

**FLUID FLOW AND HEAT TRANSFER
IN TUBE BANKS**

by

Steven Brydon Beale

Thesis submitted for the degree of Doctor of Philosophy
in the Faculty of Engineering,
University of London,
and for the Diploma of Membership
of Imperial College

Department of Chemical Engineering and Chemical Technology,
Imperial College of Science, Technology and Medicine,
London SW7 2BX

November 1992

ABSTRACT

This thesis explores fluid flow and heat transfer in in-line and staggered tube banks. A potential flow solution is derived, using complex function theory, in the form of a power series. Results are presented for a range of geometric configurations. The equations governing the transport of mass, momentum, and general scalar properties are derived in general-curvilinear co-ordinates. Both the mathematical and physical forms are given. A finite-volume discretization procedure is detailed: The finite-volume-based numerical scheme is then used to examine laminar steady flow and heat transfer in two-dimensional tube banks, and in three-dimensional fin and tube heat exchangers, and off-set fin heat exchangers. Fully-developed cross flow was assumed throughout. Both constant wall temperature and constant heat flux boundary conditions were considered. Structured orthogonal and non-orthogonal body-fitted grids were used; these being generated by means of the numerical solution of elliptic partial differential equations. Transient two-dimensional flow in in-line and staggered tube banks was also considered, it being shown that tube banks exhibit periodic behaviour qualitatively similar to those observed in single cylinders.

Numerous parametric studies were conducted. The results are presented in the form of overall and local pressure drop, heat transfer and skin friction data, as well as tables and graphs of flow visualisation studies. These are compared with existing experimental and numerical data.

PREFACE

This thesis arose as a result of two periods spent at the University of London, on leave from the National Research Council of Canada (NRC). The first period, for the academic year 1986-87, was spent working at the Computational Fluid Dynamics Unit (CFDU) at Imperial College. My advisor was D.B. Spalding, Head of the CFDU, and Professor of Heat Transfer. The second period was for a number of months in 1990, working with Prof. Spalding, both at the University and at his company, Concentration Heat and Momentum Ltd. (CHAM). The remainder of the research was conducted at the Montreal Road campus of NRC, in Ottawa.

It has been a great privilege to be associated with Prof. Spalding, and I would like to take this opportunity to thank him for providing me with the benefit of his unique insight into computational fluid dynamics. I would also like to take this opportunity to thank all former members of the CFDU for their help in matters technical and non-technical, including, but not restricted to; Dr. J.Z.Y. Wu, Dr. H.Q. Qin., Mr. M. Walsh, Mr. G. Liao, Dr. N. Fueyo, Dr. L. Jun, Dr. D. Radosavljevic, Ms. K. Taylor, Dr. F. Villasenor, Mr. F. King, Ms. F. Oliver and Ms. C.J. Webb.

After Prof. Spalding retired to an emeritus position in the Department of Chemical Engineering and Chemical Technology, Prof. G.F. Hewitt assumed formal responsibility for Prof. Spalding's students: I would like to thank him for this, and for his interest in my work. Prof. Spalding is currently managing director of CHAM Ltd., a rather unique enclave of CFD specialists. I would like to take this opportunity to thank all of the staff of CHAM for their help and assistance, and especially Dr. D.R. Glynn, Dr. M.R. Malin, Dr. J.K. Worrell, and Dr. J.C. Ludwig.

The generous support of the National Research Council is gratefully acknowledged. NRC funding covered two periods of extended leave and several shorter trips to London. The personal encouragement of Dr. I.R.G. Lowe, Mr. M.S. Chappell and Dr. R.M. Frederking of the Institute for Mechanical Engineering, is particularly appreciated. There are a number of people at NRC to whom I am indebted for technical support; Dr. A.C. Hsu, Mr. J.G. Kresko, Mr. K.S. Ng, Mr. R.J. Jerome, Dr. M.M. Oleskiw, Mr. S.J. Zurawski. Thanks are also due to Prof. D.S. Weaver, McMaster University, Dr. P.H. Oosthuizen, Queen's University, Dr. V. Watson of the NASA Ames Research Center, and Mr. N.C. Wheatley of Imperial College.

Finally, I would like to thank the person who has contributed the most towards the completion of this thesis; my wife, Reine.

TABLE OF CONTENTS

1. INTRODUCTION	18
1.1 Problem Characterisation	19
1.2 Objectives	20
1.3 Contribution of the Thesis	21
1.4 General Methodology	22
1.5 Organisation of the Thesis	24
1.6 Closure.....	26
2. LITERATURE REVIEW	27
2.1 Introduction	27
2.2 Tube Banks	27
2.2.1 Experimental data	27
2.2.2 Numerical data	30
2.3 Fin and Tube Heat Exchangers.....	31
2.3.1 Experimental data	31
2.4 Plate-fin Heat Exchangers	32
2.4.1 Experimental data	32
2.4.2 Numerical models	32
2.5 Closure.....	33
3. BASIC EQUATIONS	34

3.1 Vector Equations	34
3.2 Curvilinear Co-ordinates	36
3.2.1 Introduction	36
3.2.2 Mathematical form	38
3.2.3 Resolute and component forms	40
3.2.4 Ordinary derivative form	45
3.2.5 Discussion	47
3.3 Closure.....	49
4. POTENTIAL FLOW IN TUBE BANKS	50
4.1 Introduction	50
4.1.1 Potential flow in tube banks	52
4.2 Mathematical Details	53
4.3 Results	61
4.4 Discussion of Results	72
4.5 Discussion of Solution Procedure.....	74
4.6 Closure.....	75
5. FINITE-VOLUME EQUATIONS	76
5.1 Scalar Variable	76
5.1.1 Convection and Diffusion	77
5.1.2 Transient Term.....	80

5.1.3	Linearized Source Term	80
5.2	Continuity and Momentum	82
5.3	Body-fitted Co-ordinates	83
5.4	Porosities	86
5.5	Solution Procedure	87
5.5.1	Linear Algebraic Equations.	87
5.5.2	Relaxation	88
5.5.3	Hydrodynamic Equations.	89
5.6	Computer Code and Implementation	93
5.7	Closure.....	93
6.	GRID GENERATION.....	94
6.1	Introduction	94
6.2	Domain Boundaries	95
6.3	Basic Grid Patterns	96
6.4	Grid Generation Methodology	100
6.4.1	Differential equations	100
6.4.2	Discretization procedure.....	103
6.4.3	Boundary and initial conditions	105
6.4.4	Choice of control-functions	108
6.5	Grids used for In-line and Staggered Arrays	109

6.5.1 In-line geometry	110
6.5.2 Staggered geometry	111
6.6 Other Grid-types	113
6.7 Three-dimensional Grids	119
6.8 Closure.....	120
7. STREAM FUNCTIONS	121
7.1 Introduction	121
7.2 Governing Equations	122
7.3 Discretization	124
7.4 Graphics Utility	126
7.5 Discussion	128
7.6 Closure.....	131
8. STEADY TWO-DIMENSIONAL FLOW	132
8.1 Introduction	132
8.1.1 Boundary conditions	132
8.2 Results	139
8.2.1 Typical flow-field results	139
8.2.2 Pressure drop	149
8.2.3 Form and profile drag.....	150
8.2.4 Overall heat transfer	157

8.2.5 Local heat transfer	161
8.3 Discussion	174
8.3.1 Qualitative description	174
8.3.2 Average pressure drop	176
8.3.3 Form and profile drag.....	177
8.3.4 Overall heat transfer	179
8.3.5 Local heat transfer	181
8.3.6 Effect of Prandtl number	182
8.3.7 Choice of reference temperature	182
8.3.8 Grid independence	187
8.3.9 Convergence	192
8.4 Closure.....	194
9. UNSTEADY TWO-DIMENSIONAL FLOW	195
9.1 Introduction	195
9.1.1 Domain of interest	196
9.1.2 Initial and boundary conditions	196
9.2 Results	200
9.2.1 Flow field results.....	200
9.2.2 Time average quantities	210
9.2.3 Overall drag and heat transfer	211

9.2.4	Local coefficients	212
9.3	Discussion	222
9.3.1	Qualitative description	222
9.3.2	Frequency of oscillations	226
9.3.3	Lift and drag	228
9.3.4	Heat transfer	231
9.3.5	Convergence tests	232
9.3.6	Boundary and initial conditions	235
9.4	Closure.....	239
10.	FIN AND TUBE HEAT EXCHANGER	240
10.1	Introduction	240
10.2	Results	241
10.3	Discussion	258
10.3.1	Qualitative description of the flow	258
10.3.2	Overall pressure loss and heat transfer	261
10.3.3	Local drag and heat transfer	262
10.3.4	Grid dependence and convergence	265
10.4	Closure.....	265
11.	PLATE-FIN HEAT EXCHANGER	267
11.1	Introduction	267

11.2 Boundary Conditions	268
11.3 Definitions	270
11.4 Results	272
11.4.1 Typical flow-field results	272
11.4.2 Pressure drop and heat transfer data	276
11.5 Discussion	277
11.6 Closure.....	280
12. CONCLUSIONS AND RECOMMENDATIONS	281
13. NOMENCLATURE	285
14. BIBLIOGRAPHY	293
Appendix I. Curvilinear Co-ordinates	311
Appendix II. Elliptic Functions	337
Appendix III. Listings of Computer Programs	344

List of Figures

Figure 1. Tube banks.	19
Figure 2. Tube bank nomenclature.	20
Figure 3. Curvilinear co-ordinate system.	36
Figure 4. Control-volume in physical space.	41
Figure 5. Control-volume parallel to co-ordinate system.	45
Figure 6. Representation of a tube-bank in the complex plane.	55
Figure 7. Complex potential for an in-line square tube bank, $s/d = 1.5$	68
Figure 8. Complex potential for an equilateral-triangular tube bank, $s/d = 1.5$	68
Figure 9. Pressure contours for an in-line square tube bank, $s/d = 1.5$	69
Figure 10. Pressure contours for an equilateral tube bank, $s/d = 1.5$	69
Figure 11. Pressure contours for a rotated square tube bank, $s/d = 1.25$	70
Figure 12. Pressure coefficient vs. angle, various arrangements.	70
Figure 13. Tangential velocity vs. angle, various arrangements.	71
Figure 14. Velocity profiles at minimum cross-section, in-line square tube banks.	71
Figure 15. Cell Nomenclature.	77
Figure 16. Location of Field Variables	82
Figure 17. Simplified flow chart of solution procedure.	92
Figure 18. Some possible domain boundaries.	95
Figure 19. Primitive rectangular sub-domain.	96
Figure 20. Basic O-grid pattern.	97
Figure 21. Basic H-grid Pattern.	98
Figure 22. Basic I-grid pattern.	99
Figure 23. Two types of grid.	101
Figure 24. Flow chart of grid generation procedure.	107
Figure 25. Non-orthogonal H-grid.	109

Figure 26.	In-line H-grid with cylinders at corners.....	110
Figure 27.	In-line H-grid with cylinder at side.	110
Figure 28.	In-line I-grid.....	111
Figure 29.	Non-orthogonal diagonal H-grid.	112
Figure 30.	Diagonal I-Grid.	113
Figure 31.	Hybrid grid.....	114
Figure 32.	Block grid topology.	116
Figure 33.	In-line block H-grid.	117
Figure 34.	Staggered block H-grid.	118
Figure 35.	Three-dimensional grid.	119
Figure 36.	Stream-function graphics utility.	126
Figure 37.	Two-dimensional stream-lines.....	129
Figure 38.	Halo cells.	135
Figure 39.	Periodic boundary conditions.	137
Figure 40.	Stream-lines for an in-line square tube-bank, $s/d = 1.5$	141
Figure 41.	Velocity vectors for a rotated square tube bank, $s/d = 2$	142
Figure 42.	Pressure contours (Pa), in-line square, $s/d = 2$, $Re = 100$	143
Figure 43.	Pressure Contours, equilateral triangle, $s/d = 2$, $Re = 100$	143
Figure 44.	Temperature field, in-line square, $s/d = 1.5$, constant t_w , $Pr = 1$	144
Figure 45.	Temperature field, in-line square, $s/d = 1.5$, constant q_w , $Pr = 1$	145
Figure 46.	Temperature field, rotated square, $s/d = 1.25$, constant t_w , $Pr = 1$	146
Figure 47.	Temperature field rotated square, $s/d = 1.25$, constant q_w , $Pr = 1$	147
Figure 48.	Temperature field, in-line square, $s/d = 1.5$, $Re = 100$, various Pr	148
Figure 49.	Eu vs. Re compared to numerical data, in-line square.	151
Figure 50.	Eu vs. Re compared to numerical data, rotated square.	151
Figure 51.	Eu vs. Re compared to numerical data, equilateral triangle.	152
Figure 52.	Eu vs. Re compared to experimental data, in-line square.	152

Figure 53.	Eu vs. Re compared to experimental data, rotated square.	153
Figure 54.	Eu vs. Re compared to experimental data, equilateral triangle.....	153
Figure 55.	c_p vs. ϕ , in-line square, $s/d = 1.5$	154
Figure 56.	c_p vs. ϕ , rotated square, $s/d = 1.5$	154
Figure 57.	c_p vs. ϕ , equilateral triangle $s/d = 1.5$	155
Figure 58.	c_f vs. ϕ , in-line square, $s/d = 1.5$	155
Figure 59.	c_f vs. ϕ , rotated square, $s/d = 1.5$	156
Figure 60.	c_f vs. ϕ , equilateral triangle, $s/d = 1.5$	156
Figure 61.	Reference temperatures.	157
Figure 62.	$\overline{Nu}_{LM}$ vs. Pe compared to numerical data, in-line square, constant t_w	166
Figure 63.	j_{um} vs. Re compared to numerical data, in-line square, constant q_w	166
Figure 64.	j_{um} vs. Re compared to numerical data, rotated square, constant q_w	167
Figure 65.	Effect of wall boundary conditions on $\overline{St}$, $Pr = 1$	167
Figure 66.	$\overline{St}_{LM}$ vs. Re , staggered, constant t_w , $Pr = 1$	168
Figure 67.	Effect of t_{ref} on $\overline{St}$, in-line square, $s/d = 1.5$, $Pr = 1$	168
Figure 68.	Comparison of experimental data and correlations, in-line banks.	169
Figure 69.	Comparison of experimental data and correlations, staggered banks.	169
Figure 70.	$Nu_T/\overline{Nu}_T$ vs. ϕ , in-line square tube, $s/d = 1.5$, $Pr = 1$	170
Figure 71.	$Nu_H/\overline{Nu}_H$ vs. ϕ , in-line square tube, $s/d = 1.5$, $Pr = 1$	170
Figure 72.	$Nu_T/\overline{Nu}_T$ vs. ϕ , rotated square tube, $s/d = 2$, $Pr = 1$	171
Figure 73.	$Nu_H/\overline{Nu}_H$ vs. ϕ , rotated square tube, $s/d = 2$, $Pr = 1$	171
Figure 74.	$Nu_T/\overline{Nu}_T$ vs. ϕ , equilateral triangle, $s/d = 1.5$, $Pr = 1$	172
Figure 75.	$Nu_H/\overline{Nu}_H$ vs. ϕ , equilateral triangle, $s/d = 1.5$, $Pr = 1$	172
Figure 76.	$Nu_T/\overline{Nu}_T$ vs. ϕ , rotated square, $s/d = 1.25$, $Pr = 1$	173
Figure 77.	$Nu_H/\overline{Nu}_H$ vs. ϕ , in-line square, $s/d = 1.5$, $Re = 100$, various Pr	173
Figure 77.	Effect of grid size, in-line square, $s/d = 2$, $Re = 100$	188
Figure 78.	Effect of grid type, in-line square, $s/d = 2$, $Re = 100$	188

Figure 79.	Effect of grid type, equilateral triangle, $s/d = 2$, $Re = 100$.	189
Figure 80.	Domain boundaries.	196
Figure 81.	Boundary conditions.	198
Figure 82.	Response of a rotated square tube bank.	200
Figure 83.	Velocity vectors, in-line square, $Re = 300$.	202
Figure 84.	Stream-lines, in-line square, $Re = 300$.	203
Figure 85.	Pressure contours (Pa), in-line square, $Re = 300$.	204
Figure 86.	Temperature contours ($^{\circ}\text{C}$), in-line square, $Re = 300$.	205
Figure 87.	Velocity vectors, rotated square, $Re = 300$.	206
Figure 88.	Stream-lines, rotated square, $Re = 300$.	207
Figure 89.	Pressure contours (Pa), rotated square, $Re = 300$.	208
Figure 90.	Temperature contours ($^{\circ}\text{C}$), rotated square, $Re = 300$.	209
Figure 91.	Monitor-point values, in-line square, $Re = 1\,000$.	213
Figure 92.	Monitor-point values, rotated square, $Re = 300$.	214
Figure 93.	Sh vs. Re , in-line and staggered banks.	214
Figure 94.	Eu vs. Re , in-line and staggered banks.	215
Figure 95.	$\overline{St}_H$ vs. Re , in-line and staggered banks.	215
Figure 96.	Eu vs. t^* , in-line square.	216
Figure 97.	Eu vs. t^* , rotated square.	216
Figure 98.	c_L vs. t^* , in-line square.	217
Figure 99.	c_D vs. t^* , in-line square.	217
Figure 100.	c_L vs. t^* , rotated square.	218
Figure 101.	c_D vs. t^* , rotated square.	218
Figure 102.	$\overline{Nu}_H$ vs. t^* , in-line square.	219
Figure 103.	$\overline{Nu}_H$ vs. t^* , rotated square.	219
Figure 104.	c_P vs. t^* , in-line square, $Re = 300$.	220
Figure 105.	c_P vs. ϕ , various t^* , rotated square, $Re = 300$.	220

Figure 106.	c_f vs. ϕ , various t^* , in-line square, $Re = 300$.	221
Figure 107.	$Nu_H/\overline{Nu}_H$ vs. ϕ , various t^* , in-line square, $Re = 300$.	221
Figure 108.	Convergence test, in-line square bank.	233
Figure 109.	Convergence test, rotated square bank.	233
Figure 110.	Plate-fin fin and tube heat exchanger.	240
Figure 111.	Velocity vectors, in-line square, $l/d = 2$, $Re = 100$.	245
Figure 112.	Particle traces, in-line square, $l/d = 2$, $Re = 100$.	245
Figure 113.	Velocity vectors, rotated square, $l/d = 2$, $Re = 100$.	246
Figure 114.	Particle traces, rotated square, $l/d = 2$, $Re = 100$.	246
Figure 115.	Inlet velocity contours, rotated square, various l/d .	247
Figure 116.	Oil-flow traces, in-line square, $l/d = 2$.	248
Figure 117.	Oil-flow traces, rotated square, $s/d = 2$, $l/d = 2$.	249
Figure 118.	Oil-flow and particle traces, rotated square, $l/d = 2$, $Re = 300$.	250
Figure 119.	Temperature contours ($^{\circ}C$), rotated square, $l/d = 1/2$, $Re = 100$.	250
Figure 120.	Temperature contours ($^{\circ}C$), rotated square, $l/d = 10$, $Re = 100$.	251
Figure 121.	c_p contours, in-line square, $l/d = 2$.	252
Figure 122.	c_p contours, rotated square, $l/d = 2$.	253
Figure 123.	c_f contours, in-line square, $l/d = 2$.	254
Figure 124.	c_f contours, rotated square, $l/d = 2$.	255
Figure 125.	$Nu_T/\overline{Nu}_T$ contours, in-line square, $l/d = 2$, adiabatic fin.	256
Figure 126.	$Nu_T/\overline{Nu}_T$ contours, rotated square, $l/d = 2$, adiabatic fin.	257
Figure 127.	Rectangular offset fin plate-fin heat exchanger	267
Figure 128.	Velocity vectors, $Re = 500$.	273
Figure 129.	Pressure contours, $Re = 500$.	274
Figure 130.	Local heat transfer coefficient, $Re = 500$.	275
Figure 131.	Comparison of results with experimental data.	275

List of Tables

Table 1. B_{2n+1} coefficients for in-line square tube banks.....	62
Table 2. B_{2n+1} coefficients for rotated square banks.....	64
Table 3. B_{2n+1} coefficients for equilateral triangle banks.	66
Table 4. Summary of results, in-line square, $s/d = 1.25$, $Pr = 1$	163
Table 5. Summary of results, in-line square, $s/d = 1.5$, $Pr = 1$	163
Table 6. Summary of results, in-line square, $s/d = 2$, $Pr = 1$	163
Table 7. Summary of results, rotated square, $s/d = 1.25$, $Pr = 1$	164
Table 8. Summary of results, rotated square, $s/d = 1.5$, $Pr = 1$	164
Table 9. Summary of results, rotated square, $s/d = 2$, $Pr = 1$	164
Table 10. Summary of results, equilateral triangle, $s/d = 1.25$, $Pr = 1$	165
Table 11. Summary of results, equilateral triangle, $s/d = 1.5$, $Pr = 1$	165
Table 12. Summary of results, equilateral triangle, $s/d = 2$, $Pr = 1$	165
Table 13. Summary of results, in-line square.	210
Table 14. Summary of results, rotated square.	210
Table 15. Grid dependence tests, $Re = 300$	232
Table 16. Eu for in-line square arrangement, $s/d = 1.5$	243
Table 17. Eu for rotated square arrangement, $s/d = 2$	243
Table 18. $\overline{Nu}_{LM}$ for in-line square arrangement, $s/d = 1.5$, adiabatic fin.	244
Table 19. $\overline{Nu}_{LM}$ for in-line square arrangement, $s/d = 1.5$, constant t_w fin.	244
Table 20. $\overline{Nu}_{LM}$ for rotated square arrangement, $s/d = 2$, adiabatic fin.	244
Table 21. $\overline{Nu}_{LM}$ for rotated square arrangement, $s/d = 2$, constant t_w fin.	244
Table 22. Core dimensions.	268
Table 23. Fluid/Solid Properties	269
Table 24. Results for plate-fin heat exchanger.....	276

1. INTRODUCTION

This is not the first thesis to be written on the subject of fluid flow and heat transfer in heat exchangers; probably it is not the last to be written on the subject either. Basic knowledge of the characteristics of heat exchangers is an important subject, with many applications in the process and power industries. Experimental data are both expensive and difficult to obtain, and the fundamental flow patterns still not well understood.

Computational fluid dynamics (CFD) allows insight into a flow to be obtained by solving the equations of motion, using a computer. Many of the algorithms in use today are based on the finite-volume method, in the form pioneered by Patankar and Spalding (1972). At present, practical computer sizes and speeds still inhibit the direct simulation of flow and heat transfer in all but the smallest-scale heat exchangers. Spalding (1981 a,b) proposed a remedy in the form of a two-stage rationale:

- (i) a fine-scale simulation is performed within the actual passages of a 'typical' heat exchanger element in order to generate drag and heat transfer data.
- (ii) These data are then used as boundary conditions for a second simulation for the gross motion of the fluid in the unit as a whole.

This study is concerned primarily with the first of these two activities, namely the direct simulation of flow and heat transfer in typical elements of common heat exchangers.

1.1 Problem Characterisation

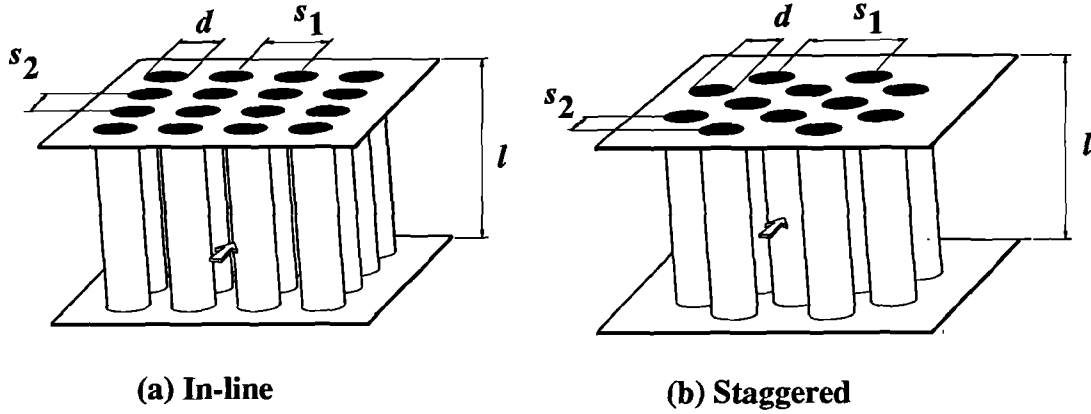


Figure 1. Tube banks.

Tube banks are one of the most common design elements used in heat exchangers. They typically consists of a number of tubes of length, l , arranged in either an in-line arrangement [Fig. 1a] or staggered arrangement [Fig. 1b] . The basic design parameters are the transverse and longitudinal pitch-to-diameter ratios, defined by,

$$a = \frac{s_1}{d} \quad (1.1)$$

$$b = \frac{s_2}{d} \quad (1.2)$$

where d is the cylinder diameter, s_1 the transverse pitch, and s_2 the longitudinal pitch, as illustrated in Fig. 1.

The most common tube bank arrangements are in-line square, equilateral triangle, and rotated square configurations. These are the three configurations that are considered in

this thesis. Under these circumstances, it is sufficient to characterise tube banks with a single pitch-to-diameter ratio, s/d , where s is the lateral pitch for in-line square, and the diagonal pitch for rotated square, and equilateral triangle banks. Tube banks are generally classified as being widely-spaced for values of $s/d > 2$ and compact for $s/d < 1.25$, the majority of tube banks falling somewhere in between. Although the flow may be at an arbitrary angle of attack, in this thesis only cross-flow is considered.

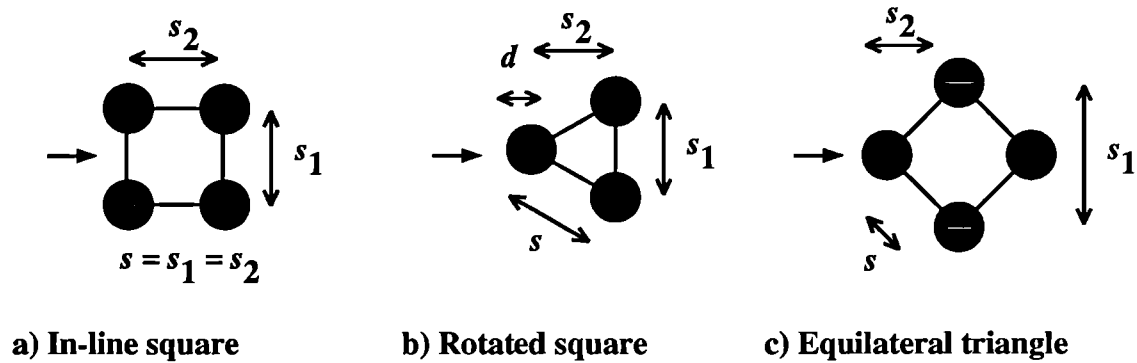


Figure 2. Tube bank nomenclature.

1.2 Objectives

The main objective of this research project was to increase understanding about the nature of flow and heat transfer in typical heat exchanger modules, by means of computational fluid dynamics. It might also be said that the objective of the work was to increase understanding about CFD through the study of heat exchangers.

Specific objectives included obtaining numerical and analytical solutions (where possible), comparing the results to previous work, both numerical and experimental, assessing the accuracy of the solution procedure, the implications of the boundary and other as-

sumptions made when setting up the problem to be solved, the suitability of various grid-types, and the merits and deficiencies of the particular numerical solver employed.

1.3 Contribution of the Thesis

In this thesis a number of individual, but related, new items are to be found:

The governing differential equations of motion are presented in general curvilinear form in two different ways; (1) the mathematical form, and (2) in terms of vector resolute and components. While the former form is relatively well-known, it is believed that the presentation of the the resolute/component form of differential equations together with their interpretation, represents a distinct and independent viewpoint.

An analytical solution for potential flow in tube banks has never been published. Therefore a potential flow analysis, based on complex variable theory, is derived, and used to analyse the flow-and pressure-fields associated with ideal fluid flow in tube banks.

A grid generation method, based on the solution of a Poisson-type equation is presented. The formulation is similar to previous ones, but has the advantage that when the source terms are set to zero, orthogonal grids result. This represents an improvement over other techniques.

Detailed results are presented of a numerical study using both orthogonal and non-orthogonal body-fitted grids for two-dimensional viscous flow and heat transfer in tube banks. Several different types of grid were employed, and the boundary conditions posed in such a fashion that they could be located in re-circulation zones. Numerous parametric studies were conducted. The results of the study are compared to previous experimental and numerical work.

The time dependent behaviour and stability of in-line and staggered tube banks to small lateral perturbations, applied upstream, is analysed. It is shown that such disturbances can be self-perpetuating. Tube banks exhibit transients qualitatively similar to vortex shedding behind single cylinders. The specific details of the flow patterns differ however, depending on geometry. Results are presented showing the qualitative details of the transient periodic behaviour in tube banks, together with transient and time-averaged data, showing quantitative effects on lift, drag, and heat transfer.

Three-dimensional effects of the plate-separation width, associated with tubes of finite length, l [Fig. 1], are considered. The results of flow visualisation studies as well as detailed pressure drop, skin friction, and local heat transfer data are presented.

1.4 General Methodology

The first stage in a computational study, normally, is to generate the grid. Several distinct types were investigated. Structured orthogonal and non-orthogonal body-fitted grids were initially generated using transfinite interpolation, followed by the numerical solution of elliptic partial differential equations.

The numerical scheme used to solve the governing equations for fluid flow and heat transfer is based on the *finite-volume method*. The discretization procedure, solution algorithms, and the computer program used were for the most part developed by previous researchers, however the computer program required to be substantially modified in order to be of a form suitable for the problems under consideration. Stream-wise periodic boundary conditions were incorporated into the computer program using a partially-explicit scheme involving the use of 'halo' cells. This scheme was employed so that boundary conditions (pressure, momentum, enthalpy, etc.) could be imposed in a manner that was synchronous with the flow algorithm. The boundary conditions may be

readily applied at boundaries where re-circulation occurs, and when there is significant non-orthogonality of the grid.

A problem involving a rectilinear grid; flow and heat transfer in a rectangular offset fin plate-fin heat exchanger, was selected prior to considering flow in tube banks. This case was chosen to gain experience, and validate the methodology used, with a problem using Cartesian co-ordinates (obviating many of the difficulties associated with body-fitted co-ordinates): The results have been included towards the end of the thesis.

Two-dimensional fluid flow and heat transfer in in-line and staggered tube banks were studied in detail. The results were compared with previous numerical analyses and also with experimental data. A potential flow solution was also generated from first principles, in order to provide a basis for comparison, free from the uncertainties of experimental results. Grid-dependence studies were conducted in order to discover the optimal type and size of grid.

It is now generally accepted that fluid flow in tube banks exhibits transient periodic behaviour for $Re > 10^2$ (depending on the bank configuration, and pitch-to-diameter ratio). The stability of laminar flow in tube banks was analysed, by applying an initial disturbance, and then re-applying the measured downstream responses upstream, in an iterative fashion. The resulting flow distribution was analysed, and the differences between the steady and unsteady-periodic flow-fields analysed in detail.

Much analytical and experimental data is based on the premise that the flow-field in tube banks is two-dimensional. One might speculate as to the circumstances under which the flow will be fully developed in one direction, yet not at all developed in the third dimension. In a deep tube bank, the flow in the longitudinal direction would also tend to be partially developed. Also tube banks with short length to diameter ratios are often deliberately used in plate-fin fin and tube heat exchangers. An analysis of fluid flow and heat

transfer for fully-developed three-dimensional steady flow was therefore undertaken, to investigate the effects of L/d on pressure drop, drag, and heat transfer, and investigate the hypothesis that the flow in real tube banks can be treated as two-dimensional.

The visualisation of complex three-dimensional flows is itself an on-going research subject. At the start of this thesis, sophisticated graphics software was not available to the author. An iterative scheme was developed in order to generate smooth 2-D and 3-D stream-lines. Although only the former could be completed in time for inclusion in the thesis, the theoretical development for the more general case has also been included.

1.5 Organisation of the Thesis

The thesis is organised as follows: Chapter 2 is a survey of literature containing relevant material, and is subdivided according to the contents of the thesis. Chapter 3 describes the basic transport equations in rectangular and general curvilinear co-ordinates. The latter are obtained as a transformation of the Cartesian equations. These are then shown to be the same as would be obtained from a 'control-volume' based approach, the physical significance of the form of the equations being stressed. Chapter 4 contains the potential flow analysis, together with tables and graphs of results.

The remainder of the thesis is concerned with the finite-volume-based numerical procedure. Chapter 5 is a description of the discretization procedure used to convert the differential equations of motion, given in Chapter 3, into linear algebraic equations as well as a description of the algorithms used to solve the finite-volume equations.

Grid generation is the subject matter of Chapter 6. The correct choice of grid being prerequisite for reliable numerical solution of the problem. This chapter describes the concepts and methodology involved in producing the various meshes used, together with il-

illustrations of typical grids used in the study. Chapter 7 is a brief description of the stream function generation methodology.

Chapter 8 contains the results of simulations for steady 2-D viscous flow. A complete description of the method used to code the periodic boundary conditions is presented together with the results of numerous parametric runs. These are compared to existing experimental and numerical data.

The stability of flow in tube banks to lateral perturbations applied upstream, is discussed in detail in Chapter 9. The results indicate that transient effects are apparent both in in-line and staggered tube banks. These assume the form of shear-layer instabilities in in-line tube banks, and Coanda-like wake-switching patterns in staggered arrangements. The time-dependent behaviour of pressure, lift, drag and heat transfer coefficients is illustrated together with flow visualisation results.

Chapters 10 and 11 are concerned with flow and heat transfer in plate-fin fin and tube heat exchangers, and rectangular offset plate-fin heat exchangers respectively. Chapter 12 contains some conclusions of a general nature together with suggestions for future research.

There are three Appendices: Appendix I contains a description of the laws of geometry and kinematics in general curvilinear geometry, prerequisite to understanding the transport equations [Chapter 3]. Appendix II details the method of enumeration of the elliptic functions used to derive the potential flow solution given in Chapter 4. Appendix III contains listings of some of the computer codes developed by the author as part of this thesis

1.6 Closure

The subject matter of the thesis was introduced. The main topics are; discussion of the basic equations, two-dimensional potential flow, steady flow and heat transfer, and unsteady flow in tube banks, three-dimensional flow and heat transfer in plate-fin heat exchangers and in fin and tube heat exchangers.

2. LITERATURE REVIEW

2.1 Introduction

The purpose of this Chapter is to introduce the reader to the subject matter of the thesis, by providing a review of relevant past research. Since entire books have now been written on the subjects of fluid flow, and heat transfer in tube banks, it is felt that an exhaustive review of all past work is unwarranted, and counter-productive. At the same time, it is intended to provide the reader with more than just a cursory introduction to the area. The text below is confined to a review of experimental and numerical work on flow and heat transfer in heat exchangers. Other subject-specific material is covered in the appropriate chapter.

2.2 Tube Banks

2.2.1 Experimental data

Numerous experimental studies have been conducted on fluid flow and heat transfer in tube banks. It would appear that Wallis and White (1938) were among the first to measure resistance to flow in tube banks. They measured overall pressure drop vs. Re , for a four-row in-line (1.5x2.6)¹ and a 7 row equilateral triangle (1.5x1.3) arrangement in the range $10^4 \leq Re \leq 3.7 \times 10^4$ and $5 \times 10^3 \leq Re \leq 3.5 \times 10^4$, respectively. Wallis (1939a,b) also conducted flow visualisation studies on tube banks. Pioneering work on heat trans-

¹For in-line square, rotated square and equilateral triangle banks, only the single pitch-to-diameter ratio, s/d , is normally given. For other arrays the two values given are the transfer and longitudinal pitches (axb)

fer in tube banks was conducted by Lohrisch (1929). This material, together with the earlier work of Thoma (1921), is discussed in the text by Jacob (1949).

Subsequently Pierson (1937) and Hoge (1937), obtained both fluid flow and heat transfer data which was correlated by Grimison (1937). Substantial data were gathered over a number of years at the University of Delaware: Omohundro et. al. (1949) measured overall pressure drop and heat transfer coefficients for a 10-row equilateral bank ($s/d = 1.25$) in the Re range 1 to 1 000. Bergelin et. al. (1949) presented similar results for in-line square and rotated square banks. Results for in-line square, rotated square and equilateral triangle tube banks were published by Bergelin et. al. (1950a), who provided correlations for the three main classifications of tube banks with $s/d = 1.25$ and 1.5 for $Re < 10^3$. Bergelin et. al. (1952) extended the results to Re of 10^4 . Most of the data from the various theses and reports of the Delaware group are documented in two reports (Bergelin et. al. 1950b, 1958). Elsewhere, overall pressure drop and heat transfer data were also reported by Jones and Monroe (1958), and Gram et. al. (1958).

Neal and Hitchcock (1967) obtained local heat transfer coefficients for flow at Re of 1.4×10^5 in a staggered bank $axb = 1.17 \times 1.33$. Hammeke et. al. (1967) studied in-line and staggered banks (2×1.4) to a maximum Re of 1.5×10^6 . Achenbach (1969) extended the local flow-field data to Re range of 10^7 for a 5 row staggered arrangement ($axb = 2 \times 1.4$). The thesis by Niggeschmidt (1975) contains details of heat transfer and drag data for in-line (2×1.25 , 2×1.5) and staggered (1.5×3.0 , 3.0×2) banks in the Re range $3 \times 10^3 \leq Re \leq 8 \times 10^5$. A later work by Achenbach (1989), contains heat transfer data for Re in the range $5 \times 10^4 \leq Re \leq 7 \times 10^6$ for a staggered (2×1.4) tube bank.

A very large set of data have been gathered at the Academy of Sciences in Kaunas, Lithuania. The recent book by Zukauskas et. al. (1988b) is a comprehensive discussion on fluid dynamics and flow-induced vibrations in tube banks, together with graphs of results. The article by Zukauskas and Ulinskas in the Heat Exchanger Design Handbook

(1983) is important, in that mathematical formulae for correlations of pressure drop and heat transfer are presented. A wide range of configurations and Re , for both in-line and staggered tube banks are considered. Correction factors for entry rows, variable viscosity etc., inclined flow and so forth, are also given. Heat transfer in tube banks is the subject a second book by Zukauskas and Ulinskas (1988a), and it includes correlations of Nu as a function of Re , together with numerous results in graphical form. The article by Zukauskas (1987) is a more compact discussion containing many of the same mathematical correlations as in the monograph above, however the formulae given in the Heat Exchanger Design Handbook, (1983) article by Gnielinski et. al., are substantially different (though apparently based upon the same group's data). An appendix to the earlier book by Zukauskas et. al. (1968) contains substantial tables of actual data on both heat transfer and fluid flow data, fluid properties etc. obtained from the authors' experimental programme of research.

A number of English language articles by the Kaunas group have appeared. These include papers by: Zukauskas and Ulinskas (1978a), average heat transfer and pressure drop in in-line and staggered banks $axb = 1.25 \times 1.25, 1.5 \times 1.5, 2 \times 2, 5 \times 10^4 \leq Re \leq 2 \times 10^6, 3 \leq Pr \leq 7$; Zukauskas and Ulinskas (1978b), overall and local heat transfer in in-line and staggered tube banks $axb = 1.25 \times 1.25, 1.5 \times 1.5, \text{ and } 2 \times 2, 1 \leq Re \leq 20\,000, 100 \leq Pr \leq 14\,000$; Zukauskas et. al. (1978c), local velocity distribution, frequency, turbulence and heat transfer in a 2×2 in-line bank, $Re = 500, Pr = 2\,540$; Zukauskas et. al. (1978d), average heat transfer and pressure drop of in-line and staggered tube banks, $axb = 1.5 \times 1.5 \text{ and } 2 \times 2, 1 \leq Re \leq 20\,000, 100 \leq Pr \leq 14\,000$; Zukauskas et. al. (1978e), effects of free stream turbulence and other parameters in transition regime. Poskas and Survila (1983) studied the turbulence distribution in staggered and in-line tube banks for $2 \times 10^5 \leq Re \leq 2 \times 10^6$, while Katinas et. al. (1989) measured transient forces in a row of tubes, and a staggered tube bank $10^3 \leq Re \leq 2 \times 10^5$.

Numerous other papers and articles on experimental results relating to specific aspects of flow and heat transfer in tube banks exist, the reader is referred to the bibliography of the books by Zukauskas and Ulinskas (1988a) and Zukauskas et. al. (1988b). The subject of transient flow in tube banks has also been the subject of research, primarily out of concern for flow-induced vibrations. The book by Zukauskas et. al. (1988) contains a substantial discussion of the subject, in addition the reader is also referred to the review paper Païdoussis (1982).

The Engineering Sciences Data Unit (ESDU), have correlated many of the data described above. Mathematical formulae are available for pressure drop (ESDU, 1980), and heat transfer (ESDU, 1973). Articles may also be found in a number of compendia on heat exchangers (Kays and London, 1964, Taborek et. al., 1983, Palen, 1986).

2.2.2 Numerical data

Thom and Apelt (1961) first considered laminar cross-flow through a staggered tube bank, $axb = 1.7 \times 2.5$, at $Re = 50$. Le Feuvre (1973) studied steady 2-D cross-flow and heat transfer in in-line tube banks by solving the transport equations using a finite difference scheme based on a stream function/vorticity approach. Constant wall temperature boundary conditions were assumed. Both laminar and turbulent flow were considered, the latter being modelled using the well-known $k-\epsilon$ model (Lauder and Spalding, 1974). Isihara and Bell (1975) obtained friction coefficients of tube banks at low Reynolds' numbers. Massey (1976), (also Lauder and Massey, 1978), using a similar approach, considered flow and heat transfer in staggered tube banks, assuming a constant heat flux boundary condition. Antonopoulos (1979, 1985, 1987a,b) studied both in-line and staggered geometries, under conditions of cross-flow, longitudinal flow and combinations of the two. Antonopoulos employed structured orthogonal body-fitted co-ordinate (BFC) grids, introducing source terms into the finite-volume equations to account for grid-

curvature effects. Fujii et. al. (1984) solved the stream function-vorticity equations for a five-row in-line tube bank. Peric (1985) and Demirdzic et. al. (1987) considered turbulent flow over staggered tube banks using non-orthogonal BFC grids.

Finite-element studies have also been published by Dhaubhadel et. al. (1987), who solved the two-dimensional Navier Stokes' and energy equations for a 5-row staggered equilateral triangle and rotated square banks with $s/d = 1.5$ and 2. Chang et. al. (1989) used a Galerkin finite-element method to solve the conservation equations cast in the stream function-vorticity formulation for in-line tube banks ($s/d = 1.5, 2$). Chen et. al. (1987) and Wong et. al. (1988) considered laminar flow and mixed convection past groupings of three and four isothermal cylinders in equilateral triangle and in-line square configurations. Faghri and Rao (1987) considered flow and heat transfer in plain and longitudinal-finned tube banks.

The writer is not aware of any transient simulations of laminar-periodic flow in tube banks, however Chang and Song (1990) published results for vortex shedding behind a pair of cylinders. Stuhmiller et. al. (1988) conducted large-eddy simulations of flow in in-line square and equilateral triangular tube banks, and reported on the spectral distribution of the turbulence.

2.3 Fin and Tube Heat Exchangers

2.3.1 Experimental data

Rich (1973, 1975) obtained data on a number of staggered plate-fin tube bank arrangements. McQuiston (1978), obtained mathematical correlations for overall friction and heat transfer. Saboya (1974), and Saboya and Sparrow (1974, 1976a,b) measured local

heat transfer coefficients in one, two and three row plate-fin and tube configurations by virtue of the analogy between heat and mass transfer. Krueckels and Kottke (1970) measured local mass transfer in a two-row in-line configuration. Webb (1980) contains reviews of these and other results; another article by Webb on the subject may be found in the book by Palen (1986).

2.4 Plate-fin Heat Exchangers

2.4.1 Experimental data

London and Shah (1968) measured overall heat transfer and pressure drop data on eight offset fin surfaces. Complete details of the experimental procedures and the data are to be found in the reports by Shah (1967) and Shah and London (1967). Results are also presented in the book by Kays and London (1964). Weiting (1975) proposed correlation formulae based on the data of Kays and London (1964), Shah and London (1968), London and Ferguson (1946), and Walters (1969).

2.4.2 Numerical models

Sparrow et. al. (1977) studied steady two-dimensional flow and heat transfer in offset fins, assuming the flow to be fully-developed with constant heat flux boundary conditions. Patankar and Prakash (1981) investigated the effects of finite-plate thickness on the flow. Kelkhar and Patankar (1989) considered three-dimensional flow, again neglecting the fin width.

2.5 Closure

This Chapter presented a survey of salient research material relating to the subject matter of the thesis.

3. BASIC EQUATIONS

3.1 Vector Equations

The basic equations of fluid mechanics and heat transfer are well known, see for example the monograph by Schlichting (1968). The flow is presumed to be incompressible, with constant properties, thus the flux laws assume the simplified divergence-free forms.

The equation of mass continuity is just,

$$\vec{\nabla} \cdot (\rho \vec{u}) = 0 \quad (3.1)$$

The principle of conservation of linear momentum for a linear viscous fluid reduces to the Navier-Stokes' equations,

$$\frac{\partial}{\partial t}(\rho \vec{u}) + \vec{\nabla} \cdot (\rho \vec{u} \otimes \vec{u}) = -\vec{\nabla} p + \vec{\nabla} \cdot \mu \vec{\nabla} \vec{u} \quad (3.2)$$

The enthalpy equation is just,

$$\frac{\partial}{\partial t}(\rho h) + \vec{\nabla} \cdot (\rho \vec{u} h) = \vec{\nabla} \cdot k \vec{\nabla} T \quad (3.3)$$

Equations 3.1 to 3.3 may be considered to have the general form,

$$\frac{\partial}{\partial t}(\rho \phi) + \vec{\nabla} \cdot (\rho \vec{u} \phi) = \nabla \cdot I \nabla \phi + S_\phi$$

TRANSIENT + CONVECTION = DIFFUSION + SOURCE

 (3.4)

consisting of the transport of some property ϕ as a result of the combined influence of transience, convection, diffusion and source-related effects. The left side of Equation 3.4 is the so-called Eulerian-divergence form of the material derivative of ϕ , while the right side usually represents constitutive laws. The Navier-Stokes' Equations may be regarded as the equation of the material derivative of three velocity components to viscous (diffusion) and pressure gradient (source) effects, while the continuity equation is obtained by taking ϕ to be unity (1), with only convection active.

In this thesis, only laminar incompressible constant-property flow is considered, so the transient term has been omitted from the continuity equation, Eq. 3.1, the viscous flux term in Eq. 3.2 has been simplified, and also the dissipation, Φ , has been neglected from the enthalpy equation Eq. 3.3. Other than this the general form has deliberately been retained (fluid properties, ρ , μ , k , etc., being left within parenthesis) so that the more universal forms can easily be recovered.

Equations 3.1 to 3.3 are vector equations, which are true for any co-ordinate system.

The Cartesian-component form of these equations is so well known, that their inclusion in this thesis is hardly justified. The vector form is considered to be an adequate basis for that which follows. One may therefore proceed directly forming the equations of motion in general curvilinear co-ordinates.

3.2 Curvilinear Co-ordinates

3.2.1 Introduction

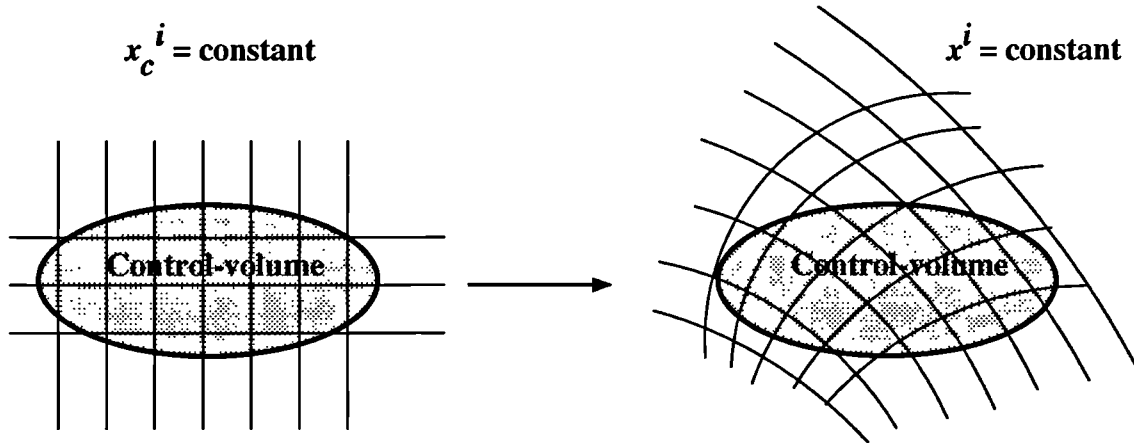


Figure 3. Curvilinear co-ordinate system.

Curvilinear co-ordinates occur, in Euclidean geometry, as the result of some mathematical transformation. The domain of the mapping (e.g. lines of constant Cartesian co-ordinates x_c) is normally called the 'physical' region, and the range (constant curvilinear co-ordinates x^j) the 'transformed' region. The reader is cautioned that it is the co-ordinate system, not the continuum, that is being mapped. The governing differential equations may be transformed, using the chain-rule for partial differentiation, to the so-called 'mathematical' (displacement and velocity vectors) components. The primitive variables may also be chosen as the 'physical' components or resolute. The purpose of this section is to show how the mathematical form reduces to the physical form, i.e. to a set of differential equations with the same characteristics as the Cartesian equations which may readily be integrated to the desired form. Although, there is nothing wrong with a mathematical approach, it is the opinion of the author that the physical approach

is more appropriate for numerical work, where knowledge of the co-ordinate transformation used to generate the grid, is irrelevant to the flow solver.

In deriving the finite-volume equations [Chapter 5] here and elsewhere, the flow is often treated as being 'locally one-dimensional'. This simplifies the problem-formulation, to the point that a rigorous derivation of the Navier-Stokes' equations in general curvilinear co-ordinates is hardly necessary (Malin et. al., 1985). There are however a number of situations where knowledge of the curvilinear form is necessary. For purely formal reasons, it is desirable to have a precise mathematical statement of the problem. Although a number of recent theses and papers (Parameswaran, 1985, Karki, 1986, Burns and Wilkes, 1987, Davidson and Hedberg, 1989, Demirdzic et. al., 1987, Melaaen, 1990, 1992) as well as older texts (Aris, 1962), have addressed this problem; none of these authors positions' are identical to this writer's.

Use is made below of the familiar concept of a 'control-volume'. The reader will appreciate that associated this concept are to two different connotations,

- (i) A volume, V , containing a manifold of fluid particles, which is locally aligned with the co-ordinate system (either Cartesian or curvilinear). In the limit $V \rightarrow 0$, the boundaries of the control-surface correspond to those of the co-ordinate system. This concept is useful to illustrate physical aspects of transport equations, and provides a convenient mental picture.
- (ii) A finite-volume of entirely arbitrary shape, i.e. not necessarily conforming to the co-ordinate system [Fig. 3].

Although the utilitarian value of (i) is apparent, the reader will appreciate that all forms of the transport equations, presented below, are equally true for case (ii).

The mathematical form of the transport equations are presented, and shown to be equivalent to those obtained from physical reasoning. In the interests of brevity, substantial material has been relegated to Appendix I, it being assumed in the body of the text that the reader is familiar with the concepts of covariance and contravariance, metric tensors, and covariant differentiation. It is virtually impossible to employ a nomenclature that is both compact and simple: The comma is used below together with the operator δ to denote covariant differentiation, or in the latter case differentiation with respect to a locally-fixed tangential direction. Partial differentiation is denoted by the usual symbol ∂ (these are identical when the differend is a scalar). Henceforth in this chapter alone, the summation convention is presumed: If an index occurs precisely twice; once as a superscript and once as a subscript, it is summed, exceptions are noted as necessary. Following convention, a superscript index occurring in the numerator of a differential operation is always treated as a subscript, e.g.

$$\frac{\delta \phi}{\delta x^i} = \phi_{,i} \quad (3.5)$$

3.2.2 Mathematical form

The divergence form of the equations of motion are as follows: The equation of mass continuity is just,

$$(\rho u^i)_{,i} = 0 \quad (3.6)$$

where u^i are the contravariant components of the velocity vector. The Conservation equation for some generalised property, ϕ , may be written as,

$$\frac{\partial}{\partial t}(\rho\phi) + (\rho u^i \phi)_{,i} = \left(g^{ij} \Gamma \frac{\partial \phi}{\partial x^j} \right)_{,j} + S_\phi \quad (3.7)$$

Either the covariant, u_i , or the contravariant u^i , velocity components, may be selected as the independent variables in the momentum equations. The contravariant form is,

$$\frac{\partial}{\partial t}(\rho u^i) + (\rho u^j u^i)_{,j} = -g^{ij} \frac{\partial p}{\partial x^j} + (g^{jk} \mu u_{i;j})_{,k} \quad (3.8)$$

while the covariant form is

$$\frac{\partial}{\partial t}(\rho u_i) + (\rho u^j u_i)_{,j} = -\frac{\partial p}{\partial x^i} + (g^{jk} \mu u_{i;j})_{,k} \quad (3.9)$$

The presence of the contravariant metric tensor, g^{ij} , in the diffusion term [Eq. 3.7] and the pressure gradient term [Eq. 3.8] effects the conversion of the covariant gradient of a scalar quantity to the appropriate contravariant form. The metric tensors have been shown in parenthesis in Eqs. 3.6 to 3.9, however they are commutative with the operation of covariant differential (Ricci's Lemma). Let it be noted that;

- (i) The dimensions of u^i and u_i do not in general correspond to speed $[LT^{-1}]$, nor those of dx^i to displacement $[L]$. It is desirable to convert these equations to less abstract forms involving physically meaningful quantities (mass, momentum, energy etc.) as a function of the 'physical' displacement $[L]$.

- (ii) The use of the covariant derivative obviates the need to account for the change in magnitude and direction of the basis vectors in the differential equations. When integrating over a finite-volume it is, however, necessary to account for these changes.

The ordinary derivative forms of the transport equations is easily derived, for example,

$$\sqrt{g} \frac{\partial}{\partial t}(\rho\phi) + \frac{\partial}{\partial x^i}(\sqrt{g}\rho u^i\phi) = \frac{\partial}{\partial x^j} \left(g^{ij} \sqrt{g} \Gamma \frac{\partial \phi}{\partial x^i} \right) + \sqrt{g} S_\phi \quad (3.10)$$

The specific forms (mass, momentum, energy, etc.) are easily obtained. The reader will note that a fixed co-ordinate system is presumed: In the event that co-ordinate system were non-inertial, the term $\sqrt{g}$ would be inside the time derivative in Eq. 3.10 (a similar term would arise in the transient terms in Eqs. 3.7 etc.)

3.2.3 Resolute and component forms

The above form of the transport equations are of great interest. They are not, however, the equations which are actually solved. Equation 3.7, for example, contains terms such as metric coefficients, which are neither required nor ever computed by the flow solver. The transport equations will therefore be written as differential equations in ‘physical’ form; these alone form the basis for the numerical integration scheme. If this is done it is possible to highlight the physical significance of the governing differential equations.

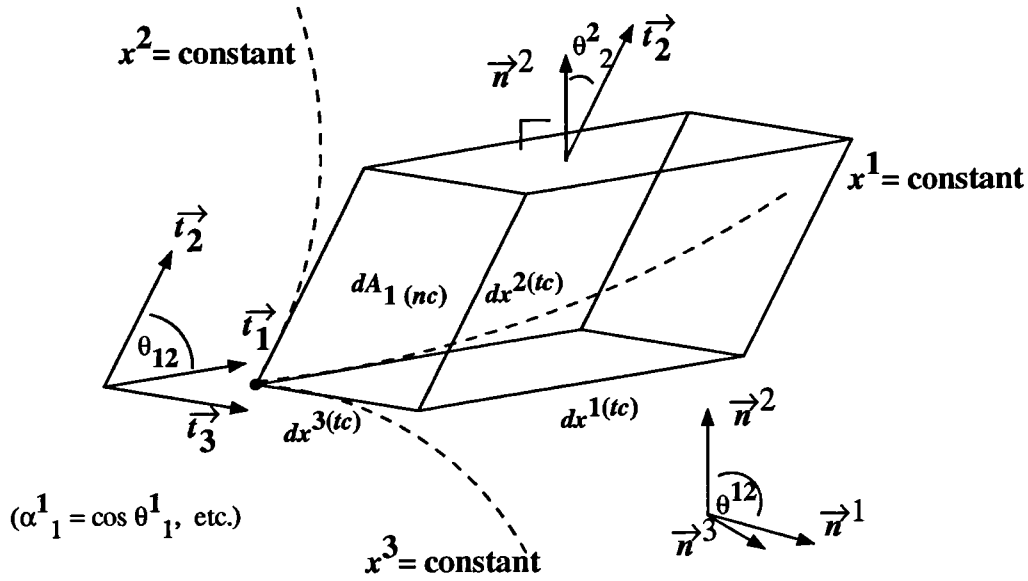


Figure 4. Control-volume in physical space.

Following Ricci and Levi-Cevita (1901), four distinct quantities having physical significance may be introduced:

“Deux systèmes réciproques simples v_i et v^i étant donnés, on peut, quelles que soient les coordonnées dx^1, dx^2, dx^3 de l'espace, regarder les expressions $v_i/\sqrt{g_{ii}}$ et $v^i/\sqrt{g^{ii}}$ comme celles des projections orthogonales d'un même vecteur sur les tangentes aux lignes coordonnées dx^i , et sur les normales aux surfaces coordonnées dx^i , pendant que les expressions $\sqrt{g_{ii}}v^i$ et $\sqrt{g^{ii}}v_i$ représentent les composantes du même vecteur respectivement selon les mêmes lignes et les mêmes normales.” (notation mine)

These will be referred to as the vector resolutes (projections) and the vector components (sometimes called ‘physical’ components). Both resolutes and components may be constructed in terms of unit tangents to the co-ordinate lines, $\vec{t}_i$, or unit normals, $\vec{n}^i$. Let the tangential and normal resolutes be denoted by $v_{i(tr)}$ and $v^{i(nr)}$, and the tangential and normal components by $v^{i(tc)}$ and $v_{i(nc)}$. The location of the indices has been selected in order to permit retention of the summation convention.

The transport equations may be written either in component or in resolute form. The latter are adopted here. The tangential displacement components, $dx^{i(tc)}$, are the obvi-

ous choice for the dependent variables. It can be shown that the metric tensors g_{ij} and g^{ij} , in the mathematical form of the equations [Eq. 3.6 to 3.9], are supplanted by the direction-cosines of the angles between the co-ordinate direction. With reference to Fig. 4, let α_{ij} denote the direction cosines between tangent vectors, $\vec{t}_i \cdot \vec{t}_j$, α^{ij} between normal vectors, $\vec{n}^i \cdot \vec{n}^j$, and α_j^i between $\vec{n}^i \cdot \vec{t}_j$. Furthermore, let $\bar{\alpha}^{ij}$, $\bar{\alpha}_{ij}$ and $\bar{\alpha}_i^j$ signify the inverse (matrix components) of α_{ij} , α^{ij} , and α_j^i , respectively ($\bar{\alpha}^{ij} = \alpha_{ij}^{-1}$ etc.). These direction-cosines, and their inverses, effect conversion of components to resolute, and vice versa.

Vector resolute possess the important property that if a vector, $\vec{q}$, is proportional to the gradient of some scalar, ϕ ,

$$\vec{q} = -I \vec{\nabla} \phi \quad (3.11)$$

then it is the tangential resolute of $\vec{q}$ which are given by,

$$q_{i(tr)} = -I \frac{\partial \phi}{\partial x^{i(tc)}} \quad (3.12)$$

The normal resolute are required to compute the flux (of ϕ) across a co-ordinate surface.

$$q^{i(nr)} = \alpha^{ij} q_{j(nc)} = \alpha^{ij} \bar{\alpha}_j^k q_{k(tr)} \quad (3.13)$$

The term $\bar{\alpha}_j^k$ convert the tangential resolute to normal components and α^{ij} converts these to normal resolute. Note that if,

$$\vec{v} = v^{i(tc)} \vec{t}_i = v_{i(nc)} \vec{n}^i \quad (3.14)$$

is some arbitrary unit vector, then $q_{i(tr)}v^{i(tc)}$ is the projection of $\vec{q}$ in the direction parallel to $\vec{v}$, whereas $q^{i(nr)}v_{i(nc)}$ is the projection of $\vec{q}$ in the direction with normal $\vec{v}$.

The scalar product of any two vectors is the product of the resolute of one vector and the components of the other [Appendix I]. Thus the normal flux of ϕ across an arbitrarily-oriented area, $d\vec{A}$, may be written equivalently as $q^{i(nr)}dA_{i(nc)}$ or $q_{i(nc)}dA^{i(nr)}$ (i.e. either project $\vec{q}$ normal to $d\vec{A}$ or vice versa). Note also that the magnitude of a vector $\vec{q}$ is neither that of the resolute nor the components, but the (square root) of the sum of their product.

These simple concepts may also be extended to second-order-tensors, to which are associated sets of matrix-components¹, resolute, and mixed resolute/components. These also have physical significance: Suppose a second-order tensor, s , is identified with $\text{grad } \vec{u}$ (incompressible flow). Due to the form of the scalar product above, it is apparent that $s_{i(tr)}j_{(tr)}$ defined as $s_{ij}\sqrt{g_{ii}}\sqrt{g_{jj}}$ (unsummed) [see Appendix I] are the tangential resolute acting on planes with normals perpendicular to $\vec{i}_i$. The resolute, $s_{i(tr)}^{j(nr)}$, acting on surfaces perpendicular to $\vec{n}^i$ are then,

$$s_{i(tr)}^{j(nr)} = \alpha^{jk}\bar{\alpha}_k^l s_{i(tr)l(tr)} = \alpha^{jk}\bar{\alpha}_k^l \frac{\partial u_{i(tr)}}{\partial x^{l(tc)}} \quad (3.15)$$

The tangent-direction resolute of the viscous-stress vector, $\vec{\sigma}(\vec{v})$, acting on plane perpendicular to some arbitrary unit-direction vector $\vec{u}$ are therefore given by,

$$\sigma(\vec{v})_{i(tr)} = \mu s_{i(tr)}^{j(nr)} u_{i(nc)} \quad (3.16)$$

¹ These are not, strictly speaking, the components of a tensor, as they do not transform according to the usual laws of covariance and contravariance.

The resolute forms of the equations of motion follow. The continuity equation is,

$$\frac{\delta}{\delta x^{j(tc)}} \left(\bar{\alpha}_i^j \rho u^{i(nr)} \right) = 0 \quad (3.17)$$

The transport of some scalar ϕ , such as enthalpy is,

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\delta}{\delta x^{j(tc)}} \left(\bar{\alpha}_i^j \rho u^{i(nr)} \phi \right) = - \frac{\delta}{\delta x^{j(tc)}} \left(\bar{\alpha}_i^j q^{i(nr)} \right) + S_\phi \quad (3.18)$$

The tangential-resolute form of the momentum equations is,

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\delta}{\delta x^{k(tc)}} \left(\bar{\alpha}_j^k \rho u^{j(nr)} u_i \right) = - \frac{\partial p}{\partial x^{i(tc)}} + \frac{\delta}{\delta x^{k(tc)}} \left(\bar{\alpha}_j^k \mu s_i^{j(nr)} \right) \quad (3.19)$$

Equations 3.17 to 3.19 are the transport equations in terms of the tangential derivatives. i.e. with respect to a fixed oblique rectilinear-co-ordinate system.

A convenient mental picture of Eqs. 3.17 to 3.19 may be formed by considering a control-volume in the form of an exact parallelepiped, i.e. one where opposite faces are parallel, which in the limit $V \rightarrow 0$ is parallel to the curvilinear co-ordinate system [Fig. 5a]. The normal resolutives appear naturally in the convection and diffusion terms [Eq. 3.18]. These correspond to the flux-terms in the control-volume derivation. For the purpose of visualising the final form of the equations of motion discussed in this thesis, one may construct a control-volume which is, not only parallel to the co-ordinate system in the limit $V \rightarrow 0$, but also conforms with the co-ordinate surfaces. Figure 5b is a schematic, shown for convenience in two-dimensions.

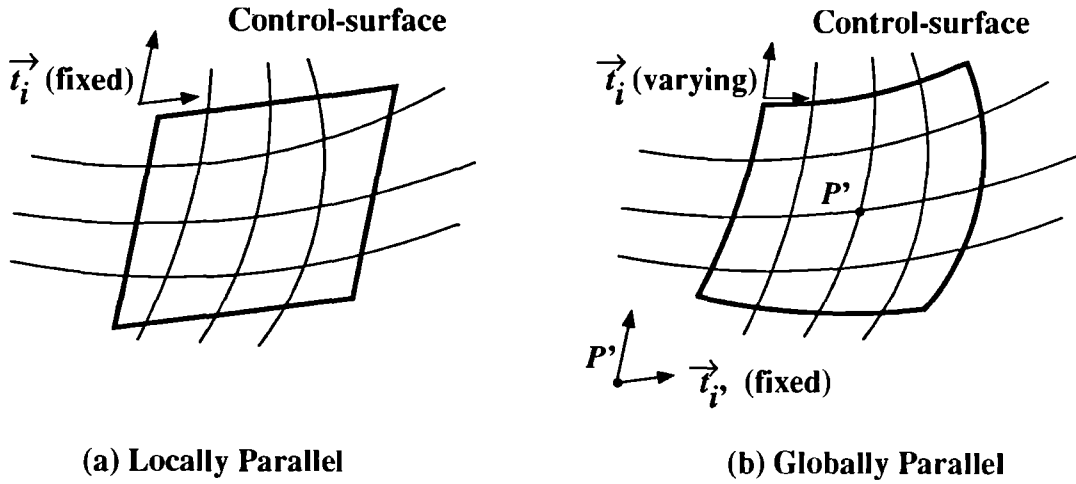


Figure 5. Control-volume parallel to co-ordinate system.

3.2.4 Ordinary derivative form

In the ordinary derivative form, first-order changes in the direction of the basis vectors must be accounted for. The cell-volume is still approximately that of a parallelepiped, however changes in elemental areas-per-unit volume, or angles, are no longer negligible.

The final form for the continuity equation is,

$$\frac{\partial}{\partial x^{j(tc)}} \left(\sqrt{\alpha} \bar{\alpha}_i^j \rho u^{i(nr)} \right) = 0 \quad (3.20)$$

The scalar transport equation is,

$$\boxed{\sqrt{\alpha} \frac{\partial}{\partial t} (\rho \phi) + \frac{\partial}{\partial x^{j(tc)}} \left(\sqrt{\alpha} \bar{\alpha}_i^j \rho u^{i(nr)} \phi \right) = - \frac{\partial}{\partial x^{j(tc)}} \left(\sqrt{\alpha} \bar{\alpha}_i^j q^{i(nr)} \right) + \sqrt{\alpha} S_\phi} \quad (3.21)$$

where the tangential resolute of $\vec{q}$ are defined by the constitutive equation [Eq. 3.11], the conversion to normal resolute is effected using Eq. 3.13, and $\sqrt{\alpha} = \det \alpha_{ij}$

Regarding conservation of momentum; when computing $\text{grad } \vec{u}$ (in terms of the partial derivative), it is necessary to account for changes in the base directions, for example,

$$s_{ij}(tr)(tr) = \frac{\partial u_{i(tr)}}{\partial x^{j(tc)}} = \frac{\partial u_{i(tr)}}{\partial x^{j(tc)}} + u^{k(tc)} \frac{\partial \vec{t}_k}{\partial x^{j(tc)}} \cdot \vec{t}_i \quad (3.22)$$

The significance of the second term on the right side of Eq. 3.22 being only that the resolute at $x^{i(tc)} + dx^{i(tc)}$ must be converted to components, then projected in the co-ordinate directions at $x^{i(tc)}$.

It is straightforward to substitute Eq. 3.22 into Eq. 3.19, and simplify. However, one additional step is required, in anticipation of integrating these equations over a finite-volume [Chapter 5] namely that the components in the momentum equation must be projected in fixed directions: Since the unit tangents and normals are uniquely defined at every point, P , in space, when integrating a vector equation over a finite-volume, V , i.e. a manifold of such points, the appropriate choice of basis vectors presents a dilemma. These are therefore selected as being parallel to the tangent vectors at some fixed point. P' , in the interior of the control-volume [Fig. 5b]. Then defining,

$$u'_{i'(tr)} \equiv \vec{u} \cdot \vec{t}_{i'} \quad (3.23)$$

where primes are used to identify quantities referred to the fixed basis. The momentum equation may be finally be written as,

$$\begin{aligned} & \sqrt{\alpha} \frac{\partial}{\partial t} \left(\rho u'_{i'(tr)} \right) + \frac{\partial}{\partial x^{k(tc)}} \left(\sqrt{\alpha} \bar{\alpha}_j^k \rho w^{j(nr)} u'_{i'(tr)} \right) \\ & = -\sqrt{\alpha} \frac{\partial p}{\partial x^{i'(tc)}} + \frac{\partial}{\partial x^{k(tc)}} \left(\sqrt{\alpha} \bar{\alpha}_j^k \mu s_{i'}^{j(nr)} \right) \end{aligned} \quad (3.24)$$

3.2.5 Discussion

Equation 3.21 is an important equation, expressing the transport of some general quantity ϕ in general curvilinear co-ordinates. The writer is not aware of the transport equation having been written in this particular simple form, previously. It is essentially the same as the Cartesian co-ordinate form [Eq. 3.4], with the presence of additional $\sqrt{\alpha}$ and $\overline{\alpha}^j_i$ terms. Transport equations represent conservation per unit volume, the differential element of volume, dV , may be written as, for example,

$$dV = dA_{1(nc)} dx^{1(tc)} \alpha^1_1 = \sqrt{\alpha} dx^{1(tc)} dx^{2(tc)} dx^{3(tc)} \quad (3.25)$$

giving rise to the extra terms in $\overline{\alpha}^j_i$ in the numerator of Eq. 3.21. For this very reason, the $\sqrt{\alpha}$ and $\overline{\alpha}^j_i$ terms vanish when Eq. 3.21 is integrated over a finite-volume, provided the volume is aligned with the co-ordinate axes [Fig. 5b]. Thus, although Eq. 3.21 is also the case, for arbitrarily-shaped finite volumes, it is not in a particularly useful form unless the surfaces of the finite-volume coincide with the co-ordinate axes. Eq. 3.21 is similar in appearance to Eq. 3.18 with the terms in $\sqrt{\alpha}$ absent: Because Eq. 3.18 contains the tangential derivative, the $\overline{\alpha}^j_i$ commute with respect to the differential operator. This is not true for Eq. 3.21. The scalar transport equation could also have been written in component form,

$$\sqrt{\alpha} \frac{\partial \phi}{\partial t} + \frac{\partial}{\partial x^{j(tc)}} \left(\sqrt{\alpha} u^{i(tc)} \phi \right) = - \frac{\partial}{\partial x^{j(tc)}} \left(\sqrt{\alpha} q^{j(tc)} \right) + \sqrt{\alpha} S_\phi \quad (3.26)$$

However there is no significant advantage over the resolute form.

The form of the momentum-equation given in Eq. 3.24 is essentially the same as Eq. 3.21. Thus, as is the case for Cartesian co-ordinates, Eq. 3.21 may be considered to

govern the transport of momentum with $\phi = u_i'_{(tr)}$, provided the conserved velocity values are projected in fixed-directions. There is no problem in writing the differential momentum equations in terms of resolutes or components in the local curvilinear directions. However a basis is needed when integrating these over a finite-volume. Whether the components are projected in locally-fixed directions prior to, or after integration is largely a matter of choice. The author's preference for the former is consistent with that of previous authors (Davidson and Hedberg, 1989, Melaaen, 1990). Equations 3.17 to 3.18 can of course, be written in terms of any of the resolutes or components: From a mathematical point-of-view all forms are entirely equivalent. The selection of the tangential resolutes as the dependent variables in the momentum equations, is admittedly somewhat arbitrary. This particular form has the slight advantage that the pressure gradient term requires no modification.

This concept of a control-volume aligned with the curvilinear co-ordinate system was exploited in this chapter. This approach parallels the use of a fixed-rectangular control volume when deriving the Cartesian-form of the equations, e.g. in elementary fluid mechanics' texts. The reader will appreciate that there is no requirement for a control-volume to be aligned with the grid; all forms of the equations, including the Cartesian form, are valid for any element of arbitrary volume: The partial-derivative-based mathematical form [Eq. 3.10] is obtained directly, using the chain-rule for differentiation. The presence of the Jacobian, $\sqrt{g}$, in the spatial derivatives accounts for the change in volume under the transformation from rectilinear elements-of-volume to rectangular parallelepipeds. The resolute form [Eq. 3.21] is obtained by algebraic manipulation of the mathematical form. The units of the variables in the resolute form are the usual ones of length, velocity, etc. (unlike Eq. 3.10). Although these forms of the transport equations are universal, the utilitarian value of the resolute form is particularly apparent, when it is required to integrate ϕ over a finite-volume which is aligned with the co-ordinate system.

3.3 Closure

Theory fundamental to the methods used in the rest of the thesis was presented in this chapter. The transport equations were presented in vector and general-curvilinear form. The curvilinear form may be written in terms of mathematical components, (physical) components or resolute. The latter were selected. The physical significance of the various terms were discussed.

4. POTENTIAL FLOW IN TUBE BANKS

4.1 Introduction

The scope of the analysis is restricted to steady, two-dimensional, incompressible potential flow. Only a brief review is provided here, the reader is referred to the monographs by Lamb (1957) and Milne-Thomson (1955) for further details.

A Potential flow is an inviscid flow, so that the momentum equation, Eq. 3.2 reduces to Euler's Equation. In the absence of body forces, Bernoulli's Law is obtained by integration as,

$$\frac{p}{\rho} + \frac{1}{2} |\vec{u}|^2 = C \quad (4.1)$$

In potential flow, the motion of the fluid is irrotational, and the velocity is the negative gradient of a scalar potential, defined by,

$$\vec{u} = - \vec{\nabla} \phi \quad (4.2)$$

The flow is presumed to be two-dimensional so a stream function, ψ , may also be defined [Eq. 7.2]. It is convenient to introduce a complex potential, W , by

$$W = \phi + i\psi \quad (4.3)$$

The complex potential, W , may be considered to be a function of the complex displacement, z ,

$$z = x_c + iy_c \quad (4.4)$$

A necessary condition that $W(z)$ is a holomorphic function (i.e. a function of z only), is that the functions ϕ and ψ are conjugate functions, i.e. ones which satisfy the Cauchy-Riemann equations,

$$\frac{\partial \phi}{\partial x_c} = \frac{\partial \psi}{\partial y_c}, \quad \frac{\partial \psi}{\partial x_c} = -\frac{\partial \phi}{\partial y_c} \quad (4.5)$$

It may readily be shown, (Milne-Thomson, 1955) that the complex potential, $W(z)$, must be a harmonic function satisfying Laplace's equation,

$$\nabla^2 W = 0 \quad (4.6)$$

Because Laplace's equation is a linear differential equation the principle of superposition may be utilised. Thus many potential flow problems may be reduced to finding a set of suitable harmonic function, which are then combined so as to satisfy the boundary conditions.

A complex velocity, U , is defined by,

$$U \equiv u - iv = -\frac{dW}{dz} \quad (4.7)$$

4.1.1 Potential flow in tube banks

The subject of potential flow past a doubly-periodic array of circular cylinders is a subject primarily of academic interest. Potential flow offers one of the few areas where an exact analytical solution may be obtained. Knowledge of the potential flow solution also provides insight into the nature of the flow, although the velocity field in a real fluid will differ significantly from that in a perfect fluid, it is to be anticipated that the pressure fields may be somewhat similar.

Dalton and Helfinstine (1971) and Yamamoto (1976) demonstrated how problems involving a number of cylinders may be solved using image doublets. Legendre (1967) and Drotz (1987) investigated the problem from a more abstract point of view. Chen (1978) and Païdoussis et. al. (1984) obtained Fourier series for potential functions associated with groups of cylinders. Suh et. al. (1989) proposed an expression for the complex potential based on a set doubly-periodic of doublets, combined with a linear term, corresponding to uniform flow. A number of solutions to other problems involving groups of circular cylinders have been obtained; Howland and McMullen (1936) provide an early review of some of these. Hsu (1964) later used the analysis of Howland and McMullen (1936) for a double row of cylinders, to model heat transfer in liquid metal flows involving tube banks.

Although the analytical tools, required to solve the problem of a doubly-periodic array of cylinders, have been available for more than half a century, there do not appear to be published articles giving solutions of the equations of motion for the more usual configurations of in-line and staggered banks of cylinders, at various pitch-to-diameter ratios. The purpose of this chapter is to provide such a solution. The methodology employed is similar to that of Knight and McMullen (1937). Analytic expressions for the potential field in the form of a power series are derived. Coefficients are tabulated for

in-line square, rotated square and equilateral triangular arrays at various pitch-to-diameter ratios.

4.2 Mathematical Details

It can be shown (Batchelor, 1967) that the solution for any potential flow involving a single cylinder, located at the origin in the complex plane, may be expressed in terms of a series of circular harmonics, that is to say in the form,

$$W = \sum_{n=0}^{\infty} A_n W_n \quad (4.8)$$

where W_0 is the field due to a simple source/sink,

$$W_0 = \log z \quad (4.9)$$

W_1 is the complex derivative of W_0 , i.e. the field due to a doublet of unit real strength, W_2 the second derivative, and so forth, so that the W_i are given by,

$$W_s = (-1)^{2s} z^{-s} \quad (4.10)$$

For the particular case of streaming motion past a stationary cylinder of radius, r_0 , the complex potential is particularly simple (Milne-Thomson 1955),

$$W = -u_0 \left(z + \frac{r_0^2}{z} \right) \quad (4.11)$$

The reader's attention is directed towards two-dimensional cross-flow past a doubly-periodic array of circular cylinders at zero angle of attack. Figure 6 shows the representation of tube banks in the Argand diagram for both in-line and staggered tube banks. For convenience, let it be assumed that the centres of the circular cylinders are located in the complex plane at $\Omega_{m,n}$ where

$$\Omega_{m,n} \equiv 2m\omega_1 + 2n\omega_2 \quad (4.12)$$

and ω_1 and ω_2 are complex numbers. For an in-line array, ω_1 is real and ω_2 imaginary, while for staggered arrays ω_1 and ω_2 are complex conjugates. The two cases are referred to as rectangular-primitive and rhombic-primitive periods, respectively. For in-line tube banks,

$$\omega_1 = \frac{s_2}{2}, \quad \omega_2 = i\frac{s_1}{2} \quad (4.13)$$

while for staggered tube banks,

$$\omega_1 = \frac{s_2}{2} + i\frac{s_1}{2}, \quad \omega_2 = \frac{s_2}{2} - i\frac{s_1}{2} \quad (4.14)$$

Let it be assumed, without loss of generality that $\phi = 0$ and $\psi = 0$ at the origin in the Argand diagram. With reference to Fig. 6, since the flow is presumed to be fully-developed, the potential difference is constant between two points located at x_c and $x_c + s_2$. Similarly the difference between values of the stream function across rows located at y_c and $y_c + s_1$ must also be constant. Symmetry demands that horizontal lines, joining circle centres, be stream-lines, and the vertical lines equipotentials.

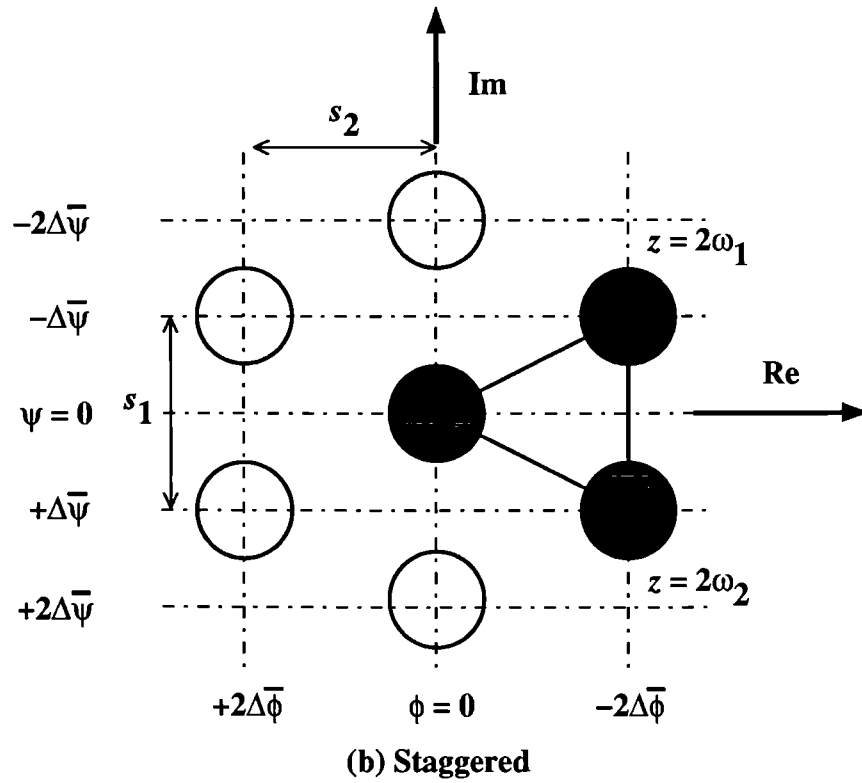
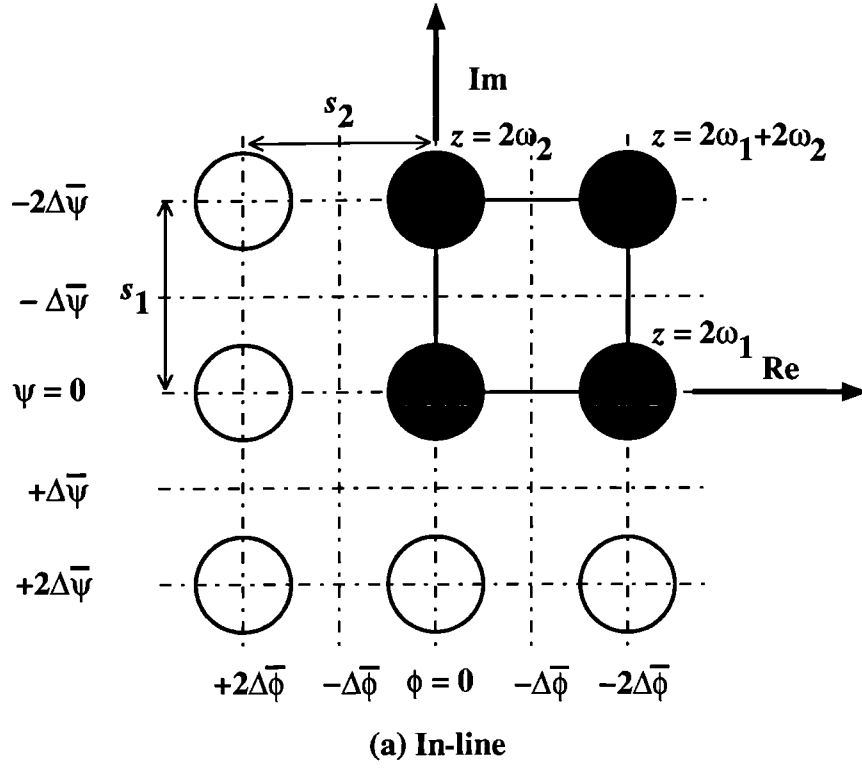


Figure 6. Representation of a tube-bank in the complex plane.

The velocity potential may be therefore be considered to be composed of a linear term, $\Delta\bar{\phi}$, giving rise to the overall potential gradient, combined with a doubly-periodic component, $\tilde{\phi}$, such that,

$$\phi = -2\Delta\bar{\phi}\frac{x_c}{s_2} + \tilde{\phi} \quad (4.15)$$

(The factor 2 is optional). The general strategy adopted is to choose a set of complex functions, $W_n(z)$, having the property that of $\text{Re}(W_n) = 0$ at $x = ns_2$, $n = \pm 1, 2, 3, \dots$. The complex potential, W , is presumed to be of the form,

$$W = -2\Delta\bar{\phi}\frac{z}{s_2} + \sum_{s=0}^{\infty} A_s W_s \quad (4.16)$$

The coefficients A_s are chosen to satisfy the additional requirement of zero normal velocity at the cylinder wall, so that Eq. 4.16 represents a general solution to the problem under consideration. The functions W_n are selected in a fashion somewhat similar to the circular harmonics defined above.

Consider first a field of sinks of unit strength, located at $\Omega_{m,n}$. The potential at some point z is given by,

$$W_0 = \log z + \sum'_{m,n} \log \left(1 - \frac{z}{\Omega_{m,n}} \right) \quad (4.17)$$

where the primed summation sign denotes the double summation over both m and n ,

$$\sum'_{m,n} \equiv \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \quad (4.18)$$

but specifically excluding the value $m = n = 0$. Using this notation, the field representing a field of doublets of unit strength, located at $\Omega_{m,n}$ may be written as,

$$\frac{dW_0}{dz} = \frac{1}{z} + \sum'_{m,n} \frac{1}{z - \Omega_{m,n}} \quad (4.19)$$

Subsequent derivatives are given by,

$$\frac{d^s W_0}{dz^s} = (-1)^{s+1} \left(\frac{1}{z^s} + \sum'_{m,n} \frac{1}{(z - \Omega_{m,n})^s} \right) \quad (4.20)$$

The odd derivatives of W are all zero along the lines of symmetry [Fig. 6] i.e. they are of the required form for use in Eq. 4.16. The even derivatives do not possess this property, but it is clear that the potential field under consideration must be odd, and therefore the even derivatives are not required. The required potential functions may thus be non-dimensionalized and written in the form,

$$W_{2s+1} = \frac{s_2^{2s+1}}{(2s)!} \frac{d^{(2s+1)} W_0}{dz^{(2s+1)}} \quad (4.21)$$

W_{2s+1} may be expanded as a Laurent (power) series yielding,

$$W_{2s+1} = z^{*-2s-1} + \sum_{n=0}^{\infty} \alpha_{2n+1} z^{*2n+1} \quad s = 0, 1, 2, \dots \quad (4.22)$$

where, the complex displacement, z^* , has been non-dimensionalized according to,

$$z^* \equiv \frac{z}{s_2} \quad (4.23)$$

These functions are closely related to Weirstrass' elliptic functions; further information is provided in Appendix II, together with details of the method of enumeration of the $\alpha_{2n+1, 2s+1}$ coefficients. The main advantage of Eq. 4.22 as compared to Eq. 4.20 is that the double summation has been reduced to a single one, greatly reducing computational work.

Defining ϕ_{2s+1} and ψ_{2s+1} by,

$$W_{2s+1} = \phi_{2s+1} + i\psi_{2s+1} \quad (4.24)$$

It is seen that;

$$\phi_{2s+1} = \rho^{-2s-1} + \sum_{n=0}^{\infty} \alpha_{2s+1, 2n+1} \rho^{2n+1} \cos(2n+1)\theta \quad (4.25)$$

where ρ and θ are the modulus and argument of z^* ,

$$z^* = \rho e^{i\theta} \quad (4.26)$$

From Eq. 4.16, the velocity potential is of the form,

$$\phi = -2\Delta \bar{\phi} z^* + \sum_{s=0}^{\infty} A_{2s+1} \phi_{2s+1} \quad (4.27)$$

(4.28)

The condition of zero radial velocity at the wall requires,

$$\left. \frac{\partial \phi_{2s+1}}{\partial \rho} \right|_{\rho=\mu} = 0 \quad (4.29)$$

where,

$$\mu = \frac{r_0}{s_2} \quad (4.30)$$

Differentiating Eq. 4.28 with respect to ρ and equating coefficients of θ , it can be shown that,

$$A_1 = -2\mu^2 \Delta \bar{\phi} + \mu^2 \sum_{s=0}^{\infty} A_{2s+1} \alpha_{2s+1,1} \quad (4.31)$$

$$A_{2n+1} = \mu^{4n+2} \sum_{s=0}^{\infty} A_{2s+1} \alpha_{2s+1,2n+1}, \quad s \geq 2 \quad (4.32)$$

A solution to this set of linear equations may be obtained by assuming,

$$A_{2n+1} = \sum_{r=0}^{\infty} A_{r,2n+1} \quad (4.33)$$

Equations 4.31 and 4.32 are satisfied by setting,

$$A_{0,1} = -2\Delta \bar{\phi} \mu^2, \quad A_{0,2n+1} = 0, \quad n \geq 2 \quad (4.34)$$

and for $r > 1$,

$$A_{r,2n+1} = \mu^{4n+2} \sum_{s=0}^{\infty} \alpha_{2s+1,2n+1} A_{r-1,2s+1} \quad (4.35)$$

The solution for ϕ is,

$$\phi = \sum_{n=0}^{\infty} A_n \left(\rho^{-2n-1} + \mu^{-2n-1} \rho^{2n+1} \right) \cos(2n+1)\theta \quad (4.36)$$

In order to satisfy the Cauchy-Reimann equations [Eq. 4.5] the complex potential, W must be of the form,

$$W = \sum_{n=0}^{\infty} B_{2n+1} \left[\left(\frac{z}{r_0} \right)^{2n+1} + \left(\frac{r_0}{z} \right)^{2n+1} \right] \quad (4.37)$$

where,

$$B_{2n+1} = \frac{A_{2n+1}}{\mu} \quad (4.38)$$

Equation 4.37 is the required solution for the complex potential. The complex velocity [Eq. 4.7] is,

$$U = - \sum_{n=0}^{\infty} \frac{2n+1}{a} B_{2n+1} \left[\left(\frac{z}{r_0} \right)^{2n+1} + \left(\frac{r_0}{z} \right)^{2n+1} \right] \quad (4.39)$$

4.3 Results

The results were generated using a FORTRAN computer program, to be found in Appendix III, using double precision complex arithmetic. Tables 1, 2, and 3 are tabulations of the B_{2n+1} coefficients [Eq. 4.37] for in-line square, rotated square, and equilateral triangle, configurations, for the pitch-to-diameter range $1.05 \leq s/d \leq 100$, assuming $\Delta\bar{\phi} = 1$.

Figure 7 shows contours of velocity potential and stream-lines for an in-line square tube bank, $s/d = 1.5$, as obtained from Eq. 4.37, using the coefficients from Table 1. Figure 8 is a similar plot for an equilateral triangular staggered tube bank, $s/d = 1.5$. Figures 9 and 10 show pressure contours for in-line square and equilateral tube banks, $s/d = 1.5$. Figure 11 is a similar plot for a compact rotated square bank, $s/d = 1.25$.

Figure 12 shows the pressure coefficient, c_p , at the cylinder wall (i.e. at $r = r_0$) as a function of the angle, ϕ , from the front leading edge of the cylinder for in-line square, rotated square, and equilateral triangular arrangements of various pitch-to-diameter ratios. The pressure coefficient is defined according to,

$$c_p \equiv 1 - \frac{P(0) - P(\phi)}{\frac{1}{2}\rho u_m^2} \quad (4.40)$$

where u_m is the bulk velocity at the minimum cross-section, i.e. at $\phi = \pi/2$.¹ Figure 13 shows the normalised tangential velocity, u_t/u_m vs. ϕ , at $r = r_0$, for square in-line tubes. The tangential velocity distributions for staggered banks (not shown) are similar.

¹ Except for compact rotated square banks with $s/d < 1.707$, where the minimum area occurs across the diagonal between two cylinders, at $\phi = 45^\circ$.

Table 1. B_{2n+1} coefficients for in-line square tube banks.

n	$s/d = 1.05$	1.10	1.20	1.25
1	-0.587098E+00	-0.571225E+00	-0.548994E+00	-0.539462E+00
3	0.106855E+00	0.807388E-01	0.527449E-01	0.437758E-01
5	-0.285692E-01	-0.140207E-01	-0.439717E-02	-0.261192E-02
7	0.126781E-01	0.600776E-02	0.228667E-02	0.156349E-02
9	-0.523465E-02	-0.178984E-02	-0.351572E-03	-0.172776E-03
11	0.209134E-02	0.514496E-03	0.818474E-04	0.423873E-04
13	-0.102459E-02	-0.221051E-03	-0.248106E-04	-0.984698E-05
15	0.447496E-03	0.640125E-04	0.407118E-05	0.147569E-05
17	-0.216539E-03	-0.278209E-04	-0.165448E-05	-0.517246E-06
19	0.102837E-03	0.904564E-05	0.252039E-06	0.618705E-07
21	-0.489362E-04	-0.366877E-05	-0.111920E-06	-0.272652E-07
23	0.243817E-04	0.133592E-05	0.180418E-07	0.309936E-08
25	-0.115915E-04	-0.501984E-06	-0.765867E-08	-0.145051E-08
27	0.589877E-05	0.200219E-06	0.138158E-08	0.173409E-09
29	-0.283964E-05	-0.710949E-07	-0.527019E-09	-0.775141E-10
31	0.145128E-05	0.301812E-07	0.108555E-09	0.102558E-10
33	-0.711936E-06	-0.104007E-07	-0.364069E-10	-0.413948E-11
35	0.362500E-06	0.456759E-08	0.858963E-11	0.620428E-12
37	-0.181301E-06	-0.156447E-08	-0.253159E-11	-0.220725E-12
39	0.917818E-07	0.694186E-09	0.679608E-12	0.377854E-13
41	-0.466706E-07	-0.240499E-09	-0.177895E-12	-0.117751E-13
43	0.235173E-07	0.106013E-09	0.536529E-13	0.230118E-14
45	-0.121094E-07	-0.375636E-10	-0.126736E-13	-0.630292E-15
47	0.608815E-08	0.162753E-10	0.422587E-14	0.139810E-15
49	-0.316120E-08	-0.593317E-11	-0.917056E-15	0.000000E+00
51	0.158909E-08	0.251234E-11	0.000000E+00	0.000000E+00
53	-0.827895E-09	-0.944414E-12	0.000000E+00	0.000000E+00
55	0.415124E-09	0.389936E-12	0.000000E+00	0.000000E+00
57	-0.214123E-09	0.000000E+00	0.000000E+00	0.000000E+00
59	0.104762E-09	0.000000E+00	0.000000E+00	0.000000E+00

Table 1 (continued).

<i>n</i>	<i>s/d</i> = 1.5	2.0	3.0	4.0
1	-0.495840E+00	-0.418097E+00	-0.306584E+00	-0.238303E+00
3	0.192961E-01	0.514655E-02	0.745457E-03	0.183336E-03
5	-0.263948E-03	-0.702128E-05	-0.396659E-07	-0.976619E-09
7	0.322275E-03	0.271509E-04	0.776816E-06	0.604488E-07
9	-0.801672E-05	-0.665822E-07	-0.741945E-10	-0.577966E-12
11	0.370929E-05	0.981671E-07	0.554759E-09	0.136590E-10
13	-0.199089E-06	-0.507929E-09	-0.111440E-12	-0.274641E-15
15	0.474841E-07	0.391010E-09	0.436406E-12	0.339977E-14
17	-0.424797E-08	-0.323752E-11	-0.139391E-15	-0.108667E-18
19	0.604550E-09	0.152044E-11	0.335090E-15	0.826070E-18
21	-0.861956E-10	-0.189552E-13	-0.159390E-18	0.000000E+00
23	0.797511E-11	0.594992E-14	0.256897E-18	0.000000E+00
25	-0.171478E-11	-0.104621E-15	0.000000E+00	0.000000E+00
27	0.109909E-12	0.232569E-16	0.000000E+00	0.000000E+00
29	-0.341008E-13	-0.556644E-18	0.000000E+00	0.000000E+00
31	0.161366E-14	0.871333E-19	0.000000E+00	0.000000E+00
33	-0.681624E-15	0.000000E+00	0.000000E+00	0.000000E+00
35	0.256195E-16	0.000000E+00	0.000000E+00	0.000000E+00
37	-0.136861E-16	0.000000E+00	0.000000E+00	0.000000E+00
39	0.437500E-18	0.000000E+00	0.000000E+00	0.000000E+00
41	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

<i>n</i>	<i>s/d</i> = 5	10	20	100
1	-0.193908E+00	-0.992207E-01	-0.499020E-01	-0.999921E-02
3	0.611046E-04	0.195416E-05	0.614265E-07	0.196935E-10
5	-0.546100E-10	-0.682210E-14	-0.837671E-18	0.000000E+00
7	0.825230E-08	0.164946E-10	0.324053E-13	0.166241E-19
9	-0.132375E-13	-0.103355E-18	-0.794730E-24	0.000000E+00
11	0.763775E-12	0.954139E-16	0.117387E-19	0.000000E+00
13	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Table 2. B_{2n+1} coefficients for rotated square banks.

n	$s/d = 1.05$	1.10	1.20	1.25
1	-0.830282E+00	-0.807835E+00	-0.776395E+00	-0.762914E+00
3	-0.151116E+00	-0.114182E+00	-0.745925E-01	-0.619083E-01
5	0.404030E-01	0.198282E-01	0.621854E-02	0.369381E-02
7	0.179295E-01	0.849626E-02	0.323384E-02	0.221111E-02
9	-0.740292E-02	-0.253122E-02	-0.497198E-03	-0.244342E-03
11	-0.295761E-02	-0.727607E-03	-0.115750E-03	-0.599447E-04
13	0.144899E-02	0.312613E-03	0.350875E-04	0.139257E-04
15	0.632856E-03	0.905273E-04	0.575752E-05	0.208695E-05
17	-0.306233E-03	-0.393448E-04	-0.233980E-05	-0.731497E-06
19	-0.145434E-03	-0.127925E-04	-0.356437E-06	-0.874981E-07
21	0.692063E-04	0.518842E-05	0.158279E-06	0.385589E-07
23	0.344809E-04	0.188928E-05	0.255150E-07	0.438316E-08
25	-0.163929E-04	-0.709914E-06	-0.108310E-07	-0.205133E-08
27	-0.834214E-05	-0.283153E-06	-0.195386E-08	-0.245238E-09
29	0.401586E-05	0.100543E-06	0.745318E-09	0.109622E-09
31	0.205243E-05	0.426826E-07	0.153520E-09	0.145038E-10
33	-0.100683E-05	-0.147088E-07	-0.514872E-10	-0.585411E-11
35	-0.512653E-06	-0.645955E-08	-0.121476E-10	-0.877418E-12
37	0.256398E-06	0.221250E-08	0.358021E-11	0.312153E-12
39	0.129799E-06	0.981729E-09	0.961113E-12	0.534366E-13
41	-0.660024E-07	-0.340117E-09	-0.251582E-12	-0.166525E-13
43	-0.332586E-07	-0.149926E-09	-0.758767E-13	-0.325437E-14
45	0.171253E-07	0.531231E-10	0.179232E-13	0.891370E-15
47	0.860997E-08	0.230168E-10	0.597629E-14	0.197730E-15
49	-0.447063E-08	-0.839078E-11	-0.129692E-14	-0.480015E-16
51	-0.224733E-08	-0.355300E-11	-0.469849E-15	-0.119799E-16
53	0.117082E-08	0.133561E-11	0.953682E-16	0.260572E-17
55	0.587075E-09	0.551455E-12	0.368938E-16	0.723971E-18
57	-0.302817E-09	-0.213629E-12	-0.712340E-17	-0.142768E-18
59	-0.148156E-09	-0.859114E-13	-0.289505E-17	-0.436591E-19

Table 2 (continued).

