (3.33) evolves according to Eq. (3.31) subject to the initial condition in Eq. (3.32). Except for some modifications owing to the stochastic forcing term in Eq. (3.33) and differences due to the expectation operation, our derivation is similar in rationale to the one leading to Eq. (3.19) in Section 3.2.3. Following the ideas of Lundgren [111], we first introduce the fine-grained density $h'(\mathbf{z}, t; \mathbf{x})$ as the transition *pdf* associated with a single sample path of the stochastic process $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$,

$$h'(\mathbf{z}, t; \mathbf{x}) = \delta\left(\mathbf{z} - \boldsymbol{\theta}(t; \cdot, \mathbf{x})\right), \tag{B.4}$$

for a deterministic initial condition

$$\boldsymbol{\theta}(t_0; l, \mathbf{x}) = \mathbf{z}_0(l, \mathbf{x}) \quad \forall l \in [0, L]. \tag{B.5}$$

In view of Section 3.2.6, $\boldsymbol{\theta}(t; l, \mathbf{x})$ represents a temporal drift-diffusion process which is smoothly parameterized by $(l, \mathbf{x})$. By construction, the transition *pdf* $h(\mathbf{z}, t; \mathbf{x})$ associated with $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$ and the fine-grained density $h'(\mathbf{z}, t; \mathbf{x})$ are related via the expectation operator $\langle \cdot \rangle$ according to

$$h(\mathbf{z}, t; \mathbf{x}) = \left\langle h'(\mathbf{z}, t; \mathbf{x}) \right\rangle. \tag{B.6}$$

Since $\boldsymbol{\theta}(t;l,\mathbf{x})$ varies smoothly in $\mathbf{x}$, we have

$$\frac{\partial h'}{\partial x_j} = -\sum_{i=1}^{n_\phi} \frac{\partial h'}{\partial z_i}\frac{\partial \theta_i}{\partial x_j}, \quad j = 1,\dots,3, \tag{B.7}$$

where, for conciseness, the arguments of $h'(\mathbf{z},t;\mathbf{x})$ and $\boldsymbol{\theta}(t;\cdot,\mathbf{x})$ have been omitted. The temporal derivative of Eq. (B.4), on the other hand, follows from Itô's formula [151, Section 3.4],

$$\frac{\partial h'}{\partial t} = -\sum_{i=1}^{n_\phi} \frac{\partial h'}{\partial z_i}\frac{\partial \theta_i}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{n_\phi}\sum_{k=1}^{3}\sigma_{ik}\sigma_{jk}\frac{\partial^2 h'}{\partial z_i \partial z_j}. \tag{B.8}$$

Here, $\sigma_{ik}(\partial\theta_i/\partial x_k,\mathbf{x},t)$ represents the diffusion coefficient of Eq. (3.33),[17]

$$\sigma_{ik}\left(\frac{\partial \theta_i}{\partial x_k},\mathbf{x},t\right) = -\sqrt{2\Gamma(\mathbf{x},t)}\frac{\partial \theta_i}{\partial x_k}. \tag{B.9}$$

By introducing Eqs. (3.33) and (B.9) into Eq. (B.8) and taking into account Eq. (B.7), we obtain

$$\begin{aligned}\frac{\partial h'}{\partial t} + \sum_{j=1}^{3}\overline{u}_j\frac{\partial h'}{\partial x_j} = & -\sum_{i=1}^{n_\phi}\frac{\partial h'}{\partial z_i}\left(s_i(l,\boldsymbol{\theta}) + m_i(\mathbf{x},t,\boldsymbol{\theta})\right) - \sqrt{2\Gamma}\sum_{j=1}^{3}\frac{\partial h'}{\partial x_j}\dot{W}_j(t) \\ & + \sum_{k=1}^{3}\left\{\sum_{i,j=1}^{n_\phi}\Gamma\frac{\partial\theta_i}{\partial x_k}\frac{\partial\theta_j}{\partial x_k}\frac{\partial^2 h'}{\partial z_i\partial z_j} - \sum_{i=1}^{n_\phi}\frac{\partial h'}{\partial z_i}\frac{\partial}{\partial x_k}\left(\Gamma\frac{\partial\theta_i}{\partial x_k}\right)\right\}.\end{aligned} \tag{B.10}$$