<i>n</i>	<i>s/d</i> = 1.5	2.0	3.0	4.0
1	-0.701223E+00	-0.591278E+00	-0.433576E+00	-0.337011E+00
3	-0.272888E-01	-0.727832E-02	-0.105424E-02	-0.259276E-03
5	0.373278E-03	0.992959E-05	0.560961E-07	0.138115E-08
7	0.455766E-03	0.383972E-04	0.109858E-05	0.854875E-07
9	-0.113374E-04	-0.941614E-07	-0.104927E-09	-0.817367E-12
11	-0.524573E-05	-0.138829E-06	-0.784548E-09	-0.193167E-10
13	0.281554E-06	0.718320E-09	0.157600E-12	0.388402E-15
15	0.671527E-07	0.552972E-09	0.617171E-12	0.480801E-14
17	-0.600754E-08	-0.457854E-11	-0.197129E-15	-0.153679E-18
19	-0.854963E-09	-0.215022E-11	-0.473889E-15	-0.116810E-17
21	0.121899E-09	0.268067E-13	0.225425E-18	0.583097E-22
23	0.112785E-10	0.841446E-14	0.366102E-18	0.000000E+00
25	-0.242506E-11	-0.147957E-15	-0.239579E-21	0.000000E+00
27	-0.155435E-12	-0.328902E-16	-0.307038E-21	0.000000E+00
29	0.482258E-13	0.786974E-18	0.000000E+00	0.000000E+00
31	0.228207E-14	0.128675E-18	0.000000E+00	0.000000E+00
33	-0.963963E-15	-0.408735E-20	0.000000E+00	0.000000E+00
35	-0.362315E-16	-0.508147E-21	0.000000E+00	0.000000E+00
37	0.193558E-16	0.000000E+00	0.000000E+00	0.000000E+00
39	0.625673E-18	0.000000E+00	0.000000E+00	0.000000E+00
41	-0.389283E-18	0.000000E+00	0.000000E+00	0.000000E+00
43	-0.116628E-19	0.000000E+00	0.000000E+00	0.000000E+00

<i>n</i>	<i>s/d</i> = 5	10	20	100
1	-0.274228E+00	-0.140319E+00	-0.705721E-01	-0.141410E-01
3	-0.864150E-04	-0.276360E-05	-0.868702E-07	-0.278509E-10
5	0.772301E-10	0.964790E-14	0.118465E-17	0.972368E-27
7	0.116705E-07	0.233268E-10	0.458281E-13	0.235101E-19
9	-0.187207E-13	-0.146166E-18	-0.112392E-23	0.000000E+00
11	-0.108014E-11	-0.134936E-15	-0.165684E-19	0.000000E+00
13	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Table 3. B_{2n+1} coefficients for equilateral triangle banks.

n	$s/d = 1.05$	1.10	1.20	1.25
1	-0.612181E+00	-0.604800E+00	-0.592130E+00	-0.585575E+00
3	-0.222637E-07	-0.111227E-07	-0.524236E-08	-0.401025E-08
5	-0.468580E-01	-0.323721E-01	-0.182422E-01	-0.140845E-01
7	0.130558E-01	0.503892E-02	0.992425E-03	0.469115E-03
9	0.215735E-08	0.539777E-09	0.103748E-09	0.569148E-10
11	0.224745E-02	0.556342E-03	0.108228E-03	0.614662E-04
13	-0.136853E-02	-0.340854E-03	-0.378821E-04	-0.139395E-04
15	-0.285706E-09	-0.301937E-10	-0.156369E-11	-0.544624E-12
17	-0.252432E-03	-0.268086E-04	-0.100643E-05	-0.326440E-06
19	0.121010E-03	0.142943E-04	0.778225E-06	0.219184E-06
21	0.425097E-10	0.220312E-11	0.329150E-13	0.643906E-14
23	0.300630E-04	0.166599E-05	0.177572E-07	0.286518E-08
25	-0.132283E-04	-0.609148E-06	-0.127146E-07	-0.263770E-08
27	-0.627966E-11	-0.161481E-12	-0.853939E-15	-0.100570E-15
29	-0.362860E-05	-0.105664E-06	-0.417407E-09	-0.388841E-10
31	0.165086E-05	0.307761E-07	0.201119E-09	0.291287E-10
33	0.924523E-12	0.115216E-13	0.229349E-16	0.174686E-17
35	0.441526E-06	0.650895E-08	0.105390E-10	0.635668E-12
37	-0.216311E-06	-0.177426E-08	-0.335984E-11	-0.326477E-12
39	-0.136474E-12	-0.800422E-15	-0.609123E-18	-0.308052E-19
41	-0.551597E-07	-0.386987E-09	-0.263733E-12	-0.108700E-13
43	0.288243E-07	0.110207E-09	0.595128E-13	0.382514E-14
45	0.202841E-13	0.546519E-16	0.158226E-19	0.535772E-21
47	0.713464E-08	0.226162E-10	0.637050E-14	0.183275E-15
49	-0.386699E-08	-0.712041E-11	-0.109786E-14	-0.462344E-16
51	-0.302880E-14	-0.370592E-17	-0.342127E-21	0.000000E+00
53	-0.945551E-09	-0.132152E-11	-0.148746E-15	-0.299862E-17
55	0.512671E-09	0.468788E-12	0.209820E-16	0.564103E-18
57	0.434584E-15	0.250834E-18	0.000000E+00	0.000000E+00
59	0.115434E-09	0.776212E-13	0.339748E-17	0.000000E+00

Table 3 (continued).

<i>n</i>	<i>s/d</i> = 1.5	2.0	3.0	4.0
1	-0.548782E+00	-0.470648E+00	-0.349666E+00	-0.273190E+00
3	-0.149609E-08	-0.378739E-09	-0.548551E-10	-0.135464E-10
5	-0.441370E-02	-0.673687E-03	-0.439408E-04	-0.611009E-05
7	0.164725E-04	0.796347E-07	0.400327E-10	0.176332E-12
9	0.596372E-11	0.250357E-12	0.313774E-14	0.137751E-15
11	0.620890E-05	0.168587E-06	0.965354E-09	0.238910E-10
13	-0.162829E-06	-0.139991E-09	-0.617807E-14	-0.484324E-17
15	-0.142286E-13	-0.988929E-16	-0.107208E-18	-0.836649E-21
17	-0.853419E-08	-0.410912E-10	-0.206567E-13	-0.909864E-16
19	0.833923E-09	0.127180E-12	0.492696E-18	0.726395E-22
21	0.298319E-16	0.337402E-19	0.346409E-23	0.000000E+00
23	0.118919E-10	0.100327E-13	0.442721E-18	0.000000E+00
25	-0.310053E-11	-0.833099E-16	0.000000E+00	0.000000E+00
27	-0.616218E-19	-0.103175E-22	0.000000E+00	0.000000E+00
29	-0.171320E-13	-0.244950E-17	0.000000E+00	0.000000E+00
31	0.962090E-14	0.447577E-19	0.000000E+00	0.000000E+00
33	0.138208E-21	0.000000E+00	0.000000E+00	0.000000E+00
35	0.266444E-16	0.000000E+00	0.000000E+00	0.000000E+00
37	-0.269414E-16	0.000000E+00	0.000000E+00	0.000000E+00
39	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
41	-0.478569E-19	0.000000E+00	0.000000E+00	0.000000E+00
43	0.726114E-19	0.000000E+00	0.000000E+00	0.000000E+00

<i>n</i>	<i>s/d</i> = 5	10	20	100
1	-0.222856E+00	-0.114432E+00	-0.576044E-01	-0.115460E-01
3	-0.452556E-11	-0.145228E-12	-0.456918E-14	-0.146532E-17
5	-0.130661E-05	-0.104831E-07	-0.824552E-10	-0.105772E-14
7	0.259125E-14	0.507575E-20	0.976852E-26	0.000000E+00
9	0.120615E-16	0.604744E-20	0.297963E-23	0.000000E+00
11	0.133928E-11	0.167895E-15	0.206808E-19	0.000000E+00
13	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

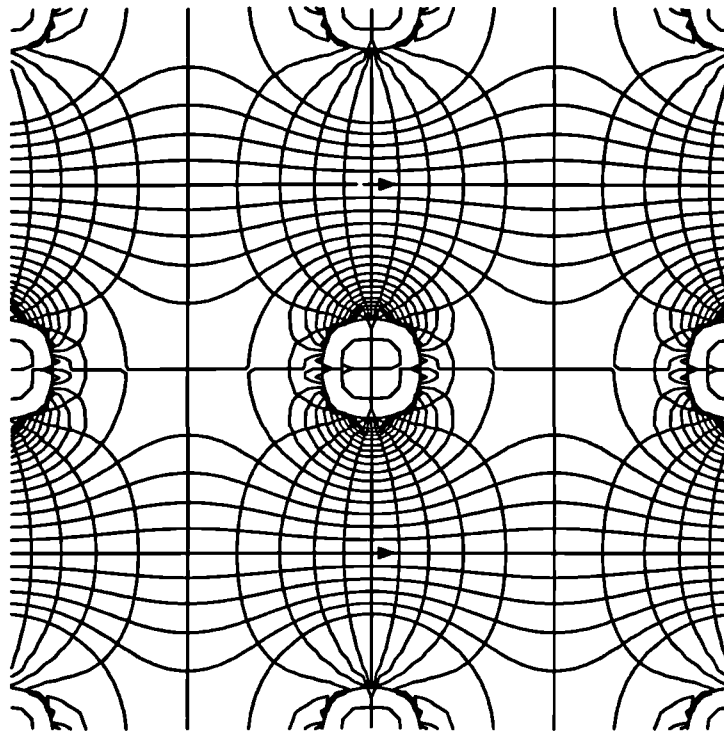


Figure 7. Complex potential for an in-line square tube bank, $s/d = 1.5$.

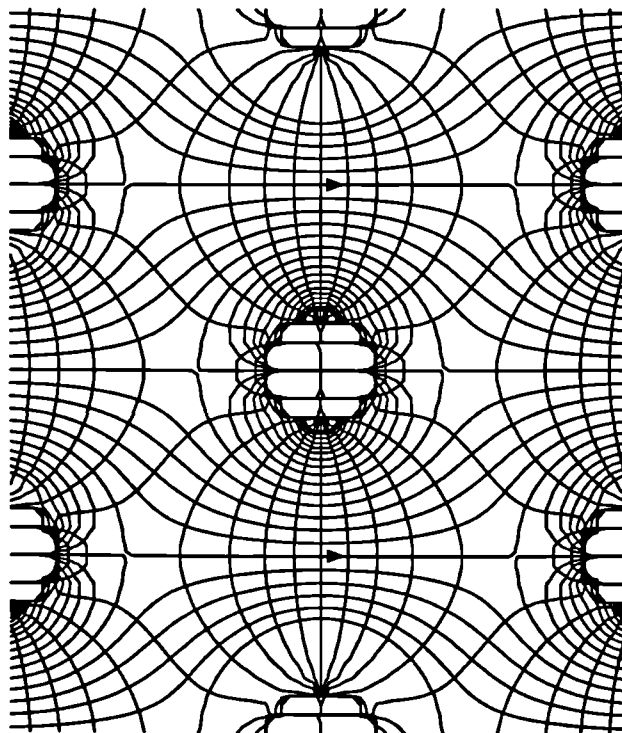


Figure 8. Complex potential for an equilateral-triangular tube bank, $s/d = 1.5$.

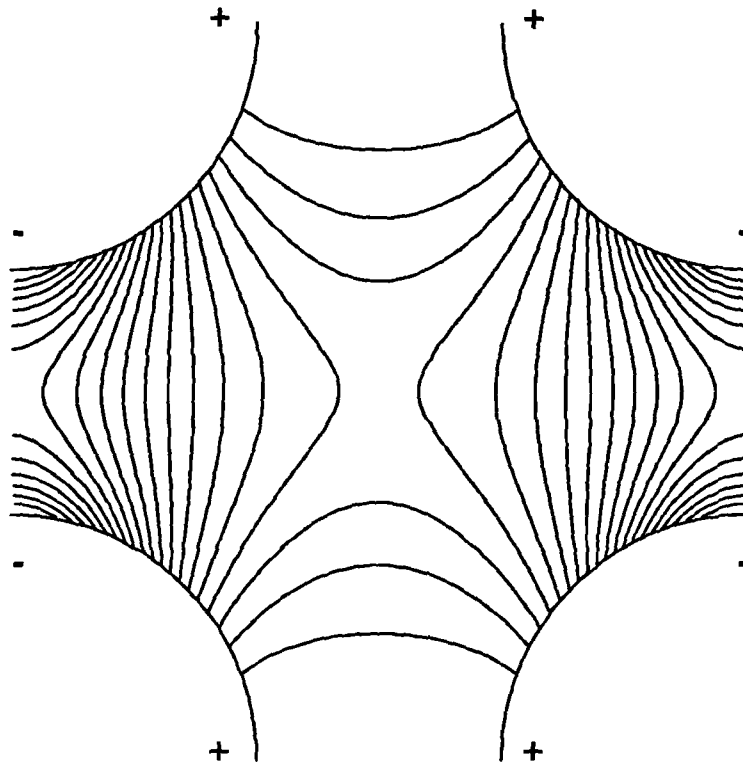


Figure 9. Pressure contours for an in-line square tube bank, $s/d = 1.5$.

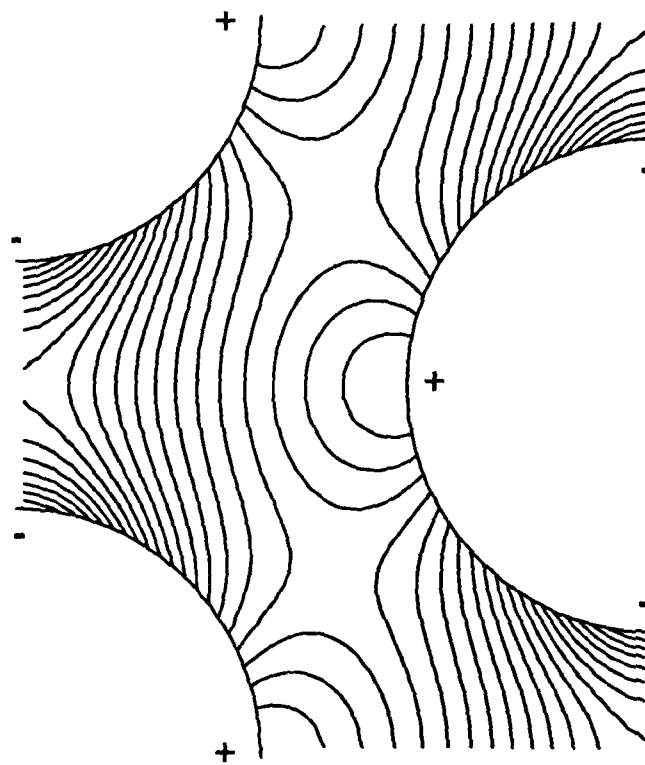


Figure 10. Pressure contours for an equilateral triangle tube bank, $s/d = 1.5$.

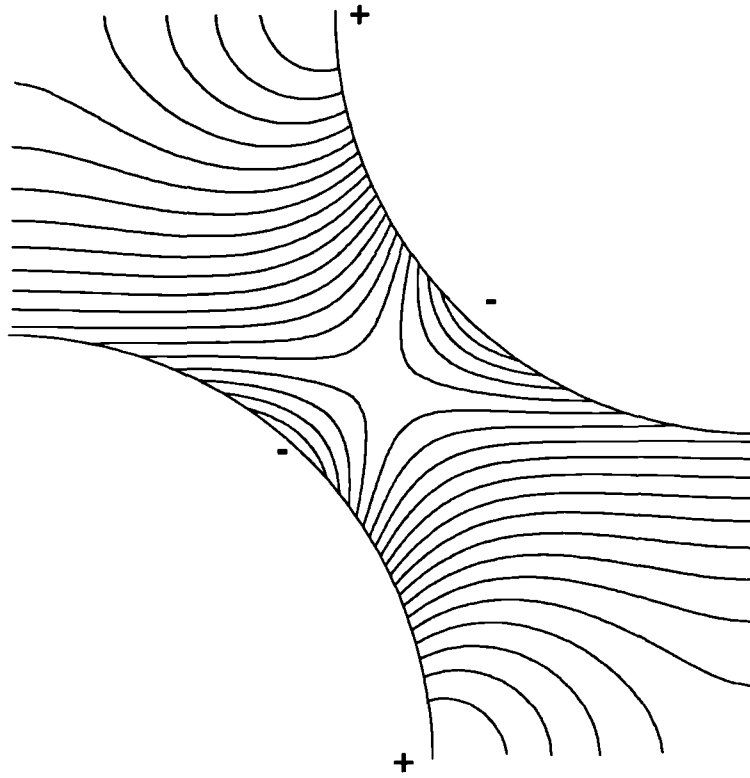


Figure 11. Pressure contours for a rotated square tube bank, $s/d = 1.25$.

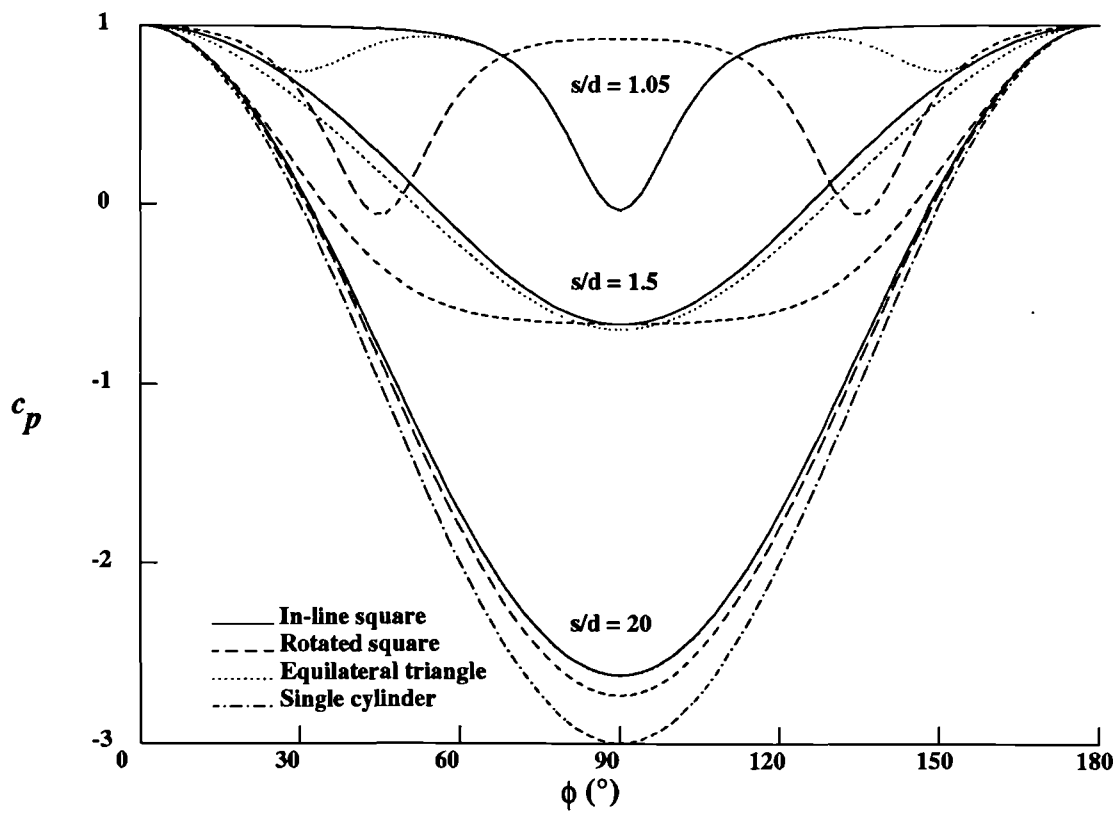


Figure 12. Pressure coefficient vs. angle, various arrangements.

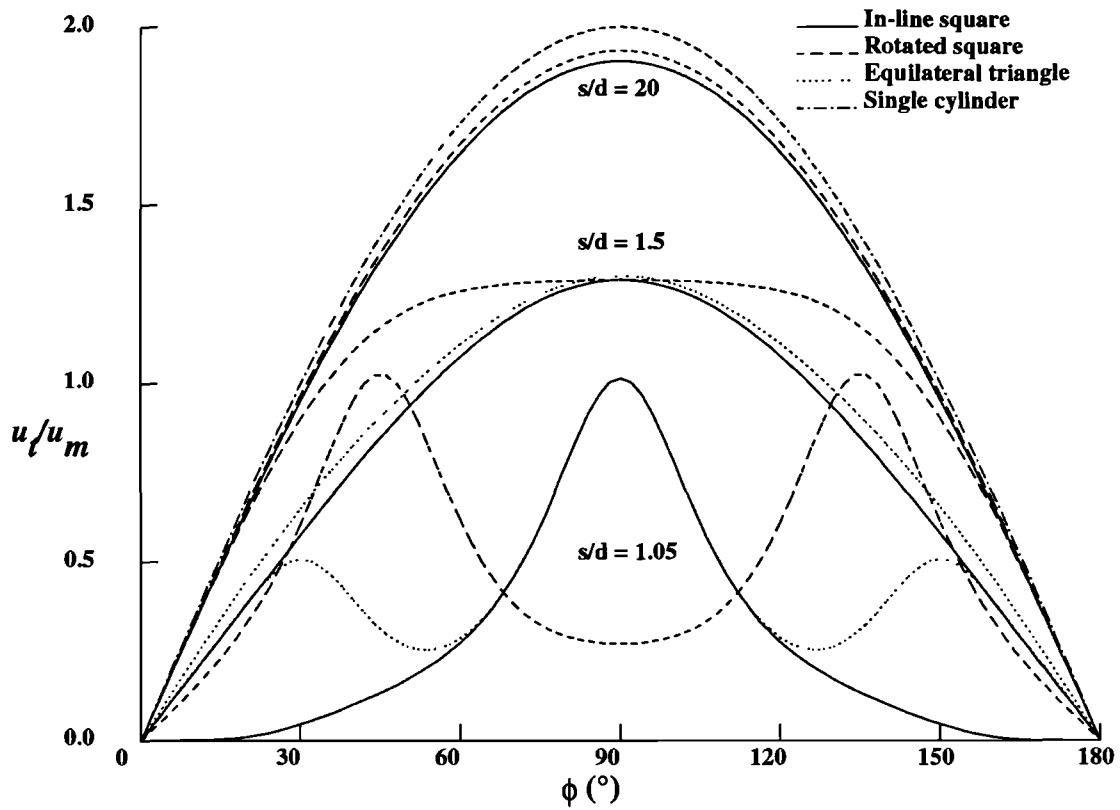


Figure 13. Tangential velocity vs. angle, various arrangements.

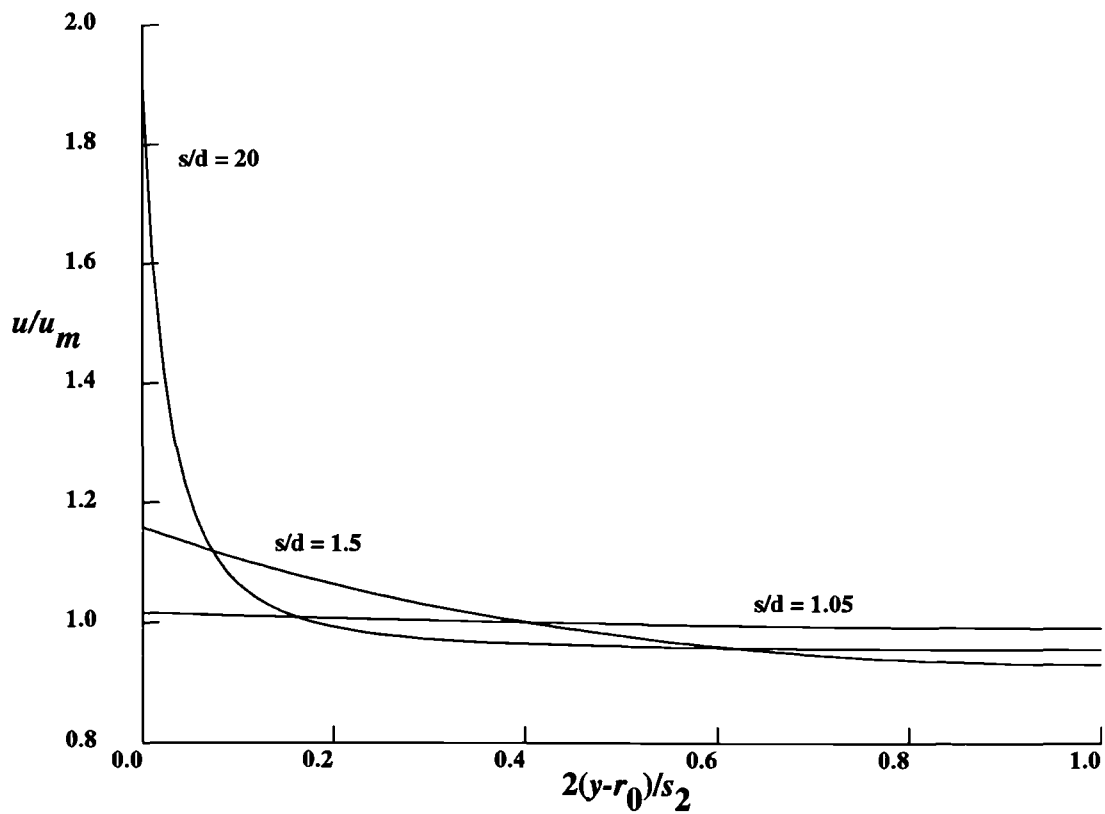


Figure 14. Velocity profiles at minimum cross-section, in-line square tube banks.

4.4 Discussion of Results

Figures 7 and 8 show values of the complex potential, for in-line and staggered geometries (stream-lines are distinguished from equipotentials by means of arrows). As might be expected, the cylinder walls are stream-lines. The reader will note that the overall gradient of the stream-function is not the same as that of the velocity potential. Velocity vectors (not shown) exhibited a maximum at the cylinder wall at $\phi = 90^\circ$, i.e. the velocity profile is convex in shape.

Figures 9 and 10 show pressure contours for moderately spaced in-line square and equilateral triangle banks. It can be seen that a global pressure minimum occurs at the cylinder wall at $\phi = 90^\circ$, and that a global maximum occurs at the leading and trailing edges ($\phi = 0^\circ$ and 180°). It can also be seen that the lines of geometric symmetry (of the bank) are lines of constant pressure gradient, and not isobars. There is a saddle-point in the pressure distribution midway between the cylinders, at $\phi = 90^\circ$, $y_c = s_1/2$, where the velocity is a local minimum in the cross-wise direction, and a maximum in the stream-wise direction. For compact banks, the velocity maximum occurs across the diagonal at $\phi = 135^\circ$, consequently for these banks, the pressure is a minimum in this region. As $s/d \rightarrow 0$ the flow and pressure distributions exhibit diagonal symmetry, as indicated in Fig. 11.

Inspection of Figs. 12 and 13 shows that for large s/d , regardless of the bank configuration, the pressure and velocity distributions are similar to those for a single cylinder; with the velocity approaching a maximum of twice the value of the bulk velocity, u_m , at $\phi = 90^\circ$, and zero at $\phi = 0^\circ$ and 180° . Conversely the pressure coefficient is a maximum of unity at $\phi = 0^\circ$ tending towards the minimum of -3 at 90° . For $s/d = 1.5$, there is substantially less variation in c_p and u_t/u_m ; for this pitch-to-diameter ratio, the maximum value of u_t is approximately $1.25u_m$ and the minimum in c_p around -0.7. For

compact tube banks, $s/d = 1.05$, there are substantial differences in the pressure and velocity distributions, for the three configurations: In in-line tube banks u_t/u_m is a maximum of slightly over unity at $\phi = 90^\circ$. In compact rotated square banks, the velocity is a maximum across the diagonal at $\phi = 45^\circ$ and 135° , thus two distinct global extrema are observed, as was discussed above. Equilateral triangular banks exhibit an intermediate behaviour, with a single global extremum at $\phi = 90^\circ$, and twin local extrema at $\phi = 45^\circ$ and 135° .

Figure 14 shows the velocity distribution at the minimum cross-section (i.e. at $\phi = 90^\circ$) for in-line tube banks. It can be seen that for low pitch-to-diameter ratios, the velocity profile is relatively constant, while at high pitch-to-diameter ratios, a large maximum occurs at the wall. It was discovered that the velocity and pressure distributions appear to be largely affected by the transverse pitch-to-diameter ratio, s_1 , and relatively unaffected s_2 . Inspection of Figs. 12 and 13 reveals that velocity and pressure distributions for equilateral triangle and rotated square banks are essentially for identical $s/d = 20$. For $s/d = 1.05$, they are again identical in the range $60^\circ \leq \theta \leq 120^\circ$, i.e. in the region around the minimum cross-section. Also for $s/d = 1.05$, it can be seen that the global extrema are the same for all three geometries.

Inspection of Tables 1 to 3 reveals that, for large pitch-to-diameter ratio, B_1 is the dominant term, and it is easy to verify that in the limit, the solution calculated according to Equation 4.37 approaches that obtained for a single cylinder [Eq. 4.11]. The coefficients tabulated in Tables 1 to 3 have been listed over a wide range of s/d , and may readily be used to compute the complex potential, and hence the velocity and pressure distributions for common arrangements of tube banks.

4.5 Discussion of Solution Procedure

The methodology employed in this thesis was, to some extent, similar that described in the paper by Knight and McMullen (1937) which contained an analysis with both even and odd harmonic functions. (The former might be used, say, to solve a heat conduction problem involving a slab containing a row of circular holes). The solution proposed by Knight and McMullen was to consider derivatives of functions consisting of sources located at $\Omega_{2m,n}$ and sinks at $\Omega_{2m+1,n}$. The even functions clearly will have zero real potential along the lines between the cylinders, i.e. they are of the desired form. Knight and McMullen incorrectly stated that the odd functions were of a form suitable for substitution in Eq. 4.16: For example the first odd function is a series of lines of dipoles with alternating polarity. It is easy to verify that for say, in-line banks, the lines of symmetry $x = x + ns_1$, $n=1,2,3,..$ correspond to lines of constant potential gradient (opposite poles attract). Thus, the potential functions upon which their solution was based could not satisfy the boundary conditions, i.e. their solution was incorrect. In addition, their method of solution was somewhat convoluted requiring the use of both Taylor and Laurent series expansions (the former can easily be avoided by treating the field as being composed of a source of sources of unit strength, at $\Omega_{m,n}$ and a set of sinks of double strength at $\Omega_{2m+1,n}$, using superposition). It must, however, be admitted that the Knight and McMullen logic did provide a foundation for the solution derived above.

The solution of Suh et. al. (1989) is also based on an inappropriate choice of harmonic functions, as may be verified by plotting iso-potentials and stream-lines based on their solution. The authors' proposed solution was based on a linear term (in z) combined with the function W_1 [Eq. 4.19] above. The authors obtained an analytical expression for W_1 , in terms of transcendental functions, eliminating the series in Eq. 4.19. Their solution is however not correct; due to the presence of the dipoles at $\Omega_{m,n}$ (m, n non-zero) the condition of zero radial velocity at $\rho = \mu$ (for all θ) is violated. A secondary set

of image doublets must be introduced to correct this problem. This necessitates the introduction of a tertiary set of doublets and so forth. The method of choosing the strength and location of the image doublets is discussed by Dalton and Helfinstine (1971) and Yamamoto (1976). This latter type of solution is only manageable for a small finite-number of cylinders. It might, however, be possible to combine the methodology of Dalton and Helfinstine (1971), with the analytical solution of Suh et. al. (1989), to obtain an alternative solution procedure to the one given in this Chapter.

4.6 Closure

The mathematical form describing potential flow in tube banks was derived. The solution for the complex potential was expressed in terms of a power series of the complex displacement. Series-coefficients were given in tables of real numbers. Flow diagrams, and pressure fields were illustrated in graphical form. Velocity and pressure coefficients were presented and analysed for common tube bank configurations, over a range of pitch-to-diameter ratios.

5. FINITE-VOLUME EQUATIONS

The basic solution procedure used in the remainder of the study is the finite-volume method. This method has been in existence for over two decades. Most of the concepts described below are well-known and hence the description of this solution procedure has been kept as brief as possible. The reader is referred to Patankar and Spalding (1972), Caretto et. al. (1972), Patankar (1980), Spalding (1980), and numerous other texts for further details. The general procedure used is a modified version of the SIMPLE (Semi-Implicit Method for Pressure-Linked Hydrodynamics Equations) algorithm (Patankar and Spalding, 1972).

5.1 Scalar Variable

The methodology involves constructing a grid made up of a manifold of cells connected together in a structured manner. Figure 15 shows a typical cell, the location of the node P is just a weighted average of the eight cell corners. Let it be postulated that the nodal value of ϕ at P , ϕ_P is related to that at its six neighbours by a set of linear algebraic equations having the form,

$$\boxed{a_W(\phi_W - \phi_P) + a_E(\phi_E - \phi_P) + a_S(\phi_S - \phi_P) + a_N(\phi_N - \phi_P) + a_L(\phi_L - \phi_P) + a_H(\phi_H - \phi_P) + a_T(\phi_T - \phi_P) + S = 0} \quad (5.1)$$

where the subscripts W, E, S, N, L, H refer respectively to the west, east, south, north, low, and high neighbours of P [see Fig. 15] and ϕ_T refers to the value of ϕ at the previous time step. Equations have the form of Eq. 5.1 are termed finite-volume or finite-domain equations, as they approximate the differential equations of motion, integrated

over the cell volume. The linking coefficients are evaluated by considering the combined influence of transience, convection, diffusion and source-related effects. Brief details are given below.

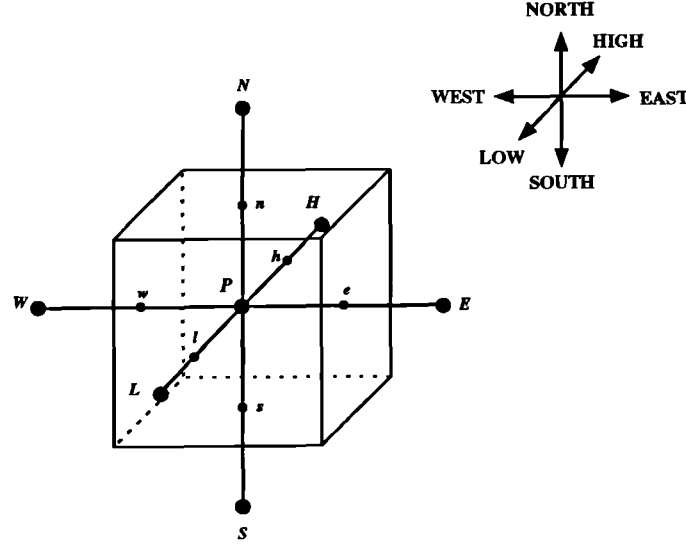


Figure 15. Cell Nomenclature.

5.1.1 Convection and Diffusion

It is convenient to combine the convection and diffusion terms by defining the total flux, $\vec{J}$, according to,

$$\vec{J} \equiv \rho \vec{u} \phi - \Gamma \vec{\nabla} \phi \quad (5.2)$$

In the absence of transient or source terms, the scalar transport equation [Eq. 3.4] reduces to,

$$\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z} = 0 \quad (5.3)$$

Integrating one obtains,

$$A_e J_e - A_w J_w + A_n J_n - A_s J_s + A_k J_k - A_l J_l = 0 \quad (5.4)$$

For the simple case of one-dimensional convection-diffusion an analytical solution is available, see Patankar (1980).

$$J_e - J_w = 0 \quad (5.5)$$

It can be shown that, under these circumstances, the finite-volume equation, Eq. 5.1, is exact if and only if,

$$a_E = \frac{m_e}{\exp(m_e/d_e) - 1} \quad (5.6)$$

$$a_W = \frac{m_w \exp(m_w/d_w)}{\exp(m_w/d_w) - 1} \quad (5.7)$$

where m_e is the mass flux across the east face given by,

$$m_e \equiv \rho u_e A_e \quad (5.8)$$

and d_e is given by,

$$d_e \equiv \frac{\Gamma_e A_e}{|\vec{PE}|} \quad (5.9)$$

m_e and d_e are sometimes referred to as convection and diffusion coefficients, respectively. m_w and d_w are defined in a similar fashion with respect to the west interface. A Peclet number, Pe , is sometimes defined by,

$$Pe \equiv \frac{m}{d} \quad (5.10)$$

An approximation to the exponential solution above, is the so-called hybrid scheme, Spalding (1972),

$$a_E = \max(0, d_e - 2|m_e|) + \max(0, -m_e) \quad (5.11)$$

$$a_W = \max(0, d_w - 2|m_w|) + \max(0, +m_w) \quad (5.12)$$

When considering the flux of ϕ in two- and three-dimensions, it is tacitly assumed that the problem may be treated as if there were three locally-independent one-dimensional fluxes, i.e.

$$\frac{dJ_x}{dx} = 0, \frac{dJ_y}{dy} = 0, \frac{dJ_z}{dz} = 0 \quad (5.13)$$

The hybrid scheme may then be extended to two- and three-dimensions using superposition.

5.1.2 Transient Term

In this study a fully-implicit scheme was employed with the time flux coefficient in Eq. 5.1 computed according to,

$$a_T = \frac{\rho_T V_T}{\delta T} \quad (5.14)$$

where ρ_T and V_T are the fluid density and cell volume (not to be confused with the source-term value below), and δT is the time-step size. Other schemes such as the explicit, and Crank-Nicholson methods are discussed in the book by Patankar (1980).

5.1.3 Linearized Source Term

The source term in the finite-volume equation [Eq. 5.1] is used both to express volumetric source terms and also boundary conditions. When used for the latter purpose, the source term typically falls between two limiting conditions.

- (a) A fixed flux i.e. a constant source, S , of ϕ , independent of the in-cell value ϕ_P (Neumann problem).
- (b) A fixed value of ϕ_P (Dirichlet problem).

The intermediate condition may be linearized according to,

$$S = C(V - \phi_P) \quad (5.15)$$

where V is a reference value, and C is a source-term coefficient. The coefficient represents the conductance of the media to ϕ , and hence a large coefficient tends to fix the in-cell value of ϕ at P to the prescribed value, V condition (b), above.

Continuity boundary conditions may be used to relate mass flux as a function of pressure (or pressure gradient), so that in this context only, ϕ_P represents the in-cell pressure, and C is proportional to the ‘permeability’ of the boundary (as given by Darcy’s Law),

$$S_P = C_P(V_P - P_P) \quad (5.16)$$

The net source of any convected scalar contains a convective contribution as a result of mass influx in addition to the prescription above,

$$C = \max(0, S_P) + C_\phi \quad (5.17)$$

Equation 5.17 is a simplified version of the so-called upwind formulation for neighbour coefficients in the finite-volume equations. In the interests of readability, only a single source term is given here and elsewhere in this text (in practice the source term will be the sum of a number terms defined by Eq. 5.15).

5.2 Continuity and Momentum

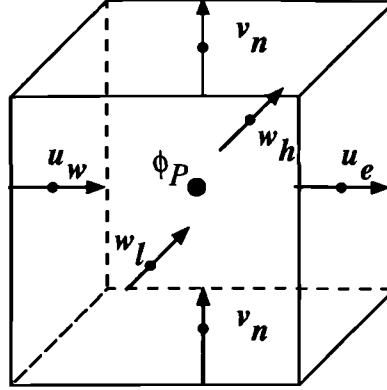


Figure 16. Location of Field Variables

The convention followed is that values of all scalar field variables, including pressure, p , are taken as being those which occur at the node P. In the case of velocity, however, stored values are presumed to be at the cell interfaces, as is shown in Fig. 16. This is the well-known ‘staggered-grid arrangement’ (Harlow and Welch, 1965) designed to obviate spurious oscillations in the pressure field. Finite-volume methods based on staggered body-fitted grids were also discussed by Parameswaran, 1985, and Karki (1986). Hsu (1981), Rhie and Chou (1983), Peric (1985) and others, have also formulated so-called co-location schemes in which velocity values are taken as Cartesian components at the cell centres. Such schemes were not considered in this study.

The finite-volume equation for continuity for an incompressible fluid is,

$$m_e - m_w + m_n - m_s + m_h - m_l - S_P = 0 \quad (5.18)$$

while the momentum equations may be written as,

$$\begin{aligned} & a_W(u_{We} - u_e) + a_E(u_{Ee} - u_e) + a_S(u_{Se} - u_e) + a_N(u_{Ne} - u_e) + \\ & a_L(u_{Le} - u_e) + a_H(u_{He} - u_e) + a_T(u_{Te} - u_e) + A_e(P_P - P_E) + S_u = 0 \end{aligned} \quad (5.19)$$

$$\begin{aligned} & a_W(v_{Wn} - v_n) + a_E(v_{En} - v_n) + a_S(v_{Sn} - v_n) + a_N(v_{Nn} - v_n) + \\ & a_L(v_{Ln} - v_n) + a_H(v_{Hn} - v_n) + a_T(v_{Tn} - v_n) + A_n(P_P - P_N) + S_v = 0 \end{aligned} \quad (5.20)$$

$$\begin{aligned} & a_W(w_{Wh} - w_h) + a_E(w_{Eh} - w_h) + a_S(w_{Sh} - w_h) + a_N(w_{Nh} - w_h) + \\ & a_L(w_{Lh} - w_h) + a_H(w_{Hh} - w_h) + a_T(w_{Th} - w_h) + A_h(P_P - P_H) + S_w = 0 \end{aligned} \quad (5.21)$$

5.3 Body-fitted Co-ordinates

The use of body-fitted co-ordinates (BFC's) is a natural extension to the finite-volume method (Malin et. al., 1985). The utilisation of source terms to account for curvilinear effects has often been used in the past in various numerical codes. This was not done in the computer code used in this study, instead it is the linking coefficients and neighbour values in the finite-volume equations [Eq. 5.1] which were modified. This type of formulation is more conservative.

The reader will recall that the stored velocity values are the tangential resolutes. In the interests of readability the compact notation u_e , v_n , w_h , will be used instead of $u_{e(tr)}$, $v_{n(tr)}$, $w_{h(tr)}$. Similarly $P-E$, $P-N$, $P-H$ refer to the tangential-displacement components, $dx^{(tc)}$, $dy^{(tc)}$, $dz^{(tc)}$, and will simply be labelled dx , dy , dz . Finally the areas A_e , A_n , A_h refer to normal components $A_{e(nc)}$, $A_{n(nc)}$, $A_{h(nc)}$. In all three cases the short-form will be used unless, unless there is an ambiguity.

In the absence of transient or source-terms, the resolute form of the transport equation [Eq. 3.21] may be written as,

$$\frac{1}{\sqrt{\alpha}} \left[\frac{\partial}{\partial x} \left(\frac{1}{\alpha_x^x} \sqrt{\alpha} J_x^{(nr)} \right) + \frac{\partial}{\partial y} \left(\frac{1}{\alpha_y^y} \sqrt{\alpha} J_y^{(nr)} \right) + \frac{\partial}{\partial z} \left(\frac{1}{\alpha_z^z} \sqrt{\alpha} J_z^{(nr)} \right) \right] = 0 \quad (5.22)$$

where α_x^x denotes the cosine of the angle between the x -tangent and x -normal directions, and α_y^y and α_z^z the y - and z -direction cosines. Recalling that the volume of the parallelepiped bounded by the tangential components dx , dy , dz is,

$$dV = \sqrt{\alpha} dx dy dz = \alpha_x^x dA_x dx = \alpha_y^y dA_y dy = \alpha_z^z dA_z dz \quad (5.23)$$

Thus Eq. 5.22 may be integrated to obtain,

$$A_e J_e^{(nr)} - A_w J_w^{(nr)} + A_n J_n^{(nr)} - A_s J_s^{(nr)} + A_h J_h^{(nr)} - A_l J_l^{(nr)} = 0 \quad (5.24)$$

The similarity between this equation and Eq. 5.4 is clear. Thus, the transport equations may be discretized as described above, subject to only two modifications.

(i) The first modification to the finite-volume method is simply that the normal resolute of $\vec{J}$ must be computed from the stored tangential values. It is still assumed that the total flux in three-dimensions may be treated as being locally one-dimensional, and the hybrid scheme applied, except that the three directions are the locally-fixed curvilinear co-ordinate directions.

For the convection flux, the procedure is as follows. The stored values are the tangential velocity resolute, u_e , v_n , w_h . Consider e.g. the east face convection flux, m_e . Values of u_e , v_n , w_h are used to construct the components of a vector, $\vec{u}$, based on the inverse of the direction cosines between the grid lines $P-E$, $P-N$, $P-H$ [see Appendix I,

Eqs. A1.57 to A1.60]. The magnitude of the convection flux normal to A_e is then obtained as the projection of these components normal to the east face, A_e .

$$m_e = \rho_e \vec{A}_e \cdot \vec{u} = \rho A_e^{(nr)} u_{e(nc)} = \rho A_{e(nc)} u_e^{(nr)} \quad (5.25)$$

All three tangential resolute contribute to the construction of the normal resolute $u_e^{(nr)}$, so that,

$$m_e = \rho A_e (F_u u_e + G_u v_n + H_u w_h) \quad (5.26)$$

and F , G , and H , are functions of geometry only¹ and are obtained directly from Eq. 3.13, using the direction cosines between $P-E$, $P-N$ and $P-H$ and the normals at e , n , h etc. Convection coefficients at the other interfaces are computed in a similar fashion.

A similar treatment is required for the diffusion flux [Eq. 5.9]: Since $\text{grad } \phi$ is a vector resolute [Eq. 3.12], it is also necessary to construct the tangential components of the diffusion flux vector and project these in the normal directions (using the same geometrical factors, F , G , and H , above). Consider for example the normal diffusion flux resolute across the east face,

$$\nabla \phi_e = F_u \frac{(\phi_E - \phi_P)}{|\vec{PE}|} + G_u \frac{(\phi_N - \phi_P)}{|\vec{PN}|} + H_u \frac{(\phi_H - \phi_P)}{|\vec{PH}|} \quad (5.27)$$

with similar terms for the other resolute. The linking coefficient, a_E , is then computed using the hybrid scheme [Eq. 5.11]. This results in additional source terms as a result of the presence of further neighbour values.

¹ The areas stored in the PHOENICS computer program are not, in fact, the normal resolute, but rather the product of the area vectors and unit displacement vectors. Thus F , G , H above differ from those defined in Rosten and Spalding (1987)

(ii) A second set of modifications is applied to the momentum-equation terms: When integrating over a finite-volume, it is necessary to refer values to locally-fixed directions (as was previously discussed in Chapter 3). Consider the east neighbour of u_e , u_{eE} say, in the u -momentum equation. In order to construct the finite-volume equation associated with a locally oblique curvilinear co-ordinate system, the components of a vector are constructed in a fashion similar to Eq. 5.26, from velocity resolute u_{eE} , v_{nE} , w_{hE} . These are then projected in direction $P-E$, i.e. parallel to u_e . This treatment of u_e and u_w accounts for grid curvature; divergence effects may be similarly treated in regard of u_{eS} , u_{eN} in the u -momentum equations¹. Curvature and divergence terms are sometimes referred to as centrifugal and Coriolis effects.

An element of bias has been introduced by the use of the triplet u_e , v_n , w_h in constructing the components of a vector at e , the use of higher order schemes e.g. involving u_e , v_n , v_s , w_l , w_h while minimising the error, involve substantial additional computational effort and storage requirements. With the current generation of computer hardware, these factors are not insignificant.

5.4 Porosities

Blockages or ‘porosities’ may be used to represent solid, or porous domains or for other reasons. These are introduced into the finite-volume equations in two different forms.

- (i) ‘Volumetric’ porosities are used to multiply all terms involving the cell volume.

¹There are minor differences in the treatment of the curvature and divergence terms not discussed here. For further details see Rosten and Spalding (1987).

- (ii) 'Area' porosities are used to multiply the East, North, and High cell areas in the finite-volume equations.

These techniques were found to be useful in a number of contexts

5.5 Solution Procedure

5.5.1 Linear Algebraic Equations.

Finite-volume equations, such as Eq. 5.1 may be solved by various iterative schemes; point-by-point, line-by-line, slab-by-slab, or whole-field solution procedures. These are discussed in Spalding (1980). Point-by-point procedures such as the Jacobi method are well-known, and will not be discussed here.

Consider, for example, a slab-by-slab procedure (a 'slab' being defined as a two-dimensional array of cells in the x-y plane). Starting from an initial guessed field, a set of slab-by-slab 'sweeps' are performed throughout the computational domain; it being assumed that the off-slab neighbour values ϕ_L , ϕ_H are known. Under these circumstances the finite-volume equations assume the explicit form,

$$\phi_P = W_P\phi_W + E_P\phi_E + N_P\phi_N + S_P\phi_S + B \quad (5.28)$$

The short-hand notation,

$$L(\phi) = 0 \quad (5.29)$$

is sometimes employed. The solution is presumed to be of the form,

$$\phi_P = N_P \phi_N + E_P \phi_E + F_P \quad (5.30)$$

A number of inner 'iterations' are required to obtain a solution to Eq. 5.30. A whole-field solver differs from a slab-by-slab procedure in that the term $H_P \phi_H$, is included in the right-hand side of Eq.5.30 (conversely in a line-by-line procedure one less term is present. All of the above solution procedures are necessarily iterative; the relative merits of each depend on the physics of the problem to be solved.

5.5.2 Relaxation

Two types of relaxation were commonly used as part of the solution procedure;

- (a) Linear relaxation.
- (b) Inertial or false-time step relaxation.

In linear relaxation the field variables are updated by adding a fraction of the previous value of ϕ to the current value.

$$\phi = (1 - \alpha)\phi^* + \alpha\phi' \quad (5.31)$$

where ϕ' and ϕ^* are the current and previous guesses for ϕ , and α is a relaxation factor.

False-time step relaxation involves adding an additional source term to the finite volume equation [Eq. 5.1] of the form,

$$S = \frac{\rho_P V_P}{\delta T_{false}} (\phi' - \phi^*) \quad (5.32)$$

5.5.3 Hydrodynamic Equations.

The SIMPLEST algorithm (Spalding, 1980, 1982) was employed in the computer code used in this study. It is a variant of the SIMPLE algorithm (Patankar and Spalding, 1972). A brief outline is given below. (A detailed discussion of SIMPLE is to be found in the book by Patankar, 1980.)

Starting from a guessed pressure field, p^* , the momentum equations [Eq. 5.19 to 5.21] are solved for u^* , v^* , and w^* . Let the true pressure, p , be given by,

$$p = p^* + p' \quad (5.33)$$

where p' are the pressure corrections. Velocity corrections, u' , v' , w' are similarly defined. The continuity equation [Eq. 5.18] is transformed into a so-called pressure correction equation, having the form of Eq. 5.1 as follows: The values of u^* , v^* , w^* will satisfy Eq. 5.18 if and only if the pressure field is correct. Otherwise residual errors, r_p , will exist,

$$r_p = m_e - m_w + m_n - m_s + m_h - m_l \quad (5.34)$$

Equation 5.34 can easily be easily written in terms of the velocity corrections, u' , v' , w' . Let it be assumed (for correction purposes) that u_e may be regarded as being a function of p_P and p_E only,

$$u_e = u_e(p_P, p_E) \quad (5.35)$$

The velocity corrections are then obtained as,

$$u'_e = \frac{\partial u_e}{\partial p_P} p'_P + \frac{\partial u_e}{\partial p_E} p'_E \quad (5.36)$$

with similar terms for v'_n and w'_h . The derivatives are obtained from the coefficients in the momentum equations [Eq. 5.19 to 5.21]. For example, by writing Eq. 5.19 in the explicit form,

$$u_e = a + b(p_P - p_E) \quad (5.37)$$

It is apparent that,

$$\frac{\partial u_e}{\partial p_P} = -\frac{\partial u_e}{\partial p_E} = b \quad (5.38)$$

The pressure correction equation is obtained by substituting Eq. 5.36 into Eq. 5.34. A linear algebraic equation having the usual form is obtained,

$$\begin{aligned} a_W(p'_W - p'_P) + a_E(p'_E - p'_P) + a_S(p'_S - p'_P) + a_N(p'_N - p'_P) + \\ a_L(p'_L - p'_P) + a_H(p'_H - p'_P) + r_P = 0 \end{aligned} \quad (5.39)$$

where the coefficients are given by, for example,

$$a_E = \rho_e A_e \frac{\partial u_e}{\partial p_E} \quad (5.40)$$

The residuals, r_p , are diminished by adding the pressure corrections p' to the pressure field: Having added the pressure and velocity corrections, the momentum equations are again solved based on these new values. The process is reiterated as necessary until the residual errors fall below an accepted criterion. Figure 17 is a simplified flow chart of the entire solution procedure. Velocity and enthalpy fields were, in general, solved using slab-wise or point-by-point solution procedures (reflecting the overall dominance of convection); the pressure correction equation was always solved using the whole-field method. Linear relaxation was applied to the pressure field, while inertial relaxation was used with respect to other variables.

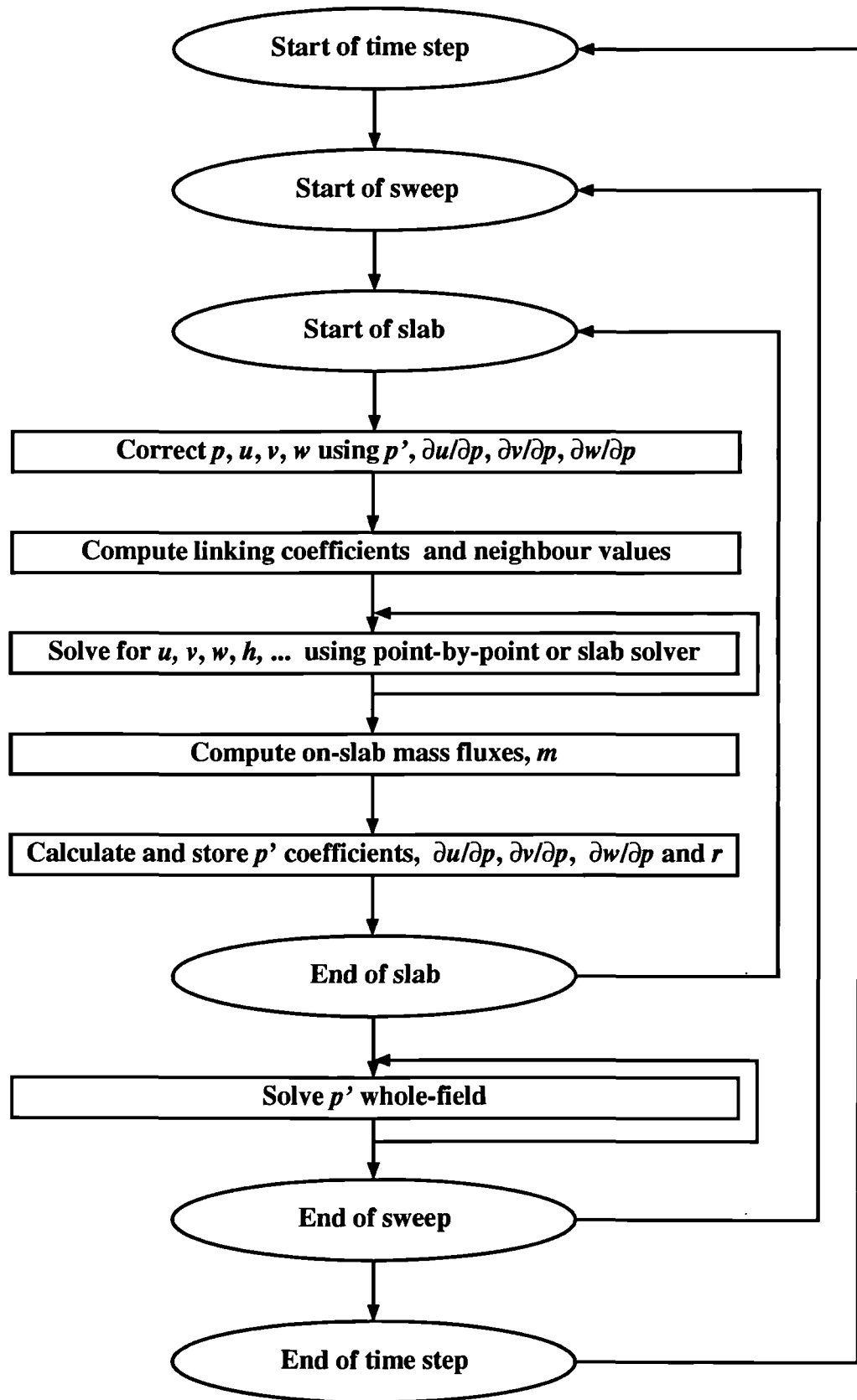


Figure 17. Simplified flow chart of solution procedure.

5.6 Computer Code and Implementation

The computer code used for the numerical study was the research-oriented finite-volume-based program PHOENICS (Parabolic, Hyperbolic or Elliptic Numerical Integration Scheme) (Spalding, 1982, Spalding, 1984, Rosten and Spalding, 1987), versions 1.3 to 1.6. The development work was done using DEC VAX computers, while the majority of the final results were generated using PHOENICS version 1.6 on an IBM RS/6000 workstation operating in a UNIX environment

5.7 Closure

The finite-volume method was outlined. The differential equations of motion are integrated over a control-volume, and approximated, to form a set of coupled linear algebraic equations. The finite-volume equations, are then solve iteratively. The procedure used is a variant of the SIMPLE algorithm.

6. GRID GENERATION

6.1 Introduction

The subject of grid generation is itself a topic of research. The reader is referred to the book on the subject by Thompson et. al. (1985) and the review paper by Thompson and Warsi (1982). The goal was neither to generate the 'ideal' grid, nor to show that the solution procedure could handle any grid type, but simply to generate a reasonable grid which could be used in modelling the flow. The general strategy involved the use of body-fitted co-ordinate (BFC) grids composed of quadrilateral or hexahedral elements, connected together in a structured fashion. In the course of the study, several distinct types of grid were considered, The use of unstructured grids, or embedded multi-grids was not considered. As a preamble to discussing the details of the grid generation scheme, some general criteria as to desirable features are listed below.

- (i) The co-ordinate lines should be as smooth as possible.
- (ii) Families of grid lines should not intersect at highly oblique angles.
- (iii) One set of grid lines should be aligned with the main flow direction, in order to minimise numerical diffusion.
- (iv) It should be possible to concentrate grid-lines in regions of interest (boundary layers, wakes etc.).
- (v) At certain boundaries the grid lines should be mutually orthogonal.

In the course of development, it was discovered that these criteria are frequently mutually exclusive, so that the choice of grid is often a compromise.

6.2 Domain Boundaries

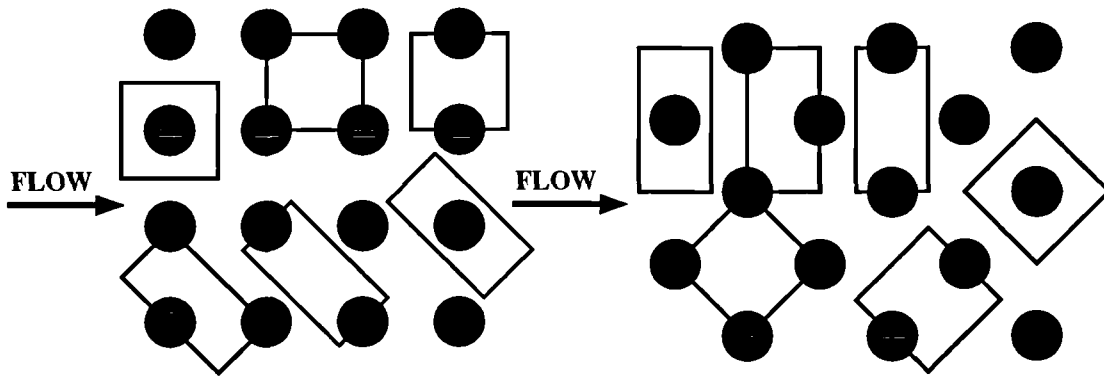


Figure 18. Some possible domain boundaries.

The first step in generating a grid for a complex multiply-connected region is to partition the region into a number of basic units or domains, within each of which a grid is constructed, i.e. to define the domain boundaries. The units (not necessarily the same), may then be connected together. For widely spaced tube banks, owing to the doubly-periodic nature of the geometry, the primitive unit can always be chosen to be either a rectangle or a rhombus, for both in-line and staggered geometries. For compact tube banks, certain restrictions apply.

Consider the region bounding precisely one cylinder, which may be located at the centre, sides or corners of the domain. Figure 18 shows a number of possible domain boundaries. The simplest grids may be generated from one quarter of the domain, exploiting the symmetry of the geometry. The attention of the reader is therefore directed

towards the region shown in Figure 19, which is the most basic domain possible for the problem under consideration.

The next step is to define the corners of the grid. An immediate problem arises with the domain shown in Fig. 19 in that there are five 'corners' ($ABCDE$), so that regardless of the choice of grid, there will always be at least one so-called 'special point', where the smoothness criterion (i) will not apply. The choice of location of this special point should be such as to minimise the impact on the flow and pressure fields.

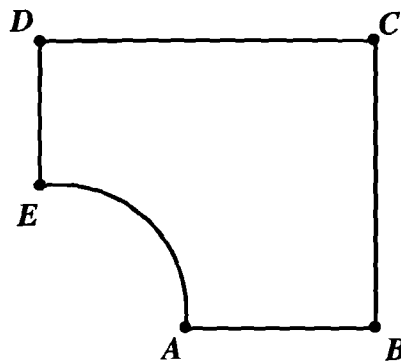


Figure 19. Primitive rectangular sub-domain.

6.3 Basic Grid Patterns

It is natural to choose the grid corners to coincide with four of the five corners of the domain shown in Fig. 19. Under these circumstances, three constructions are possible, depending on the choice of location of the special point. Figures 20, 21, and 22 show the three basic grid-types which may be constructed for the domain shown in Fig. 19,

when C , A , or B are chosen as the special point, respectively. Following current convention, these will be referred to as O-grids¹, H-grids and I-grids, respectively.

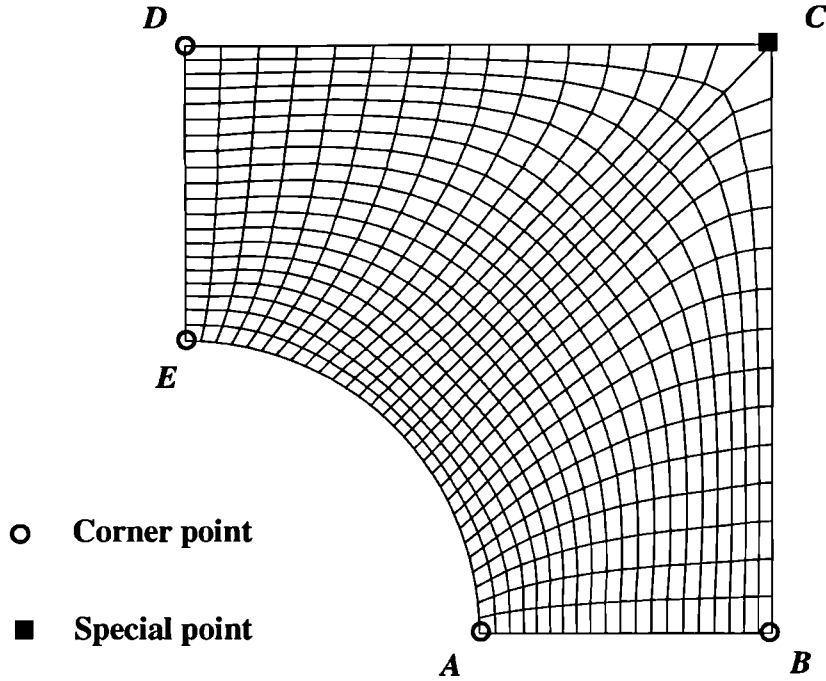


Figure 20. Basic O-grid pattern.

The O-grid [Fig. 20] is frequently employed when simulating external aerodynamic flows, where point C may be far removed from the cylinder, allowing for a smooth polar grid to be concentrated near the wall. In this regard, the O-grid appears to be superior, at least when generating results involving inviscid or boundary-layer flow. However for small pitch-to-diameter ratios (typically $1.25 \leq s/d \leq 2$ in this study), the effects of the special point, at C , may still be significant, via the influence of the entrance and exit boundary conditions. Furthermore the grid cannot be aligned with the main flow direction, so that numerical diffusion will be significant. Because of this, the O-grid was

¹The reason for the names 'O', 'H' and 'I' becomes apparent only if the grids are reflected in each axis: An O-grid may be constructed with a cylinder at the centre, an I-grid with two half-cylinders on either side, while an H-grid may be constructed with the cylinder at either the corners, or the sides (hence the name). Since in this latter case there is some ambiguity of terminology, the terms 'O', 'H' and 'I' will be used to describe grids derived from the primitive units [Figs. 20 to 22] regardless of the location of the cylinders.

not used in this study (however for parallel or counter-flow studies this grid-type would be appropriate).

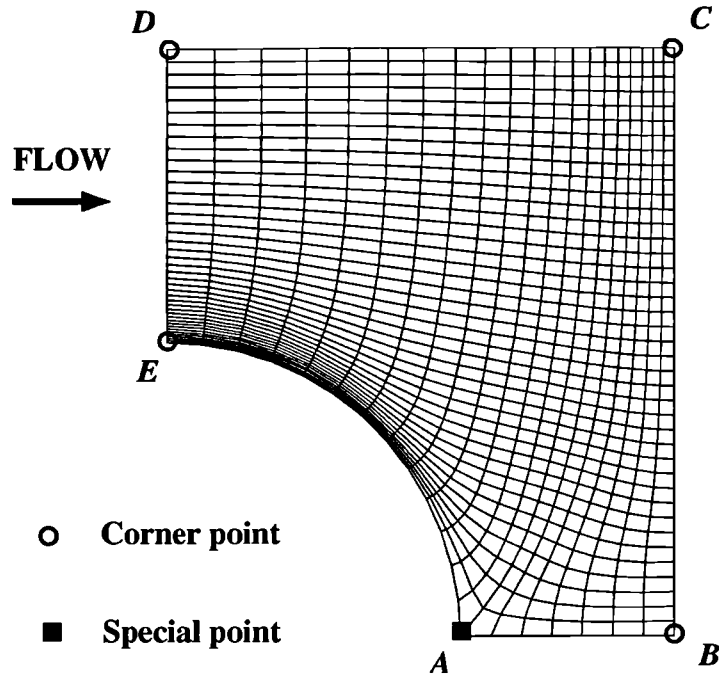


Figure 21. Basic H-grid Pattern.

The H-grid shown in Fig. 21 was used by Thom and Apelt (1961), and Antonopoulos (1979) among others. It was also the initial choice of this author. A disadvantage of this grid is that it exhibits most serious grid distortion at the cylinder wall itself, due to the proximity of the special point at A. As a result of this, it is difficult to concentrate grid cells in this zone. The grid lines are oriented along the stream-lines corresponding to streaming potential flow, so that for viscous flow, where numerical diffusion is to be minimised, the H-grid may be preferable, provided that the flow is in the indicated direction. Another advantage of the H-grid is that it is possible to construct grids with the cylinders located at either the sides or the corners of the domain of interest. Thus it is possible to construct a grid for a row of multiple in-line cylinders.

With the I-grid, Fig. 22, the singular point B is again located away from the wall. The grid may be readily concentrated around the cylinder wall. Furthermore with the flow in the indicated direction, the grid lines are roughly aligned with the bulk flow field, at least for tube banks where the pitch-to-diameter ratio is not too large. One disadvantage, discussed in Chapter 8, of I-grids (and O-grids) is that the inlet/outlet boundary conditions are more complex due to the location of the cylinders across the centre of the domain.

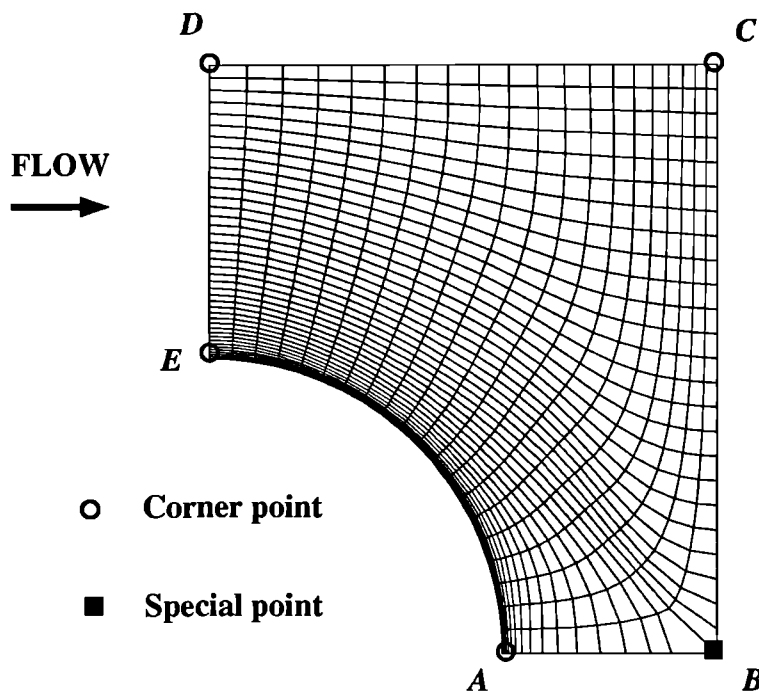


Figure 22. Basic I-grid pattern.

Both I-grids and H-grids were considered in this study, with a preponderance of the latter. Although the basic grid-types may be constructed from either rectangular or rhombic boundaries, grids used in this study were based exclusively on rectangular boundaries: This is due to the fact that grids based on rhombic domains are generally skewed with respect to the main flow.

6.4 Grid Generation Methodology

6.4.1 Differential equations

At the time of writing, the most popular method of producing grids involves the solution of elliptic partial differential equations for the Cartesian co-ordinates, x_c, y_c , of the grid nodes. This is the procedure which was selected for this study. Grids are generally classified as being orthogonal or non-orthogonal. Both were considered. The use of orthogonal grids is sometimes justified on the basis that grid lines may be generated which are parallel to iso-potentials and stream-lines for two-dimensional inviscid flow. Thus the grid-lines will be roughly aligned with the stream-lines for viscous flow. A further advantage of this type of grid is the fact that lines joining cell nodes are parallel to lines perpendicular to cell interfaces, and more importantly smooth (other than at special points). Orthogonal body-fitted grids are generated by solving Laplace's equation,

$$\nabla^2 \vec{r} = 0 \quad (6.1)$$

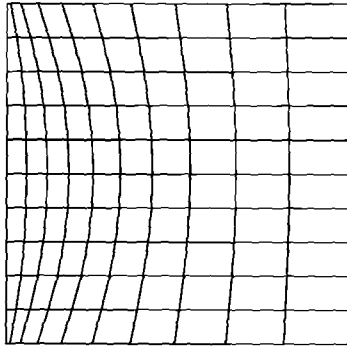
for the position $\vec{r} = (x_c, y_c)$, of the set of grid points (x, y) , $x = 1, 2, 3, \dots, nx$, $y = 1, 2, 3, \dots, ny$. Such a set of equations may readily be discretized (e.g. Spalding et. al., 1986).

Orthogonal grids have the disadvantage that the grid-lines may not readily be concentrated in desirable regions, particularly in regions of high curvature near special point(s), where the grid tends to diverge. While moving this point away from the cylinder is advantageous in that a fine grid may be generated in the region, the distortion is moved to some other place where it will influence the flow-field. As the pitch-to-diameter ratio is reduced, other grid distortion effects become apparent.

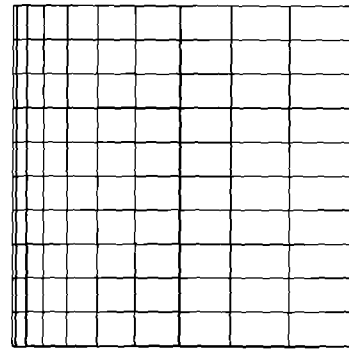
These problems are inherent to orthogonal grids. It is for this reason that non-orthogonal grids were also considered. The chosen remedy was to introduce source-terms, as necessary, into the right side of Eq. 6.1, i.e. to form Poisson's equations, having the general form,

$$\nabla^2 \vec{r} = \vec{P} = \begin{pmatrix} P \\ Q \end{pmatrix} \quad (6.2)$$

where $P(x, y)$ and $Q(x, y)$ are user-defined 'control-functions'. A number of analogies can be constructed to describe Eq. 6.2, for example heat conduction with internal sources.



(a) Laplacian based on inverted form of ordinary derivative



(b) Laplacian based on covariant derivative

Figure 23. Two types of grid.

In two-dimensions, Eq. 6.2 may be written in terms of partial derivatives as,

$$\frac{\partial}{\partial x} \left(\sqrt{g} g^{11} \frac{\partial x_c}{\partial x} \right) + \frac{\partial}{\partial x} \left(\sqrt{g} g^{12} \frac{\partial x_c}{\partial y} \right) + \frac{\partial}{\partial y} \left(\sqrt{g} g^{12} \frac{\partial x_c}{\partial x} \right) + \frac{\partial}{\partial y} \left(\sqrt{g} g^{22} \frac{\partial x_c}{\partial y} \right) = \sqrt{g} P \quad (6.3)$$

$$\frac{\partial}{\partial x} \left(\sqrt{g} g^{11} \frac{\partial y_c}{\partial x} \right) + \frac{\partial}{\partial x} \left(\sqrt{g} g^{12} \frac{\partial y_c}{\partial y} \right) + \frac{\partial}{\partial y} \left(\sqrt{g} g^{12} \frac{\partial y_c}{\partial x} \right) + \frac{\partial}{\partial y} \left(\sqrt{g} g^{22} \frac{\partial y_c}{\partial y} \right) = \sqrt{g} Q \quad (6.4)$$

g^{11} , g^{22} , and g^{12} are the components of the contravariant (inverse) metric-tensor and $\sqrt{g}$ is the Jacobian of the transformation. It is important to realise that the Laplacian operator used above is the divergence of the covariant derivative [see Appendix I]. The author's rationale is based on the supposition, that if P and Q are chosen to be zero, then Eq. 6.2 should reduce the orthogonal case [Eq. 6.1].

Equations of the form,

$$g^{11}\frac{\partial^2 x_c}{\partial x^2} + 2g^{12}\frac{\partial^2 x_c}{\partial x \partial y} + g^{22}\frac{\partial^2 x_c}{\partial y^2} = 0 \quad (6.5)$$

$$g^{11}\frac{\partial^2 y_c}{\partial x^2} + 2g^{12}\frac{\partial^2 y_c}{\partial x \partial y} + g^{22}\frac{\partial^2 y_c}{\partial y^2} = 0 \quad (6.6)$$

have often been employed by a number of previous workers (e.g. see Thompson et. al., 1974). The act of moving the metric coefficients outside the derivative affects the form of grid: For example, when the boundary-point distribution around a rectangular domain is non-uniform, the grid computed in the interior according to Eqs. 6.5 and 6.6 is of the form shown in Fig. 23a, (see also Thompson et. al., 1985). Figure 23b, shows the grid which results from the application of Eq. 6.3 and 6.4, in the absence of source terms P and Q .

If (as often is the case) it is desired that the grid be orthogonal at one or more domain boundaries, it is necessary to prescribe appropriate source terms. The method of deciding upon these source terms is complex. On the other hand, the author's formulation [Eq. 6.3 and 6.4] possesses the quality of being nearly orthogonal over the majority of the grid, other than in the immediate vicinity of regions where control-functions are applied.

6.4.2 Discretization procedure

Equations 6.3 and 6.4 may readily be discretized in a conservative fashion, and converted into a pair of linear algebraic equations having the form $L(x_c, y_c) = 0$,

$$a_W(\phi_W - \phi_P) + a_E(\phi_E - \phi_P) + a_S(\phi_S - \phi_P) + a_N(\phi_N - \phi_P) + S = 0 \quad (6.7)$$

where $\phi = x_c$ or y_c . The linear coefficients are given by,

$$a_E = \left(\sqrt{g} g^{11} \right)_e, \quad a_N = \left(\sqrt{g} g^{22} \right)_n \quad (6.8)$$

a_S and a_W are identical to a_N and a_E at the south and west cells. The x_c source term is given by,

$$\begin{aligned} S = & \left(\sqrt{g} g^{12} \right)_e (\phi_{ne} - \phi_{se}) - \left(\sqrt{g} g^{12} \right)_w (\phi_{nw} - \phi_{sw}) \\ & + \left(\sqrt{g} g^{12} \right)_n (\phi_{ne} - \phi_{nw}) - \left(\sqrt{g} g^{12} \right)_e (\phi_{se} - \phi_{sw}) - \left(\sqrt{g} P \right)_P \end{aligned} \quad (6.9)$$

The term for y_c is identical but with Q_P replacing P_P . The inverse-metric-tensor components are computed as follows;

$$g^{11} = \frac{g_{22}}{g}, \quad g^{22} = \frac{g_{11}}{g}, \quad g^{12} = -\frac{g_{12}}{g} \quad (6.10)$$

with,

$$g = \det(g_{ij}) = g_{11}g_{22} - g_{12}^2 \quad (6.11)$$

Interfacial values of metrics were computed by interpolating values of (x_c, y_c) as recommended by Thompson et. al. (1985), for example,

$$(g_{11})_e = (x_{cE} - x_{cP})^2 + (y_{cE} - y_{cP})^2 \quad (6.12)$$

$$(g_{22})_e = \frac{1}{16}(x_{cN} + x_{cNE} - x_{cS} - x_{cSE})^2 + \frac{1}{16}(y_{cN} + y_{cNE} - y_{cS} - y_{cSE})^2 \quad (6.13)$$

$$(g_{12})_e = \frac{1}{4}(x_{cE} - x_{cP})(x_{cN} + x_{cNE} + x_{cS} + x_{cSE}) + \frac{1}{4}(y_{cE} - y_{cP})(y_{cN} + y_{cNE} + y_{cS} + y_{cSE}) \quad (6.14)$$

Similar relations may easily be derived for $(g_{11})_n$, $(g_{22})_n$, and $(g_{12})_n$ and also $(g_{11})_p$, $(g_{12})_p$, and $(g_{22})_p$ required to compute g_P in Eq. 6.9. Although the practice of Thompson et. al. (1985) of averaging displacements, rather than metrics (Spalding et. al., 1986) was followed, this was not found to strongly affect the results. This is surprising; Thompson et. al. stress as a basic rule, that the former procedure should always be followed.

It was found that e.g. when $(g_{11})_s$ at $y = 2$ was based on NE , SE , NW , SW , the grid near the special point was highly compressed due to the concavity of the grid-lines. Better (though less orthogonal) results were obtained by basing these values on those at NE , NW , E , and W .

6.4.3 Boundary and initial conditions

Two types of boundary condition were considered namely,

- (a) Prescribed values (Dirichlet problem).
- (b) Prescribed boundary gradients (Neumann problem).

In this context the latter are always zero. In the absence of source-terms, P and Q , it is impossible to prescribe both boundary values and gradients simultaneously.

The method of prescribing the boundary conditions was found to strongly affect the quality of the grid. In particular it is to be noted that for orthogonal grids, ($P = Q = 0$), if the problem is posed as a Dirichlet problem, the correct solution ($g_{12} = 0$) may not be forthcoming due to over-prescription. Therefore, a mixture of fixed value and zero-slope conditions were almost always applied.

When applying zero-slope boundary conditions, various formulae were applied; (i) along straight edges e.g. at the east boundary ($x = 1$), y_{cP} was fixed to $1.5y_{cE} - 0.5y_{cEE}$ (ii) Near to cylinder walls this did not produce good results and so boundary points were extrapolated from the two interior node points assuming all three points lie on a circular arc, e.g. $y_{cP} = (x_{cE}^2 + y_{cE}^2)^{1/2}$. A linear combination of these two conditions were applied along any given boundary. Boundary points were also allowed to slide around the circumference of cylinders (except near to the special point) in some cases. In nearly all cases linear relaxation was required.

When Neumann boundary conditions are applied along more than one side, the solution is not unique, i.e. the initial conditions affect the final form of the grid: Cells were in-

initially concentrated near to solid walls using a geometric series for the mesh. (Grid-points were often concentrated along several boundaries in order to produce a well-conditioned grid)

The grid generation procedure was as follows.

- (i) Transfinite interpolation was used to generate an initial grid.
- (ii) A few iterations of the elliptic solver were applied to the interior cell corner nodes.
- (iii) Zero-flux boundary conditions were then imposed on prescribed boundaries.
- (iv) Steps (i) - (iii) were then reiterated a number of times.

A pre-existing solver was initially used to generate most of the orthogonal grids. For non-orthogonal grids, a FORTRAN utility, to be found in Appendix III, was written together with a set graphics subroutines to allow interactive display of the grid, as the solver iterated. The latter proved to be invaluable for monitoring the grid generation process. (For the case $P = Q = 0$, the two programs generated reasonably similar results.) Initial fields were generated using transfinite interpolation. The pair of equations having the form of Eq. 6.9, were then solved using a line-solver based on the tri-diagonal matrix algorithm (in hindsight a point-by-point procedure might well have been used). Several iterations were conducted in both the x and y directions. Some source term relaxation was required. Figure 24 is a flow chart of the grid generation sequence.

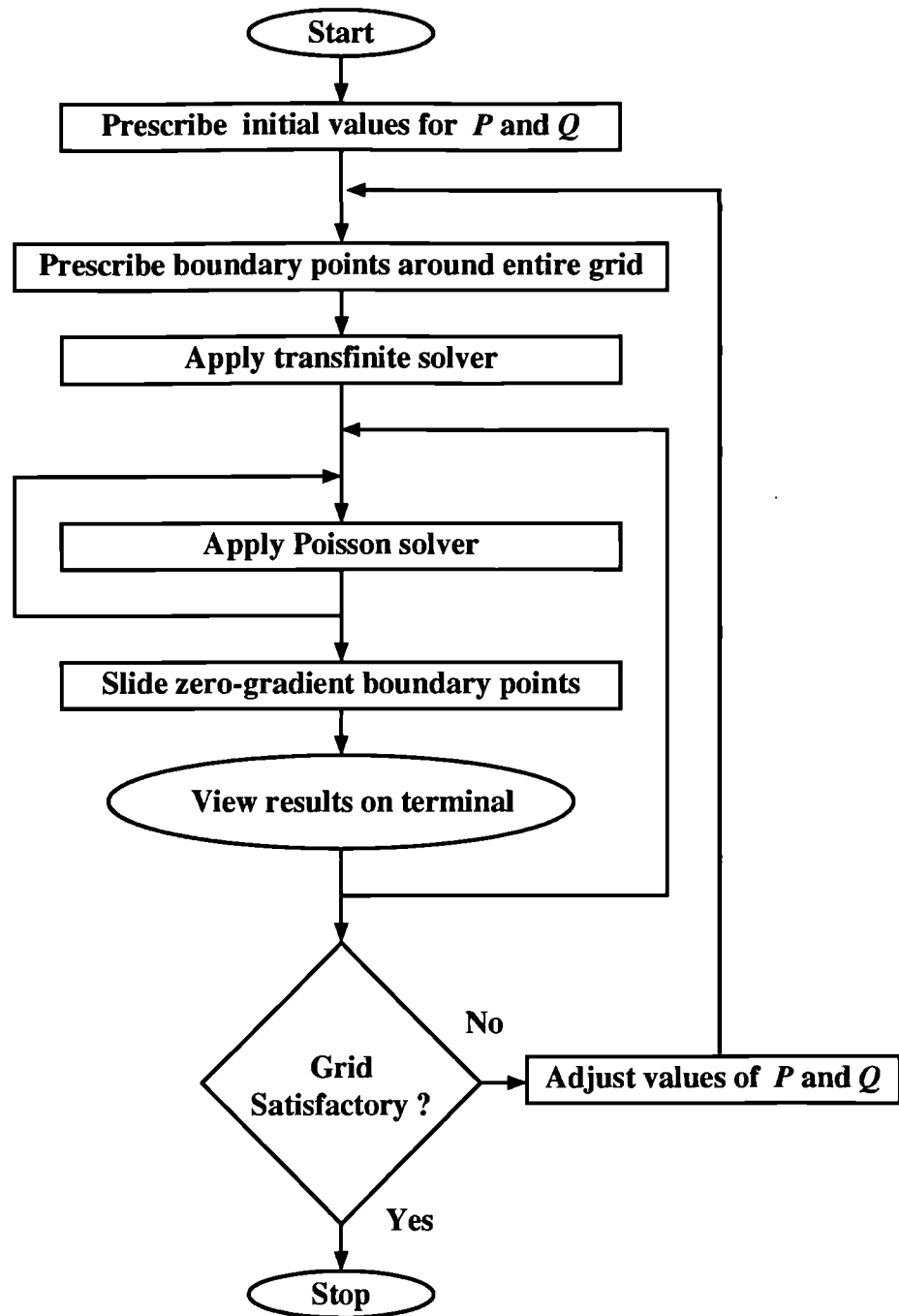


Figure 24. Flow chart of grid generation procedure.

6.4.4 Choice of control-functions

Values of P and Q were prescribed as follows (Thompson et. al., 1985). A line-attraction to the boundary line $x = x_0$, may be obtained according to,

$$P(x, y) = -a \operatorname{sign}(x - x_0) e^{-c|x - x_0|} \quad (6.15)$$

and a point-attraction to the boundary point (x_0, y_0) by

$$P(x, y) = -b \operatorname{sign}(x - x_0) e^{-d[(x - x_0)^2 + (y - y_0)^2]^{\frac{1}{2}}} \quad (6.16)$$

with similar terms for Q . Point- and line-attractions were used at the special point, as illustrated below, the constants a , b , c , d being adjusted interactively until the desired effect was achieved.

One disadvantage of the control-functions defined by Eq. 6.15 and 6.16 is that, except for very large values of the exponents c and d , there is a tendency for grid lines to be pulled away from the opposite boundary to that on which the point-of-interest lies (especially with Neumann boundary conditions). The author discovered that that this could be minimised by applying an equal and opposite control-function at the appropriate (imaginary) mirror-point. The application of both point- and line-attractions proved to be of use: For most staggered grids, the strategy was to apply attractions to all the boundary points lying on the circular arc corresponding to the cylinder wall. For many grids it was neither possible nor necessary to obtain fully converged solutions with the desired features; in these cases only a few iterations were performed. Some other types of control functions, such as sources (based on the potential flow theory) were also considered.

Figure 25 shows the effect of the use of control-functions on an H-grid. It can be seen that the divergence of the grid near to the special point has been decreased by the use of attractions at this point.

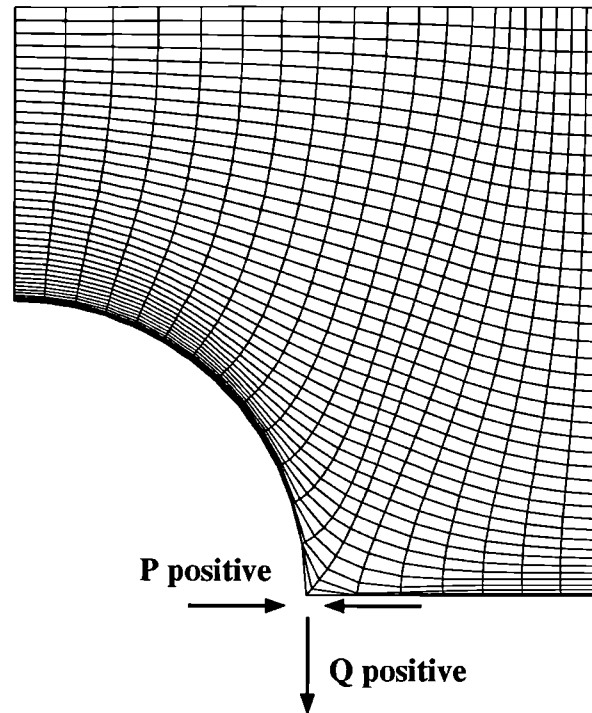


Figure 25. Non-orthogonal H-grid.

6.5 Grids used for In-line and Staggered Arrays

For steady flow in both in-line and staggered tube banks, symmetry is such that only one-half of the domain(s) shown in Fig. 18 (i.e. twice the primitive quarter-domain) need be considered. For unsteady flow, however, the entire domain must be considered.

6.5.1 In-line geometry

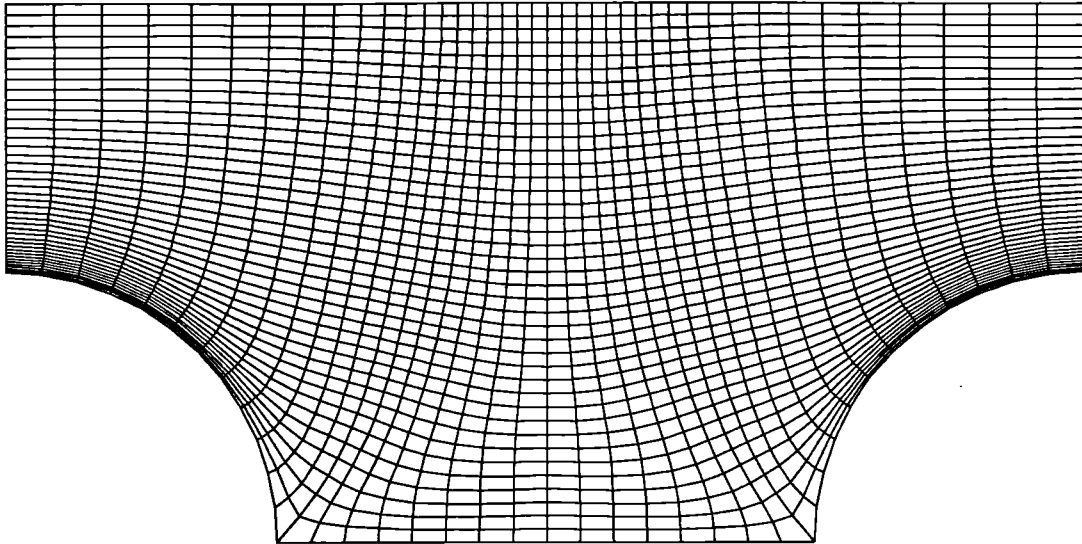


Figure 26. In-line H-grid with cylinders at corners.

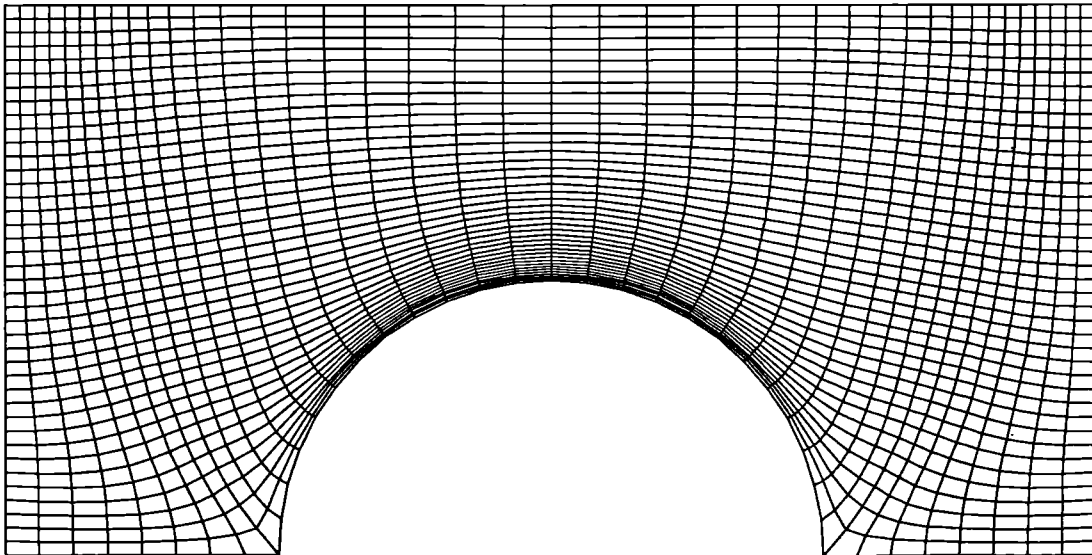


Figure 27. In-line H-grid with cylinder at side.

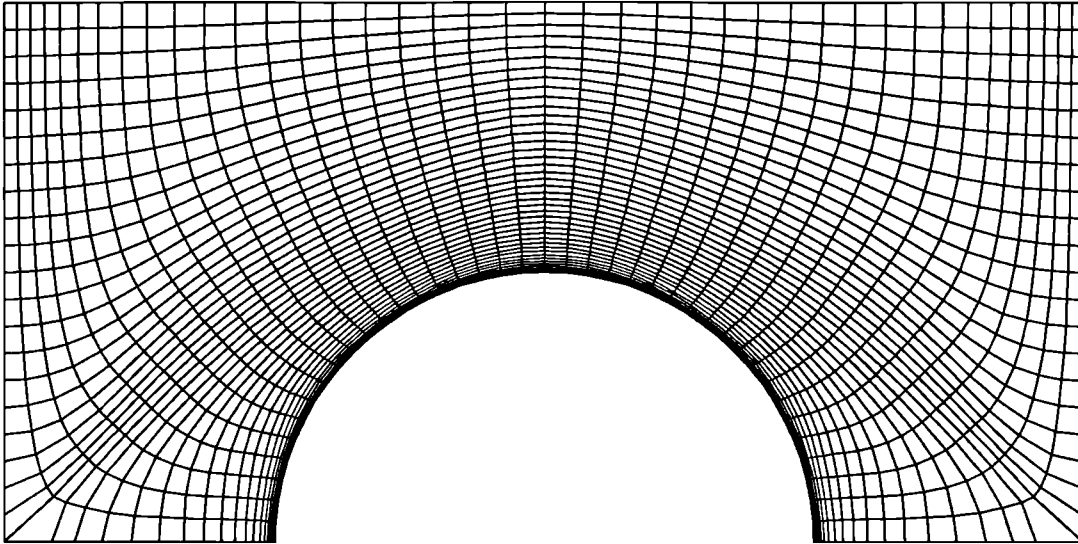


Figure 28. In-line I-grid.

Most of the grids ultimately used for studying in-line banks were of the orthogonal variety (for reasons discussed in Chapter 8). Figures 26 to 28 show in-line H- and I-grids, respectively. The H-grids are shown with quarter-cylinders located at the corners of the domain [Fig. 26] or with a $\frac{1}{2}$ -cylinder laterally placed [Fig. 27]. With the I-grid [Fig. 28] only the latter configuration is possible.

6.5.2 Staggered geometry

For widely-spaced staggered geometry, the same centrally located I- and H-grids used for in-line flow may be again be used. Two additional configurations were also considered:

An H-type grid may be also be constructed with the half-cylinders located at the diagonal corners of the domain. This may be constructed from the basic pattern, or in the

event that the tube bank is compact, the requirement that line $B-C$ [Fig. 21] be a symmetry line may be relaxed. (Under these circumstances the domain boundaries should be considered to be twice the quarter region illustrated in Fig. 19.) Figure 29 shows this type of grid, which will be referred to as a 'diagonal H-grid'. These grids were always non-orthogonal.

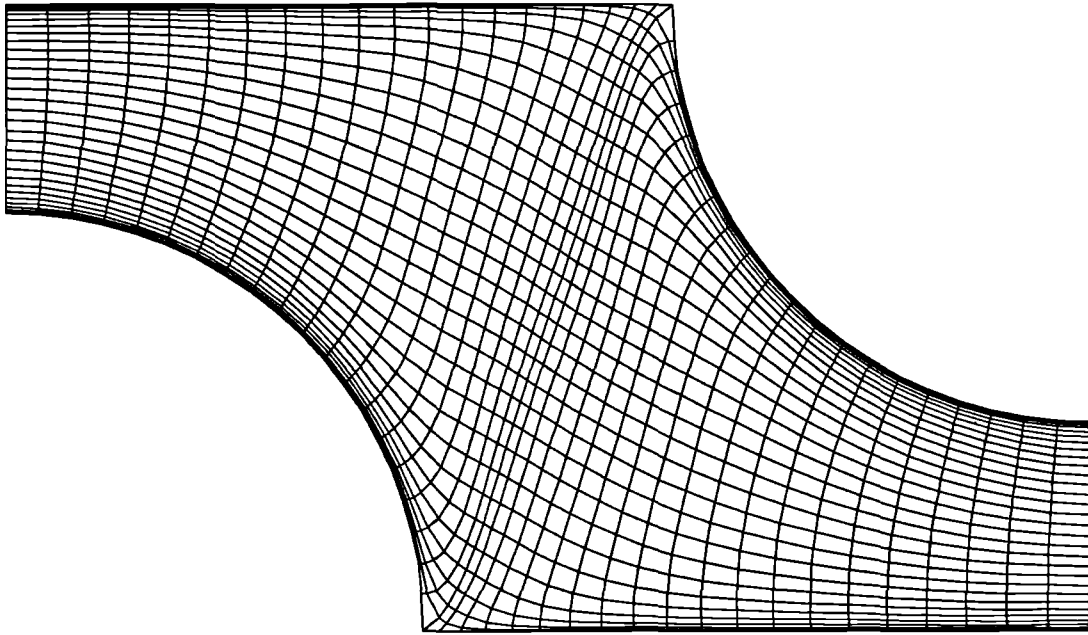


Figure 29. Non-orthogonal diagonal H-grid.

Another possible arrangement, useful for compact rotated square banks, known as the 'diagonal I-grid' is shown in Fig. 30. Note that, for the case shown, with the same number of cells in each co-ordinate direction the grid-pattern can be also be generated as an O-grid, with rhombic domain boundaries. Rhombic-based H- and I-grids were not considered, as these are skewed with respect to the main flow direction, inducing false diffusion.

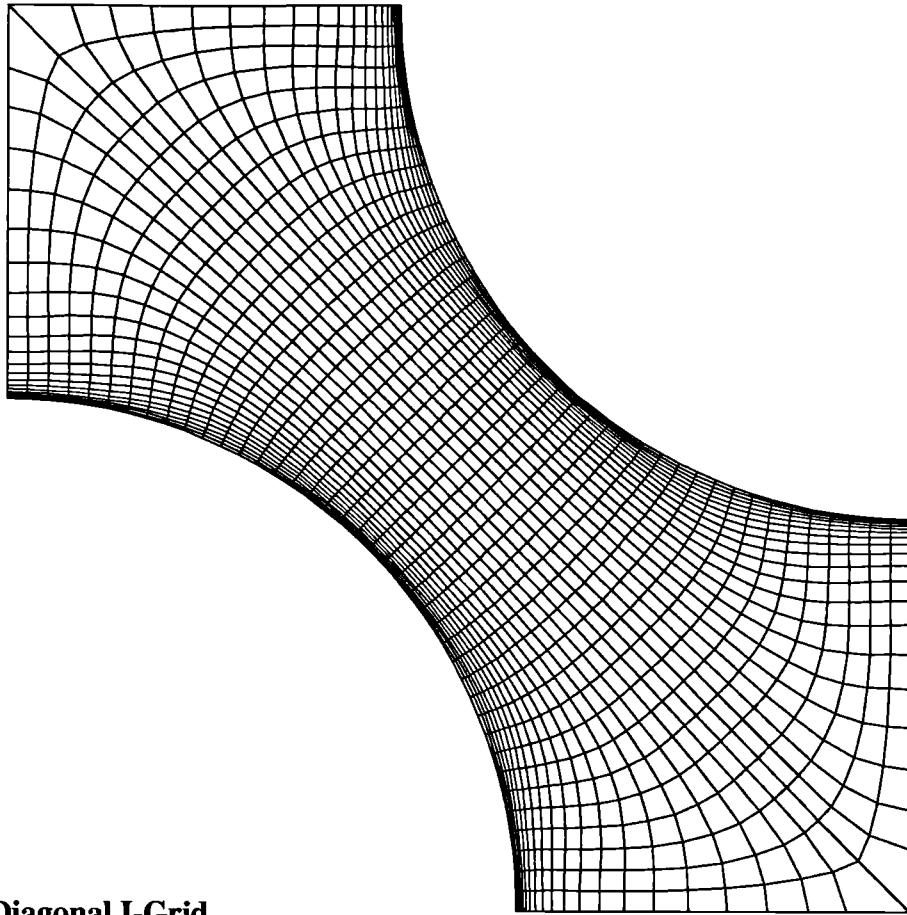


Figure 30. Diagonal I-Grid.

These basic types provide a reasonable basis for examining the flow regime. The relative merits of these grids are discussed further in Chapter 8, and elsewhere, in conjunction with the results of the flow-field simulations.

6.6 Other Grid-types

There is of course no *a priori* reason to locate the special point at any of the corners of the domain. For example, if the special point is located somewhere between *A* and *B* [Fig. 31], a grid possessing features in common with both H-grid and O-grid types is obtained. This is what Antonopoulos (1979) referred to as the “curvilinear-polar” grid,

here it will be referred to as a hybrid grid. This is a very simple example of a so-called multi-block grid¹.

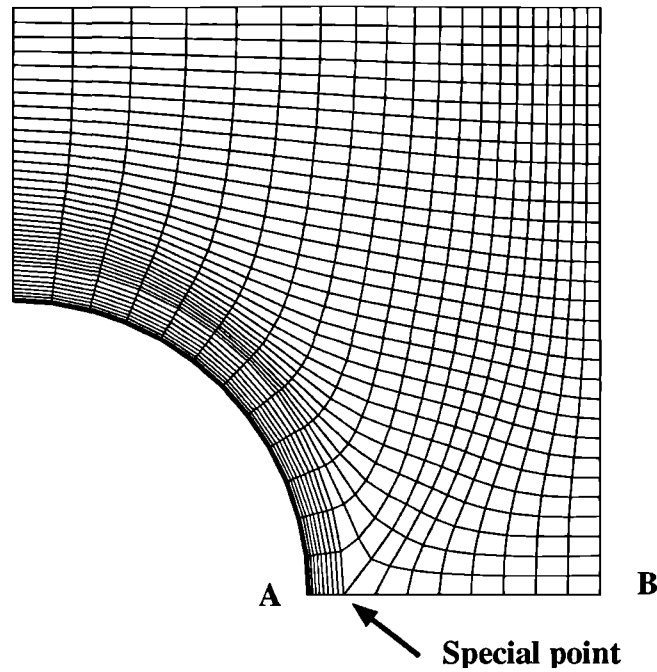


Figure 31. Hybrid grid.

Finally, the readers attention is directed to multiply-connected regions (containing several cylinders). It is always possible to construct a grid using the basic sub-blocks of H-grids and O-grids wound around the cylinders in a 'zig-zag' manner: The use of such a grid presents no additional mathematical difficulties, however there are two significant defects:

- (i) The grid directions cannot be made to correspond to the flow direction over the entire domain, so that numerical diffusion is not minimised (this is especially true for staggered geometry).

¹ The apparently missing cells are not shown, but are present. These are blocked using porosities during the flow-field solution procedure.

- (ii) In a structured computer code, such as the one used in this study, the flow solver normally proceeds in a regular fashion. Not only are additional links required across the various sub-domains (i.e. storage allocation for velocity values and additional neighbour coefficients and values for all field values), but also the solution algorithm may adversely affected by the grid not being aligned with the main flow direction.

Thus for essentially pragmatic reasons, yet another grid-strategy was devised. The final type of grid considered in this study involved construction of primitive units where, not only was the grid as a whole structured, but also all of the units themselves connected together as a regular array. Figure 32 shows this type of grid. Regardless of the type of tube bank, the grid is topologically Cartesian (the corners of the grid are at *O*, *B*, *C* and *D* in Fig. 32). The major difference between this and previous grids, is that the grid cells have been driven through the cylinder itself. (This region is later blocked using porosities.) The initially rectangular cylinder is then transformed to a circular cross-section. The effect of this transformation is to create a special point within the interior of the grid on the cylinder wall. Although there is significant distortion in the solid-region, where two adjacent sides are almost co-linear, in the fluid region the cell normals and tangents are always within 45° of each other. The grid lines may be readily concentrated in various regions (wake, boundary layer, free-stream etc.) independently.

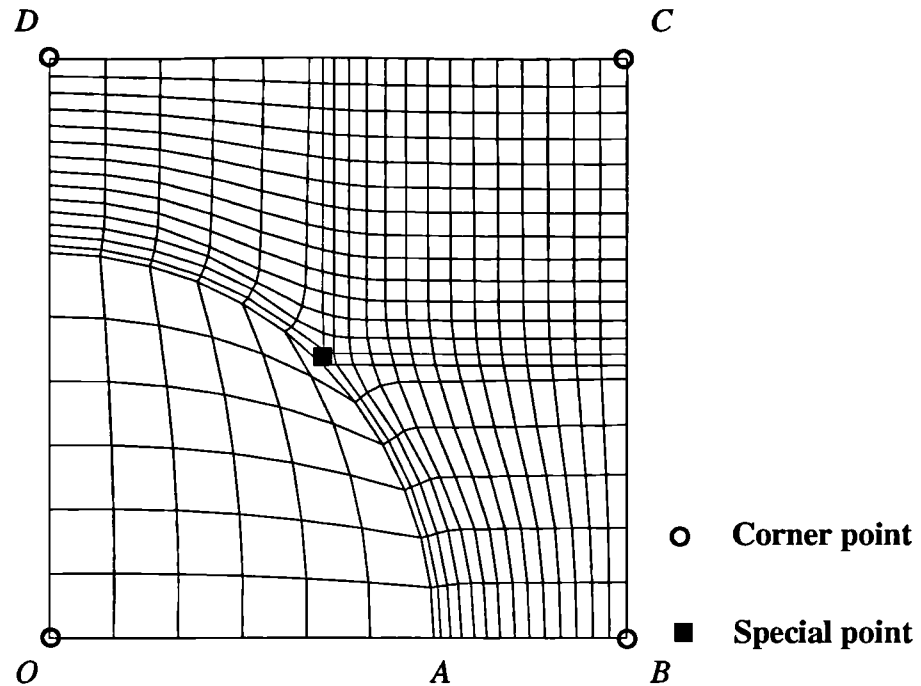


Figure 32. Block grid topology.

It is not necessary to locate the special point at 45° ; by moving it close to 90° , it is possible to generate grids which are topologically similar to H-grids, but with the potential to encompass a manifold of cylinders. Figures 33 and 34 show examples of this type of grid, as used to study transient flow in in-line and staggered tube banks. These types of grid will henceforth be referred to as a 'block H-grids'.

It is technically possible to move the special point to 90° , i.e. to create true H-grids, and still pass cells through the cylinders, by overlaying one or more lines of cells along the centreline between the cylinders. However this causes complications, not only when computing the metric coefficients in the grid generation software, but also within the flow solver itself (where it is necessary to account for the zero displacements in the diffusion coefficients, and staggered velocity fields). Since there is little to be gained by this, the practice was not adopted. Grid generation is important, only in as much as it provides a framework for the flow solver. It is the solution of the problem which ultimately is of interest.

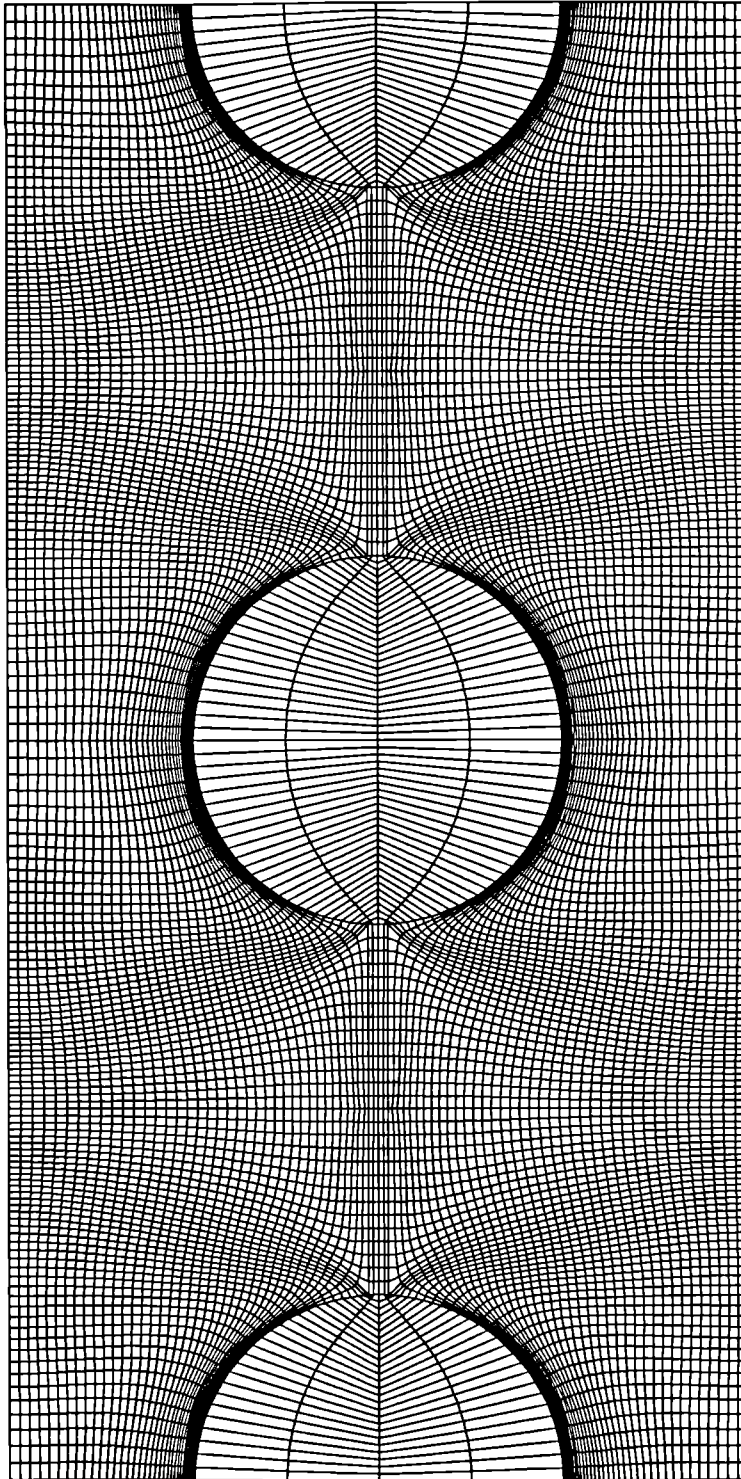


Figure 33. In-line block H-grid.

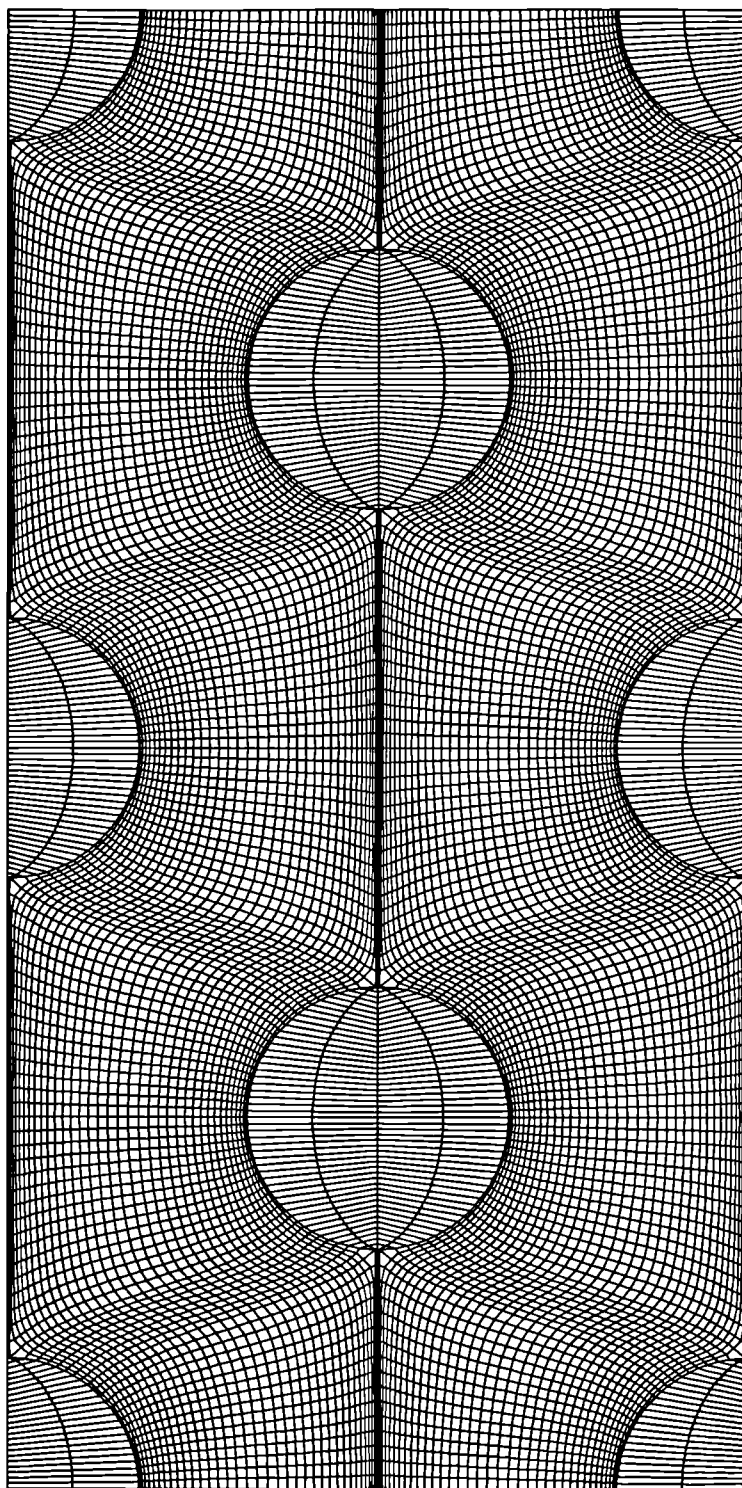


Figure 34. Staggered block H-grid.

6.7 Three-dimensional Grids

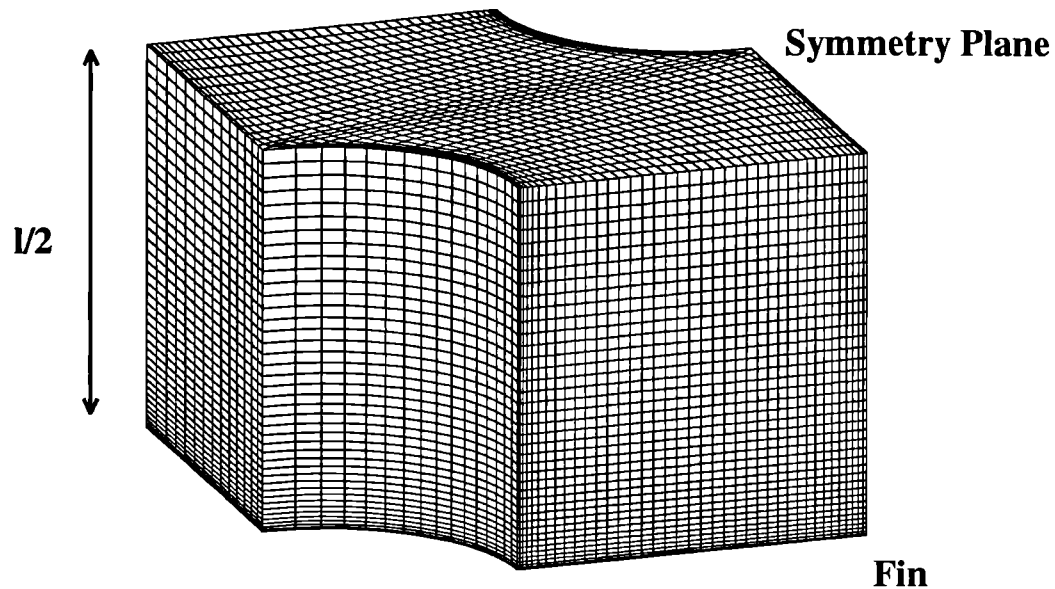


Figure 35. Three-dimensional grid.

When simulating fluid flow and heat transfer in plate-fin and tube heat exchangers, both in-line H-and diagonal H- and I-grids were deployed. Figure 35 shows an example of one such grid based on a diagonal H-pattern. The grid is concentrated in the azimuthal direction, using a geometric progression.

The grids used to study 3-D fluid flow in heat exchanger modules were based on the basic 2-D patterns described above. A FORTRAN utility, written by the author, converted the data set to three-dimensions. The same program also allowed the user to generate multiple modules in 2-D or 3-D, from a single basic pattern, by multiple translations and

reflections, and also the user could add one or more line of 'halo cells' (discussed in Chapter 8) if required.

6.8 Closure

The grid generation procedure was introduced as a step-wise procedure. The domain boundaries are defined, and the location of the grid corners and special points identified. The grid nodes along the sides and interior of the domain are then generated. The latter is achieved, initially, by transfinite interpolation from the boundary points. Following this, an elliptic generation procedure is applied. Values at certain boundaries are then adjusted, as well as source terms, P and Q , in the finite-volume equations. The process is reiterated, until a satisfactory grid was obtained. Treatment of the method of prescribing source-terms was discussed in detail. A number of different grid types; O-grids, H-grids, and I-grids as well as some simple multi-block grids were presented.

7. STREAM FUNCTIONS

7.1 Introduction

The general concepts involved in generating stream functions are in many ways similar to those required for grid generation. Unlike the latter, numerical generation of stream functions has received little attention to-date. A significant amount of time was devoted to developing both 2-D and 3-D stream function algorithms. As a result of difficulties encountered in the latter more general case, only 2-D stream-lines are displayed in the thesis. The mathematical basis for the 3-D case is described in some detail below, for future application.

There are two methods of generating stream functions:

- (a) By 'integration'.
- (b) By numerical solution of partial differential equations.

In two-dimensions, (a) has the advantage of being non-iterative, although in three-dimensions this is not the case. There are situations where method (b) yields better results in two-dimensions, as is discussed below. Since integration is a limiting case of the more general case (b), the stream function methodology will be introduced in this context.

At first sight it might appear that the use of partial differential equations to generate graphical data is excessively luxurious. Partial differential equations are widely used for grid generation, i.e. pre-processing. There, of course, no requirement to employ such

schemes. The justification for the additional computational cost is based on the quality of the results. For the same reason, there is no reason not to use iterative schemes when generating graphical output (post-processing).

7.2 Governing Equations

A stream function, ψ , is any function such that,

$$\rho \vec{u} \cdot \vec{\nabla} \psi = 0 \quad (7.1)$$

A stream function has the physical significance that constant ψ -values are stream-lines in 2-D, and stream-surfaces in 3-D, i.e. they are tangent to the velocity field. Equation 7.1 is a scalar transport equation with convection only present. This is not a form suitable for solution by the methods given in this thesis. For 2-D incompressible planar flow a stream function may be defined by,

$$u_c = \frac{\partial \psi}{\partial y_c}, \quad v_c = -\frac{\partial \psi}{\partial x_c} \quad (7.2)$$

such that Eq. 7.1 is identically zero. Equation 7.2 may also be written as,

$$\nabla^2 \psi = -\omega \quad (7.3)$$

where,

$$\omega \equiv \frac{\partial v_c}{\partial x_c} - \frac{\partial u_c}{\partial y_c} \quad (7.4)$$

is the two-dimensional fluid vorticity. Equation 7.3 is a Poisson equation, i.e. a diffusion-source transport equation, with $\Gamma = 1$. Equations 7.2 and 7.3 are equivalent forms suitable for solution using finite-difference and finite volume methods, respectively.

For three-dimensional flow it is possible (Giese, 1951, Yih, 1957) to define two stream functions, ψ_1, ψ_2 , which satisfy Eq. 7.1 by,

$$\rho \vec{u} \equiv \lambda \vec{\nabla} \psi_1 \times \vec{\nabla} \psi_2 \quad (7.5)$$

The reader's attention is confined to the case of incompressible flow (without loss of generality choose $\lambda = \rho$). If ψ_1 and ψ_2 are chosen to be mutually orthogonal functions, it follows that,

$$\vec{\nabla} \psi_1 = \frac{\vec{\nabla} \psi_2 \times \vec{u}}{\vec{\nabla} \psi_2 \cdot \vec{\nabla} \psi_2} \quad (7.6)$$

$$\vec{\nabla} \psi_2 = -\frac{\vec{\nabla} \psi_1 \times \vec{u}}{\vec{\nabla} \psi_1 \cdot \vec{\nabla} \psi_1} \quad (7.7)$$

Equations 7.6 and 7.7 are the required form of equations for solution using a finite-difference scheme. By choosing $\psi_2 = \vec{k}$, the 2-D case [Eq. 7.2] is recovered.

7.3 Discretization

The governing equations for orthogonal stream functions [Eq. 7.6 and 7.7] are in a form suitable for discretization. Let it be supposed that ψ_2 is (temporarily) known. From Eq. 7.6,

$$\frac{\partial \psi_1}{\partial x} = \frac{1}{\nabla \psi_2^2} \left(\frac{w \partial \psi_2}{\partial y} - \frac{v \partial \psi_2}{\partial z} \right) \quad (7.8)$$

$$\frac{\partial \psi_1}{\partial y} = \frac{1}{\nabla \psi_2^2} \left(\frac{u \partial \psi_2}{\partial z} - \frac{w \partial \psi_2}{\partial x} \right) \quad (7.9)$$

$$\frac{\partial \psi_1}{\partial z} = \frac{1}{\nabla \psi_2^2} \left(\frac{v \partial \psi_2}{\partial x} - \frac{u \partial \psi_2}{\partial y} \right) \quad (7.10)$$

Thus it is possible to write a set of a linear algebraic equation having the general form of Eq. 5.1,

$$a_E = \frac{A_e}{|\vec{P-E}|} \quad (7.11)$$

and similar terms for a_W, a_N etc. The source terms are as follows

$$S = S_E - S_W + S_N - S_S + S_H - S_L \quad (7.12)$$

These are computed as follows: A velocity vector, $\vec{u}_P$ is constructed at P based on the average of the three pairs u_w and u_e, v_s and v_n, w_l and w_h . Similarly a vector $\text{grad } \psi_2$

is constructed based on the averages $(\psi_{2E} - \psi_{2P})/P-E$ and $(\psi_{2P} - \psi_{2W})/W-P$ etc. The cross product of these two vectors is formed to obtain,

$$\vec{S}_P = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}_P = \vec{u}_P \times \vec{\nabla} \psi_{2P} \quad (7.13)$$

To obtain,

$$S_E = \frac{1}{2} A_e (S_{xP} + S_{xE}) \quad (7.14)$$

with similar terms for the other neighbours. The resultant ψ_1 may then be used to solve for ψ_2 in a similar fashion at the next sweep. Thus the pair of coupled linear algebraic equations,

$$L(\psi_1, \psi_2) = 0 \quad (7.15)$$

are solved iteratively. Note that any attempt to eliminate ψ_2 from the equation for ψ_1 results in the recovery of a pair of equations having the form of Eq. 7.1.

The 3-D formulation given above is strictly correct only for orthogonal stream functions in orthogonal BFC grids. It follows from the resolute form of the vector product (Appendix D), that to obtain correct results in non-orthogonal grids, it is necessary to compute all of the tangential and normal resolute of $\vec{u}$, $\text{grad } \psi_1$, and $\text{grad } \psi_2$ and to divide by the direction cosines between the normal directions. This additional overhead was not considered justified.

The 2-D case may be obtained for example by setting $\psi_2 = z$. Under these circumstances, the finite-difference equation for ψ_2 is the same as the finite-volume equation

obtained by integrating the stream-function-vorticity equation [Eq. 7.3]. In practice, however, a separate 2-D utility was written in order to (a) preserve storage and (b) account for non-orthogonality.

7.4 Graphics Utility

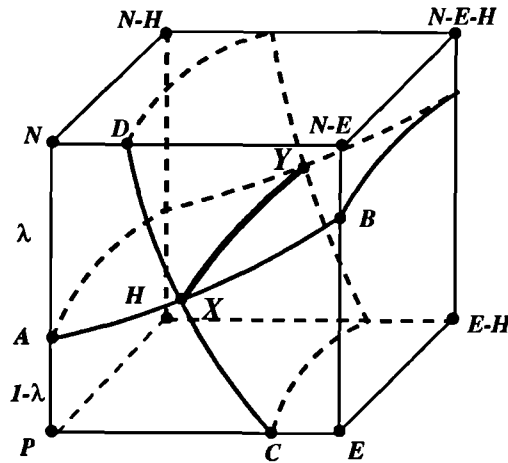


Figure 36. Stream-function graphics utility.

A graphics procedure was written by the author in order to plot stream-lines from field data. A grid was constructed with corners corresponding to the cell nodes P . Let it be said that a set of corner values “span” a given value, V , if at least one corner value is less than V , and at least one greater than V . It is desired to locate stream-lines corresponding to $\psi_1 = V_1$, and $\psi_2 = V_2$. The treatment was as follows:

- (i) Each cell was parsed to see if the eight corner values spanned both $\psi_1 = V_1$, and $\psi_2 = V_2$.
- (ii) The cell faces were in turn parsed to see if they spanned both $\psi_1 = V_1$ and $\psi_2 = V_2$.
- (iii) If so, the four corner pairs (e.g. $P-E$, $E-NE$, $NE-N$, $N-P$) on the given face were parsed to see if they spanned either $\psi_1 = V_1$ or $\psi_2 = V_2$. The position of points A and B [Fig. 36], corresponding to the intersection of the surface $\psi_1 = V_1$, etc., with the cell face were computed by linear interpolation from the positions of the corner points P , N , etc.,
$$\lambda = (V_1 - \psi_P)/(\psi_N - \psi_P)$$
- (iv) The lines $A-B$ and $B-C$ were analysed to see if they intersect. (It is possible for a set of corners to span a pair of values, without the two surfaces intersecting each other). The point of intersection X of the two lines (approximating the curves) was then computed.
- (v) Step (ii)-(iv) were repeated until a second point Y found, at which point, the line $X-Y$ was drawn.

The reader will note that although a rectilinear cell has been shown, in general the grid is body-fitted. This means that some approximation is required if linear geometry is used to compute the points of intersection X and Y (the use of higher-order polynomials was considered unjustified). The approximations used always resulted in consistent values of points X , Y , etc., at neighbouring cells. The process was repeated over the whole field for successive pairs of values over the entire range of values resulting in a double-contour field.

7.5 Discussion

Test conducted on a number of elementary flow problems showed that the 3-D stream function methodology worked well in both Cartesian grids, provided the grids were reasonably uniform. Stream-lines generated on these elementary problems were similar to the 2-D case, in that stream-lines either enter and exit the domain of interest, or form simple closed loops. This is in contrast to particle-tracking methods where as a result of round-off, vortical zones are indicated by endless spirals. (In near-wall zones, where velocities are small, particle-drift can give false impressions of the flow field. The stream-line results were less ambiguous than the equivalent results generated using existing particle utilities. The stream functions, defined by Eqs. 7.6 and 7.7. also have the property that the line density is proportional to that mass flow.

Difficulties were encountered when simulating complex flow in body-fitted coordinates, particularly when non-uniform grids were concentrated, e.g. in wall regions, i.e. when areas and lengths were proportionally larger in specific directions. Under these circumstances meaningful results proved very difficult to obtain, even when large relaxation was applied. Regrettably, a satisfactory solution to this problem could not be found in time for inclusion into the thesis, and for this reason the procedure was abandoned in favour of particle-tracking utilities, for the purposes of illustrating the 3-D results of Chapter 10. It is for this reason that no results are supplied at this time. However it is maintained that, in spite of the problems encountered in this initial attempt to generate stream-lines in 3-D, the concept appears to be quite promising: There are relatively few tools available to illustrate CFD results, and this subject-area should be pursued in the future. In addition to displaying stream-lines (either stationary or otherwise) the technique could also readily be adapted to display vortex lines in three dimensions.

The complexity of obtaining a solution is as a result of the coupled-nature of the governing equations: Any attempt to de-couple Eqs. 7.6 and 7.7. leads back to Eq. 7.1. The existing flow-solvers [Chapter 5] were for the most part employed, although there is no reason to not to suppose that other solvers might generate superior results. For example a slab-solver, marching across the main flow direction in the directions of $\text{grad } \psi_1$ and $\text{grad } \psi_2$ (as appropriate) may well prove better suited to the problem at hand, than one which proceeds in the direction of the main flow. In 3-D, it is also possible to obtain results by 'integration'; de-activating all but one linking coefficient in two mutually perpendicular directions, from the finite-volume equations. However, in this case, the procedure cannot be made entirely non-iterative; not only are the source terms coupled, but also off-slab neighbour values are present. Provided the stability problems can be solved, it is possible that such methodologies could prove useful in the future.

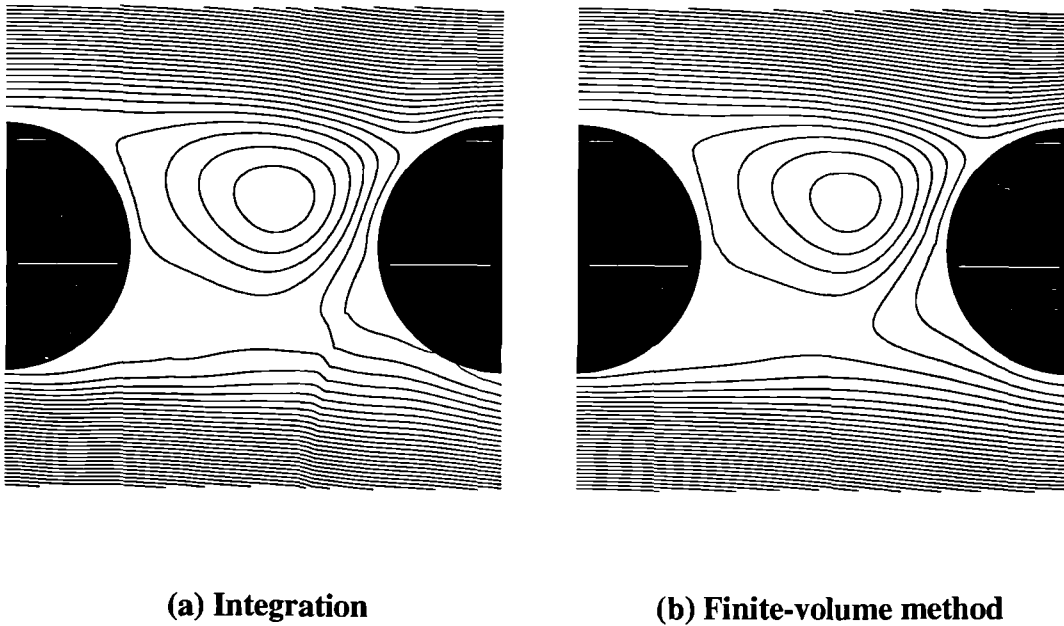


Figure 37. Two-dimensional stream-lines.

Iterative procedures were however employed in this thesis when considering 2-D transient flow in tube banks. Equation 7.4 may readily be integrated, and discretized using the methods described in Chapter 5. Appendix III contains a procedure developed by

the author for this purpose, appropriate for both Cartesian and non-orthogonal BFC problems. The procedure was applied to existing results after the end of the solution procedure. In all cases convergence was rapid. The reader will note the source term need only be computed once in the 2-D case, and that there is no need to allocate storage for 'porosities', or other redundant variables.

Figure 37 shows stream-lines obtained (a) by integration, and (b) using the finite-volume procedure. The results are from Chapter 9, the grid used being the orthogonal multi-block grid shown in Fig. 33. It can be seen that there is a significant discontinuity in the stream-lines obtained using the non-iterative procedure [lower right side Fig. 37a]. These discontinuities were systematic, always occurring at the same locations, and are due to the fact that small rounding errors in the stream function-values passing from the fluid to the solid region are not corrected due to the uni-directional nature of the algorithm (integration is from the top down). Substantially worse examples of this problem (than shown in Fig. 37a) were at times observed.

The finite-volume-based solution to Eq. 7.4, does not suffer from this defect: Owing to the fact that all four linking coefficients are active, numerical error is rapidly dissipated in a uniform fashion. In general the iterative procedure is recommended for problems involving multiply-connected geometries, multi-block grids involving the use of porosities, and/or where highly non-orthogonal grids are employed. Where Cartesian, or body-fitted grids derived from the basic patterns shown in Figs. 21 and 22 are employed, there is little advantage to using a finite-volume method, and, for reasons of economy, a non-iterative procedure should be employed.

7.6 Closure

A method for generating and plotting stream-surfaces and stream-lines in two-dimensions and three-dimensions has been described. The method which involved the solution of a pair of coupled linear algebraic equations in three-dimensions, and a single equation for two-dimensions, was described.

8. STEADY TWO-DIMENSIONAL FLOW

8.1 Introduction

In Chapter 4, the flow of an ideal fluid in tube banks was explored, using classical analytical methods. This chapter is concerned with two-dimensional viscous flow and heat transfer. Let it be assumed that the flow is in the x - y plane with the x the stream-wise direction¹. The flow is characterised by a Reynolds' Number, Re , which is defined as,

$$Re \equiv \frac{u_m d}{\nu} \quad (8.1)$$

where ν is the kinematic viscosity of the fluid, u_m is the bulk velocity in the minimum inter-tube cross-section, and d is the diameter of the cylinder.

8.1.1 Boundary conditions

Treatment of the boundary conditions is sufficiently important to warrant a fairly detailed description. The prescription is similar to methods commonly used for many non-periodic fluid-flow problems; namely the prescription of mass flux upstream, and of pressure downstream. Consider, for convenience, an in-line tube bank. (The same process is employed with staggered tube banks, except that the positions and signs of variables are reversed as necessary). Let it be assumed that the flow is fully-developed so that values of velocity are identical at stages $x_c + ns_2$, $n = 1, 2, 3, \dots$. The velocity field is

¹ The flow was in fact simulated in the y - z plane with z the stream-wise, y the cross-wise direction (and x the azimuthal direction for the 3-D problems described in Chapters 10 and 11). For unsteady 2-D flow [Chapter 9] the x - y plane was selected with y the stream-wise direction. These details are only of concern to users of the PHOENICS computer code.

presumed cyclic, while the pressure distribution may be considered to be composed of a doubly-periodic component together with a linear term in the stream-wise direction (by analogy to the velocity potential in Chapter 4). Let the overall pressure drop between any two points (x_c, y_c) and $(x_c + s_2, y_c)$ being a constant $\Delta\bar{p}$. Thus,

$$p(x_c, y_c) = p(x_c + s_2, y_c) + \Delta\bar{p} \quad (8.2)$$

$$\vec{u}(x_c, y_c) = \vec{u}(x_c + s_2, y_c) \quad (8.3)$$

For constant wall temperature, t_w , a non-dimensional temperature, θ , must be such that,

$$\theta(x_c, y_c) = \theta(x_c + s_2, y_c) \quad (8.4)$$

where,

$$\theta(x_c, y_c) \equiv \frac{t(x_c, y_c) - t_w}{t_b(x_c) - t_w} \quad (8.5)$$

where t_b is the bulk temperature at x [see Fig. 61],

$$t_b(x_c) = \frac{\int u_c \cdot t \, dy_c}{\int u_c \, dy_c} \quad (8.6)$$

In terms of enthalpy, h ,

$$h(x_c, y_c) = c_1 h(x_c + s_2, y_c) + c_2 \quad (8.7)$$

where,

$$c_1 = \frac{t_b(x_c) - t_w}{t_b(x_c + s_2) - t_w} \quad (8.8)$$

$$c_2 = c_p t_w (1 - c_1) \quad (8.9)$$

For constant heat flux, q_w , Eq. 8.7 may again be used, but with

$$c_1 = 1 \quad (8.10)$$

$$c_2 = c_p [t_b(x + s_2) - t_b(x)] \quad (8.11)$$

Let a distinction be made between the actual boundary conditions, and the means by which they are prescribed. A number of implementations are possible: It was decided to code the periodic requirement in the form of actual boundary conditions (pressure, velocity, etc.) i.e. as source terms, S , consisting of a coefficient C , and value V [Eq. 5.15] in the finite-volume equations rather than to modify the finite-volume equations by introducing additional neighbour coefficients and values at the boundaries. Although the latter is some ways preferable, for certain grids the exit and entrance regions cannot be naturally connected together in a structured manner.

In this research project a partially-explicit double-substitution scheme was adopted, using a single line of downstream halo cells. Figure 38 shows the means by which the halo cells were added, (NB: For I-grids, unnecessary halo cells were blocked using porosities).

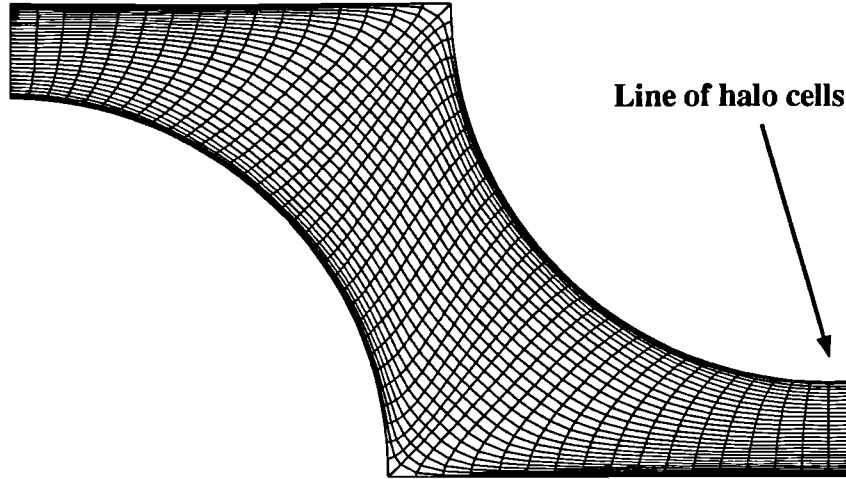


Figure 38. Halo cells.

Figure 39 illustrates the logic of the boundary prescription rationale: By adding a row of downstream halo cells, storage is automatically allocated for the (otherwise missing) exit velocities, u_{nx-1} . These downstream values are then used to compute the upstream normal resolute which when multiplied by fluid density, ρ , render the upstream mass flux source term for the continuity equation at the start of the next sweep, according to,

$$S_m = V_p = c\rho A_e u^{(nr)} \quad (8.12)$$

where the constant, c , is given by,

$$c = \frac{u_0}{u_b} \quad (8.13)$$

u_0 is the desired and, u_b the actual bulk velocity, based on the sum of the exit velocity normal resolute, $u^{(nr)}$, preventing the Reynolds' number, Re , from wandering. Under most circumstances, the normal resolute $u_{nx-1}^{(nr)}$ is similar in value to the tangential resolute¹ u_{nx-1} , however for some grids (e.g. in-line-I-grids with staggered flow), the

¹ As discussed in Chapter 3, the conserved values are the tangential resolute $u_{(tr)}$ and $v_{(tr)}$. In the interests of readability, these will be referred to simply as u and v . Normal resolute will be distinguished from these as $u^{(nr)}$ and $v^{(nr)}$.

grid cells at domain boundaries may not be orthogonal, so that the normal velocity resolute must be computed according to,

$$u^{(nr)} = Fu_{nx-1} + Gv_{nx-1} \quad (8.14)$$

At the outlet (i.e. the downstream halo cells at $i = nx$), either (a) the exit mass flux, or (b) the downstream pressure may be prescribed according to the appropriate upstream distribution. In the second case the mass efflux is prescribed indirectly, via the u -momentum equation. Prescription (b) was selected; downstream pressures, V_p , being fixed to the previously stored upstream values, p_1 , less the average upstream pressure (this rises to a fixed value as the solution proceeds). The choice of the pressure coefficient, C_p , [Eq. 5.16] must be sufficiently large to fix the in-cell value to the desired value. Some relaxation of the pressure was required, so the downstream pressure field was prescribed according to;

$$V_p = \alpha(p_1 - \bar{p}_1) + (1 - \alpha)V_p^* \quad (8.15)$$

where V_p^* is the previous in-cell value, α is a linear relaxation factor. p_1 is the pressure in the corresponding upstream cell, and $\bar{p}_1$ the average upstream pressure.

In the scalar transport and momentum equations, downstream scalar values, h_{nx-1} , etc., and tangential resolute u_{nx-1} , v_{nx-1} , etc. are substituted as upstream values, V_ϕ . Figure 39 is a simplified representation of the scheme: Since BFC grids are employed, the downstream velocity values are modified to account for grid curvature and divergence when substituted upstream, i.e. V_u and V_v are in fact functions of both $u^{(tr)}_{nx-1}$ and $v^{(tr)}_{nx-1}$. Downstream enthalpy values, h_{nx-1} , are scaled according Eq. 8.7 to obtain upstream values, V_h .

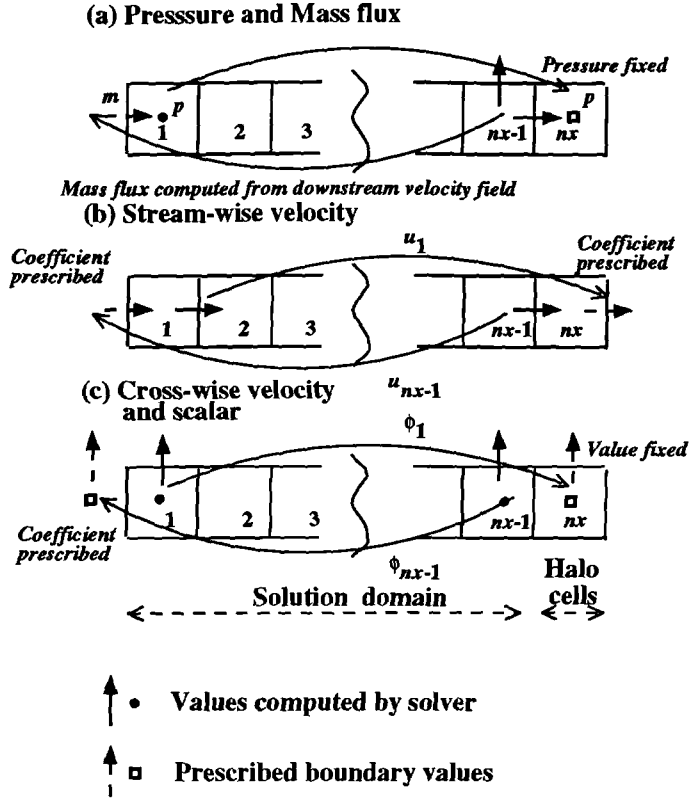


Figure 39. Periodic boundary conditions.

Previous authors restricted the choice-of-domain so that the entrance and exit boundaries always corresponded to regions with no recirculation. Under these circumstances stream-wise diffusion may be presumed negligible, for $Re \gg 1$. This results in two simplifications:

- (i) The linear coefficient, C_ϕ , used to prescribe the upstream boundary source terms is just the convection flux S_m [Eq. 8.12] (the so-called upwind-scheme).
- (ii) No downstream values for conserved quantities are required. Only pressure/mass flux needs to be prescribed downstream.

The requirement that the entrance and exit be located in such a region unnecessarily restricts the choice of grid-types, and this presumption was not made here. It was therefore assumed the entrance and exit coincide with regions where recirculation could occur. The following additional modifications were made:

- (a) The upstream coefficients, C_ϕ are set according to the hybrid scheme [Eq. 5.11] as,

$$C_\phi = \max(0, \frac{FA}{\delta x} - 2|S_m|) + \max(0, -S_m) \quad (8.16)$$

Momentum-equation coefficients were averaged as appropriate to account for the staggered grid geometry. Furthermore, the effects of non-orthogonality on both the convective and diffusive contributions were considered.

- (b) Downstream neighbour values of field variables are required: These are obtained in a similar fashion to above, but based on the values at the first cell, ϕ_1 . For stream-wise momentum, the value V_u at $i = nx$ is computed from the velocity values u_1, v_1 at $i = 1$, (modified to account for grid curvature), the C_u coefficients being computed according to the hybrid scheme. In the case of cross-wise-momentum it is sufficient to fix the downstream in-cell value, V_v , at nx to the upstream value, v_1 (without modification), using a large coefficient, C_v . Similarly, the in-cell enthalpy at $i = nx$ is fixed to a value V_h , based on h_1 , using the inverse form of Eq. 8.7. The method of setting the downstream pressure and upstream mass flux remains unchanged.

8.2 Results

The effect of the following parameters was investigated:

- (i) Bank configuration (in-line square, rotated square, or equilateral triangle)
- (ii) Pitch-to-diameter ratio, s/d .
- (iii) Reynolds' number, Re .
- (iv) Prandtl number, Pr .
- (v) Thermal boundary conditions (constant q_w vs. constant t_w)
- (vi) Grid type, size, and location with respect to boundaries.

The results of the analysis are presented below in the form of figures and graphs.

8.2.1 Typical flow-field results

Figure 40 shows stream-lines for an in-line tube bank, $s/d = 1.5$, for $10 \leq Re \leq 1\,000$, (generated using existing non-iterative techniques). Figure 41 shows velocity vectors in a rotated square bank, $s/d = 2$, over the same Re range; in the interests of clarity vectors are plotted at every second grid cell. Figures 42 to 43 show pressure contours in-line and equilateral tube banks, $s/d = 2$, for $Re = 100$.

Figures 44 and 45 show contours of the temperature distribution for an in-line tube bank, $s/d = 1.5$, for $10 \leq Re \leq 1\,000$, generated for $Pr = 1$ under constant temperature and constant wall flux boundary conditions. The temperature, θ , has been non-dimensionalized according to,

$$\theta(x,y) \equiv \frac{t-t_i}{t_e-t_i} \quad (8.17)$$

where t_i and t_e are the bulk temperatures at the inlet and exit of the domain [Fig. 61]. Figures 46 and 47 are similar plots for a compact rotated square bank, $s/d = 1.25$, $30 \leq Re \leq 300$. The data shown in Figures 44 to 47 shows not only the effect of varying Re , but also the impact of constant q_w and constant t_w boundary conditions upon the temperature field. Figure 48 shows the outcome of varying Pr on the temperature field, for an in-line bank, $s/d = 1.5$, $Re = 100$, constant q_w . Results are illustrated for $Pr = 0.1$, 10, and 100 (the $Pr = 1$ case is displayed in Fig. 44b).

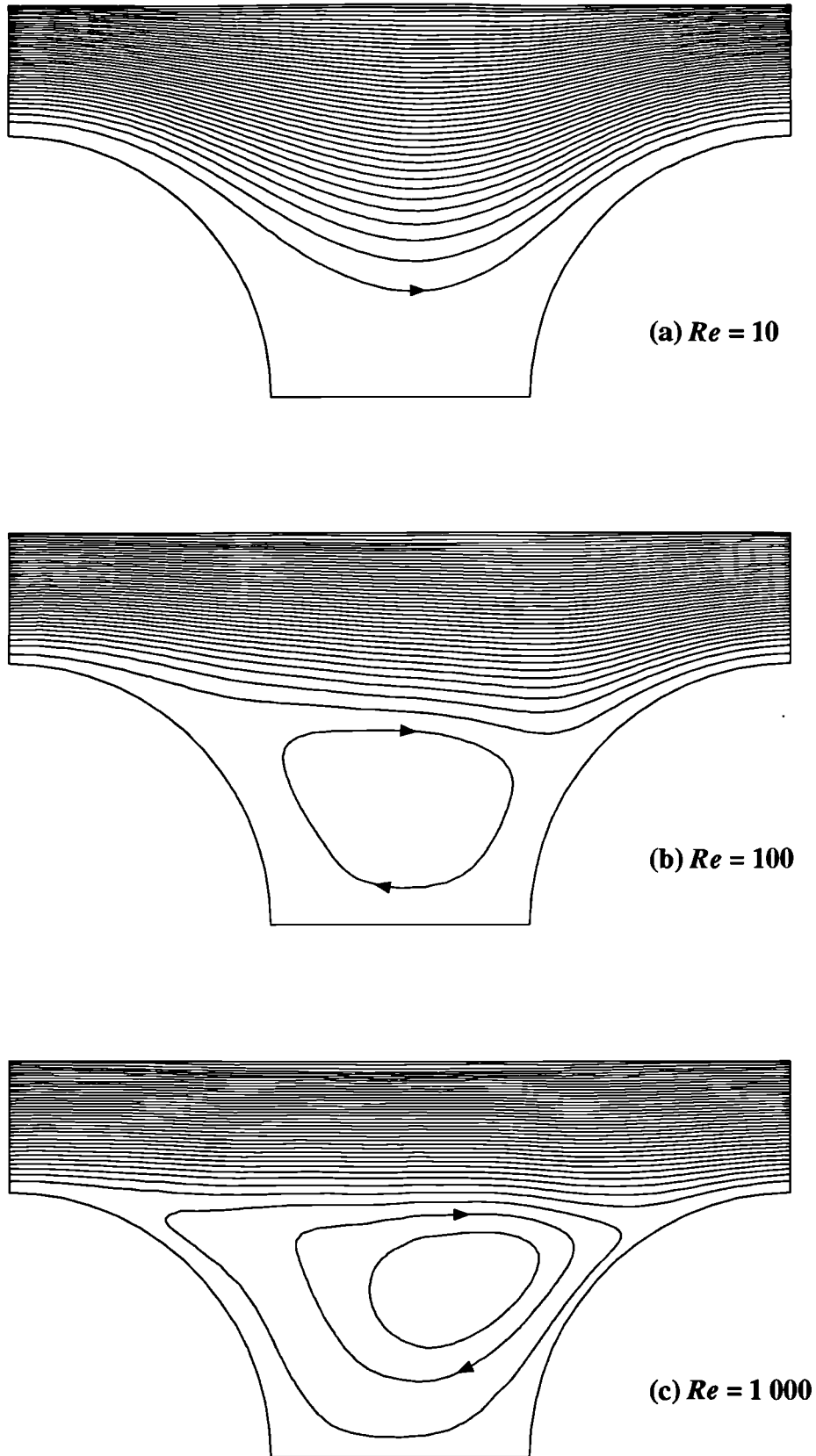


Figure 40. Stream-lines for an in-line square tube-bank, $s/d = 1.5$.

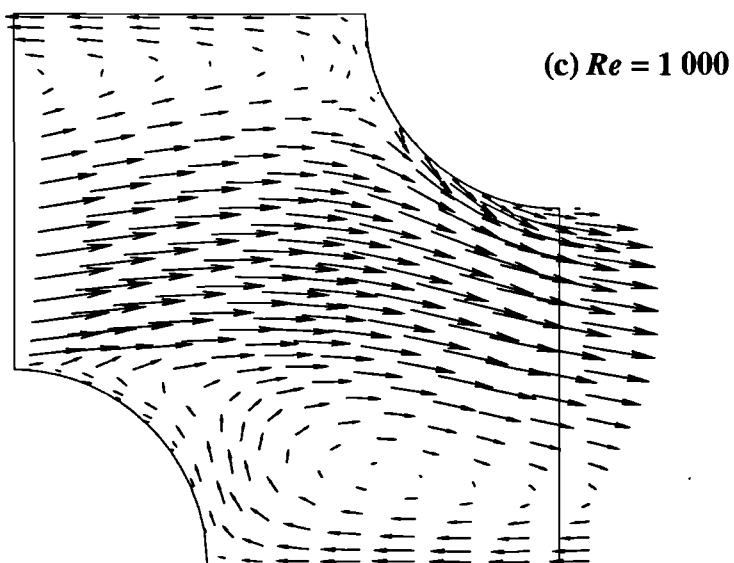
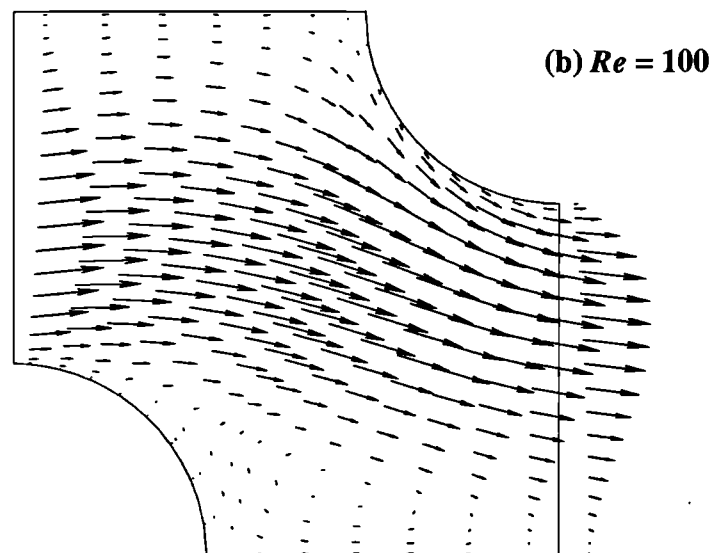
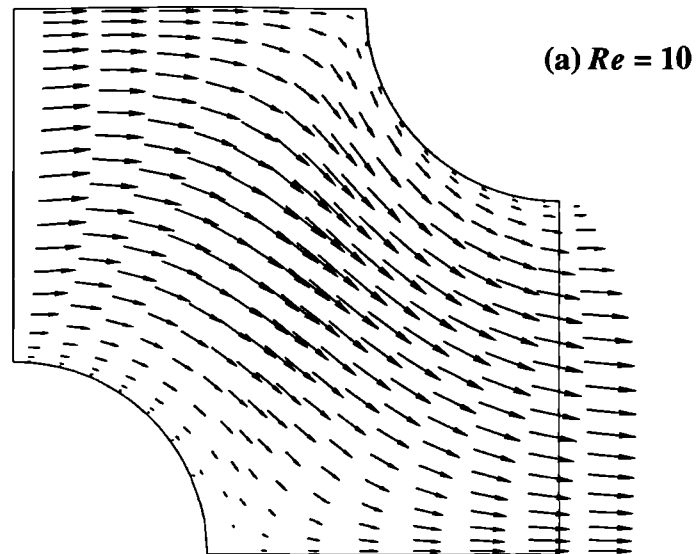


Figure 41. Velocity vectors for a rotated square tube bank, $s/d = 2$.

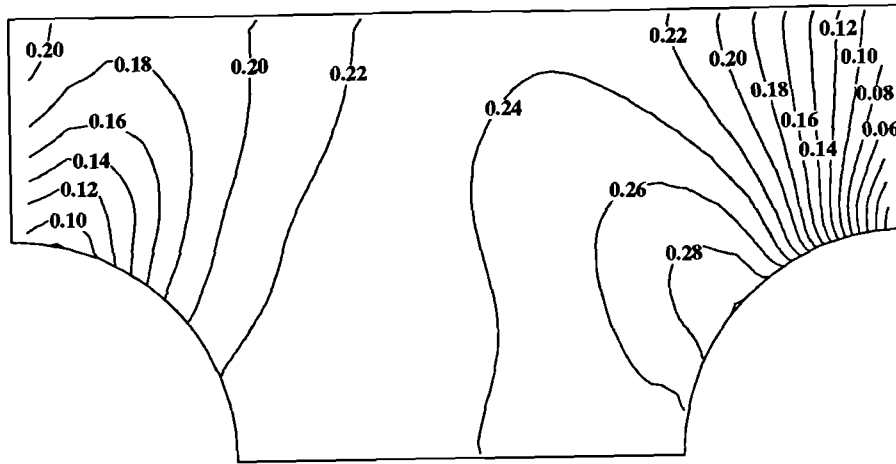


Figure 42. Pressure contours (Pa), in-line square, $s/d = 2$, $Re = 100$.

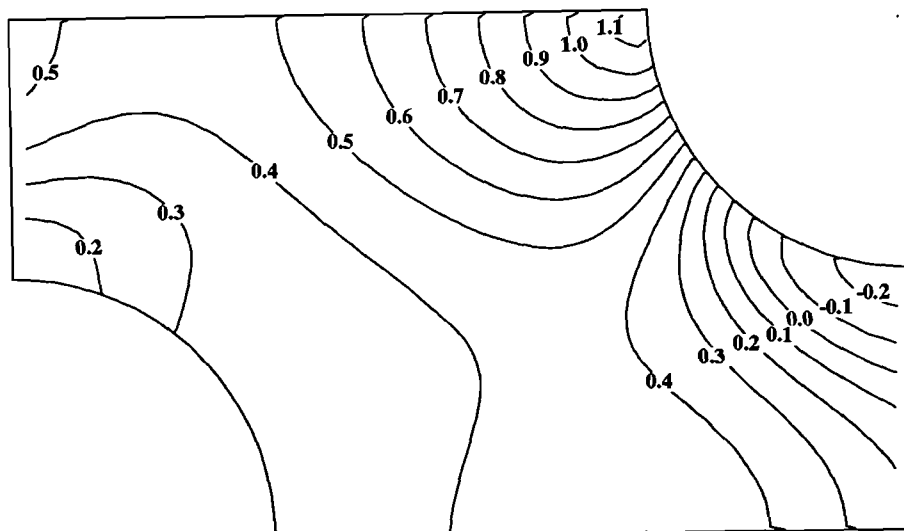


Figure 43. Pressure Contours, equilateral triangle, $s/d = 2$, $Re = 100$.

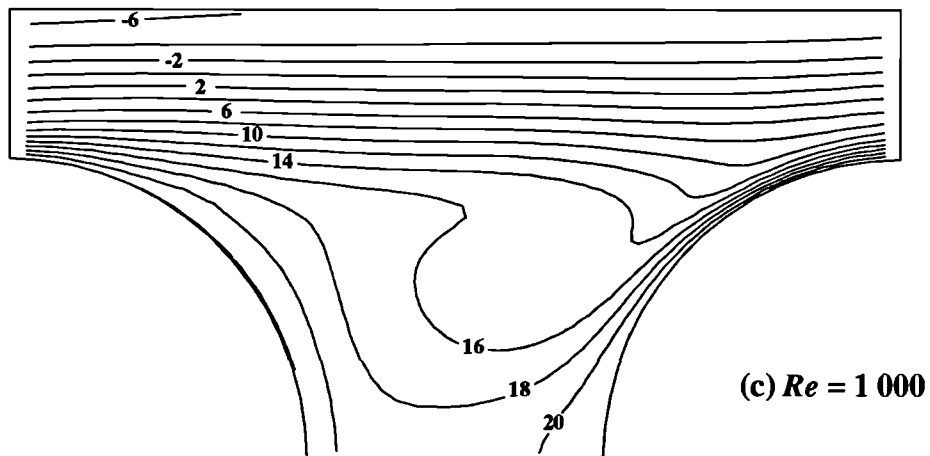
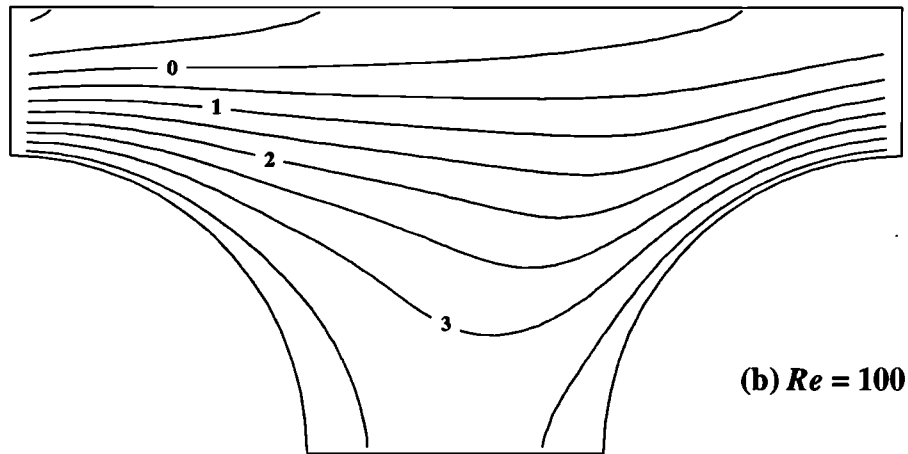
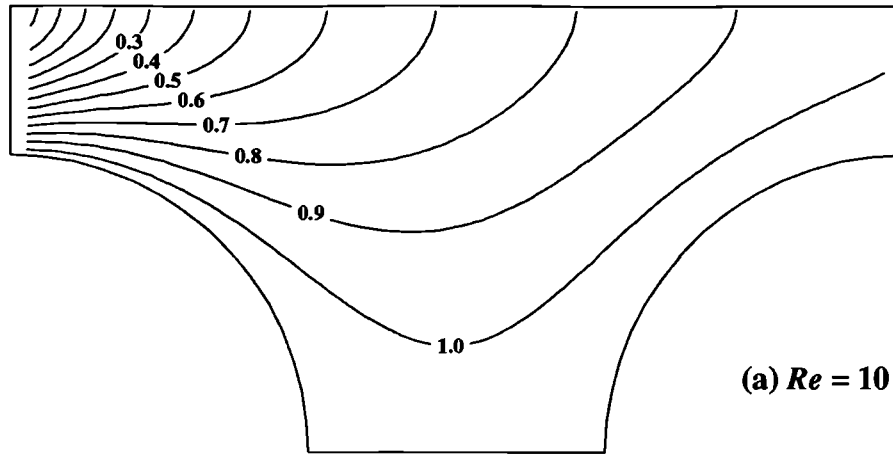


Figure 44. Temperature field, in-line square, $s/d = 1.5$, constant t_w , $Pr = 1$.

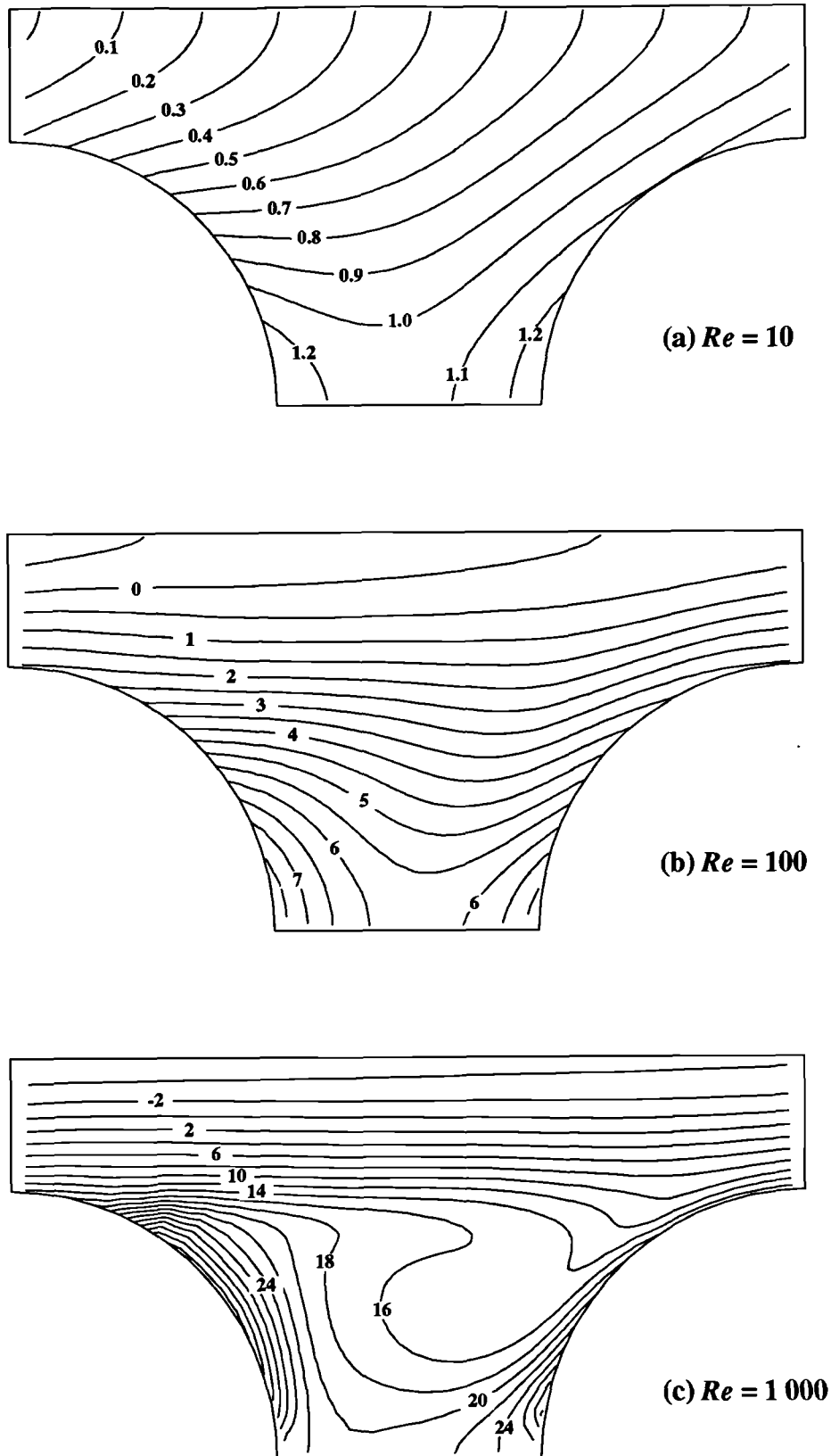


Figure 45. Temperature field, in-line square, $s/d = 1.5$, constant q_w , $Pr = 1$.

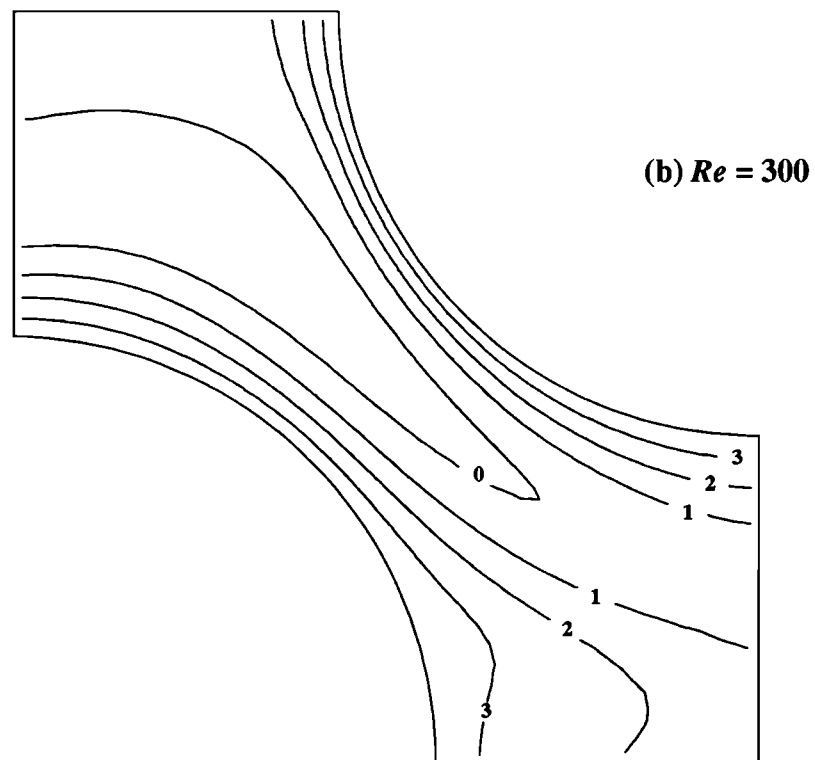
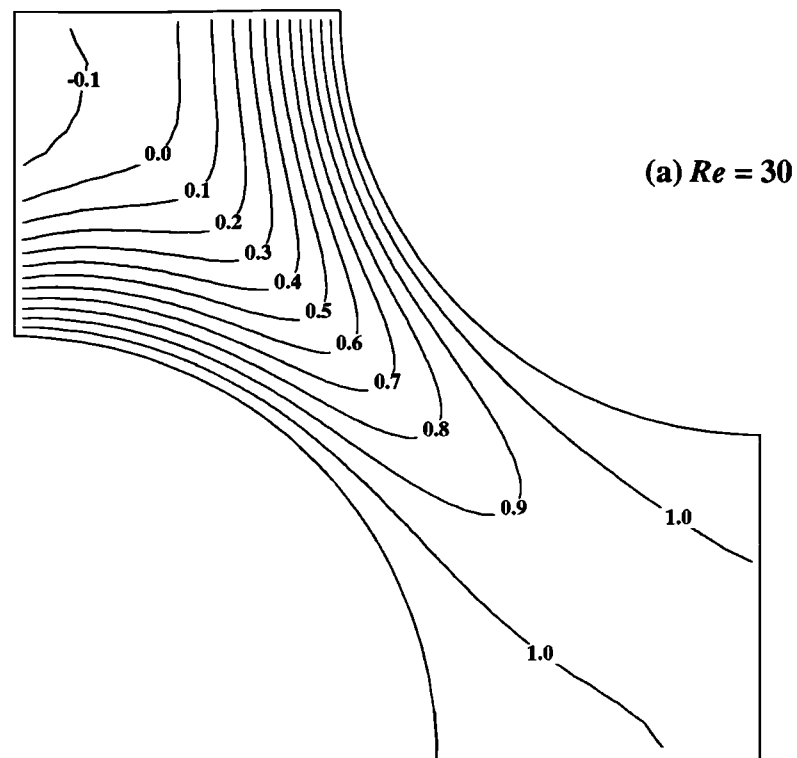


Figure 46. Temperature field, rotated square, $s/d = 1.25$, constant t_w , $Pr = 1$.

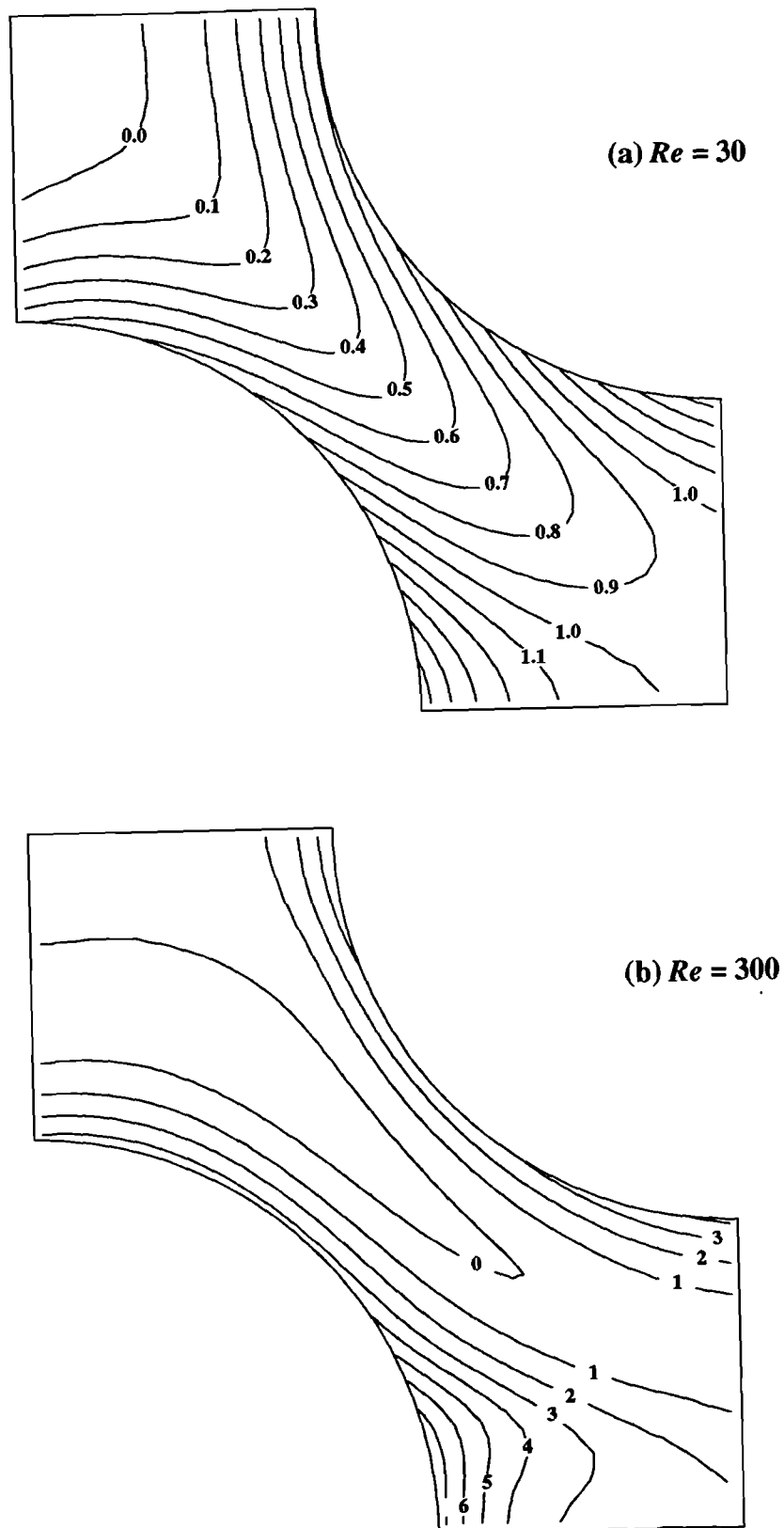


Figure 47. Temperature field rotated square, $s/d = 1.25$, constant q_w , $Pr = 1$.

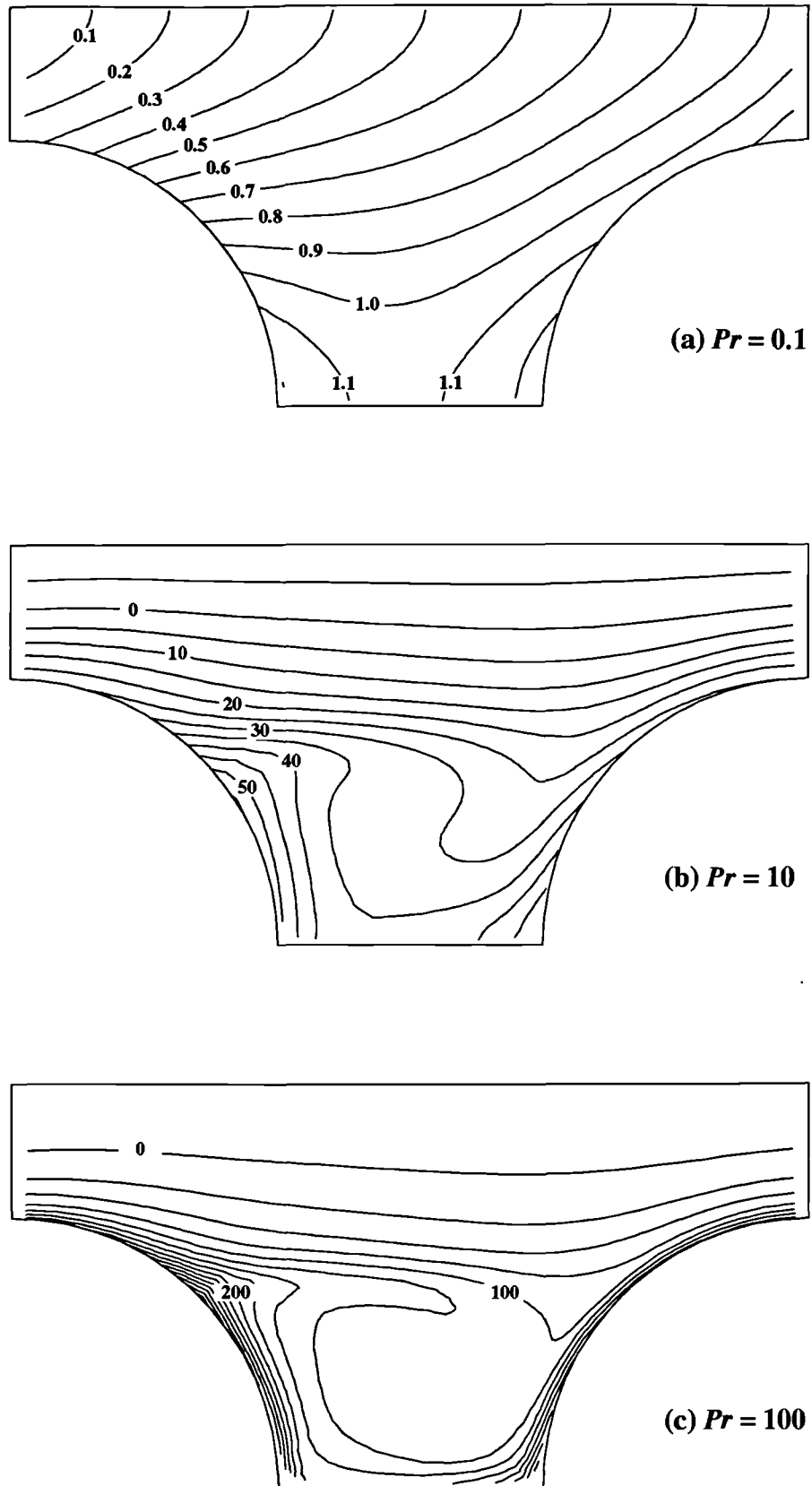


Figure 48. Temperature field, in-line square, $s/d = 1.5$, $Re = 100$, various Pr .

8.2.2 Pressure drop

Overall pressure loss is expressed in terms of an Euler Number, Eu^1 , defined by,

$$Eu \equiv \frac{\Delta \bar{p}}{\frac{1}{2} \rho u_m^2} \quad (8.18)$$

which is to be considered a function of Re [Eq. 8.1]. Values of Eu were computed based on the unweighted average of the upstream pressure values.

Figures 49 to 51 are comparisons of the author's data with the numerically-generated data of Le Feuvre (1973), Massey (1976), and Antonopoulos (1979), for in-line and staggered tube banks with $s/d = 1.25$ to 2. For the results shown in Fig. 49 (in-line square) the author's data were generated using 40x40 orthogonal H-grids. The data for rotated square banks [Fig. 50] were generated using non-orthogonal H-grids of size 40x30 ($s/d = 1.5$) and 30x40 ($s/d = 2$). For $s/d = 1.25$ a 40x40 diagonal I-grid [Fig. 30], was used. The results for equilateral-triangular tube banks [Fig. 51] were generated using 30x50 ($s/d = 1.25$), 40x30, ($s/d = 1.5$), 40x40 ($s/d = 2$) non-orthogonal H-grids.

Figs 52 to 54 show the results of the author's research compared to the empirically-based correlations of the Engineering Sciences Data Unit (1980) and the Zukauskas and Ulinskas correlations, taken from the Heat Exchanger Design Handbook (1983). (These will henceforth be referred to as ESDU and HXDH, respectively). Also shown are the isothermal data of Bergelin et. al. (1950b, 1958) in-line and staggered banks, Zukauskas et. al. (1968) in-line only, and Wallis and White (1938) staggered only. All data have been reduced to parameters consistent with those defined above.

¹ In some definitions the "1/2" is excluded from the denominator.

8.2.3 Form and profile drag

Figures 55 to 57 show the local pressure coefficient, c_p , as a function of the angle, ϕ , from the leading edge of the cylinder, for the in-line square, rotated square, and equilateral triangle banks with $s/d = 1.5$

$$c_p \equiv 1 - \frac{p(0) - p(\phi)}{\frac{1}{2}\rho u_m^2} \quad (8.19)$$

Figures 58 to 60 show the local friction coefficient, c_f , defined according to,

$$c_f \equiv \frac{\tau_w}{\frac{1}{2}\rho u_m^2} = \mu \left. \frac{\partial u}{\partial y} \right|_w \frac{1}{\frac{1}{2}\rho u_m^2} \quad (8.20)$$

where τ_w is the wall shear stress, and μ the dynamic viscosity, for the three bank-types, $s/d = 1.5$. The velocity gradient in Eq. 8.20 is computed as a difference scheme; in the interests of readability, here and throughout this text, such differences will be normally be shown as derivatives. Similarly, where summations are used to approximate integrals, these will be shown as the latter.

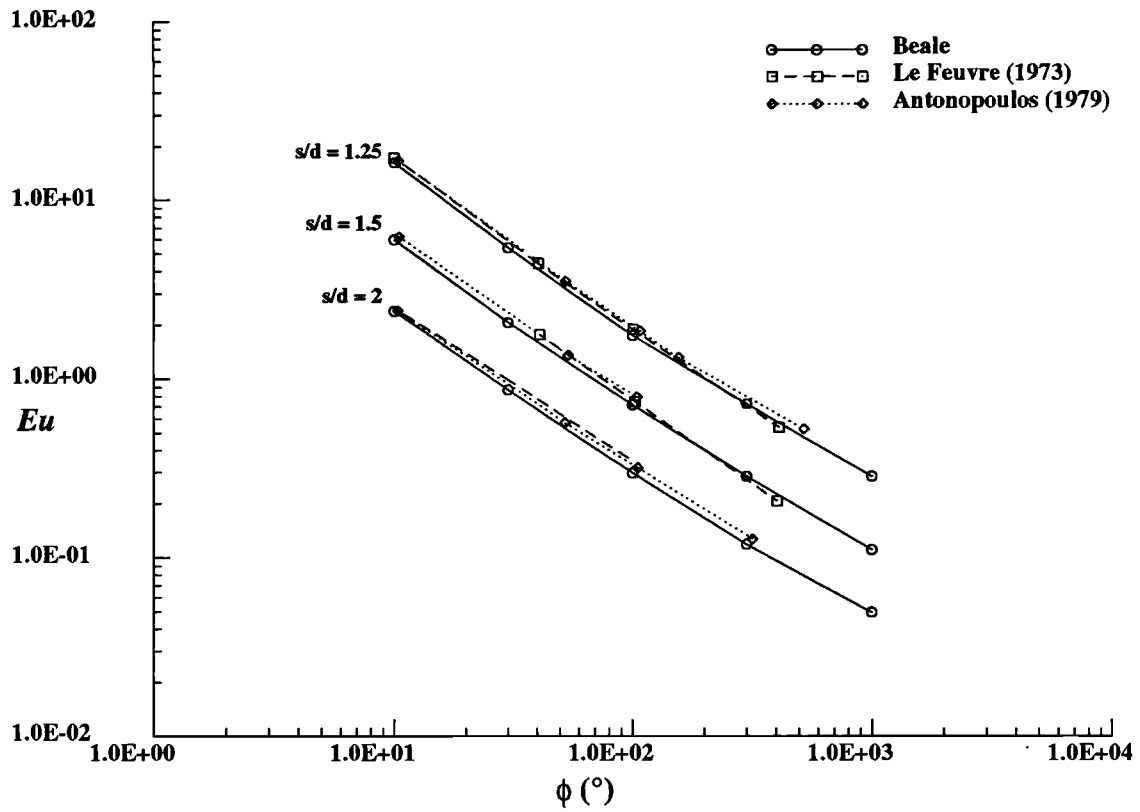


Figure 49. Eu vs. Re compared to numerical data, in-line square.

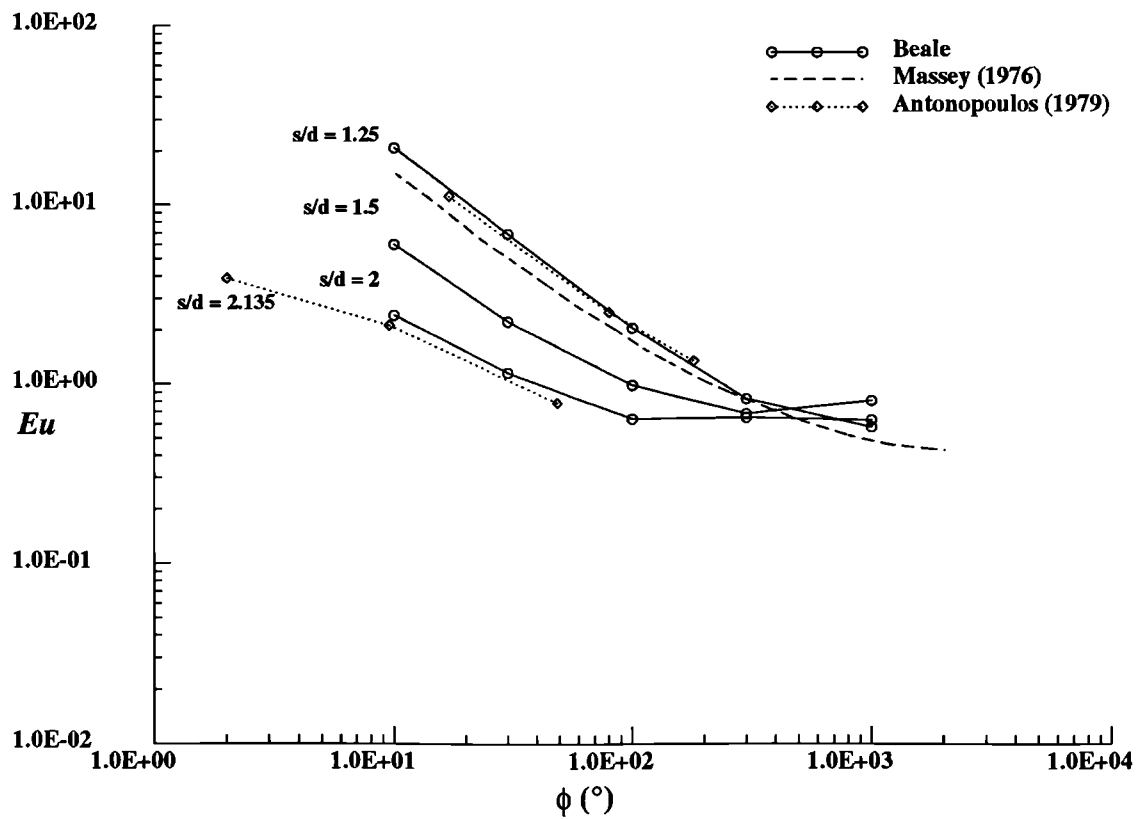


Figure 50. Eu vs. Re compared to numerical data, rotated square.

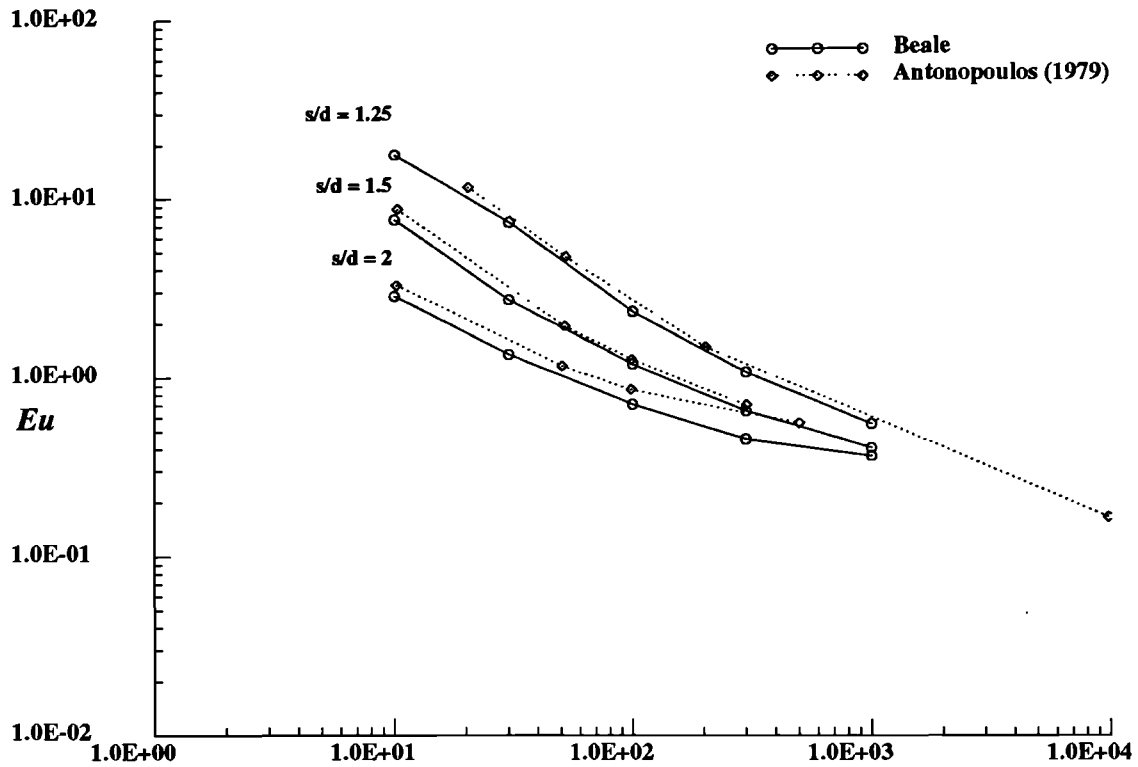


Figure 51. Eu vs. Re compared to numerical data, equilateral triangle.

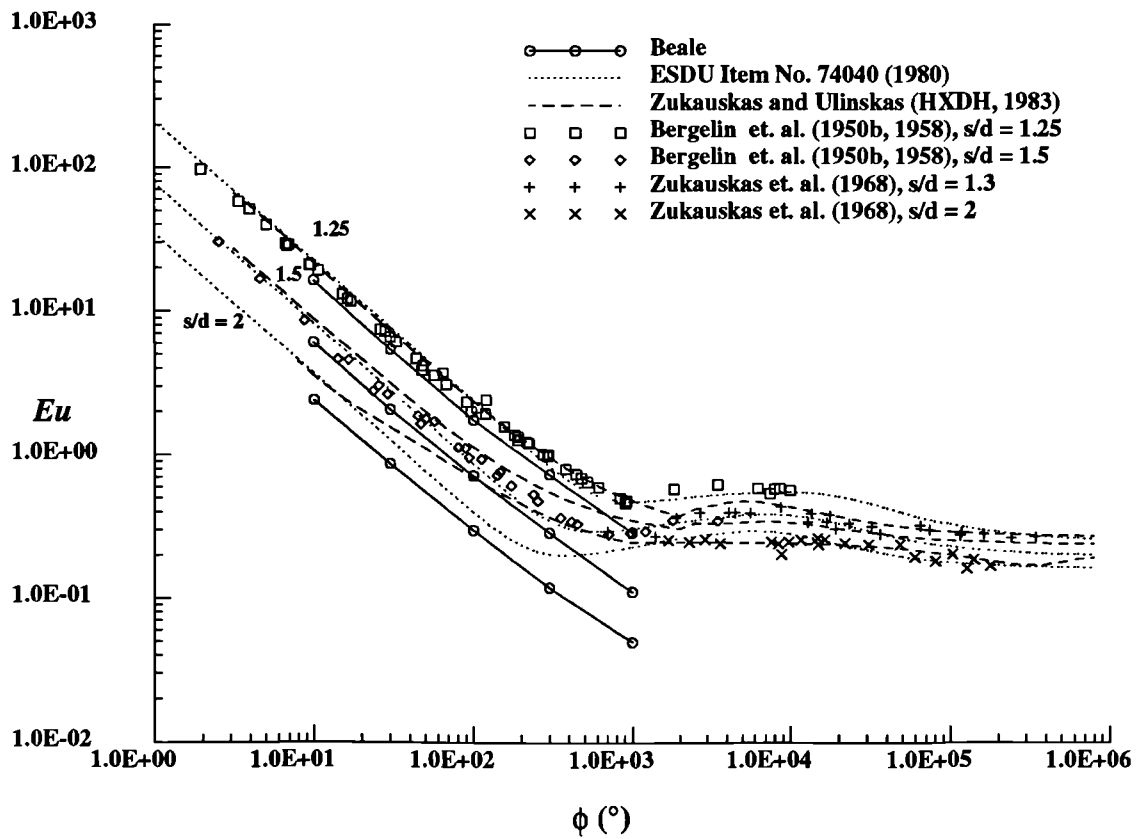


Figure 52. Eu vs. Re compared to experimental data, in-line square.

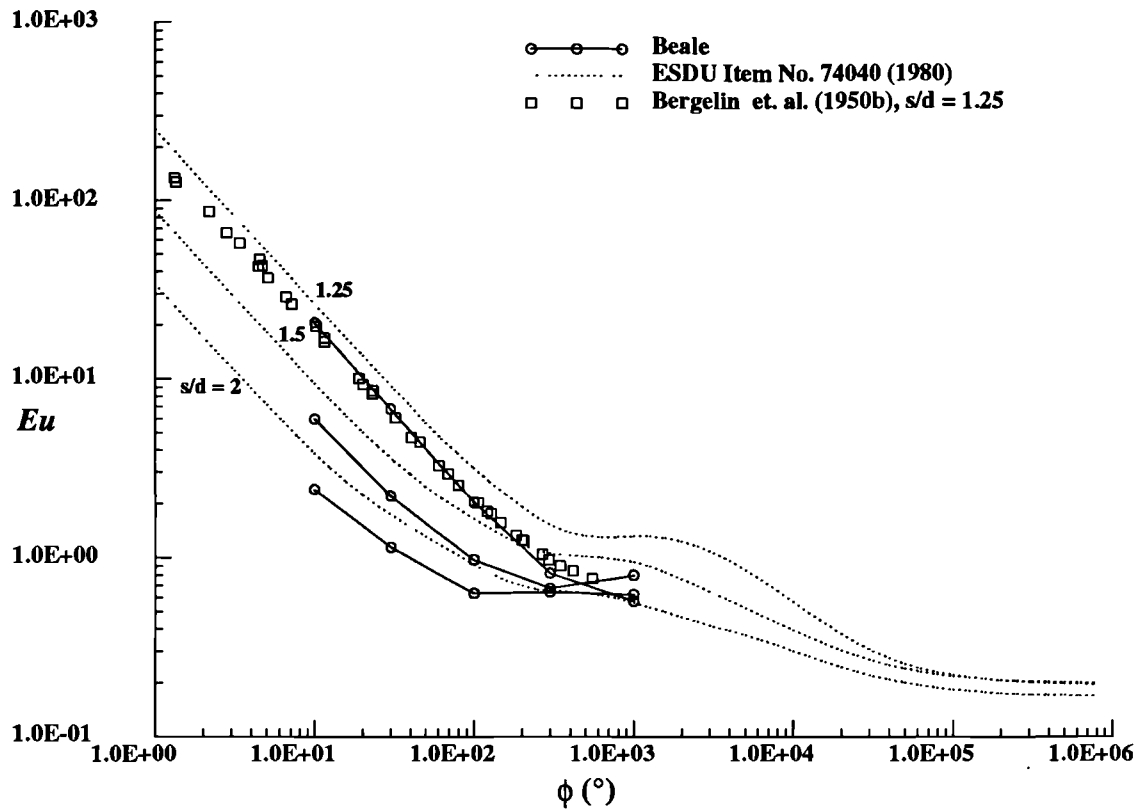


Figure 53. Eu vs. Re compared to experimental data, rotated square.

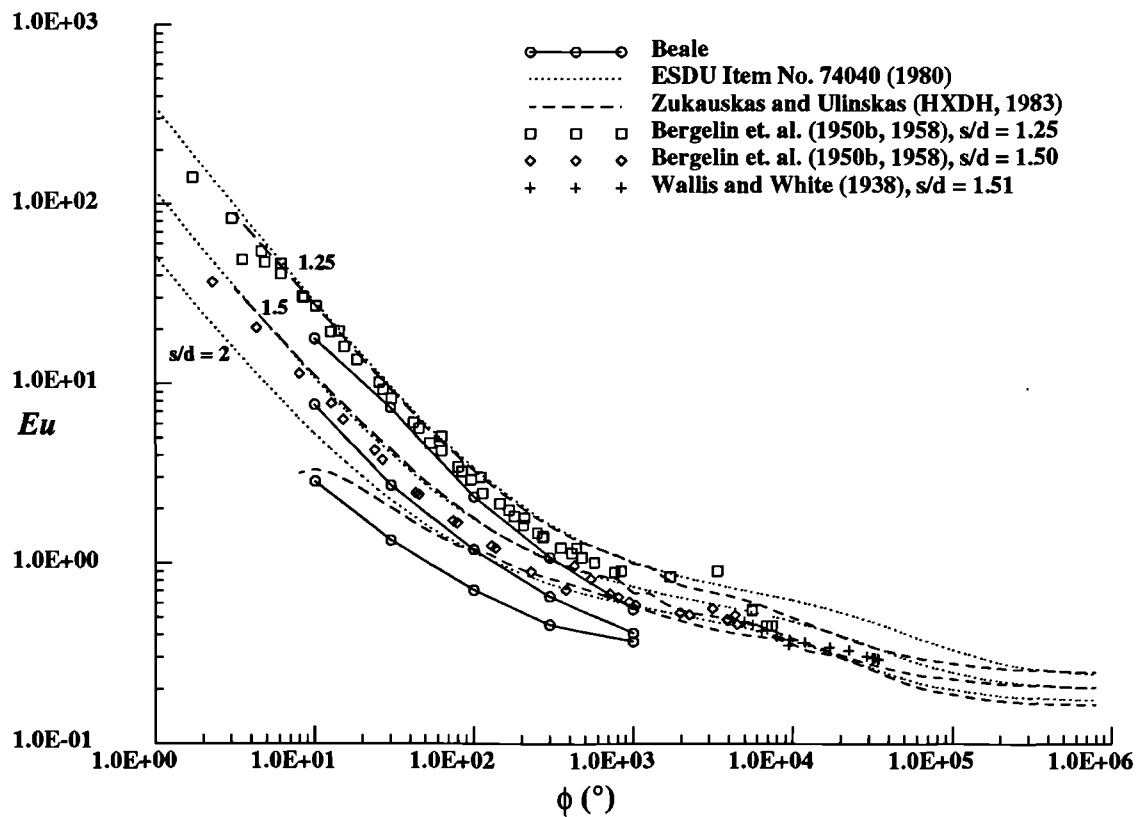


Figure 54. Eu vs. Re compared to experimental data, equilateral triangle.

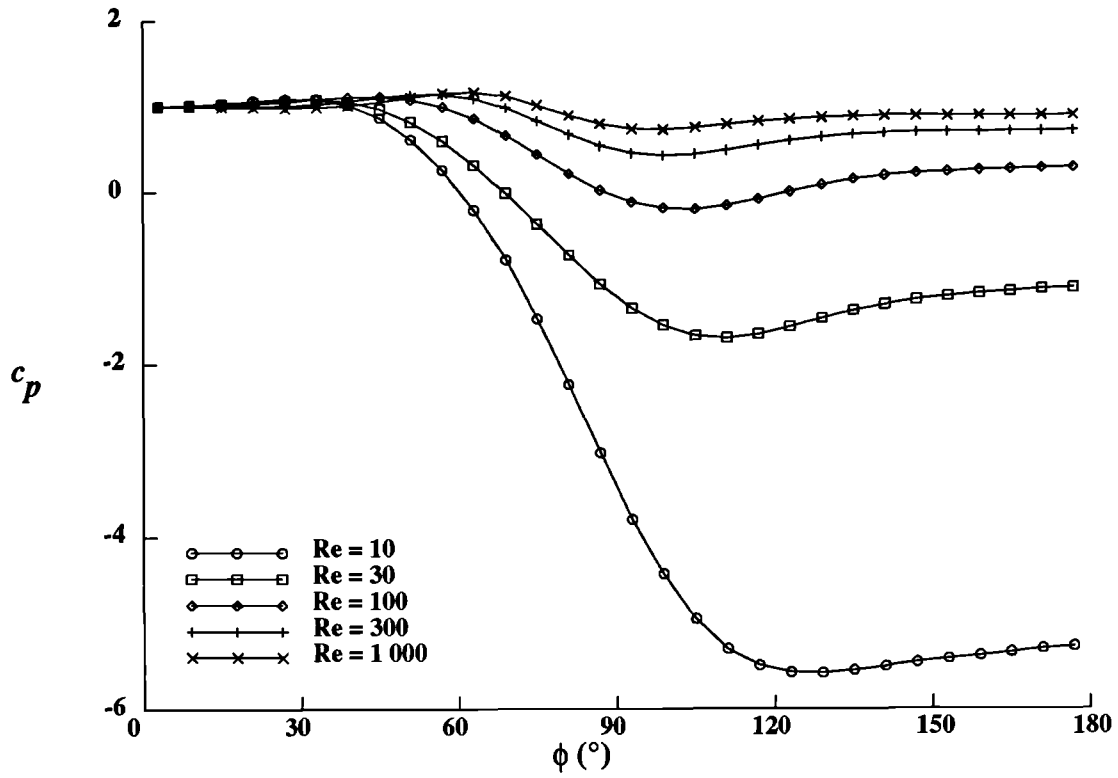


Figure 55. c_p vs. ϕ , in-line square, $s/d = 1.5$.

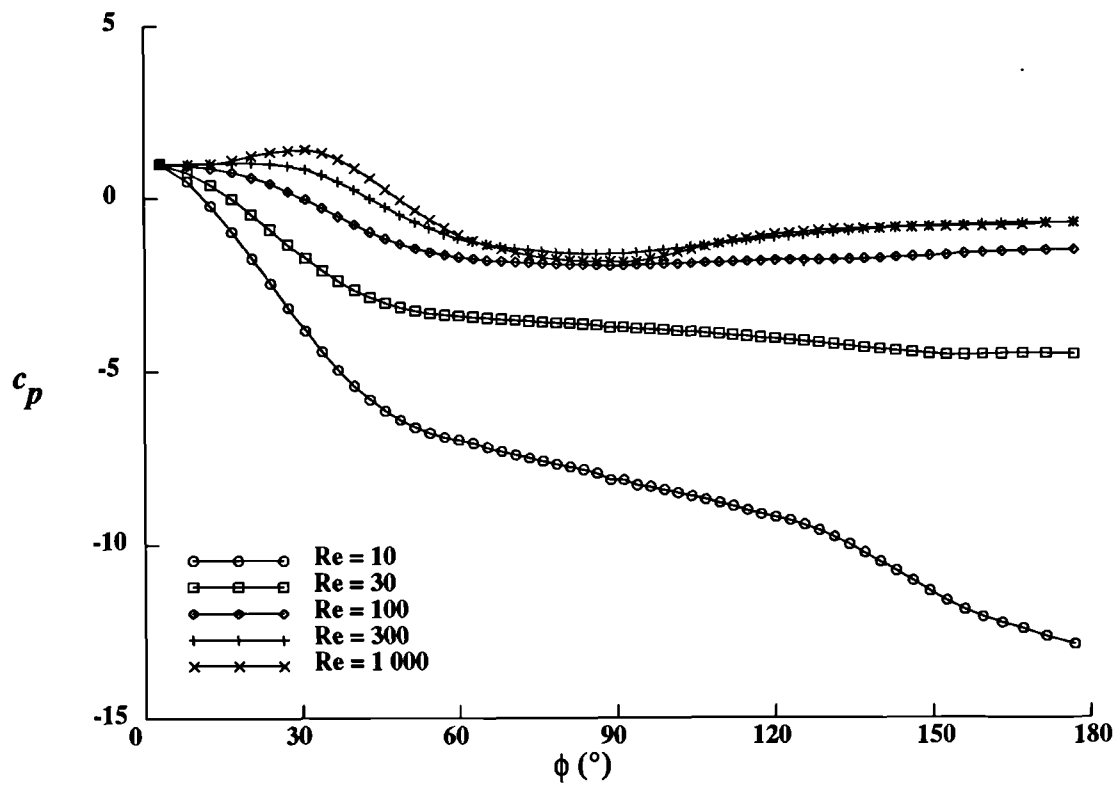


Figure 56. c_p vs. ϕ , rotated square, $s/d = 1.5$.