Since $\boldsymbol{\theta}(t;\cdot,\mathbf{x})$ is independent of $\mathbf{z}$ and $h'(\mathbf{z},t;\mathbf{x})$ varies smoothly in $\mathbf{x}$, we may reformulate the final terms in Eq. (B.10) with the aid of Eq. (B.7) and Schwarz's rule as follows

$$-\sum_{i=1}^{n_\phi}\left\{\left(\Gamma\frac{\partial\theta_i}{\partial x_k}\right)\frac{\partial^2 h'}{\partial x_k\partial z_i} + \frac{\partial h'}{\partial z_i}\frac{\partial}{\partial x_k}\left(\Gamma\frac{\partial\theta_i}{\partial x_k}\right)\right\} \tag{B.11}$$

$$= -\frac{\partial}{\partial x_k}\left(\Gamma\sum_{i=1}^{n_\phi}\frac{\partial h'}{\partial z_i}\frac{\partial\theta_i}{\partial x_k}\right) = \frac{\partial}{\partial x_k}\left(\Gamma\frac{\partial h'}{\partial x_k}\right), \quad k = 1,\dots,3. \tag{B.12}$$

From the commutation property of the expectation operator $\langle\cdot\rangle$, the analogue of Eq. (3.18) for $\langle\cdot\rangle$ and Eq. (B.12), we obtain after taking the expectation of Eq. (B.10) (B.10)

$$\begin{aligned}\frac{\partial h}{\partial t} + \sum_{j=1}^{3}\overline{u}_j\frac{\partial h}{\partial x_j} = & \sum_{k=1}^{3}\frac{\partial}{\partial x_k}\left(\Gamma\frac{\partial h}{\partial x_k}\right) - \sum_{i=1}^{n_\phi}\frac{\partial}{\partial z_i}\left(hs_i(\cdot,\mathbf{z}) + hm_i(\mathbf{x},t,\mathbf{z})\right) \\ & - \sum_{j=1}^{3}\left\langle\sqrt{2\Gamma}\frac{\partial h'}{\partial x_j}\dot{W}_j(t)\right\rangle.\end{aligned} \tag{B.13}$$

[17]If the micromixing model $m_i(\mathbf{x},t,\boldsymbol{\theta})$ includes a Brownian diffusion term in Itô's sense, then the matrix $\boldsymbol{\sigma}$ also encompasses the corresponding diffusion coefficients.

Here, the final term vanishes due to the martingale property of the Itô stochastic integral [151, Section 3.2]. By consequence, Eq. (B.13) reduces to the evolution equation

$$\frac{\partial h}{\partial t} + \sum_{j=1}^{3} \overline{u}_j \frac{\partial h}{\partial x_j} = \sum_{k=1}^{3} \frac{\partial}{\partial x_k} \left(\Gamma \frac{\partial h}{\partial x_k} \right) - \sum_{i=1}^{n_\phi} \frac{\partial}{\partial z_i} \left(h s_i(\cdot, \mathbf{z}) + h m_i(\mathbf{x}, t, \mathbf{z}) \right) \tag{B.14}$$

which coincides with our modelled transport equation for the joint scalar-number density *pdf* $f(\mathbf{z}; \mathbf{x}, t)$ in Eq. (3.31).

B.2.2 Variable density flows at low Mach number

In this section, an evolution equation is obtained for the transition *pdf* $h(\mathbf{z}, t; \mathbf{x})$ associated with the stochastic process $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$ that evolves according to Eq. (4.33) subject to the initial condition $\boldsymbol{\theta}(t_0; \cdot, \mathbf{x}) = (\mathbf{Y}_0(\mathbf{x}), N_{\rho,0}(\cdot, \mathbf{x}))$. Following the rationale outlined in Appendix B.2.1, we introduce the fine-grained density function $h'(\mathbf{z}, t; \mathbf{x})$ associated with $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$,

$$h'(\mathbf{z}, t; \mathbf{x}) = \delta(\mathbf{z} - \boldsymbol{\theta}(t; \cdot, \mathbf{x})), \tag{B.15}$$

and recall that applying the expectation operator $\langle \cdot \rangle$ to Eq. (B.15) yields the transition *pdf* $h(\mathbf{z}, t; \mathbf{x})$,

$$h(\mathbf{z}, t; \mathbf{x}) = \left\langle h'(\mathbf{z}, t; \mathbf{x}) \right\rangle = \left\langle \delta(\mathbf{z} - \boldsymbol{\theta}(t; \cdot, \mathbf{x})) \right\rangle. \tag{B.16}$$