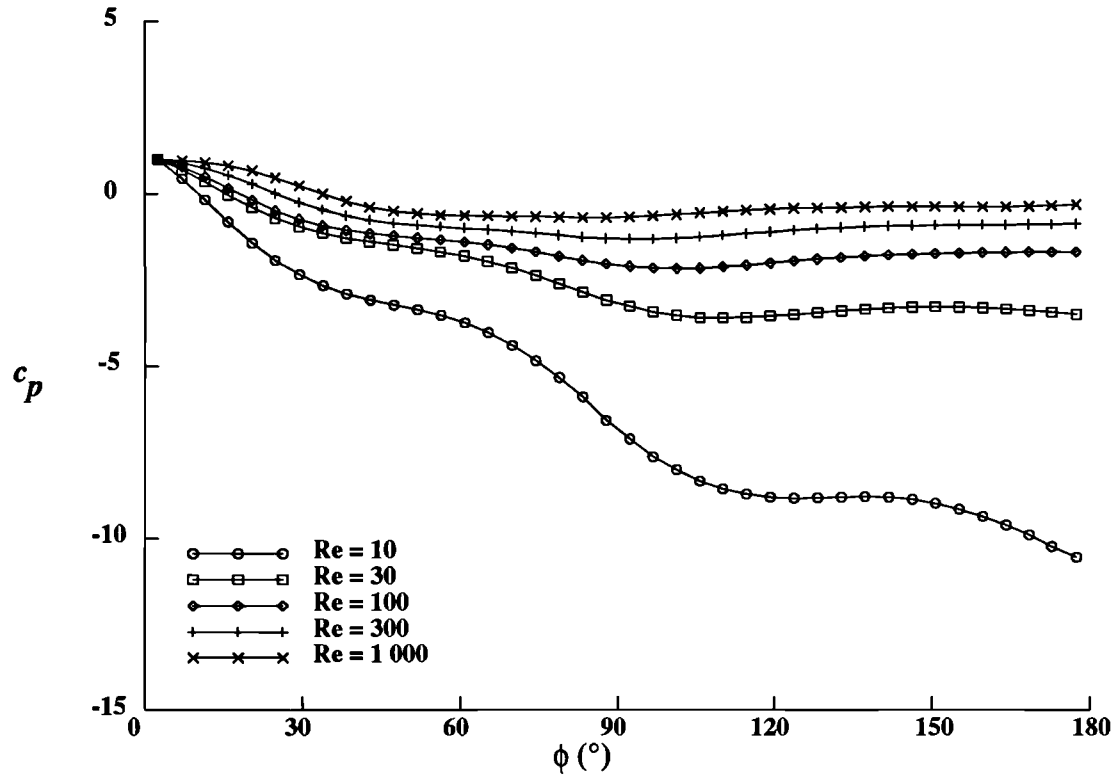


Figure 57. c_p vs. ϕ , equilateral triangle $s/d = 1.5$.

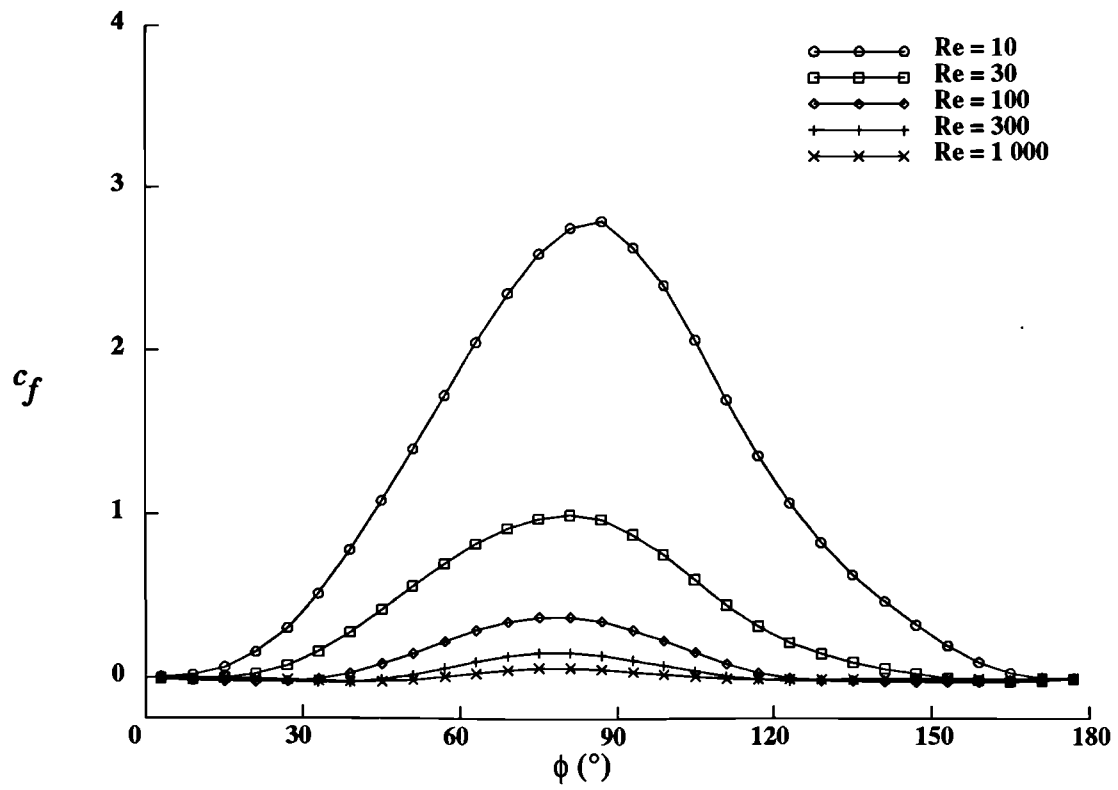


Figure 58. c_f vs. ϕ , in-line square, $s/d = 1.5$.

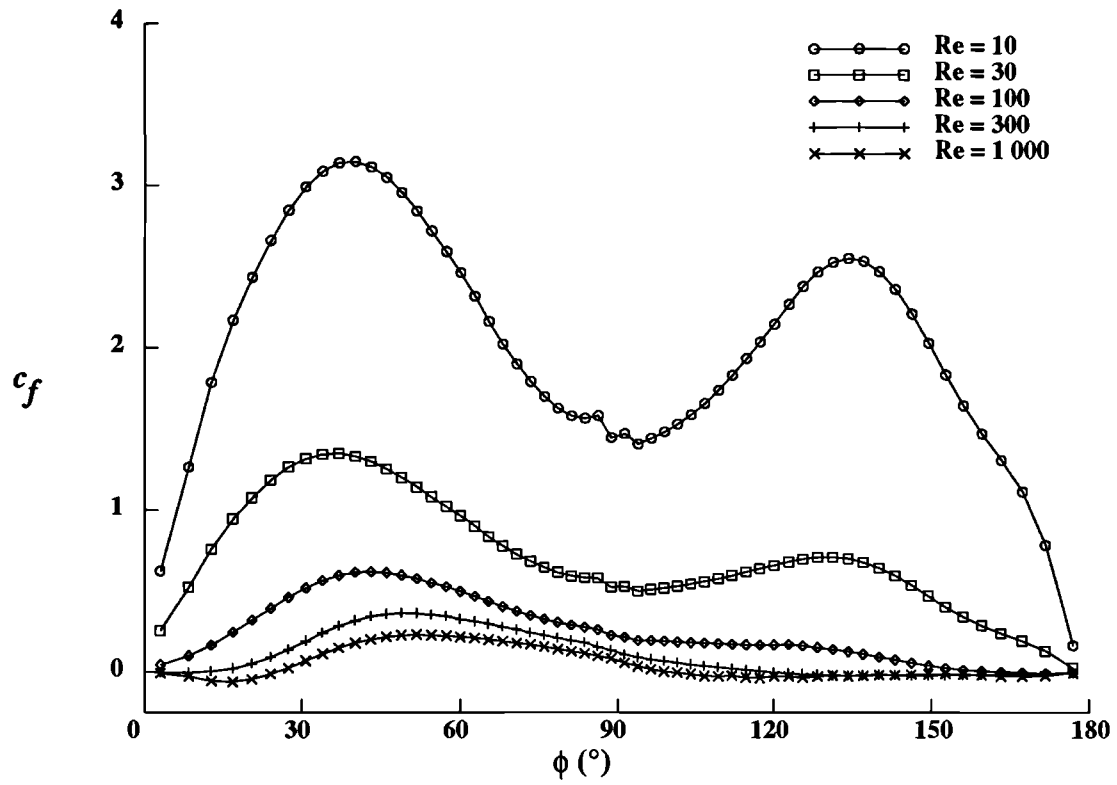


Figure 59. c_f vs. ϕ , rotated square, $s/d = 1.5$.

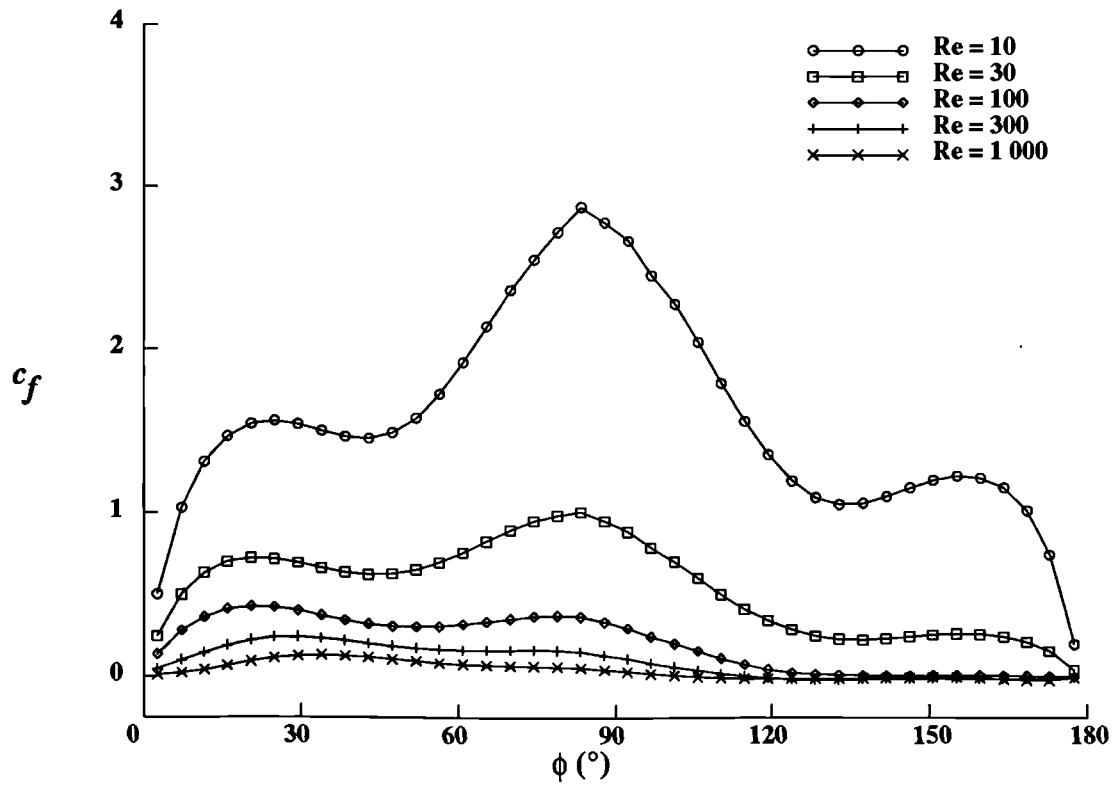


Figure 60. c_f vs. ϕ , equilateral triangle, $s/d = 1.5$.

8.2.4 Overall heat transfer

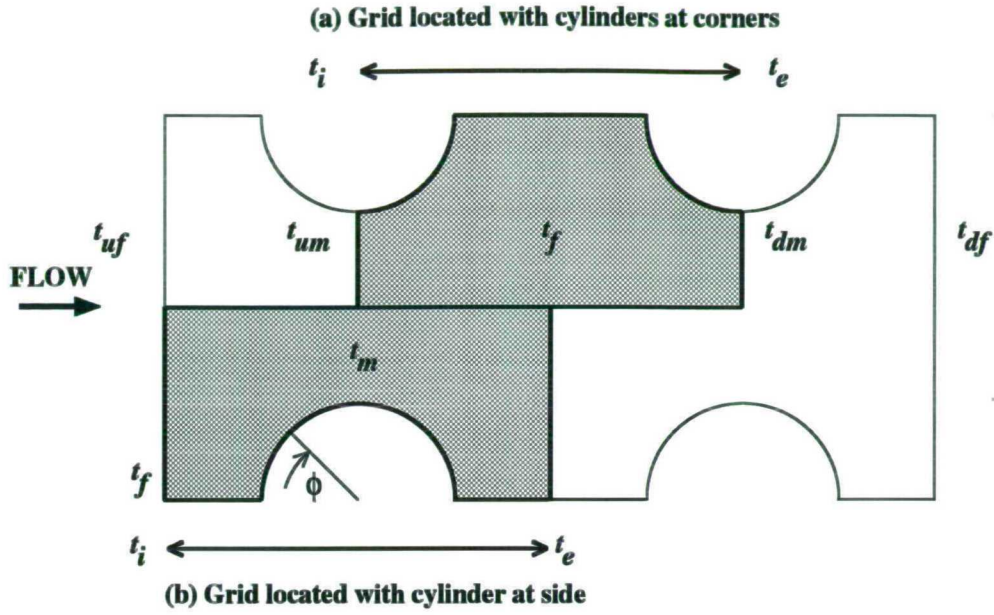


Figure 61. Reference temperatures.

The overall Nusselt Number, $\overline{Nu}$, is defined by,

$$\overline{Nu} \equiv \frac{\overline{h}d}{k} \quad (8.21)$$

where d is the cylinder diameter, k is the fluid thermal conductivity, and $\overline{h}$ the average heat transfer is given by,

$$\overline{h} \equiv \frac{Q}{\frac{d}{2} \int_0^\pi (t_w - t_{ref}) d\phi} \quad (8.22)$$

where Q is the total heat transfer, t_w is the wall temperature, and t_{ref} is a reference fluid temperature. For constant t_w , the total heat transfer, Q , is computed from the wall flux,

q_w . For constant wall flux, it is the wall temperature which requires to be computed as a function of ϕ .

Different authors have selected different values of t_{ref} upon which to base their definitions. The reader will note that the grid may be such that the cylinders are located at (a) the corners or (b) the side (or centre) of the grid [Chapter 6]. Consider the latter case [Fig. 61b]. The reference fluid temperature, t_{ref} may be chosen as either (i) the inter-tube bulk temperature upstream of the cylinder, t_f , (i.e. midway between the row under consideration and the previous row), or (ii) as t_m , the bulk temperature at $\phi = 90^\circ$, corresponding to the minimum cross-section¹ for the row under consideration. The former is preferred here. In order that these definitions be grid-independent for the more usual case (a) when the cylinders are located at the domain corners [Fig. 61a], t_{ref} is set to t_{uf} for $\phi < 90^\circ$, and t_f for $\phi > 90^\circ$ [Fig. 61a] (N.B. t_{uf} must be extrapolated). For all in-line banks, Nusselt numbers based on t_f will henceforth be denoted by $\overline{Nu}_T$ or $\overline{Nu}_H$, depending upon whether constant q_w or t_w boundary conditions are prescribed.

For staggered tube banks, t_f is not usually available, owing to the grid topology. Under these circumstances, with reference to Fig. 61a, it is assumed that, t_f may be approximated by, t_f' where,

$$t_f' \equiv \frac{t_{um} + t_{dm}}{2} \quad (8.23)$$

For all staggered banks Nusselt numbers referred to as $\overline{Nu}_T$ or $\overline{Nu}_H$ are based on the appropriate upstream value of t_f' .

Another possible definition used, for example by Massey (1976) for case (a) above [Fig. 61a], is to set t_{ref} equal to the upstream value t_{um} for all values of ϕ . Let this

¹ Except for rotated square banks with $s/d < 1.707$, where u_m is the bulk velocity across the diagonal. However t_m is always computed at $\phi = 90^\circ$.

selection be denoted by $\overline{Nu}_{um}$ etc. Yet another possibility, for constant t_w , is to base values on the log-mean temperature difference,

$$\Delta t_{LM} \equiv \frac{t_e - t_i}{\log_e \left(\frac{t_i - t_w}{t_e - t_w} \right)} \quad (8.24)$$

where, t_i and t_e are the bulk inlet and exit temperatures (regardless of the cylinder location). It can be shown that for in-line and widely spaced staggered banks, $\overline{Nu}_{LM}$ may be computed from an overall energy balance as,

$$\overline{Nu}_{LM} = Re Pr^{\frac{2(a-1)}{\pi}} \log \left(\frac{t_w - t_i}{t_w - t_e} \right) \quad (8.25)$$

The local wall temperature being computed from the interior values using the known wall gradient. $\overline{Nu}_{LM}$ may also be computed from the known wall heat flux, by extrapolating a reference temperature using Eq. 8.24.

An overall Stanton number, $\overline{St}$, may be defined by,

$$\overline{St} \equiv \frac{\bar{h}}{\rho c_p u_m} \quad (8.26)$$

In an attempt to eliminate the Pr dependence, a Colburn heat transfer factor, j , is sometimes defined by,

$$j \equiv \overline{St} Pr^{\frac{2}{3}} \quad (8.27)$$

Bergelin et. al. (1950b) modified the heat transfer factor, j' , according to,

$$j' \equiv j \left(\frac{\mu_w}{\mu} \right)^{0.14} \quad (8.28)$$

to account for variations in the wall viscosity. Alternatively,

$$B \equiv \overline{Nu} Pr^{-\frac{1}{3}} \left(\frac{\mu_w}{\mu} \right)^{0.14} \quad (8.29)$$

is a similarly defined modified Nusselt number. Zukauskas (1968) proposed instead the use of,

$$K \equiv \overline{Nu} Pr^{-0.36} \left(\frac{Pr_f}{Pr_w} \right)^{-.25} \quad (8.30)$$

in the place of Eq. 8.28. Heat transfer results are given below; where comparisons with others are made, the appropriate definitions are wherever possible applied.

Figure 62 shows the author's overall heat transfer data for in-line banks $s/d = 1.25$ and 2, constant t_w , $Pr = 1$, compared to those of Le Feuvre (1973). The data are plotted in terms $\overline{Nu}_{LM}$ based on the log-mean temperature difference vs. the Peclet number, Pe , (i.e. the product $Re \cdot Pr$). Figure 63 is a similar comparison of this writer's data for in-line tube banks, $s/d = 1.25$ and 2, constant q_w , at various Re and Pr , with the work of Antonopoulos (1979). The constant t_w data of Bergelin et. al. (1950b, 1958) are also shown. Figure 64 is a comparison for rotated square banks, $s/d = 1.25$, with Massey's (1976) and Antonopoulos' (1979) data.

Figure 65 is a plot of $\overline{St}_T$ and $\overline{St}_H$ vs. Re for in-line tube banks, $s/d = 1.25, 1.5, 2$, and for $Pr = 1$, compared to the empirical correlation of Zukauskas and Ulinskas (1988a).

Figure 66 shows $\overline{St}_{LM}$ vs. Re compared to Zukauskas and Ulinskas' correlation, for rotated square and equilateral triangle banks, $s/d = 1.25, 1.5, 2$, $Pr = 1$. Figure 67 shows Nu vs. Re for an in-line tube bank, $s/d = 1.5$, $Pr = 1$. Nu shown are based on t_f , t_m , and Δt_{LM} . In the latter case two values are shown, computed from (i) the wall heat flux and (ii) using an overall energy balance [Eq. 8.25].

Figures 68 and 69 are a comparison of some of the experimental data and empirical correlations which are available. The j'_{LM} data of Bergelin et. al. (1950b, 1958) is shown together with the j'_{LM} correlation of ESDU (1973), the K_{LM} correlation of Gnielinski et. al. (HXDH, 1983), and also the K_f correlation of Zukauskas and Ulinskas (1968, 1988a) (also Zukauskas, 1987).

Tables 4 to 12 inclusive, are a summary of the results of this Chapter. Values of Eu are shown for in-line square, rotated square and equilateral triangle tube banks, $s/d = 1.25, 1.5$, and 2 . Values of $\overline{Nu}_{LM}$, (based on an overall heat balance), for constant t_w are also displayed as are $\overline{Nu}_T$ and $\overline{Nu}_T$. All heat transfer coefficients are for $Pr = 1$.

8.2.5 Local heat transfer

The local Nusselt Number, Nu , is defined as,

$$Nu \equiv \frac{hd}{k} \quad (8.31)$$

where the local heat transfer coefficient, h , is given by,

$$h \equiv \frac{q_w}{(t_w - t_{ref})} \quad (8.32)$$

Figures 70 to 75, inclusive show normalised values of Nu as a function of ϕ at various Re , for $Pr = 1$, under both constant heat flux and constant temperature boundary conditions. Figure 76 is a similar plot for a compact rotated square bank, constant t_w , $s/d = 1.25$, $Pr = 1$, Nu_T and Nu_H are again based on t_f (in-line) and t_f' (staggered). Figure 77 shows the Pr dependence for an in-line tube bank, $s/d = 1.5$, $1 \leq Pr \leq 100$

Table 4. Summary of results, in-line square, $s/d = 1.25$, $Pr = 1$.

Re	Eu	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	16.4	4.65	1.30	1.83
30	5.39	6.71	2.44	2.96
100	1.75	7.06	4.66	4.18
300	0.730	7.74	6.57	5.77
1 000	0.287	10.5	9.83	9.42

Table 5. Summary of results, in-line square, $s/d = 1.5$, $Pr = 1$.

Re	Eu	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	6.05	4.11	1.75	2.19
30	2.06	4.71	3.02	3.15
100	0.716	4.85	4.15	3.88
300	0.287	5.47	5.15	4.99
1 000	0.111	7.35	7.18	7.81

Table 6. Summary of results, in-line square, $s/d = 2$, $Pr = 1$.

Re	Eu	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	2.41	3.16	2.06	2.26
30	0.867	3.28	2.76	2.84
100	0.297	3.45	3.27	3.26
300	0.119	3.85	3.78	3.86
1 000	0.0498	5.01	4.97	5.17

Table 7. Summary of results, rotated square, $s/d = 1.25$, $Pr = 1$.

<i>Re</i>	<i>Eu</i>	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	20.7	12.3	1.81	2.55
30	6.85	17.5	4.58	12.4
100	2.05	19.6	9.23	11.8
300	0.824	21.7	12.6	13.8
1 000	0.574	33.8	21.1	24.5

Table 8. Summary of results, rotated square, $s/d = 1.5$, $Pr = 1$.

<i>Re</i>	<i>Eu</i>	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	6.00	7.07	2.73	4.83
30	2.22	8.44	5.22	6.41
100	0.981	9.39	7.35	7.84
300	0.680	11.8	9.90	9.65
1 000	0.804	26.0	21.7	24.2

Table 9. Summary of results, rotated square, $s/d = 2$, $Pr = 1$.

<i>Re</i>	<i>Eu</i>	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	2.41	4.35	3.28	3.99
30	1.15	5.11	4.61	5.15
100	0.636	6.20	5.99	6.16
300	0.647	11.9	11.7	12.7
1 000	0.627	25.1	24.7	26.6

Table 10. Summary of results, equilateral triangle, $s/d = 1.25$, $Pr = 1$.

Re	Eu	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	17.8	6.94	0.135	2.67
30	7.37	11.6	0.962	6.62
100	2.35	14.2	5.87	11.5
300	1.08	16.9	12.4	17.8
1 000	0.556	22.8	19.7	23.9

Table 11. Summary of results, equilateral triangle, $s/d = 1.5$, $Pr = 1$.

Re	Eu	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	7.68	5.86	0.842	2.80
30	2.72	7.75	3.47	5.67
100	1.19	9.75	7.65	10.2
300	0.658	12.9	11.6	13.6
1 000	0.409	19.4	18.0	20.2

Table 12. Summary of results, equilateral triangle, $s/d = 2$, $Pr = 1$.

Re	Eu	$\overline{Nu}_{LM}$	$\overline{Nu}_T$	$\overline{Nu}_H$
10	2.87	4.34	2.19	2.93
30	1.34	5.64	4.33	5.31
100	0.711	7.77	6.84	7.89
300	0.457	10.7	9.70	10.6
1 000	0.368	19.9	18.1	19.6

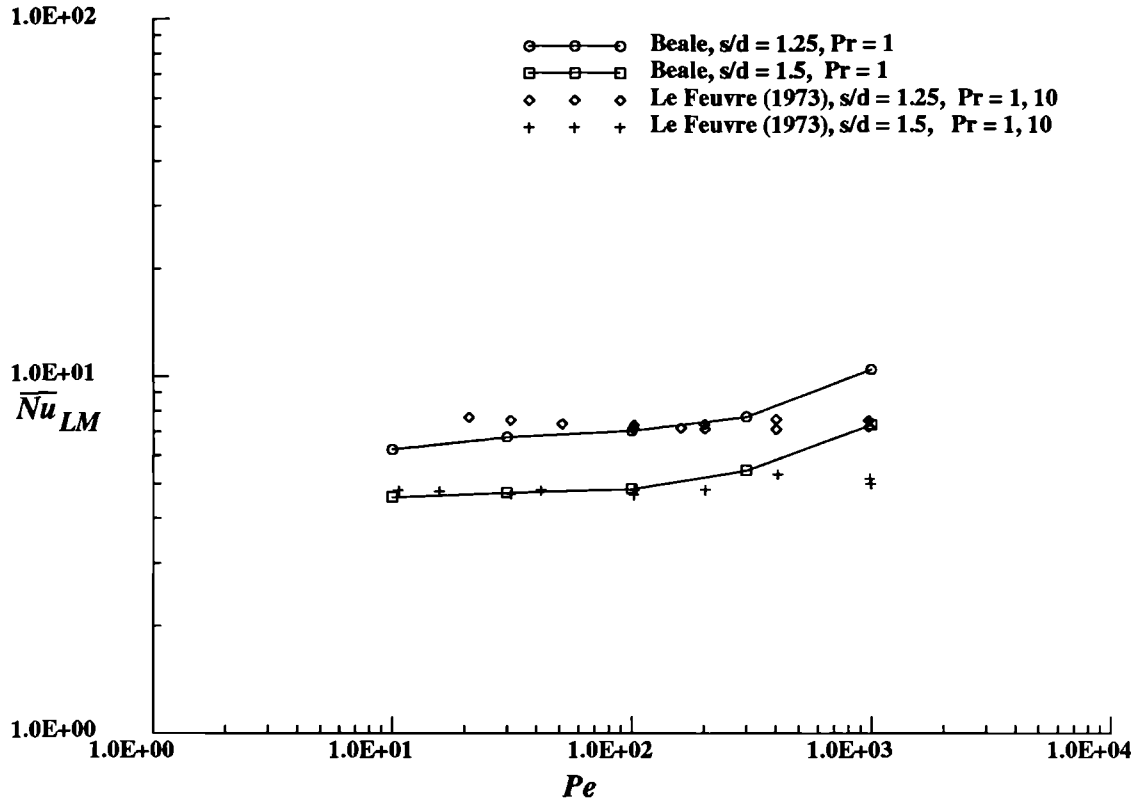


Figure 62. $\overline{Nu}_{LM}$ vs. Pe compared to numerical data, in-line square, constant t_w .

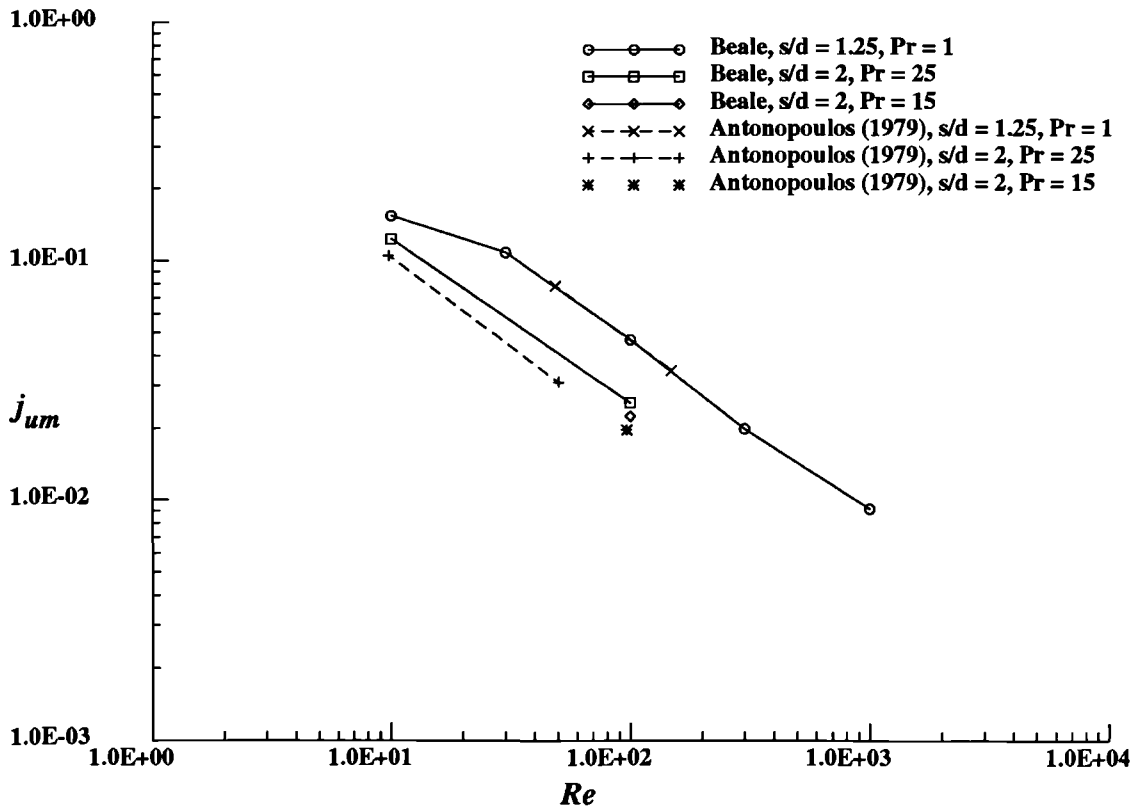


Figure 63. j_{um} vs. Re compared to numerical data in-line, square, constant q_w .

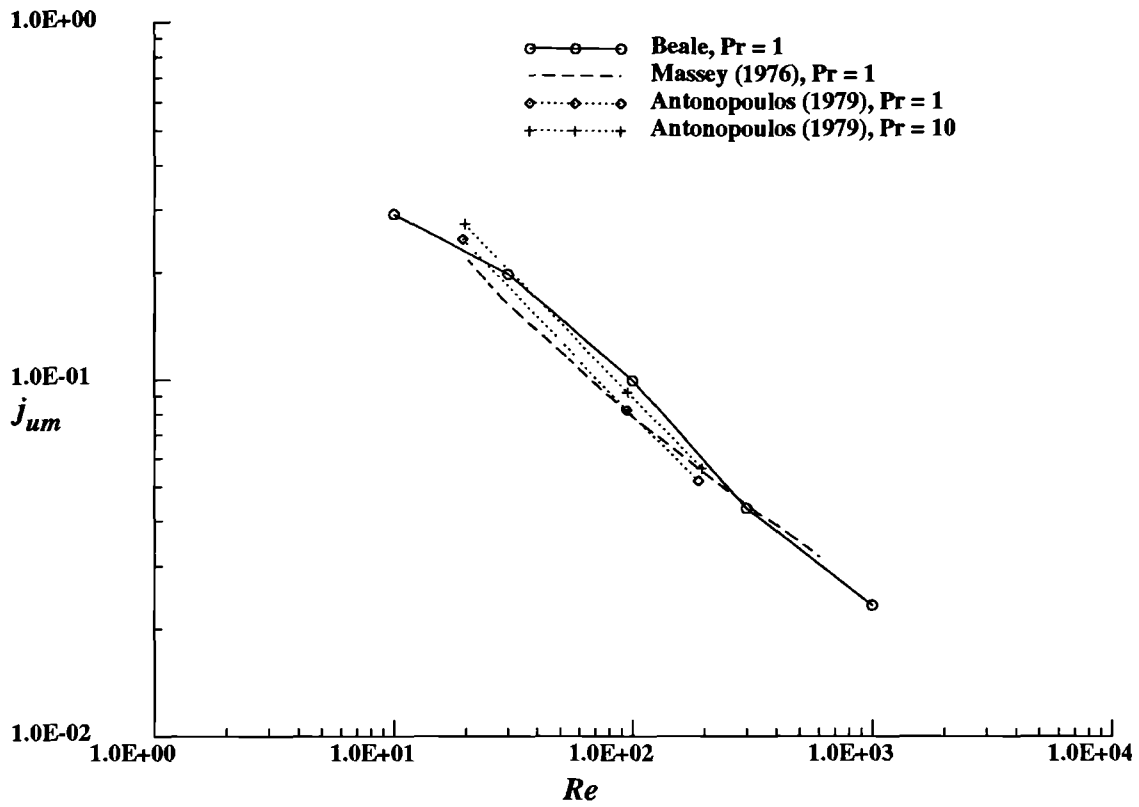


Figure 64. j_{um} vs. Re compared to numerical data, rotated square, constant q_w .

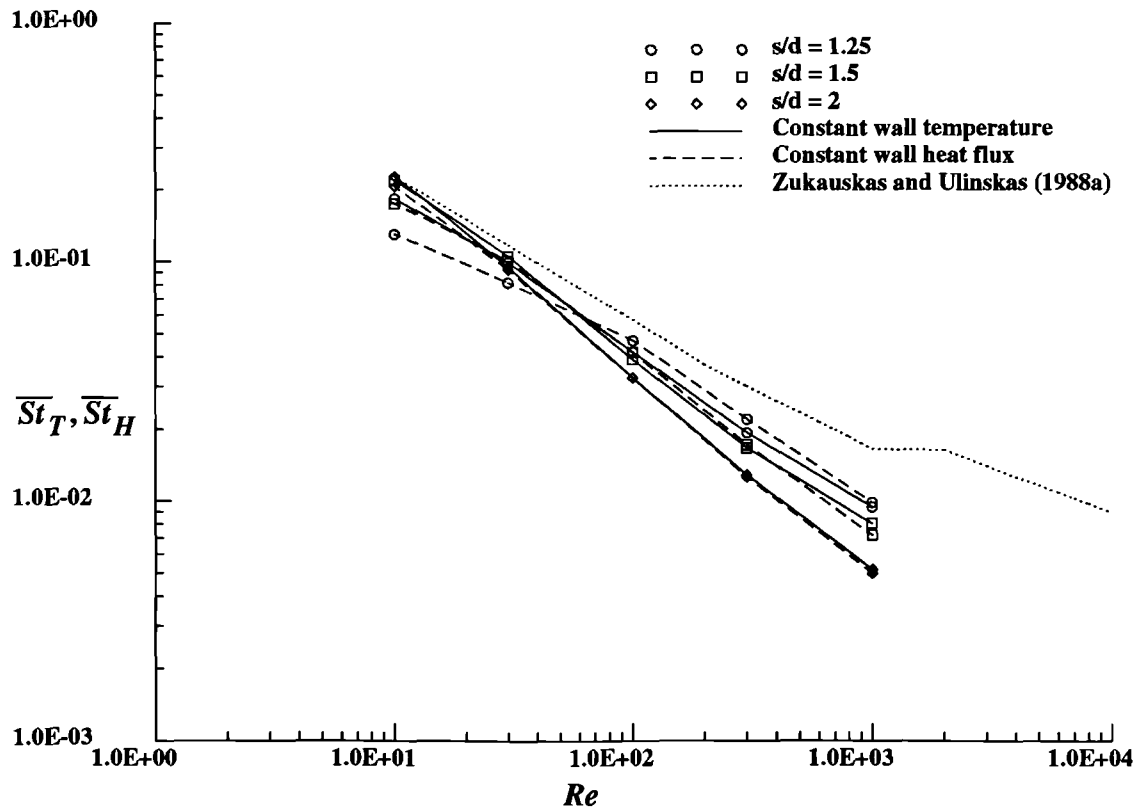


Figure 65. Effect of wall boundary conditions on $\overline{St}$, $Pr = 1$.

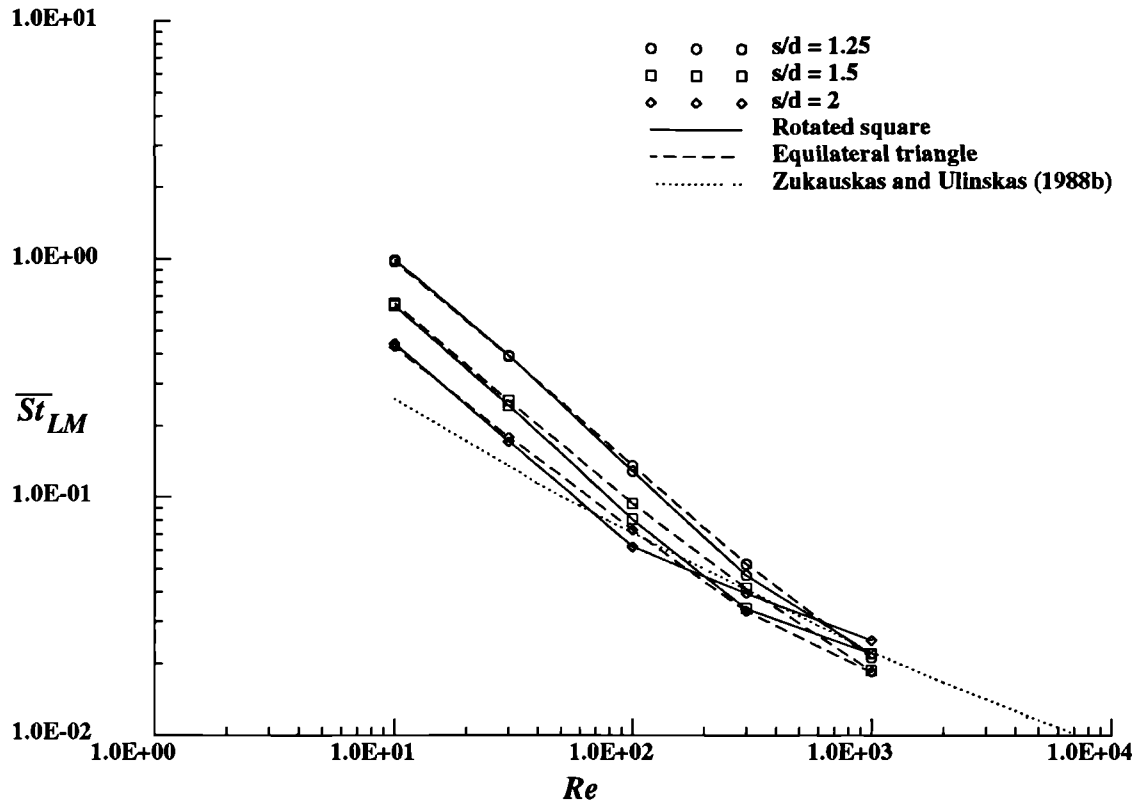


Figure 66. $\overline{St}_{LM}$ vs. Re , staggered banks, constant t_w , $Pr = 1$.

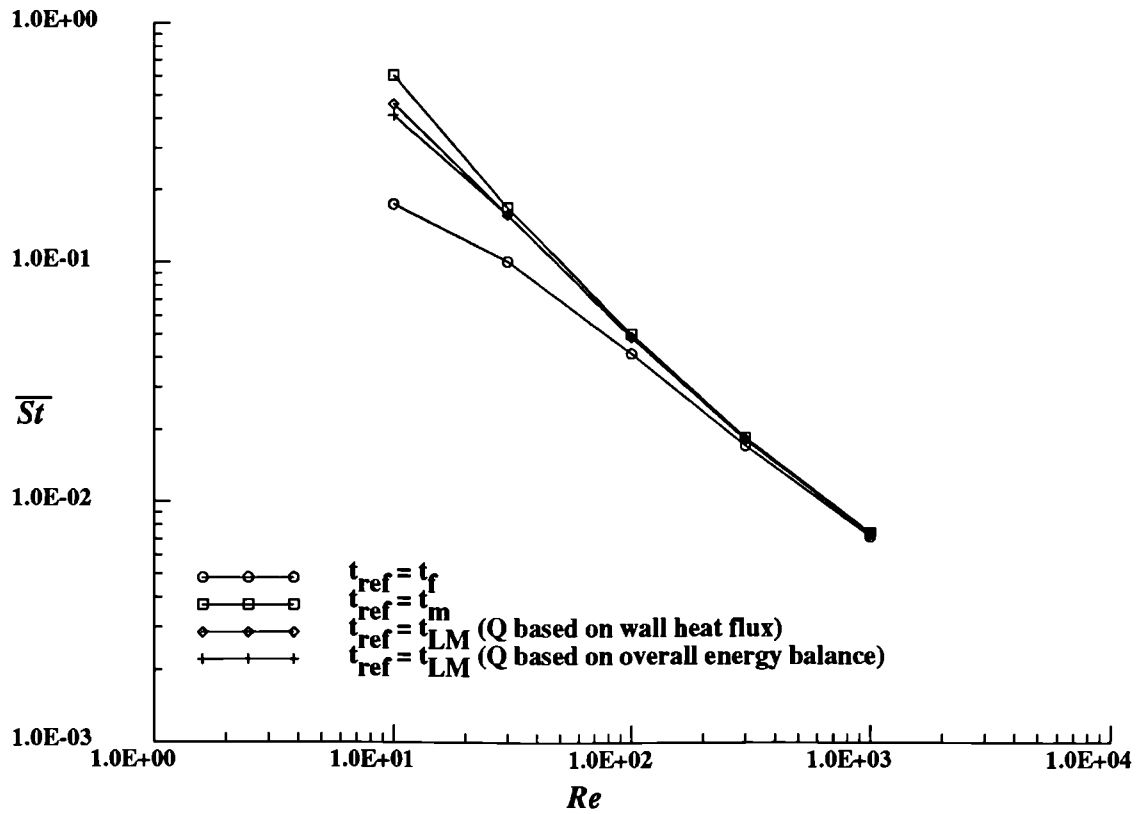


Figure 67. Effect of t_{ref} on $\overline{St}$, in-line square, $s/d = 1.5$, $Pr = 1$.

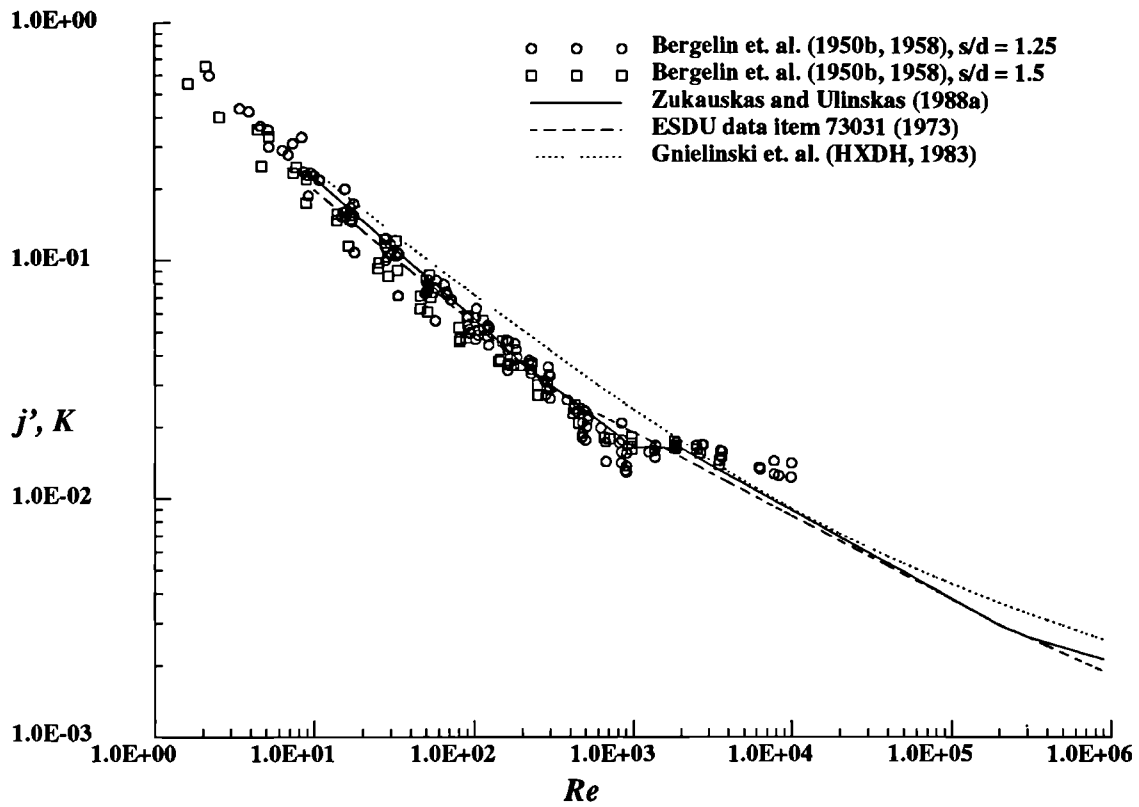


Figure 68. Comparison of experimental data and correlations, in-line banks.

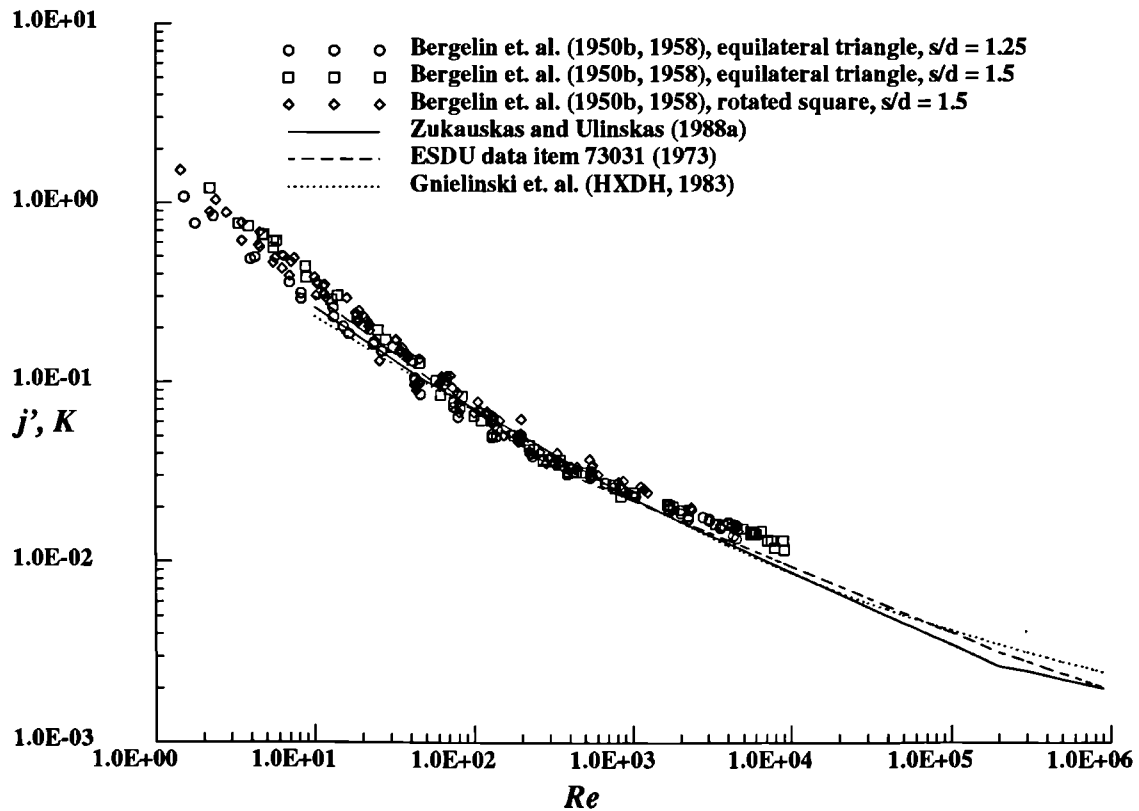


Figure 69. Comparison of experimental data and correlations, staggered banks.

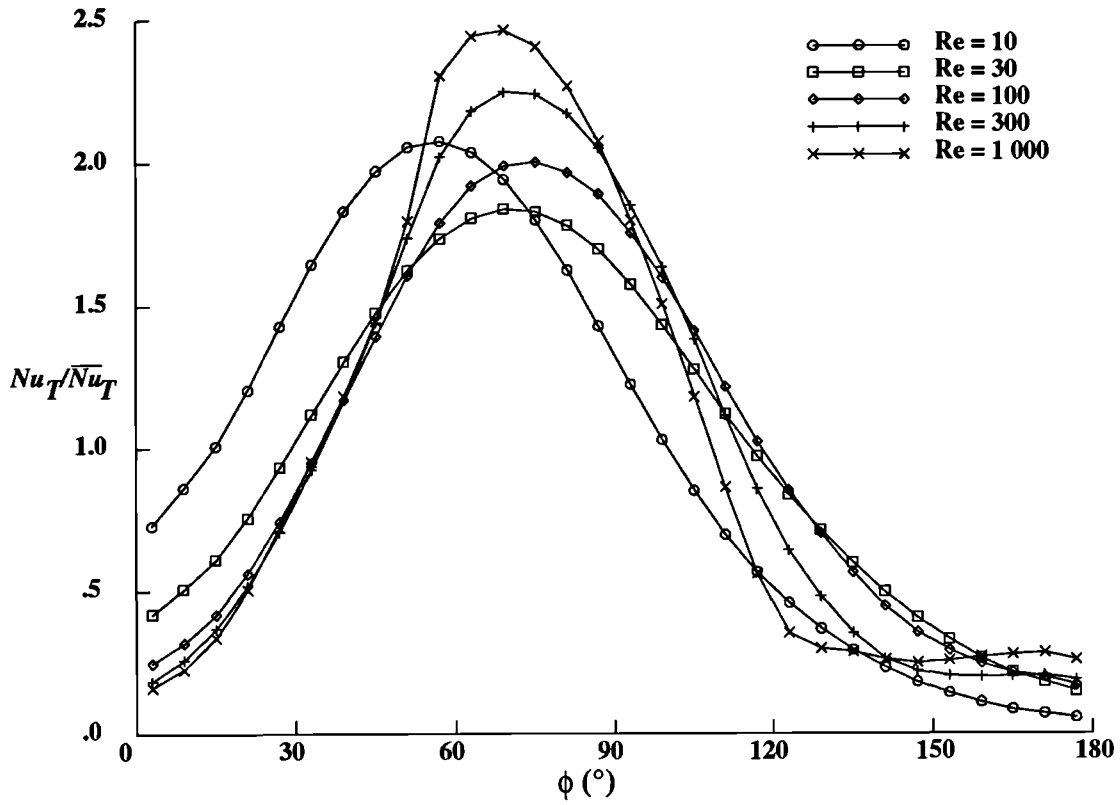


Figure 70. $Nu_T / \overline{Nu}_T$ vs. ϕ , in-line square, $s/d = 1.5$, $Pr = 1$.

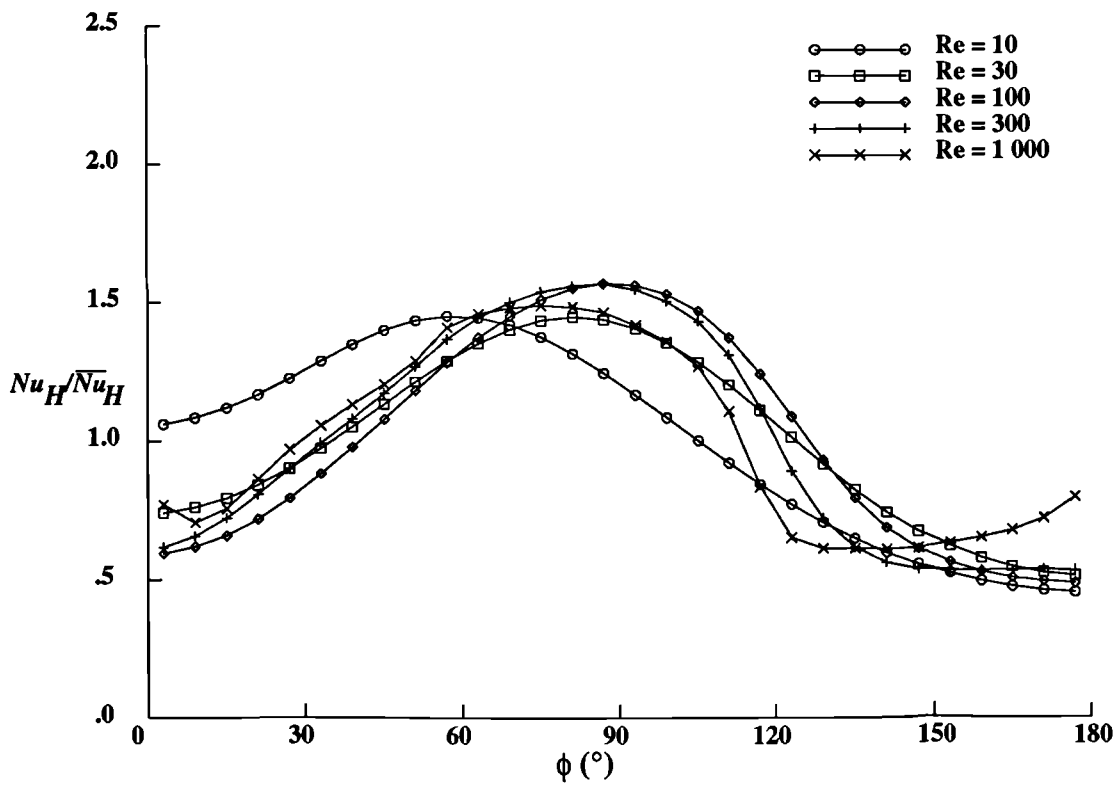


Figure 71. $Nu_H / \overline{Nu}_H$ vs. ϕ , in-line square $s/d = 1.5$, $Pr = 1$.

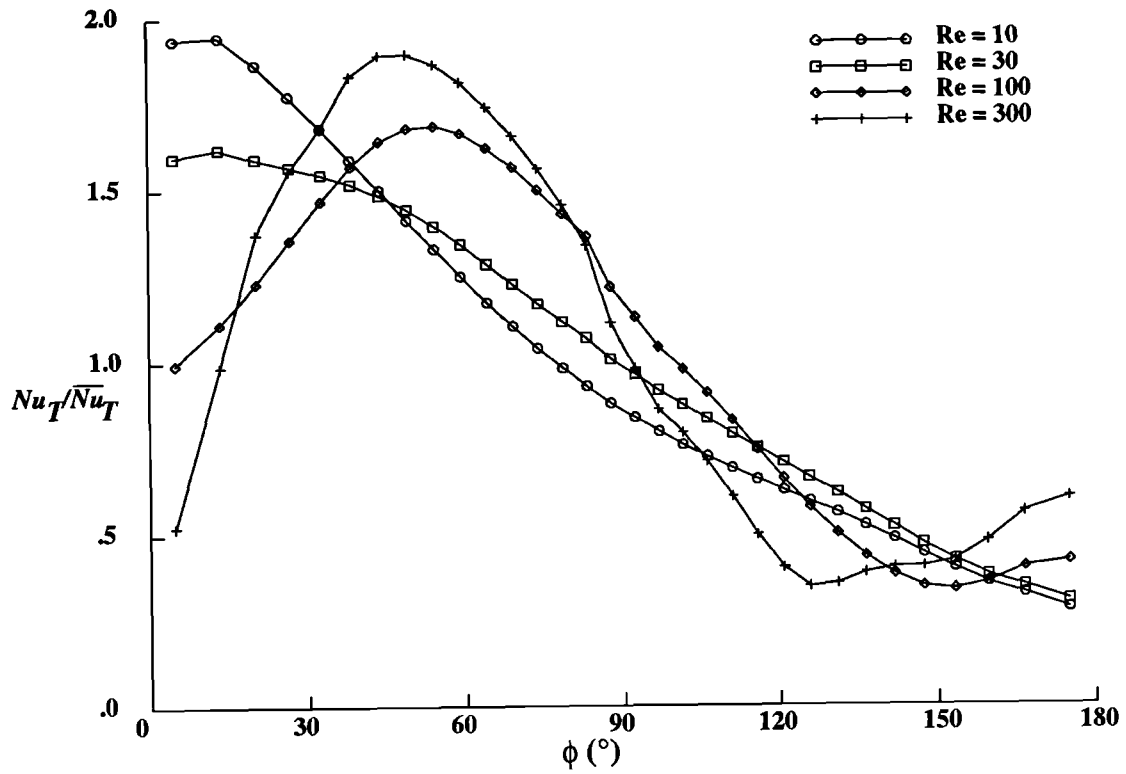


Figure 72. $Nu_T / \overline{Nu}_T$ vs. ϕ , rotated square, $s/d = 2$, $Pr = 1$.

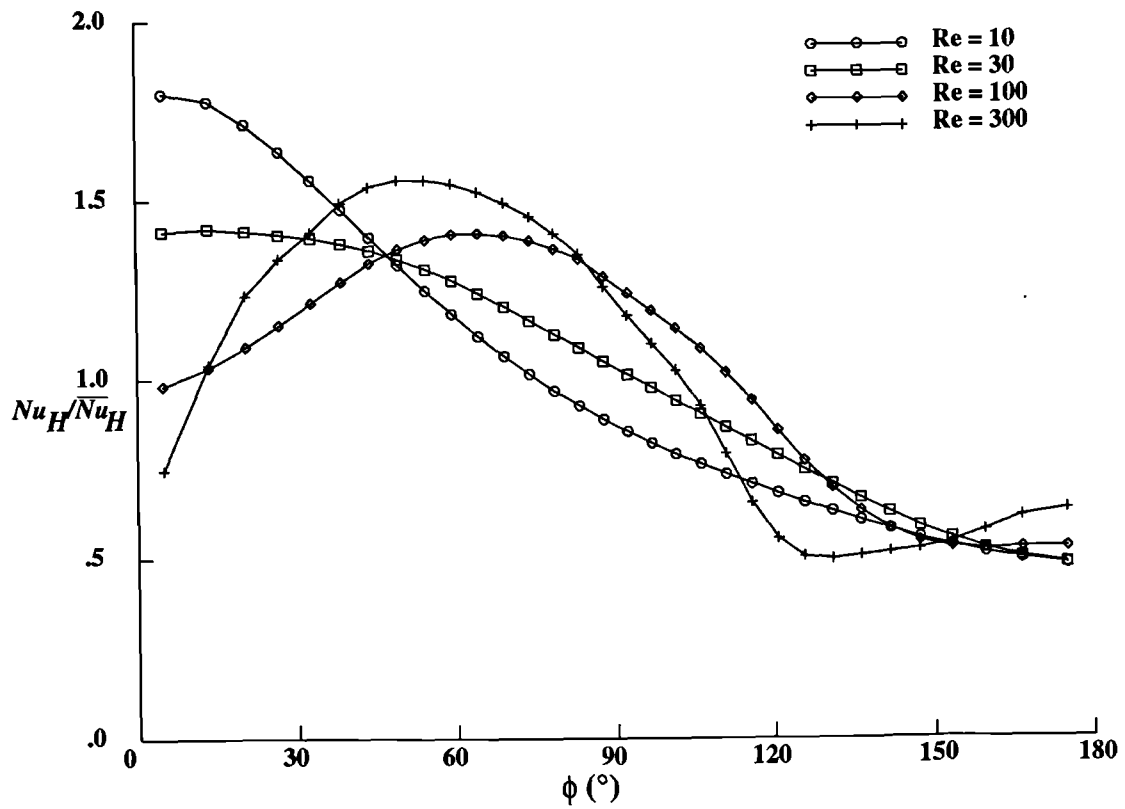


Figure 73. $Nu_H / \overline{Nu}_H$ vs. ϕ , rotated square, $s/d = 2$, $Pr = 1$.

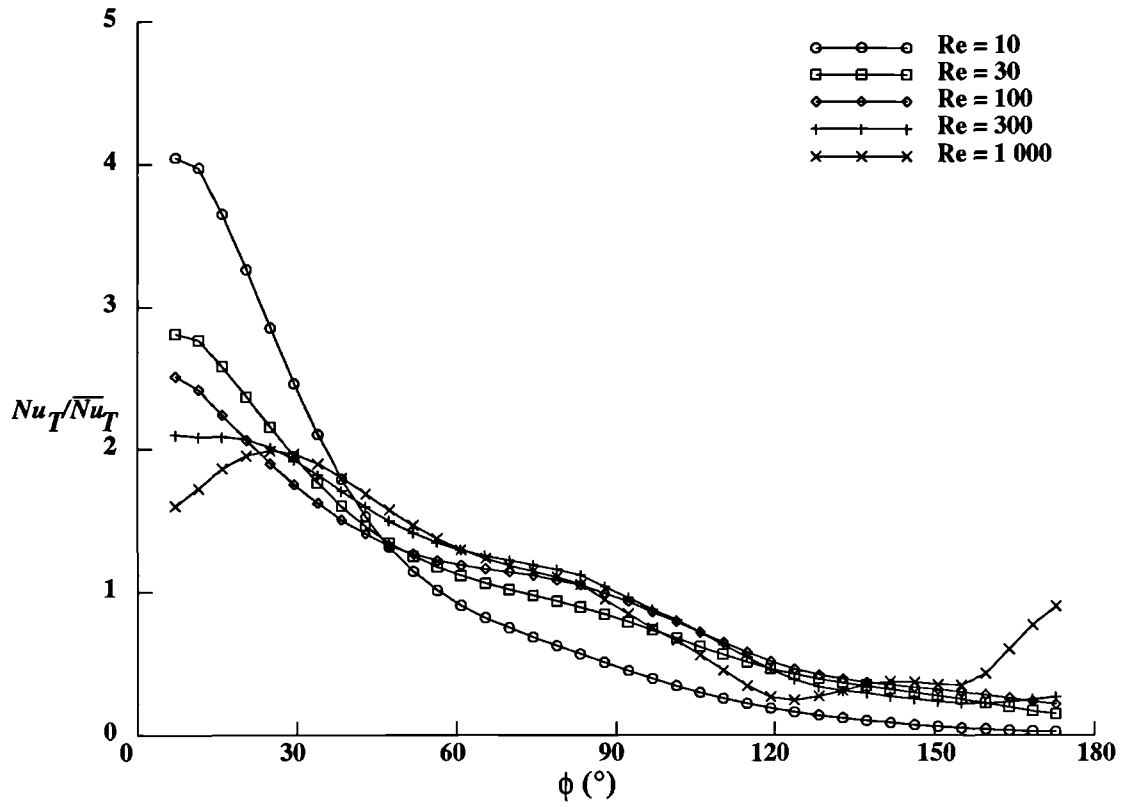


Figure 74. $Nu_T / \overline{Nu}_T$ vs. ϕ , equilateral triangle, $s/d = 1.5$, $Pr = 1$.

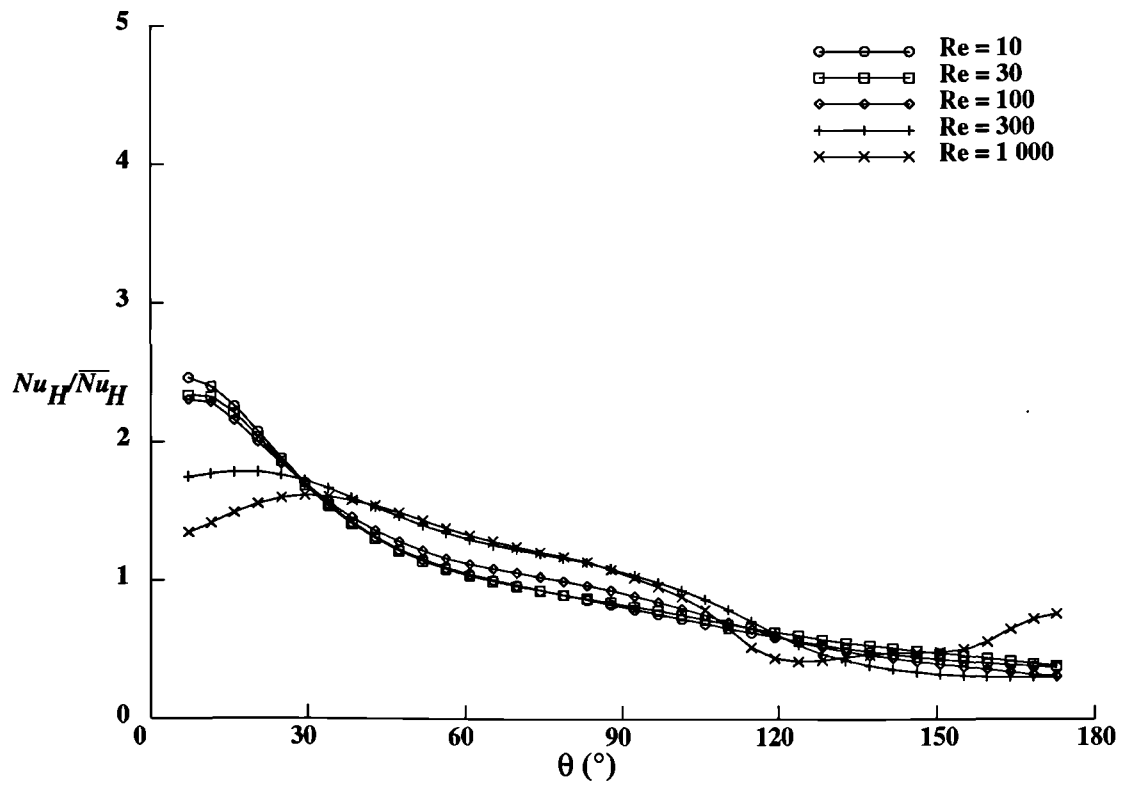


Figure 75. $Nu_H / \overline{Nu}_H$ vs. ϕ , equilateral triangle, $s/d = 1.5$, $Pr = 1$.

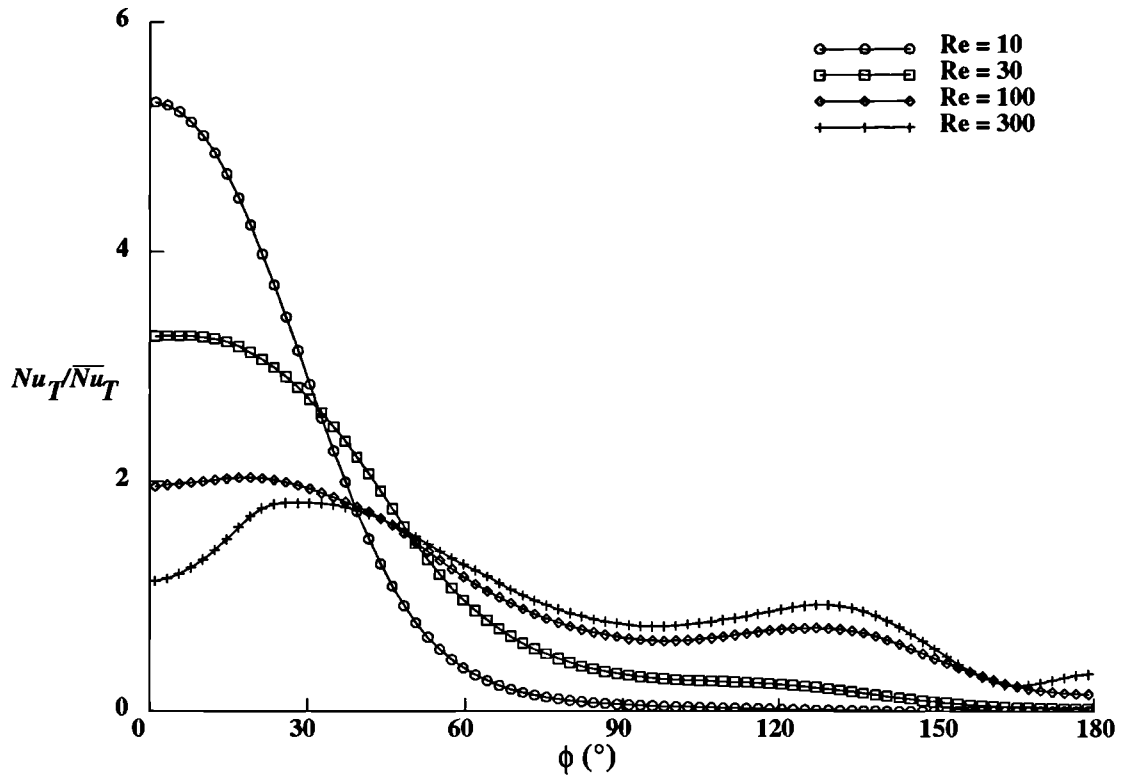


Figure 76. $Nu_T / \bar{Nu}_T$ vs. ϕ , rotated square, $s/d = 1.25$, $Pr = 1$.

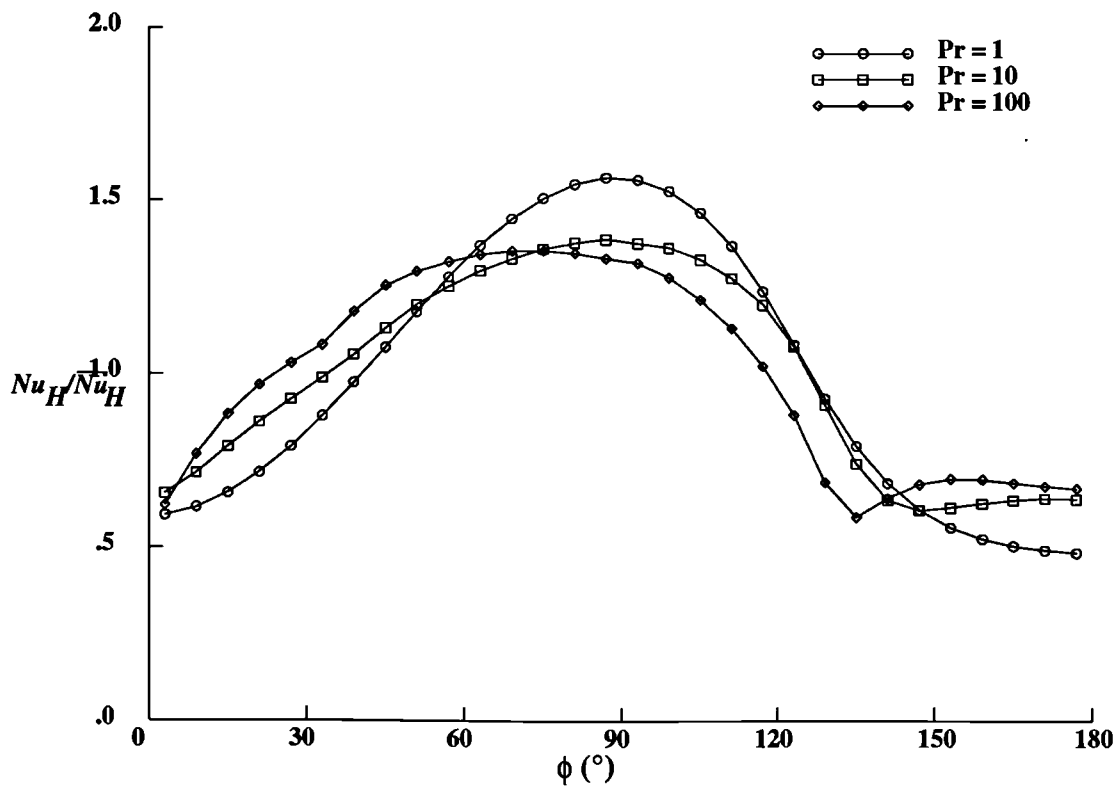


Figure 77. $Nu_H / \bar{Nu}_H$ vs. ϕ , in-line square, $s/d = 1.5$, $Re = 100$, various Pr .

8.3 Discussion

8.3.1 Qualitative description

At low Re , a symmetric streaming flow pattern is observed for all geometries. As Re increases [Figs. 40 and 41], separation of the flow occurs, and a vortex pair develops in the wake, the usual explanation being that as a result of fluid friction, there is insufficient energy for the flow to overcome the adverse pressure gradient occurring in the latter part of the cylinder. For flow in in-line tube banks [Fig. 40], each cylinder is shaded by the preceding member, so that re-attachment of the flow occurs away from the centreline. For staggered configurations the flow bifurcates at the front leading-edge of the cylinder, the two streams merging in the wake downstream of the vortex. At low Re , the velocity profile is developed [Fig. 41a], so the velocity field immediately upstream of the cylinder is a maximum along the symmetry-line $y_c = s_1/2$. At higher Re , however, the velocity field exhibits a channel-like profile, with the flow maximum occurring away from the symmetry-line [Fig. 41b]. For all of the tube banks considered, the size of the vortex grows with increasing Re . Regardless of the pitch-to-diameter ratio or Re value, only one vortex pair is ever observed in the wake. There is significant enlargement and elongation of the vortical zone in widely-spaced banks at high Re [Fig. 41c] displacing the attachment point away from $\phi = 0^\circ$. It seems rather unlikely that these high Re results are physically meaningful. A similar behaviour was observed with equilateral triangle tube banks, except that in this case, due to the differing geometry, the vortex expansion tends to occur more so laterally up the side of the side of the cylinder, rather than out back into the wake.

Comparison of the (dimensional) pressure contours [Figs. 42 and 43] and non-dimensional pressure coefficient data with those for potential flow [Figs. 9 and 10] re-

veals that for both in-line and staggered tube banks, the pressure field is, at least, qualitatively similar. For viscous flow in in-line banks, the pressure minimum occurs somewhat downstream of the 90° observed for potential flow. The pressure maximum occurs near the point of flow attachment, rather than at $\phi = 0^\circ$. Staggered banks however, do exhibit a local maximum at the front stagnation point. It can be seen that both in-line and staggered banks also exhibit a secondary local maximum in the mid-stream at the minimum cross-section (corresponding to the saddle-point in the potential-flow pressure field).

Inspection of the non-dimensional temperature contours [Figs. 44 to 47], normalised with respect to the overall bulk temperature rise [Eq. 8.17] reveals that at low Re , the variation in the temperature distribution is small compared to the overall temperature change. It can be seen that in-line and rotated square banks exhibit symmetry across the lateral and diagonal lines of symmetry, respectively; indicative of the dominance of conduction over convection at very low Re . For $Re \geq 300$, it can be seen that there are large variations between the cylinder wall and the free-stream temperatures. In other words the doubly-periodic component of the temperature distribution is large in comparison to the overall bulk-temperature change. As might be expected for constant q_w , the isotherms cut the cylinder wall, indicating that the wall temperature is changing.

From Fig. 48, it can be seen that for $0.1 \leq Pr \leq 1$, the temperature distribution is relatively uniform (N.B. The $Pr = 1$, θ distribution is shown in Fig. 45b). As Pr is further increased in the range $10 \leq Pr \leq 100$, it can be seen that the temperature gradient becomes very large (several hundred times the overall change) and is concentrated near the cylinder wall. Note that an isothermal zone forms around the location of the vortex.

8.3.2 Average pressure drop

Inspection of Figs. 49 to 51 and Tables 4 to 12 reveals that Eu decreases with increasing Re . Values of Eu also decrease with increasing s/d and, at low Re , are of roughly the same magnitude for the three different bank arrangements under consideration (in-line square, rotated square, and equilateral triangular), at the same s/d . As Re increase, it can be seen that Eu is most for the rotated square configuration, somewhat less for the equilateral triangle banks, with in-line square banks offering the least flow resistance. Comparison of this writer's data with the existing numerical data of others, indicates that agreement is in general good; For in-line square banks the author's data are slightly lower than those of Le Feuvre (1973) and Antonopoulos (1979). For rotated square tubes, the data of Massey (1976) is significantly lower; however the results of Antonopoulos (1979) are in very close agreement with this writer's work. For equilateral triangular tube banks, the author's data are again slightly lower than those of Antonopoulos (1979). The grids used in the latter cases were highly non-orthogonal, and this probably contributed to the poorer agreement. Data of the other workers were obtained by digitising photocopies of their theses, using a tablet and mouse, and may therefore be subject to some error.

When the results are compared to the experimental data, agreement is only fair. It can be seen that, for in-line tube banks the results of this study are reasonably close to the experimental data of Bergelin et. al. (1950b, 1958), though a little lower. Agreement with ESDU (1980) and, in particular, the Zukauskas and Ulinskas' (HXDH, 1983) correlation is poorer; the author's pressure-drop predictions being significantly lower, in both cases. There are substantial disagreements between the Zukauskas and Ulinskas HXDH and the ESDU correlations, and the Bergelin et. al. data (the latter values lying substantially below the former). This is intriguing since both correlations are in-part based on the Bergelin et. al. data. The effect is most pronounced for in-line tube banks at

$100 \leq Re \leq 1\,000$. For rotated square banks, $s/d = 1.25$, agreement of this writer's results with the data of Bergelin et. al. is quite good. No experimental data were available for $s/d = 1.5$ and 2. It can be seen that the numerical results are somewhat smaller in magnitude than the ESDU correlations. The trend continues for equilateral triangular banks, where the author's data are close to, but somewhat lower than the Bergelin et. al. data for $s/d = 1.25$, and 1.5. Some discontinuities across the different Re ranges in the Zukauskas and Ulinskas HXDH correlation, have been deliberately smoothed out.

The Bergelin et. al. experiments were conducted on 10- and 14-row tube-banks. The data shown in Figs. 52 to 54 are uncorrected for entry effects: When these data are corrected using the guidelines suggested by Zukauskas and Ulinskas in the Heat Exchanger Design Handbook (1983), the resultant data actually agree very well with the results of the author, and further disagree with the two correlations (upon which they are in-part based).

8.3.3 Form and profile drag

Figures 55 to 57 show pressure coefficient, c_p , as a function of angle ϕ from the leading edge of the cylinder for in-line and staggered tube banks. It can be seen that in all cases the variation in c_p is most significant at low Re . For in-line tube banks, the pressure distribution is relatively constant at the front of the cylinder, where there is little flow, decreasing to a minimum around $\phi = 90^\circ$ to 120° for $Re = 10$, and again fairly constant at the rear of the cylinder. As Re increases, the location of the pressure minimum is displaced upstream. In contrast to potential flow [Fig. 12], there is almost no pressure recovery in the downstream zone. For rotated square configurations, at low Re , the pressure decreases monotonically from the leading edge of the cylinder ($\phi = 0^\circ$) to a minimum at $\phi = 180^\circ$. At intermediate Re , the pressure distribution is flat over the latter half of the cylinder while at high Re , the pressure rises to a maximum at around $\phi = 35^\circ$,

decreasing to a minimum at $\phi = 90^\circ$, with some recovery further downstream. Equilateral triangle banks exhibit features similar to rotated square banks at intermediate Re , over the entire Re range. c_p vs. ϕ profiles for $s/d = 1.25$ and 2 (not shown) are qualitatively the same as those for $s/d = 1.5$, the magnitude of the variations increases with decreasing s/d . Equilateral triangle banks tend to exhibit proportionally larger changes in c_p vs. ϕ , compared to rotated square.

Figures 58 to 60 are similar plots of friction coefficient, c_f vs. ϕ . For in-line tube banks, c_f is for the most part positive, reaching its highest value near the minimum-section at $\phi = 90^\circ$. Values of $c_f = 0$, indicative of separation and re-attachment are apparent for all but the smallest values of Re . For $s/d = 1.5$, the reattachment/separation angles are $15^\circ/154^\circ$ ($Re = 30$), $34^\circ/123^\circ$ ($Re = 100$) and $47^\circ/115^\circ$ ($Re = 300$). For $s/d = 1.25$ the corresponding values are $15^\circ/163^\circ$, $31^\circ/124^\circ$, $48^\circ/115^\circ$, and for $s/d = 2$ they are $0^\circ/146^\circ$, $32^\circ/127^\circ$, $44^\circ/115^\circ$. For $s/d = 1.25$ and 2, the c_f profiles are similar to those exhibited; in the former case the peaks tend to be greater in magnitude, while the converse is true for the latter.

For rotated square banks the shape of the c_f profiles are a function of s/d . For $s/d = 1.5$ [Fig. 59] two distinct maxima are observed $c_f = 3.2$ at 40° and $c_f = 2.4$ at 135° , at $Re = 10$. By $Re = 100$, only a single maximum of 0.6 at around 45° is observed. For $s/d = 1.5$ (not shown) at $Re = 10$, the skin friction increases to a single maximum of 2.2 at around $\phi = 43^\circ$, thereafter decreasing monotonically to zero. As Re increases, the magnitude of the maximum decreases to around 0.45, and the location moves downstream to around $\phi = 60^\circ$ for $Re = 300$. For $Re = 1\,000$, the extremum moves back upstream in location. For $s/d = 1.25$, on the other hand, up to $Re = 300$, two equal sized peaks are observed at around $\phi = 45^\circ$ and 135° , with a local minimum around $\phi = 85-90^\circ$. The separation angles corresponding to Re values 100, 300 are; 166° , 155° ($s/d=1.25$), 163° , 119° ($s/d=1.5$), 121° , 106° ($s/d=2$).

Equilateral triangle c_f profiles for $s/d=1.5$ [Fig. 60] display a single global maximum, at 80° to 90° , and two local maxima at 25° and 155° at low Re [Fig. 60]. $s/d=2$ profiles (not shown) showed good agreement with those of Antonopoulos (1976). For Re of 100 and 300 the angles are 178° , 116° ($s/d = 1.5$), 123° , 101° ($s/d = 2$). The reader will note that when generating local-coefficient data, here and elsewhere in the thesis, that staggered velocity values were treated as if they acted at cell centres. This introduces slight errors.

8.3.4 Overall heat transfer

Figure 62 is a comparison of overall heat transfer in terms of $\overline{Nu}_{LM}$ for in-line tube banks ($s/d = 1.25$ and 1.5 , constant t_w) with the data of Le Feuvre (1973). Le Feuvre suggested $\overline{Nu}_{LM}$ to be independent of the Peclet number, Pe , (i.e. the product $Re Pr$). This is in contrast to the empirical correlations (not shown) which suggest an increase of $\overline{Nu}$ with Re . The author's results, which are for $Pr = 1$, indicate that there is a small increase in $\overline{Nu}_{LM}$ with Re , however, as can be seen from Fig. 62, agreement with Le Feuvre is quite reasonable. Values of $\overline{Nu}_{LM}$ computed by the author [Fig. 62] differ from those displayed in Tables 4 to 12, in that the former are computed from the wall flux, while the latter are obtained from an overall heat balance [Eq. 8.25].

Figure 63 shows the author's data, for in-line tubes, constant q_w , compared with Antonopoulos (1979). The heat transfer factor j is shown as a function of Re . j is based on a reference upstream minimum cross-section temperature t_{um} [Fig. 61]. The author's data are in very close agreement with those of Antonopoulos for $s/d = 1.25$, $Pr = 1$. For $s/d = 2$, $Pr = 15$ and 25 , the author's data are somewhat higher than Antonopoulos' values. As is shown in Fig. 48, the effect of increasing Pr is to reduce the size of the thermal boundary layer, so that grid dependence may be significant. Antonopoulos noted good agreement with the $s/d = 1.25$ data of Bergelin et. al. (1950b, 1958) (not shown).

Figure 64 is a plot of j (based on t_{um}) vs. Re , for rotated square banks, $s/d = 1.5$; the results of the writer being compared to those of Massey (1976) and Antonopoulos (1979). It can be seen that agreement is in all cases reasonably good. The other authors' both noted good agreement with the data of Bergelin et. al. (1950b, 1958) for this value of s/d .

Figure 65 show $\overline{St}_T$ and $\overline{St}_H$ (based on t_f) as a function of Re for in-line tube banks at various pitch-to-diameter ratios for $Pr = 1$. Also shown are Zukauskas and Ulinskas' (1988a) K_f correlation [Eq. 8.30]. It can be seen that the agreement between the two sets of results is only approximate, the slope of the numerical results being substantially steeper than those of the correlation. The Zukauskas and Ulinskas' (1988a) correlation presumes K_f to be independent of pitch-to-diameter ration, however it would appear, from the numerical results that there is some dependence on s/d .

Figure 66 shows $\overline{St}_{LM}$ vs. Re , for rotated square and equilateral triangle banks $s/d = 1.25$ to 2 . $\overline{St}$ is based here on the log-mean temperature, computed from the heat flux at the wall, and not Eq. 8.24, for reasons which are discussed below. It can be seen that although there is broad agreement between these results and Zukauskas and Ulinskas' (1988a) correlation, it would appear from the simulations, that there is more heat transfer at $s/d = 1.25$ than at $s/d = 2$. The differences between $\overline{St}$ are largest for small Re ; as Re increases the $\overline{St}$ values converge on a value quite close to Zukauskas and Ulinskas' correlation value.

Figures 68 and 69 are comparisons of the available experimental data and the various correlations. In contrast to the pressure drop data, there appears to be relatively good agreement between the heat transfer data of Bergelin et. al. (1950b, 1958), ESDU (1973), Gnielinski et. al. in HXDH (1983), and Zukauskas and Ulinskas (1988a) correlations. This is in spite of the fact that the data and correlations are apparently based on a

mixture of constant q_w and t_w measurements, a variety of reference temperatures, t_{ref} , and different assumptions regarding the Pr_w/Pr_f dependence.

8.3.5 Local heat transfer

Figure 70 shows values of Nu_T (normalised with respect to $\overline{Nu}_T$) vs. ϕ for an in-line tube bank, $s/d = 1.5$, at various Re . It can be seen that at $Re = 10$ the local heat transfer coefficient is just over twice the mean value at $\phi = 55^\circ$. At $Re = 30$, the maximum value moves downstream to $\phi = 70^\circ$, and decreases to 1.8. As Re increases further the tendency reverses with the maximum value increasing in value and moving slightly upstream in location. This reversal was less marked and occurred at lower Re for $s/d = 1.25$, while for $s/d = 2$ the reverse was found to be true. Figure 71 shows $Nu_H/\overline{Nu}_H$ for in-line tube banks under conditions of constant heat flux. It can be seen that the distributions are qualitatively similar to those for constant t_w . The variation in $Nu/\overline{Nu}$ is substantially less for constant q_w and the maximum value of Nu_H occurs somewhat downstream than for constant t_w .

Figures 72 and 73 show $Nu_T/\overline{Nu}_T$ and $Nu_H/\overline{Nu}_H$, respectively, for a widely-spaced rotated square tube bank, $s/d = 2$. It can be seen that, as above, the distributions are qualitatively similar, especially at low Re , where Nu decreases monotonically. At higher Re one or more local extrema are observed. A local maximum occurs downstream as convection in the rear of the cylinder increases due to the growth of the vortex. Figure 76 shows $Nu_T/\overline{Nu}_T$ for compact rotated square tube bank, $s/d = 1.25$. It can be seen that at low Re , almost all heat transfer occurs at the front of the cylinder; at higher Re , twin peaks occur in the heat transfer coefficient distribution, due to flow acceleration in the diagonally-oriented minimum cross-section.

Figure 74 and 75 show $Nu_T/\overline{Nu}_T$ and $Nu_H/\overline{Nu}_H$ distributions for an ($s/d = 1.5$) equilateral-triangle tube bank. The distributions are similar to those for rotated square banks. For the staggered geometries it is apparent that the variation in $Nu_T/\overline{Nu}_T$ is greater than $Nu_H/\overline{Nu}_H$, but the effect is less pronounced than for the case of in-line geometry. For the staggered geometries, data from the first and last cells has been removed, as this clearly reflects grid-related error, in the region of the special points. It should be apparent that profile drag, skin friction, and local heat transfer are all a strong function of geometry, and that is little similarity between fluid friction and heat transfer.

8.3.6 Effect of Prandtl number

Figure 77 shows the Pr dependence for in-line tube banks. Both Le Feuvre (1973) and Antonopoulos (1979) observed $Nu/\overline{Nu}$ to be relatively insensitive to Pr , in the range $1 \leq Pr \leq 100$. On this basis they compared local heat transfer with the data of Zukauskas et. al. (1968), for $Pr = 528$. The results of this writer would seem to concur. Note however, that if $\overline{Nu}$ is a function of $Pr^{1/3}$, or thereabouts, it is not strongly dependent on Pr . For values of $Pr > 100$, the thermal boundary-layer becomes highly concentrated at the cylinder wall, suggesting that the methods used by this and previous authors may have limited Pr range. In this regard, at least, the I-grid is superior to the H-grid in that it may readily be concentrated at the wall.

8.3.7 Choice of reference temperature

The definition of, h [Eq. 8.32] is based on an arbitrary fluid reference temperature, t_{ref} . It was discovered that, at low Re , the choice of t_{ref} is of much importance. Various workers have used different t_{ref} . For example, the definition of h based on upstream free-stream temperature, t_f , used frequently in this thesis, is that preferred by Zukauskas

et. al. (1968, 1988a). Alternatively, Gnielinski et. al. (HXDH, 1983), Bergelin et. al. (1950b, 1958), and ESDU (1973) all use values based on the log-mean temperature difference, Δt_{LM} . Le Feuvre (1973) based overall $\overline{Nu}$ calculations on Δt_{LM} , however local heat transfer calculations were based on, t_m , chosen in a grid-independent fashion, i.e. t_{um} for $\phi > \pi/2$ and t_{dm} for $\phi < \pi/2$ [see Fig. 61]. Massey (1976) used the upstream value, t_{um} , exclusively as a basis for all calculations. Antonopoulos (1979) used Massey's definition to compute $\overline{St}$ values, and Le Feuvre's definition when computing local values of Nu . The latter were, however, normalised with respect to t_f' [Eq. 8.23].

At first sight, it might seem that any consistent module-average reference is appropriate. The matter is, however, not entirely straightforward: Figure 67 shows $\overline{St}$ vs. Re for a constant t_w in-line tube bank ($s/d = 1.5$), with $\overline{St}$ based on various definitions. It can be seen that there is a significant variation in the value of $\overline{St}$, depending on the choice of t_{ref} . The slope of the $\overline{St}_m$ distribution is steepest, with $\overline{St}_f$ (i.e. $\overline{St}_T$) the least steep, and $\overline{St}_{LM}$ in between. This is because, as was mentioned above [see Figs. 44 to 47], the change in bulk temperature, $t_b(x_c)$, is large compared to the local temperature variation at low Re , (and small by comparison at high Re). These differences are quite large for $Re < 300$, above this all definitions of h converge to a single value. The differences are less significant for widely-spaced banks, and more-so for compact banks. If t_{ref} is defined upstream of the cylinder, h , will in general be lower than if it is based on t_m at $\phi = 90^\circ$.

The problem is compounded for constant q_w boundary conditions, where both the bulk fluid temperature, t_b , and the wall temperature, t_w , change: Under these circumstances, at sufficient Re , the change in the bulk temperature is linear,

$$dt_b = \frac{q_w d}{\rho u_m c_p (s_1 - d)} d\phi \quad (8.33)$$

between the attachment and separation points. Outside of this region stream-wise diffusion may be significant, so the relationship is only approximate. The temperature in the minimum cross-section at $\phi = 90^\circ$, t_m , is thus almost the same as the mean-bulk temperature, $\bar{t}_b$. However $\overline{Nu}$ based on the average temperature, t_m differs from $\overline{Nu}$, based on the average temperature difference $t_w(\phi) - t_b(\phi)$. Indeed, at sufficiently low Re , it is possible (and these were observed) to obtain negative values of $\overline{St}_m$! This is due to the apparent temperature difference becoming very large and ultimately negative due to high values of h at the front of the cylinder [see Fig. 73].

Thus it was discovered through experience, that large disparities can and do arise as a result of the various choices of definitions of reference temperatures upon which $\overline{St}$ and $\overline{Nu}$ are based. Due to the fact that $\overline{Nu}$ vs. Re profiles have a positive slope, the disagreement is worse when this measure is used to characterise heat transfer performance. For this reason, a comprehensive set of results comparing this writer's data, and empirical work is not presented here, and it is suggested that previous workers who claimed good low Re results in comparison e.g. to the data of Bergelin et. al. (1950b, 1958) were fortunate in their choice of reference values.

A local heat transfer coefficient based on a local cross-wise bulk temperature, $t_b(\phi)$ [Eq. 8.6] appears attractive. This value may be averaged (integrated) to obtain, $\overline{Nu}$, or $\overline{St}$. This was not done, primarily because t_b is difficult to compute in curvilinear geometry (and to measure experimentally). Also such a quantity is of limited value upstream of the attachment point, and downstream of the separation point. Patankar et. al. (1977) considered this problem, and suggested that the absolute value $|u|$ be used to compute, t_b , when considering recirculating flows, reflecting the fact that the velocity, $u(y_c)$, is at times negative. This notion was considered, but rejected by the author, because it leads to the nonsensical situation whereby a change in t_b may be effected entirely by flow-related effects, for example in an adiabatic region. It would appear that outside of zones where heat transfer is dominated by conduction perpendicular to and

convection parallel to a wall, the practice of assuming a linear relation between heat flux and temperature difference, and the usefulness of concepts such as h and t_b is to some extent limited. Further difficulties arise with compact rotated square tube banks, where the heat transfer surfaces overlap somewhat (in the stream-wise direction). It could be argued that the bulk temperature should therefore be obtained by averaging across the diagonal direction. Notwithstanding these difficulties, from a theoretical position, the use of t_b probably represents the most meaningful definition of t_{ref} available, for use in calculating $\bar{h}$.

Basing the overall heat transfer coefficient $\bar{h}$ on the log-mean temperature difference, Δt_{LM} , is consistent with the definition of $\bar{h}$ as the average (local) heat transfer coefficient, under constant t_w boundary conditions, provided the stream-wise diffusive influence is second-order. A significant advantage of this definition of $\bar{h}$ is that it can be readily calculated (or measured) based on knowledge of the inlet and exit conditions alone [Eq. 8.25]. The values of $\overline{Nu}_{LM}$ given in the summary of results, Tables 4 to 12, are based on an overall heat balance. A comparison of values obtained from the wall flux (the appropriate t_{ref} is computed from t_i and Δt_{LM}) showed very good agreement with those obtained from an energy balance, except for compact tube banks at very low Re [see Fig. 67], when downstream temperature gradients are tiny, and stream-wise diffusion is again significant. Significant disagreement was also observed for compact rotated square tube banks ($s/d = 1.25$ and 1.5), where heat transfer occurs from twice the usual surface area, over a portion of the flow path, thus rendering void the assumptions upon which Eq. 8.25 is based (values given in Tables 7 and 8 are uncorrected i.e. they reflect this error). It might appear that $\overline{Nu}$ computed from the wall heat flux should be less accurate than values obtained from an energy balance, due to the error associated with the assumed linear temperature profile at the wall. This is a naive argument; the same assumption is tacitly made when posing the boundary conditions themselves. For constant q_w conditions, however, it is not possible to compute $\overline{Nu}$ from a heat balance. Under these conditions both t_w and t_b are required.

The use of the upstream t_f provides a conservative measure, with the advantage of always being positive (an advantage shared by t_{um} and t_f'). The choice of t_f for in-line and t_f' [Eq. 8.23] for staggered geometry, as the basis for defining h , was selected as a compromise. It was primarily chosen so that constant q_w and constant t_w heat transfer cases may be compared for similar geometries. It is fully appreciated that there are minor inconsistencies when comparing the results of in-line and staggered geometries. For compact rotated square banks with $s/d < 1.707$, due to the fact that an adiabatic inter-tube zone does not exist, a unique definition of t_f is not apparent. For widely-spaced tube banks, where such a zone does exist, it is not easy to compute t_f , again due to the BFC grid geometry. Thus t_f' was adopted as an approximation to t_f . (The only temperature that is both unique and readily available for all of the geometries, grids, and thermal boundary conditions is t_m . The shortcomings associated with definitions based on this quantity have already been discussed above.)

For the case of constant wall heat flux, t_f' is clearly a good approximation for t_f [see Eq. 8.33]. This is not necessarily the case, under conditions of constant wall temperature. However, for constant t_w , the normalised distribution $Nu/\overline{Nu}$ is independent of whether t_{ref} is chosen as t_f , t_f' , t_m , a fact which follows from the form of the definition of $\overline{h}$, (Zukauskas, 1987). For these two reasons, even though the values of $\overline{Nu}$ given for in-line geometries [Tables 4 to 6] are not strictly comparable to those for staggered geometries [Tables 7 to 12], the normalised local Nu distributions [Figs. 70 to 75] are. This choice of definitions does provide the reader with, at least, some idea of the local heat transfer distributions.

The origins of the problem of the reference temperature are rooted in the assumption of a constant, module-average reference temperature. While it is true that there is a physical location in the domain where the temperature equals the appropriate reference value, there is no reason for such a location not to vary, as a function of Re , Pr , constant q_w or t_w . It would therefore appear that for low Re , there simply is no single temperature that

is representative. In hindsight, the additional effort involved in generating a utility to interpolate $t_b(x_c)$ from the curvilinear grid may have been worthwhile. Although such a definition is not entirely unambiguous, it is recommended for future work. It is to be emphasised that the problems associated with the choice of t_{ref} are low- Re related, for $Re > 1\,000$, due to the small changes in the overall temperature between modules, all appropriately defined reference temperatures converge on a single value, and can thus be used with confidence.

8.3.8 Grid independence

The effect of the grid-type size as well as location of the boundaries was analysed for both in-line and staggered tube banks: Figure 77 shows c_p , c_f and Nu_T as a function of ϕ , for in-line H-grids composed of 20x20, 40x40, and 80x80 cells. Figure 78 is plot of similar data reproduced here to illustrate the effect of the choice of grid type, and location. It can be seen that the results for the two H-grids are essentially identical regardless of whether the cylinders are located at the corners or the sides of the domain. Those obtained using the I-grid differ only slightly from the above. Figure 79 shows a study for an equilateral tube bank with $s/d = 2$. It can be seen that in this case there are substantial differences in the local pressure and heat transfer coefficient data, depending on whether an H- or I-grid is selected. The reasons for this are discussed below.

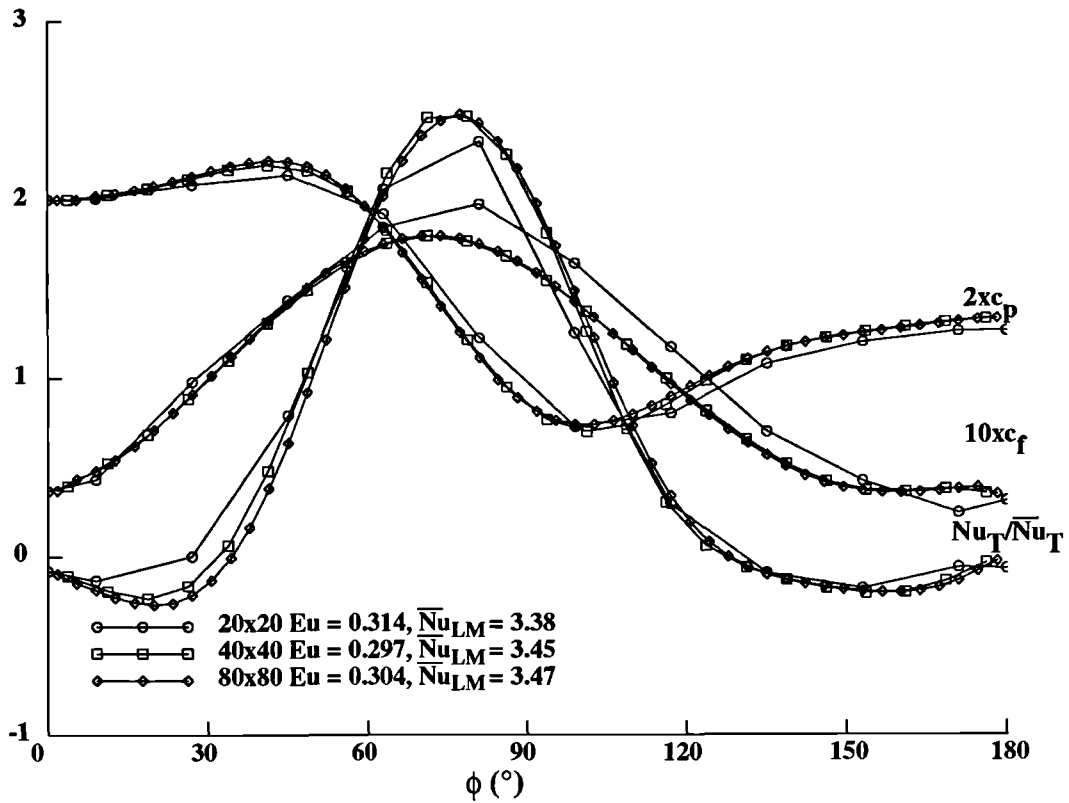


Figure 77. Effect of grid size, in-line square, $s/d = 2$, $Re = 100$.

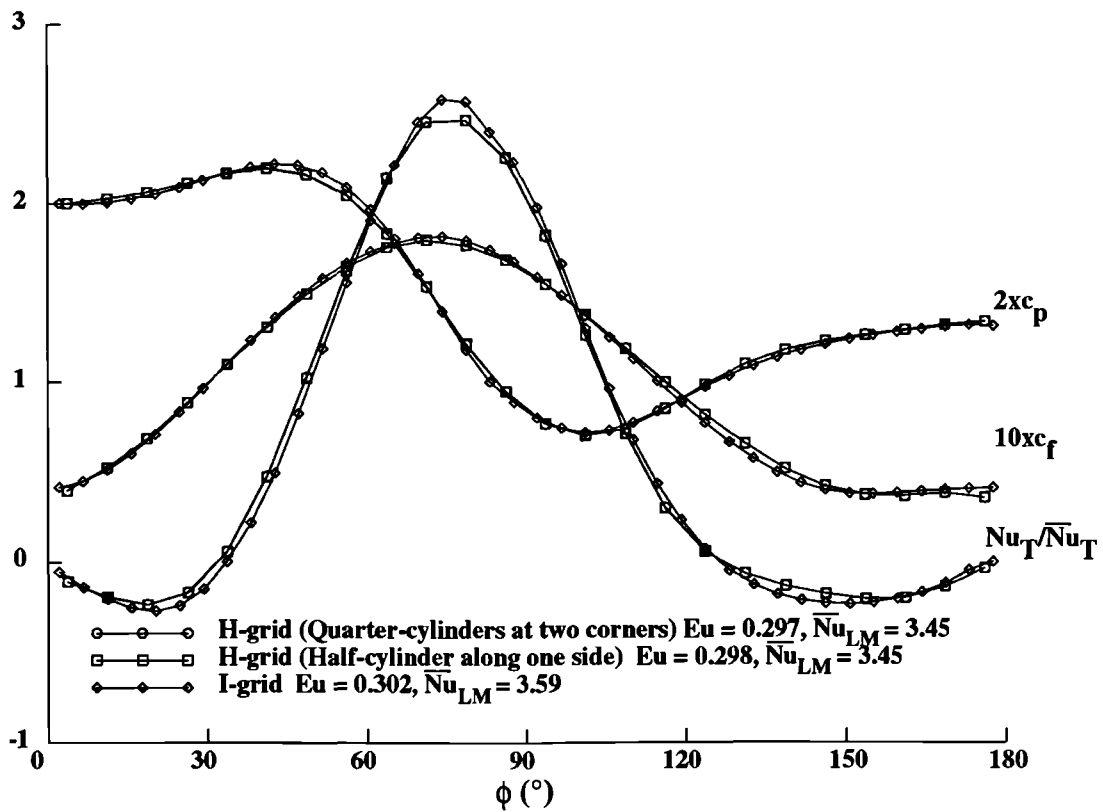


Figure 78. Effect of grid type, in-line square, $s/d = 2$, $Re = 100$.

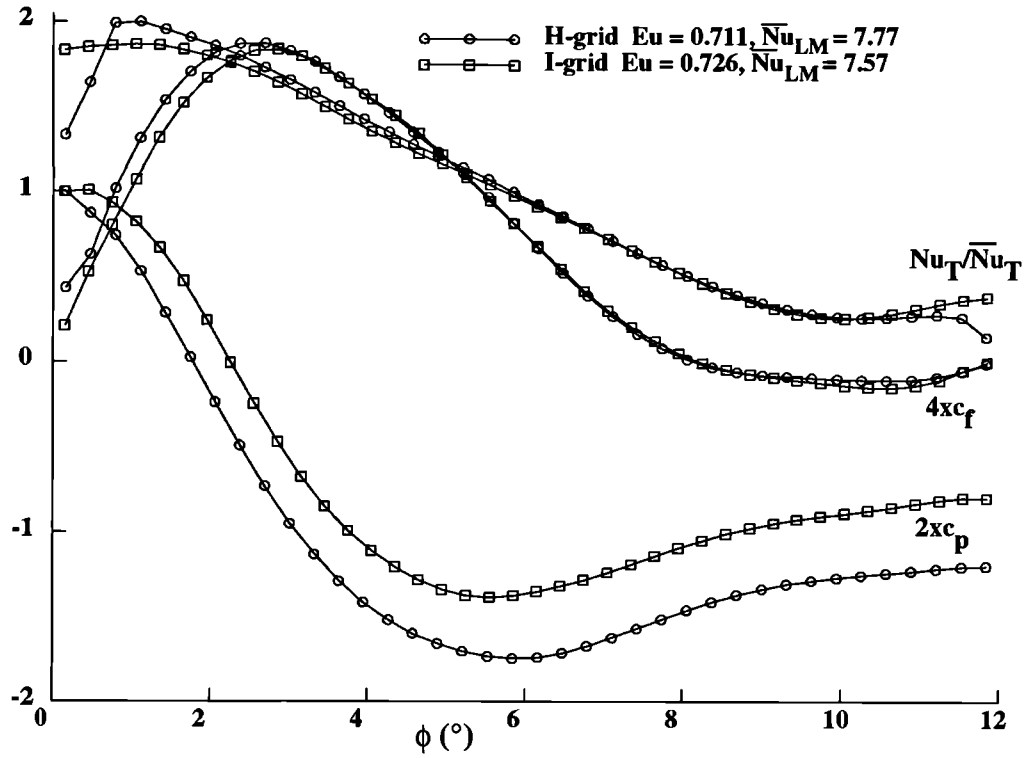


Figure 79. Effect of grid type, equilateral triangle, $s/d = 2$, $Re = 100$.

Figure 77 suggests that a reasonable degree of grid independence has, apparently, been achieved for the size of grids used in this study. It can be seen that there is a change of 6% in the value of Eu , between the 20x20 and 40x40 H-grid results, decreasing to a -2% change when increasing the mesh density to 80x80. The corresponding changes in $\overline{Nu}_{LM}$ are 2% and 0.6%. Inspection of the c_p , c_f and $Nu_T/\overline{Nu}_T$ reveals that there is little improvement to be had in refining the grid beyond 40x40.

Inspection of Fig. 78 shows that for in-line tube banks, there is close agreement between the H-grid results, regardless of whether the cylinder(s) are located in the more usual corner position [Fig. 26, also Fig. 61a] or alternatively-located at the side [Fig. 27, Fig. 61b], showing that the methodology is equally valid for boundaries where recirculation is present or absent. It is also apparent from Fig. 78 that quite good agreement, in terms of c_p , c_f and $Nu_T/\overline{Nu}_T$, is obtained between the results of H-grid and I-grid based simulations,

For the staggered geometry [Fig. 79], location of the boundary, for H-grids, was again found to be irrelevant (not shown). There is, however, little advantage to be had from locating the cylinder at the sides of the grid, and furthermore the in-line H-grids are poorly aligned with the (staggered) flow.

In contrast to the above, when results obtained using an in-line I-grid [Fig. 28] are compared to those from a diagonal H-grid [Fig. 29], there is substantial disagreement in the pressure coefficient data. The reason is that, for in-line banks, the special point in the grid [Chapter 6] is always located in a region where the velocity is small, regardless of the grid-type. For widely-spaced staggered geometry, using an I-grid, the upstream special point is now located in the free-stream, where the velocity is a maximum: Use of the I-grid resulted in an erroneous extremum in the pressure field being observed in this region. The effect was less noticeable for compact tube banks, and at higher Re , when the centreline velocity profile was only partially developed (e.g. using diagonal I-grids). It is to be concluded that location of the special point is of paramount concern.

Further examination revealed that, regardless of the grid-type, there is almost always some slight distortion of the pressure field in the region of the special point. This may be minimised by locating the special point in a region of low flow. Conventional wisdom in external aerodynamics, is to move the special point as far away from the solid surface as possible: For the class of flow under consideration, there is nowhere sufficiently distant from the cylinder to serve such a need. This is unfortunate since the I-grid has some very attractive features; in particular grid cells may readily be concentrated around the entire cylinder wall. For in-line flow this is not particularly important, since the flow attaches and separates well away from the front and rear of the cylinder. For staggered flow, it would however be desirable to be able to concentrate the grid in the former region. This is especially true when it is desired to identify secondary flows [see Chapter 10, on fin and tube heat exchangers]. The main impact of the special point

location appears to be on the pressure field (and hence Eu); the impact on the velocity and temperature fields, though still significant, is less severe.

The use of non-orthogonal H-grids allowed, for some degree of control over the grid to be established, especially for staggered grids. Again, however, if the grid was concentrated very close to the special point, pressure extrema were observed. Thus, even though it is possible, to construct H-grids with the cells packed substantially closer around the cylinder than is shown in Chapter 6, this proves to be a self-defeating exercise, in that pressure-distortions near the special point(s) decrease rather than increase confidence in the results. This problem was also found to arise when using the hybrid grid [Fig. 31], used by Antonopoulos (1979) which in spite of its attractive features was therefore rejected. (Antonopoulos did not discuss the local pressure distribution, in any detail, in his thesis). Block H-grids [Fig. 32] also produced pressure and velocity-field distortions near the special point when it was located at 45° . As the point is moved towards 90° , the impact was found to decrease. It was discovered that provided the special point was moved to 85° to 90° [Figs. 33 and 34], the problem was minimised. For steady fully-developed 2-D flow there is little justification (for the additional storage requirements required) in the use of these grids, however for simulations involving a manifold of such cylinders they are useful. It would appear that this phenomenon is a defect of many conventional finite-volume-based methods; Peric (1985), for example, noted similar pressure extrema using a co-location scheme with non-orthogonal BFC grids. It seems doubtful that so-called higher-order schemes will improve the situation, unless they represent flow-oriented, rather than grid-oriented integration schemes.

To summarise; although a number of grids were tried, most were ultimately rejected. For in-line tube banks, there is no need to concentrate grid cells at $\phi = 0^\circ$ and 180° , and therefore orthogonal grids were used. For staggered grids, non-orthogonal H-grids proved to be more useful, the main exception being compact staggered grids where diagonal-I grids were employed. The best grids for the current generation of flow solv-

ers, are still the H- and I- types employed by Antonopoulos (1976), modified using control-functions, P and Q , to conform better to the wall.

8.3.9 Convergence

Convergence was for the most part achieved without difficulty. Some difficulties were initially encountered due to fluctuations in the pressure field. The downstream in-cell pressure was initially fixed to the desired value using a single value for the pressure coefficient, C_p . A large pressure coefficient was found to amplify naturally-occurring fluctuations in the (upstream) pressure field, while a small coefficient was not sufficient to fix the in-cell pressure to the desired value. The solution was to scale the pressure coefficient on a cell-by-cell basis, in such a manner as to be problem-independent, assuming stream-wise convection to be dominant. Under these circumstances the only significant terms remaining in the continuity equation [Eq. 5.18] are,

$$AC_p(V_p - p_p) + \rho Au_w = 0 \quad (8.34)$$

where V_p is the desired pressure value, and p_p the current in-cell pressure. If this pressure difference is denoted by Δp_{cell} , then $\Delta p_{cell}/\rho u_w^2$, should be a function of the z -direction cell-momentum Peclet number only. Thus, by dividing the pressure coefficient by the quantity $|u_w|$, the pressure distribution is correctly specified in an always positive way which is independent of geometry, density etc.

$$C_P = \frac{C'}{|u|} \quad (8.35)$$

A coefficient of $C' = 10^6$ was found to be sufficient for this purpose. Even when in-flow did occur (so that stream-wise convection is not the only influence), Eq. 8.35 was

to found to be a reasonable formulation. The reader should also note that the modifications to the upstream boundary conditions to account for curvature and divergence, and momentum-equation coefficients are necessary; neglecting these results in, say, significant local deviations in the pressure distribution.

It is worth remarking that other methods of prescribing boundary conditions are possible: Patankar et. al. (1977) used a transformation of the transport equations, to convert the periodic boundary conditions into cyclic boundary conditions. Their approach was both simple and elegant. The preference of the author, however, to use the methodology described above is for three reasons: (a) The flow Reynolds' number, Re , could be directly prescribed. (b) For staggered tube banks, flow symmetry was exploited, decreasing the required number of cells by 50%. (c) The constant wall temperature thermal boundary condition was easily prescribed.

Occasionally, inconsistencies were noticed in c_p and c_f between inlet and exit values. These typically occurred at low Re , when using highly non-orthogonal grids. Although the cause was never determined, these probably resulted from some minor difference between treatment at the boundaries, and the interior grid points. In addition discontinuities in the pressure field at other boundaries were at times displayed. These were apparent in compact in-line and staggered grids, at low Re , at $j = 1$ and ny , probably arising from the bias occurring when computing vectorial quantities in the staggered geometry.

Some numerical studies were conducted assuming potential flow (not shown), in order to validate the methodology and also to gain confidence. Three distinct approaches were considered

- (a) By allocating the store normally occupied by pressure, to calculate a velocity potential [Eq. 4.2], with $\vec{u}$ proportional to $\text{grad } p$. This is achieved by eliminating the inertial and viscous terms, and introducing volumetric

source terms for u , v in the momentum equations. (This approach is also suitable for the solution problems involving fluid flow in porous media.)

- (b) By solving for velocity potential as a scalar variable, ϕ , which satisfies Laplace's equation, i.e. diffusion-only active.
- (c) By neglecting the viscous terms in the momentum equation, and solving the Euler-form of the momentum equations.

In general, the best results were obtained using method (a) with very close agreement being observed between the numerical results, and the analytical-based results of Chapter 4 being observed. There was some disagreement between the results generated using method (b), due to the bias in values used to compute diffusion flux vectors. Method (c) is least good: As a result of numerical diffusion, separation was, at times observed. It is therefore not recommended. Although some authors have suggested the capability of CFD codes to precisely predict potential flow to be of fundamental importance, the Navier-Stokes' equations [Eq. 3.2] contain other terms, which are also important. The best computer code is one which minimises the errors associated with of all of these, rather than any one term in particular.

8.4 Closure

A detailed analysis of two-dimensional fluid flow and heat transfer in in-line square, rotated square and equilateral triangle tube banks was presented. The results were compared to existing experimental and numerical data. Numerous parametric results were presented, together with a detailed discussion.

9. UNSTEADY TWO-DIMENSIONAL FLOW

9.1 Introduction

Although in the past there has been speculation as to whether tube banks exhibit transient-periodic behaviour (Le Feuvre, 1973), it is now generally accepted that such phenomena do occur, even in compact tube banks. The 16 mm cine-film by Weaver and Abd-Rabbo (1984), for example, shows this phenomenon both in stationary and vibrating tube banks (see also the papers by Weaver and Abd-Rabbo, 1984, and Abd-Rabbo and Weaver, 1985). Vortex shedding in tube banks is a subject for concern in that it can contribute to flow-induced vibration. The paper Païdoussis (1982) is a thorough review of numerous articles on the subject.

Transient-periodic phenomena are typically characterised by a Strouhal number, Sh ,

$$Sh \equiv \frac{fd}{u_m} \quad (9.1)$$

where d is the cylinder diameter, u_m is the bulk velocity, and f the characteristic frequency. Zukauskas et. al. (1988b) give correlations of Sh vs. Re for in-line and staggered tube banks, based on experimental data.

This Chapter describes a numerical procedure used to investigate unsteady flow in tube banks. The scope of the problem is confined to in-line and rotated square tube banks, $s/d = 2$. Both fluid flow and heat transfer, under constant q_w boundary conditions are considered.

9.1.1 Domain of interest

For the work described in this chapter, multiple stream-wise modules were employed. For in-line tube banks, two complete modules were used, while for staggered banks, four were normally employed. Figure 80 shows the arrangements. In both cases block H-grids of the type illustrated in Figs. 33 and 34 (as discussed in Chapter 6) were used. The choice of location of the cylinders (at the centre or corners of the domain) was selected primarily to provide a good view of the vortex generation process.

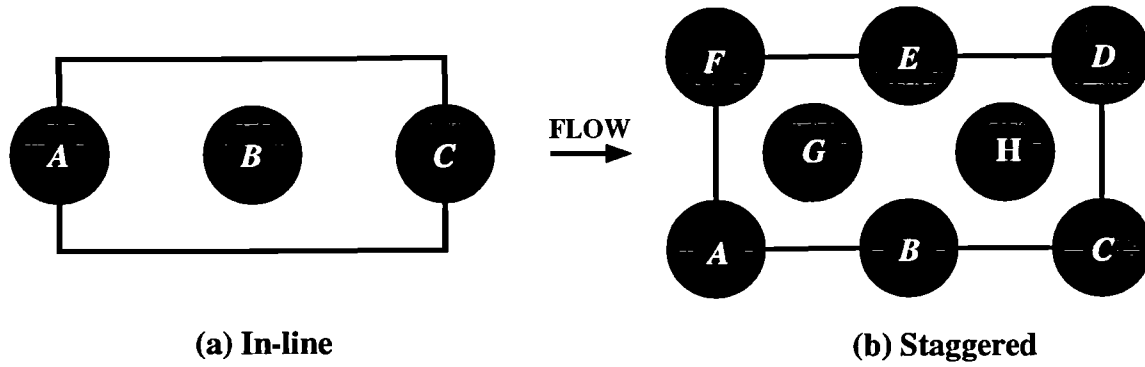


Figure 80. Domain boundaries.

9.1.2 Initial and boundary conditions

Let it be supposed that in the interior of the tube bank, there is a transient-periodic type of flow such that,

$$p(x_c, y_c, t) = p(x_c + s_2, y_c, t + \tau_x) + \Delta \bar{p} = p(x_c, y_c + s_1, t + \tau_y) \quad (9.2)$$

$$\vec{u}(x_c, y_c, t) = \vec{u}(x_c + s_2, y_c, t + \tau_x) = \vec{u}(x_c, y_c + s_1, t + \tau_y) \quad (9.3)$$

where τ_x and τ_y are phase differences. There are two difficulties associated with these boundary conditions;

- (a) The double-substitution process, described in Chapter 8, must be replaced by some sort of four-way substitution process, at all system boundaries.
- (b) There is no way of knowing, *a priori*, the phase differences τ_x and τ_y .

In order to simplify the problem, it was decided to make some additional assumptions, namely,

- (a) All the tubes in a given row are in phase ($\tau_x = 0$), so that cyclic boundary conditions may be applied across the lateral boundaries of the domain [Fig. 80]
- (b) The downstream pressure field may be calculated from the downstream velocity field according to Bernoulli's Law [Eq. 4.1]

If the domain boundaries are now chosen to correspond to entrance/exit zones, then, the four-way substitution process reduces to a single (downstream to upstream) substitution. The approach adopted then was as follows;

- (i) A constant flow, $u(0,y,t) = u_m$, is applied at the inlet of the upstream module, over several time-steps. At the same time a sinusoidal disturbance, $u(x,0,t) = u_0 \sin(\omega t)$ is applied across the lateral boundary *A-B* [Fig. 80] of the upstream module for several cycles
- (ii) The dynamic response at the inlet (u and v) and the side (v only) of the downstream unit [see Fig. 81] are measured and stored.

- (iii) The measure time-dependent responses are now applied upstream.
- (iv) Steps (ii) and (iii) are reiterated.

The initial response was applied for three complete cycles, at an assumed Sh in the range 0.1 to 0.2. The procedure was automated, by storing the response values in ring buffers, updated at each time step, as is illustrated schematically in Figure 81.

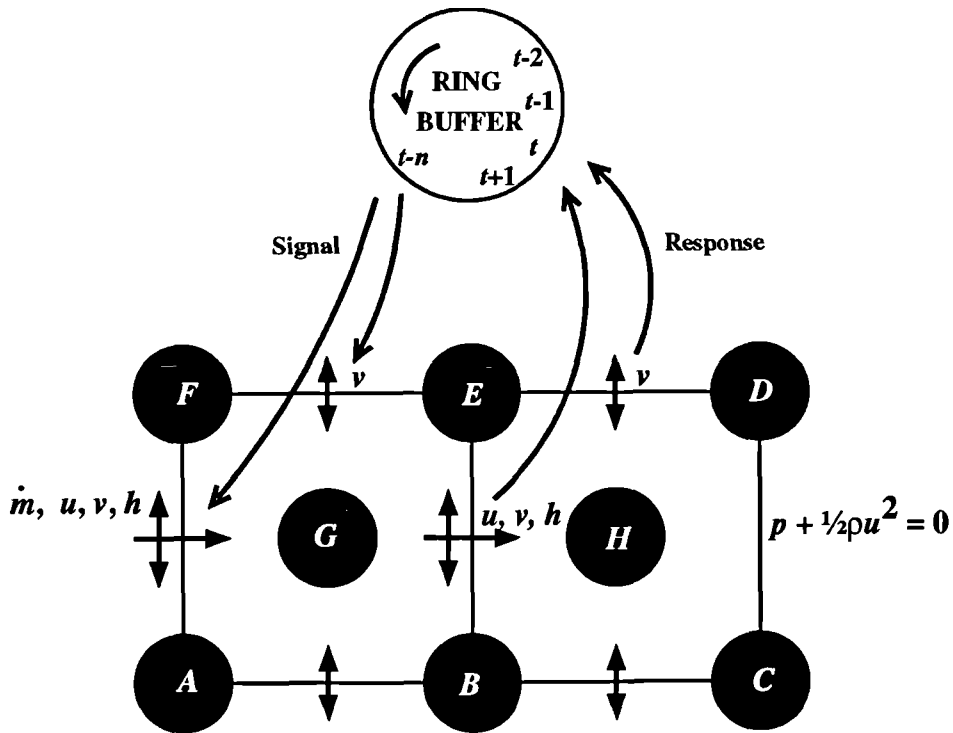


Figure 81. Boundary conditions.

The inlet and exit were selected to be minimum cross-sections, so that only convection was considered when computing the upstream u -momentum boundary conditions. Upstream and lateral u -values were fixed using a large coefficient. The downstream stagnation pressure in the halo-cells was assumed zero so that,

$$p_{nx} = -\frac{1}{2}\rho u_{nx-1}^2 \quad (9.4)$$

the in-cell pressure value being fixed (approximately) to the dynamic pressure by setting the pressure coefficient according to,

$$C_P = \sqrt{\frac{2p}{\bar{p}_P}} \quad (9.5)$$

$$V_P = 0 \quad (9.6)$$

A similar result can also be obtained by assuming C' [Eq. 8.35] to be $\frac{1}{2}$.

Heat transfer was assumed to occur at constant wall heat flux, it being assumed that the long-term time-average upstream bulk temperature was zero. This was achieved by summing the bulk temperature across B ,

$$\bar{t}_b = \frac{1}{T} \sum_{n=1}^T t_b \quad (9.7)$$

and subtracting the quantity $\rho c_p \bar{t}_b$ from the values previously stored in the ring buffer, prior to substitution upstream.

Simulations were conducted for in-line and rotated square tube banks with $s/d = 2$ at Re of 100, 300, 1 000, and 3 000. Animation sequences of vector plots, particle traces, stream-lines and pressure contours were prepared using flow visualisation software (Plessel, 1989, Walatka et. al., 1990).

9.2 Results

9.2.1 Flow field results

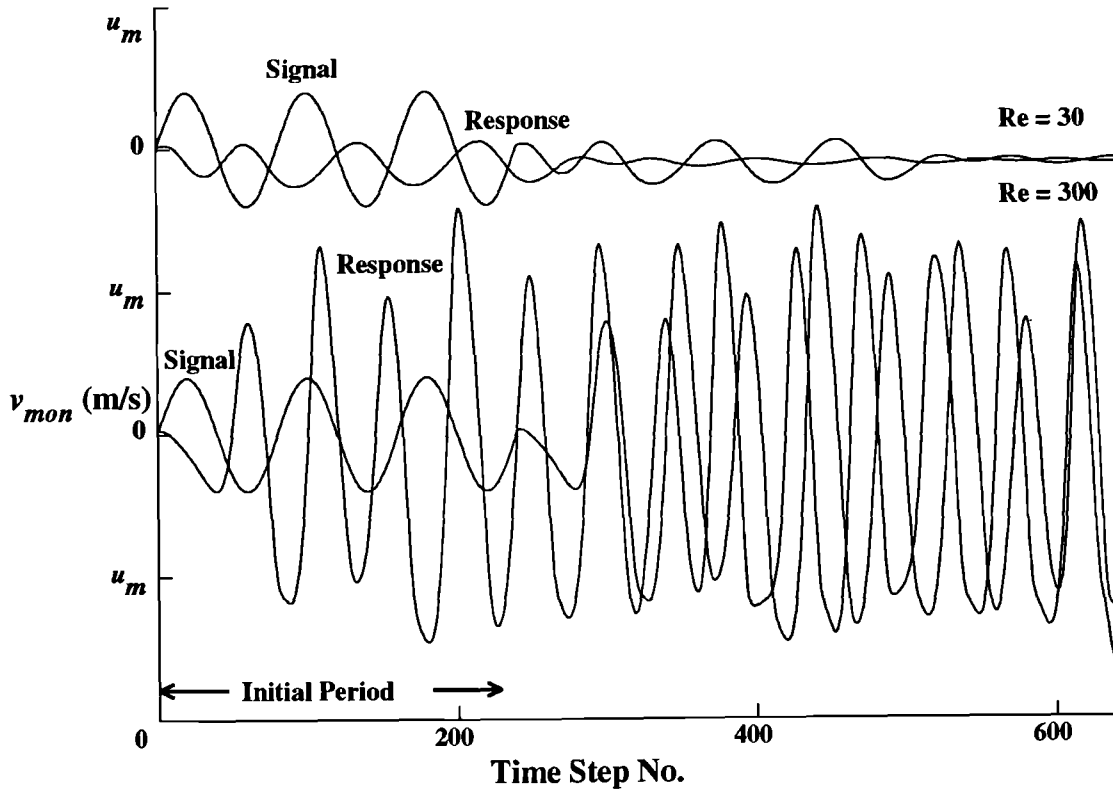


Figure 82. Response of a rotated square tube bank.

Results are presented for in-line and staggered geometries in the range $Re = 30$ to 3 000. In both cases grids of size 160x80 were employed. For the staggered geometry a total of 640 time steps was used, while for in-line tube banks, 960 time steps were used. These times corresponded to 8 and 12 cycles per run at the assumed frequencies. In both cases, 40 sweeps per time step and 20 iterations per sweep were performed.

Figure 82 is a plot of the dynamic downstream response to the applied disturbance, for the rotated square tube bank at $Re = 30$ and 300. The values shown are the cross-wise

v -velocities at two monitor-points located midway between cylinders A - B (upstream) and B - C (downstream) [Fig. 81]. The downstream response is itself applied as the upstream disturbance after three cycles (periods). It can be seen that at $Re = 30$, the periodic disturbance is rapidly damped out. For $Re = 300$ a transient periodic behaviour is established.

Figures 83 to 86 show a sequence of (six) plots of velocity vectors, stream-lines, pressure contours and temperature contours for the in-line bank at $Re = 300$. Figures 87 to 90 are similar plots for the rotated square bank, also for $Re = 300$. The data shown for the in-line bank are for the upstream module A - B [Fig. 80], and the middle zone B - H - E - G for the staggered bank. Inspection of Figs. 83 to 86 reveals information about the transient behaviour of the motion, which is discussed in detail below.

Figures 91 and 92 show values of the cross-wise velocity, v_{mon} , the fluctuating component of the stream-wise velocity, u_{mon} , and the pressure, p_{mon} vs. time, at a monitor point location. In both cases, the monitor point is located midway between cylinders B and C [Fig. 80]. The time, t^* , is normalised according to,

$$t^* \equiv \frac{tu_m}{d} \quad (9.8)$$

where t is the time from the start of the simulation, d is the cylinder diameter, and u_m is the mean bulk velocity. Data is shown for the latter half of the simulation only. (In the interests of clarity, values of p_{mon} and v_{mon} are offset by one unit each, in the y -direction.)

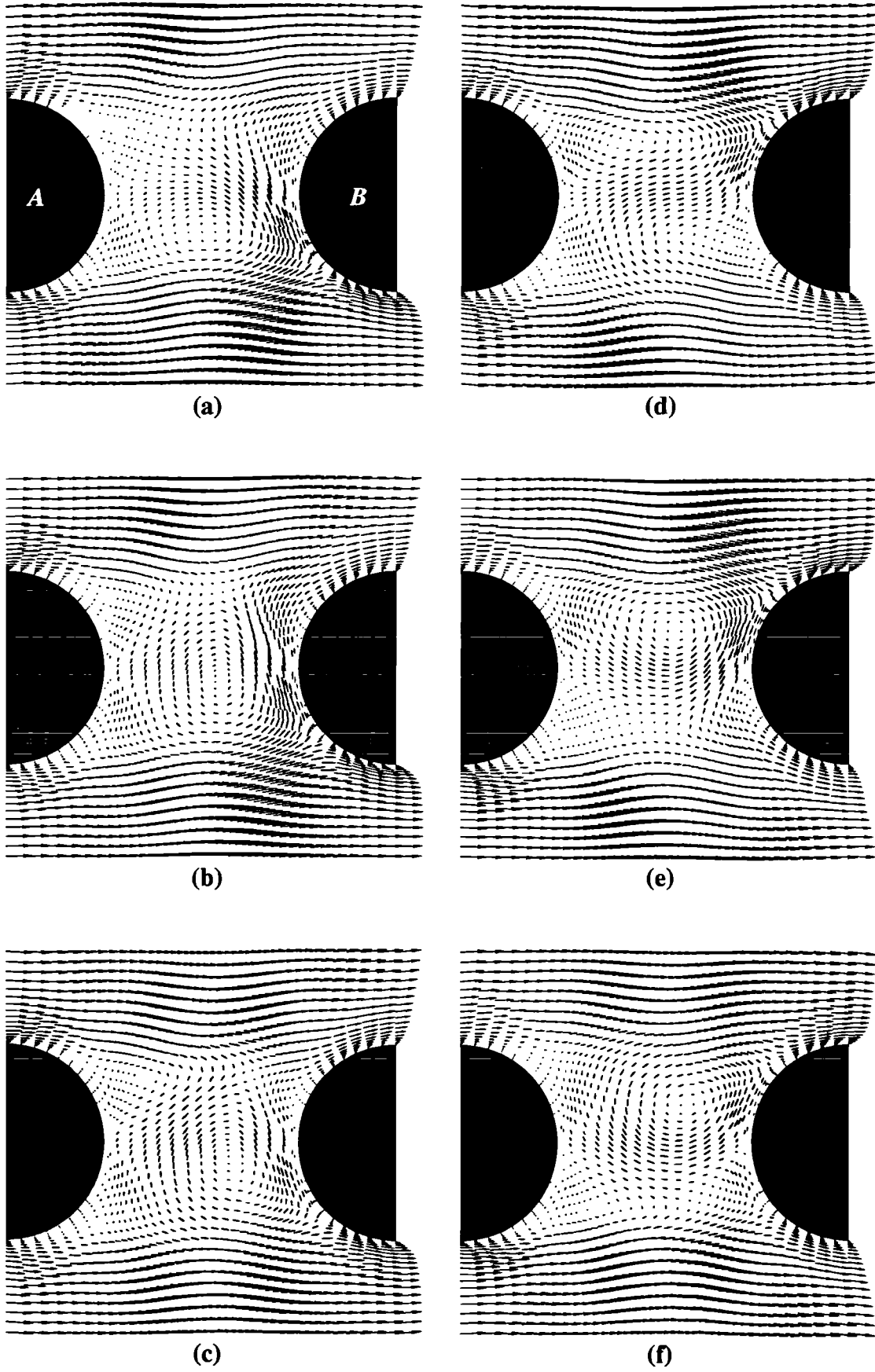


Figure 83. Velocity vectors, in-line square, $Re = 300$.

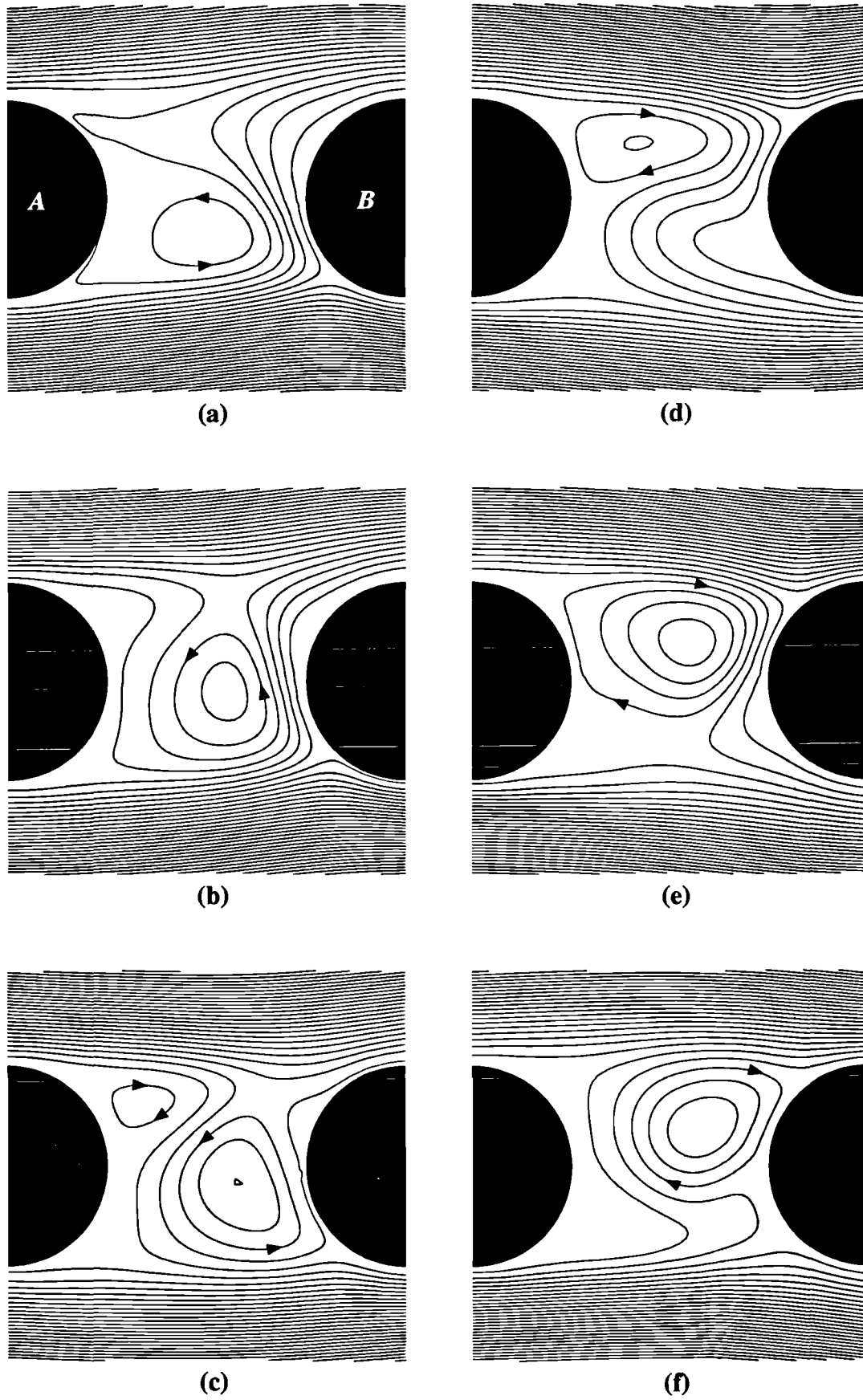


Figure 84. Stream-lines, in-line square, $Re = 300$.

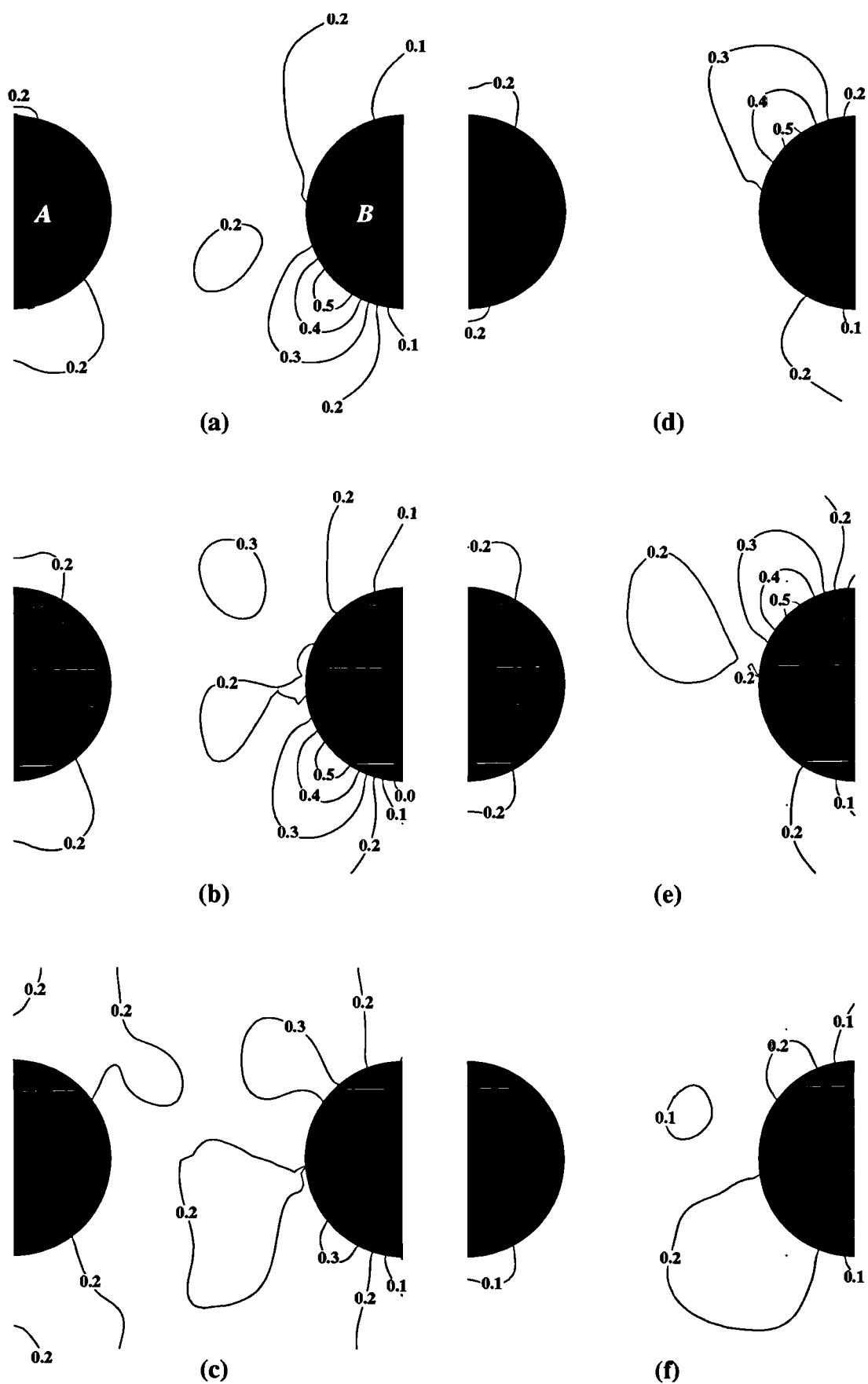
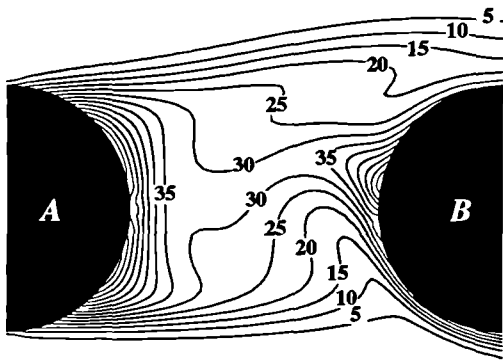
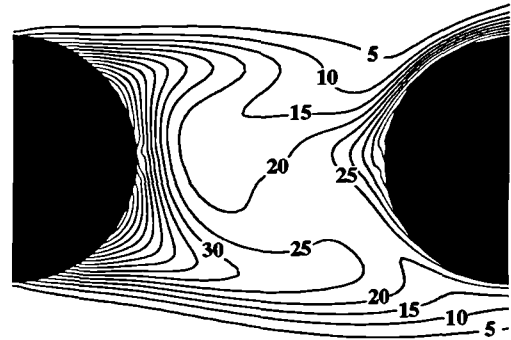


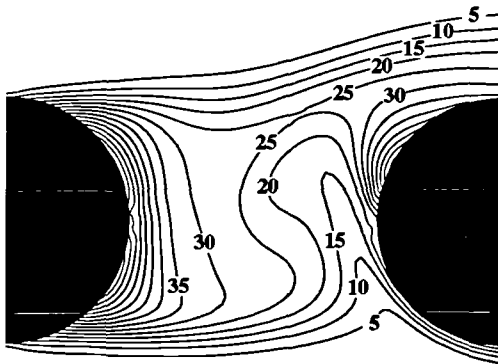
Figure 85. Pressure contours (Pa), in-line square, $Re = 300$.



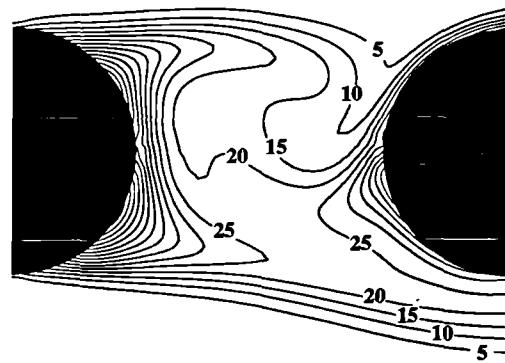
(a)



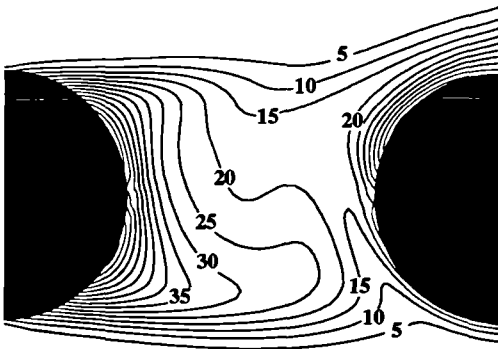
(d)



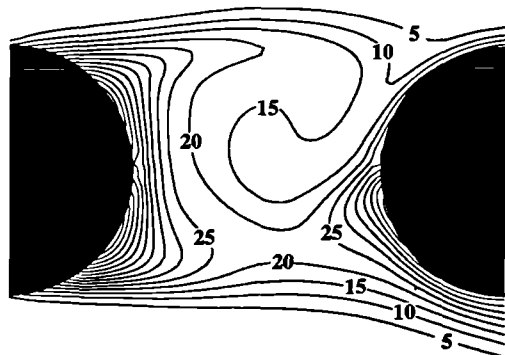
(b)



(e)



(c)



(f)

Figure 86. Temperature contours ($^{\circ}\text{C}$), in-line square, $Re = 300$.

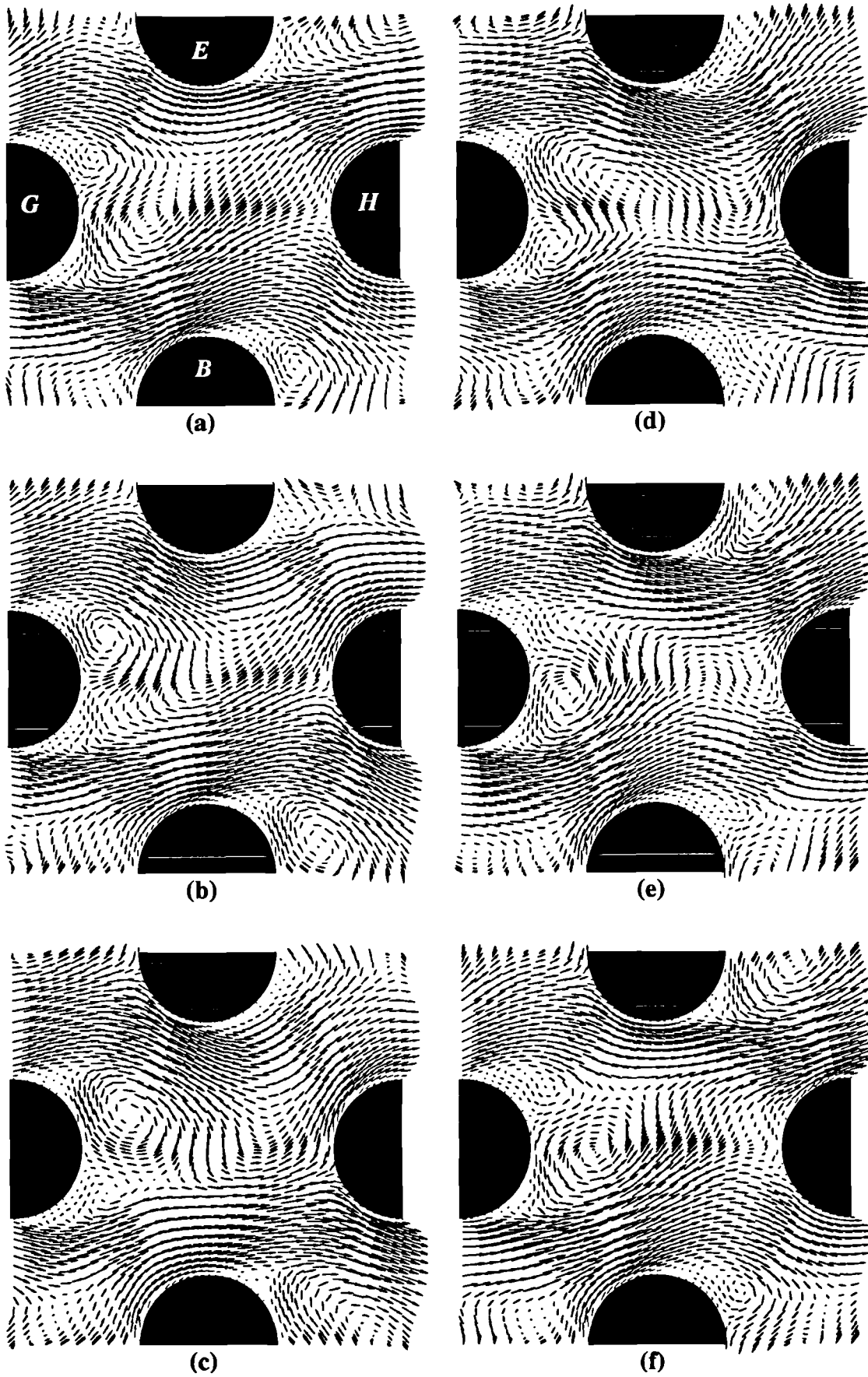


Figure 87. Velocity vectors, rotated square, $Re = 300$.

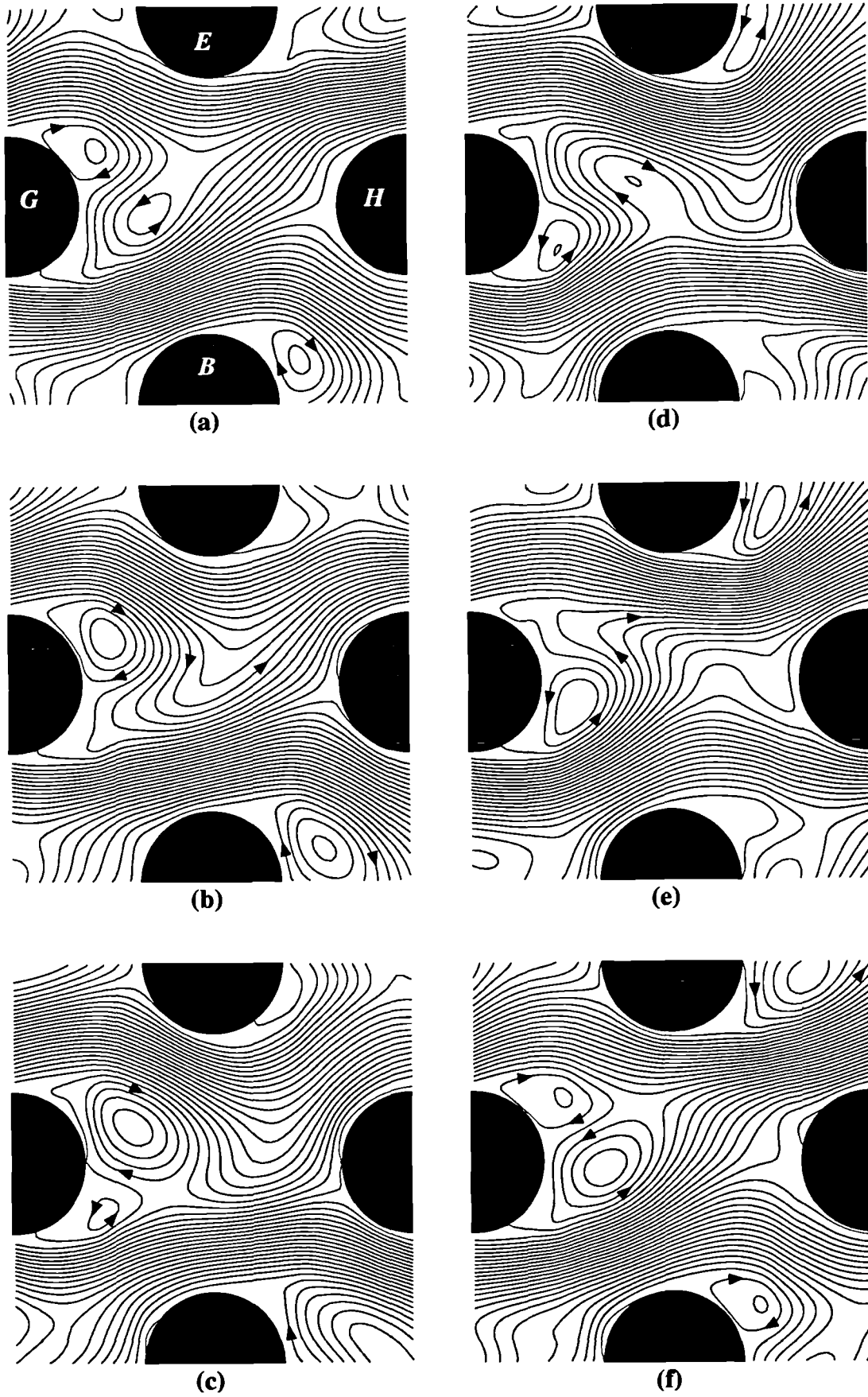


Figure 88. Stream-lines, rotated square, $Re = 300$.

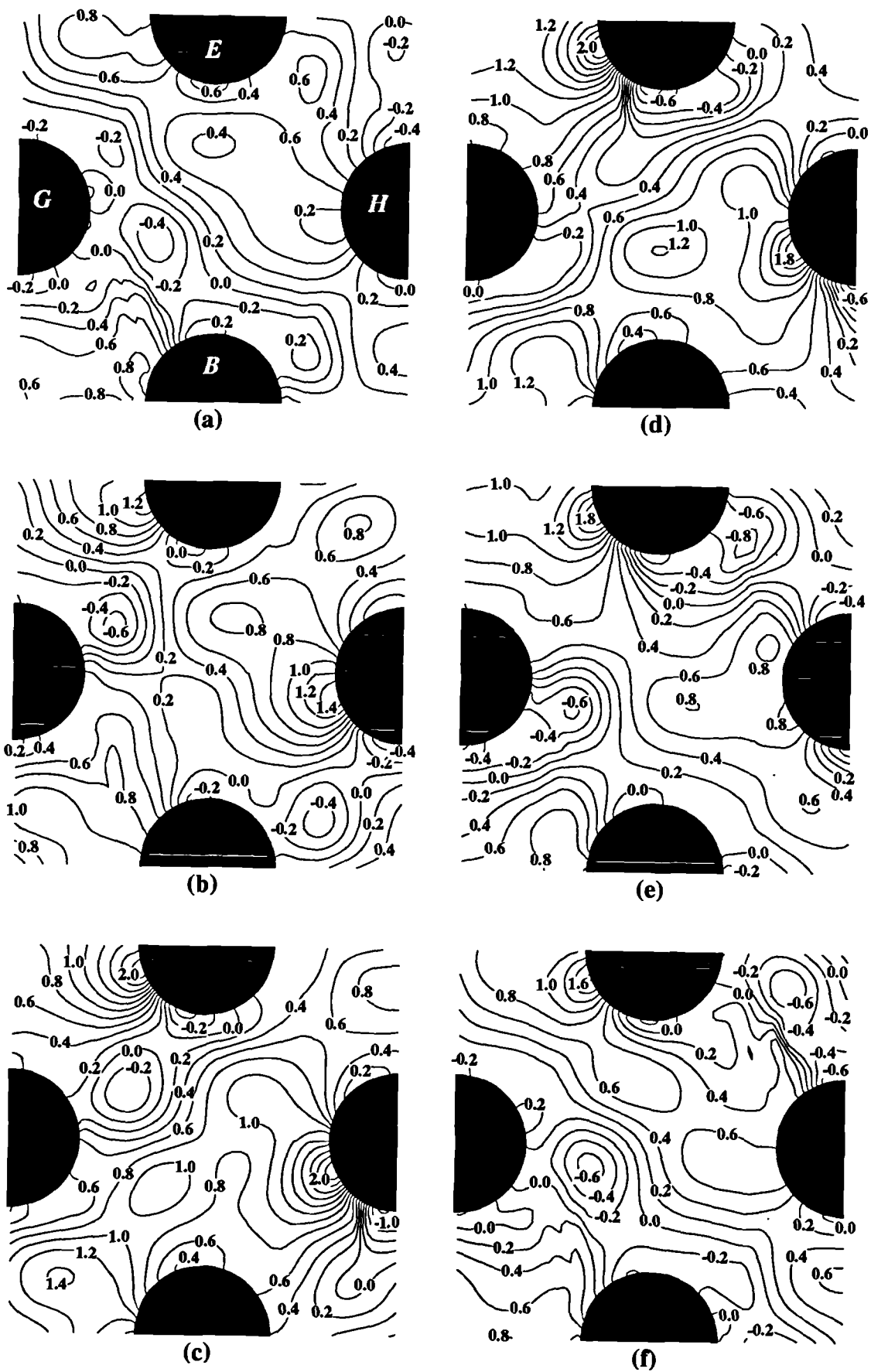


Figure 89. Pressure contours (Pa), rotated square, $Re = 300$.

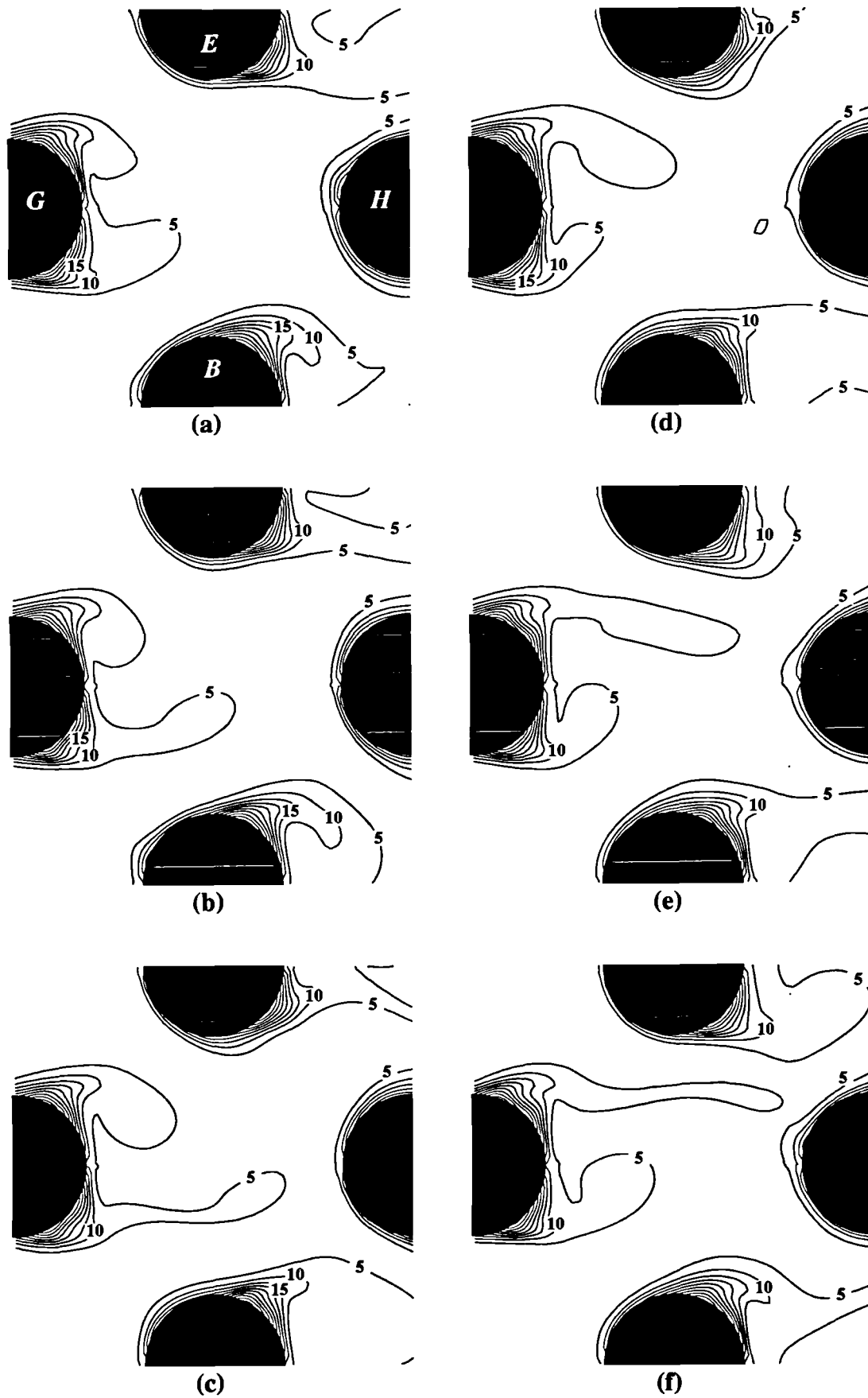


Figure 90. Temperature contours (°C), rotated square, $Re = 300$.

9.2.2 Time average quantities

Figure 93 shows Sh vs. Re . Sh were computed from the cross-wise fluctuations (in v) at the monitor point location. Figure 94 is a comparison of the results of this Chapter, in terms of time-average and space-average Eu , to the results presented in Chapter 8, assuming the flow to be steady. Figure 95 is a similar comparison for heat transfer with $\overline{St}_H$ being plotted as a function of Re . It can be seen that the slopes of both curves are less steep than those obtained for steady flow. Tables 13 and 14 are a summary of the data displayed in Figs. 93 to 95 inclusive. Values of Sh , Eu and $\overline{Nu}_H$ are in all cases averaged over the last half of each simulation. The quantity, $f\tau_x$, is the fraction of a cycle by which the upstream cross-wise transient leads the downstream value across a single row, being computed from the v -values at the two monitor points (midway between $A-B$ and $B-C$).

Table 13. Summary of results, in-line square.

Re	Sh	$f\tau_x$	Eu	$\overline{Nu}_H$
100	0.0882	0.44	0.317	3.92
300	0.0920	0.41	0.171	6.73
1 000	0.116	0.55	0.104	17.3
3 000	0.101	0.52	0.193	40.8

Table 14. Summary of results, rotated square.

Re	Sh	$f\tau_x$	Eu	$\overline{Nu}_H$
30	-	-	0.923	6.38
100	0.270	0.73	0.568	8.87
300	0.353	0.89	0.366	15.5
1 000	0.310	0.76	0.248	30.0
3 000	0.257	0.82	0.156	56.6

9.2.3 Overall drag and heat transfer

Figures 96 and 97 show Eu vs. t^* at various Re . Values shown are the difference between spatially averaged pressures along a line bisecting $A-B$ and those across along a line bisecting $B-C$ [Fig. 80] for the in-line bank, or in the case of the staggered bank, one-half of this value. It can be seen that as Re increases, the mean pressure drop decreases, and the magnitude of the fluctuations increases significantly.

Figures 98 and 99 shows the instantaneous lift and drag coefficients, c_L and c_D for the in-line bank. These are defined as the net force due to lift or drag per unit projected area, normalised with respect to $\frac{1}{2}\rho u_m^2$, and are computed as,

$$c_D = \frac{1}{\rho u_m^2} \left(\int_0^\pi p \cos \phi + \tau_w \sin \phi d\phi + \int_\pi^{2\pi} p \cos \phi - \tau_w \sin \phi d\phi \right) \quad (9.9)$$

$$c_L = \frac{1}{\rho u_m^2} \left(\int_0^\pi -p \sin \phi + \tau_w \cos \phi d\phi - \int_\pi^{2\pi} p \sin \phi + \tau_w \cos \phi d\phi \right) \quad (9.10)$$

where p and τ_w are the appropriate pressure and shear stress values at the wall. Values of c_L and c_D displayed in Figs. 98 and 99 are computed around test cylinder B [Fig. 80a] for the in-line bank at various Re . Figures 100 and 101 are similar plots of c_L and c_D vs. t^* , this time for Cylinder G in the rotated square tube bank [Fig. 80b].

Figures 102 and 103 show $\overline{Nu}_H$ vs. t^* for in-line and staggered geometries. $\overline{Nu}_H$ is based on an overall heat transfer coefficient defined according to Eq. 8.22 as,

$$\bar{h} \equiv \frac{Q}{\frac{d}{2} \int_0^\pi (t_w - t_{ref}) d\phi} \quad (9.11)$$

The reference temperature, t_{ref} is taken here as the time-average bulk temperature in the upstream inter-tube space. This fixed value is easily computed from the presumed inlet temperature and wall flux.

9.2.4 Local coefficients

Figures 104 and 105 show the time behaviour of the instantaneous pressure coefficient, c_p , as a function of the angle ϕ , for in-line and staggered tube banks at $Re = 300$. The pressure coefficient is defined according to,

$$c_p \equiv 1 - \frac{\bar{p}(0) - p(\phi)}{\frac{1}{2}\rho u_m^2} \quad (9.12)$$

where the reference pressure is taken as the time-average value (over the second-half of the simulation) at $\phi = 0$. The curves (a) to (e) in Fig. 104 correspond to the data shown for the six captions of Figs. 83 to 86; data is for the downstream cylinder B , the angle ϕ being taken in a clockwise sense from the front-leading edge. Similarly the data shown in Fig. 105 corresponds to cylinder G in Figs. 87 to 90.

Figure 106 shows the instantaneous values of the friction coefficient for the in-line tube bank. c_f , is defined according to

$$c_f \equiv \pm \mu \frac{\partial u}{\partial y} \bigg|_w \frac{1}{\frac{1}{2} \rho u_m^2} \quad (9.13)$$

where the sign of c_f is taken positive for $0^\circ < \phi < 180^\circ$ and negative for $180^\circ < \phi < 360^\circ$. Figure 107 shows values $Nu_H/\overline{Nu}_H$, again for the in-line bank, $Re = 300$. Nu_H is based on a local heat transfer, h , defined by,

$$h \equiv \frac{q_w}{(t_w - t_{ref})} \quad (9.14)$$

where t_{ref} is the theoretical mean bulk temperature upstream of the cylinder, as defined above.

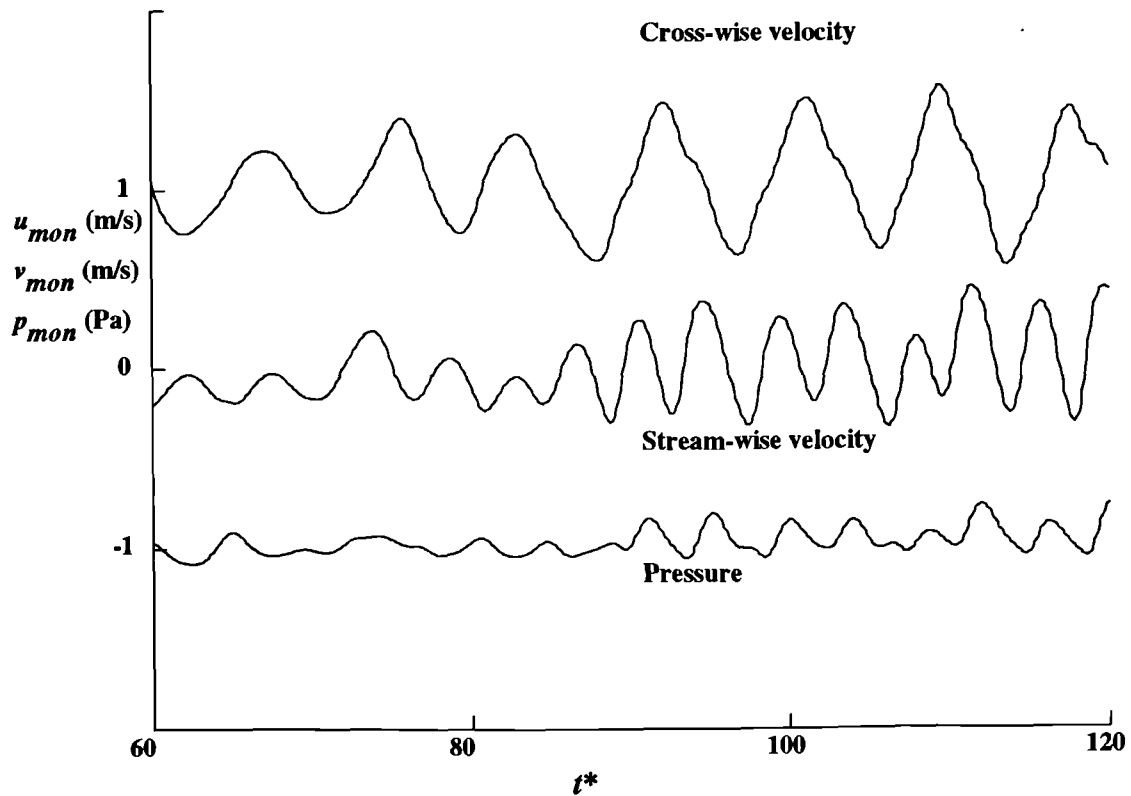


Figure 91. Monitor-point values, in-line square, $Re = 1\,000$.

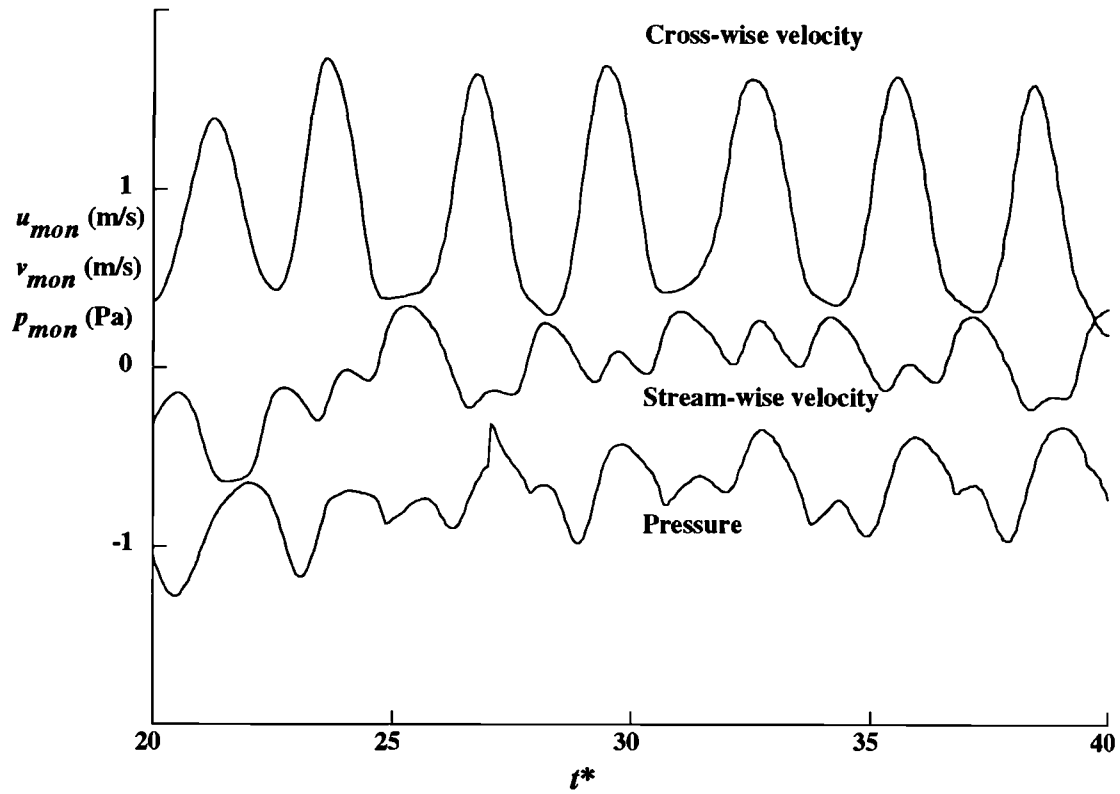


Figure 92. Monitor-point values, rotated square, $Re = 300$.

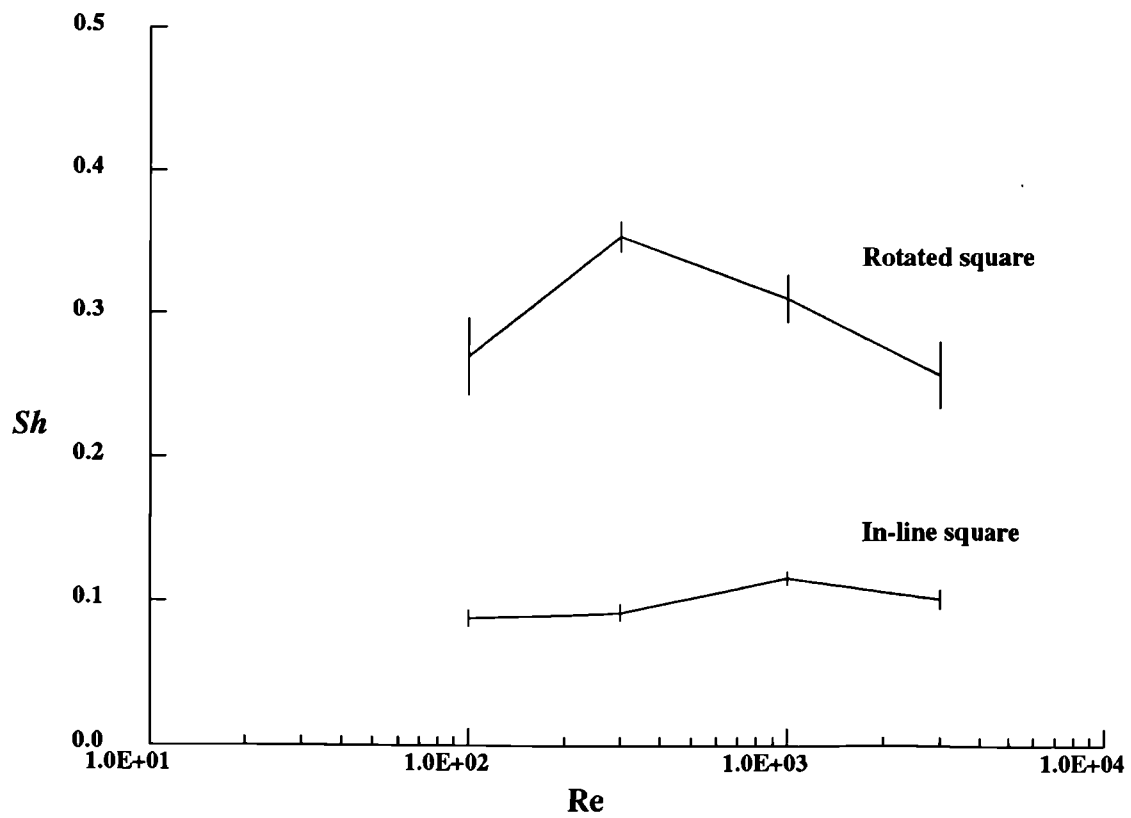


Figure 93. Sh vs. Re , in-line and staggered banks.

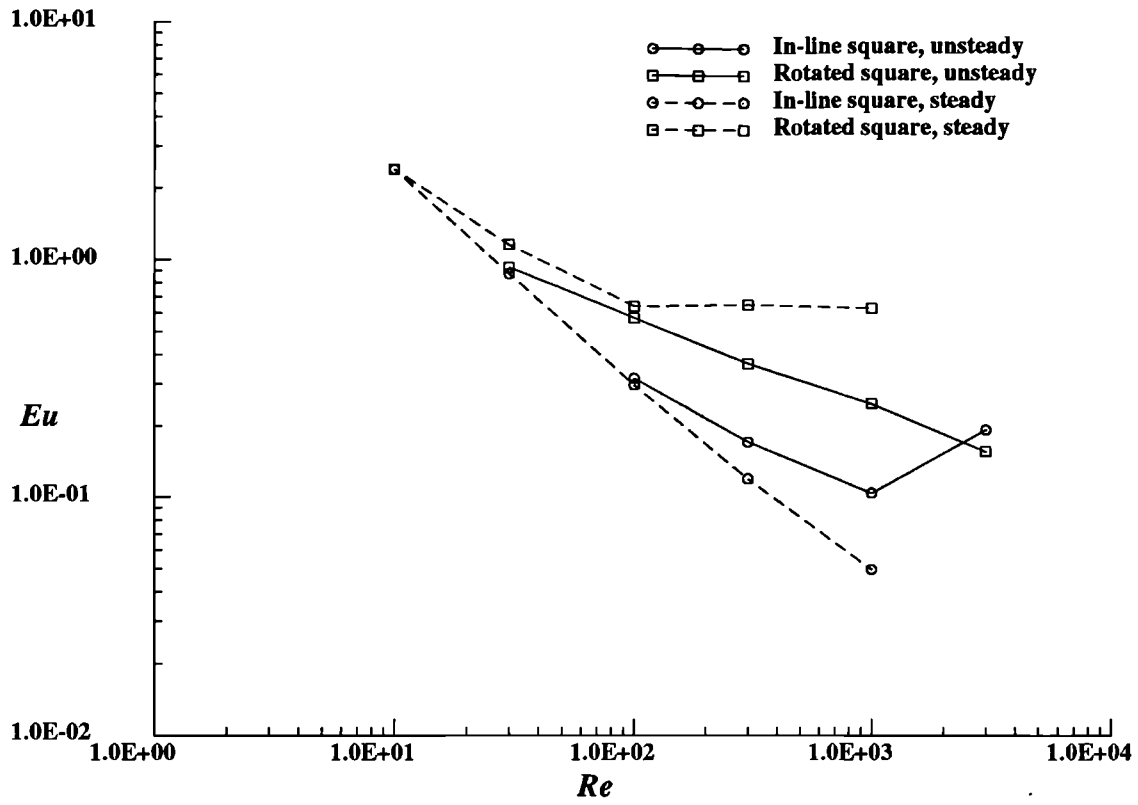


Figure 94. Eu vs. Re , in-line and staggered banks.

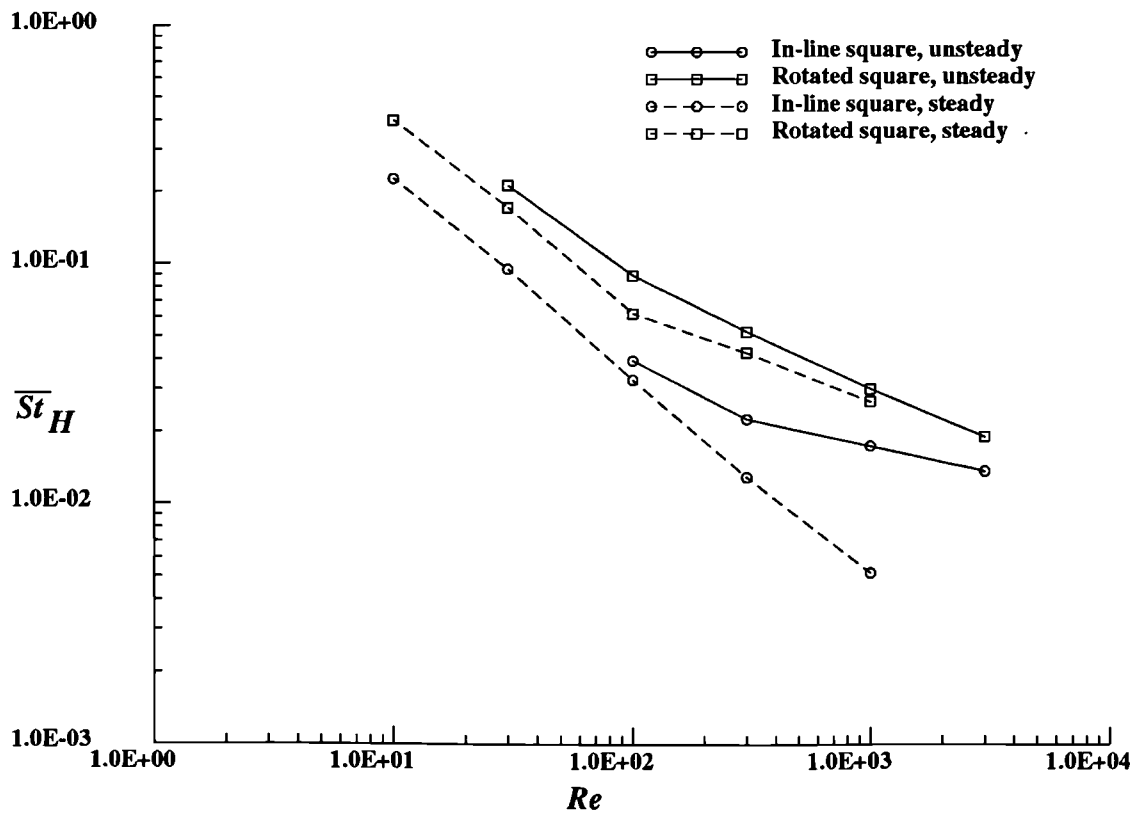


Figure 95. $\overline{St}_H$ vs. Re , in-line and staggered banks.

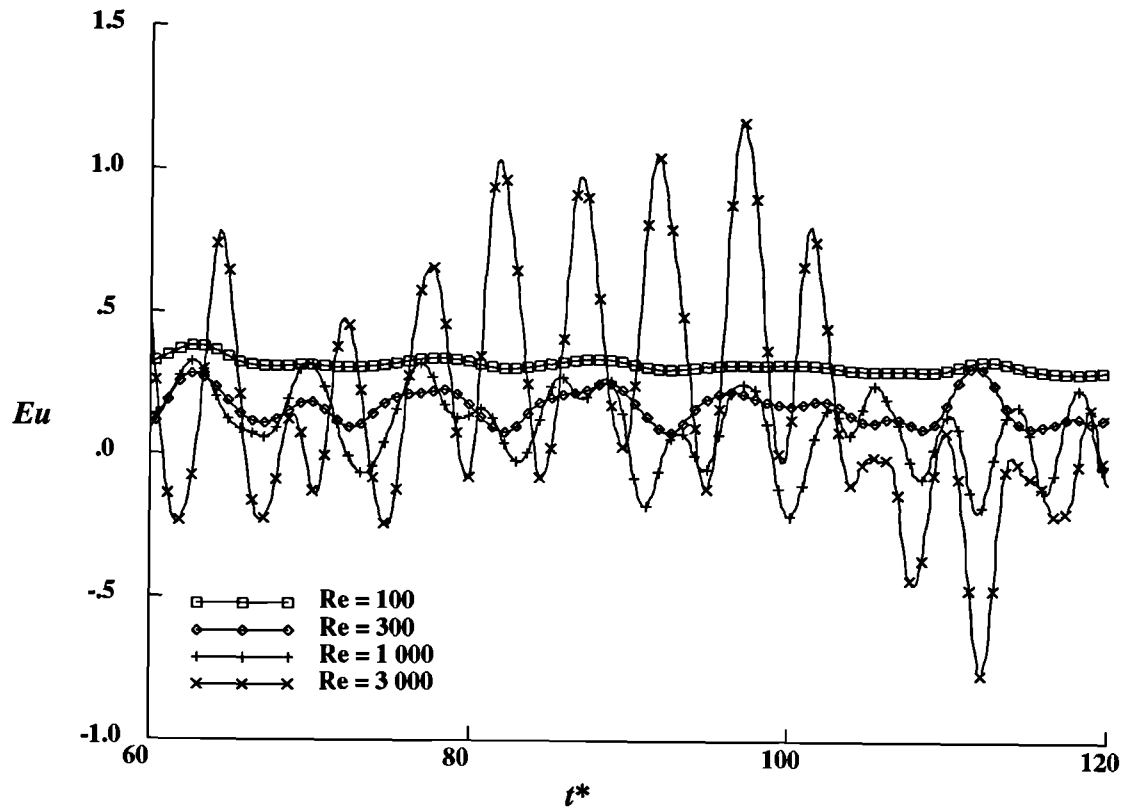


Figure 96. Eu vs. t^* , in-line square.

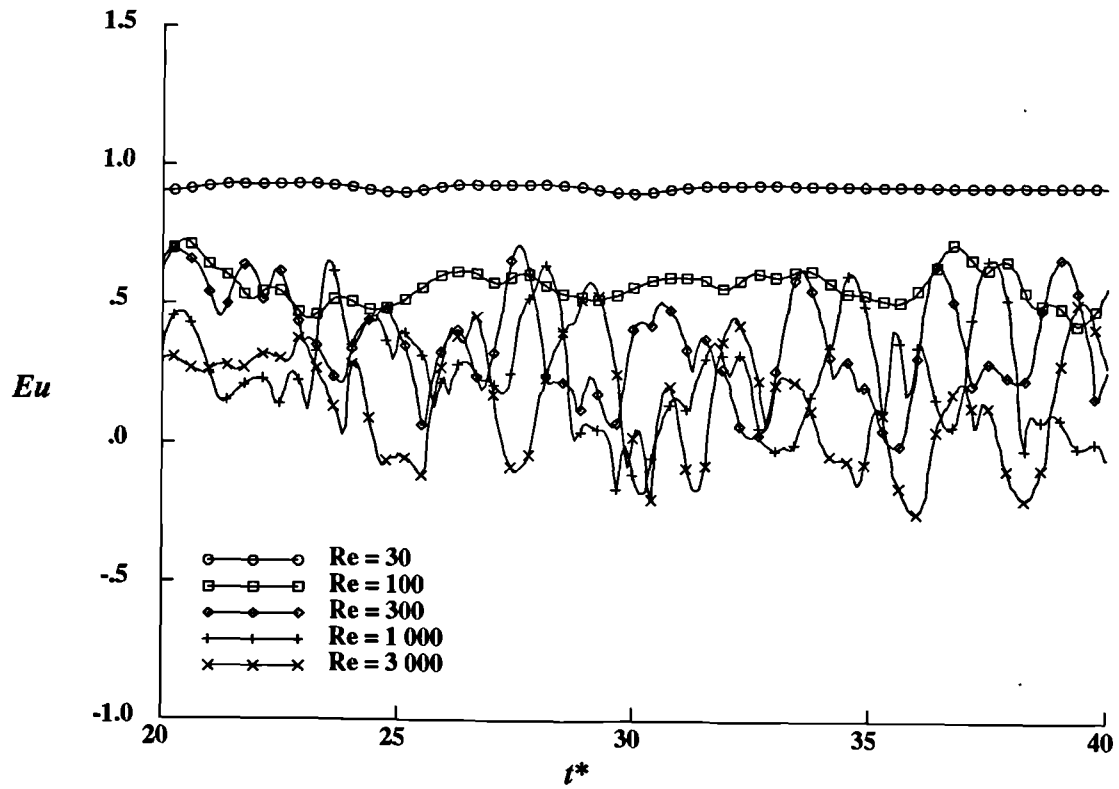


Figure 97. Eu vs. t^* , rotated square.

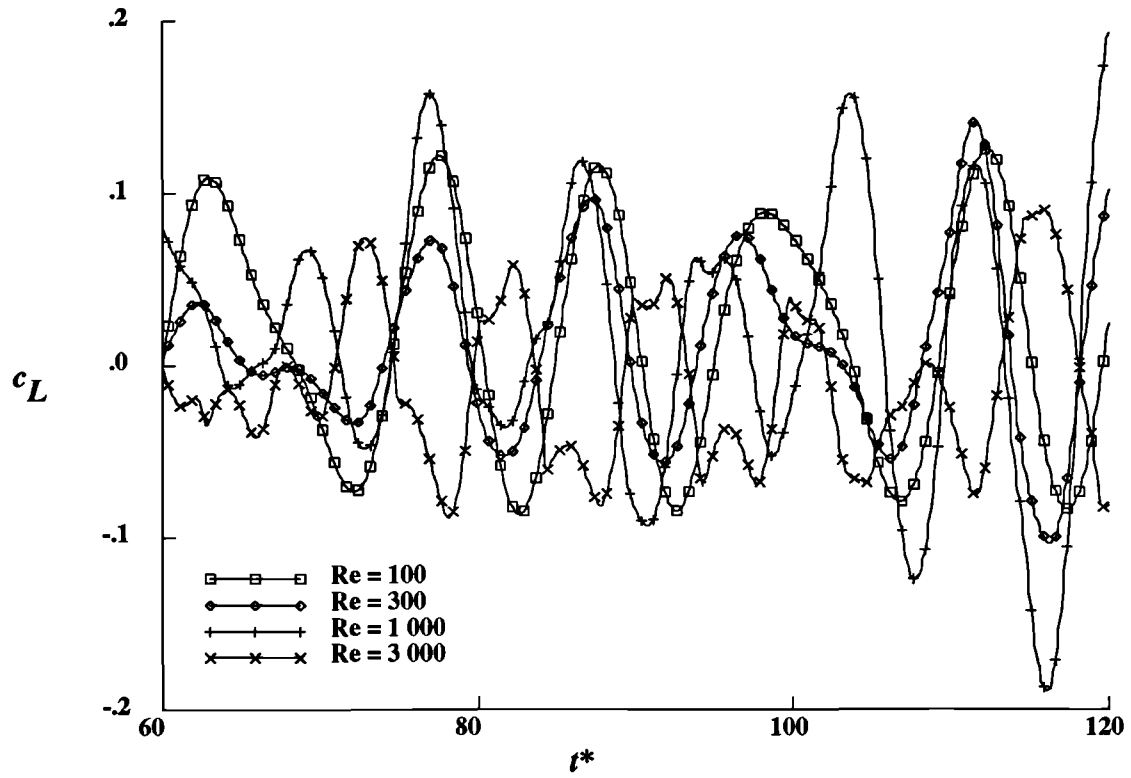


Figure 98. c_L vs. t^* , in-line square.

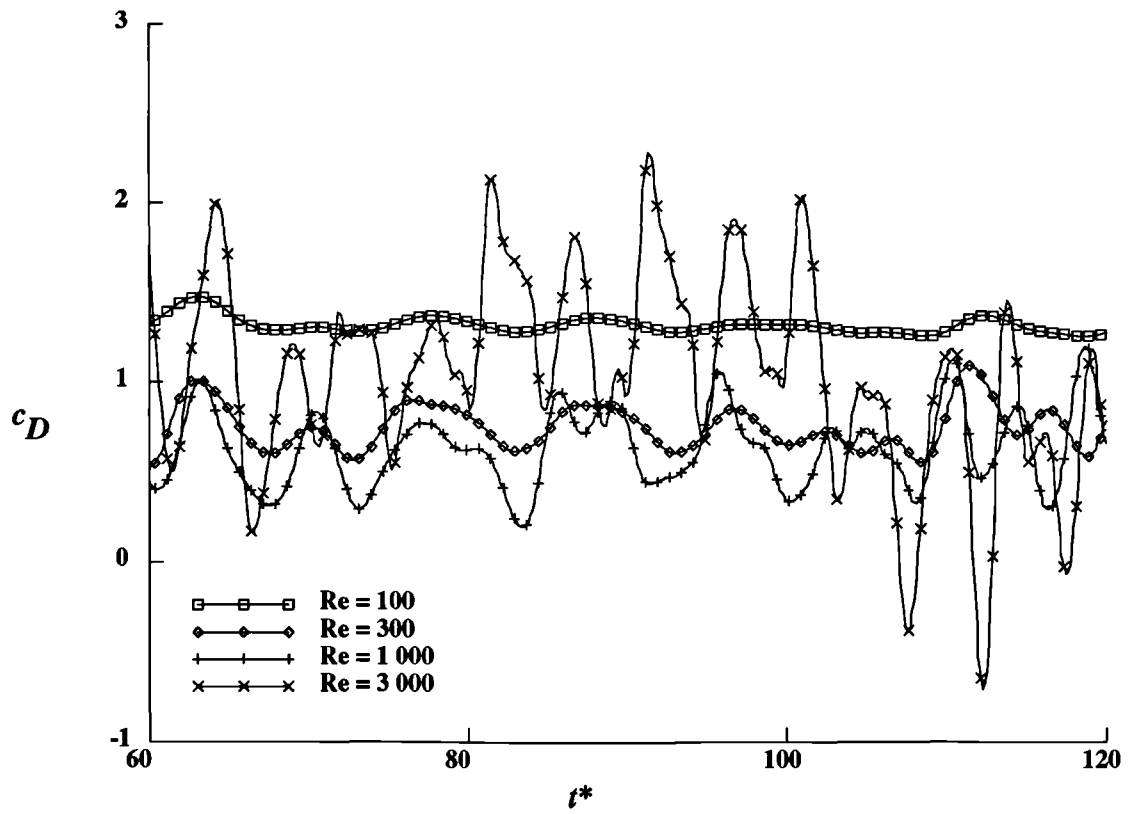


Figure 99. c_D vs. t^* , in-line square.

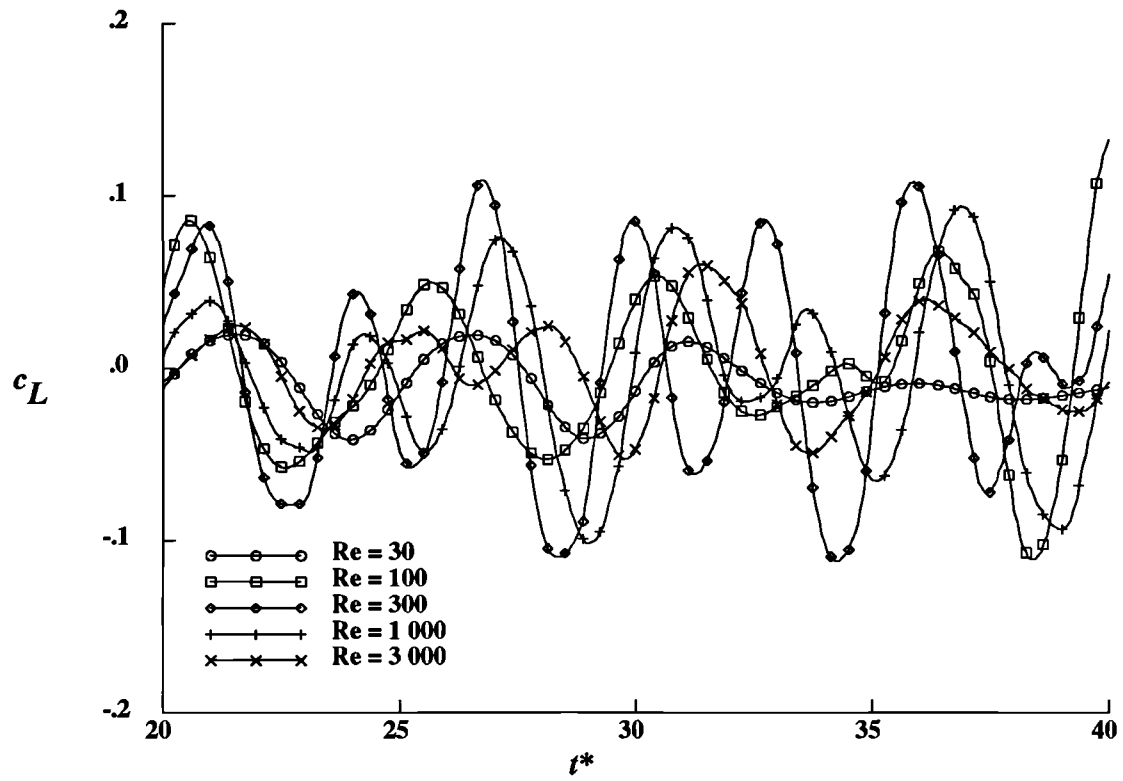


Figure 100. c_L vs. t^* , rotated square.

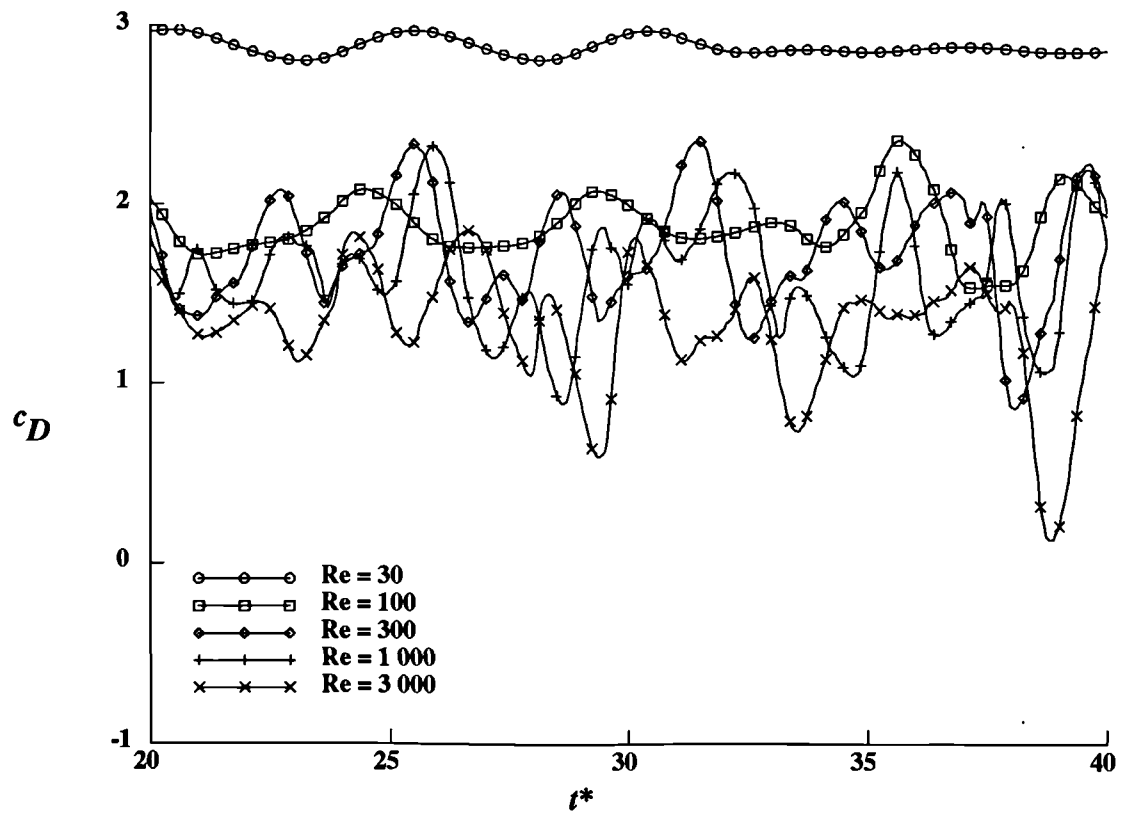


Figure 101. c_D vs. t^* , rotated square.

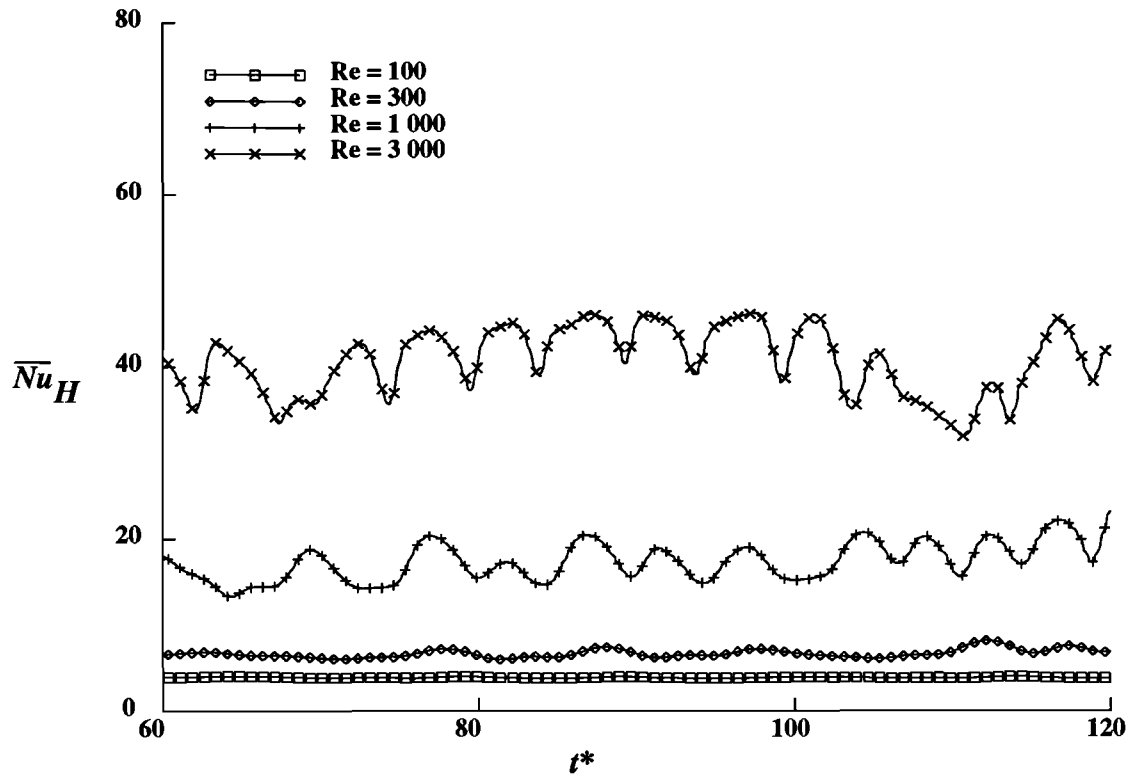


Figure 102. $\overline{Nu}_H$ vs. t^* , in-line square.