The spatial derivatives of $h'(\mathbf{z}, t; \mathbf{x})$ are given by

$$\frac{\partial h'}{\partial x_j} = -\sum_{i=1}^{n_\phi} \frac{\partial h'}{\partial z_i} \frac{\partial \theta_i}{\partial x_j}, \quad j = 1, \ldots, 3. \tag{B.17}$$

Since the fine-grained density depends on time through the stochastic process $\boldsymbol{\theta}(t; \cdot, \mathbf{x})$ (Eq. (B.15)), the temporal derivative of $h'(\mathbf{z}, t; \mathbf{x})$ can be obtained from Itô's formula [151, Section 3.4],

$$\frac{\partial h'}{\partial t} = -\sum_{i=1}^{n_\phi} \frac{\partial h'}{\partial z_i} \frac{\partial \theta_i}{\partial t} + \frac{1}{2} \sum_{i,j=1}^{n_\phi} \sum_{k=1}^{3} \frac{\partial^2 h'}{\partial z_i \partial z_j} \sigma_{ik} \sigma_{jk}, \tag{B.18}$$

where $\sigma_{ik}(\partial \theta_i / \partial x_k, \mathbf{x}, t)$ represents the diffusion coefficient associated with Eq. (4.33),

$$\sigma_{ik} \left(\frac{\partial \theta_i}{\partial x_k}, \mathbf{x}, t \right) = -\sqrt{2 \Gamma(\mathbf{x}, t)} \frac{\partial \theta_i}{\partial x_k}. \tag{B.19}$$

ntroducing Eqs. (4.33) and (B.19) into Eq. (B.18) and taking into account Eq. (B.17) as well as the identity

$$\sum_{i=1}^{n_\phi} \sum_{k=1}^{3} \left\{ \overline{\rho} \Gamma \frac{\partial \theta_i}{\partial x_k} \sum_{j=1}^{n_\phi} \frac{\partial^2 h'}{\partial z_j \partial z_i} \frac{\partial \theta_j}{\partial x_k} - \frac{\partial h'}{\partial z_i} \frac{\partial}{\partial x_k} \left(\overline{\rho} \Gamma \frac{\partial \theta_i}{\partial x_k} \right) \right\} = \sum_{k=1}^{3} \frac{\partial}{\partial x_k} \left(\overline{\rho} \Gamma \frac{\partial h'}{\partial x_k} \right) \tag{B.20}$$

leads to the following evolution equation for the fine-grained *pdf* $h'(\mathbf{z}, t; \mathbf{x})$

$$\begin{aligned}\overline{\rho}\frac{\partial h'}{\partial t} + \sum_{j=1}^{3}\overline{\rho}\tilde{u}_j\frac{\partial h'}{\partial x_j} = \sum_{k=1}^{3}\frac{\partial}{\partial x_k}\left(\overline{\rho}\Gamma\frac{\partial h'}{\partial x_k}\right) - \overline{\rho}\sqrt{2\Gamma}\sum_{k=1}^{3}\frac{\partial h'}{\partial x_k}\dot{W}_k(t) \\ - \sum_{i=1}^{n_\phi}\frac{\partial h'}{\partial z_i}\overline{\rho}\left(s_i(\cdot,\boldsymbol{\theta}) + m_i(\mathbf{x},t,\boldsymbol{\theta})\right).\end{aligned} \tag{B.21}$$

By applying the expectation operator $\langle\cdot\rangle$ to Eq. (B.21), we obtain on account of the commutation property of $\langle\cdot\rangle$, the $\langle\cdot\rangle$-analogue of Eq. (3.18) and by the martingale property of the Itô stochastic integral

$$\overline{\rho}\frac{\partial h}{\partial t} + \sum_{j=1}^{3}\overline{\rho}\tilde{u}_j\frac{\partial h}{\partial x_j} = \sum_{k=1}^{3}\frac{\partial}{\partial x_k}\left(\overline{\rho}\Gamma\frac{\partial h}{\partial x_k}\right) - \sum_{i=1}^{n_\phi}\frac{\partial}{\partial z_i}\left(\overline{\rho}h\left(s_i(\cdot,\mathbf{z}) + m_i(\mathbf{x},t,\mathbf{z})\right)\right). \tag{B.22}$$

This evolution equation corresponds to our *physical* modelled *pdf* transport equation for $\tilde{f}(\mathbf{z}; \mathbf{x}, t)$ given in Eqs. (4.25), (4.28) and (4.30).