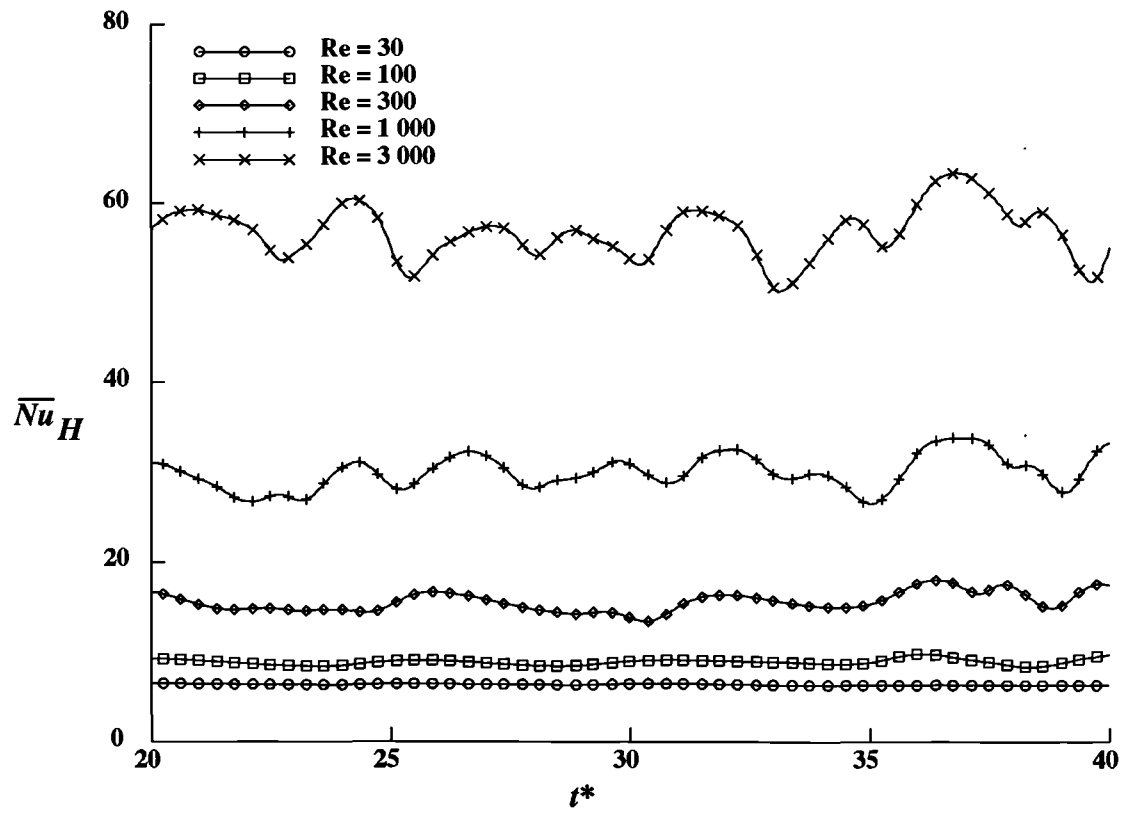


Figure 103. $\overline{Nu}_H$ vs. t^* , rotated square.

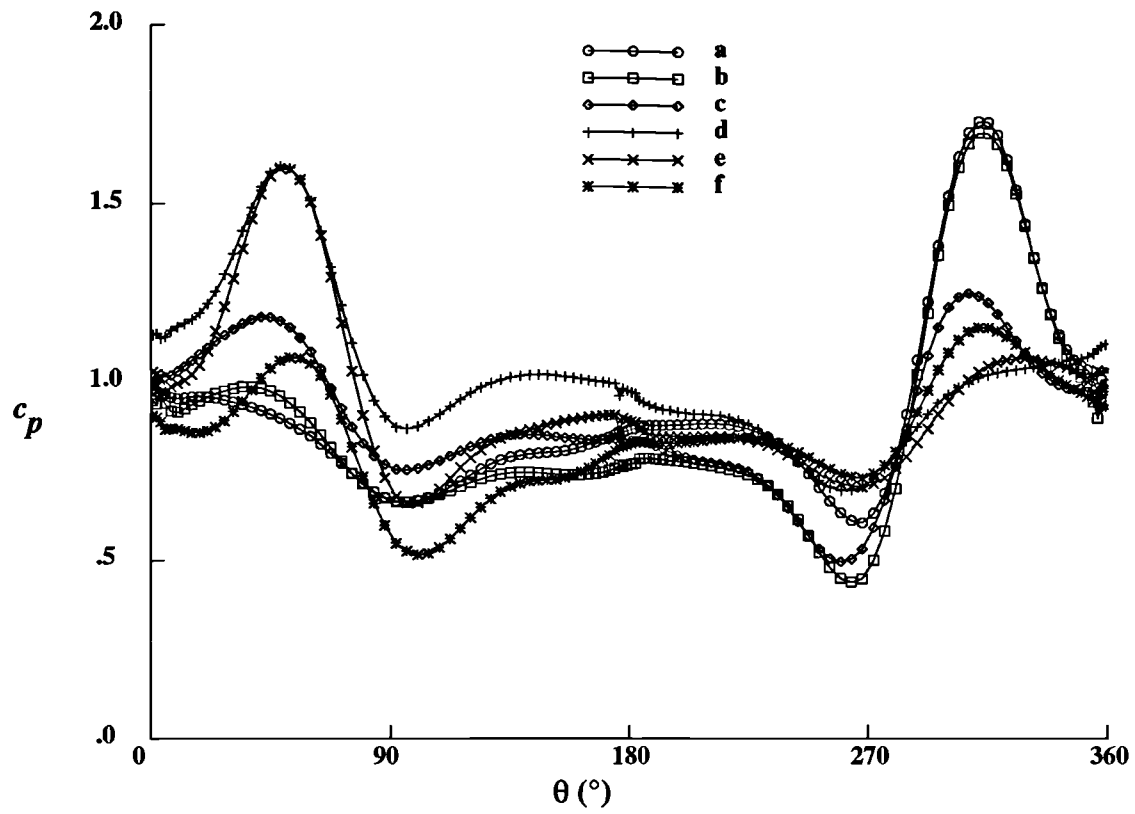


Figure 104. c_p vs. t^* , in-line square, $Re = 300$.

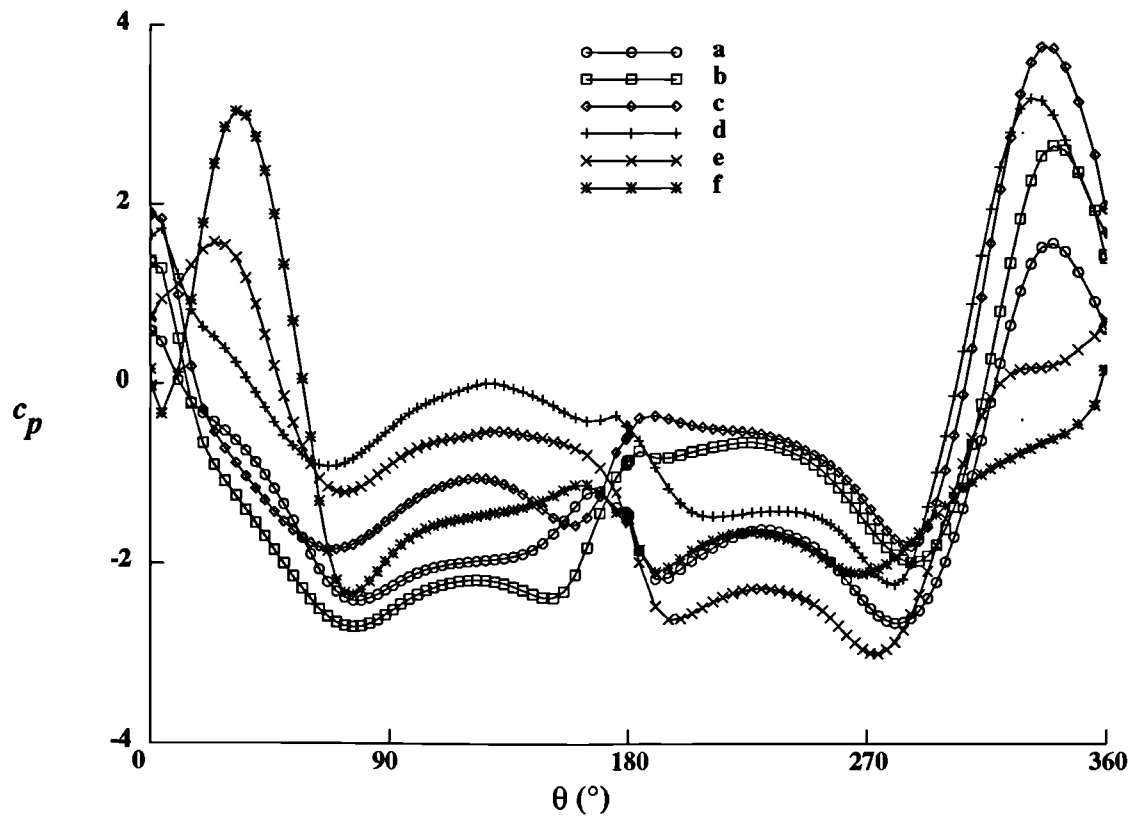


Figure 105. c_p vs. ϕ , various t^* , rotated square, $Re = 300$.

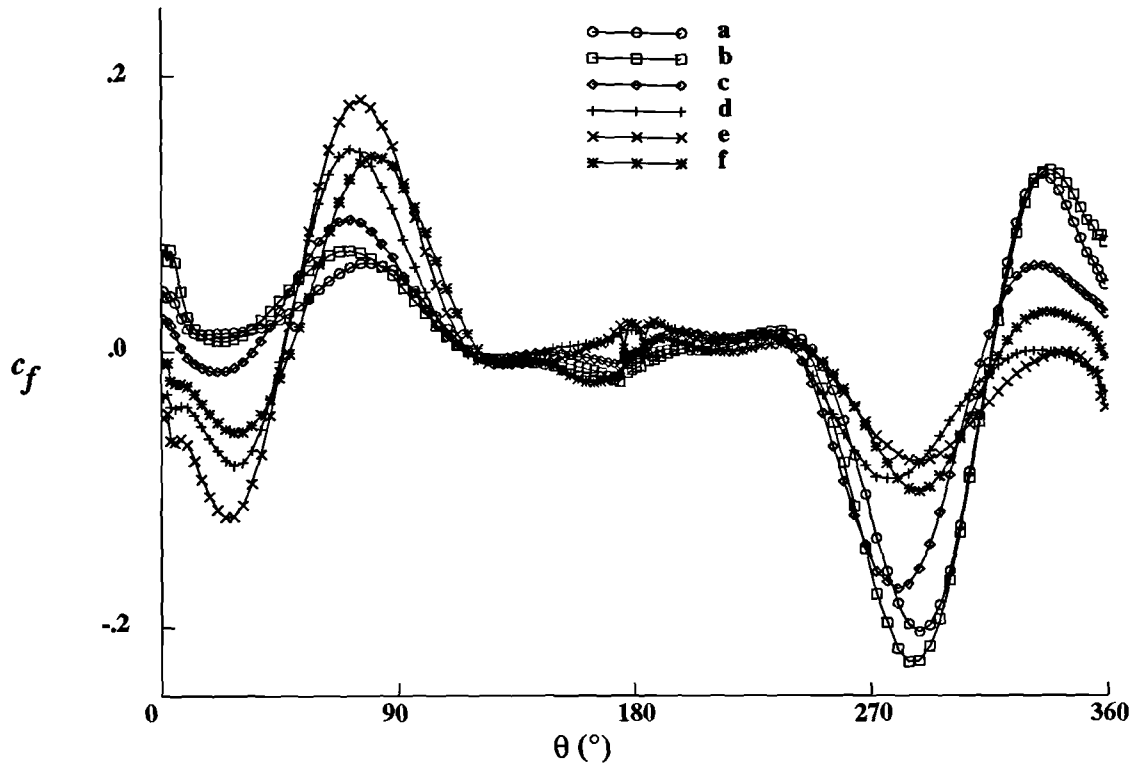


Figure 106. c_f vs. ϕ , various t^* , in-line square, $Re = 300$.

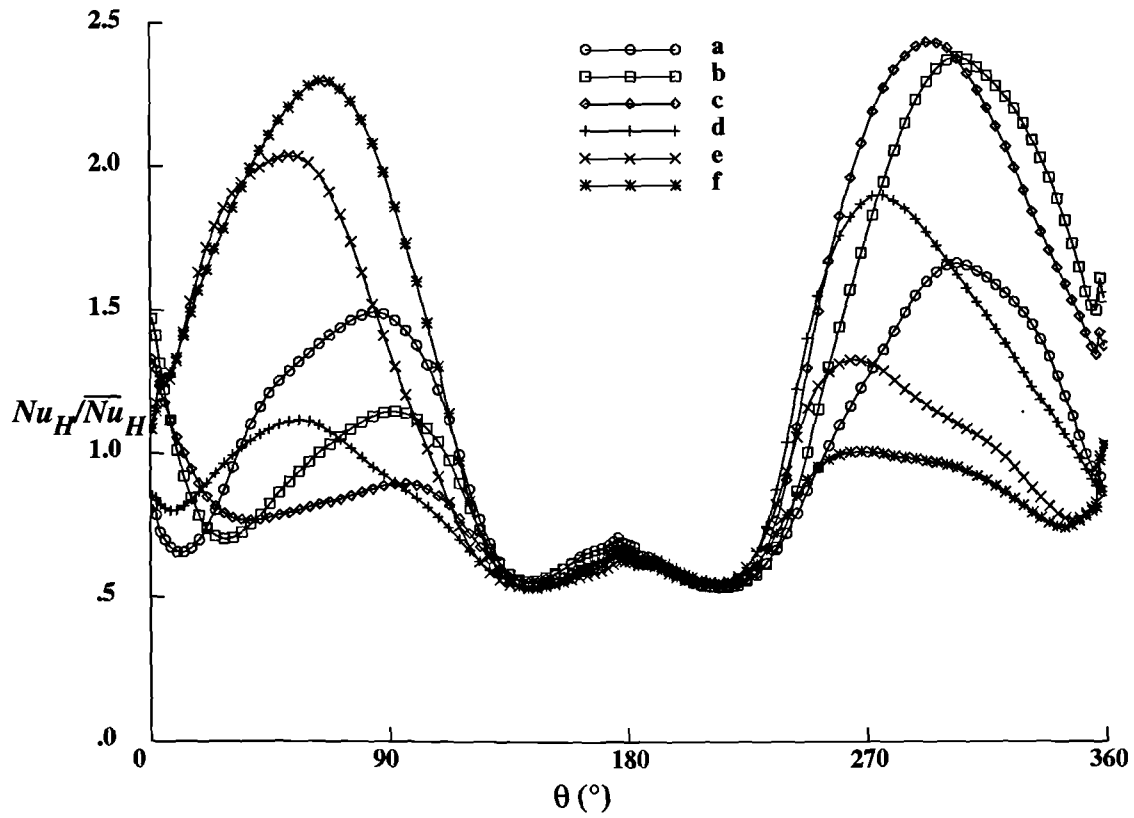


Figure 107. $Nu_H / \overline{Nu}_H$ vs. ϕ , various t^* , in-line square, $Re = 300$.

9.3 Discussion

9.3.1 Qualitative description

The results show that at low Re , any applied disturbance is rapidly damped out, after a few cycles. At higher Re , the downstream response to upstream perturbations, is substantially larger than the original disturbance, and as a result of the feedback mechanism, the disturbance does not decay. There is thus seen to be a ‘transistor’ or ‘triode-like’ effect, resulting in a net gain in the fluctuating component of the velocity field due to the constant stream-wise motion: By supplying sufficient energy via the mean flow, lateral perturbations are amplified. For staggered banks this effect was immediately apparent [see Fig. 82]. For in-line banks the phenomenon is more equivocal. However when the measured response is re-applied, the lateral fluctuations increase, albeit in a somewhat haphazard fashion.

Figures 83 and 84 show the time-dependent velocity field for an in-line tube bank in the form of vectors and stream-lines: (a) At the start of the cycle a counter-clockwise vortex has formed between the cylinders [lower-half Fig.84a]. (b) The counter-clockwise vortex moves into the central inter-tube zone, and in doing so (c) spawns the formation of a small clockwise vortex behind the upstream cylinder. (d) This new clockwise vortex elongates and expands to fill the inter-tube region displacing the original counter-clockwise vortex in its vanguard. The first vortex starts to decay as a result of friction forces. (e) The clockwise vortex fills the entire inter-tube space (f) eventually becoming detached from the upstream cylinder, and thereby inducing the formation of another counter-clockwise vortex on the lower-side of the upstream cylinder.

Further insight into the nature of the flow was obtained from the computer-generated animation sequences (not shown): As each vortex develops, fluid particles near the cylinder wall enter the inter-tube zone from the free-stream, reinforcing the direction of rotation, and increasing the size of the vortex. As each enlarged vortex is subsequently displaced downstream, and eventually dissipated to one or other side of the front of the subsequent cylinder, fluid particles are ejected back out into the main stream. The fast-moving free-stream undulates, depending on the direction of rotation of the vortices, i.e. there is a shear-layer instability between the wake, and the main flow. For $s/d = 2$, only one vortex is ever fully-formed in the inter-tube space, although up to three partially formed vortices of opposing spin may be present momentarily: Tests showed that for larger values of s/d , the behaviour is more like that for single cylinders, in that several vortices co-exist in the wake at all times. The latter are also apparent in the early flow visualisation studies of Wallis (1939a,b).

Figures 87 and 88 shows the flow field as a function of time for the rotated square tube bank at $Re = 300$. The details of the vortex development are qualitatively similar to the in-line case although it is evident that there are also some differences: (a) A clockwise vortex has developed at the upper rear-side of upstream cylinder G . Below and downstream of this first vortex, a second counter clockwise vortex can also be seen. (b) The clockwise vortex grows, probably as a result of the favourable velocity gradient in the shear-layer (between the wake and the free-stream) tending to increase angular momentum due to fluid friction. (c) This not only increases the size of the vortex, but also moves it into the wake, to the rear of cylinder G . As the vortex moves across the centreline, it encounters an opposing shear-layer (on the under-side of the cylinder) entraining it, and decreasing angular momentum. This results in a region of counter-clockwise vorticity being created between the vortex and the main stream. (d) The clockwise vortex is displaced downstream and the new clockwise vortex begins to grow behind the rear-lower side of the cylinder. The (clockwise) vortex moves with the main flow and (e) is destroyed as it passes through the minimum cross-section of the subsequent row

B-E. By (f) the cycle is complete. Owing to alternate quantities of clockwise and counter-clockwise momentum being imparted, the wake is 'S'-shaped, and switches from side-to-side, in a ciliary fashion (something which is only really apparent from the use of animated colour vector graphics). The attachment point at the front of the downstream cylinder H also oscillates as the streams stick to either side of the wall resulting in a Coanda-like effect. The reader will note that the stream-line illustrations shown in Fig. 84 and 88 were generated using the iterative scheme, discussed in Chapter 7. Flow-fields were found to be essentially the same at higher Re : For the in-line bank, the vectors in the vortex increase in size as Re increases, but apart from this the vector and stream-line plots were found to be remarkably similar over a wide range of Re for both tube banks. Only minor differences were noted regarding the size of vortex, number present at any time and so forth.

Figures 85 and 89 show the pressure field (Pa) as a function of time. Careful inspection of Fig. 85 (in-line), reveals that in contrast to steady flow [Chapter 8], there is only ever one high pressure zone on one or other side of the cylinder near the front, where flow attachment occurs. This happens on the opposite side to whichever one upon which a given vortex is being destroyed. A series of low pressure zones are apparently associated with the vortices themselves (for the most part away from the actual centre of the vortex). The low pressure zones move downstream and neutralise the frontal high, just as the vortex itself is destroyed. At the same moment a new high is established on the other side of the cylinder where the flow is attaching.

The same trends are exhibited for staggered geometry [Fig. 89], the single-frontal pressure maximum oscillating as above. However there are now multiple extrema (both high and low) in the inter-tube space, and the pattern is more complicated, probably due to the wake switching. For both geometries, the pressure maxima at the sides of the cylinder (around $\phi = 90^\circ$), also oscillate, although only a little. The transient pressure fields are far more complex than those that were observed for steady flow [Figs. 42 and 43].

One might speculate as to the influence of the pressure field on the transient nature of the flow: For potential flow [Chapter 4], it was shown that the pressure is a relative maximum along the centre-line between cylinders in the wake regions, at $\phi = 0^\circ$ and 180° . For both in-line and staggered geometries the magnitude of the pressure is substantially smaller in the free stream outside the wake [see Figs. 9 and 10]. Roughly similar trends were also observed for steady viscous flow. In staggered tube banks, in particular, adjacent low pressure zones at the sides of alternate rows, corresponding to $\phi = \pm 90^\circ$, suggest that the wake may be in a state of unstable equilibrium, i.e. the lateral pressure gradient may assist in propagating an initial disturbance. On the other hand the presence of an overall favourable pressure gradient in the stream-wise direction tends to impede vortex formation, possibly explaining why such phenomena occur more readily at lower Re in single cylinders than in tube banks.

Figures 86 and 90 show the fluctuating temperature fields ($^\circ\text{C}$). Inspection of Fig. 86 reveals that the formation of the vortices has a significant effect upon the temperature field: As each vortex becomes fully-developed, it detaches and occupies the central zone between the two cylinders [see Fig. 84b]. At this time there is relatively strong cross-flow from the attachment point at the front of the downstream cylinder *B*. The effect is to pull (cold) fluid at the free-stream temperature in, and across the (hot) downstream cylinder *B*, resulting in a trough in the temperature field near the lower front side of cylinder *B* [Fig. 86b]. This sharp temperature gradient quickly collapses as the vortex loses angular speed and is dissipated on the lower front side of cylinder *B*, [Fig. 86c]. It is seen that as the subsequent (clockwise) vortex fills the inter-tube space another tongue of (cold) fluid is drawn from the attachment point on the upper side of cylinder *B* [Figs. 86d,e]. It is thus apparent that there is substantial gross mixing of free-stream and inter-tube fluids due to the alternate swirling motion of the vortices. In the staggered arrangement [Fig. 90], most of the mixing appears to occur away from the cylinder in the wake; under these circumstances the observed effect is of corpuscles of (hot) fluid being shed from alternate sides of the upstream cylinder. Because only a few contours

are shown, the temperature field appears much simpler than it actually is. As the wake switches from side to side, the temperature of the fluid washing the downstream cylinder changes. However, even if the fluid temperature is constant, this washing action will enhance heat transfer from the front of the cylinder, due to increased convection. The reader should appreciate that the use of contours to illustrate stream-lines or temperature fields is inherently subjective.

9.3.2 Frequency of oscillations

Inspection of Figure 93 and Tables 13 reveals that measured Sh were of the order of 0.09 for the in-line arrangement at low Re , increasing only very slightly to around 0.12 at $Re = 1\,000$. For the staggered geometry [Table 14] Sh were larger, in the range 0.26 to 0.35, with substantially more variation. The vortex shedding frequency, f , was calculated from the cross-wise v -velocity at a single spatial reference position, corresponding to the location of the monitor points [Figs. 91 and 92]. Because only a few cycles were used in computing these values, they should be considered approximate. The error bars shown in Fig. 93 are computed from the standard error of the mean, based on the second half of the simulation, typically 6 to 8 complete cycles.

Interpolation of the correlation curves in the book by Zukauskas et. al. (1988b), based on the experimental data of several authors, suggests that the Sh values should be around 0.25 (in-line square) and 0.34 (rotated square). It would appear that while the results of this study are in good agreement with existing experimental data for staggered banks, the numerically-based Sh measurements are only 40% of those recorded by Zukauskas et. al. (1988b).

Inspection of the monitor point data [Figs. 91 and 92] reveals that the cross-wise v -velocity, oscillates in a repeatable sinuous fashion, at frequency f . There are two pairs of

maxima and minima in the corresponding fluctuating component of the stream-wise u -velocity. i.e. the frequency in the stream-wise direction is apparently twice that in the cross-wise direction, so that Sh based on $\vec{u}$ is anisotropic. The magnitude of the u -velocity fluctuations, while less than those of the v -velocity, are quite significant. Depending on the relative strength of the two components the pressure trace is at either f or $2f$ (or more usually a combination of the two). It seems likely that any experimental data, particularly those based on pressure measurements, would reflect this fact. It is interesting to speculate as to the presence of higher harmonics, and the influence of the pressure fluctuations on the flow field (coupling). A power spectral analysis of the transient would almost certainly display two or more frequencies, a fact which has been observed by experimenters (e.g. Price et. al., 1987).

In both in-line and staggered tube banks, the amplitude of the lateral v -fluctuations was observed to be quite large; at $Re = 100$, the peak oscillations were around 20% of the bulk stream-wise velocity for rotated square tube banks. For $Re \geq 300$ peak oscillations increased to more than one-half of the mean value. Above this, the amplitude of the v -fluctuations remained essentially constant, repeatable and sinusoidal-like. In-line tube banks displayed a similar behaviour, however the Re required for the amplitude to reach the maximum value, again more than one-half of u_m was somewhat higher. In addition to appearing to be less susceptible to periodic behaviour than staggered arrangements, transient behaviour in-line tube banks was generally more orderly over a wider Re range. In both geometries, the lateral fluctuations tend to be stable at high Re , but somewhat erratic at low Re . The converse is true for the stream-wise and pressure fluctuations: The magnitude of the monitor-point stream-wise u -velocity, and pressure fluctuations increased in strength over the entire Re range, with the pressure signal, in particular, becoming less stable at higher Re . It must be admitted that the monitor data exhibited in Figs 91 and 92 are among the most consistent of these data. At other times, eccentric behaviour was observed. Fluctuations of the value of the mean, variations in the period,

and larger-scale group-envelope type behaviour, typically associated with the superposition of two phases at different frequencies were all observed.

Table 13 shows the quantity, $f\tau_x$, which represents the time difference between the upstream and downstream cross-wise v -velocity fluctuation, expressed as a fraction of a cycle. For the in-line tube bank, the upstream and downstream oscillations were observed to be very close to $\frac{1}{2}$ cycle (or 180°) out-of-phase with each other, regardless of Re . This is consistent with the observations of Weaver and Abd-Rabbo (1985). It can be seen that the upstream transient precedes the downstream value by 0.44 of a cycle at $Re = 100$, increasing to an maximum of 0.55 at $Re = 1\,000$. Inspection of Table 14 for the staggered bank reveals that the apparent upstream phase-lead was somewhat greater; between between 0.73 and 0.89 of a cycle and more Re dependent. The phase difference initially increases, then subsequently decrease, as also apparently does Sh . Values of $f\tau_x$, exhibited in Table 14, were obtained from the phase difference across two complete modules (by adding 1 and dividing by 2). The reader will note that, even though the cylinders are offset, the phase across two rows is still twice that across the diagonal, provided that τ_y is zero.

9.3.3 Lift and drag

Figure 94 shows that the time-average Eu converges to the steady-state value in the low Re range, with reasonable agreement. In view of the many assumptions made when applying the boundary conditions here, it would be surprising for agreement to be perfect. All of the correlations cited in Chapter 8 (ESDU, 1980, HXDH, 1983) are based on extrapolation for $s/d \geq 2$, and since there are no actual experimental data of which this author is aware, these are not included here.

Figures 96 and 97 show Eu as a function of time for in-line and staggered geometries, respectively. (N.B. in the interests of clarity, a symbol is plotted every 6th time step in Figs. 96 to 103). Inspection of Fig. 96 reveals that, at $Re = 100$, the average pressure drop between two modules is 0.317 with a population standard deviation of 0.018. As Re increases the mean value decreases, as is shown in Table 13, while the periodic component becomes substantially larger. The population standard deviations at $Re = 300$, 1 000, and 3 000 are 0.055, 0.130, and 0.404. The apparent increase in the mean value of Eu at $Re = 3 000$, is probably due either to the simulation being of insufficient length, or to the Re range of usefulness where free-stream turbulence is can no longer be ignored, being exceeded. Inspection of Fig. 96 reveals that at high Re , there are times where there is instantaneously an overall favourable pressure gradient. It can also be seen that as Re increases, the oscillations change from being primarily at f to $2f$, as discussed above.

For the staggered geometry, the periodic behaviour is more random, as might be anticipated from the complex nature of the pressure field [Fig. 89]. It is again seen that the fluctuating component of Eu increases as the mean value decreases: The mean and standard deviation are 0.568 and 0.063 at $Re = 100$, 0.366 and 0.185 at $Re = 300$, 0.248 and 0.194 at $Re = 1 000$ and 0.156 and 0.202 at $Re = 3 000$. Thus it is clear that in both in-line and staggered geometries, the fluctuating component of the pressure field is large when compared to the steady-state value at higher Re . The reader will note that the data of Fig. 96 represent the average pressure drop across a single module, whereas those of Fig. 97 are based upon one-half the pressure drop across two modules. These data may therefore reflect the difference in phase.

Figures 98 and 99 show c_L and c_D vs. time for the in-line array, while Figs. 100 and 101 are similar plots for the rotated square bank. It can be seen that for both geometries c_L oscillations are sinusoidal-like. For the rotated square arrangement it can be seen that, for $Re = 30$, the c_L fluctuations have barely died out by the end of the simulation. As

Re . increases, the c_L fluctuations increase, initially thereafter decreasing slightly. Measured standard deviations in this quantity are 0.066, 0.051, 0.077, and 0.044 for the in-line bank, and 0.047, 0.062, 0.052, 0.027 for the staggered bank corresponding to Re of 100, 300, 1 000, 3 000. For the in-line bank the peak c_L value occurs at $Re = 1\ 000$, compared to the staggered bank where it occurs at $Re = 300$.

Overall c_D values decrease with increasing with Re . For in-line banks the mean values are 1.31, 0.75, 0.63, and 1.05 at Re of 100, 300, 1 000, 3 000. The corresponding values for the rotated square banks are 1.88, 1.73, 1.61, 1.32. The fluctuating component of c_D is large compared to the mean value, increasing steadily for the in-line bank, but tapering off at higher Re , for the rotated square bank (the standard deviations are 0.05, 0.12, 0.21, 0.55, in-line, and 0.05, 0.17, 0.31, 0.32, 0.33, staggered).

Figures 104 and 105 show $c_p(\phi)$ for the same sequence of events that as is shown in Figs. 84 to 86 inclusive (in-line) and Figs. 88 to 90 (staggered) for $Re = 300$. Comparison of these figures with those obtained from the steady flow analysis [Figs. 55 and 56], which are for $s/d = 1.5$, reveals substantial quantitative differences: For both geometries considered, large pressure rises are associated with the establishment of the attachment point at one or other sides of the front of the cylinder. A strong maximum occurs on only one side of the cylinder; the maxima rise and fall on opposite side of the cylinder(s). Further downstream, the pressure minima are also substantially affected by the transient nature of the flow. It can be seen that significant fluctuations in c_p are observed around the entire periphery of the cylinder. Some slight discontinuities are apparent in the pressure distributions near to the (four) special points, at the leading and trailing edges.

Figure 106 shows $c_f(\phi)$ for the in-line bank, $Re = 300$. A strong transient behaviour is again apparent. Comparison of this figure with say, Fig. 58 (steady flow, $s/d = 1.5$) reveals at times a large negative value of c_f between 0° and 45° , indicative of the relative strength of the shear-inducing vortices in transient flow. Values of $c_f = 0$ give the in-

stantaneous location of the attachment and separation points, and only one of each is apparent at any given time. The reader will recall that c_f as defined in Eq. 9.13 is taken as positive for $0^\circ \leq \phi < 180^\circ$ and negative for $180^\circ \leq \phi < 360^\circ$. This slightly unusual definition has been chosen in order to avoid discontinuities at $\phi = 0^\circ$ and 180° , which would otherwise arise. Close inspection of Fig. 106 reveals some distortion of c_f near the special points. Such distortions were substantially greater in the grid used for the rotated square bank due to grid expansion and non-orthogonality near the special points. For this reason profiles of derived quantities, such as c_f and $\overline{Nu}_H$, are not shown for this case.

9.3.4 Heat transfer

Figure 95 shows time-average $\overline{St}_H$ as a function of Re . The corresponding $\overline{Nu}_H$ values are tabulated in Tables 13 and 14. It can be seen the slopes of the unsteady $\overline{St}_H$ curves are less steep than those observed for steady flow [Chapter 8], and overall values are somewhat higher, probably as a result of the increased mixing of the fluid. Figures 102 and 103 show average $\overline{Nu}_H$ as a function of time. Both the time-average and the fluctuating component of $\overline{Nu}_H$ increase with Re . The standard deviations in $\overline{Nu}_H$ were 0.07, 0.47, 2.15, 3.87 for in-line and 0.32, 1.07, 1.86, and 2.86 for rotated square geometry at $Re = 100, 300, 1\,000$, and $3\,000$, i.e. the fluctuating component of $\overline{Nu}_H$ increases steadily with Re , but is a much smaller fraction of the mean value than was the was observed for either Eu , or c_D .

Both $\overline{St}_H$ and $\overline{Nu}_H$ were based on an assumed constant upstream fluid temperature. This is, of course, not the same as the instantaneous bulk temperature (not shown). Investigation showed that t_f also fluctuated, and increasingly so, with increasing Re . However these fluctuations were only a few degrees C, i.e. very much smaller than the difference between the wall and bulk temperatures [Figs. 86 and 90], so the use of this simpler

quantity is considered justified. (See Chapter 8 for a discussion of the impact of the choice of t_{ref} upon the value of h .)

Figure 107 shows local Nu_H distribution around cylinder B [Fig. 80], in-line bank, at $Re = 300$. It can be seen that there is substantial variation in $Nu_H/\overline{Nu}_H$ with time. Heat transfer is a maximum at the front of the cylinder near to the attachment point. The heat transfer transient appears to lag slightly behind the shear stress. Staggered banks (not shown) also display transient behaviour, as a result of the wake switching backwards and forwards, over the front of the cylinder. As a result of this, the point of maximum heat transfer moves around a zone at the front of the cylinder (for steady flow, peak heat transfer occurs at $\phi = 0^\circ$ in widely-spaced staggered tube banks)

It is apparent that both fluid flow, and heat transfer in tube banks are substantially affected by transient effects for $Re \geq 100$. In the case of pressure-drop and drag, large fluctuations are apparent. Skin-friction and heat transfer are also substantially different due to the nature of the flow in the inter-tube space being fundamentally altered.

9.3.5 Convergence tests

Table 15. Grid dependence tests, $Re = 300$.

Type	No. of Sweeps	No. of Steps	Sh
In-line square	40	480	0.0891
In-line square	40	960	0.0920
In-line square	100	960	0.0925
In-line square	100	1 960	0.0933
Rotated square	40	320	0.289
Rotated square	40	640	0.353
Rotated square	100	640	0.358
Rotated square	100	1 280	0.356

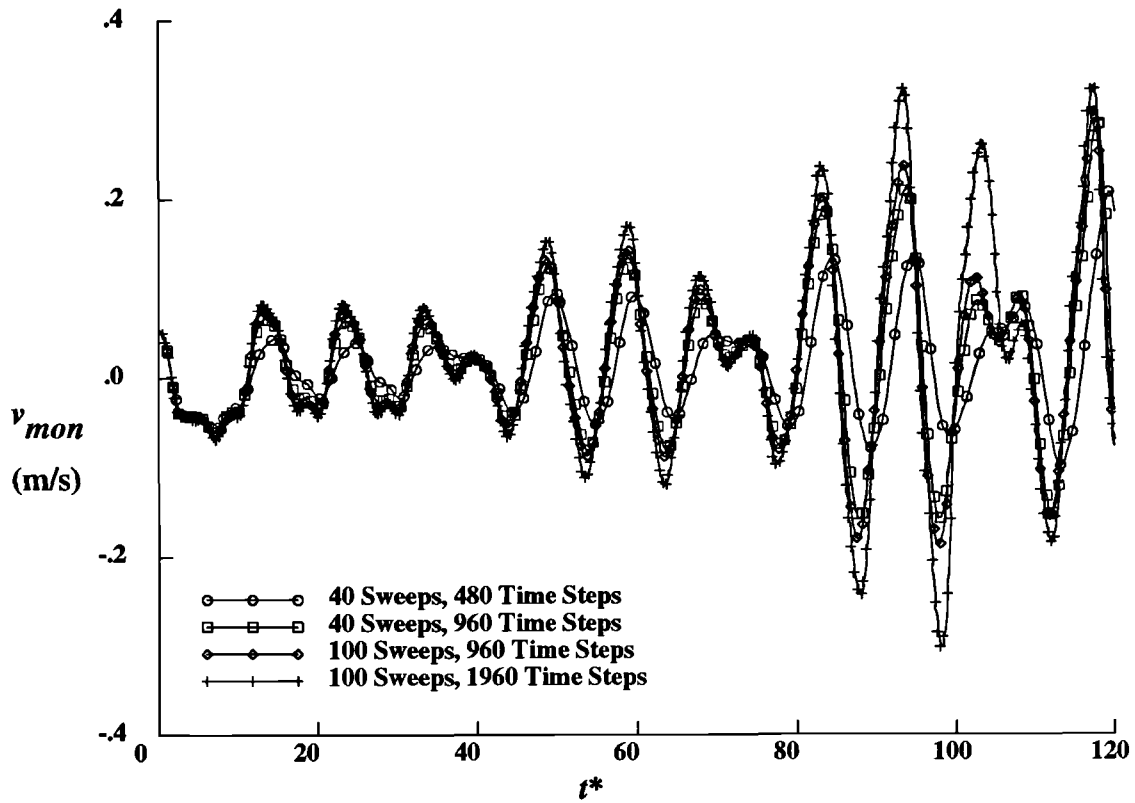


Figure 108. Convergence test, in-line square bank.

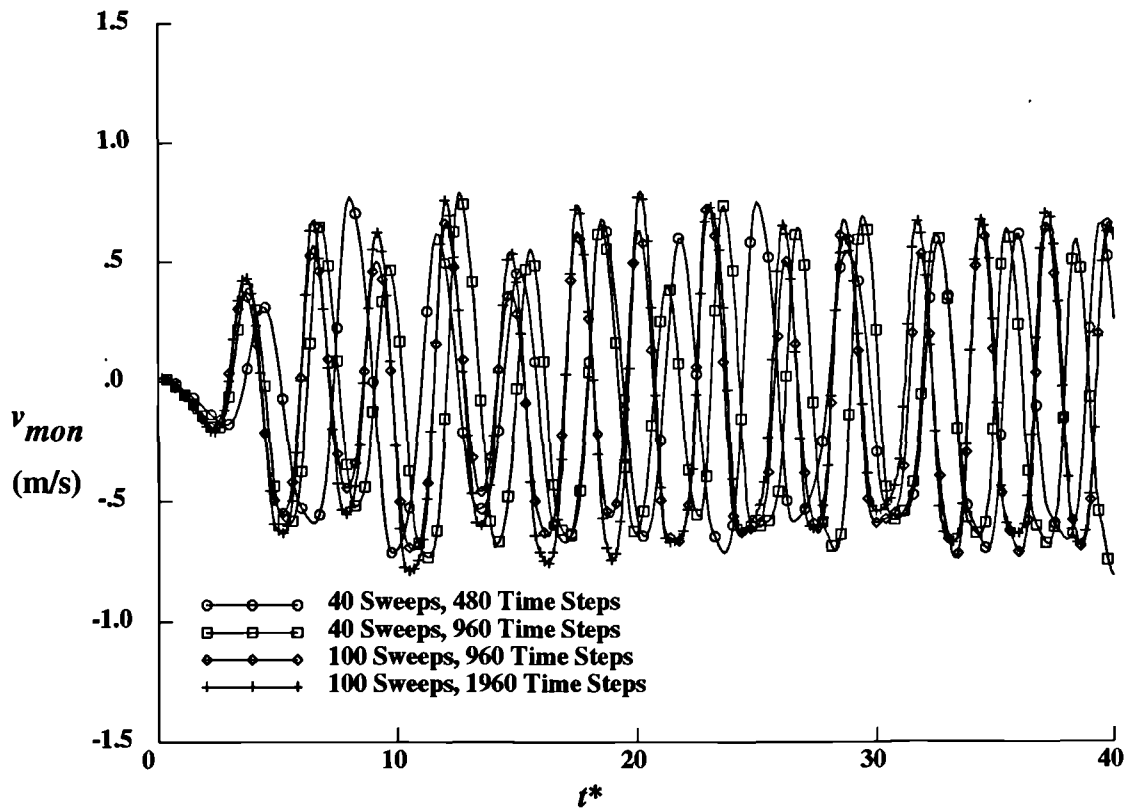


Figure 109. Convergence test, rotated square bank.

Tests were conducted to investigate whether the solution was fully converged. Generally speaking the approach was to increase the number of time steps, rather than the number of sweeps, until a measure of grid independence was achieved. The use of coarse grids was found to damp out the oscillatory behaviour, while the use of grids finer than the 80x160 meshes that were adopted in this study could not be entertained, due to the finite-limits of disk and central processor unit, on the workstation available for this project.

It can be seen that for the rotated square bank at $Re = 300$, Sh values obtained using 40 sweeps and 640 time steps were just 0.7% lower than those obtained using 100 sweeps per time step and 1 280 time steps (8 cycles at the assumed Sh). Similarly for the in-line bank, the difference between Sh obtained using 40 sweeps and 960 time steps and those obtained using 100 sweeps, and 1 920 steps is just -1.3% (12 cycles at the assumed Sh). Inspection of Fig. 108 shows that as the number of time steps and sweeps increases, the peaks in the monitor-location velocity distribution become larger, suggesting that, in spite of the very good Sh agreement, grid dependence has not entirely been achieved for in-line banks. Interestingly enough, the pressure profiles showed much better agreement. For the rotated square bank, the peak fluctuations show much less sensitivity to the number of sweeps and time steps, indicating that convergence here is quite satisfactory. The 1 920 step run took about one week on an IBM RS/6000 model 530 computer.

The reader will recall that block H-grids were used in this study. For in-line grids this is not strictly necessary; an ordinary H-grid could have been used, provided the cylinders are located at the corners and sides of the domain (though this does not provide a good view of vortex formation). However for staggered banks, due to the multiply-connected geometry, there is little other choice. Note that unlike steady flow, the grids here must be reasonably orthogonal on all the boundaries, including symmetry lines, making it difficult to concentrate cells near the special point. It was discovered through experience that, provided the special point is located near the rear of the cylinder, the grids are gen-

erated good results (otherwise spurious extrema in the pressure distributions and distortions of the vector field were observed). It is not possible to construct a grid that is aligned with the secondary vortical flow, and an assessment of the effects of numerical diffusion is required. The paper by Abd-Rabbo and Weaver (1986), suggested that for staggered geometry, the vortices survive past the minimum cross-section to the front of the subsequent row. This was not observed in this study, and may possibly be due to the influence of numerical diffusion. Another area for concern is the use of the implicit scheme in view of the small time steps. It is quite possible other methods not considered as part of this study, e.g. the Crank-Nicholson scheme, could render superior results. However in tube banks, unlike external transient flow past simple bluff bodies, the choice of scheme may be subordinate to the physical realism of the chosen boundary conditions, the choice of which is by no means obvious.

9.3.6 Boundary and initial conditions

Several types of boundary conditions were considered prior to those that were finally adopted: Initially, periodic boundaries [Chapter 8], corresponding to $\tau_x = \tau_y = 0$, were imposed at the inlet and exit. This almost always resulted in the transient motion being damped out, although experiments on rectangular cylinders (in Cartesian co-ordinates) did generate periodic behaviour, provided the aspect ratio was extremely small (i.e. like baffles). Occasionally, in staggered banks, the flow would become unstable and ‘flop-over’ to one or other side where it would remain for the remainder of the simulation. This is as a result of the lack of an explicit time-averaged zero-momentum boundary condition in the cross-wise direction. This behaviour was also noted by Stuhmillar et al. (1988) who obtained large-eddy simulation results for turbulent flow in tube banks, but paradoxically were unsuccessful in attempts to simulate laminar-periodic flow: In spite of the more complex physics associated with turbulent flow, the presence of addi-

tional source terms in the momentum equations would appear to render an unsteady solution more amenable, and hence less boundary-condition dependent.

Any attempt to impose a constant downstream pressure was also found to completely damp out all periodic motion. Some attempts were made to use a two-way upstream-downstream substitution process (similar to that described in Chapter 8), but measuring the phase-difference between disturbance and response, and adjusting the length of the ring buffers dynamically. This procedure was abandoned as being overly-complex, and difficult to control.

For the computer code used in this study, periodicity at the lateral boundary conditions was found to be insufficient to maintain the motion on its own. Only by providing an explicit feedback mechanism across the upstream half of the lateral boundary, could the motion be maintained. Tests showed the downstream flow to be extremely responsive to the initial excitation when this was applied in a span-wise fashion across the lateral boundary *A-B* [Fig. 80]. Conversely, fluctuations applied perpendicular to the main flow *A-F* [Fig. 80] were quickly damped out. Only for very widely-spaced banks, $s/d > 3$, was it possible to obtain transient behaviour spontaneously (in the absence of an excitation mechanism), in all other cases a span-wise stimulus was required. The same situation prevailed when re-applying downstream values upstream: The stream-wise terms, while still important, are secondary to the lateral values. This appears to be a situation where the small-scale motion and the convective-interactions are inseparable.

The reader who (like this author) feels some initial resistance to the idea of partially prescribing lateral boundaries may draw consolation from the fact that, since sources of mass, momentum, etc. do at times emanate from these boundaries, they are in fact ‘upstream’ for a significant fraction of the cycle, and cannot therefore be ignored for the problem at hand. Thus for transient periodic problems with a significant lateral component, the term ‘upstream’ should refer, not merely to a particular direction, but rather to

a (solid) angle encompassing all possible directions from whence matter can arrive. At the same time, if the flow is prescribed along the entire lateral boundary, then the iterative trial-and-error nature of the methodology is lost. In general it would fair to say that the larger the value of s/d , the easier it is to generate instabilities. The presence of an increasingly favourable pressure gradient as $s/d \rightarrow 0$ provides a partial explanation. Compact banks represent a significant challenge, and should be the subject of future research.

There may be some justification for the assumption that the lateral transients are approximately in phase (i.e. $\tau_y = 0$). Both Wallis (1939) and Zukauskas et. al. (1988b) have suggested that for many tube banks this is a reasonable approximation. It must, however, be admitted that there is something inherently paradoxical in this premise: It would appear that these fluctuation could only initially arise (and be propagated) in the presence of minor differences in velocity and/or pressure between adjacent cross-wise modules. Such differences could easily arise, for example, if the flow were not truly two-dimensional. The existence of even the slightest mass flow in the longitudinal direction could therefore affect the nature of the transient. So far as the author is aware, there are, as yet, no flow visualisation or other studies, demonstrating whether periodic flow in tube banks are fundamentally of a two-dimensional or a three-dimensional nature. The above assumptions were made in order to obtain a working solution with the resources available at the time of writing. The assumptions of two-dimensionality and lateral periodicity are worthy of future study; without super-computer access, neither a 2-D manifold, nor the transient behaviour of a single 3-D module of finite length [Chapter 10] could be entertained.

The exit boundary condition, namely, that the downstream pressure may be computed from potential flow theory has its origins in 'boundary-layer theory'. Comparison of the steady-state solution obtained using Eq. 9.4 with the results of Chapter 8, revealed that this method of prescribing the downstream flow field, though only approximate, is rea-

sonable. The reader will note that the prescription [Eq.9.4] is only strictly true if the stream-wise velocity is large in comparison to the cross-wise component. Some distortion of the pressure field at the exit was observed, owing to the fact that the velocity profile near to the cylinder walls is concave and not convex, as is the case for potential flow. This, however was localised and did not significantly affect the results. The velocity field in this region was quite reasonable. A possible alternative procedure would be to prescribe a fixed stream-wise mass-flux (Neuman boundary condition) at the exit. This, however, does not address the need to fix the pressure on a time-wise basis somewhere in the domain-at-large. Some additional distortion in the flow field was also apparent at the upstream boundary, this time due to the fact that curvature and divergence effects were not accounted for, when computing boundary coefficients. It is believed that, while the boundary conditions may have affected some quantitative aspects of the transient process, they are for the most part based on reasonable assumptions.

Conventional logic dictates that the fully-converged steady-state solution should form the basis for the initial field. This was not found to be the ideal choice here: For widely-spaced banks, and single bluff bodies, the steady-state solution results in a pair of vortices being formed, as the solution proceeds, the vortices elongate substantially, eventually forming a dead zone downstream of the cylinder. Experience showed that once such a (physically unrealistic) dead zone was formed it was extremely difficult to generate the transient periodic behaviour, (in tube banks, the elongation was less apparent, though the end result was often the same). For tube banks, this was not the case, however, the vortices did become abnormally enlarged and elongated at higher Re [see Figs. 40 and 41, Chapter 8]. For this reason, a partially converged steady-state solution, where the wake vortices had barely formed was used as the basis form the initial, $T = 0$, solution.

9.4 Closure

The transient response of in-line square and rotated square tube banks to lateral upstream disturbances was investigated. It was shown that for $Re > 100$, such disturbances are self-propagating. The results of flow-field simulations were presented, illustrating the nature of the transient phenomena. Details of the vortex generation mechanism were discussed. Tables and graphs of Sh , Eu , $\overline{Nu}_H$, c_D , and c_L data were provided. Substantial differences were found to occur in these values from those obtained based on a steady flow analysis.

10. FIN AND TUBE HEAT EXCHANGER

10.1 Introduction

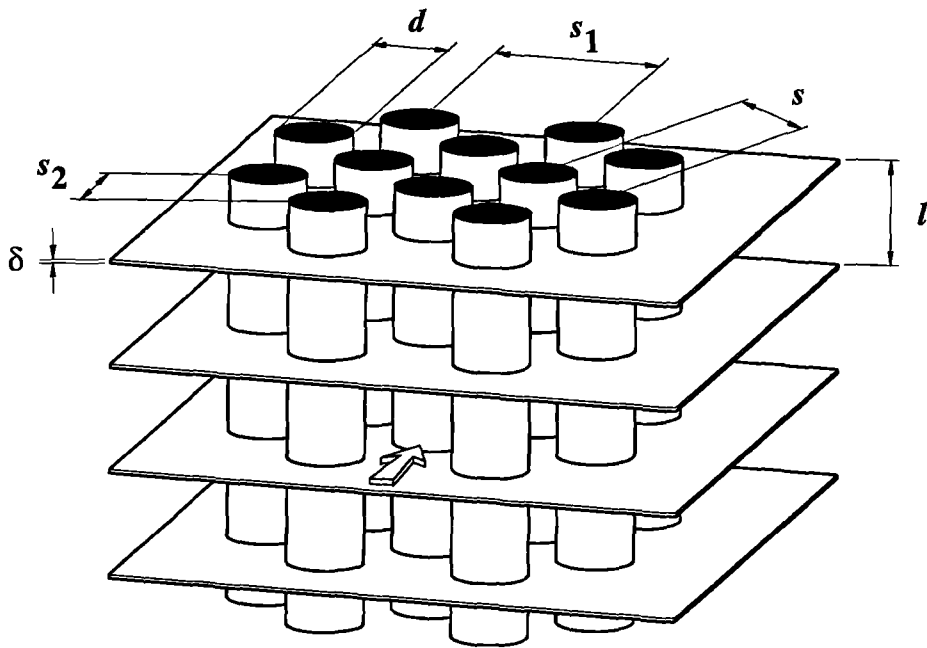


Figure 110. Plate-fin fin and tube heat exchanger.

Up to now, the flow in tube banks has been considered to be two-dimensional. In practice the finite-length, l , of the tubes [Fig. 1] may result in the flow being different from the idealised 2-D flow field. The purpose of this chapter is to investigate this factor. Staggered tube banks using fins with small values of l/d are deliberately employed in the refrigeration and air-conditioning industries, owing to their compact size. Figure 110 is a schematic of one such fin and tube heat exchanger unit.

The scope of the analysis is again confined to fully-developed periodic cross-flow. Let x denote the stream-wise, y the cross-wise, and z the longitudinal direction. Both in-line

square ($s/d = 1.5$) and rotated square ($s/d = 2$) tube banks were considered with length-to-diameter ratios, l/d , of $1/2$, 2 and 10. The fin was presumed to be of negligible thickness, $\delta = 0$. Two limiting thermal boundary conditions were considered:

- (i) Constant tube temperature, t_w , and adiabatic fin, $q_{fin} = 0$.
- (ii) Constant tube temperature and constant fin temperature, $t_{fin} = t_w$.

Fully-developed flow and heat transfer were assumed, so that the boundary conditions are essentially the same as in Chapter 8, and will therefore not be further discussed here.

Flow and heat transfer in in-line and rotated square banks of pitch-to-diameter ratio, $s/d = 1.5$ and 2, respectively, was studied. The main parameter investigated was the effect of varying the longitudinal aspect ratio l/d . Values of $1/2$, 2, and 10 were considered. For $l/d = 2$, Re was varied in the range $30 \leq Re \leq 300$.

10.2 Results

The results are shown below in Figs. 111 to 126 and Tables 16 to 21. Figures 111 and 112 show velocity vectors and particle traces for the in-line arrangement, at $Re = 100$. Figures 113 and 114 are similar plots for the staggered geometry. The location of the particle traces was chosen to highlight regions of interest, outside the main flow.

Figure 115 shows the inlet velocity profile for a rotated square bank, iso-contours being used indicate constant values of stream-wise velocity. It can be seen that for $l/d = 2$, the magnitude of the velocity field is a strong function of the z distance. Conversely for

$l/d = 10$, the velocity profile is essentially two-dimensional, except in the region very close to the fin.

The results of the flow-field simulations were rearranged so that the data could be displayed on a plane corresponding to the surface of the cylinder. Figures 116 and 117 are particle traces confined to the wall of the cylinder, sometimes known as ‘oil-flow’ traces, obtained using flow visualisation software (Walatka et. al., 1990). The ordinate is the angle ϕ from the front-leading edge of the cylinder, and the abscissa the non-dimensional distance, z^* ,

$$z^* \equiv \frac{z}{l} \quad (10.1)$$

ranging from $z^* = 0$ at the fin, to $z^* = 1/2$ at the plane-of-symmetry, midway between fins. Traces are shown for $30 \leq Re \leq 300$. The oil-flow traces display separation and attachment zones on the wall of the cylinder. Figure 118a shows more oil-flow work for the rotated square cylinder at $Re = 300$, on the symmetry planes at $z^* = 0$ and upstream of the cylinder at $\phi = 0^\circ$, Fig. 118b shows particle traces seen from a different angle.

Figures 119 and 120 show (dimensional) temperature contours on the planes of symmetry. Figure 119 is a comparison of the temperature distribution, under conditions of (a) adiabatic fin, $q_{fin} = 0$, and (b) constant temperature fin, $t_{fin} = t_w$, for the rotated square array with $l/d = 1/2$, while Fig. 120 shows similar contours for a high aspect-ratio, rotated square bank with $l/d = 10$. In both cases $Re = 100$.

Figures 121 and 122 are contours of local pressure coefficient $c_p(\phi, z^*)$ on the wall of the surface of the cylinder, the pressure coefficient being defined here according to,

$$c_p \equiv 1 - \frac{p(\phi=0, z^*=1/2) - p(\phi, z^*)}{\frac{1}{2}\rho u_m^2} \quad (10.2)$$

Figures 123 and 124 are similar contours of friction coefficient $c_f(\phi, z^*)$, defined according to,

$$c_f \equiv \frac{|\tau_w|}{\frac{1}{2}\rho u_b^2} = \mu \frac{\partial u_t}{\partial y} \bigg|_w \frac{1}{\frac{1}{2}\rho u_b^2} \quad (10.3)$$

where the u_t is the tangential speed,

$$u_t \equiv \sqrt{u^2 + w^2} \quad (10.4)$$

Figures 125 and 126 show values of $Nu_T/\overline{Nu}_T$ as a function of ϕ and z^*

Tables 16 to 21 are a summary of the overall pressure drop and heat transfer data obtained from this project in terms of Eu and $\overline{Nu}_{LM}$, based on an overall energy balance [Eq. 8.25], for in-line and rotated square banks, as a function of l/d . Also reproduced here, for comparison are the 2-D data from Chapter 8.

Table 16. Eu for in-line square arrangement, $s/d = 1.5$.

Re	$l/d = 1/2$	2	10	2-D
30		2.60		2.06
100	2.11	0.893	0.746	0.716
300		0.354		0.287

Table 17. Eu for rotated square arrangement, $s/d = 2$.

Re	$l/d = 1/2$	2	10	2-D
30		1.77		1.15
100	2.27	0.852	0.680	0.636
300		0.642		0.647

Table 18. $\overline{Nu}_{LM}$ for in-line square arrangement, $s/d = 1.5$, adiabatic fin.

Re	$l/d = 1/2$	2	10	2-D
30	4.88	4.36	4.69	4.71
100		4.55		4.85
300		5.22		5.47

Table 19. $\overline{Nu}_{LM}$ for in-line square arrangement, $s/d = 1.5$, constant t_w fin.

Re	$l/d = 1/2$	2	10
30	11.8	4.60	4.75
100		4.75	
300		5.39	

Table 20. $\overline{Nu}_{LM}$ for rotated square arrangement, $s/d = 2$, adiabatic fin.

Re	$l/d = 1/2$	2	10	2-D
30	6.54	5.09	6.12	5.11
100		6.28		6.20
300		12.1		11.9

Table 21. $\overline{Nu}_{LM}$ for rotated square arrangement, $s/d = 2$, constant t_w fin.

Re	$l/d = 1/2$	2	10
30	27.9	6.37	6.13
100		7.62	
300		14.5	

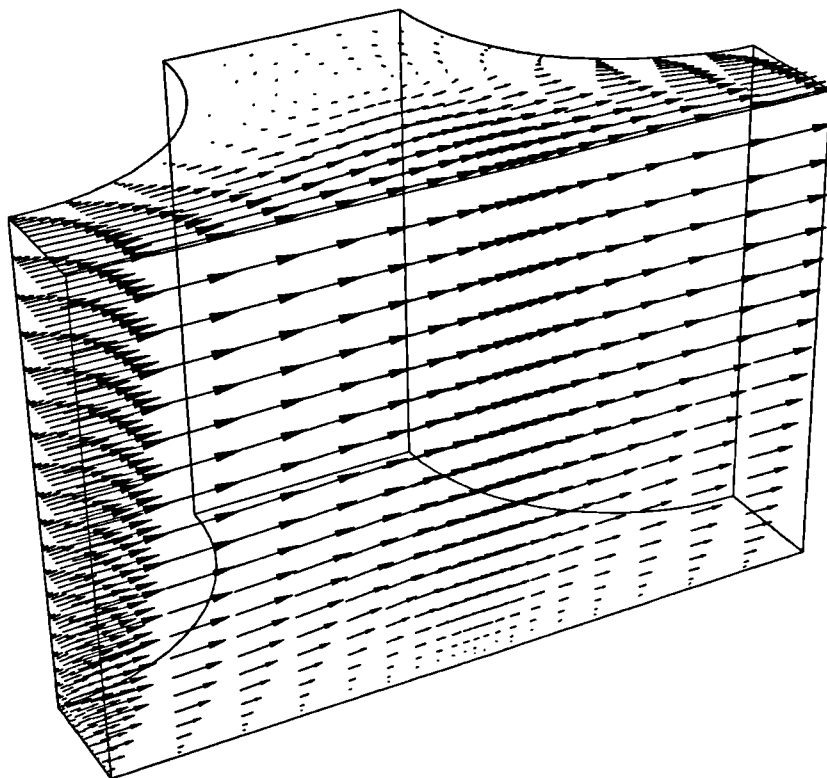


Figure 111. Velocity vectors, in-line square, $U/d = 2$, $Re = 100$.

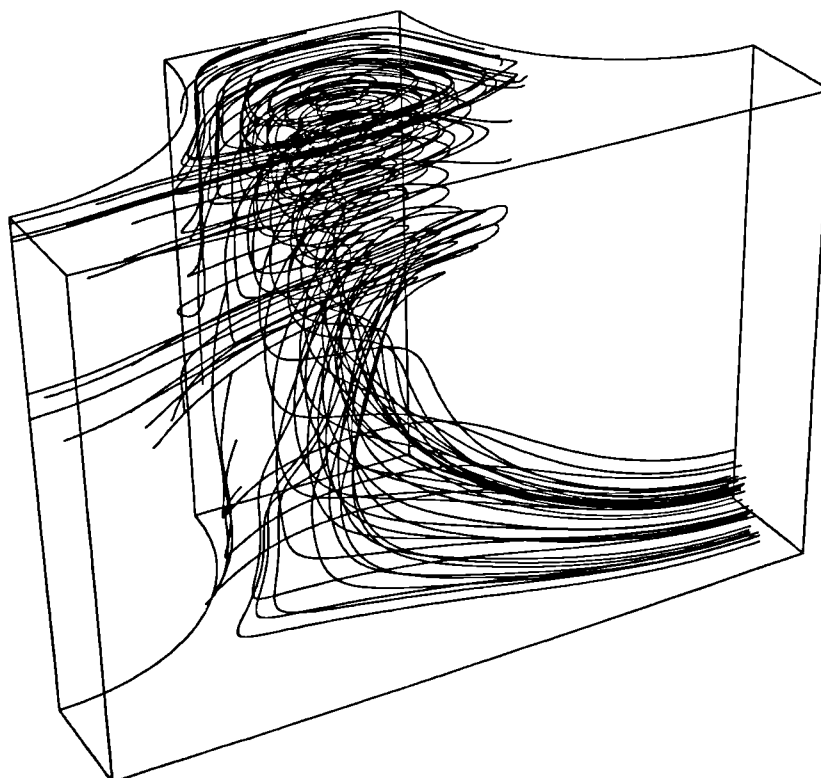


Figure 112. Particle traces, in-line square, $U/d = 2$, $Re = 100$.

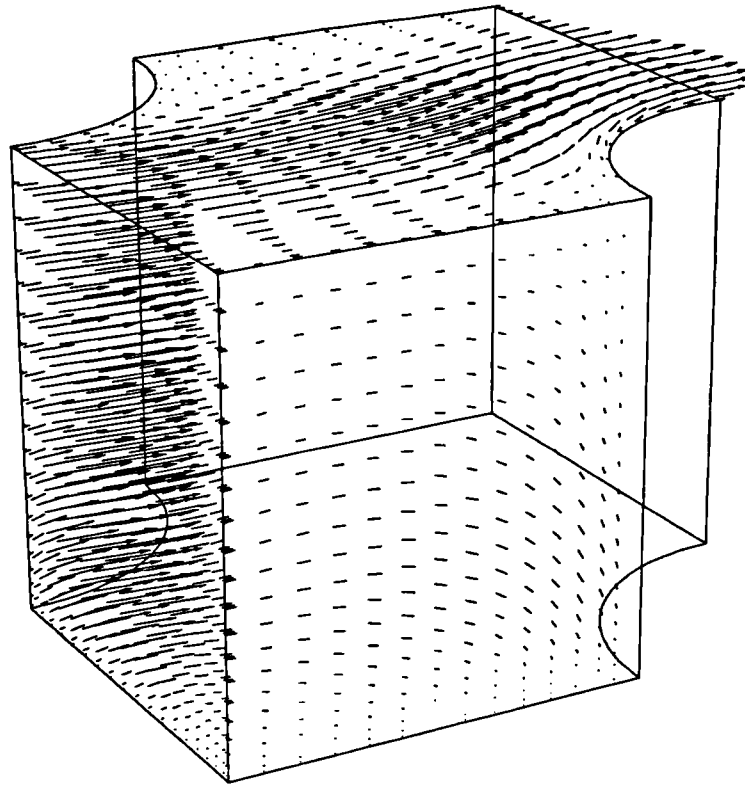


Figure 113. Velocity vectors, rotated square, $l/d = 2$, $Re = 100$.

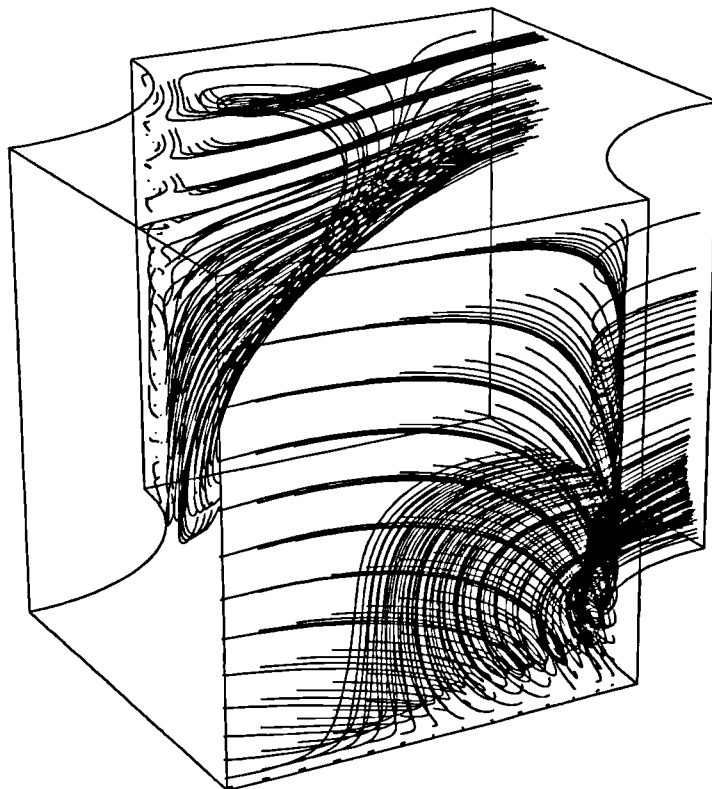
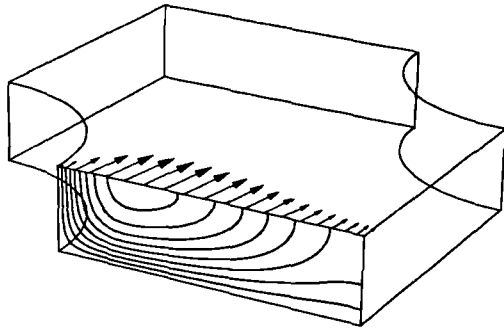
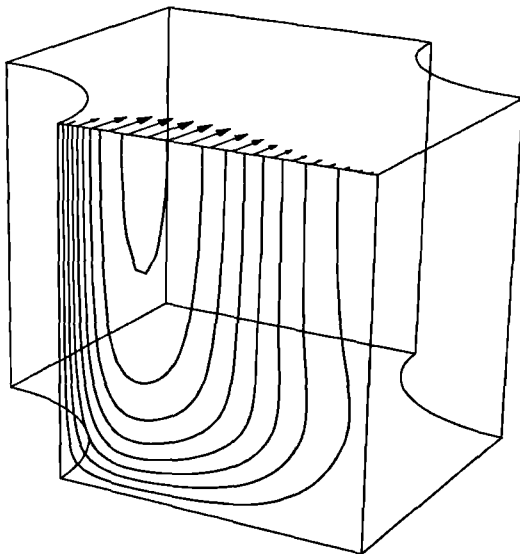


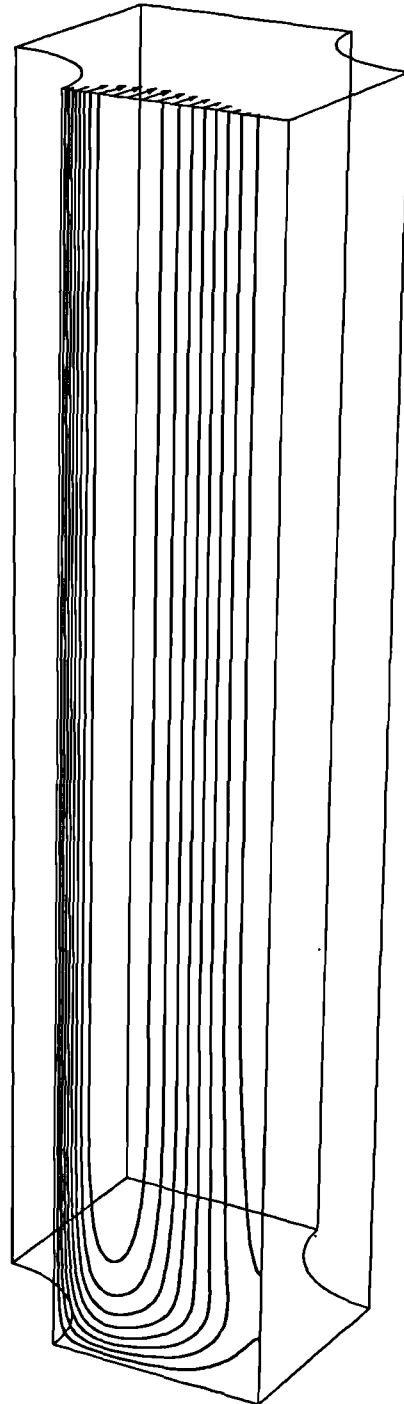
Figure 114. Particle traces, rotated square, $l/d = 2$, $Re = 100$.



(a) $l/d = 1/2$



(a) $l/d = 2$



(a) $l/d = 10$

Figure 115. Inlet velocity contours, rotated square, various l/d .

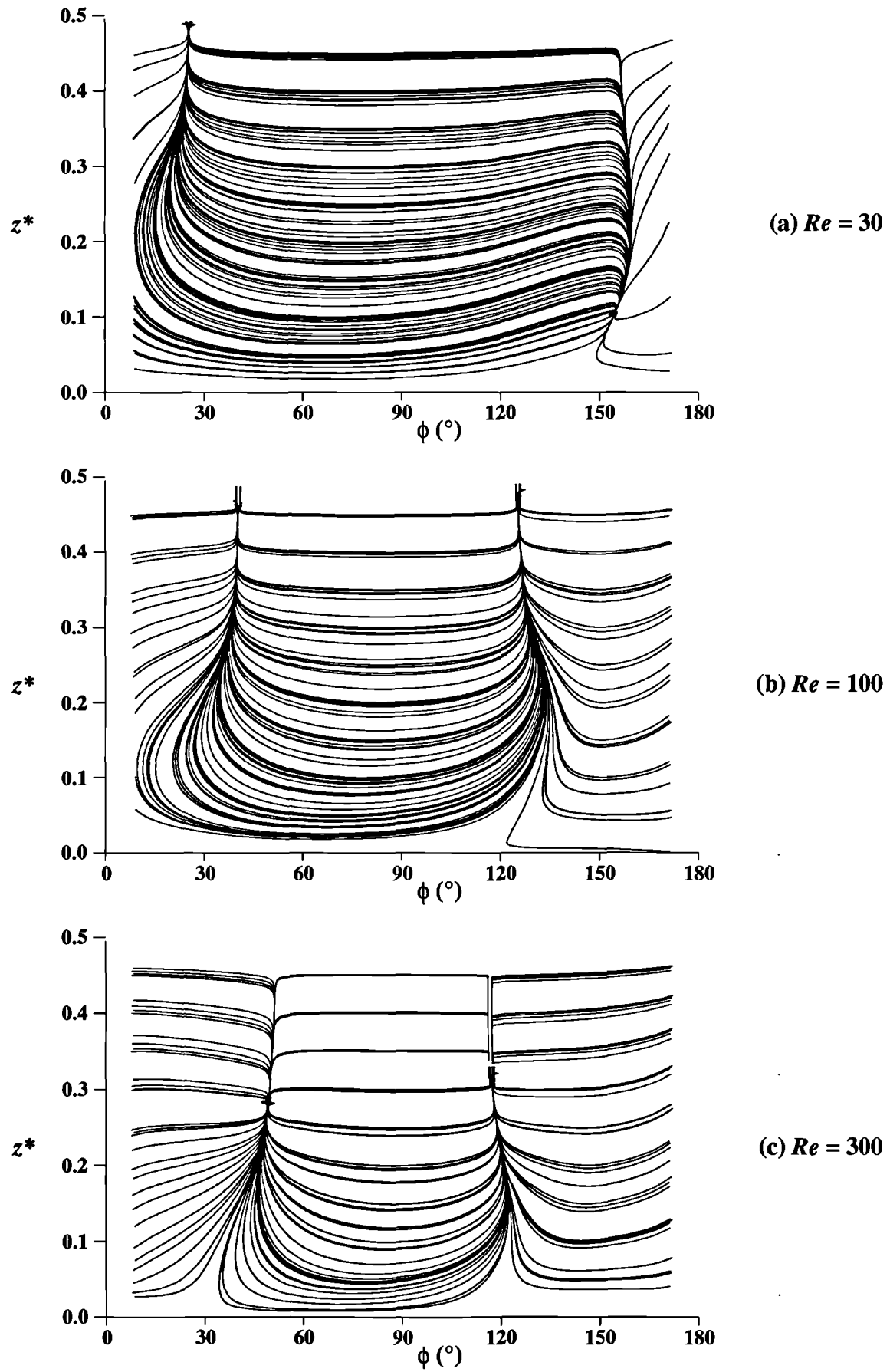


Figure 116. Oil-flow traces, in-line square, $l/d = 2$.

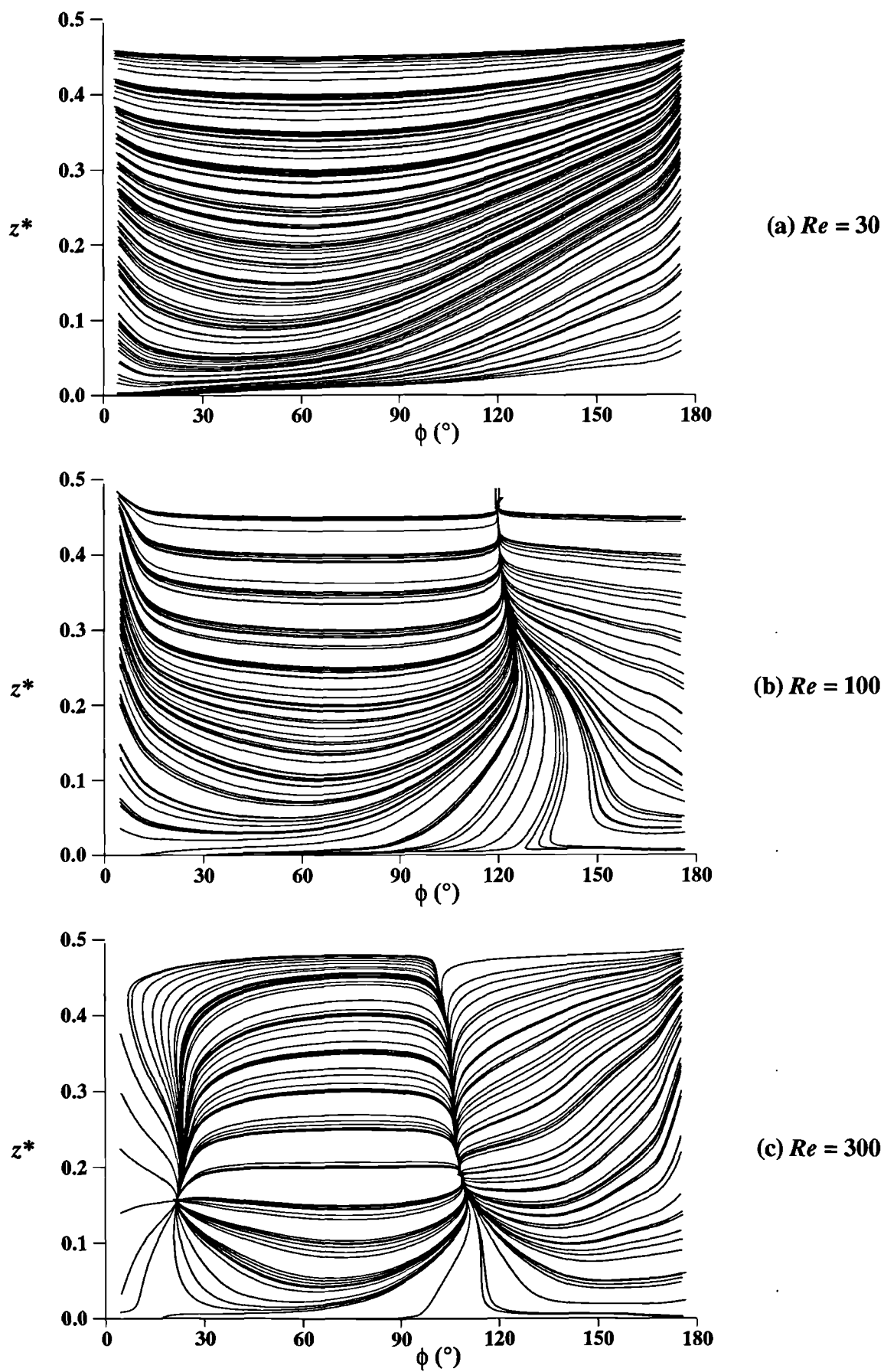


Figure 117. Oil-flow traces, rotated square, $l/d = 2$.

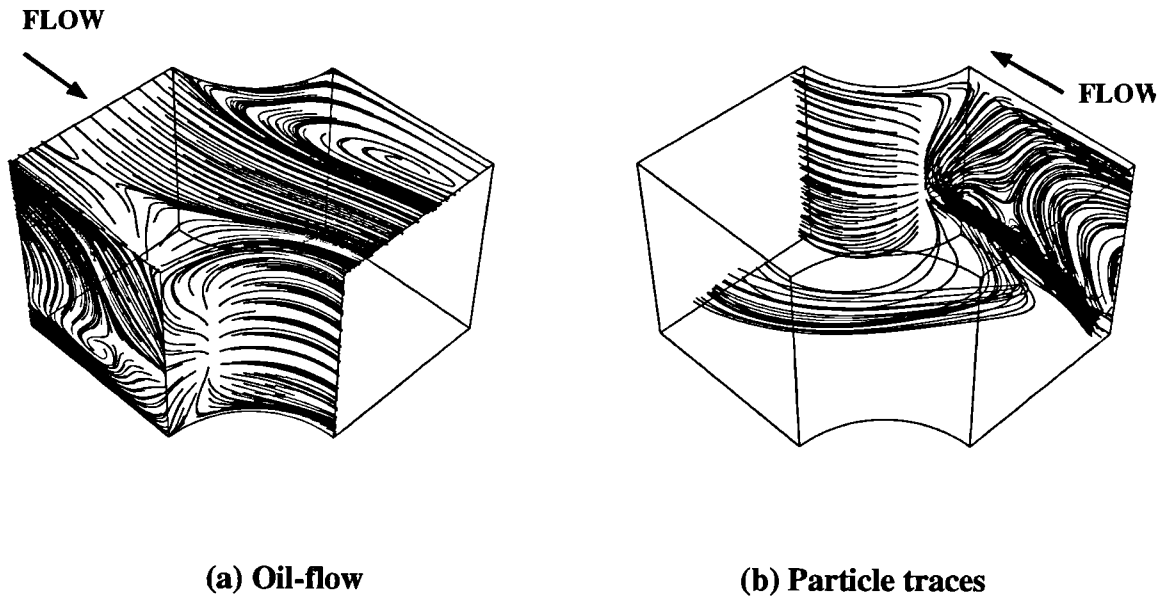


Figure 118. Oil-flow and particle traces, rotated square, $l/d = 2$, $Re = 300$.

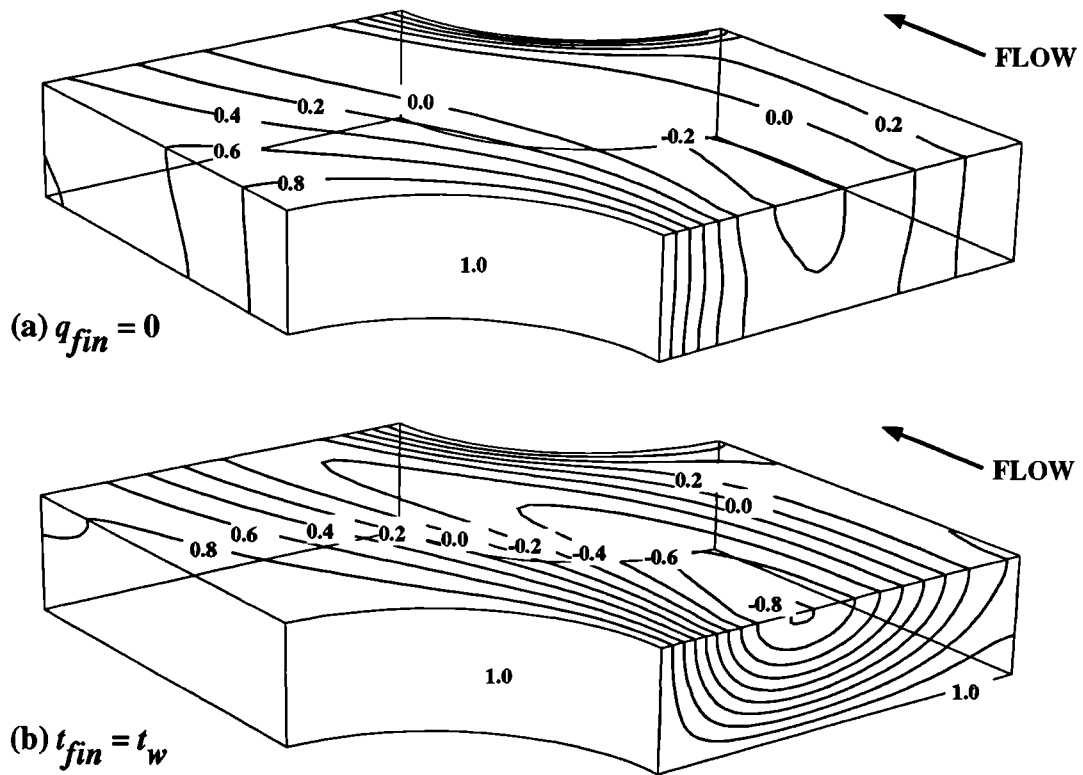


Figure 119. Temperature contours ($^{\circ}\text{C}$), rotated square, $l/d = 1/2$, $Re = 100$.

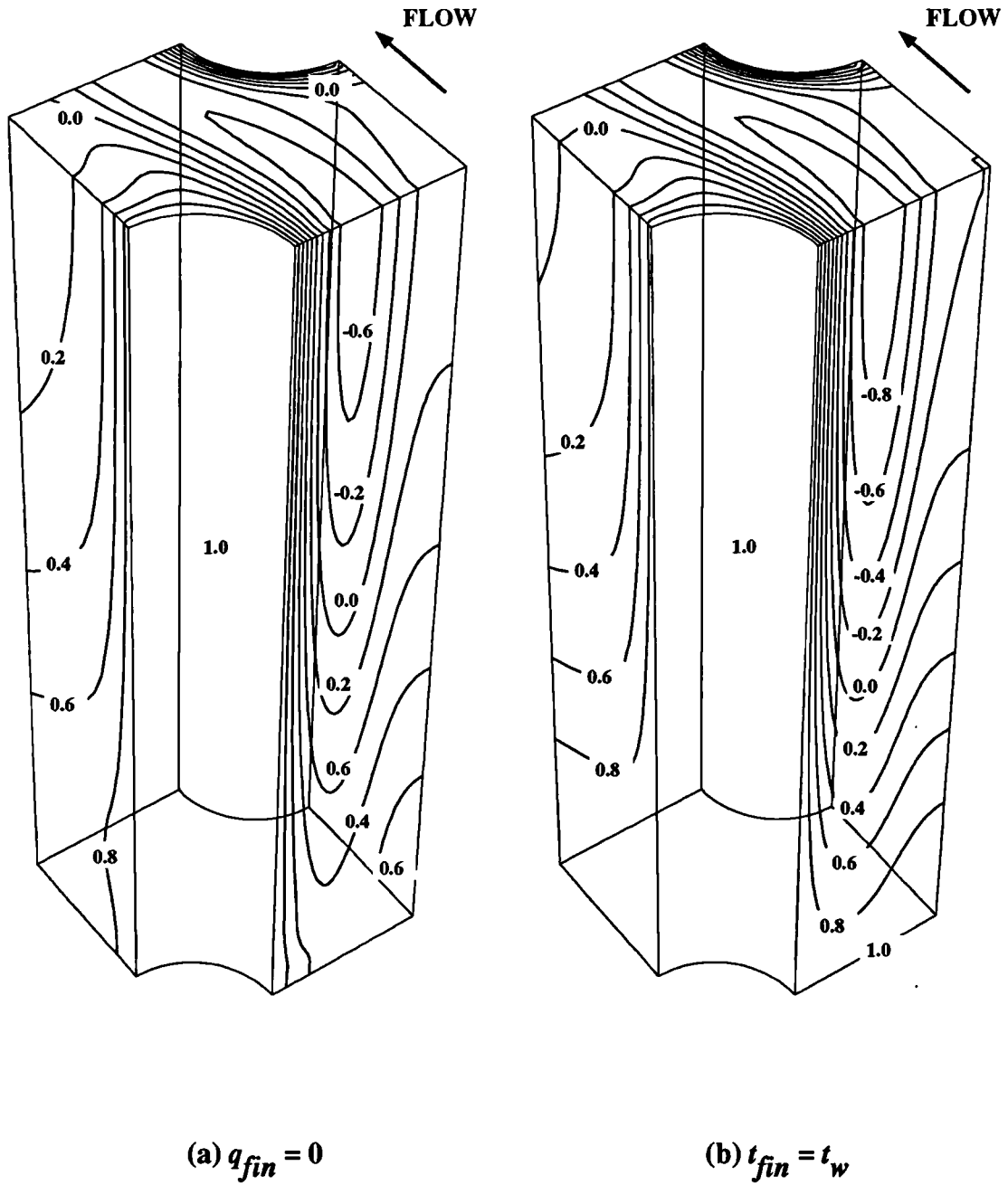


Figure 120. Temperature contours ($^{\circ}$ C), rotated square, $l/d = 10$, $Re = 100$.

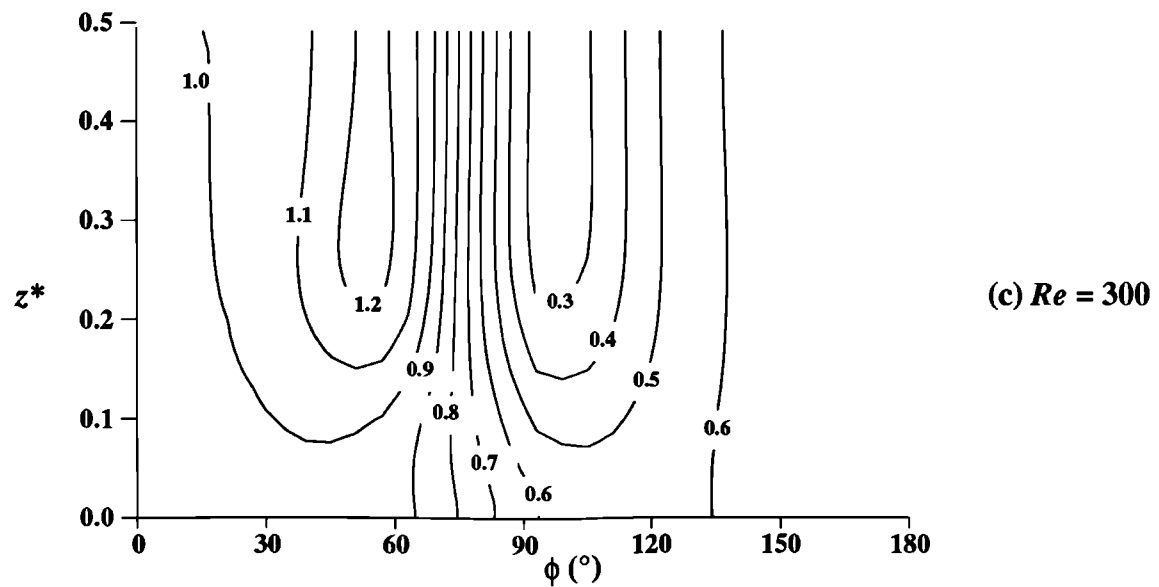
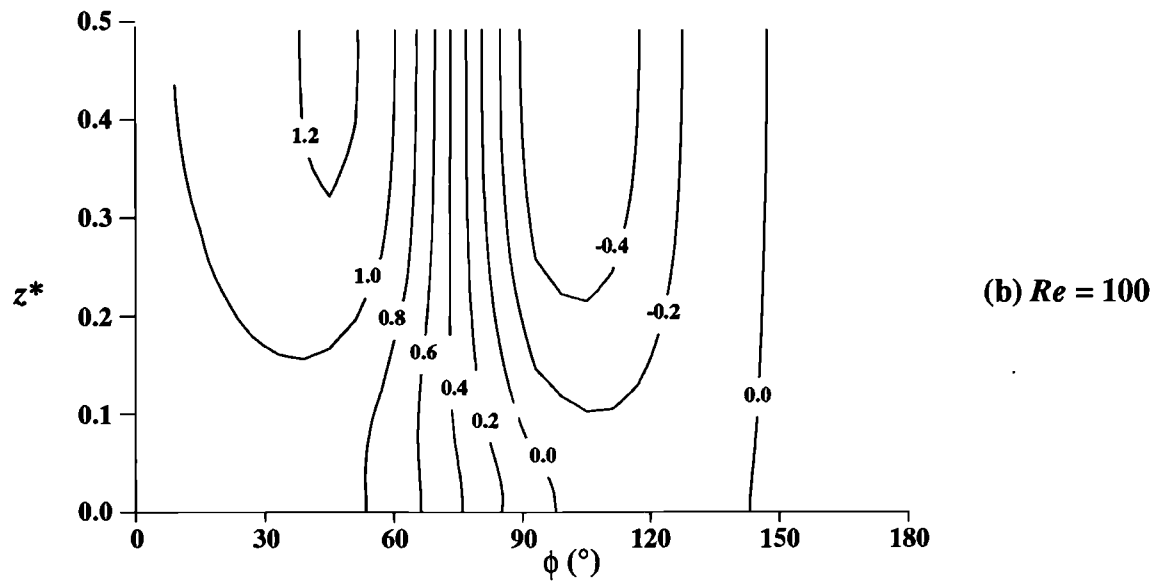
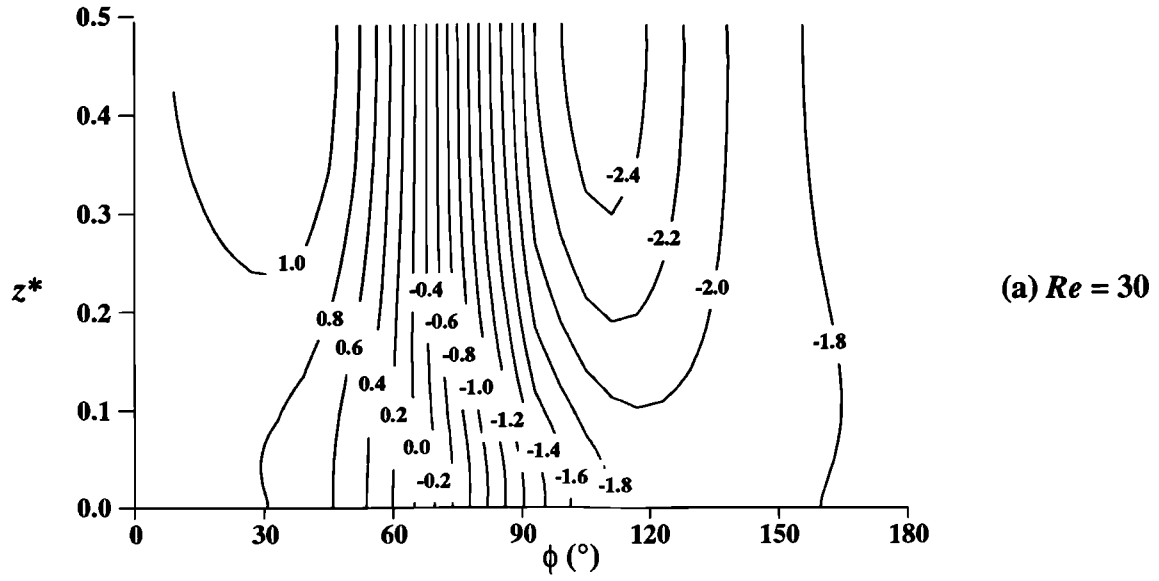


Figure 121. c_p contours, in-line square, $l/d = 2$.

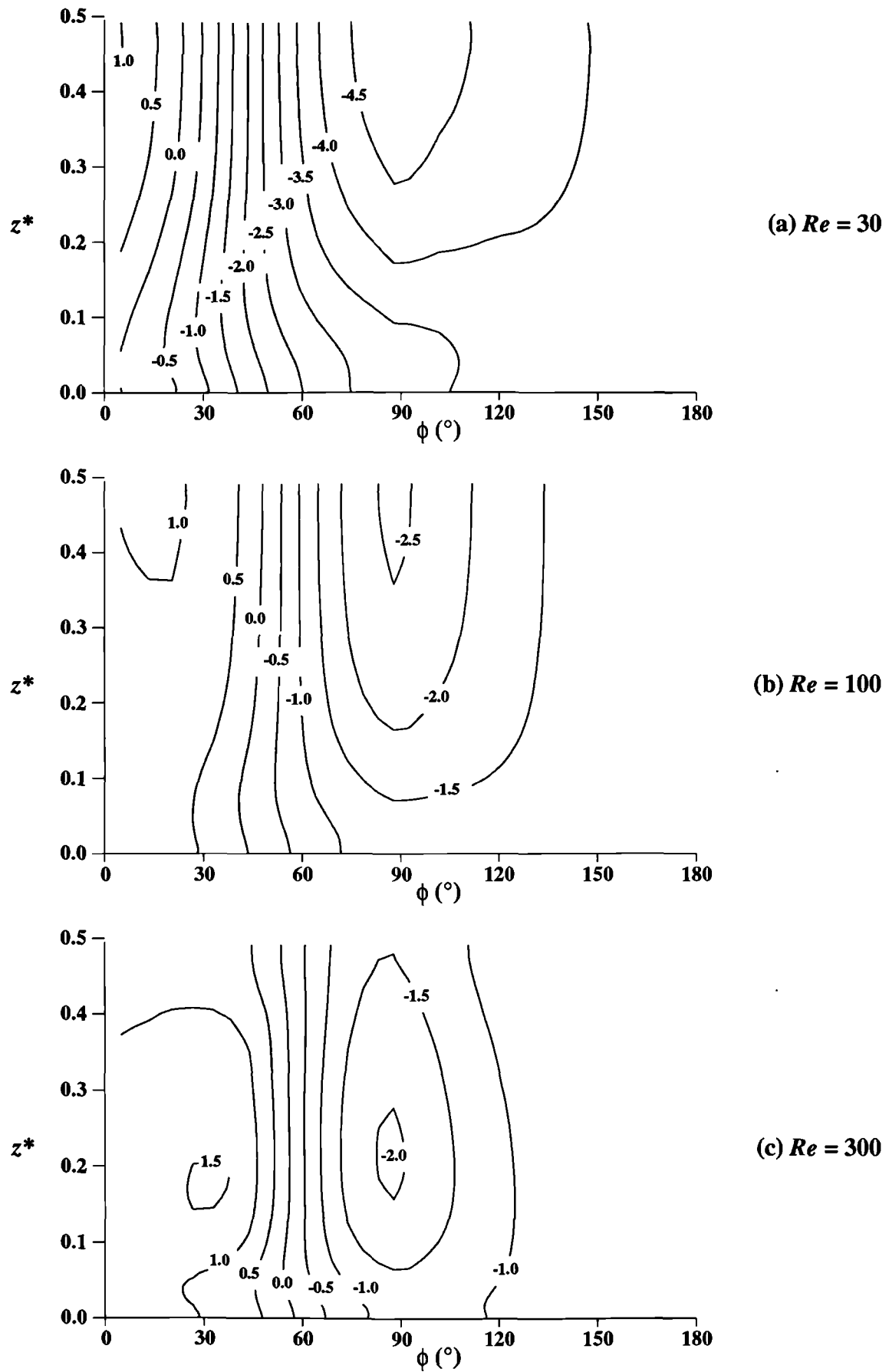


Figure 122. c_p contours, rotated square, $l/d = 2$.

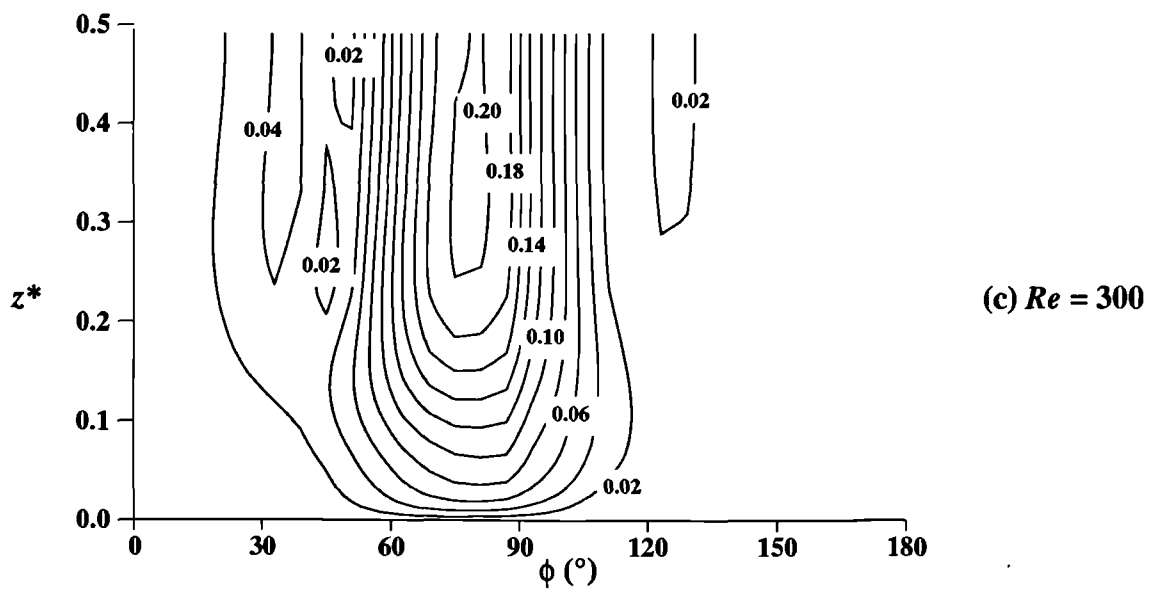
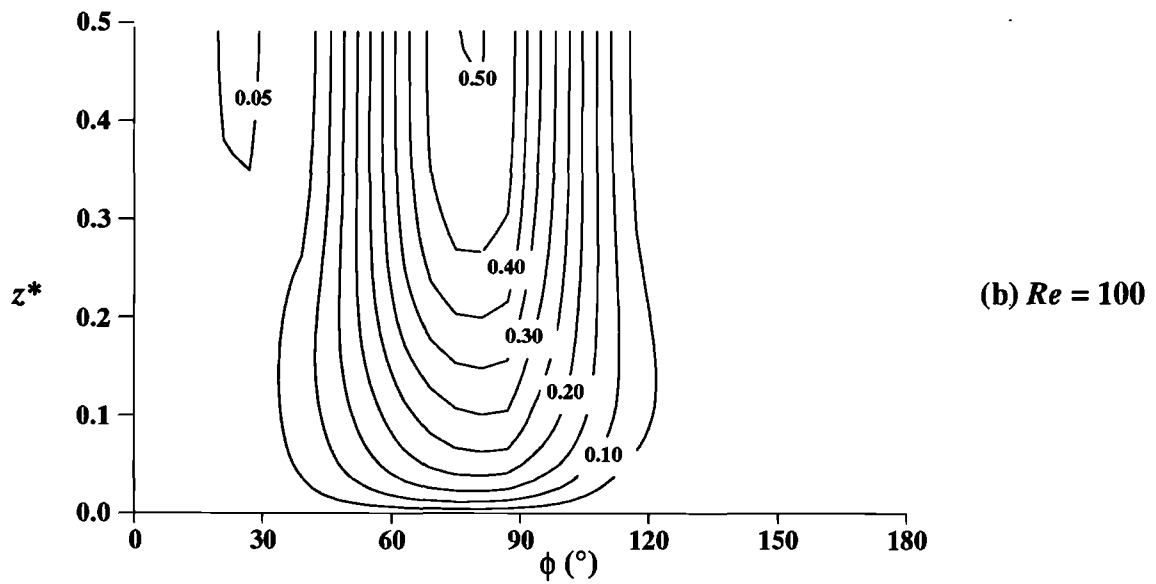
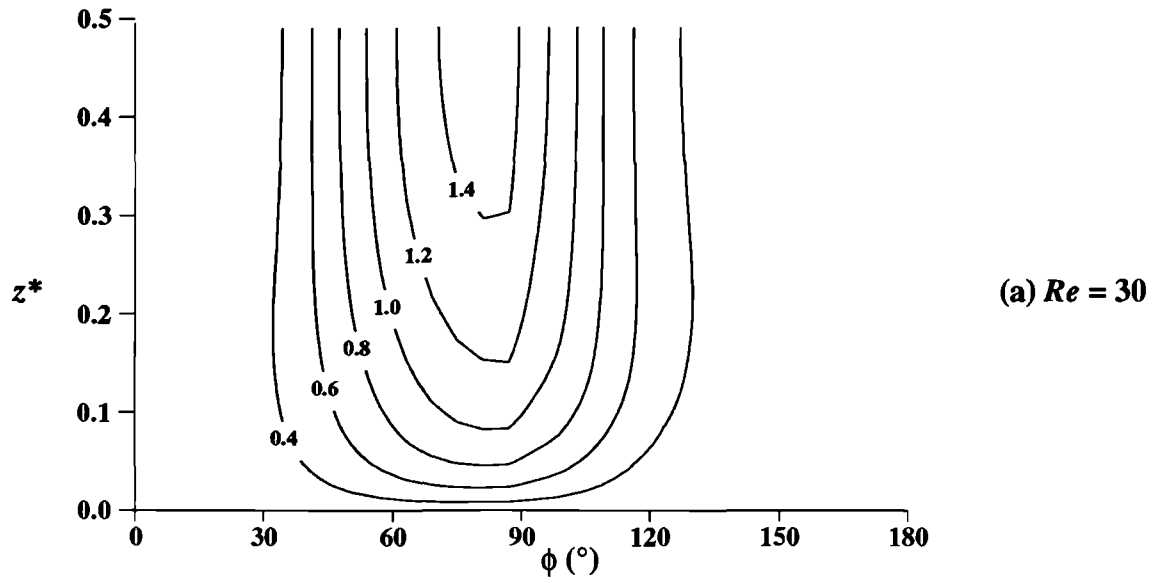
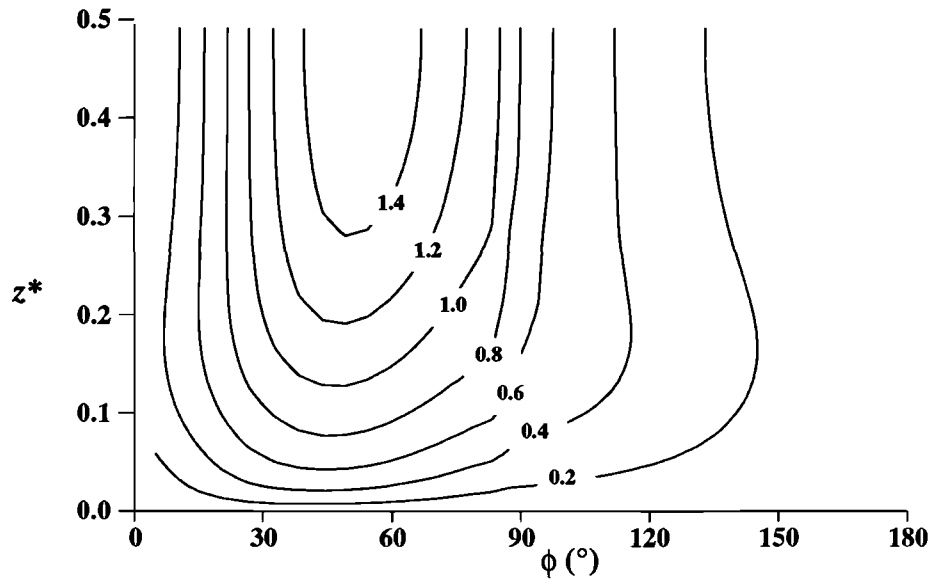
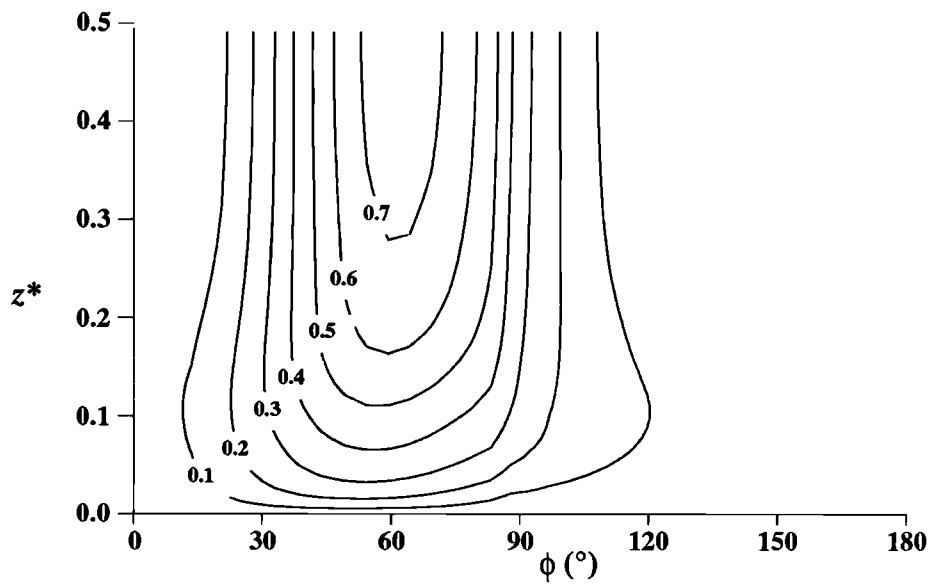


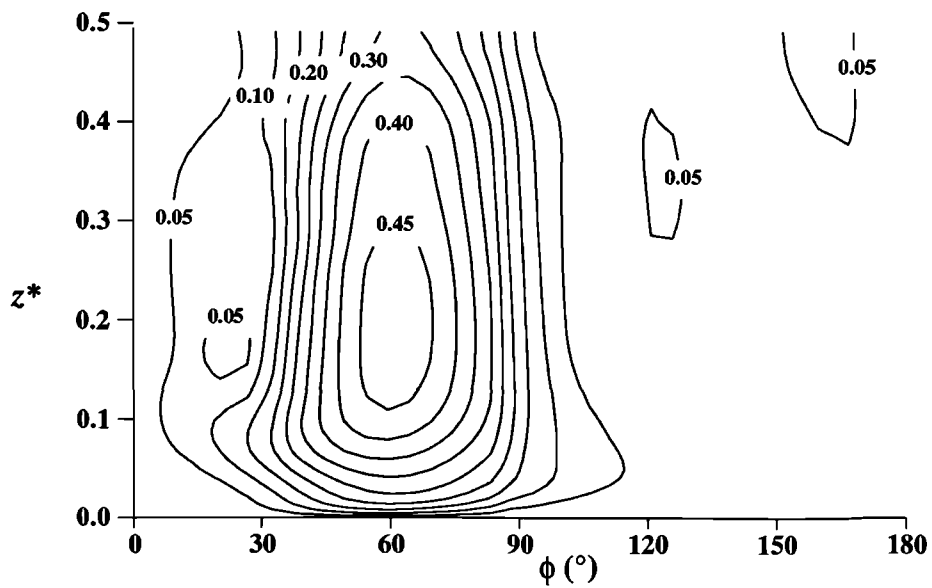
Figure 123. c_f contours, in-line square, $l/d = 2$.



(a) $Re = 30$



(b) $Re = 100$



(c) $Re = 300$

Figure 124. c_f contours, rotated square, $l/d = 2$.

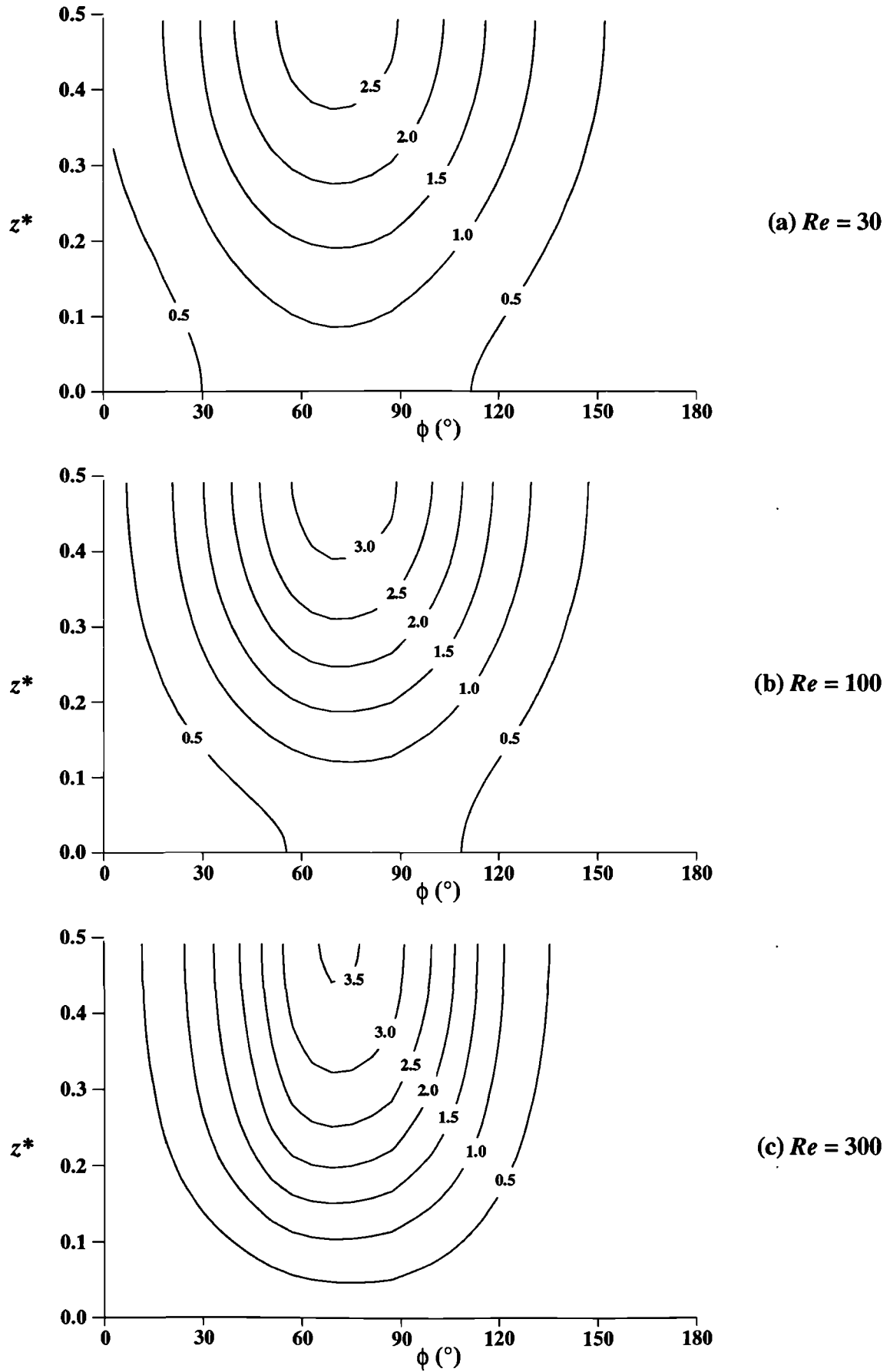


Figure 125. $Nu_T/\overline{Nu}_T$ contours, in-line square, $l/d = 2$, adiabatic fin.

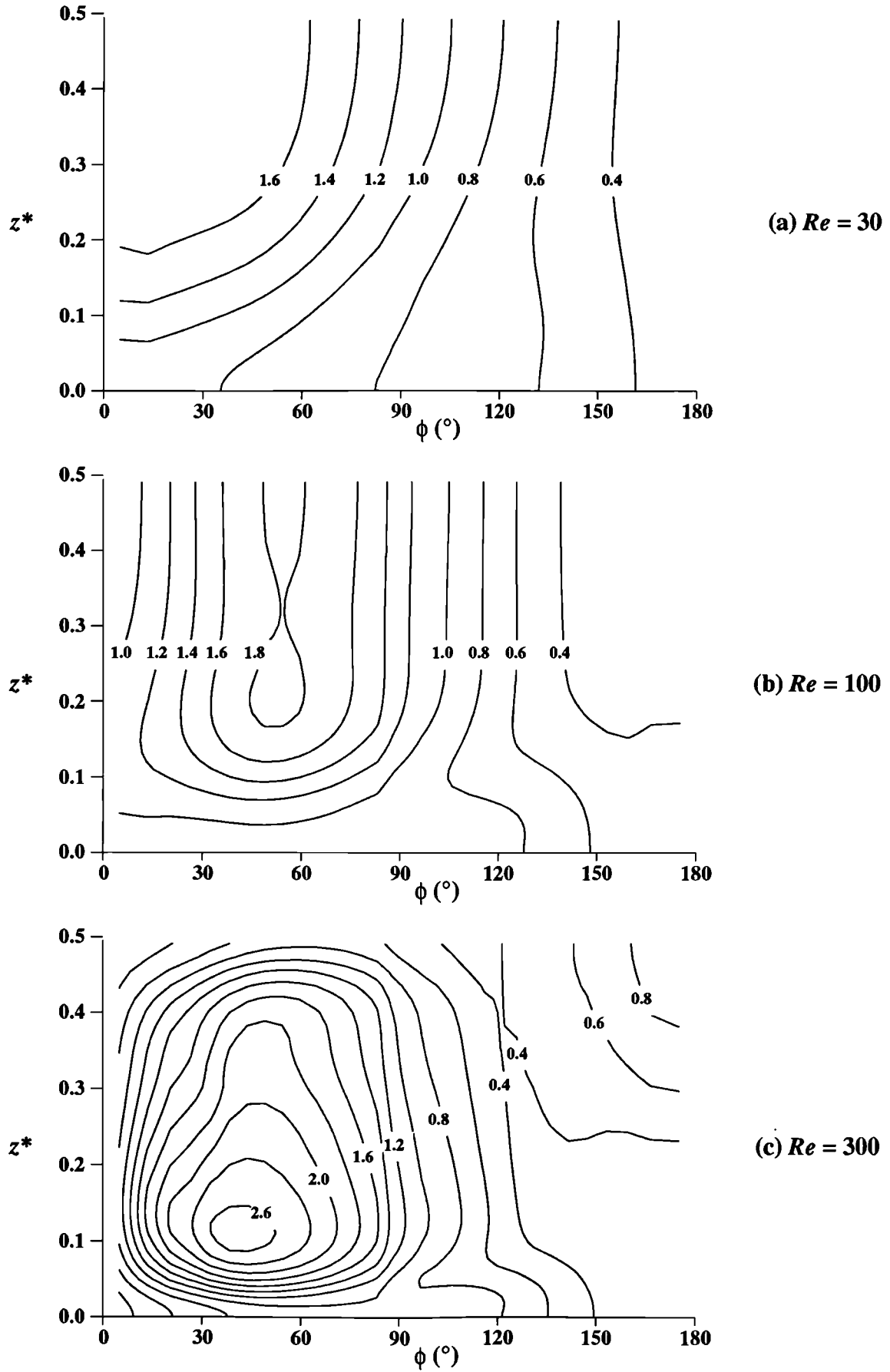


Figure 126. $Nu_T / \overline{Nu}_T$ contours, rotated square, $l/d = 2$, adiabatic fin.

10.3 Discussion

10.3.1 Qualitative description of the flow

Analysis of the figures reveals a complex pattern of flow and heat transfer. Figure 111 is a plot of velocity vectors for the in-line bank, $l/d = 2$, $Re = 100$; vectors are shown at the inlet, and in the planes of symmetry. Figure 113 is a similar vector plot for the rotated square bank, vectors being shown in the symmetry (x - y) plane, upstream of the cylinder in the vertical x - z plane, as well as across the inlet. (N.B. Vectors are not shown for every cell node.) In both cases, the velocity distribution in the horizontal x - y plane is qualitatively similar to that observed in the 2-D case, with a vortex (pair) being exhibited in the wake region. For the in-line bank, the velocity distribution in the minimum cross-section is parabolic-shaped, both in the transverse, and the longitudinal directions.

For the staggered tube bank, it can be seen that a channel-like velocity profile is apparent in the cross-wise direction, with the flow maximum occurring somewhere between the cylinder wall and the symmetry plane, as a result of the influence of the previous upstream vortex (not shown). The velocity distribution in the longitudinal z -direction is not parabolic-shaped, rather it is relatively constant except near to the fin, i.e. a boundary-layer-type velocity profile.

Inspection of Figure 115 shows the inlet velocity profile to be a function of l/d ; for small l/d the inlet profile is a strong function of height (i.e. parabolic-shaped), becoming two-dimensional as the aspect ratio l/d , is increased. It would appear that, in spite of the presumed laminar fully-developed (periodic) boundary conditions, that provided l/d is sufficiently large, the presence of the transverse shear layer impedes the longitudinal velocity field from developing, resulting in a boundary-layer profile. In-line tubes exhibit

the same tendency, except that higher Re are required for the shape of the velocity to change from parabolic-shaped to slug-shaped. This is probably due to the fact that the shear layers in staggered tubes occur on alternate sides at half the distance of those in in-line banks.

Close inspection of the velocity vectors and particle traces for the staggered tube bank [Figs. 113 and 114] near the front of the cylinder, in the vertical x - z plane, reveals the flow to be in a downward direction, i.e. a secondary flow pattern in the vertical plane is apparent. At the same time a general upward trend is apparent in the main wake-vortex, which is no longer two-dimensional, but grows up and out from the root [Fig. 114]. For external flow over extended surfaces, it is well-known that one or more 'horse-shoe' vortices occur near the root, depending on the flow regime, see for example Eckert and Langston (1987). Glynn and Rawnsley (1988) conducted numerical simulations for a single extended surface, using code based on the same numerical scheme as this author. In this study, similar vortices were sometimes observed for staggered tube banks. For the most part, however, all that was observed was a downward tendency of the flow in this region [Fig. 114]. At higher Re , the influence of the upstream wake-vortex appears to decrease the centreplane velocity (causing a channel-like profile), to some extent suppressing the formation of the secondary vortex. However the situation is complex; due to the proximity of neighbouring cylinders, the trailing (horizontal) and secondary (vertical-horseshoe) vortices interact and compete with each other, both in the stream-wise and cross-wise directions. This can result in elaborate three-dimensional secondary flow patterns being observed [Fig. 118].

For in-line banks, it would appear from the particle traces [Fig. 112] that fluid particles entering the vortex from the faster moving free-stream are drawn downwards by the action of the vortex. (The reader is cautioned that particle traces may be subject to numerical error.) Secondary phenomena were also observed, although the situation is compounded by the fact that attachment no longer occurs at $\phi = 0^\circ$, hence that the down-

ward vortical phenomena were again less clear, occurring as they did further round the front of the cylinder. Nonetheless a distinct 'kink' or 'twist' was seen in particle traces originating at the cylinder root near the attachment zone.

The oil-film traces [Fig. 116 and 117] reveal aspects of the nature of the flow near to the walls of the cylinder. From Fig. 116a (in-line bank, $s/d = 1.5$, $l/d = 2$, $Re = 30$), it can be seen that flow attachment, indicated by the diverging flow lines at around $\phi = 25^\circ$, occurs only at the top third of the cylinder, i.e. well away from the root (where the cylinder joins the fin at $z^* = 0$). Separation, on the other hand, appears to occur over most of the of the vertical length of the cylinder at around $\phi = 155^\circ$ (converging flow lines). It can be seen that prior to attachment at the front of the cylinder, the flow exhibits a definite downward tendency, and that the converse is true at the rear subsequent to separation. As Re increases [Fig. 116b,c], the flow pattern becomes somewhat more symmetric, with the attachment point moving slightly downstream (increasing ϕ), and the separation point upstream (decreasing ϕ), as the inter-tube vortex enlarges. Oil-traces are used here to show separation and attachment. In contrast to the 2-D case, c_f as defined by Eq. 10.3, is always positive and therefore cannot be used to indicate separation or attachment.

Figure 117 shows oil-flow patterns for rotated square tube banks, $s/d = 2$, $l/d = 2$, at various Re . The patterns are substantially different than for the in-line case: At $Re = 30$, there is a streaming flow, with the traces tending to go downwards towards the root ($z^* = 0$), at the front of the cylinder, for $\phi \leq 60^\circ$, and subsequently up, away from the fin towards the plane-of-symmetry ($z^* = 1/2$) as ϕ increases. There is no separation. At $Re = 100$, a vortex pair is established in the wake, as indicated by the separation line at around 120° , attachment is at the front of the cylinder, $\phi = 0^\circ$. The downwards tendency of the stream-lines at the leading edge is more marked. Further details of the nature of this flow at $Re = 100$, may also be had from the particle trace pattern [Fig. 114]. By $Re = 300$, the inter-tube vortex has enlarged with separation occurring downstream at

around 105° . Upstream, an entirely different flow pattern is apparent; a fluid jet apparently impinges on the cylinder at around $z^* = 0.15$, $\phi = 25^\circ$ causing the apparent divergence in Fig. 117c: As a result of the wake-vortex elongating towards the front of the subsequent cylinder, and down into the secondary flow zone, described above, there is a strong interaction between the wake and horseshoe vortices. Figure 118 reveals further details. Without entirely ruling out the possibility of this being realistic, one might speculate at this Re , transient-periodic behaviour will have commenced, so that what is being exhibited is perhaps a good example of an unrealistic flow pattern, resulting from inappropriate assumptions.

Figures 119 and 120 show (dimensional) temperature contours at the inlet and symmetry planes, for a rotated square bank $l/d = 1/2$, and 10 respectively. In each case contours are shown for both adiabatic fin and constant temperature fin. In the first case it can be seen that the effect of the wall boundaries is to change the temperature profiles substantially, in the latter case, the impact is minor.

10.3.2 Overall pressure loss and heat transfer

Tables 16 and 17 show values of Eu as a function Re . As might be expected, the effect of decreasing l/d is to increase the required overall pressure gradient. It can be seen that for the in-line bank, $l/d = 1/2$, Eu is 295% of the 2-D value of 0.716, for $Re = 100$; for $l/d = 2$, Eu is 125% of the 2-D value, while the value for $l/d = 10$ is only 105% of the 2-D value. For the staggered bank, the ratios are 357%, 134% and 107% respectively. Thus, the commonly-held assumption that flow in tube banks may be treated as 2-D for $l/d \geq 10$, does appear to be reasonable, at least when considering overall pressure drop at sufficiently high Re .

It is seen from Tables 18 and 20 that values of the overall heat transfer coefficient for the 3-D case, when the fin is assumed to be adiabatic, are also quite similar in magnitude to observed 2-D values: In some cases heat transfer is enhanced, in other cases it is slightly reduced. However the differences are small for both in-line, and rotated square banks.

As might be expected, the effect of presuming the fin to be at constant t_w , rather than adiabatic, is more pronounced for small l/d : Overall heat transfer was 242% (in-line) and 427% (rotated square) times the adiabatic fin values at $Re = 100$, for $l/d = 1/2$. It can be seen that for $l/d \geq 2$, there is very little benefit to be had at all. Comparing $l/d = 1/2$ to the 2-D case, at $Re = 100$, it is seen that for in-line geometry, constant t_w fin, while the heat transfer coefficient is 243%, the required pumping power is 295% of the 2-D case, so there is no real advantage, unless size is a consideration. For staggered geometry, the values are 450% and 357%, so there is some gain to employing the fin and tube geometry as opposed to a plain tube bank.

The two limiting cases (adiabatic, and constant t_w fin) typically span the case of a fin material of finite conductivity. (A method whereby such cases may be simulated is given in Chapter 11.) Pressure drop and heat transfer data are not compared to experimental data, such as that of Rich (1973, 1975), or correlations (McQuiston, 1978), as these are for a higher Re range than was considered here.

10.3.3 Local drag and heat transfer

Further insight into the nature of the flow is obtained from the pressure coefficient data, $c_p(\phi, z^*)$ [Figs. 121 and 122]. For the in-line case at $Re = 30$, it can be seen that the pressure coefficient is a global maximum of around 1.2 at $z^* = 1/2$, $\phi = 35^\circ$, and a global minimum of -2.6 at $z^* = 1/2$, $\phi = 110^\circ$. As Re is increased, the maximum moves to 45°

($Re = 100$) and 55° ($Re = 300$) decreasing slightly in magnitude, to around 1.25. The global minima are -0.6 at 105° and -0.3 at 100° , with some pressure recovery further downstream. Thus (i) the downstream minima decrease with increasing Re , (ii) the extrema move towards each other, as a result of vortex enlargement, and (iii) the distributions elongate in the z^* direction, i.e. the pressure distributions become more two-dimensional. Comparison of the data of Fig. 121 with the 2-D data of Fig. 55 reveals that the results are consistent.

For the staggered case [Fig. 122] $30 \leq Re \leq 100$, the maximum c_p of around 1.1 occurs near to the front of the cylinder around $\phi = 15^\circ$, $z^* = 0.5$ at $Re = 30$, moving down and out very slightly as Re increases. The pressure minima of -5 ($Re = 30$), and -2.6 ($Re = 100$), occur at 90° , significantly further upstream than was the case for in-line flow. These results are consistent with the 2-D results [Fig. 56]. At $Re = 300$, a relatively large maximum of 1.6 in c_p is observed at $z^* = 0.175$, $\phi = 30^\circ$, as a result of the flow regime discussed above. This distribution bears no resemblance to the 2-D case. Some slight disparities in c_p and c_f are apparent at $\phi = 90^\circ$ for the staggered bank at $Re = 30$. The reasons for this were discussed in Chapter 8.

The reader will recall that for potential flow (in widely-spaced geometry), the pressure coefficient was a maximum, at $\phi = 0^\circ$, and 180° , and a minimum at 90° , where the flow is a maximum, i.e. the pressure gradient is adverse for $\theta > 90^\circ$, (which in turn gives rise to the main x - y vortex, due to fluid friction, as was discussed Chapter 8). Since the mean velocity, $u_m(z)$, increases with z , if p is roughly the potential-flow value, it follows that the pressure maxima and minima must be of proportionally greater magnitude at $z^* = \frac{1}{2}$ than at $z = 0$. Analysis of Figs. 121 and 122 reveals this to be the case. Thus, there is a pressure gradient downward towards the fin at the front of the cylinder ($\theta \approx 0^\circ$), and upward away from the fin at the side ($\theta \approx 90^\circ$). It is believed that this axial pressure gradient not only gives rise to the z -oriented secondary flow, but also distorts the shape of the inter-tube vortices.

Figures 123 and 124 show contours of friction coefficient, $c_f(\phi, z^*)$. It can be seen that maximum shear occurs at around $\phi = 85^\circ$ (in-line), for $Re = 30$, [Fig. 123a] and moving slightly upstream with increasing Re [Fig. 123b,c]. For the staggered geometry [Fig. 124], the maximum shear occurs at around $\phi = 50^\circ$. In both cases, as might be expected the shear distribution is a maximum at $z^* = 1/2$. Other than for the data of Fig. 124c, comparison of the c_f values with the 2-D data [Figs. 58 and 59] reveals a consistent pattern.

Figures 125 and 126 show $Nu_T \overline{Nu}_T$ contours on the surface of the cylinder. It can be seen that for the in-line tube banks [Fig. 125], local heat transfer is substantially influenced by the finite-length of the tube (more so than form and profile drag). Inspection of the contours reveals that the $Nu_T \overline{Nu}_T$ distribution is relatively insensitive to Re , in the range $30 \leq Re \leq 300$. The maximum heat transfer of $2.8 \overline{Nu}_T$ at $Re = 30$ increases to 3.6 times the mean value, occurring in both cases at $\phi = 70^\circ$, i.e. slightly upstream of the minimum cross-section, and concentrated midway between fins at $z^* = 1/2$. As with much of the data above, this local maximum is somewhat greater than the peak observed for 2-D heat transfer [Fig. 70], although the form of distribution is qualitatively similar. Nu_T is based on the upstream fluid temperature, as was discussed in Chapter 8.

For the staggered arrangement, at $Re = 30$ the maximum of slightly less than $1.8 \overline{Nu}_T$ occurs at $\phi = 25^\circ$, increasing to around $2.7 \overline{Nu}_T$ and moving downstream to around 60° at $Re = 100$, i.e. the form of the distribution is more Re dependent than for the in-line bank. This behaviour is similar to that exhibited in Fig. 72. Figure 126c suggests that at $Re = 300$, the heat transfer distribution is unrealistic. Comparison of Figs. 125 and 126 with Figs. 123 and 124 suggest that heat transfer and skin friction profiles are substantially different, particularly for staggered tube banks.

10.3.4 Grid dependence and convergence

The grid used in this study were large by most present-day standards. For in-line geometry grids of $40 \times 40 \times 20$, $40 \times 40 \times 40$, $40 \times 40 \times 80$ were used corresponding to l/d values of $\frac{1}{2}$, 2, 10. For the staggered geometry, the equivalent values were $30 \times 40 \times 20$, $30 \times 40 \times 40$, $30 \times 40 \times 80$. Owing to the size of the meshes, no further refinement in the longitudinal direction was possible, at least for $l/d = 10$. The 2-D grid dependence tests used in Chapter 8 were used as a basis for determining these grid sizes: Since in all cases the cell size in the axial direction were as fine as, or finer than those in other directions, it is maintained that the 2-D grid dependence results provide a reasonable measure of confidence in the results. It should be noted that the use of the diagonal H-grid hinders the ability to concentrate cells in the region at the front of the cylinder $\phi = 0^\circ$, necessary to capture the fine detail of the upstream secondary flow. This is one situation where an in-line I-grid [Fig. 22] would have been superior for simulating flow in rotated square tube banks. For the reasons discussed in Chapter 8, this grid-type was not used.

For the largest grids used in this study, computer central processor unit limits necessitated the use of scratch files on disk. Full convergence required, in some cases in excess of 1 000 or more sweeps, and this combined with the necessary paging out to the above disk files meant that more than 24 computer central processor-hours per run were often required. It is for this reason that only a relatively small number of parametric runs were undertaken.

10.4 Closure

Flow and heat transfer in plate-fin fin and tube heat exchangers, i.e. tube banks of finite length, was considered. It was shown that some aspects of the flow are substantially dif-

ferent than for the two-dimensional case. However, provided the length-to-diameter ratio is of the order of 10:1 or greater, the irrecoverable pressure losses, and overall heat transfer factors differ negligibly from the two-dimensional values, for the *Re* range considered in this study. Local coefficients were displayed in the form of contour plots and discussed in detail.

11. PLATE-FIN HEAT EXCHANGER

11.1 Introduction

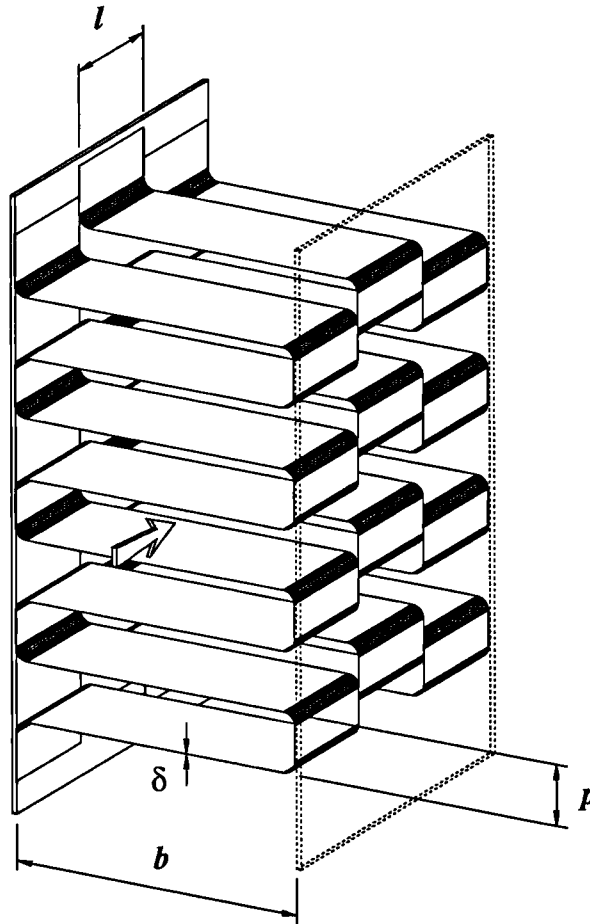


Figure 127. Rectangular offset fin plate-fin heat exchanger

An offset rectangular plate-fin surface, consists of two plates separated a distance, b , with a series of fin-strips, of length, l , thickness, δ , inserted in the passage between the two plates, as is shown in Fig. 127. The fin-strips are offset or staggered so that the leading edge of each strip is in the centreline of the previous fin. A heat exchanger is normally built out of a number of such units sandwiched together in parallel.

The work described in this chapter is essentially the same as was published by Beale (1990). The offset fin heat exchanger module was specifically chosen in order to validate the methodology used elsewhere in this thesis, using a problem involving solely Cartesian co-ordinates. The geometry of the problem considered, namely that of Core 105 in the paper by London and Shah (1968), and Shah and London (1967) was chosen as a validation model for the numerical model. This core had dimensions given below in Table 22.

Table 22. Core dimensions.

Quantity	Value	(Units)
b	1.905	(mm)
p	1.054	(mm)
l	2.822	(mm)
δ	0.1016	(mm)

A domain of $b \times p/2 \times l$ was considered. A grid of $30 \times 16 \times 40$ (not shown) was selected, a single slab of halo-cells being added downstream, as before. The grid was driven through the fin, which was covered by 4 and 2 cells in the x and y directions, representing distances of δ and $\delta/2$ in the horizontal and vertical directions respectively.

11.2 Boundary Conditions

The periodic boundary conditions are similar to those described in Chapter 8 and will not be repeated here. Only constant wall temperature boundary conditions were considered. The domain was chosen to bisect two consecutive fins; because of this, the periodic boundary conditions were prescribed using the simplified version (convection-only upstream) described in Chapter 8, except that the downstream enthalpy field was prescribed to take into consideration conduction in the fin. The entire procedure is much

simplified as a result of the Cartesian grid. The finite thickness of the fin was accounted for by using harmonic averaging of diffusion coefficients (Patankar, 1980). Laminar kinematic viscosity was stored whole-field, and set to a large number ($\nu_w = 10^6$), for cells covered by the fin, while in the flow ν was set to a value corresponding that of the working fluid (air), $\nu = \nu_f^1$. Because the flow is presumed to be steady, the fin density and specific heat are not relevant, and were set to fluid values without ambiguity. The fin Prandtl number, Pr_w , was thus set to $\rho_f c_{pf} \nu_f / k_w$, so that conjugate heat transfer in the fin was accounted for.

Core 105 (Shah and London, 1967, London and Shah, 1968) consisted of an Aluminium alloy fin, with air as heat transfer fluid. The exterior of the plate surfaces were maintained approximately constant by means of condensing steam. Properties used were based on values given in Appendix II of the report by Shah (1967), and are listed below in Table 23.

Table 23. Fluid/Solid Properties

Quantity	Value	(units)
ρ_f	0.9944	(kg/m ³)
μ_f	20.82	(μ Pa·s)
k_f	0.03035	(W/m·K)
c_p	1.016	(J/kg·K)

¹N.B. the subscript w refers to the solid (wall) of the fin and plate, and f refers to the fluid (not the fin).

11.3 Definitions

The flow Reynolds' number, Re , is based on a hydraulic diameter, D_h , defined by,

$$D_h = \frac{4A_c}{A} \quad (11.1)$$

where A_c is the minimum free-flow area,

$$A_c = (b - \delta)(p - \delta) \quad (11.2)$$

and A is the total heat transfer area, including all fin edges etc.,

$$A = 2l(b - \delta) + 2l(p - \delta) + 2b\delta + (p - \delta)\delta \quad (11.3)$$

Overall heat transfer and fluid flow friction are conveniently correlated in terms of Fanning friction factor, f , and Colburn heat transfer factor, j , defined by;

$$f = \frac{\tau}{\frac{1}{2}\rho u_m^2} \quad (11.4)$$

$$j = \overline{St} Pr_f^{\frac{2}{3}} \quad (11.5)$$

where τ is the equivalent shear stress (due to both form drag and skin friction) and u_m is the bulk velocity through A_c . It can be shown that f and j are given by,

$$f = \frac{D_K \Delta \bar{P}}{2l \rho u_m^2} \quad (11.6)$$

$$j = \frac{Pr_f^{\frac{2}{3}}}{\log \left(\frac{t_i - t_w}{t_e - t_w} \right)} \quad (11.7)$$

where, t_i and t_e are the incoming and outgoing bulk temperatures of the fluid, and t_w is the wall (i.e. the plate) temperature. Alternatively, the overall heat transfer may be expressed in terms of an average Nusselt number, $\overline{Nu}$ (based on the log-mean temperature) defined by,

$$\overline{Nu} = Re Pr \overline{St} \quad (11.8)$$

Local Nusselt numbers, Nu , were also computed,

$$Nu = k_{eff} \frac{dt}{dn} \frac{1}{(t_w - t_b)} \quad (11.9)$$

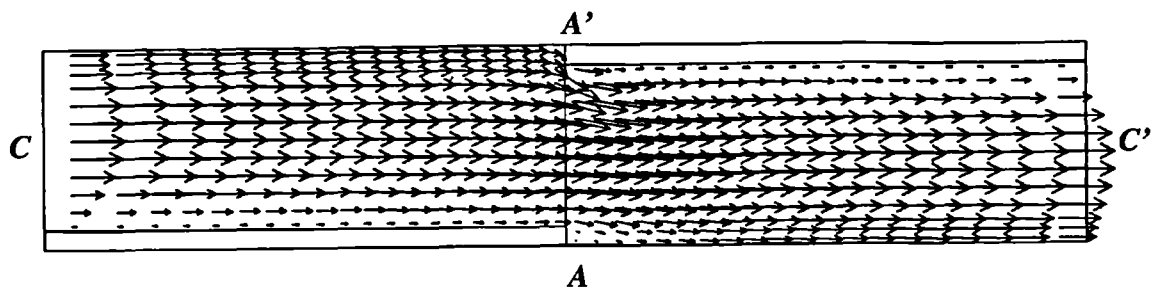
where dt/dn is the temperature gradient normal to the plate or fin, and k_{eff} is an effective thermal conductivity equal to either the harmonic average of k_f and k_w (at the fin-fluid interface), or just k_w (at the plate-interface), and t_b is a local bulk temperature.

11.4 Results

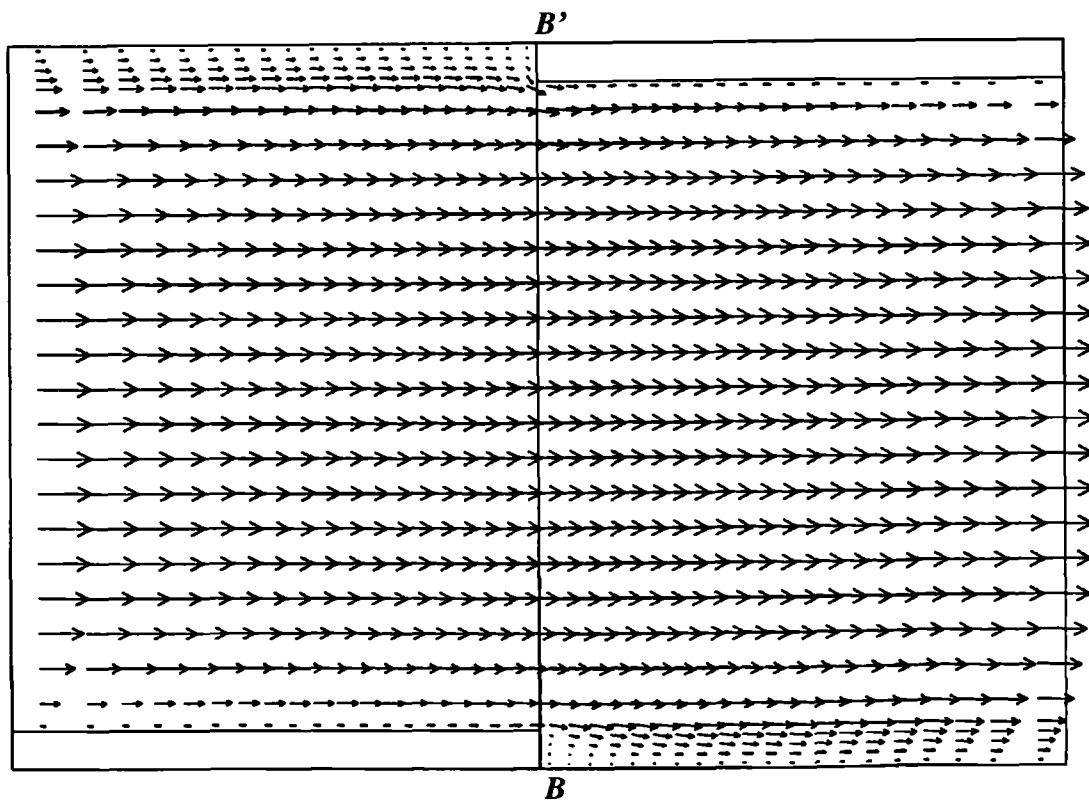
11.4.1 Typical flow-field results

Figure 128 shows velocity distributions in centrally-located planes. Figure 128a shows flow between adjacent fins (separated by distance $p/2$), while Fig. 128b shows the flow between the plates (separated by distance b). The inter-fin flow [Fig. 128a] may be likened to a boundary-layer or developing flow at the fin surface, with a developing wake at a distance $p/2$ from the surface. It can be seen that the flow in the inter-plate plane is similar to a fully-developed duct-flow, i.e. the velocity profile is parabolic-shaped. Some recirculation occurs behind the narrow backward-facing edges of the fin, however this is barely visible.

Figure 129 shows the pressure distribution on similar planes. It can be seen that there is a high pressure zone at the front leading edge of the fin [Fig. 129a] and the plate [Fig. 129b]. The pressure decreases rapidly at the front of the fin, as indicated by the concentration of contours (resembling a dipole) in this region. Twin low pressure zones are observed at the trailing edge of the fin. Note that the pressure contours are concentrated at the start of the fin row [B-B' in Fig. 129b], indicating that pressure losses are a maximum in this region.

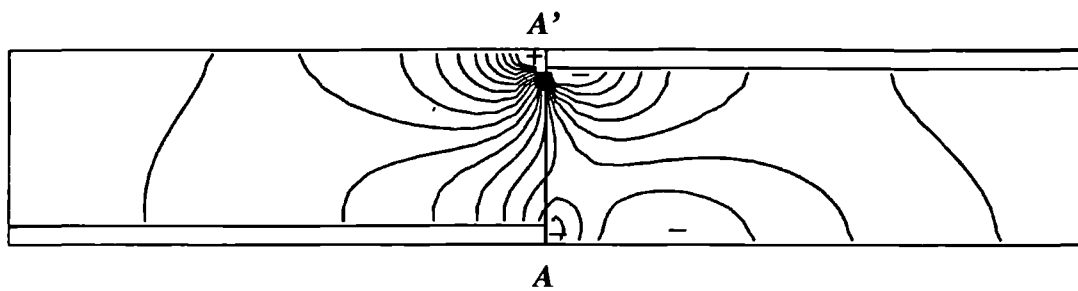


(a) Between Fins

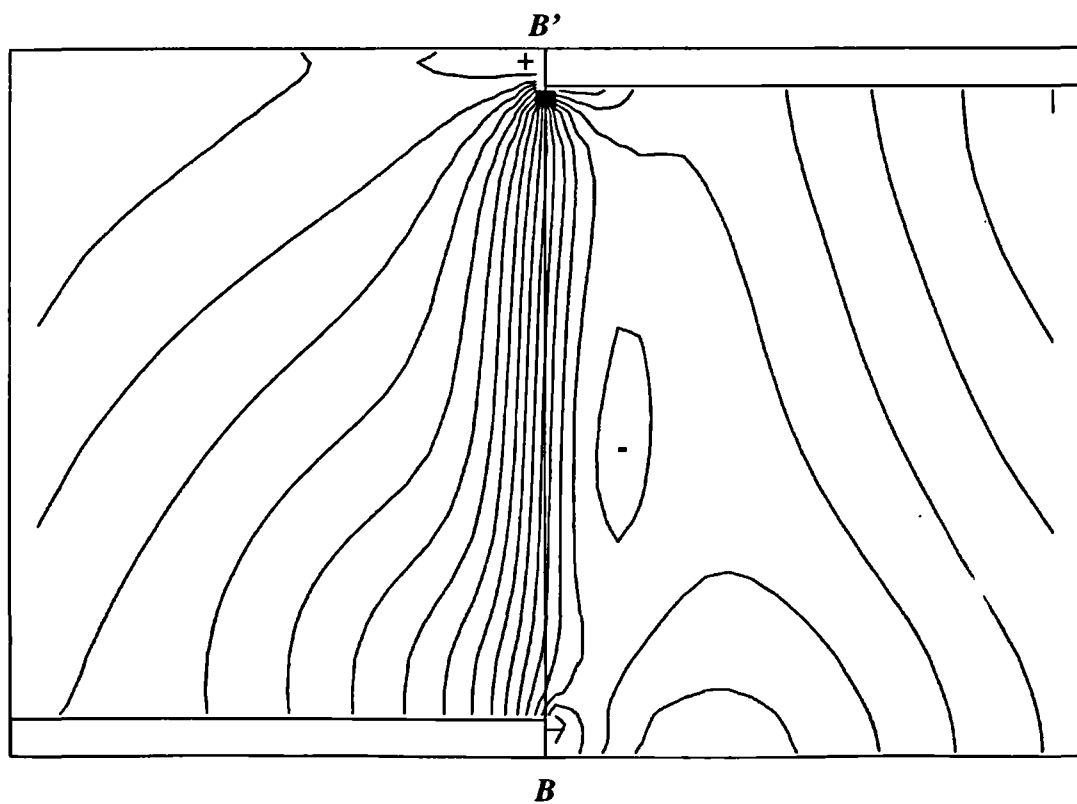


(b) Between Plates

Figure 128. Velocity vectors, $Re = 500$.



(a) Between Fins



(b) Between Plates

Figure 129. Pressure contours, $Re = 500$.

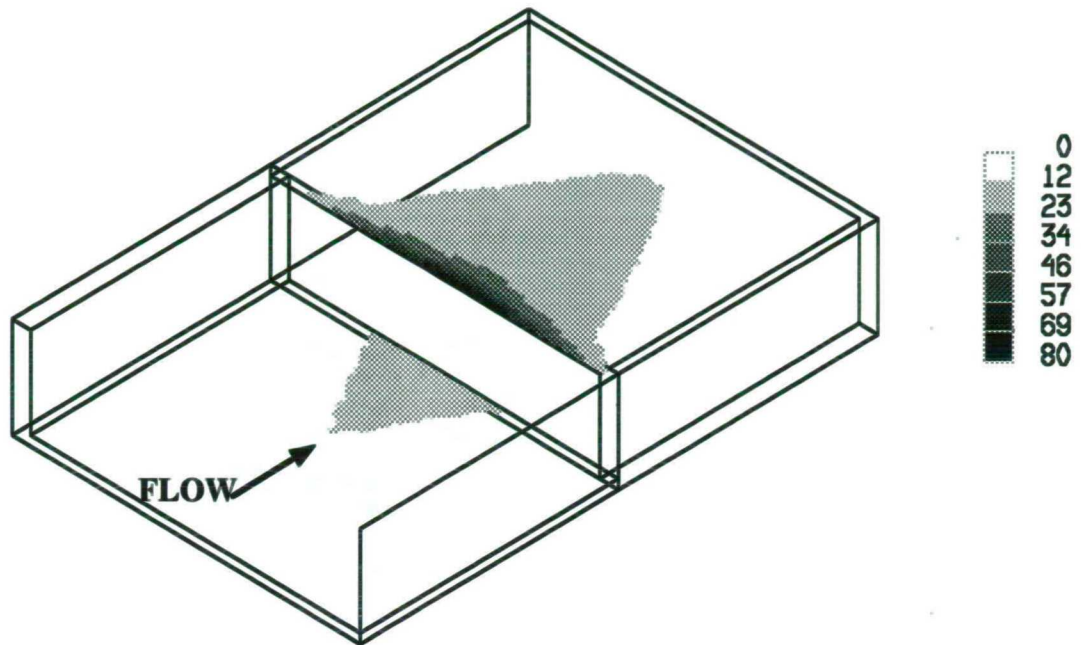


Figure 130. Local heat transfer coefficient, $Re = 500$.

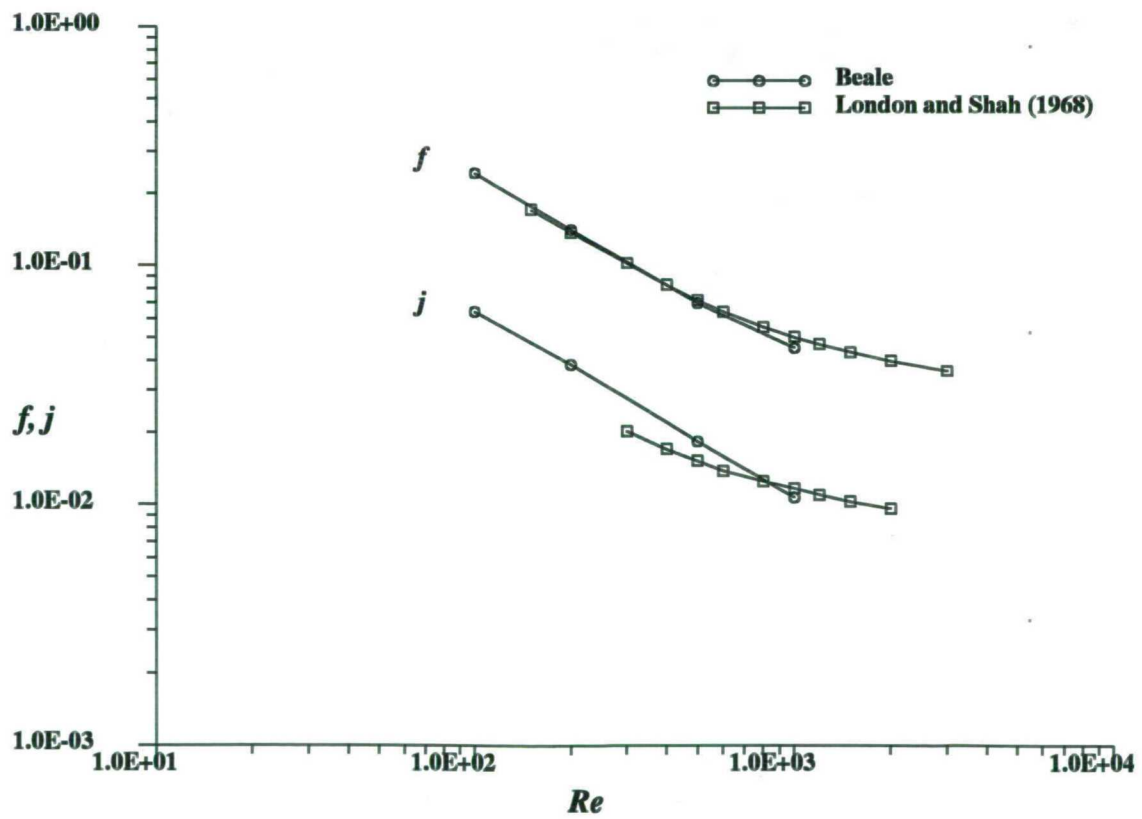


Figure 131. Comparison of results with experimental data.

11.4.2 Pressure drop and heat transfer data

Figure 130 shows the distribution of the local Nusselt number, Nu , on the horizontal fin surface for $Re = 500$. The maximum local Nusselt number is approximately 10 times the mean value, and occurs towards the centre-front region of the fin. A second (much smaller) local maximum in Nu is seen towards the rear of the preceding fin. Values of Nu on the plate and vertical portions of the fin (not shown) were close to the mean value of 8.1.

Table 24. Results for plate-fin heat exchanger.

Re	f	j
100	0.243	0.00639
200	0.148	0.00384
500	0.0696	0.00183
1 000	0.0451	0.00107

Table 24 is a summary of the overall results of this numerical study in terms of the friction coefficient f , and heat transfer j . Computed values of the friction coefficient, f , range from 0.243 at $Re = 100$, to 0.0451 at $Re = 1\,000$. Values of j range from 0.064 at $Re = 100$ to 0.011 at $Re = 1\,000$. This corresponds to $\overline{Nu}_{LM}$ of 5.7 and 10.4, respectively. Figure 131 is a comparison of the results of this study with the smoothed experimental data of London and Shah (1968). Agreement with the experimental data is excellent. It can be seen that there is fair agreement for heat transfer as a function of Re , when compared to experimental results, and that the slope of the computed j vs. Re curve, while parallel to the computed f vs. Re , is significantly steeper than the experimentally obtained j vs. Re curve.

11.5 Discussion

Inspection of Fig. 129 shows that the pressure in the transverse plane to be a maximum at the front of the fin (Point A'), where the flow impinges on the leading edge. The pressure decreases extremely rapidly in this region, as indicated by the convergence of the isobars around Point A'. A local minimum occurs in the rear stagnation zone at the fin trailing edge (Point A). This pressure distribution is characteristic of all rectangular offset fin surfaces that were analysed; i.e. the pressure distribution for this class of flow is fundamentally different from that assumed in conventional flat-plate boundary-layer theory.

Because of the finite fin thickness, the flow accelerates at the start of each fin row [$B-B'$, Fig. 128] where there is a contraction in the free-flow area. The pressure gradient is a maximum along this front, suggesting that drag also is a maximum in this region. Additional runs conducted assuming negligible fin-thickness ($\delta = 0$) exhibited pressure contours similar to those in Fig.129a, however the span-wise distribution, in contrast to the distribution shown in Fig. 129b, was relatively constant.

At moderate Re , a small recirculation zone is established in the stagnation zone behind the trailing edge of the fin. Although hardly visible, recirculation is important in that it affects both drag and heat transfer. The recirculation zone is 'L-shaped' corresponding to the backward-facing trailing edge of the fin surface. Associated with this zone is a pressure trough located downstream of the trailing edge of the fin. The trough occurs behind both the vertical and horizontal fin edges (the apparent minimum at the centre of Fig. 129a) is only present at high Re , (in fact the same trough that occurs in Fig 129b). These secondary extrema occur only for $\delta > 0$. As Re is increased, inter-fin flow development decreases due to the continual restarting of the boundary layer; inspection of

Fig. 128 shows the peak velocity is actually located between the wake region and the boundary layer zone (along $C-C'$).

Figure 130 shows that the local Nusselt number, Nu , is a maximum of approximately ten times the mean value, in a central zone near the front of the fin. Although the average heat transfer coefficient increases with Re , there is not a great deal of change in the local Nu distribution as a function of Re : Inspection of Fig. 130 reveals that local heat transfer at the trailing edge of the fin is also a local maximum. This is due to recirculation, and, as expected, this zone expands with increasing Re . The reader will note that concerns about the choice of reference temperature [Chapter 8] are less relevant here. Because of the streaming nature of the flow, and the rectilinear geometry, local Nu are based on values of local bulk temperature (and velocity), and not the upstream fluid temperature, or some other module-average value.

The results of this study, in terms of Colburn heat transfer and friction factors vs. Re , as shown in Fig. 131, are quite good. The friction coefficients, in particular, show excellent agreement with the smoothed data of London and Shah (1968) up to $Re = 1\,000$. At higher Re , one would anticipate that significant free-stream turbulence would result in change in the slope of the f vs. Re curve exhibited by the experimental data in Fig. 131. The reason for the poorer agreement between computed and measured values of the heat transfer factor is not clear. It is to be noted, however, that the slope of the computed Colburn heat transfer factor is parallel to the friction coefficient curve, and that experimental measurements were made over a relatively narrow Re range (compared to flow friction data). Both the grid and the flow regime in this problem are simpler than those associated with tube banks; better agreement with experimental data might be anticipated.

The results of this study also showed better agreement with the experimental data than those of previous numerical results (Sparrow et al. 1977, Patankar and Prakash 1981),

where computed friction coefficients were significantly lower and heat transfer coefficients higher than experimentally measured values. This is due to the fact that, not only was the finite thickness of the fin, but also the three-dimensional nature of the flow was accounted for in the model used in this study.

It is observed from Fig. 128b that the velocity profile is predominantly parabolic-shaped in the plane between the plates. Because of this fact, the rate of shear is always a maximum at a distance somewhere midway between the two plates, and as a result, local drag and heat transfer are a maximum in a zone at the central leading edge of the fin. This is clearly shown in Fig. 130. For laminar flow in a plate-fin heat exchanger, drag and heat transfer will always be a maximum towards the front of the fin midway between the plates regardless of aspect ratio, fin thickness etc. Any problem posed in a 2-D form would imply a slug-shaped profile, so that the local maxima would be lines rather than point-zones. In offset fin heat exchangers, three-dimensional effects are as important as those associated with the fin thickness, for the Re range under consideration: This explanation was borne out using the thin-fin analysis mentioned above: 3-D thin fin drag coefficients exhibited some deviation from the results presented here, while 2-D thin fin results differed substantially more. (Some caution is required when directly comparing different sets of results, because definitions of D_h , A_c , and A , and hence Re , f , and Nu are affected significantly by the assumptions). It is interesting to note that the opposite conclusion was reached when studying flow in high-aspect ratio fin and tube heat exchangers: In plate-fin heat exchangers, the periodic interruptions do not prevent the flow from developing into a parabolic-shaped profile, in offset fin heat exchangers, the converse is true.

Associated with $\overline{Nu}$ is a (dimensional) overall heat transfer coefficient, $U = \overline{Nu} k_f / D_h$. Shah and London (1967) calculated an overall heat transfer coefficient, $\bar{h}$,

$$\bar{h} = \frac{U}{\eta_0} \quad (11.10)$$

where η_0 is the so-called overall temperature effectiveness of the fin, taking into account (approximately) the variation of temperature along the horizontal portion of the fin. Values of η_0 were computed and found to be in excess of 99%, indicating that in spite of the fact that an Aluminium fin was used, fin conduction is very high, so that essentially identical results would have been obtained had the fin material been assumed to be at constant temperature. This would not necessarily be the case if different materials, or geometry were employed.

11.6 Closure

Laminar fluid flow and heat transfer in a rectangular offset fin plate-fin heat exchanger surface was analysed. Excellent agreement was obtained between computed friction coefficients and the smoothed data of London and Shah (1968). Agreement between heat transfer factors was quite good, with computed values being higher than measured values. The transport processes in offset fin surfaces are affected both by the three-dimensional nature of the flow, and by the finite thickness of the fin.

12. CONCLUSIONS AND RECOMMENDATIONS

This thesis was concerned with the analysis of fluid flow and heat transfer in common heat exchanger design configurations, primarily tube banks. Both analytical and numerical methods were employed.

The governing equations of motion in general curvilinear co-ordinates were presented and discussed in detail, it being shown that the resolute form is particularly simple, and readily comparable to the Cartesian equations. This form of the transport equations may then be integrated and solved, using the finite-volume method, subject to modifications.

An analytical solution for potential flow showed that widely-spaced tube banks exhibited potential fields similar to those associated with a single cylinder. In compact banks, however, there were substantial differences. Coefficients were tabulated to facilitate calculation of the potential function. The potential-flow pressure and velocity distributions were described in detail.

A grid generation technique based on the solution of Poisson equations was detailed. The use of partial-differential equations for grid generation can generate good grids, however this is largely dependent upon the skill and experience of the programmer. There does not appear to be any *a priori* method of prescribing source terms; both skill and experience are required. A generalised procedure for sliding boundary points (Neumann boundary condition) on arbitrarily shaped boundaries is highly desirable. The use of alternative procedures such as a biharmonic solver (corresponding to the creeping-flow analogy) should be considered. For the type of flow solver used in this study, orthogonal curvilinear grids should be employed unless there is good reason to do otherwise.

An analysis was presented for numerically generating stream-functions. In two-dimensions, non-iterative techniques should be employed under most circumstances. Exceptions to this rule include situations where multi-block or highly non-orthogonal grid are employed. Under these circumstances a finite-volume-based solution to the stream function-vorticity equation is recommended. A technique for generating stream-lines and stream-surfaces in three-dimensions was also detailed. A number of difficulties prevented the technique from being applied to the flow-field solutions generated in this thesis. This topic should be considered as a subject worthy of future research, since it would provide a useful tool for viewing the results of CFD work using a field-approach.

A detailed numerical analysis of steady two-dimensional flow and heat transfer revealed much information about flow and heat transfer in tube banks. It was shown that in the laminar regime, heat transfer coefficients are strongly affected by the choice of reference temperature. Correlation between the numerical results and experimental data was only fair, though correlation with previous numerical work was in general good. Substantial spread between the third party experimental data and various empirical correlations was also noted. Since flow and heat transfer in tube banks is an important subject, it would be desirable for the research community to arrive at a consensus, both in regard to a consistent set of parameters and the appropriate correlations. If necessary 'round-robin' tests should be conducted.

The level of effort required in grid generation is directly related to the sophistication and robustness of the flow solver. The H-grids that were used, for the most part, in this study suffer from a number defects; I-grids or O-grids could be more readily concentrated in the boundary-layer. The sensitivity of the flow-solver to the location of the special point is a matter for concern: Higher-order schemes which are flow-oriented rather than grid-oriented should be considered in the future. In the event that such schemes can be shown to eliminate the constraints upon the choice of grid type, and lo-

cation of the special point(s), structured grids will probably continue to be used for some time in the future.

The stability of flow in tube banks was analysed. It was demonstrated that small upstream fluctuations are amplified by via the mean flow, resulting in a net gain in the fluctuating component, downstream. Vortex shedding and shear layer instabilities were observed in both in-line and staggered geometries. In the staggered arrangement the wake switched from side-to-side. The results are qualitatively similar to experimentally-obtained flow visualisation studies, although some specific differences were apparent. Pressure drop, lift, drag, and heat transfer were all shown to strongly affected by the transient nature of the flow. Transient flow offers significant challenges; fine grid, sufficient time-steps, and well-posed boundary conditions are prerequisite. For recirculating flows, when the boundaries are close, the question of how best to prescribe the boundary conditions is a major issue. Future research should focus not only on this, but also on the application of different solvers, other than the implicit scheme used in this study.

The analysis of three-dimensional flow and heat transfer in plate-fin fin and tube heat exchangers showed evidence of the presence of secondary flow patterns, and their influence upon pressure drop and heat transfer. A numerical study of the stability of such flows would be of great interest, and could probably be undertaken relatively soon, as computer storage and memory increase sufficiently. The steady-state analysis showed that although most measures of performance are different from those obtained using the two-dimensional analysis, for large length-to-diameter ratio, the overall pressure drop, and heat transfer coefficients approach the two-dimensional value. An analysis of an offset fin plate-fin surface yielded good agreement with existing experimental data.

Turbulent transport was not considered in this study. A number of previous workers have simulated turbulent flow in tube banks using transport models (such as the well-known k - ϵ model). Such models have a limited range of validity, at least one order of

magnitude above the critical Re of 2×10^5 . Few heat exchangers ever actually operate at these very high Re . Most operate in the intermediate range $10^2 < Re < 10^5$. In this regime, the boundary-layer is laminar, but the free stream is turbulent. Knowledge of the spectral distribution of the turbulence for this class of flow is lacking. There may also be low frequency periodic phenomena occurring at certain frequency bands. It is perhaps indicative of the limitations of the present-day analytical tools, that such a project was not undertaken. Large-eddy simulation may well hold some promise for the future. The challenges involved in modelling fluid-flow and heat transfer in the intermediate Re range should not be understated.

Significant breakthroughs are still required, before numerical analysis will supersede empirical work in heat exchanger design. However understanding of the basic transport processes within the passages of typical heat exchanger elements has been significantly enhanced by the application of computational fluid dynamics.

13. NOMENCLATURE

English letter symbols

A	Area, total heat transfer area	(m^2)
A_c	Minimum free-flow area	(m^2)
A_i	Coefficient used to compute B_i	(m^2/s)
A_{ij}	Coefficient used to compute A_i	(m^2/s)
A^{ij}	Adjugate of α_{ij}	()
a	Coefficient in finite-volume equation	(kg/s^2)
a	Line attraction strength	(any)
B_i	Coefficient in power series for complex potential	(m^2/s)
b	Heat exchanger plate width	(m)
b	Line attraction decay factor	(any)
C	Constant	(any)
C	Source term coefficient	(kg/s^2)
C'	Pressure coefficient	($kg/m \cdot s$)
c	Constant	()
c	Point attraction strength	(any)
c_k	Constant use to calculate Wierstrass' ζ function	()
p	Specific heat	($J/kg \cdot ^\circ K$)
c_1	Enthalpy scale factor	()
c_2	Enthalpy scale factor	()
D_h	Hydraulic diameter	(m)
d	Cylinder diameter	(m)
d	Diffusion coefficient	(kg/s),
d	Point attraction decay factor	(any)
e^i	Cartesian components of contravariant basis vector	(any)

e_i	Cartesian components of covariant basis vector	(any)
F	Coefficient used to convert tangential to normal resolute	()
f	Frequency	(Hz)
G	Coefficient used to convert tangential to normal resolute	()
g^{ij}	Contravariant (inverse) metric tensor	(any)
g_{ij}	Covariant metric tensor	(any)
g	Square of Jacobian, $\det g_{ij}$	(any)
H	Coefficient used to convert tangential to normal resolute	()
h	Enthalpy	(W/m ²)
h	Local heat transfer coefficient	(W/m ² ·°K)
h	Heat transfer coefficient	(W/m ² ·°K)
J	Total flux	(any)
k	Thermal conductivity	(W/m·°K)
l	Tube length, fin length	(m)
m	Mass flux, convection coefficient	(kg/s)
n	Normal	()
n^i	Cartesian components of unit normal	()
P	Fin pitch	(m)
P	Control-function	()
p	Pressure	(N/m ²)
Q	Heat transfer rate	(W)
Q	Control-function	()
q	Heat flux	(W/m ²)
q	Diffusion flux	(any)
r	Position	(m)
r_0	Radius	(m)
r	Residual continuity error	(kg/s)
S	Source term in finite-volume equation	()
s	Pitch	(m)

s_{ij}	Covariant components of velocity gradient tensor	(any)
s_1	Cross-wise pitch	(m)
s_2	Stream-wise pitch	(m)
T	Time	(s)
t	Tangent	()
t	Temperature	(°K)
t	Time	(s)
t_i	Cartesian components of unit tangent	()
u	x -direction velocity, tangential velocity resolute	(m/s ²)
u^i	Contravariant velocity component	(any)
u_i	Covariant velocity component	(any)
U	Complex velocity	(m/s)
U	Overall heat transfer coefficient	(W/m ² ·°K)
V	Source-term value	(arbitrary)
V	Volume	(m ³)
v	y -direction velocity, tangential velocity resolute	(m/s ²)
W	Complex potential	(m ² /s)
W_i	Potential function	(m ² /s)
w	z -direction velocity, tangential velocity resolute	(m/s ²)
x	Displacement, tangential displacement component	(m)
x^i	Contravariant displacement component	(any)
y	Displacement, tangential displacement component	(m)
z	Complex displacement	(m)
z	Displacement, tangential displacement component	(m)

Greek letter symbols

α	$\det \alpha_{ij}$	()
α	Linear relaxation factor	()
α^{ij}	Direction cosines between unit normals	()
α_{ij}	Direction cosines between unit tangents	()
α_i^i	Direction cosines between unit tangent and normal	()
α_{ij}	Coefficients in elliptic function power series	(m ² /s)
Γ	Exchange coefficient	(kg/m·s)
δ	Covariant, tangential derivative operator	()
δ	Fin thickness	(m)
ζ	Weirstrass' function	(any)
η_0	Temperature effectiveness	()
θ	Argument of non-dimensional complex displacement	()
θ_{ij}	Angle between tangents $\vec{t}_i$ and $\vec{t}_j$	()
θ^{ij}	Angle between normals $\vec{n}^i$ and $\vec{n}^j$	()
θ_i^j	Angle between tangent $\vec{t}_i$ and normal $\vec{n}^j$	()
μ	Dynamic viscosity	(m ² /s)
ν	Kinematic viscosity	(m ² /s)
ρ	Density	(kg/m ³)
ρ	Modulus of non-dimensional complex displacement	()
Σ'	Double summation operator excluding double zero	()
σ	Stress	(N/m ²)
σ	Weirstrass' function	(any)
τ	Period, phase	(s)
τ	Shear stress	(N/m ²)
Φ	Viscous dissipation	(m ² /s ²)
ϕ	Angle from front leading-edge of cylinder	(°)
ϕ	Generalised variable	(any)
ϕ	Velocity potential	(m ² /s)

χ_j^i	Transformation matrix	()
ψ	Stream function	(m ² /s)
$\Omega_{m,n}$	Location of cylinder centres in complex plane	(m)
ω	Angular frequency	(rad/s)
ω	Vorticity	(/s)
ω_1	Complex primitive half period	(m)
ω_2	Complex primitive half period	(m)

Other symbols

∂	Partial derivative operator	()
$\wp$	Weirstrass' function	(any)

Non-dimensional numbers

a	Cross-wise pitch-to-diameter ration, s_1/d
b	Stream-wise pitch-to-diameter ratio, s_2/d
B	Modified Nusselt number, $Nu Pr^{-0.36}(\mu_w/\mu)^{0.14}$
c_D	Drag coefficient, $Drag/1/2\rho u_m^2 d$
c_f	Friction factor, $\tau_w/1/2\rho u_m^2$
c_L	Lift coefficient, $Lift/1/2\rho u_m^2 d$
c_p	Pressure coefficient, $1-(p_0-p)/1/2\rho u_m^2$
Eu	Euler number, $\Delta\bar{p}/1/2\rho u_m^2$
j	Colburn heat transfer factor, $St Pr^{2/3}$
j'	Modified Colburn heat transfer factor, $j Pr^{-1/3}(\mu/\mu_w)^{0.14}$
K	Heat transfer factor, $St Pr^{-0.36}(Pr_f/Pr_w)^{-0.25}$
Nu	Nusselt number, hd/k
Pe	Peclet number, m/d ,

Pe	Peclet number $RePr$
Pr	Prandtl number, $\mu c_p/k$
Re	Reynolds' number, $\rho u_m d/\mu$
Sh	Strouhal number, fd/u_m
St	Stanton number, $h/\rho c_p u_m$
z^*	Non-dimensional distance, z/l
η	Temperature effectiveness, U/h
μ	Non-dimensional radius, r_0/s_2
θ	Non-dimensional temperature, $(t - t_i)/(t_e - t_i)$

Superscripts

i	Contravariant component
(nr)	Normal resolute
T	Transpose
(tc)	Tangential component
$-$	Inverse matrix
$-$	Mean value, non-periodic component
$\sim$	Periodic, doubly-periodic component
$\rightarrow$	Vectorial quantity
$*$	Non-dimensional
$*$	Guessed value
$'$	Correction value
$'$	Projection in a fixed direction

Subscripts

<i>b</i>	Bulk
<i>c</i>	Cartesian
<i>d</i>	Downstream
<i>E</i>	Node centre of east cell
<i>e</i>	East cell interface
<i>e</i>	Exit grid boundary
<i>eff</i>	Effective
<i>f</i>	Fluid, upstream fluid
<i>false</i>	False
<i>fin</i>	Fin
<i>H</i>	Constant heat flux
<i>H</i>	Node centre of high cell
<i>h</i>	High cell interface
<i>i</i>	Inlet grid boundary
<i>i</i>	Covariant component
<i>L</i>	Node centre of low cell
<i>LM</i>	Log mean
<i>l</i>	Low cell interface
<i>m</i>	Mass
<i>m</i>	Mean, bulk value
<i>mon</i>	Monitor-point value
<i>N</i>	Node centre of north cell
<i>n</i>	North cell interface
<i>n</i>	Normal
<i>(nc)</i>	Normal component
<i>P</i>	Node centre of cell
<i>P</i>	Pressure

<i>ref</i>	Reference value
<i>S</i>	Node centre of south cell
<i>s</i>	South cell interface
<i>T</i>	Previous time step
<i>t</i>	Tangential
<i>(tr)</i>	Tangential resolute
<i>u</i>	x-direction staggered cell
<i>u</i>	Upstream
<i>v</i>	y-direction staggered cell
<i>W</i>	Node centre of west cell
<i>w</i>	Wall
<i>w</i>	West cell interface
<i>w</i>	z-direction staggered cell
<i>x</i>	x-direction
<i>y</i>	y-direction
<i>z</i>	z-direction
<i>0</i>	Reference value, overall
<i>,</i>	Covariant derivative operator

14. BIBLIOGRAPHY

1. Abd-Rabbo, A., and Weaver, D.S. 1986. A Flow Visualisation Study of Flow Development in a Staggered Tube Array. *Journal of Sound and Vibration*, Vol.106, No. 2, pp. 241-256.
2. Abramowitz, M., and Stegun, I.A. 1970. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards Applied Mathematics Series No. 55, United States Government Printing Office, Washington.
3. Achenbach, E. 1969. Investigations On The Flow Through a Staggered Tube Bank at Reynolds Numbers up to $Re = 10^7$. *Wärme-und Stoffübertragung*, Vol. 2, pp. 47-52.
4. Achenbach, E. 1989. Heat Transfer From a Staggered Tube Bank in Cross-flow at High Reynolds Numbers. *International Journal of Heat and Mass Transfer*, Vol. 32, No. 2, pp. 271-280.
5. Antonopoulos, K.A. 1979. Prediction of Flow and Heat Transfer in Rod Banks. Ph.D. Thesis, Imperial College, University of London.
6. Antonopoulos, K.A. 1985. Heat Transfer in Tube Assemblies under Conditions of Laminar, Axial, Transverse and Inclined Flow. *The International Journal of Heat and Fluid Flow*. Vol. 6, pp. 193-204.

7. Antonopoulos, K.A. 1987a. Pressure Drop during Laminar Oblique Flow through In-Line Tube Assemblies. *International Journal of Heat and Mass Transfer*, Vol. 30, No. 4, pp. 673-681.
8. Antonopoulos, K.A. 1987b. The Prediction of Turbulent Inclined Flow in Rod Banks. *Computers and Fluids*. Vol. 14, No. 4, pp. 361-378.
9. Aris, R. 1962. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Prentice-Hall, Englewood Cliffs, N.J.
10. Batchelor, G.K. 1967. *An Introduction to Fluid Mechanics*. Cambridge University Press, London.
11. Beale, S.B. 1990. Laminar Fully Developed Flow and Heat Transfer in an Offset Rectangular Plate-fin Surface. *The PHOENICS Journal of Computational Fluid Dynamics and its Applications*. Vol. 3, No. 1, pp. 1-38.
12. Bergelin, O.P., Brown, G.A., and Doberstein, S.C. 1952. Heat-transfer and Fluid Friction during flow across Banks of Tubes, Part IV. *Transactions of the ASME*., Vol. 74, pp. 953-960.
13. Bergelin, O.P., Brown, G.A., Hull, H.L., and Sullivan, F.W. 1950a. Heat-transfer and Fluid Friction during Viscous Flow across Banks of Tubes, Part III. *Transactions of the ASME*, Vol. 72, pp. 881-888.
14. Bergelin, O.P., Colburn, A.P., and Hull, H.L. 1950b. Heat Transfer and Pressure Drop during Viscous Flow across Unbaffled Tube Banks. *Engineering Experiment Station Bulletin No. 2*, University of Delaware.

15. Bergelin, O.P., Davis, E.S., and Hull, H.L. 1949. A Study of Three Tube Arrangements in Unbaffled Tubular Heat Exchangers. Transactions of the ASME, Vol. 71, pp. 369-374.
16. Bergelin, O.P., Leighton, M.D., Lafferty, W.L., and Pigford, R.L. 1958. Heat Transfer and Pressure Drop during Viscous and Turbulent Flow across Baffled and Unbaffled Tube Banks. Engineering Experiment Station Bulletin No. 4, University of Delaware.
17. Burns, A.D., and Wilkes, N.S., 1987. A Finite Difference Method for the Computation of Fluid Flows in Complex Three Dimensional Geometries. AERE-R 12342, Harwell.
18. Caretto, L.S., Gosman, A.D., Patankar, S.V., and Spalding, D.B. 1973. Two Calculation Procedures for Steady, Three-dimensional Flows with Recirculation. Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics. Springer Verlag - Lecture Notes in Physics. Vol. 2, No. 19, pp. 60-68.
19. Chang, K.S., and Song, C.J. 1990. Interactive Vortex Shedding from a Pair of Circular Cylinders in a Transvers Arrangement. International Journal for Numerical Methods in Fluids, Vol. 11, pp. 317-329.
20. Chang, Y., Beris, A.N., and Michaelides, E.F. 1989. A Numerical Study of Heat and Momentum Transfer for Tube Banks In Crossflow. International Journal for Numerical Methods in Fluids, Vol. 9, pp. 1381-1394.
21. Chen, K.C., Wong, K.L., Cleaver, J.W. 1986. Finite Element Solutions of Laminar Flow and Heat Transfer of Air in a Staggered and an In-line Tube Bank. The International Journal of Heat and Fluid Flow, Vol. 7, No. 7.

22. Chen, S.S. 1978. Crossflow-induced Vibrations of Heat Exchanger Tube Banks. Nuclear Engineering and Design, Vol. 47, pp. 67-86.
23. Dalton, C., and Helfinstine, R.A. 1971. Potential Flow Past a Group of Circular Cylinders. Transactions of the ASME Volume D, Journal of Basic Engineering, Vol. 93, pp. 636-642.
24. Davidson, L., and Hedberg, P. 1989. Mathematical Derivation of a Finite Volume Formulation for Laminar Flow in Complex Geometries. International Journal for Numerical Methods in Fluids, Vol. 9, pp. 531-540.
25. Demirdzic, I., Gosman, A.D., Issa, R.I., and Peric, M. 1987. A Calculation Procedure for Turbulent Flow in Complex Geometries. Computers and Fluids. Vol. 15, No. 3, pp. 251-273.
26. Dhaubhadel, M.N., Reddy, J.N., and Telionis, D.P. 1987. Finite-element Analysis of Fluid Flow and Heat Transfer for Staggered Banks of Cylinders in Cross Flow. International Journal for Numerical Methods in Fluids, Vol. 7, pp. 1325-1342.
27. Drotz, A. 1987. Détermination analytique de potentiels d'écoulements non visqueux, incompressibles et bidimensionnels autour d'obstacles circulaires. Journal de Mécanique théorique et appliquée. Vol. 6, No. 1, pp. 23-45.
28. Eckerle, W.A., and Langston, L.S. 1987. Horseshoe Vortex Formation around a Cylinder. Journal of Turbomachinery, Vol. 109, pp. 278-285.
29. Engineering Sciences Data Unit. 1973. Convective heat transfer during crossflow of fluids over plain tube banks. ESDU Data Item No. 73031, London, November.

30. Engineering Sciences Data Unit. 1980. Pressure Loss during Crossflow of Fluids with Heat Transfer over Plain Tube Banks without Baffles. ESDU Data Item No. 74040, London.
31. Faghri, M., and Rao, N. Numerical Computation of Flow and Heat Transfer in Finned and Unfinned Tube Banks. International Journal for Heat and Mass Transfer, Vol. 30, No. 2, pp. 363-372.
32. Fujii, M., Fujii, T, and Nagata, T. 1984. A Numerical Analysis of Laminar Flow and Heat Transfer of Air in an In-line Tube Bank. Numerical Heat Transfer, Vol. 7, pp. 89-102.
33. Giese, J.H. 1951. Stream Functions for Three-Dimensional Flows. Journal of Mathematics and Physics, Vol. 30, pp. 31-35.
34. Glynn, D.R., and Rawnsley, S.M. 1988. Numerical Prediction of a Horseshoe Vortex. Advances in Underwater Technology, Ocean Science and Offshore Engineering, Vol. 15. pp. 145-155.
35. Gram, A.J., Mackey, C.C, and Monroe, E.S. 1958. Convection Heat-transfer and Pressure-drop of Air Flowing across In line Tube Banks. Transactions of the ASME, Vol. 80, pp. 25-35.
36. Grimison, E.D. 1937. Correlation and Utilisation of New Data on Flow Resistance and Heat Transfer for Crossflow of Gases Over Tube Banks. Transactions of the ASME, Vol. 59, pp. 583-594.
37. Hammeke, K., Heinecke, E., and Scholz, F. 1967. Wärmeüberangs-und druckverlustmessungen an querangeströmten glattrohrbündeln, insbesondere bei hohen rey-

- noldszahlen. *International Journal of Heat and Mass Transfer*, Vol. 10, pp. 427-446.
38. Hancock, H. 1958. *Theory of Elliptic Functions*. Dover, New York.
 39. Harlow, F.H., and Welch, J.E. 1965. Numerical Calculation of Time-dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids*. Vol. 8, No. 12, pp. 2182-2189.
 40. *Heat Exchanger Design Handbook*. 1983. Vol. 1-4, Hemisphere, New York.
 41. Howland, R.C.J., and McMullen, B.W. 1936. Potential Functions Related to Groups of Circular Cylinders. *Proceedings of the Cambridge Philosophical Society*, Vol. 32, pp. 402-415.
 42. Hsu, C.F. 1981. A Curvilinear-coordinate Method for Momentum, Heat and Mass Transfer in Domains of Irregular Geometry. Ph.D. Thesis. University of Minnesota.
 43. Hsu, C.J. 1964. Analytical Study of Heat Transfer to Liquid Metals in Cross-flow through Rod Banks. *International Journal of Heat and Mass Transfer*, Vol. 7, pp. 431-446.
 44. Huge, E.C., 1937. Experimental Investigation of Effects of Equipment Size on Heat Transfer and Flow Resistance in Crossflow Tube banks. *Transactions of the ASME*, Vol. 59, pp. 563-572.
 45. Isihara, K., and Bell, K. 1975. Friction factors for In-line Tube Banks at Low Reynolds Numbers. *AIChE Symposium Series*, Vol. 68, pp. 74-80.

46. Jacob, M. 1949. Heat Transfer. Vol. 1. Wiley. New York.
47. Jones, C.E., and Monroe, E.S. 1958. Convection Heat-transfer and Pressure-drop of Air Flowing across In-line Tube Banks, Part I. Transactions of the ASME, Vol. 80, pp. 18-24.
48. Katinas, V.I., Markevicius, A.A., Rodionov, V.V., and Alekseyev, S.A. 1989. Transient Flow Forces Induced in Heat Exchanger Tube Bundles. Fluid Mechanics-Soviet Research, Vol. 18, No. 4.
49. Kays, W.M., and London, A.L. 1964. Compact Heat Exchangers. 2nd ed., McGraw-Hill, New York.
50. Kelkar, K.M., and Patankar, S.V. 1989. Numerical Prediction of Heat Transfer and Fluid Flow in Rectangular Offset-fin Arrays. Numerical Heat Transfer, Part A, Vol. 15, pp. 149-164.
51. Knight, R.C, and McMullen, B.W. 1937. The Potential of a Screen of Circular Wires between Two Conducting Planes. Philosophical Magazine, Vol. 7, No. 24, pp. 35-47.
52. Kreuekels, S.W., and Kottke, V. 1970. Investigation of the Distribution of Heat Transfer on Fins and Finned Tube Models. Chemical Engineering and Technology, Vol. 42, pp. 355-362.
53. Launder, B.E., and Massey, T.H. 1978. The Numerical Prediction of Viscous Flow and Heat Transfer of Air in an Tube Banks. Transactions of the ASME, Journal of Heat Transfer, Vol. 100, pp. 565-571.

54. Launder, B.E., and Spalding, D.B. 1974. The Numerical Computation of Turbulent Flows. Computer Methods in Applied Mechanics and Engineering, Vol. 3, pp. 269-289.
55. Lamb, H. 1957. Hydrodynamics, 6th ed. Cambridge University Press (also Dover, 1945).
56. Lawden, D.F. 1975. An Introduction to Tensor Calculus and Relativity. Chapman and Hall, London.
57. Le Feuvre, R.F. 1973. Laminar and Turbulent Forced Convection Processes through In-line Tube Banks. Ph.D. Thesis, Imperial College, University of London.
58. Legendre, R., 1967. Écoulement Plan d'un fluide incompressible autour de plusieurs obstacles. Publication No. 116, Offices National d'Études et de Recherches Aérospatiales, Chatillon. (*In French*).
59. Ling, C.B., and Tsai, C.P. 1960. Evaluation at Half Periods of Weierstrass' Elliptic Function with Rectangular Primitive Period-parallelogram. Mathematics of Computation, Vol. 14, pp. 67-70.
60. Ling, C.B., and Tsai, C.P. 1964. Evaluation at Half Periods of Weierstrass' Elliptic Function with Rhombic Primitive Period-parallelogram. Mathematics of Computation, Vol. 18, pp. 433-440
61. Ling, C.B. 1965. Tables of Values of σ_{2s} Relating to Weierstrass' Elliptic Function. Mathematics of Computation, Vol. 19, pp. 123-127.

62. Lohrisch, W. 1929. Forschungsarbeiten auf dem Gebiete des Ingenieurwesens. No. 322, p. 46. (*In German*).
63. London, A.L., and Shah, R.K. 1968. Offset Rectangular Plate-Fin Surfaces - Heat Transfer and Flow Friction Characteristics. Transactions of the ASME, Series A, Vol. 90, July, pp. 218-228.
64. Malin, M.R., Rosten, H.I., Spalding, D.B., and Tatchell, D.G. 1985. Application of PHOENICS to Flow around Ships' Hulls. 2nd International Symposium on Ship Viscous Resistance, Gothenburg.
65. Massey, T.H. 1976. The Prediction of Flow and Heat-transfer in Banks of Tubes in Cross-flow. Ph.D. Thesis, Council for National Academic Awards.
66. McConnell, A.J. 1957. Applications of Tensor Analysis. Dover, New York.
67. McQuiston, F. 1978. Heat, Mass and Momentum Transport Coefficients for Plate-fin-tube Heat Transfer Surfaces with Staggered Tubes. ASHRAE Transactions, Vol. 84, Part 1, pp. 294-308.
68. Melaaen, M.C. 1990. Analysis of Curvilinear Non-orthogonal Coordinates for Numerical Calculation of Fluid Flow in Complex Geometries. Dr. Ing. Thesis, University of Trondheim.
69. Melaaen, M.C. 1992. Calculation of Fluid Flows with Staggered and Nonstaggered Curvilinear Non-Orthogonal Grids - The Theory. Numerical Heat Transfer, Part B, Vol. 21, No. 1, pp. 1-19.

70. Milne-Thomson, L.M. 1955. Theoretical Hydrodynamics. 3rd ed. MacMillan, New York.
71. Neal, S.B.H.C., and Hitchcock, J.A. 1967. Heat Transfer and Gas Flow Processes within a Bank of Close Pitched Plain Tubes in Cross-flow. CERL Note No. RD/L/N9/67.
72. Niggeschmidt, W. 1975. Druckverlust und Wärmeübergang bei fluchtenden, versetzten und teilversetzten querangeströmten Rohrbündeln. Doctoral Dissertation, University of Darmstadt. (*In German*).
73. Omohundro, G.A., Bergelin, O.P., and Colburn, A.P. 1949. Heat-transfer and Fluid Friction during Viscous Flow across Banks of Tubes. Transactions of the ASME, Vol. 71, pp. 27-34.
74. Palen, J.W. (*Editor*) 1986. Heat Exchanger Sourcebook. Hemisphere, New York.
75. Païdoussis, M.P. 1982. A Review of Flow-induced Vibrations in Reactors and Reactor Components. Nuclear Engineering and Design, Vol. 74, pp. 31-60.
76. Païdoussis, M.P., Mavripilis, D., and Price, S.J. 1984. A Potential-flow Theory for the Dynamics of Cylinder Arrays in Cross-flow. Journal of Fluid Mechanics, Vol. 146, 1984, pp. 227-252.
77. Parameswaran, S. 1985. Finite Volume Equations for Fluid Flow Based on Non-Orthogonal Velocity Projections. Ph.D. Thesis, Imperial College, University of London.

78. Patankar, S.V. 1980. Numerical Heat Transfer and Fluid Flow. Hemisphere, New York.
79. Patankar, S.V., and Spalding, D.B. 1972. A Calculation Procedure for Heat, Mass, and Momentum Transfer in Three-dimensional Parabolic Flows. International Journal of Heat and Mass Transfer. Vol. 15, pp. 1787-1806.
80. Patankar, S.V., Liu, C.H., and Sparrow, E.M. 1977. Fully-developed Flow and Heat Transfer in Ducts having Streamwise-periodic Variations of Cross-sectional Area. Journal of Heat Transfer, Vol. 99, pp. 180 -186.
81. Patankar, S.V., and Prakash, C. 1981. An Analysis of the Effect of Plate Thickness on Laminar Flow and Heat Transfer in Interrupted Plate Passages. International Journal of Heat and Mass Transfer, Vol. 24, No. 11, pp. 1801-1810.
82. Peric, M.P. 1985. A Finite Volume Method for the Prediction of Three-dimensional Fluid Flow in Complex Ducts. Ph.D. Thesis, Imperial College, University of London.
83. Pierson, O.L. 1937. Experimental Investigation of the Influence of Tube Arrangement on Convection Heat Transfer and Flow Resistance in Crossflow of Gases over Tube Banks. Transactions of the ASME, Vol. 59, pp. 563-572.
84. Plessel, T. 1989. GAS - Graphics Animation System, Version 2.2, User's Manual. Sterling Software, Palo Alto, California. February. (NASA Contract No. 11555)

85. Poskas, P.S., and Survila, V.J. 1983. Fluctuations of Velocity of Cross Flow of Air in the Space between Tubes in banks. Heat Transfer-Soviet Research, Vol. 15, No. 1, pp. 75-86.
86. Price, S.J., Païdoussis, M.P., Macdonald, R., and Mark, B. 1987. The Flow-induced Vibration of a Single Flexible Cylinder in a Rotated Square Array of Rigid Cylinder with Pitch-to-diameter Ratio of 2.12. Journal of Fluids and Structures, Vol. 1, pp. 359-378.
87. Rhie, C.M., and Chow, W.L. 1983. Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation. AIAA Journal, Vol. 21, No. 11, pp. 1525-1532.
88. Ricci, G., and Levi-Civita, T. 1901. Méthodes de calcul différentiel absolu et leurs applications. Mathematische Annalen, Vol. 54, pp. 125-201. (*In French*)
89. Rich, D.G. 1973. The Effects of Fin Spacing on the Heat Transfer and Friction Performance of Smooth Plate-Fin-Tube Heat Exchangers. ASHRAE Transactions, Vol. 79, Part 2, pp. 137-145.
90. Rich, D.G. 1975. The Effect of the Number of Tube Rows on Heat Transfer Performance of Smooth Plate-fin-tube Heat Exchangers. ASHRAE Transactions, Vol. 81, Part 1, pp. 307-317.
91. Rosten, H.I., and Spalding, D.B. 1987. The PHOENICS Equations. TR/99, Concentration Heat and Momentum Ltd., London.
92. Saboya, F.E.M. 1974. Local and Average Transfer Coefficients in Plate Fin and Tube Exchanger Configurations. Ph.D. Thesis. University of Minnesota.

93. Saboya, F.E.M., and Sparrow, E.M. 1974. Local and Average Transfer Coefficients for One-row Plate Fin and Tube Heat Exchanger Configurations. *Journal of Heat Transfer*, Vol. 96, pp. 265-274.
94. Saboya, F.E.M., and Sparrow, E.M. 1976a. Transfer Characteristics of Two-row Plate Fin and Tube Heat Exchanger Configurations. *International Journal of Heat and Mass Transfer*, Vol. 19, pp. 41-49.
95. Saboya, F.E.M., and Sparrow, E.M. 1976b. Experiments on a Three-row Fin and Tube Heat Exchanger. *Journal of Heat Transfer*, Vol. 98, pp. 26-34.
96. Schlichting, H. 1968. *Boundary-layer Theory*. 6th ed. McGraw Hill. New York. (*Translated by J. Kestin*).
97. Shah, R.K. 1967. Data Reduction Procedures for the Determination of Convective Surface Heat Transfer and Flow Friction Characteristic Steam-to-Air Test Cores. TR No. 64, Department of Mechanical Engineering, Stanford University.
98. Shah, R.K., and London, A.L. 1967. Offset Rectangular Plate-fin Surfaces - Heat Transfer and Flow Friction Characteristics. TR No. 66, Department of Mechanical Engineering, Stanford University.
99. Sokolnikoff, I.S. 1951. *Tensor Analysis, Theory and Applications*. Wiley, New York.
100. Spalding, D.B. 1972. A Novel Finite-difference Formulation for Differential Expressions Involving Both First and Second Derivatives. *International Journal for Numerical Methods in Engineering*. Vol. 4, pp. 551-559.

101. Spalding, D.B. 1980. Mathematical Modelling of Fluid-mechanics, Heat-transfer and Chemical-reaction Processes: A Lecture Course. HTS/80/1, Computational Fluid Dynamics Unit, Imperial College, University of London.
102. Spalding, D.B. 1981a. Methods of Calculating Heat Transfer within the Passages of Heat Exchangers. HTS/81/4, Computational Fluid Dynamics Unit, Imperial College, University of London.
103. Spalding, D.B. 1981b. The Calculation of Heat-exchanger Performance. HTS/81/5, Computational Fluid Dynamics Unit, Imperial College, University of London.
104. Spalding, D.B. 1982. Four Lectures on the PHOENICS Code. CFD/82/5, Computational Fluid Dynamics Unit, Imperial College, University of London.
105. Spalding, D.B. 1984. PHOENICS 1984 - A Multi-dimensional Multi-phase General-purpose Computer Simulator for Fluid Flow, Heat Transfer and Combustion. CFD/84/18, Computational Fluid Dynamics Unit, Imperial College, University of London.
106. Spalding, D.B., Rosten, H.I., Glynn, D.R., Malin, M.R., and Edwards, D.R. 1986. Lecture Panels for the Use of the Body-fitted Coordinates in PHOENICS. TR/126. Concentration Heat and Momentum Ltd., London.
107. Sparrow, E.M., Baliga, B.R., and Patankar, S.V. 1977. Heat Transfer and Fluid Flow Analysis of Interrupted-wall Channels, with Application to Heat Exchangers. Journal of Heat Transfer, Vol. 99, pp. 4-11.

108. Spiegel, M.R. 1959. Theory and Problems of Vector Analysis and an Introduction to Tensor Analysis. Schaum's Outline Series. McGraw-Hill, New York.
109. Stuhmillar, J.H., Chilukuri, R., Masiello, P.J., Chan., R.K.C., Yu, J.H.Y., and Ho., K.H.H. 1988. Prediction of Localized Flow Velocities and Turbulence in a PWR Steam Generator. Project S310-14, Final Report, Electric Power Research Institute, Palo Alto, California. (EPRI NP-5555)
110. Suh, Y.B., Somasundaram, S., and Anand, N.K. 1989. Remarks on the Potential Cross Flow over Tube Banks. Transactions of the ASME, Journal of Applied Mechanics. Vol. 56, pp. 476-479.
111. Taborek, J., Hewitt, G.F., and Afgan, N. 1983. Heat Exchangers: Theory and Practice. Hemisphere, Washington.
112. Tannery, J, and Molk, J. 1893-1902. Elements de la Theory des Fonctions Elliptiques. Vol. 1-4, Gauthier-Villars, Paris. (*In French*).
113. Thom, A., and Apelt, C.J. 1961. Field Computations in Engineering and Physics. Van Nostrand, London.
114. Thoma, H. 1921. Hochleistungskessel. Springer, Berlin. (*In German*).
115. Thompson, J.F., Thames, F.C., and Mastin, C.W. 1974. Automatic Numerical Generation of Body-fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-dimensional Bodies. Journal of Computational Physics, Vol. 15, pp. 299-319.

116. Thompson, J.F., and Warsi, Z.U.A. 1982. Boundary-fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review. *Journal of Computational Physics*, Vol. 47, pp. 1-108.
117. Thompson, J.F., Warsi, Z.U.A., and Mastin, C.W. 1985. *Numerical Grid Generation, Foundations and Applications*. Elsevier, New York and Amsterdam.
118. Walatka, P.P., Buning, P.G., Pierce, L., Elson, P.A., 1990. *PLOT3D User's Manual*. NASA TM 101067, NASA Ames Research Center, Moffett Field, California.
119. Wallis, R. Pendennis, and White, C.M. 1938. Resistance to Flow through Nests of Tubes. *Engineering*, Vol. 146, pp. 605-607.
120. Wallis, R. Pendennis. 1939a. Photographic Study of Fluid Flow between Banks of Tubes. *Engineering*, Vol. 148 , pp. 423-425.
121. Wallis, R. Pendennis, 1939b. A Photographic Study of Fluid Flow between Banks of Tubes. *Proceedings of the IMechE*, Vol. 142, pp. 379-387.
122. Walters, F.M. 1969. Hypersonic Research Engine Project-Phase IIA, Category I Test Report on Fin Heat Transfer and Pressure Drop Testing. Data Item No. 63.02, AiResearch Manufacturing Company, Document No. AP-69-5348.
123. Weaver, D.S., and Abd-Rabbo, A. 1985. A Flow Visualisation Study of a Square Array of Tubes in Water Crossflow. *Transactions of the ASME, Journal of Fluids Engineering*, Vol. 107.
124. Webb, R.L. 1980. Air-side Heat Transfer in Finned Tube Heat Exchangers. *Heat Transfer Engineering*, Vol. 1, No. 3, pp. 33-49.

125. Weiting, A.R. 1975. Empirical Correlations for Heat Transfer and Flow Friction Characteristics of Rectangular Offset-Fin Plate-Fin Heat Exchangers. *Journal of Heat Transfer, Transactions of the ASME, Series C*, Vol. 97, pp. 488-490.
126. Whittaker, E.T., and Watson, G.N. 1958. *A Course of Modern Analysis*. 4th ed. Cambridge University Press, Cambridge.
127. Wong, K.L., and Chen, C.K. The Finite Element Solutions of Laminar Flow and Combined Convection of Air in a Staggered or an In-line Bank. *Wärme-und Stoffübertragung*, Vol. 23, No. 2, pp. 93-101.
128. Yamamoto, T. 1976. Hydrodynamic Forces on Multiple Circular Cylinders. *Journal of the Hydraulic Division, Proceedings of the ASCE*, HY9, pp. 1193-1210.
129. Yih, C.S. 1957. Stream Functions in Three-dimensional Flows. *La Houille Blanche*, No. 3, pp. 445-450.
130. Zukauskas, A.A., Makaravikis, V., and Slanciauskas, A. 1968. Heat Transfer in Banks of Tubes in Cross-flow of Liquids. Mintis, Vilnius. (*In Russian*).
131. Zukauskas, A.A., Ulinskas, R.V. 1978a. Heat Transfer Efficiency of Tube Banks in Crossflow at Critical Reynolds Numbers, *Heat Transfer-Soviet Research*, Vol. 10, No. 5, pp. 9-15.
132. Zukauskas, A.A., and Ulinskas, R.V. 1978b. Some Aspects of the Heat Transfer for Banks of Tubes in Cross-flow for the Low Range of Reynolds Numbers, *Proceedings of the 6th International Heat Transfer Conference, Toronto*, pp. 243-248.

133. Zukauskas, A.A., Ulinskas, R.V., Bubelis, E.S., and Sipavicius, C. 1978c. Heat Transfer from a Tube bank in Crossflow at Low Reynolds Numbers, Heat Transfer-Soviet Research, Vol. 10, No. 5, pp. 29-32.
134. Zukauskas, A.A., Ulinskas, R.V., and Sipavicius, S.J., 1978d. Average Heat Transfer and Pressure Drop in Cross Flow of Viscous Fluids over Tube Banks at Low *Re*. Heat Transfer-Soviet Research, Vol. 10, No. 6, pp. 90-100.
135. Zukauskas, A.A., Zyugzda, I.I., and Survila, V. Yu. 1978e. Effect of Turbulence on Heat Transfer from Cylinders in Crossflow at Critical Reynolds Numbers, Heat Transfer-Soviet Research, Vol. 10, No. 5, pp. 1-8.
136. Zukauskas, A.A. 1987. Heat Transfer from Tubes in Crossflow. Advances in Heat Transfer, Vol. 18., pp. 87-159, Academic Press.
137. Zukauskas, A.A., and Ulinskas, R.V. 1988a. Heat Transfer in Tube Banks in Crossflow. Hemisphere, New York.
138. Zukauskas, A.A., Ulinskas, R.V., and Katinas, V. 1988b. Fluid Dynamics and Flow-induced Vibrations of Tube Banks. Hemisphere, New York. (*English-edition Editor, J. Karni*)

Appendix I. Curvilinear Co-ordinates

A1.1 Introduction

Several texts have been written on the subject of general tensor calculus (McConnell, 1947, Sokolnikoff, 1951, Spiegel, 1959, Aris, 1962, Lawden, 1978). Tensor calculus is a powerful and elegant tool which may readily be used to describe phenomena in n -dimensional Reimannian space. The scope of the analysis, here, is confined to three-dimensional Euclidean spaces so that, with reference to Figure A1.1, a fixed Cartesian co-ordinate system with unit vectors $\vec{e}_{ci}$ ($i = 1, 2, 3$) can always be constructed. Let x_c^i be the Cartesian co-ordinates of some point P . Let it be assumed that there exists a transformation, χ_j^i , such that x^i are the co-ordinates of P in a general curvilinear co-ordinate system,

$$dx^i = \chi_j^i dx_c^j \quad - (A1.1)$$

with

$$\chi_j^i \equiv \frac{\partial x^i}{\partial x_c^j} \quad - (A1.2)$$

The summation convention is assumed, i.e. an index which occurs as both a superscript and a subscript is summed. Also any superscript index occurring in the numerator of a derivative is treated as if it were a subscript.