B.3 An adaptive grid discretization of Eq. (3.33)

In an explicit formulation, the time evolution of the coordinate transformation over the upcoming (fractional) time step, $t \in [t_k, t_{k+1}]$, $k = 0, 1, \ldots$, is forecast explicitly in terms of the current coordinate transformation $\bar{l}(\tau, \mathbf{x}, t_k)$ and the current solution for the LES-filtered number density $\overline{N}(l, \mathbf{x}, t_k) \equiv \overline{F}(\bar{\tau}(l, \mathbf{x}, t_k), \mathbf{x}, t_k)$.

By defining the transformed stochastic fields $\boldsymbol{\phi}(t; \tau, \mathbf{x})$ according to

$$\boldsymbol{\theta}(t; l, \mathbf{x}) \equiv \boldsymbol{\phi}(t; \bar{\tau}(l, \mathbf{x}, t), \mathbf{x}) \tag{B.23}$$

and introducing Eq. (B.23) into Eq. (3.33), we obtain after evaluation at $\bar{l}(\tau, \mathbf{x}, t)$ and in combination with Eqs. (2.9) through (2.12)

$$\begin{aligned}&\frac{\partial \phi_i}{\partial t} + \sum_{j=1}^{3}\left(\overline{u}_j(\mathbf{x},t) + \sqrt{2\Gamma(\mathbf{x},t)}\dot{W}_j(t)\right)\frac{\partial \phi_i}{\partial x_j} + \frac{1}{w}\frac{\partial \phi_i}{\partial \tau}\left(G(\bar{l},\boldsymbol{\phi}) - g'(\tau,\mathbf{x},t)\right) \\ &= \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\Gamma(\mathbf{x},t)\frac{\partial \phi_i}{\partial x_j}\right) + \frac{\Gamma(\mathbf{x},t)}{w^2}\frac{\partial}{\partial \tau}\left(\frac{\partial \phi_i}{\partial \tau}\sum_{j=1}^{3}\left(\frac{\partial \bar{l}}{\partial x_j}\right)^2\right) \\ &\quad - \frac{2\Gamma(\mathbf{x},t)}{w}\sum_{j=1}^{3}\frac{\partial^2 \phi_i}{\partial x_j \partial \tau}\frac{\partial \bar{l}}{\partial x_j} + s_i(\bar{l},\boldsymbol{\phi}) + m_i(\bar{l},\mathbf{x},t,\boldsymbol{\phi}), \quad i = 1, \ldots, n_\phi,\end{aligned} \tag{B.24}$$

where $g'(\tau, \mathbf{x}, t)$ involves the contribution of $\bar{l}(\tau, \mathbf{x}, t)$ to the overall growth rate (compare with Eq. (2.22))

$$g'(\tau,\mathbf{x},t) = \frac{\partial \bar{l}}{\partial t} + \sum_{j=1}^{3}\left(\overline{u}_j(\mathbf{x},t) - \frac{\partial \Gamma(\mathbf{x},t)}{\partial x_j} + \sqrt{2\Gamma(\mathbf{x},t)}\dot{W}_j(t)\right)\frac{\partial \bar{l}}{\partial x_j} \\ - \Gamma(\mathbf{x},t) b(\tau,\mathbf{x},t) \tag{B.25}$$

and $b(\tau,\mathbf{x},t)$ summarizes the second derivatives of $\bar{l}(\tau,\mathbf{x},t)$,

$$b(\tau,\mathbf{x},t) = \sum_{j=1}^{3}\frac{\partial^2 \bar{l}}{\partial x_j^2} + \frac{1}{w^2}\frac{\partial^2 \bar{l}}{\partial \tau^2}\sum_{j=1}^{3}\left(\frac{\partial \bar{l}}{\partial x_j}\right)^2. \tag{B.26}$$

Similar to Eqs. (3.35) through (3.37), the fractional steps for convection/diffusion, mixing and fluid phase reaction corresponding to Eq. (B.24) are given by

$$\frac{\partial \phi_i}{\partial t} + \sum_{j=1}^{3}\left(\overline{u}_j(\mathbf{x},t) + \sqrt{2\Gamma(\mathbf{x},t)}\dot{W}_j(t)\right)\frac{\partial \phi_i}{\partial x_j} = \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\Gamma(\mathbf{x},t)\frac{\partial \phi_i}{\partial x_j}\right), \\ i = 1,\ldots,n_\phi, \tag{B.27}$$

and

$$\frac{\partial \phi_i}{\partial t} = m_i(\mathbf{x},t,\boldsymbol{\phi}), \quad i = 1,\ldots,n_\phi, \tag{B.28}$$

$$\frac{\partial \phi_i}{\partial t} = \dot{\omega}_i(\boldsymbol{\phi}), \quad i = 1,\ldots,n_\phi - 1. \tag{B.29}$$

Note that, also under the coordinate transformation, the stochastic scalars $\phi_i(t;\tau,\mathbf{x})$ which physically describe the fluid phase ($i = 1,\ldots,n_\phi - 1$) evolve independently of the transformed particle property τ.

The PBE fractional step encompasses additional transport terms that account for the redistribution of resolution as the LES-filtered number density evolves

$$\frac{\partial \phi_{n_\phi}}{\partial t} + \phi_{n_\phi}\frac{\partial G(\bar{l},\boldsymbol{\phi})}{\partial l} + \frac{1}{w}\frac{\partial \phi_{n_\phi}}{\partial \tau}\left(G(\bar{l},\boldsymbol{\phi}) - g'(\tau,\mathbf{x},t)\right) \\ = \frac{\Gamma(\mathbf{x},t)}{w^2}\frac{\partial}{\partial \tau}\left(\frac{\partial \phi_{n_\phi}}{\partial \tau}\sum_{j=1}^{3}\left(\frac{\partial \bar{l}}{\partial x_j}\right)^2\right) - \frac{2\Gamma(\mathbf{x},t)}{w}\sum_{j=1}^{3}\left(\frac{\partial^2 \phi_{n_\phi}}{\partial \tau \partial x_j}\frac{\partial \bar{l}}{\partial x_j}\right) + \dot{s}(\bar{l},\boldsymbol{\phi}). \tag{B.30}$$

Except for the Wiener term, Eq. (B.30) corresponds to Eq. (2.16). If, in a practical implementation, the mixed partial derivatives $\partial^2\phi_{n_\phi}/\partial x_j \partial \tau$, $j = 1,\ldots,3$, are evaluated at the previous point in time, then Eq. (3.38) can be solved independently for each spatial grid point.

Similar to Eq. (3.39), an observable $H(\mathbf{Y}(\mathbf{x},t), N(\cdot,\mathbf{x},t))$ can be approximated by Monte-Carlo estimates based on n_f realizations $\boldsymbol{\phi}^{(i)}(t;\tau,\mathbf{x})$, $i = 1,\ldots,n_f$, of the transformed stochastic fields $\boldsymbol{\phi}(t;\tau,\mathbf{x})$,

$$\overline{H}(\mathbf{Y}(\mathbf{x},t),N(\cdot,\mathbf{x},t)) \approx \frac{1}{n_f}\sum_{i=1}^{n_f} H(\boldsymbol{\phi}^{(i)}(t;\bar{\tau}(\cdot,\mathbf{x},t),\mathbf{x})). \tag{B.31}$$

Appendix C

Algorithmic details of the Radau5 GPU implementation

C.1 Radau5

Figures C.1 and C.2 depict a flow chart of the OpenCL reimplementation of the Radau5 algorithm.

C.2 Determining the number of kernel invocations

Figure C.3 lists the pseudo-code for an algorithm to determine the number of ODE systems n_p per kernel invocation for the overlapping computation/data transfer scheme of Section 5.5.5.

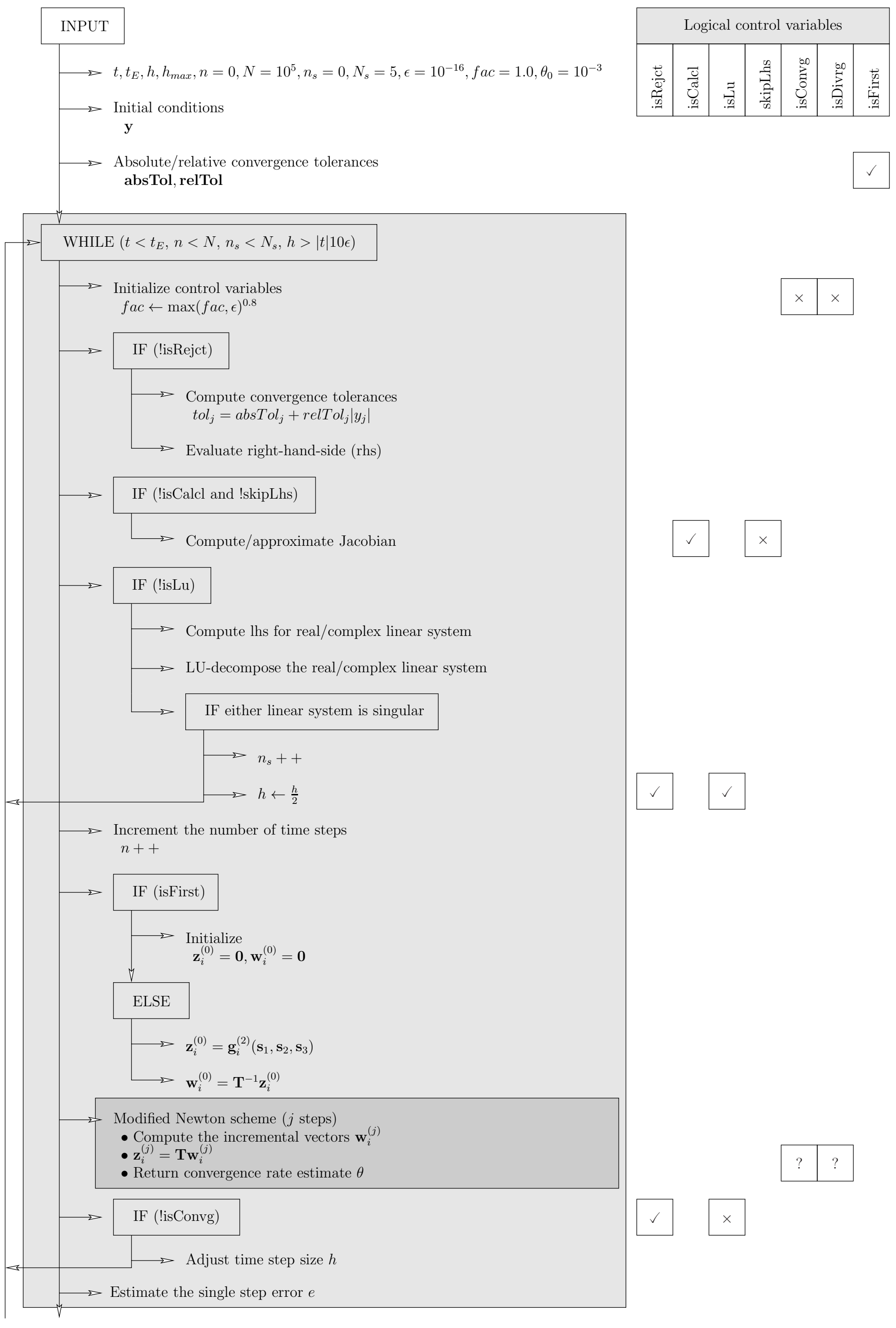
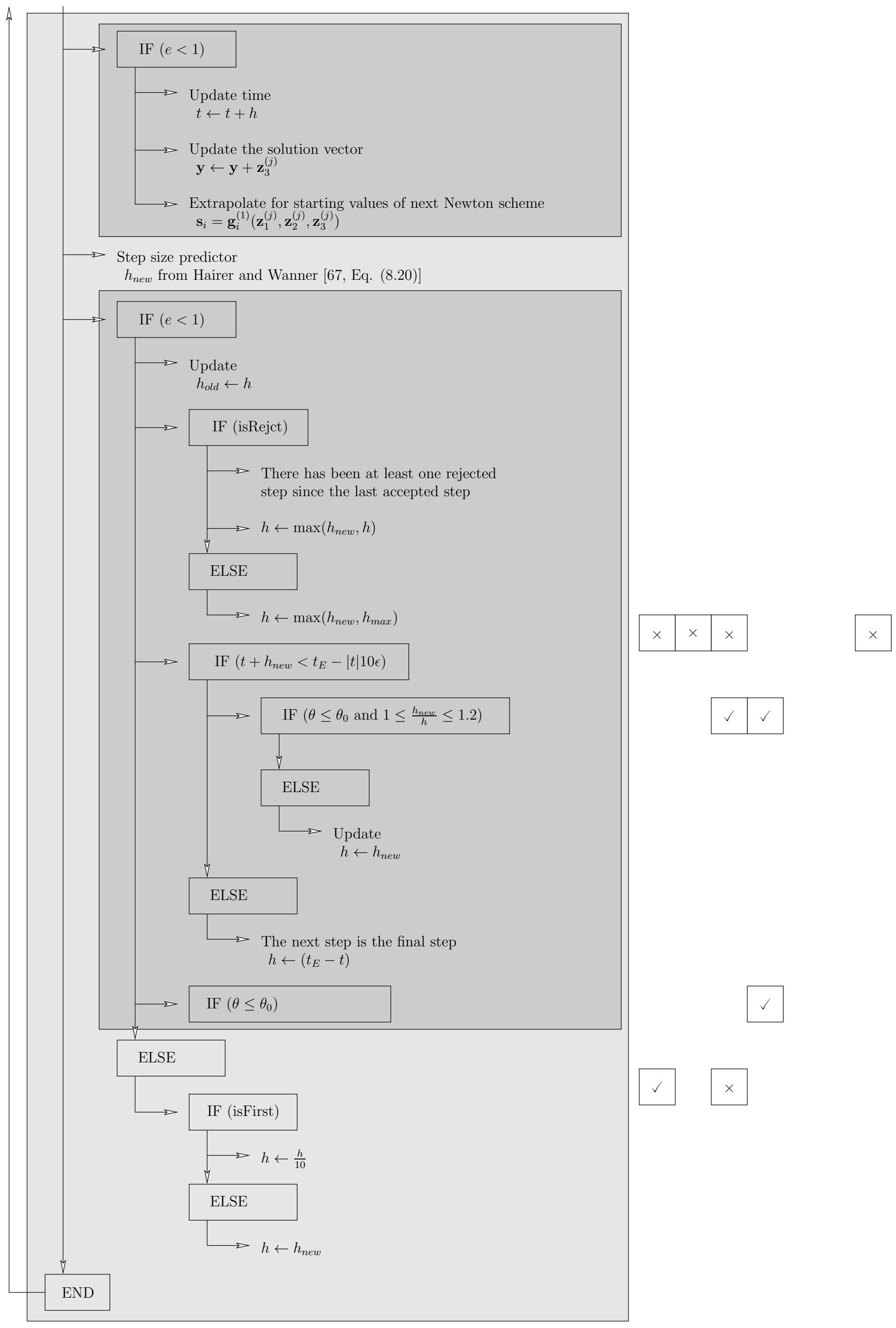

INPUT
$t, t_E, h, h_{max}, n=0, N=10^5, n_s=0, N_s=5, \epsilon=10^{-16}, fac=1.0, \theta_0=10^{-3}$
Initial conditions
$\mathbf{y}$
Absolute/relative convergence tolerances
absTol, **relTol**
Logical control variables
isRejct
isCalcl
isLu
skipLhs
isConvg
isDivrg
isFirst
WHILE ($t < t_E$, $n < N$, $n_s < N_s$, $h > |t|10\epsilon$)
Initialize control variables
$fac \leftarrow \max(fac, \epsilon)^{0.8}$
IF (!isRejct)
Compute convergence tolerances
$tol_j = absTol_j + relTol_j|y_j|$
Evaluate right-hand-side (rhs)
IF (!isCalcl and !skipLhs)
Compute/approximate Jacobian
IF (!isLu)
Compute lhs for real/complex linear system
LU-decompose the real/complex linear system
IF either linear system is singular
$n_s{+}{+}$
$h \leftarrow \frac{h}{2}$
Increment the number of time steps
$n{+}{+}$
IF (isFirst)
Initialize
$\mathbf{z}_i^{(0)} = \mathbf{0}, \mathbf{w}_i^{(0)} = \mathbf{0}$
ELSE
$\mathbf{z}_i^{(0)} = \mathbf{g}_i^{(2)}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$
$\mathbf{w}_i^{(0)} = \mathbf{T}^{-1}\mathbf{z}_i^{(0)}$
Modified Newton scheme (j steps)
• Compute the incremental vectors $\mathbf{w}_i^{(j)}$
• $\mathbf{z}_i^{(j)} = \mathbf{T}\mathbf{w}_i^{(j)}$
• Return convergence rate estimate θ
IF (!isConvg)
Adjust time step size h
Estimate the single step error e
✓
× ×
✓ ×
✓ ✓
? ?
✓ ×

Figure C.1 Flow chart indicating the control flow and the step size adjustment scheme of the OpenCL reimplementation of Radau5. This depiction slightly differs from the original Fortran 77 implementation by Hairer and Wanner [67] since the goto's have been removed and the individual steps encapsulated. The variables $T = \mathbf{T}, W = (\mathbf{w}_1^T, \mathbf{w}_2^T, \mathbf{w}_3^T)^T$ and $Z = (\mathbf{z}_1^T, \mathbf{z}_2^T, \mathbf{z}_3^T)^T$ appear in the context of Reference [67, Chapter IV.8], while $\mathbf{g}_i^{(1)}, \mathbf{g}_i^{(2)}$ and $\mathbf{s}_i$, $i = 1, \ldots, 3$, are auxiliary functions and vectors, respectively, which symbolize the interpolation of starting values for the Newton iteration from past solution vectors. (Continued in Figure C.2.)



IF $(e < 1)$
Update time
$t \leftarrow t + h$
Update the solution vector
$\mathbf{y} \leftarrow \mathbf{y} + \mathbf{z}_3^{(j)}$
Extrapolate for starting values of next Newton scheme
$\mathbf{s}_i = \mathbf{g}_i^{(1)}(\mathbf{z}_1^{(j)}, \mathbf{z}_2^{(j)}, \mathbf{z}_3^{(j)})$
Step size predictor
h_{new} from Hairer and Wanner [67, Eq. (8.20)]
IF $(e < 1)$
Update
$h_{old} \leftarrow h$
IF (isRejct)
There has been at least one rejected step since the last accepted step
$h \leftarrow \max(h_{new}, h)$
ELSE
$h \leftarrow \max(h_{new}, h_{max})$
IF $(t + h_{new} < t_E - |t|10\epsilon)$
IF $(\theta \leq \theta_0$ and $1 \leq \frac{h_{new}}{h} \leq 1.2)$
ELSE
Update
$h \leftarrow h_{new}$
ELSE
The next step is the final step
$h \leftarrow (t_E - t)$
IF $(\theta \leq \theta_0)$
ELSE
IF (isFirst)
$h \leftarrow \frac{h}{10}$
ELSE
$h \leftarrow h_{new}$
END

Figure C.2 (Continued from Figure C.1.) Flow chart indicating the control flow and the step size adjustment scheme of the OpenCL reimplementation of Radau5.

```
Input: $n_s$, $n_r$, $m_{sp}$
  1. Determine the remaining memory $r_m$ that is available on the device
  2. Query the maximum allocatable buffer size $r_b$
  3. Compute the size of the memory buffers required by a single ODE system
         $$n_b = \{4n_s(3+n_s) + (2+n_s) + n_r m_{sp}\}8\,\mathrm{B} + \{2n_s\}4\,\mathrm{B} \qquad \text{(C.1)}$$
  4. Compute the maximum number of ODE systems that can be solved in a single $t$-cycle
         $$n_1 = \text{floor}\left(\frac{\max(r_m, r_b)}{n_b}\right) \qquad \text{(C.2)}$$
  5. Compute the maximum number of ODE systems that can be solved in a single $p$-cycle
         $$n_2 = \text{floor}\left(\frac{\max(r_m, r_b)}{n_b + (n_s+2)8\,\mathrm{B}}\right) \qquad \text{(C.3)}$$
  6. Compute
         $$n_{hd} = \text{floor}\left(\bar{t}_{hd}\frac{BW}{(n_s+2)8\,\mathrm{B}}\right) \qquad \text{(C.4)}$$
  7. If $n > \min(n_{hd}, n_1)$, then (do $s$ $p$-cycles)
       If $n_2 > n_{hd}$, then
         $s = \max\left(1, \text{ceil}\left(\frac{n}{2n_{hd}}\right)\right)$ (Eq. (5.33))
       Else
         $s = \max\left(1, \text{ceil}\left(\frac{n}{2n_2}\right)\right)$ (Eq. (5.30))
       End
     Else (do a single $t$-cycle)
       $n_p = n$
     End
Output: $n_p$
```

Figure C.3 Scheme for determining the number of ODE systems n_p per kernel invocation.