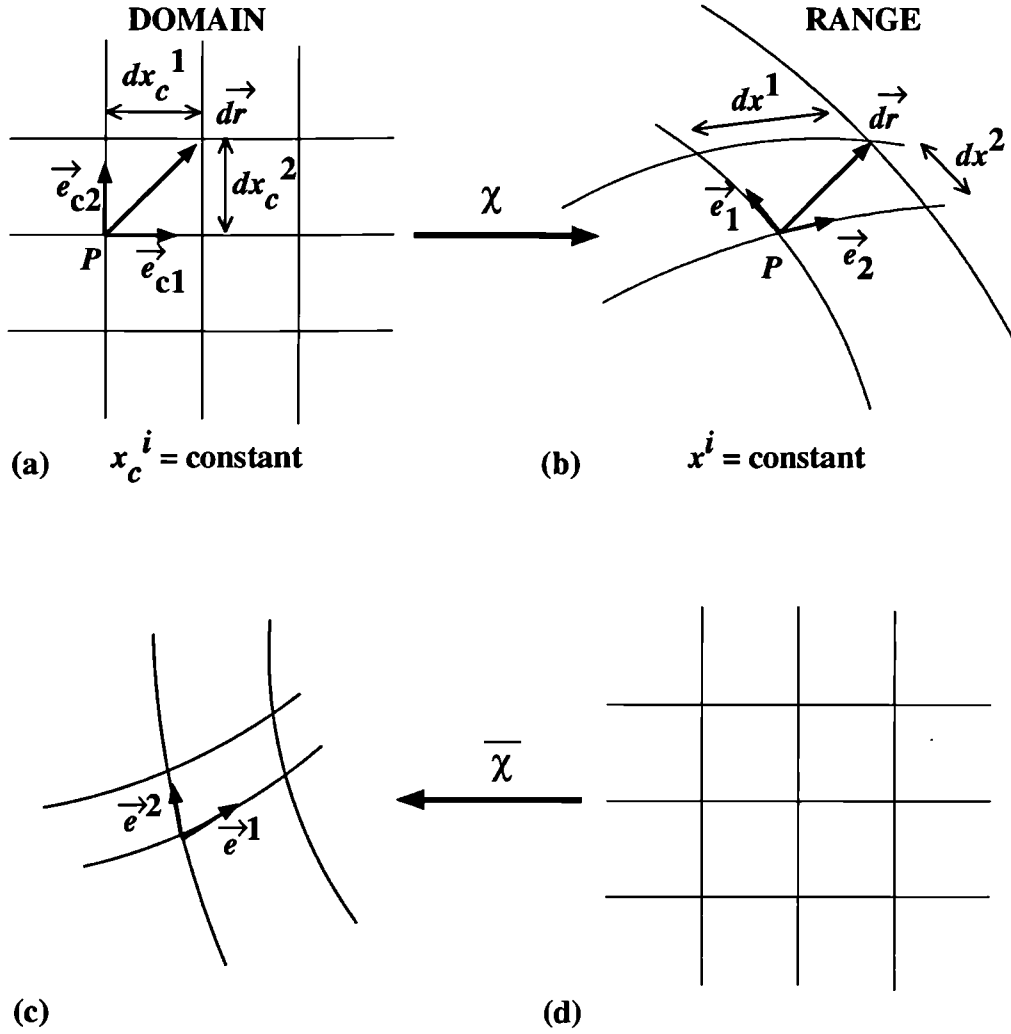


Figure A1.1 Co-ordinate transformation.

If the domain of the mapping defined Eq. A.1.1 is the set of straight lines $x_c^i = \text{constant}$, the range is a set of curves of constant x^i , say. Figure A1.1 is a schematic illustrating this correspondence (shown for convenience in two-dimensions). The inverse transformation is defined by,

$$\bar{\chi}_j^i = \frac{\partial x_c^i}{\partial x^j} \quad - \text{(A1.3)}$$

Consider an arbitrary line element, $\vec{dr}$, referred to the fixed Cartesian co-ordinate system as,

$$\vec{dr} = dx_c^i \vec{e}_{ci} \quad - (A1.4)$$

The same vector may be written as,

$$\vec{dr} = dx^i \vec{e}_i \quad - (A1.5)$$

where,

$$\vec{e}_i \equiv \frac{\partial \vec{r}}{\partial x^i} = \chi_i^j \vec{e}_{cj} \quad - (A1.6)$$

The reader will note that it is not the vector, $\vec{dr}$, but its components which are transformed under Eq. A1.1. The basis vectors defined by Eq. A1.6 are known as covariant basis vectors. They have the physical significance of being tangent vectors to the curves $x^i = \text{constant}$ [Fig. A1.1b]. A second set of basis vectors may be defined according to¹,

$$\vec{e}^j \equiv \vec{\nabla} x^j = \chi_j^i \vec{e}_{ci} \quad (j \text{ summed}) \quad - (A1.7)$$

The basis vectors defined by Eq. A1.7 are known as contravariant basis vectors: If one considers the set of curves in the domain, which map onto straight lines parallel to the Cartesian co-ordinates in the range [Fig. A1.1c and A1.1d], then these curves have $\vec{e}^i$ as tangent vectors. Alternatively, if in terms of the mapping defined by Eq. A1.1 rather

¹ Because in this text the Cartesian co-ordinates are written exclusively as superscripts, and the Cartesian basis vectors as subscripts, it is not always possible to observe the summation convention.

than considering the set of straight lines in the domain [Fig. A1.1a], one considers the co-ordinate planes,

$$\vec{r} \cdot \vec{e}_{ci} = \text{constant} \quad - (A1.8)$$

Then the locus of the range of the mapping are (three) families of surfaces normal to the contravariant basis vectors.

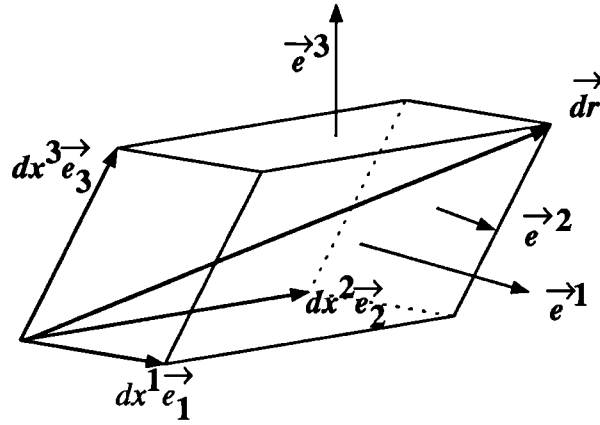


Figure A1.2 Element of volume aligned with the co-ordinate system.

Any vector, $\vec{a}$, may be expressed equivalently in either of two forms,

$$\vec{a} = a_i \vec{e}^i \quad - (A1.9)$$

$$\vec{a} = a^i \vec{e}_i \quad - (A1.10)$$

a_i are known as the covariant components and a^i the contravariant components of $\vec{a}$.

The reader will note that the properties of a vector (as opposed to its components) are

independent of the co-ordinate system, and that this is achieved by writing the vectors as the product of a covariant component and a contravariant basis vector (or vice versa).

The covariant and contravariant basis vectors are dual, so that,

$$\vec{e}^i \cdot \vec{e}_j = \chi_k^i \bar{\chi}_j^k = \delta_j^i \quad - (A1.11)$$

where δ_j^i is the Kronecker delta,

$$\begin{aligned} \delta_j^i &= 1, & i &= j \\ \delta_j^i &= 0, & i &\neq j \end{aligned} \quad - (A1.12)$$

The length of the line element defined by Eq. A1.4 is just,

$$ds^2 = dx_c^i dx_c^i \quad (i \text{ summed}) \quad - (A1.13)$$

The same length referred to the new co-ordinate system is given, in the quadratic form by,

$$ds^2 = g_{ij} dx^i dx^j \quad - (A1.14)$$

where the components of the covariant metric tensor, g_{ij} , are defined by,

$$g_{ij} \equiv \vec{e}_i \cdot \vec{e}_j = \chi_i^k \chi_j^k \quad (k \text{ summed}) \quad - (A1.15)$$

The contravariant (or inverse) metric tensor, g^{ij} , is,

$$g^{ij} \equiv \vec{e}^i \cdot \vec{e}^j \quad - (A1.16)$$

The metric tensors allow for conversion between covariant and contravariant vectors (i.e. raising and lowering of indices);

$$a_i = g_{ij}a^j \quad - (A1.17)$$

$$a^i = g^{ij}a_j \quad - (A1.18)$$

It can be shown that g^{ij} and g_{ij} are related according to,

$$g^{ij} = \frac{c^{ij}}{g}, \quad g_{ij} = g c_{ij} \quad - (A1.19)$$

where c^{ij} are the cofactors of g_{ij} , c_{ij} are the cofactors of g^{ij} , and g is given by,

$$\sqrt{g} = |J| = \sqrt{\det g_{ij}} = \vec{e}_1 \cdot \vec{e}_2 \times \vec{e}_3 = \frac{1}{\vec{e}^1 \cdot \vec{e}^2 \times \vec{e}^3} \quad - (A1.20)$$

J is the Jacobian of the transformation defined by Eq. A1.1,

$$J \equiv \det(\chi_j^i) = \frac{1}{6} \epsilon_{ijk} \epsilon^{lmn} \chi_l^i \chi_m^j \chi_n^k \quad - (A1.21)$$

and ϵ_{ijk} and ϵ^{ijk} are permutation symbols (i.e. ± 1 for $i \neq j \neq k$ depending on the order of the permutation, or zero otherwise).

The volume, dV , of the elementary parallelepiped illustrated in Figure A1.2 is just,

$$dV = \sqrt{g} dx^1 dx^2 dx^3 \quad - (A1.22)$$

The reader will note although the length, ds , is independent of the choice of co-ordinate system, the volume dV , [Fig. A1.1b] is not the same as the volume of the equivalent cubic region bounded by the Cartesian co-ordinates dx_c^i [Fig. A1.1a].

A1.2 Scalar and vector products

The scalar product of two vectors is defined by

$$\vec{a} \cdot \vec{b} = (a_i \vec{e}^i) \cdot (b^j \vec{e}_j) = a_i b^i = g_{ij} a^i b^j = g^{ij} a_i b_j \quad - (A1.23)$$

The magnitude of a vector is thus,

$$|\vec{a}| = \sqrt{g_{ij} a^i a^j} = \sqrt{g^{ij} a_i a_j} \quad - (A1.24)$$

The angle between two vectors is,

$$\cos \theta = \frac{a^i b_i}{\sqrt{a^j a_j b^k b_k}} \quad - (A1.25)$$

The magnitude of the covariant basis vectors is,

$$|e_i| = \sqrt{g_{ii}} \quad (unsummed) \quad - (A1.26)$$

The direction cosines of the angle, θ_{ij} , between tangent vectors, $\vec{e}_i$ and $\vec{e}_j$ is,

$$\alpha_{ij} = \frac{g_{ij}}{\sqrt{g_{ii} g_{jj}}} \quad (unsummed) \quad - (A1.27)$$

where,

$$\alpha_{ij} = \cos \theta_{ij} \quad - (A1.28)$$

The components of the inverse matrix, $\bar{\alpha}^{ij}$, are obtained using the usual laws of linear algebra, namely,

$$\bar{\alpha}^{ij} = \frac{1}{\det \alpha_{ij}} (A^{ij})^T \quad - (A1.29)$$

where $(A^{ij})^T$ is the transpose of the adjugate of α_{ij} . The direction cosines between the contravariant basis vectors, α^{ij} may be obtained simply by raising the indexes on i and j . The direction cosines of the angle between a covariant basis vector, $\vec{e}_i$, and a contravariant basis vector, $\vec{e}^j$, are given by,

$$\alpha_j^i = \frac{g^{ij}}{\sqrt{g_{ik}g_{kj}}} \quad (unsummed) \quad - (A1.30)$$

α_j^i is a diagonal matrix, i.e.

$$\alpha_j^i = 0 \quad i \neq j \quad - (A1.31)$$

The inverses are just,

$$\bar{\alpha}_i^j = \frac{1}{\alpha_i^j} = \sqrt{g^{jj}} \sqrt{g_{ii}} \quad (unsummed) \quad - (A1.32)$$

Consider then vector product of two vectors,

$$\vec{a} = \vec{b} \times \vec{c} \quad - (A1.33)$$

Using the fact that the covariant and contravariant basis vectors are dual, for example,

$$\vec{e}^1 = \frac{\vec{e}_2 \times \vec{e}_3}{\vec{e}_1 \cdot \vec{e}_2 \times \vec{e}_3} = \frac{\vec{e}_2 \times \vec{e}_3}{\sqrt{g}} \quad - (A1.34)$$

$$\vec{e}_1 = \frac{\vec{e}^2 \times \vec{e}^3}{\vec{e}^1 \cdot \vec{e}^2 \times \vec{e}^3} = \sqrt{g} \vec{e}^2 \times \vec{e}^3 \quad - (A1.35)$$

The vector product of two vectors may be written,

$$a_i \vec{e}^i = \sqrt{g} \epsilon_{ijk} b^j c^k \vec{e}^i = \sqrt{g} \begin{vmatrix} \vec{e}^1 & \vec{e}^2 & \vec{e}^3 \\ b^1 & b^2 & b^3 \\ c^1 & c^2 & c^3 \end{vmatrix} \quad - (A1.36)$$

$$a^i \vec{e}_i = \frac{1}{\sqrt{g}} \epsilon^{ijk} b_j c_k \vec{e}_i = \frac{1}{\sqrt{g}} \begin{vmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} \quad - (A1.37)$$

A1.3 Vector resolutives and components

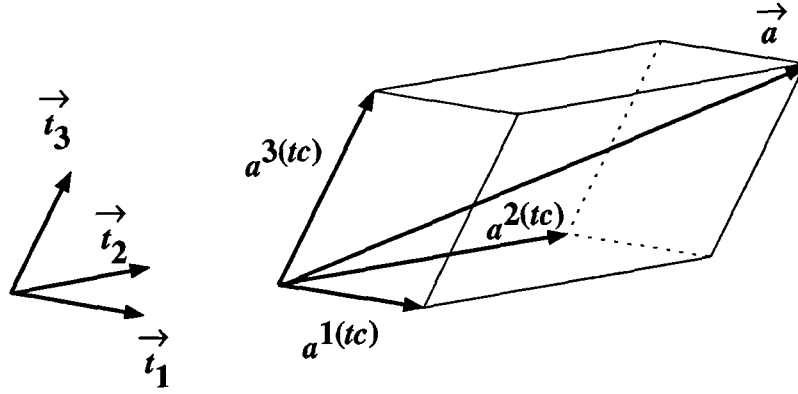


Figure A1.3 Components of a vector.

A set of unit basis vectors, tangent to the co-ordinate lines is,

$$\vec{t}_i = \frac{\vec{e}_i}{\sqrt{g_{ii}}} \quad (\text{unsummed}) \quad - \text{ (A1.38)}$$

A set of unit basis vectors, normal to the co-ordinate surfaces is;

$$\vec{n}^i = \frac{\vec{e}^i}{\sqrt{g^{ii}}} \quad (\text{unsummed}) \quad - \text{ (A1.39)}$$

It is convenient to define the components of a vector in terms of these unit bases as follows,

$$\vec{a} = a^{i(tc)} \vec{t}_i \quad - (A1.40)$$

$$\vec{a} = a_{i(nc)} \vec{n}^i \quad - (A1.41)$$

where the parenthetic superscripts and subscripts *tc* and *nc* denote the tangential components and normal components respectively. The tangential and normal components of a vector may be related to the contravariant and covariant components (Ricci and Levi-Civita, 1901) according to,

$$a^{i(tc)} = \sqrt{g_{ii}} a^i \quad (unsummed) \quad - (A1.42)$$

$$a_{i(nc)} = \sqrt{g^{ii}} a_i \quad (unsummed) \quad - (A1.43)$$

A vector may thus be thought of, in the usual sense of Lamé, as a directed line segment with three components referred to a basis of unit tangents or normals to the three co-ordinate directions [Fig.A1.3]. Let it be noted that since dx^i are contravariant displacement-vector components, the tangential components of an element of displacement, $dx^{i(tc)}$, are,

$$dx^{i(tc)} = \sqrt{g_{ii}} dx^i \quad (unsummed) \quad - (A1.44)$$

The transformation defined by Eq. A1.1 and hence the dimensions of dx^i are quite arbitrary (e.g. r [L] and θ [] in polar co-ordinates). Conversely the dimensions of the displacements defined by Eq. A1.44 will always be [L], so that those of derived quantities

(e.g. velocity) will also be consistent. However physical quantities do not transform as tensors according to the laws of covariance and contravariance.

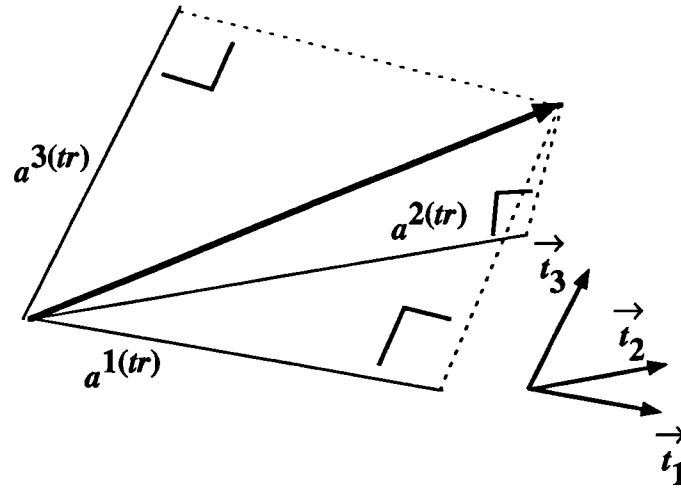


Figure A1.4 Resolutes (projections) of a vector.

The vector components are not the same as the vector resolutes, obtained by projecting the three components of a vector in the direction of the unit basis vectors.

$$a_{i(tr)} = \vec{a} \cdot \vec{t}_i \quad - (A1.45)$$

$$a^i(nr) = \vec{a} \cdot \vec{n}^i \quad - (A1.46)$$

The quantities defined by Eqs. A1.45 and A1.46 will be referred to as tangential and normal resolutes, indicated by the subscript *tr* and superscript *nr*, respectively,

$$a_{i(tr)} = \frac{a_i}{\sqrt{g_{ii}}} \quad (\text{unsummed}) \quad - (A1.47)$$

$$a^{i(nr)} = \frac{a^i}{\sqrt{g^{ii}}} \quad (\text{unsummed}) \quad - (A1.48)$$

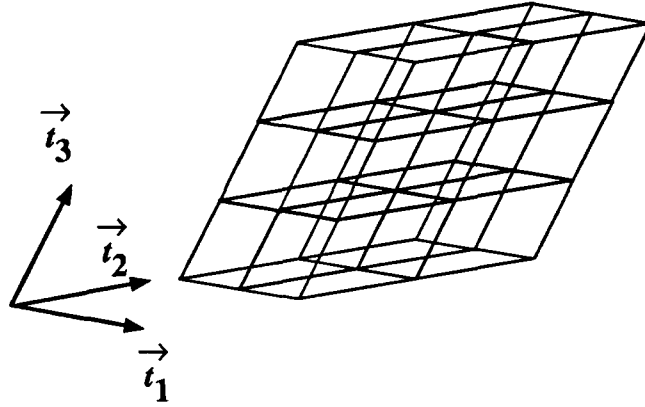


Figure A1.5 Oblique rectilinear co-ordinate system.

In a non-orthogonal co-ordinate system, it is important to distinguish between the four different vector resolutes and components. An oblique rectangular co-ordinate system [Fig. A1.5] is obtained by means of an affine transformation of a Cartesian grid i.e. it is non-orthogonal, but the basis vectors are invariant.

An orthogonal co-ordinate system is one in which the normal and tangential directions are parallel. Under these circumstances the off-diagonal terms of the metric tensor are identically zero, and the normal and tangential resolutes and components are the same. Thus an orthogonal grid may be obtained as a rotation and a stretch at every point in

space. Let the direction cosines between the curvilinear and Cartesian co-ordinates be denoted by, l^{ij} ,

$$l^{ij} = \frac{1}{\sqrt{g_{ii}}} \frac{\partial x_c^i}{\partial x^j} = \frac{1}{\sqrt{g^{jj}}} \frac{\partial x^j}{\partial x_c^i} \quad (\text{unsummed}) \quad - (A1.49)$$

where,

$$\det(l^{ij}) = \pm 1 \quad - (A1.50)$$

The elements of l^{ij} are equal its co-factors.

$$l^{ij} = \pm \frac{1}{2} \epsilon_{ijk} \epsilon_{lmn} l^{jm} l^{kn} \quad - (A1.51)$$

Using this fact, it can be shown that,

$$\frac{\partial}{\partial x_c^k} \left(g^{jk} \sqrt{g} \frac{\partial x_c^i}{\partial x^j} \right) = 0 \quad - (A1.52)$$

Equation A1.52 is just Laplace's equation based on the covariant operator (see below).

In a non-orthogonal co-ordinate system, the tangential resolute and components are related according to,

$$a_{i(tr)} = \alpha_{ij} a_j^{(tc)} \quad - (A1.53)$$

$$a^{i(tc)} = \bar{\alpha}^{ij} a_j^{(tr)} \quad - (A1.54)$$

where α_{ij} are the direction cosines between the co-ordinate directions and $\overline{\alpha}^{ij}$ are their inverses [Eq. A1.29].

Similar linear relations also exist for the normal resolutes and components.

$$a^{i(nr)} = \alpha^{ij} a_{j(nc)} \quad - (A1.55)$$

$$a_{i(nc)} = \overline{\alpha}_{ij} a^{j(nr)} \quad - (A1.56)$$

The tangential vector components may also be projected in the normal directions, and vice versa,

$$a_{i(tr)} = \alpha_i^j a_{j(nc)} \quad - (A1.57)$$

$$a_{i(nc)} = \overline{\alpha}_i^j a_{j(tr)} \quad - (A1.58)$$

$$a^{i(nr)} = \alpha_j^i a^{j(tc)} \quad - (A1.59)$$

$$a^{i(tc)} = \overline{\alpha}_j^i a^{j(nr)} \quad - (A1.60)$$

The scalar product may now be written in terms of the product of a resolute and a component as

$$\vec{a} \cdot \vec{b} = a^{i(tc)} b_{i(tr)} = a_{i(nc)} b^{i(nr)} = \alpha_{ij} a^{i(tc)} b^{j(tc)} \quad - (A1.61)$$

The component form of the vector product is

$$a_i(nc) \vec{n}^i = \begin{vmatrix} \sin \theta_{23} \vec{n}^1 & \sin \theta_{13} \vec{n}^2 & \sin \theta_{12} \vec{n}^3 \\ b^1 & b^2 & b^3 \\ c^1 & c^2 & c^3 \end{vmatrix} \quad - (A1.62)$$

while the resolute form is,

$$a^1(nr) = \frac{1}{\sin \theta_{23}} \begin{vmatrix} b_2(tr) & b_3(tr) \\ c_2(tr) & c_3(tr) \end{vmatrix}, \quad a^2(nr) = -\frac{1}{\sin \theta_{13}} \begin{vmatrix} b_1(tr) & b_3(tr) \\ c_1(tr) & c_3(tr) \end{vmatrix}, \quad a^3(nr) = \frac{1}{\sin \theta_{12}} \begin{vmatrix} b_1(tr) & b_2(tr) \\ c_1(tr) & c_2(tr) \end{vmatrix} \quad - (A1.63)$$

A1.4 Geometrical Considerations

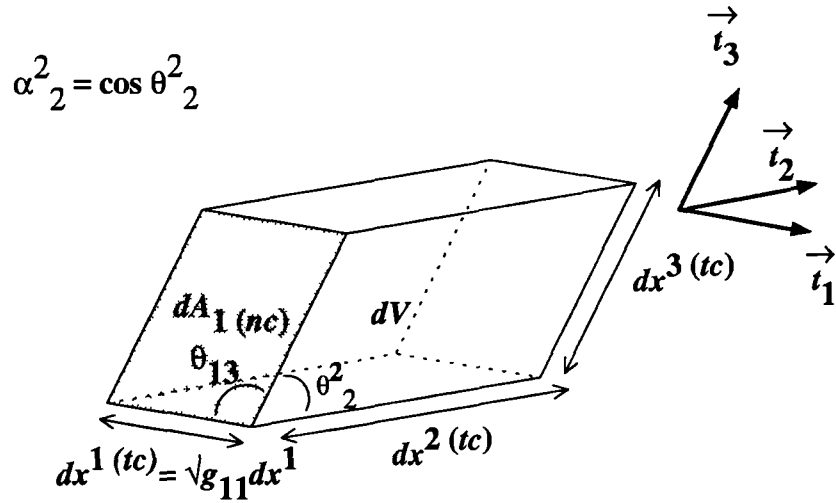


Figure A1.6 Dimensions of an element of volume.

Consider then an elementary-volume in the form of a parallelepiped with sides of length, $dx^{i(tc)}$, as shown in Fig. A1.6. From Eqs. A1.36 and A1.37, it is apparent that,

$$dA_1 = \sqrt{g}dx^2dx^3, \quad dA^1 = \frac{1}{\sqrt{g}}\left(dx^2dx^3\sqrt{g_{22}}\sqrt{g_{33}} - dx^2dx^3g_{23}\right) \quad - (A1.64)$$

So that, as expected, the magnitude of this element of area is given by,

$$dA_{1(nc)} = \sqrt{dA^1dA_1} = dx^{2(tc)}dx^{3(tc)}\sin\theta_{23} \quad - (A1.65)$$

Thus the volume of the element, dV , may be written as,

$$dV = \sqrt{g}dx^1dx^2dx^3 = dA_id x^i = dA_{i(nc)}dx^{i(tc)}\alpha_i^i \quad (unsummed) \quad - (A1.66)$$

A1.5 Covariant Derivative

The covariant derivative of a scalar is just the ordinary derivative,

$$\phi_{,i} \equiv \frac{\partial\phi}{\partial x^i} \quad - (A1.67)$$

For a vector however, the partial derivative contains a contribution due to the change in magnitude and direction of the basis. Covariant differentiation (denoted by a comma) is defined in such a way as to remove this dependence, e.g. for the contravariant components of a vector,

$$u^i{}_{,j}\vec{e}_i \equiv \frac{\partial(u^i\vec{e}_i)}{\partial x^j} = \frac{\partial u^i}{\partial x^j}\vec{e}_i + u^i\frac{\partial\vec{e}_i}{\partial x^j} \quad - (A1.68)$$

The basis vectors may be eliminated from the identity, using Eq. A1.11, as follows,

$$u^i_{,j} = \frac{\partial u^i}{\partial x^j} + u^k \frac{\partial \vec{e}_k}{\partial x^j} \cdot \vec{e}^i \quad - (A1.69)$$

The covariant derivative of a covariant vector is,

$$u_{i,j} = \frac{\partial u_i}{\partial x^j} + u_k \frac{\partial \vec{e}^k}{\partial x^j} \cdot \vec{e}_i \quad - (A1.70)$$

Covariant differentiation, thus defined, may be considered to represent spatial differentiation with respect to a fixed-oblique-rectilinear co-ordinate system [Fig. A1.5] locally parallel to the curvilinear co-ordinate system. Thus, covariant derivatives of vector quantities are independent of bending or stretching of the co-ordinate lines in space. An important corollary resulting from this definition of covariant differentiation, known as Ricci's lemma is that the covariant derivative of either metric tensor is zero. The operations of covariant differentiation and raising (or lowering) indices are therefore commutative. It is common to introduce Christoffels's symbols (of the second kind) such that,

$$\{^k_{ij}\} \vec{e}_k = \frac{\partial \vec{e}_i}{\partial x^j} \quad - (A1.71)$$

Although Cristoffels' symbols play a prominent role in tensor calculus, they are not used here. The reader will note from Eq. A1.6 that,

$$\frac{\partial \vec{e}_i}{\partial x^j} = \frac{\partial \vec{e}_j}{\partial x^i} \quad - (A1.72)$$

It can be shown [using Eqs. A1.20 and A1.69] that the divergence of a vector $\vec{u}$ is given by,

$$\vec{\nabla} \cdot \vec{u} = u^i_{,i} = \frac{1}{\sqrt{g}} \frac{\partial}{\partial x^i} (\sqrt{g} u^i) \quad - (A1.73)$$

The Laplacian of a scalar, ϕ , is just,

$$\nabla^2 \phi = g^{ij} \phi_{,ij} = \frac{1}{\sqrt{g}} \frac{\partial}{\partial x^j} \left(\sqrt{g} g^{ij} \frac{\partial \phi}{\partial x^i} \right) \quad - (A1.74)$$

It is worth digressing briefly, to consider how such terms arise in transport equations. Consider for example,

$$\frac{\partial}{\partial x_c^i} \int_V \rho u_c^i \phi dV_c = \frac{\partial}{\partial x^j} \int_V x_c^j x_i^k \rho u_k \phi \sqrt{g} dx^1 dx^2 dx^3 = \frac{\partial}{\partial x^j} \int_V \rho u^i \phi \sqrt{g} dx^1 dx^2 dx^3 \quad - (A1.75)$$

Since

$$\frac{\partial x_c^j}{\partial x^i} = \frac{\partial}{\partial x_c^i} \left(\frac{\partial x^j}{\partial x_c^i} \right) = 0 \quad - (A1.76)$$

Forms such Eq. A1.75 are useful when seeking to obtain transport equations in terms of the partial derivative, from first principles, as a result of a co-ordinate transformation. It is seen that the Jacobian arises naturally to account for the change of volume due to the transformation. The significance of the right and left sides of Eq. A1.75 being the difference between integrating over a series of elementary parallelepipeds, instead of cubic volumes. The actual volume, V , is the same in both cases. (The choice of V , is arbitrary, and need not correspond with the either the curvilinear co-ordinate or the

Cartesian co-ordinate directions.) The diffusion term is easily derived in a similar fashion.

A1.6 Tangential Derivative

By analogy to the covariant derivative, the symbol δ is used below to denote the operation of differentiation with respect to a fixed-tangential direction. The tangential derivative of a scalar is the same as the ordinary derivative,

$$u_{i(tr)} = \frac{\partial \phi}{\partial x^{i(tc)}} = \frac{\partial \phi}{\partial x^{i(tc)}} \quad - (A1.77)$$

where, as might be expected, the tangential-gradient operator generates vector resolutes in the three tangential directions. The normal resolutes are easily computed,

$$u^{i(nr)} = \alpha_j^i \bar{\alpha}^{jk} \frac{\partial \phi}{\partial x^{k(tc)}} \quad - (A1.78)$$

The tangential derivative of a vector may be defined as,

$$\frac{\partial u^{i(tc)}}{\partial x^{j(tc)}} \vec{t}_i \equiv \frac{\partial}{\partial x^{j(tc)}} \left(u^{i(tc)} \vec{t}_i \right) = \frac{\partial u^{i(tc)}}{\partial x^{j(tc)}} \vec{t}_i + u^{i(tc)} \frac{\partial \vec{t}_i}{\partial x^{j(tc)}} \quad - (A1.79)$$

For convenience put,

$$s_j^{i(tc)} \equiv \frac{\partial u^{i(tc)}}{\partial x^{j(tc)}} \quad - (A1.80)$$

The basis vectors are then eliminated: For tangential components,

$$s_{j(tr)}^{i(tc)} = \frac{\partial u^{i(tc)}}{\partial x^{j(tc)}} = \frac{\partial u^{i(tc)}}{\partial x^{j(tc)}} + \bar{\alpha}^{il} u^{k(tc)} \frac{\partial \vec{t}_k}{\partial x^{j(tc)}} \cdot \vec{t}_l \quad - (A1.81)$$

For normal components,

$$s_{ij(nc)(tr)} = \frac{\partial u_{i(nc)}}{\partial x^{j(tc)}} = \frac{\partial u_{i(nc)}}{\partial x^{j(tc)}} + \bar{\alpha}_{il} u_{k(nc)} \frac{\partial \vec{n}^k}{\partial x^{j(tc)}} \cdot \vec{n}^l \quad - (A1.82)$$

For tangential resolute,

$$s_{ij(tr)(tr)} = \frac{\partial u_{i(tr)}}{\partial x^{j(tc)}} = \frac{\partial u_{i(tr)}}{\partial x^{j(tc)}} + u^{k(tc)} \frac{\partial \vec{t}_k}{\partial x^{j(tc)}} \cdot \vec{t}_i \quad - (A1.83)$$

For normal resolute,

$$s_{j(tr)}^{i(nr)} = \frac{\partial u^{i(nr)}}{\partial x^{j(tc)}} = \frac{\partial u^{i(nr)}}{\partial x^{j(tc)}} + u_{k(nc)} \frac{\partial \vec{n}^k}{\partial x^{j(tc)}} \cdot \vec{n}^i \quad - (A1.84)$$

The meaning of the terms on the right-hand side of Eq. A1.81 to A1.84 is clear. The first term represents the change between $x^{j(tc)}$ and $x^{j(tc)+dx^{j(tc)}}$ of the i^{th} component or resolute of vector $\vec{u}$. The second term represents the difference between the projection in the component direction i of a constant $\vec{u}$ at $x^{j(tc)}$ and $x^{j(tc)+dx^{j(tc)}}$ due to changes in the co-ordinate directions. Since these are projections, in the case of Eq. A.181 and A1.82, they must be converted to components. Let it be noted that,

$$\frac{\partial u^{i(tc)}}{\partial x^{j(tc)}} \vec{t}_i = \frac{\partial u_{i(nc)}}{\partial x^{j(tc)}} \vec{n}^i \quad - (A1.85)$$

can only, in general, be true if,

can only, in general, be true if,

$$u^{k(tc)} \frac{\partial \vec{t}_k}{\partial x^{j(tc)}} = u^{k(nc)} \frac{\partial \vec{n}^k}{\partial x^{j(tc)}} \quad - (A1.86)$$

Thus Eq. A1.82, for example, may also be written in the form,

$$\frac{\partial u_{i(nc)}}{\partial x^{j(tc)}} = \frac{\partial u_{i(nc)}}{\partial x^{j(tc)}} + \bar{\alpha}_i u^{k(tc)} \frac{\partial \vec{t}_k}{\partial x^{j(tc)}} \cdot \vec{n}_l \quad - (A1.87)$$

The expression for $\text{div } \vec{u}$ in terms of the tangential derivative is,

$$\vec{\nabla} \cdot \vec{u} = \frac{\partial u^{i(tc)}}{\partial x^{i(tc)}} = \frac{\partial}{\partial x^{i(tc)}} \left(\bar{\alpha}_j^i u^{j(nr)} \right) = \bar{\alpha}_j^i \frac{\partial u^{j(nr)}}{\partial x^{i(tc)}} \quad - (A1.88)$$

where the angles and the inverses commute with the action of tangential differentiation.

An important expression for $\text{div } \vec{u}$ in terms of the ordinary partial derivative will now be derived: $\text{div } \vec{u}$ may be conveniently written in operator notation as,

$$\nabla \cdot \vec{u} = \left(\bar{\alpha}_j^i \vec{n}^j \frac{\partial}{\partial x^{i(tc)}} \right) \cdot u^{k(tc)} \vec{t}_k = \frac{\partial u^{k(tc)}}{\partial x^{k(tc)}} + u^{k(tc)} \bar{\alpha}_j^i \vec{n}^j \cdot \frac{\partial \vec{t}_k}{\partial x^{i(tc)}} \quad - (A1.89)$$

where interchanging the indices on the last term is justified because $\vec{t}_i = \partial \vec{r} / \partial x^{i(tc)}$.

Now,

$$\bar{\alpha}_1 \vec{n}^1 = \frac{1}{\vec{n}^1 \cdot \vec{t}_1} \vec{n}^1 = \frac{\vec{t}_2 \times \vec{t}_3}{\vec{t}_1 \cdot \vec{t}_2 \times \vec{t}_3} \quad - (A1.90)$$

Similar relations may also be derived for $\bar{\alpha}_2 \vec{n}^2$ and $\bar{\alpha}_3 \vec{n}^3$. Hence it can be shown that,

$$\bar{\alpha}_j \vec{n}^j \cdot \frac{\partial \vec{t}_i}{\partial x^{k(tc)}} = \frac{1}{\vec{t}_1 \cdot \vec{t}_2 \times \vec{t}_3} \frac{\partial}{\partial x^{k(tc)}} (\vec{t}_1 \cdot \vec{t}_2 \times \vec{t}_3) = \frac{1}{\sqrt{\alpha}} \frac{\partial \sqrt{\alpha}}{\partial x^{k(tc)}} \quad - (A1.91)$$

where $\alpha = \det \alpha_{ij}$. Thus $\text{div } \vec{u}$, may be written as,

$$\text{div } \vec{u} = \frac{\partial u^{i(tc)}}{\partial x^{i(tc)}} = \frac{1}{\sqrt{\alpha}} \frac{\partial}{\partial x^{i(tc)}} (\sqrt{\alpha} u^{i(tc)}) = \frac{1}{\sqrt{\alpha}} \frac{\partial}{\partial x^{i(tc)}} (\bar{\alpha}_j^i \sqrt{\alpha} u^{j(nr)}) \quad - (A1.92)$$

which is the required form. In a similar fashion, the Laplacian of a scalar, $\text{div grad } \phi$, may be written as,

$$\nabla^2 \phi = \frac{1}{\sqrt{\alpha}} \frac{\partial}{\partial x^{i(tc)}} \left(\bar{\alpha}^{ij} \sqrt{\alpha} \frac{\partial \phi}{\partial x^{j(tc)}} \right) \quad - (A1.93)$$

where the inverse cosines convert the tangential resolutives of $\text{grad } \phi$ to tangential components [see Eq. A1.78]. Note that the cosines in Eqs. A1.92 and A1.93 do not commute with the partial derivative.

A1.7 Integration of scalar and vector quantities

The divergence of a vector may be integrated to yield,

$$\int_V \vec{\nabla} \cdot \vec{u} dV = \int_A u^{i(nr)} dA_{i(nc)} \quad - (A1.94)$$

where use has been made of Eq. A1.66. Equation A1.93 may be integrated to yield,

$$\int_V \nabla^2 \phi dV = \int_A (\text{div } \phi)^{(nr)} dA_{i(nc)} = \int_A \alpha_j^{ijk} \frac{\partial \phi}{\partial x^{k(nc)}} dA_{i(nc)} \quad - (A1.95)$$

where $\text{div } \phi$ is based on the tangential derivative.

When computing the flux of a scalar quantity, by integrating $\text{div } \vec{u}$ or $\text{div grad } \phi$, over a finite-volume aligned with the curvilinear co-ordinate surfaces, the quantity $\text{div } \vec{u}$ or $\text{div grad } \phi$ need only be computed normal to the control-surface. If, however, the control-volume is not aligned with the co-ordinates system, Eqs. A1.94 and A1.95, are still valid, except that normal co-ordinates $dA_{i(nc)}$ are the components of a surface which is not itself a co-ordinate surface.

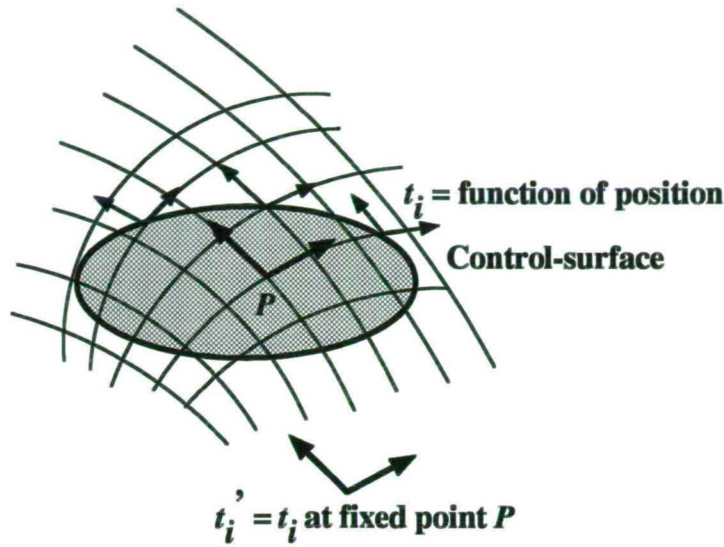


Figure A1.8 Fixed basis vectors.

The above procedure can also be applied to the flux of a vector quantity, through a finite-volume. However one problem arises; consider for example,

$$\vec{F} = \int_V \vec{\nabla} \cdot \vec{s} dV = \int_A \left(s^{ij(tc)(nr)} \vec{t}_i \right) dA_{j(nc)} \quad - (A1.96)$$

Now $\vec{t}_i$ is a function only of position $\vec{r}$, but $\vec{F}$ is a function of a region V . Therefore it is necessary to adopt a set of basis vectors, independent of position, in order that $\vec{F}$ may be written in either component or resolute form.

Since the choice of fixed reference vectors is arbitrary, let the fixed basis vectors, $\vec{t}_i'$, be chosen as being parallel to $\vec{t}_i$ at some interior point in V .

$$\vec{F} = F^{i'} \vec{t}_{i'} \quad - (A1.97)$$

where primes are used to refer quantities to the fixed tangential directions $\vec{t}_i$. Then,

$$F_{i'(tr)} = \int_A s^{kj(tc)(nr)} \vec{t}_k \cdot t_{i'} dA_{j(nc)} \quad - (A1.98)$$

or,

$$F_{i'(tr)} = \int_A s_{i'}^{j(tc)} dA_{j(nc)} \quad - (A1.99)$$

where,

$$s_{i'}^{j(tc)} = s^{kj(tc)(nr)} \vec{t}_k \cdot \vec{t}_{i'} \quad - (A1.100)$$

The problem does not arise when considering the flux of a scalar, i.e. no reference basis is required.

Appendix II. Elliptic Functions

Weierstrass' functions σ , ζ , and $\wp$ are defined by,

$$\log \sigma(z) \equiv \log z + \sum'_{m,n} \left[\log \left(1 - \frac{z}{\Omega_{m,n}} \right) + \frac{z}{\Omega_{m,n}} + \frac{z^2}{2\Omega_{m,n}^2} \right] \quad - (A2.1)$$

$$\zeta(z) \equiv \frac{1}{z} + \sum'_{m,n} \left(\frac{1}{z - \Omega_{m,n}} + \frac{1}{\Omega_{m,n}} + \frac{z}{\Omega_{m,n}^2} \right) \quad - (A2.2)$$

$$\wp(z) \equiv \frac{1}{z^2} + \sum'_{m,n} \left(\frac{1}{(z - \Omega_{m,n})^2} - \frac{1}{\Omega_{m,n}^2} \right) \quad - (A2.3)$$

where,

$$\Omega_{m,n} \equiv 2m\omega_1 + 2n\omega_2 \quad - (A2.4)$$

and ω_1 and ω_2 are complex numbers, and m and n are integers. The primed summation sign denotes the double summation over both m and n ,

$$\sum'_{m,n} \equiv \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \quad - (A2.5)$$

but specifically excluding the value $m = n = 0$. $\wp(z)$ is a doubly-periodic function, with periods $2\omega_1, 2\omega_2$,

$$\wp(z + 2\omega_i) = \wp(z), \quad i = 1, 2 \quad - (A2.6)$$

The notation $\wp(z|\omega_1, \omega_2)$ is used where it is necessary to highlight the periods of $\wp$. Functions which are doubly periodic and analytic, i.e. possess no singularities other than poles, are termed elliptic functions. $\wp$ and all its derivatives are elliptic functions. The function ζ and $\log \sigma$ are not strictly elliptic (though often referred to as such), but are quasi-periodic,

$$\sigma(z + 2\omega_i) = -e^{2\eta_i(z+\omega_i)}\sigma(z), \quad i = 1, 2 \quad - (A2.7)$$

$$\zeta(z + 2\omega_i) = \zeta(z) + 2\eta_i, \quad i = 1, 2 \quad - (A2.8)$$

where η_1 and η_2 are complex constants.

Elliptic functions of a complex variable may be considered analagous to the more common circular functions of a real variable. The functions $\log \sigma$, and all even derivatives of $\wp$ are even functions of z , while the functions ζ , $\wp'$ and all odd derivatives, of $\wp$ are odd functions. The properties of Weirstrass' functions are well documented; see for example the books by Tannery and Molk (1893), Hancock (1958), and Whittaker and Watson (1958).

The function, W_0 , defined in Chapter 4 may then be written in terms of Weirstrass' functions as,

$$W_0 = \log \sigma(z) - \sum'_{m,n} \frac{z^2}{2\omega_{m,n}^2} \quad - (A2.9)$$

The derivatives of W_0 may be written in terms of the Weirstrass' functions as;

$$\frac{dW_0}{dz} = \zeta(z) - \sum'_{m,n} \frac{z}{\Omega_{m,n}^2} \quad - (A2.10)$$

$$\frac{d^2W_0}{dz^2} = -\wp(z) - \sum'_{m,n} \frac{1}{\Omega_{m,n}^2} \quad - (A2.11)$$

$$\frac{d^sW_0}{dz^s} = -\frac{d^{(s-2)}}{dz^{(s-2)}} \wp(z), \quad s \geq 3 \quad - (A2.12)$$

The potential functions defined by Eq. 4.21 may be therefore be written as,

$$W_1(z^*) = \zeta(z^* | \omega_1^*, \omega_2^*) - \frac{z^*}{\Omega_{m,n}^{*2}} \quad - (A2.13)$$

$$W_{2s+1}(z^*) = \frac{1}{(2s)!} \wp^{(2s-1)}(z^* | \omega_1^*, \omega_2^*) \quad - (A2.14)$$

where,

$$z^* \equiv \frac{z}{s_2} \quad - (A2.15)$$

$$\omega_i^* = \frac{\omega_i}{s_2}, \quad i = 1, 2 \quad - (A2.16)$$

$$\Omega_{m,n}^* = \frac{\Omega_{m,n}}{s_2} \quad - (A2.17)$$

The double summations in Eq. A2.2 may be reduced to a single one by expanding $1/(z-\Omega_{m,n})$ in a geometric series,

$$\zeta(z) = \frac{1}{z} - \sum_{n=2}^{\infty} c_n z^{2n-1} \quad - (A2.18)$$

where,

$$c_k = \sum' \frac{1}{\Omega_{m,n}^{2k}} \quad - (A2.19)$$

The c_k were computed as follows. For rectangular primitive-period parallelogram i.e. (in-line banks) ω_1 is real and ω_2 is imaginary,

$$\omega_1 = x = \frac{s_2}{2} \quad - (A2.20)$$

$$\omega_2 = iy = i\frac{s_1}{2} \quad - (A2.21)$$

values of c_1 , c_2 , and c_3 were obtained using the following relationships,

$$c_1 = \left(\frac{\pi}{2x}\right)^2 \left(\frac{1}{3} - 2 \sum_{n=1}^{\infty} \frac{1}{\sinh^2(n\pi\lambda)}\right) \quad - (A2.22)$$

$$c_2 = \left(\frac{\pi}{2x}\right)^4 \left(\frac{1}{45} + \frac{2}{3} \sum_{n=1}^{\infty} \frac{2}{\sinh^2(n\pi\lambda)} + \frac{3}{\sinh^4(n\pi\lambda)}\right) \quad - (A2.23)$$

$$c_3 = \left(\frac{\pi}{2x}\right)^6 \left(\frac{2}{945} + \frac{2}{15} \sum_{n=1}^{\infty} \frac{2}{\sinh^2(n\pi\lambda)} + \frac{15}{\sinh^4(n\pi\lambda)} + \frac{15}{\sinh^6(n\pi\lambda)}\right) \quad - (A2.24)$$

where,

$$\lambda = \frac{y}{x} \quad - (A2.25)$$

For proof of the above, the reader is referred to Ling and Tsai (1960) which is essentially the same methodology as in the earlier paper of Knight and McMullen (1936).

For rhombic primitive-period parallelogram (staggered banks), ω_1 and ω_2 are complex conjugates¹,

$$\omega_1 = x + iy = \frac{1}{2}(s_2 + is_1) \quad - (A2.26)$$

$$\omega_2 = x - iy = \frac{1}{2}(s_2 - is_1) \quad - (A2.27)$$

The following formulae are based on those of Ling and Tsai (1964), corrected for typographical errors,

$$c_1 = \left(\frac{\pi}{2x}\right)^2 \left(\frac{1}{3} + 2 \sum_{n=1}^{\infty} \frac{1}{\cosh^2((2n-1)\pi\lambda)} - \frac{1}{\sinh^2((2n-1)\pi\lambda)} \right) \quad - (A2.28)$$

$$c_2 = \left(\frac{\pi}{2x}\right)^4 \left(\frac{1}{45} + \frac{4}{3} \sum_{n=1}^{\infty} \frac{-1}{\cosh^2((2n-1)\pi\lambda)} + \frac{3}{2 \cosh^4((2n-1)\pi\lambda)} + \frac{1}{\sinh^2(2n\pi\lambda)} + \frac{3}{2 \sinh^2(2n\pi\lambda)} \right) \quad - (A1.29)$$

¹The half-periods used by Ling and Tsai (1964) were x and $x/2 + iy$. These produce the same values of c_n as those defined above.

$$c_3 = \left(\frac{\pi}{2x}\right)^6 \left[\frac{2}{945} + \frac{4}{15} \sum_{n=1}^{\infty} \left(\frac{1}{\cosh^2(2n-1)\pi\lambda} - \frac{15}{2 \cosh^4(2n-1)\pi\lambda} \right) \right] \\ + \left(\frac{\pi}{2x}\right)^6 \frac{4}{15} \sum_{n=1}^{\infty} \left(\frac{15}{2 \cosh^6(2n-1)\pi\lambda} - \frac{1}{\sinh^2 2n\pi\lambda} - \frac{15}{2 \sinh^4 2n\pi\lambda} - \frac{15}{2 \sinh^6 2n\pi\lambda} \right) \quad - (A2.30)$$

where,

$$\lambda = \frac{y}{2x} \quad - (A2.31)$$

Regardless of whether the primitive-period parallelogram is rectangular or rhombic, c_k can be calculated using the recursion relationship,

$$c_k = \frac{3}{(2k+1)(2k-1)(k-3)} \sum_{n=2}^{k-2} (2n-1)(2k-2n-1)c_n c_{k-n}, \quad k \geq 4 \quad A2.32$$

Values of c_1 , c_2 , and c_3 were calculated using a 400 term series, but excluding values of $n\pi\lambda$ greater than 30. These coefficients were found to agree to within 1 decimal digit of the 16D calculations of Ling and Tsai (1960, 1964) and Ling (1965). Values of $c_4, \dots, c_{200}$ were calculated using a 200 term series based on Eq. A2.32. Selected values of $\log \sigma$, ζ , $\wp$, and $\wp'$, were also computed along the real and imaginary number lines, and were found to agree exactly with the 8D values of Abramowitz and Stegun (1970).

The potential functions [Eq. A2.14] may be written as,

$$W_{2s+1} = z^{*-2s-1} + \sum_{n=0}^{\infty} \alpha_{2n+1} z^{*2n+1} \quad s = 0, 1, 2, \dots \quad - (A2.33)$$

where,

$$\alpha_{1 \ 2n-1} = -\frac{c_n}{s_2^{2n}} \quad - (A2.34)$$

$$\alpha_{2s+1 \ 2n+1} = \left(\frac{2n+2s+2}{2n+1} \right) \frac{2s+1}{2n+2s+2} \alpha_{1 \ 2n+2s+1} \quad s \geq 1 \quad - (A2.35)$$

alternatively a recursion formula may be written for $\alpha_{2s \ 2n+1}$ as follows

$$\alpha_{2s+1 \ 2n+1} = \frac{(2n+3)(2n+2)}{2s(2s-1)} \alpha_{2s-1 \ 2n+3} \quad s \geq 1 \quad - (A2.36)$$

The first 60 values of c_k were used to calculate $\alpha_{1 \ 2n+1}$. The recursion relation Eq. A2.36 was then used in preference to Equation A2.35 to calculate $\alpha_{2s+1 \ 2n+1}$. Due to the presence of the factorial function in Eq. A2.35, very large intermediate numbers are generated in both the numerator and the denominator, use of Eq. A2.36 suppresses this, to some extent. The number of terms (in s and n) used varied, depending on the pitch-to-diameter ratio. Values shown in the computer listing below [Appendix III] are the actual values used to compute the B_{2n+1} coefficients given in Tables 1 to 3.

Appendix III. Listings of Computer Programs

A3.1 Potential flow

```
C  THIS PROGRAM GENERATES TABLES OF B COEFFICIENTS FOR POTENTIAL FLOW IN
C  TUBE BANKS. TABLES OF PRESSURE AND VELOCITY DATA ARE ALSO GENERATED
C  THE FOLLOWING FILES ARE CREATED:
C
C  PCOPlot.DAT  PRESSURE COEFFICIENT VS. ANGLE
C  VTPlot.DAT   TANGENTIAL VELOCITY VS. ANGLE
C  VOPlot.DAT   STREAM-WISE VELOCITY AT PHI = 90
C  TABFIL.DAT   B COEFFICIENTS IN TABULAR FORM FOR S/D
C
C
PROGRAM PCOPlot
COMPLEX W, V, Z
DOUBLE PRECISION AA(0:200,0:200)
DOUBLE PRECISION AAA(0:200), BBB(0:200)
DOUBLE PRECISION ALPHA(0:200,0:200)
PARAMETER (NRUN=4,NANGLE = 90)
DOUBLE PRECISION SSS(1:NRUN), B(1:NRUN,0:200)
DOUBLE PRECISION DIAM, RD, S1, S2
REAL P(1:NRUN,0:NANGLE), VT(1:NRUN,0:NANGLE), U(1:NRUN,0:NANGLE)
LOGICAL INLINE, SQUARE
CHARACTER*12 TABFIL
C
COMMON/PI/ PI
COMMON/WALPOT/PHI0, PSI0
COMMON/BCOEFs/AA,ALPHA,AAA,BBB
COMMON/COORDS/X,Y,RHO,THTA
COMMON/HLPFR/OM0, OM1, OM2
COMMON/SS/S1,S2,INLINE
DATA SSS/1.05,1.1,1.2,1.25/
C  DATA SSS/1.5,2.0,3.0,4.0/
C  DATA SSS/5.0,10.0,20.0,100.0/
C  DATA SSS/2.0,2.0,2.0,2.0/
900 FORMAT(6F10.4)
910 FORMAT (' ANGLE', 4(F10.2), ' SINGLE')
920 FORMAT (' ANGLE', 4(F10.2))
930 FORMAT (A,/)
940 FORMAT (' N',4(3X,' S/D = ',F6.2),/)
950 FORMAT (I4,4(1X,E13.6))
    PI =4.D+00*DATAN(1.D+00)
    DIAM = 1.
C
C  FOR IN-LINE SQUARE SET INLINE=.TRUE., SQUARE=.TRUE.
C  FOR ROTATED SQUARE SET INLINE=.FALSE., SQUARE=.TRUE.
C  FOR EQUILATERAL TRIANGLE SET INLINE=.FALSE., SQUARE=.FALSE.
C
    INLINE = .FALSE.
```

SQUARE = .TRUE.

C

```
IF (INLINE) THEN
  IF (SSS(3).LE.1.25) THEN
    TABFIL = 'IN1.DAT'
  ELSE IF (SSS(3).LE.4.0) THEN
    TABFIL = 'IN2.DAT'
  ELSE
    TABFIL = 'IN3.DAT'
  END IF
ELSE IF (SQUARE) THEN
  IF (SSS(3).LE.1.25) THEN
    TABFIL = 'ROT1.DAT'
  ELSE IF (SSS(3).LE.4.0) THEN
    TABFIL = 'ROT2.DAT'
  ELSE
    TABFIL = 'ROT3.DAT'
  END IF
ELSE
  IF (SSS(3).LE.1.25) THEN
    TABFIL = 'TRI1.DAT'
  ELSE IF (SSS(3).LE.4.0) THEN
    TABFIL = 'TRI2.DAT'
  ELSE
    TABFIL = 'TRI3.DAT'
  END IF
END IF
```

C

```
OPEN(80,FILE='PCOPLLOT.DAT',FORM='FORMATTED',
$STATUS='NEW',ACCESS='SEQUENTIAL')
OPEN(81,FILE='VTPLLOT.DAT',FORM='FORMATTED',
$STATUS='NEW',ACCESS='SEQUENTIAL')
OPEN(82,FILE='V0PLLOT.DAT',FORM='FORMATTED',
$STATUS='NEW',ACCESS='SEQUENTIAL')
OPEN(83,FILE=TABFIL,FORM='FORMATTED',
$STATUS='NEW',ACCESS='SEQUENTIAL')
WRITE (*,*) (' FILE ',TABFIL,' CONTAINS TABLE OF COEFFICENTS')
WRITE (*,*) (' FILE PCOPLLOT.DAT CONTAINS PLOTTING DATA')
WRITE (*,*) (' FILE VTPLLOT.DAT CONTAINS PLOTTING DATA')
WRITE (*,*) (' FILE V0PLLOT.DAT CONTAINS PLOTTING DATA')
DO 60 IRUN = 1, NRUN
  IF (INLINE) THEN
    S1 = SSS(IRUN)
    S2 = SSS(IRUN)
  ELSE IF (.NOT.SQUARE) THEN
    S1 = SSS(IRUN)
    S2 = SSS(IRUN)*(3.**.5)/2.
  ELSE
    S1 = (2.**.5)*SSS(IRUN)
    S2 = .5*S1
  END IF
  IF (SSS(IRUN).LT.1.25) THEN
    NTERMS = 100
  ELSE IF (SSS(IRUN).LT.2.) THEN
    NTERMS = 60
  ELSE IF (SSS(IRUN).LE.4.0) THEN
    IF ((.NOT.INLINE).AND.(.NOT.SQUARE)) NTERMS = 43
```

```
IF ((.NOT.INLINE).AND.SQUARE) NTERMS = 43
IF (INLINE) NTERMS = 41
ELSE
NTERMS = 14
END IF
CALL COMPOT(1.,1.,W,V,NTERMS)
DO 10 I = 0, 200
B(IRUN,I) = BBB(I)
10 CONTINUE
A = SNGL(S1)/DIAM
BB = SNGL(S2)/DIAM
RAD = DIAM/2.
UBULK = 0.0
DO 20 N = 0, NTERMS
IF (ABS(BBB(2*N+1)).GT.1.E-20) THEN
TERM = BBB(2*N+1)*(A**(2*N+1) - A**(-2*N-1))
UBULK = UBULK + TERM*((-1.)**(N+1))
END IF
20 CONTINUE
UBULK = UBULK/(RAD*(A-1))
IF ((.NOT.INLINE).AND.SQUARE) THEN
IF (A.GE.(2.*BB*BB-.5)) THEN
SCAL = (A-1.)/(SQRT(A*A+4.*BB*BB)-2.)
UBULK = SCAL*UBULK
WRITE (*,*) ' COMPACT BUNDLE: SCALE FACTOR =', SCAL
END IF
END IF
C
C CALCULATE VT(THTA) AND P(THTA) AT R = RAD
C
DO 40 I = 0, NANGLE
THTA = PI*(FLOAT(I)/NANGLE)
P(IRUN,I) = 0.
VT(IRUN,I) = 0.
DO 30 N = 0, NTERMS
IF (ABS(BBB(2*N+1)).GT.1.E-20) THEN
VT(IRUN,I) = VT(IRUN,I) -
$FLOAT(2*N+1)*SNGL(BBB(2*N+1))*SIN((2*FLOAT(N)+1.)*THTA)
END IF
30 CONTINUE
P(IRUN,I) = VT(IRUN,I)*VT(IRUN,I)
P(IRUN,I) = 1. - 4.*P(IRUN,I)/
$(RAD*RAD*UBULK*UBULK)
VT(IRUN,I) = 2.*VT(IRUN,I)/(RAD*UBULK)
40 CONTINUE
C
C CALCULATE U(Y) AT X = 0
C
UMAXX = 0.
Y = RAD
DO 50 N = 0, NTERMS
IF (ABS(BBB(2*N+1)).GT.1.E-20) THEN
UMAXX = UMAXX +
$((-1.)**(N+1))*FLOAT(2*N+1)*SNGL(BBB(2*N+1))*
$((Y/RAD)**(2*N) + (RAD/Y)**(2*N+2))
END IF
50 CONTINUE
```

```
UMAXX = UMAXX/RAD
Y = S1/2.
DO 60 N = 0, NTERMS
  IF (ABS(BBB(2*N+1)).GT.1.E-20) THEN
    UMINX = UMINX +
    $((-1.)**(N+1))*FLOAT(2*N+1)*SNGL(BBB(2*N+1))*
    $((Y/RAD)**(2*N) + (RAD/Y)**(2*N+2))
  END IF
60 CONTINUE
  UMINX = UMINX/RAD
  DO 80 I = 0, NANGLE
    Y = RAD + (FLOAT(I)/FLOAT(NANGLE))*(S1/2. - RAD)
    U(IRUN,I) = 0.
    DO 70 N = 0, NTERMS
      IF (ABS(BBB(2*N+1)).GT.1.E-20) THEN
        U(IRUN,I) = U(IRUN,I) +
        $((-1.)**(N+1))*FLOAT(2*N+1)*SNGL(BBB(2*N+1))*
        $((Y/RAD)**(2*N) + (RAD/Y)**(2*N+2))
      END IF
70 CONTINUE
    U(IRUN,I) = U(IRUN,I)/(RAD*UBULK)
80 CONTINUE
C
  WRITE (80,910) (SSS(IRUN), IRUN = 1, NRUN)
  WRITE (81,910) (SSS(IRUN), IRUN = 1, NRUN)
  DO 90 I = 0, NANGLE
    THTA = PI*(FLOAT(I)/NANGLE)
    DTHTA = 180.*(FLOAT(I)/NANGLE)
    PSINGLE = 1 - 4.*SIN(THTA)*SIN(THTA)
C   OUTPUT TANGENTIAL VELOCITY VS. ANGLE.
    VTSINGLE = 2.*SIN(THTA)
    WRITE (81,900) DTHTA, (VT(IRUN,I), IRUN = 1, NRUN), VTSINGLE
C   OUTPUT PRESSURE COEFFICIENT VS. ANGLE.
    WRITE (80,900) DTHTA, (P(IRUN,I), IRUN = 1, NRUN), PSINGLE
90 CONTINUE
C
  WRITE (82,920) (SSS(IRUN), IRUN = 1, NRUN)
C   OUTPUT U(Y) VS Y
  DO 100 I = 0, NANGLE
    Y = FLOAT(I)/FLOAT(NANGLE)
    WRITE (82,900) Y, (U(IRUN,I), IRUN = 1, NRUN)
100 CONTINUE
C
  IF (INLINE) WRITE (83,930) ' INLINE SQUARE BUNDLE'
  IF ((.NOT.INLINE).AND.SQUARE)
    $WRITE (83,930) ' ROTATED SQUARE BUNDLE'
  IF ((.NOT.INLINE).AND.(.NOT.SQUARE))
    $WRITE (83,930) ' EQUILATERAL TRIANGLE BUNDLE'
C   OUTPUT COEFFICIENTS TO A TABLE
  WRITE (83,940) (SSS(IRUN), IRUN = 1, NRUN)
C   IF (INLINE) THEN
  ISTART = 1
  IINC = 2
  DO 110 I = ISTART, NTERMS, IINC
    WRITE (83,950) I, (B(IRUN,I), IRUN = 1, NRUN)
110 CONTINUE
C
```

```
CLOSE(80,STATUS='KEEP')
CLOSE(81,STATUS='KEEP')
CLOSE(82,STATUS='KEEP')
CLOSE(83,STATUS='KEEP')
STOP
END
C
SUBROUTINE COMPOT(X,Y,W,V,NTERMS)
C
C THIS SUBROUTINE COMPUTES THE POTENTIAL W AND VELOCITY V
C AT A POINT X + IY IN THE COMPLEX PLANE
C
IMPLICIT NONE
COMMON/SS/S1,S2,INLINE
COMMON/PI/ PI
COMMON/WALPOT/PHI0, PSI0
COMMON/BCOEFs/AA,ALPHA,AAA,BBB
COMMON/HLFPER/OM0, OM1, OM2
DOUBLE COMPLEX OM1, OM2
COMPLEX W, V, Z, ZSTAR
DOUBLE PRECISION OM0
DOUBLE PRECISION BINOM, CCO, BCO
DOUBLE PRECISION AAA(0:200), BBB(0:200)
DOUBLE PRECISION AA(0:200,0:200)
DOUBLE PRECISION ALPHA(0:200,0:200)
DOUBLE PRECISION LAMDA, MU, PI, CC, PHI0, PSI0
DOUBLE PRECISION RHO, THTA
DOUBLE PRECISION DIAM, RD, S1, S2
REAL X, Y, RSTAR
REAL AMM, ANN
INTEGER NTERMS, N
LOGICAL IDBG, IFLAG, INLINE
DATA IFLAG/.FALSE./
C
PHI0 = 1.D+00
IF (INLINE.EQ..TRUE.) THEN
OM0 = S2/2.D+00
OM1 = DCMPLX(S2/2.D+00,0.D+00)
OM2 = DCMPLX(0.D+00,S1/2.D+00)
DIAM = 1.D+00
ELSE
OM0 = S2/2.D+00
OM1 = DCMPLX(S2,-1.D+00*S1/2.D+00)
OM2 = DCMPLX(S2,+1.D+00*S1/2.D+00)
DIAM = 1.D+00
END IF
C
RD = DIAM/2.D+00
CC = DBLE(OM1)
MU = DIAM/(2.D+00*S2)
PI =4.D+00*DATAN(1.D+00)
C
C COMPUTE B COEFFICIENTS
C
CALL BCO(S1,S2,DIAM,INLINE,NTERMS)
C
C CHECK WITHIN PRIMITIVE REGION
```


C

```
IF (INLINE.EQ..TRUE.) THEN
  AMM = ANINT(X/SNGL(S2))
  ANN = ANINT(Y/SNGL(S1))
ELSE
  AMM = ANINT(X/SNGL(S2))
  IF (ABS(AMOD(AMM,2.)).LT..1) THEN
    ANN = ANINT(Y/SNGL(S1))
  ELSE
    ANN = AINT(Y/SNGL(S1))
  IF (Y.GT.0.) ANN = ANN + 0.5
  IF (Y.LT.0.) ANN = ANN - 0.5
  END IF
  END IF
  Z = CMPLX(X,Y)
  Z = Z - CMPLX(AMM*SNGL(S2),ANN*SNGL(S1))
```

C

```
RHO = SQRT(X*X + Y*Y)/S2
THTA = DBLE(ATAN2(Y,X))
W = CMPLX(0.,0.)
V = CMPLX(0.,0.)
ZSTAR = Z/RD
RSTAR = REAL(ZSTAR)*REAL(ZSTAR) + AIMAG(ZSTAR)*AIMAG(ZSTAR)
RSTAR = SQRT(RSTAR)
```

C

C COMPUTE W AND V TAKING CARE TO AVOID FLOATING POINT UNDERFLOW

C

```
IF (RSTAR.GT.3.E-01) THEN
  DO 10 N = 1, NTERMS
    W = W + BBB(N)*((ZSTAR)**N + (1/ZSTAR)**N)
    V = V - BBB(N)*CMPLX((N/RD))*((ZSTAR)**(N-1) - (1/ZSTAR)**(N+1))
10  CONTINUE
  W = W - 2.*CMPLX(AMM*PHI0,ANN*PSI0)
  ELSE IF (RSTAR.GT.1.E-03) THEN
    DO 20 N = 1, 10
      W = W + BBB(N)*((ZSTAR)**N + (1/ZSTAR)**N)
20  CONTINUE
  W = W - 2.*CMPLX(AMM*PHI0,ANN*PSI0)
  ELSE IF (RSTAR.GT.1.E-06) THEN
    DO 30 N = 1, 5
      W = W + BBB(N)*((ZSTAR)**N + (1/ZSTAR)**N)
30  CONTINUE
  W = W - 2.*CMPLX(AMM*PHI0,ANN*PSI0)
  ELSE IF (RSTAR.GT.1.E-06) THEN
    DO 40 N = 1, 5
      W = W + BBB(N)*((ZSTAR)**N + (1/ZSTAR)**N)
40  CONTINUE
  W = W - 2.*CMPLX(AMM*PHI0,ANN*PSI0)
  ELSE IF (RSTAR.GT.1.E-10) THEN
    DO 50 N = 1, 3
      W = W + BBB(N)*((ZSTAR)**N + (1/ZSTAR)**N)
50  CONTINUE
  W = W - 2.*CMPLX(AMM*PHI0,ANN*PSI0)
  ELSE
    W = - 2.*CMPLX(AMM*PHI0,ANN*PSI0)
  END IF
  IF (RSTAR.LE.1.E+00) V = CMPLX(0.,0.)
```

```

RETURN
END
C
SUBROUTINE BCO(S1,S2,DIAM,INLINE,NTERMS)
C
C THIS PROGRAM CALCULATES THE B_2S+1 COEFICIENTS USED TO
C COMPUTE THE POTENTIAL FIELD
C
C THE FOLLOWING FILE IS CREATED:
C
C BCO.DAT    TABLE OF B COEFFICIENTS
C
IMPLICIT NONE
COMMON/PI/ PI
COMMON/WALPOT/PHI0, PSI0
COMMON/BCOEF/AA,ALPHA,AAA,BBB
DOUBLE COMPLEX Z, SIGMA, LOGSIG, PEE, ZETA
COMMON/COORDS/X,Y,RHO,THTA
REAL X, Y, PHI, PSI
DOUBLE COMPLEX OM1, OM2
DOUBLE PRECISION OHM_2, OHM_4, OHM_6
DOUBLE PRECISION BINOM
DOUBLE PRECISION OM0
DOUBLE PRECISION S1, S2, DIAM
DOUBLE PRECISION C1(1:200)
DOUBLE PRECISION AAA(0:200), BBB(0:200)
DOUBLE PRECISION AA(0:200,0:200)
DOUBLE PRECISION ALPHA(0:200,0:200)
DOUBLE PRECISION LAMDA, MU, PI, CC, PHI0, PSI0
DOUBLE PRECISION G2, G3, E1
DOUBLE PRECISION FACT
DOUBLE COMPLEX SUM, TERM
DOUBLE PRECISION CONST, SQRT3, DUMMY
DOUBLE PRECISION RHO, THTA
DOUBLE PRECISION TERM1, TERM2, SMAX, TMAX
LOGICAL IDBG, IEVEN, INLINE
INTEGER K, M, N, NTERMS, S, R, ICOUNT
DATA C1 /200*0.D+00/
DATA ALPHA, AA / 40401*0.D+00, 40401*0.D+00/
900 FORMAT(1X,A,1X,PD14.7,1X,PD14.7)
910 FORMAT(2(1X,A,1X,PD14.7,1X,PD14.7))
920 FORMAT(' ALPHA('I3,1X,I3,')',1PD11.4)
930 FORMAT(' AA('I2,1X,I2,') = ',1PD11.4)
940 FORMAT(' AAA('I3,') = ',1PD10.3,3X,'BBB('I3,') = ',1PD10.3)
950 FORMAT('//, ' S1 = ',1PD10.4,5X, ' S2 = ',1PD10.4,
    $, 'DIAM = ',1PD10.4,5X, ' NTERMS = ',I4)
960 FORMAT('//,5X, ' I',12X, ' B(I) MAXVAL(I)')
970 FORMAT(5X,I3,5X,1PD12.4,5X,1PD12.4)
980 FORMAT('//, ' INLINE GRID', //)
990 FORMAT('//, ' STAGGERED GRID', //)
995 FORMAT('//, ' PHI0 = ',1PD10.4,5X, ' PSI0 = ',1PD10.4, //)
C
OPEN(84,FILE='BCO.DAT',FORM='FORMATTED',
$STATUS='NEW',ACCESS='SEQUENTIAL')
MU = DIAM/(2.D+00*S2)
OM0 = S2/2.D+00
IF (INLINE.EQ..TRUE.) THEN

```

```
OM1 = DCMLPX(S2/2.D+00,0.D+00)
OM2 = DCMLPX(0.D+00,S1/2.D+00)
ELSE
OM1 = DCMLPX(S2,-1.D+00*S1/2.D+00)
OM2 = DCMLPX(S2,+1.D+00*S1/2.D+00)
END IF
C
C  CALCULATE C COEFFICIENTS
C
C  ZETA(ZIW1,W2) = 1/Z - C(K)Z**2K, K=2,3,4,...
C
CALL CCO(C1,OM1,OM2,INLINE,200)
C
IDBG = .FALSE.
IF (IDBG) THEN
DO 10 K = 1, NTERMS
WRITE (*,*) K, C1(K)
10 CONTINUE
END IF
C
C
C  COMPUTE ALPHA(1,N) COEFFICIENTS
C
C  W(1) = ALPHA(1,S)(Z/(2*W1))**S    S = 0,1,2,...
C
S = 0
N = 0
DO 30 N = 1, NTERMS/2.
ALPHA(1,2*N-1) = -1.D+00*C1(N)*((2.D+00*OM0)**(2*N))
30 CONTINUE
IDBG = .FALSE.
IF (IDBG.EQ..TRUE.) THEN
Z = DCMLPX(X,Y)
ZETA = Z**-1
DO 40 N = 2, NTERMS
ZETA = ZETA + ALPHA(1,N)*(Z**N)/((2.D+00*OM0)**(N+1))
40 CONTINUE
WRITE (*,*)
WRITE (*,900) ' Z    = ', Z
WRITE (*,900) ' ZETA  = ', ZETA
END IF
C
C  COMPUTE THE ALPHA(2S+1,2N+1) COEFFICIENTS
C
C
C  W(2S+1) = ZSTAR**-2S-1 + ALPHA(2N+1,2S+1)ZSTAR**2N+1
C  S = 0,1,2,...
C
DO 50 S = 1, NTERMS/2
DO 50 N = 0, NTERMS/2
ALPHA(2*S+1,2*N+1) = ALPHA(2*S-1,2*N+3)*(2*N+3)*(2.*N+2.) /
$((2.*S)*(2.*S-1))
50 CONTINUE
C
IDBG = .FALSE.
IF (IDBG.EQ..TRUE.) THEN
DO 60 S = 0, NTERMS
DO 60 N = 0, NTERMS - S
```

```
WRITE (*,920) S, N, ALPHA(S,N)
60 CONTINUE
END IF
IDBG = .FALSE.
C
C COMPUTE THE AA FUNCTIONS.
C
C AAA(2N+1) = AA(R,2N+1) R = 0,1,2,...
C
AA(1,0) = -2.D+00*PHI0*MU*MU
DO 70 N = 2, NTERMS
AA(N,0) = 0.D+00
70 CONTINUE
DO 90 R = 1, NTERMS
DO 90 N = 0, NTERMS/2
AA(2*N+1,R) = 0.D+00
DO 80 S = 0, NTERMS/2
AA(2*N+1,R) = AA(2*N+1,R) + ALPHA(2*S+1,2*N+1)*AA(2*S+1,R-1)
80 CONTINUE
AA(2*N+1,R) = (MU**(4*N+2))*AA(2*N+1,R)
90 CONTINUE
IDBG = .FALSE.
IF (IDBG.EQ..TRUE.) THEN
DO 100 N = 0, NTERMS
DO 100 R = 0, NTERMS
WRITE(*,930) N, R, AA(N,R)
100 CONTINUE
END IF
IDBG = .FALSE.
C
C COMPUTE THE AAA AND BBB FUNCTIONS.
C
C W = BBB(2N+1)((Z/RD)**(2N+1) + (RD/Z)**(2N+1))
C N = 0,1,2,...
C BBB(2N+1) = AAA(2N+1)/MU
C
DO 120 N = 0, NTERMS
AAA(N) = 0.D+00
DO 110 R = 0, NTERMS
AAA(N) = AAA(N) + AA(N,R)
110 CONTINUE
BBB(N) = AAA(N)*(MU**(-1*N))
IF (IDBG.EQ..TRUE.) WRITE (*,940) N, AAA(N), N, BBB(N)
120 CONTINUE
C
C WRITE DATA TO FILE.
C
IDBG = .FALSE.
IF (IDBG.EQ..TRUE.) THEN
IF (INLINE.EQ..TRUE.) THEN
WRITE (84,980)
ELSE
WRITE (84,990)
END IF
WRITE (84,950) S1, S2, DIAM, NTERMS
WRITE (84,960)
TERM1 = 0.D+00
```

```

TERM2 = 0.D+00
ICOUNT = 0
DO 130 N = 1, NTERMS, 2
ICOUNT = ICOUNT + 1
TERM1 = TERM1 - 1.D+00*BBB(N)*((S2/DIAM)**N + (DIAM/S2)**N)
TERM2 = TERM2 + ((-1.D+00)**ICOUNT)*BBB(N)*
$((S1/DIAM)**N - (DIAM/S1)**N)
SMAX = DMAX1(S1,S2)
TMAX = BBB(N)*((SMAX/DIAM)**N + (DIAM/SMAX)**N)
WRITE (84,970) N, BBB(N), TMAX
130 CONTINUE
PSI0 = TERM2
WRITE(*,*) TERM1, TERM2
WRITE (84,995) PHI0, PSI0
END IF
CLOSE(84,STATUS='KEEP')
RETURN
END
C
SUBROUTINE CCO(C, W1, W2, INLINE, NTERMS)
C
C THIS SUBROUTINE RETURNS THE COEFFICIENTS OF THE
C LAURENT SERIES FOR:
C
C  $ZETA(Z/W1,W2) = 1/Z - C(K)*Z(2*K-1), \quad K = 2,3,4,\dots$ 
C
IMPLICIT NONE
DOUBLE PRECISION C(1:200)
DOUBLE COMPLEX W1, W2
DOUBLE PRECISION OHM_2, OHM_4, OHM_6
DOUBLE PRECISION BINOM
DOUBLE PRECISION G2, G3, E1
DOUBLE COMPLEX Z, SIGMA, LOGSIG, PEE, ZETA
DOUBLE PRECISION FACT
DOUBLE COMPLEX SUM, TERM
DOUBLE PRECISION RHO, THTA
DOUBLE PRECISION ETA, H1
REAL X, Y, PHI, PSI
INTEGER K, M, NTERMS, N
LOGICAL IDBG, IEVEN, INLINE
COMMON/COORDS/X,Y,RHO,THTA
IDBG = .FALSE.
CALL SUBOHM(W1,W2,INLINE,OHM_2,OHM_4,OHM_6)
G2 = OHM_4*60.D+00
G3 = OHM_6*140.D+00
C(1) = OHM_2
C(2) = OHM_4
C(3) = OHM_6
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' W1   = ', W1
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' W2   = ', W2
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' OHM2  = ', OHM_2
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' OHM4  = ', OHM_4
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' OHM6  = ', OHM_6
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' ETA1  = ', C(1)
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' G2   = ', G2
IF (IDBG.EQ..TRUE.) WRITE(*,*) ' G3   = ', G3

```

C

```
DO 20 K = 4, NTERMS
C(K) = 0.D+00
DO 10 M = 2, K-2
C(K) = C(K) + DBLE((2*M-1)*(2*K-2*M-1))*C(M)*C(K-M)
10 CONTINUE
C(K) = C(K)*3.D+00/DBLE((2*K+1)*(2*K-1)*(K-3))
20 CONTINUE
IDBG = .FALSE.
IF (IDBG.EQ..TRUE.) THEN
DO 30 K = 1, NTERMS
WRITE (*,*) K, C(K)
30 CONTINUE
IDBG = .FALSE.
END IF
IDBG = .FALSE.
ETA = 2.D+00*W1*C(1)
IF (IDBG.EQ..TRUE.) THEN
WRITE (*,*) ' INPUT X, Y'
READ (*,*) X, Y
Z = DCMPLX(X,Y)
PEE = 1.D+00/(Z*Z)
ZETA = 1/Z
LOGSIG = CDLOG(Z)
FACT = 1.D+00
DO 40 N = 2, 20
FACT = FACT*FLOAT(N)
PEE = PEE + DCMPLX((2*N-1)*C(N)*(Z**(2*N-2)))
ZETA = ZETA - DCMPLX(C(N)*(Z**(2*N-1)))
LOGSIG = LOGSIG - DCMPLX(C(N)*(Z**(2*N)))/DCMPLX(2*N)
40 CONTINUE
SIGMA = CDEXP(LOGSIG)
WRITE (*,*)
WRITE (*,*) ' Z      = ',Z
WRITE (*,*) ' P      = ',PEE
WRITE (*,*) ' ZETA    = ',ZETA
WRITE (*,*) ' SIGMA   = ',SIGMA
WRITE (*,*) ' LOGSIG  = ',LOGSIG
WRITE (*,*) ' Z*ZETA  = ',ZETA*Z
WRITE (*,*) ' Z*Z*PEE  = ',PEE*Z*Z
WRITE (*,*)
IDBG = .FALSE.
END IF
RETURN
END
C
SUBROUTINE SUBOHM(W1,W2,INLINE,OHM_2,OHM_4,OHM_6)
C
C THIS SUBROUTINE CALCULATES 1/OMEGA M-N**K K = 2,4,6
C FOR RECTANGULAR AND RHOMBIC PRIMITIVE PERIODS.
C
IMPLICIT NONE
COMMON/PI/ PI
DOUBLE COMPLEX W1, W2
DOUBLE PRECISION OHM_2, OHM_4, OHM_6
DOUBLE PRECISION LAMDA, PI
DOUBLE PRECISION TERM1, TERM2
DOUBLE PRECISION SUM1, SUM2, SUM3, SUM4, SUM5, SUM6
```

```
INTEGER N
LOGICAL INLINE
C
C IF (INLINE.EQ..TRUE.) THEN
C
C   RECTANGULAR
C
C   LAMDA = DABS(DIMAG(W2)/DREAL(W1))
C
C   SUM1 = 0.D+00
C   SUM2 = 0.D+00
C   SUM3 = 0.D+00
C   DO 10 N = 1, 400
C     TERM1 = FLOAT(N)*PI*LAMDA
C     IF (ABS(TERM1).GT.30.) THEN
C       TERM1 = 0.D+00
C     ELSE
C       TERM1 = 1.D+00/DSINH(TERM1)
C       TERM1 = TERM1*TERM1
C     END IF
C     SUM1 = SUM1 + TERM1
C     SUM2 = SUM2 + TERM1*TERM1
C     SUM3 = SUM3 + TERM1*TERM1*TERM1
10  CONTINUE
C   OHM_2 = PI*PI*(1.D+00/3.D+00 - 2.D+00*SUM1)
C   $/(4.D+00*DBLE(W1)*DBLE(W1))
C   OHM_4 = (2.D+00/3.D+00)*(2.D+00*SUM1 + 3.D+00*SUM2)
C   OHM_4 = 1.D+00/45.D+00 + OHM_4
C   OHM_4 = ((PI/(2.D+00*DBLE(W1)))**4)*OHM_4
C   OHM_6 = (2.D+00/15.D+00)*(2.D+00*SUM1 + 15.D+00*SUM2
C   $+ 15.D+00*SUM3)
C   OHM_6 = 2.D+00/945.D+00 - OHM_6
C   OHM_6 = ((PI/(2.D+00*DBLE(W1)))**6)*OHM_6
C
C   ELSE
C
C   RHOMBIC
C
C   LAMDA = DABS(DIMAG(W1)/(2.D+00*DREAL(W1)))
C
C   SUM1 = 0.D+00
C   SUM2 = 0.D+00
C   SUM3 = 0.D+00
C   SUM4 = 0.D+00
C   SUM5 = 0.D+00
C   SUM6 = 0.D+00
C   DO 20 N = 1, 200
C     TERM1 = 2.D+00*FLOAT(N)*PI*LAMDA
C     IF (ABS(TERM1).GT.80.D+00) THEN
C       TERM1 = 0.D+00
C     ELSE
C       TERM1 = 1.D+00/DSINH(TERM1)
C       TERM1 = TERM1*TERM1
C     END IF
C     TERM2 = (2.D+00*FLOAT(N) - 1.D+00)*PI*LAMDA
C     IF (ABS(TERM2).GT.80.D+00) THEN
C       TERM2 = 0.D+00
```

```
ELSE
TERM2 = 1.D+00/DCOSH(TERM2)
TERM2 = TERM2*TERM2
END IF
SUM1 = SUM1 + TERM2
SUM2 = SUM2 + TERM1
SUM3 = SUM3 + TERM1 +
$3.D+00*TERM1*TERM1/2.D+00
SUM4 = SUM4 - TERM1 -
$15.D+00*TERM1*TERM1/2.D+00 -
$15.D+00*TERM1*TERM1*TERM1/2.D+00
SUM5 = SUM5 - TERM2 +
$3.D+00*TERM2*TERM2/2.D+00
SUM6 = SUM6 + TERM2 -
$15.D+00*TERM2*TERM2/2.D+00 +
$15.D+00*TERM2*TERM2*TERM2/2.D+00
20 CONTINUE
OHM_2 = 1.D+00/3.D+00 + 2.D+00*(SUM1 - SUM2)
OHM_2 = OHM_2*((PI/(2.D+00*DREAL(W1)))**2)
OHM_4 = 1.D+00/45.D+00 + 4.D+00/3.D+00*(SUM3 + SUM5)
OHM_4 = OHM_4*((PI/(2.D+00*DREAL(W1)))**4)
OHM_6 = 2.D+00/945.D+00 + 4.D+00/15.D+00*(SUM4 + SUM6)
OHM_6 = OHM_6*((PI/(2.D+00*DREAL(W1)))**6)
END IF
RETURN
END
C
DOUBLE PRECISION FUNCTION BINOM(N,R)
C
IMPLICIT NONE
INTEGER I, ILST, N, R
LOGICAL IDBG
C
BINOM = 1.D+00
IF ((N.LE.1).OR.(R.EQ.0).OR.(R.EQ.N)) THEN
RETURN
ELSE
ILST = JMAX0(N,R)
DO 10 I = 0, ILST
BINOM = BINOM*DMAX1(1.D+00,DBLE(N-I))/
$(DMAX1(1.D+00,DBLE(R-I))*DMAX1(1.D+00,DBLE(N-R-I)))
10 CONTINUE
END IF
IDBG = .FALSE.
IF(IDBG.EQ..TRUE.) WRITE(*,*) N, ' C', R, ' = ', BINOM
RETURN
END
C
```


A3.2 Grid generation

Q1 FILES

TALK=T;RUN(1, 1);VDU=X11-TERM

GROUP 1. Run title and other preliminaries
TEXT(H-GRID GENERATOR)

```
REAL(S,S1, S2, DIAM)
REAL (Y1,Y2,Z1,YPOS, SCLGRD)
REAL (PWRY, PWRZ, PWRY2, PWRCYL, PWRWAKE, RAD, RKK, PI2)
INTEGER(IZ1)
BOOLEAN(LINLINE)
LINLINE=T
DIAM = 1.
S = 1.5
  ** IN-LINE SQUARE
S1 = S;S2 =S
  ** ROTATED SQUARE
  S1 = S*SQRT(2);S2 = S/SQRT(2)
  ** EQUILATERAL TRIANGLE
  S1 = S;S2 = S1*(3**.5)/2.
SCLGRD = 1.1
PWRY = 2.
PWRY2 = 1.
PWRZ = 2.
PWRCYL = 1.
PWRWAKE = 1.
NY = 40
NZ = 20
Z1 = SCLGRD*NZ*3.*DIAM/(2.*S2 + DIAM)
IZ1 = Z1-1
  GROUP 2. Transience time-step specification
STEADY=T
  GROUP 3. X-direction grid specification
  GROUP 4. Y-direction grid specification
GRDPWR(Y,NY,S1/2.,1.)
  GROUP 5. Z-direction grid specification
GRDPWR(Z,NZ,S2/2.,1.0)
  GROUP 6. Body-fitted coordinates or grid distortion
BFC=T

PI2 = 3.14159/2.
RAD = DIAM/2.
DO KK = 1, IZ1+1
  RKK = PI2*(1.-(1.-(KK-1.)/IZ1)**PWRCYL)
  ZC(1,1,:KK:) = RAD*SIN(RKK)
  YC(1,1,:KK:) = RAD*COS(RKK)
  ZC(2,1,:KK:) = RAD*SIN(RKK)
  YC(2,1,:KK:) = RAD*COS(RKK)
ENDDO

DOMAIN(1,NX+1,NY+1,NY+1,1,NZ+1)
SETLIN(ZC,ZL-(ZL-ZF)*(1-LNK)**PWRZ)
```

```
DOMAIN(1,NX+1,NY+1,NY+1,IZ1+1,NZ+1)
SETLIN(ZC,ZL-(ZL-ZF)*(1-LNK))
IF (LINLINE) THEN
DOMAIN(1,NX+1,1,NY+1,1,1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWR Y)
ELSE
DOMAIN(1,NX+1,1,NY/2+1,1,1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWR Y)
DOMAIN(1,NX+1,NY/2+1,NY+1,1,1)
SETLIN(YC,YL-(YL-YF)*(1-LNJ)**PWR Y)
ENDIF
DOMAIN(1,NX+1,1,NY+1,NZ+1,NZ+1)
SETLIN(YC,YF+(YL-YF)*LNJ)
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
SETLIN(YC,YC - YF + DIAM/2.)
SETLIN(ZC,ZC - ZF)
DOMAIN(1,NX+1,1,1,IZ1+1,NZ+1)
SETLIN(ZC,ZF+(ZL-ZF)*LNK**PWR WAKE)
IF (LINLINE) THEN
DOMAIN(1,NX+1,1,NY+1,NZ+1,NZ+1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWR Y2)
ELSE
DOMAIN(1,NX+1,1,NY/2+1,NZ+1,NZ+1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWR Y2)
DOMAIN(1,NX+1,NY/2+1,NY+1,NZ+1,NZ+1)
SETLIN(YC,YL-(YL-YF)*(1-LNJ)**PWR Y2)
ENDIF
VIEW(I,1)
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
MAGIC(T)
DO II = 1, NZ/2
SLIDN=T;SLIDH=F;SLIDL=F;SLIDS=F
LITXC=1;LITYC=20;LITZC=20
RELYC=.1;RELZC=.1
FIXDOM(1,1,NX+1,1,1,1,IZ1+1)
MSWP=NY/10
DO JJ = 2, NY
YPOS = (JJ-1.)/NY
Y1 = (YC(1,,:JJ,2)**2 + ZC(1,,:JJ,2)**2)**.5
Y2 = 1.5*YC(1,,:JJ,2) - .5*YC(1,,:JJ,3)
YC(1,,:JJ,1) = (1.-YPOS)*Y1 + YPOS*Y2
YC(2,,:JJ,1) = YC(1,,:JJ,1)
ENDDO
MAGIC(L)
VIEW(I,1)
ENDDO
```

- GROUP 7. Variables stored, solved & named
- GROUP 8. Terms (in differential equations) & devices
- GROUP 9. Properties of the medium (or media)
- GROUP 10. Inter-phase-transfer processes and properties
- GROUP 11. Initialization of variable or porosity fields
- GROUP 12. Convection and diffusion adjustments
- GROUP 13. Boundary conditions and special sources
- GROUP 14. Downstream pressure for PARAB=.TRUE.
- GROUP 15. Termination of sweeps
- GROUP 16. Termination of iterations

GROUP 17. Under-relaxation devices
GROUP 18. Limits on variables or increments to them
GROUP 19. Data communicated by satellite to GROUND
GROUP 20. Preliminary print-out
GROUP 21. Print-out of variables
GROUP 22. Spot-value print-out
GROUP 23. Field print-out and plot control
GROUP 24. Dumps for restarts
STOP

TALK=T;RUN(1, 1);VDU=X11-TERM

GROUP 1. Run title and other preliminaries
TEXT(I-GRID GENERATOR)
REAL(S,S1, S2, DIAM)
REAL (Y1,RY1,RY2,YPOS,ZPOS)
REAL (PWRY,PWRZ,PWRCYL, PWRWAKE)
REAL (RAD,RKK,PI2)
INTEGER(NZM1,NZP1,NYP1,Y1P)
DIAM = 1.
S = 2.
** IN-LINE SQUARE
S1 = S;S2 = S
** ROTATED SQUARE
S1 = S*SQRT(2);S2 = S/SQRT(2)
** EQUILATERAL TRIANGLE
S1 = S;S2 = S*(3.**.5)/2.

PWRY = 1.5
PWRZ = 1.25
PWRCYL = 1.
PWRWAKE = 3.
NY = 40
NZ = 20
NZP1 = NZ + 1
NZM1 = NZ - 1
NYP1 = NY + 1
Y1 = 1.5*NY*(S2-DIAM)/(S2-DIAM+S1)
GROUP 2. Transience time-step specification
GROUP 3. X-direction grid specification
GROUP 4. Y-direction grid specification
GRDPWR(Y,NY,S1/2,1.0)
GROUP 5. Z-direction grid specification
GRDPWR(Z,NZ,S2/2,1.0)
GROUP 6. Body-fitted coordinates or grid distortion
BFC=T
SETPT(1,1,NZ+1,0.,0.,0.)
SETPT(2,1,NZ+1,1.,0.,0.)
DOMAIN(1,NX+1,1,1,1,NZ+1)
SETLIN(ZC,ZF+(ZL-ZF)*LNK)
SETPT(1,Y1+1,NZ+1,0.,0.,S2/2)
SETPT(2,Y1+1,NZ+1,1.,0.,S2/2)
DOMAIN(1,NX+1,1,Y1+1,NZ+1,NZ+1)
SETLIN(YC,YF)
SETLIN(ZC,ZF+(ZL-ZF)*LNJ)
DOMAIN(1,NX+1,Y1+1,NY+1,NZ+1,NZ+1)
SETLIN(ZC,ZL)

```
SETLIN(YC,YF+(YL-YF)*LNJ)
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
MAGIC(T)
PI2 = 3.14159/2.
RAD = DIAM/2.
DO KK = 1, NZ+1
RKK = PI2*(1.-(1.-(KK-1.)/NZ)**PWRCYL)
ZC(1,1,;KK:) = RAD*SIN(RKK)
YC(1,1,;KK:) = RAD*COS(RKK)
ZC(2,1,;KK:) = RAD*SIN(RKK)
YC(2,1,;KK:) = RAD*COS(RKK)
ENDDO
DOMAIN(1,NX+1,1,NY+1,1,1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWRY)
DOMAIN(1,NX+1,1,Y1+1,NZ+1,NZ+1)
SETLIN(YC,YF)
SETLIN(ZC,ZF+(ZL-ZF)*LNJ**PWRWAKE)
DOMAIN(1,1,NY+1,NY+1,1,NZ+1)
SETLIN(ZC,ZL+(ZF-ZL)*(1-LNK)**PWRZ)
VIEW(I,1)
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
MAGIC(T)
VIEW(I,1)
RELYC=1.;RELZC=1.
DO II= 1, 4
MSWP=5;LITXC=0;LITYC=10;LITZC=10
SLIDN=T
FIXDOM(1,1,NX+1,1,Y1+1,NZ+1,NZ+1)
MAGIC(L)
DO JJ = 2, Y1
YPOS = ((JJ-1.)/Y1)**2
RY1 = (YC(1,;JJ,;NZ)**2 + ZC(1,;JJ,;NZ)**2)**.5
RY2 = 2.*ZC(1,;JJ,;NZ) - ZC(1,;JJ,;NZM1)
ZC(1,;JJ,;NZP1) = (1.-YPOS)*RY1 + YPOS*RY2
YPOS = (JJ-1.)/NY
RY1 = (YC(1,;JJ,;2)**2 + ZC(1,;JJ,;2)**2)**.5
RY2 = 1.5*YC(1,;JJ,;2) - .5*YC(1,;JJ,;3)
YC(1,;JJ,;1) = (1.-YPOS)*RY1 + YPOS*RY2
YC(2,;JJ,;1) = (1.-YPOS)*RY1 + YPOS*RY2
ZC(1,;JJ,;NZP1) = 1.5*ZC(1,;JJ,;NZ) - .5*ZC(1,;JJ,;NZM1)
ZC(2,;JJ,;NZP1) = 1.5*ZC(1,;JJ,;NZ) - .5*ZC(1,;JJ,;NZM1)
ENDDO
YPOS = Y1/NY
Y1P = Y1+1
RY1 = (YC(1,Y1P,2)**2 + ZC(1,Y1P,2)**2)**.5
RY2 = 1.5*YC(1,Y1P,2) - .5*YC(1,Y1P,3)
YC(1,Y1P,1) = (1.-YPOS)*RY1 + YPOS*RY2
YC(2,Y1P,1) = (1.-YPOS)*RY1 + YPOS*RY2
DO JJ = Y1+2, NY+1
YPOS = (JJ-1.)/NY
RY1 = (YC(1,;JJ,;2)**2 + ZC(1,;JJ,;2)**2)**.5
RY2 = 1.5*YC(1,;JJ,;2) - .5*YC(1,;JJ,;3)
YC(1,;JJ,;1) = (1.-YPOS)*RY1 + YPOS*RY2
YC(2,;JJ,;1) = (1.-YPOS)*RY1 + YPOS*RY2
ENDDO
VIEW(I,1)
ENDDO
```

GROUP 7. Variables stored, solved & named
 GROUP 8. Terms (in differential equations) & devices
 GROUP 9. Properties of the medium (or media)
 GROUP 10. Inter-phase-transfer processes and properties
 GROUP 11. Initialization of variable or porosity fields
 GROUP 12. Convection and diffusion adjustments
 GROUP 13. Boundary conditions and special sources
 GROUP 14. Downstream pressure for PARAB=.TRUE.
 GROUP 15. Termination of sweeps
 GROUP 16. Termination of iterations
 GROUP 17. Under-relaxation devices
 GROUP 18. Limits on variables or increments to them
 GROUP 19. Data communicated by satellite to GROUND
 GROUP 20. Preliminary print-out
 GROUP 21. Print-out of variables
 GROUP 22. Spot-value print-out
 GROUP 23. Field print-out and plot control
 GROUP 24. Dumps for restarts
 STOP

TALK=T;RUN(1, 1);VDU=X11-TERM

GROUP 1. Run title and other preliminaries
 TEXT(STAGGERED I-GRID GENERATOR)
 REAL(S, S1, S2, DIAM, RAD)
 REAL (Y1,NYCONC,PWRY,PWRZ,RELX,RY1,RY2,YPOS,PWRCEN)
 REAL(Z1,RZ1,RZ2,ZPOS)
 REAL(PWRCYL,RKK,PI2)
 INTEGER(JMINUS,JPLUS,KPLUS1,KPLUS2,NZP1,NZM1,NYP1,NYM1)
 INTEGER(IY1,IZ1)
 DIAM = 1.
 S = 1.25

** EQUILATERAL TRIANGLES
 S1 = S;S2 = S*(3.**.5)/2.
 ** REVERSE EQUILATERAL TRIANGLE
 S2 = S/2.;S1 = S*(3.**.5)
 ** ROTATED SQUARE
 S1 = S*SQRT(2);S2 = S/SQRT(2)
 PWRY = 3.;PWRZ=3.
 PWRCYL=1.;PWRCEN = 3.0
 NY = 40;NZ = 40
 NZP1 = NZ+1
 NZM1 = NZ-1
 NYP1 =NY+1
 NYM1 = NY-1
 Y1 = NY/2

GROUP 2. Transience time-step specification
 STEADY=T

GROUP 3. X-direction grid specification
 GROUP 4. Y-direction grid specification
 GRDPWR(Y,NY,S1/2,1.)

GROUP 5. Z-direction grid specification
 GRDPWR(Z,NZ,S2,1.)

GROUP 6. Body-fitted coordinates or grid distortion
 BFC=T

```
DOMAIN(1,NX+1,1,1,1,NZ+1)
SETLIN(ZC,ZF)
DOMAIN(1,NX+1,NY+1,NY+1,1,NZ+1)
SETLIN(ZC,ZL)
SETPT(1,Y1+1,NZ+1,0.,0.,S2)
SETPT(2,Y1+1,NZ+1,1.,0.,S2)
SETPT(1,NY-Y1,1,0.,S1/2,0.)
SETPT(2,NY-Y1,1,1.,S1/2,0.)
DOMAIN(1,NX+1,1,Y1+1,NZ+1,NZ+1)
SETLIN(YC,YF)
SETLIN(ZC,ZF+(ZL-ZF)*LNJ**PWRY)
DOMAIN(1,NX+1,Y1+1,NY+1,1,1)
SETLIN(YC,YL)
SETLIN(ZC,ZL+(ZF-ZL)*((1-LNJ)**PWRY))
DOMAIN(1,NX+1,Y1+1,NY+1,NZ+1,NZ+1)
SETLIN(ZC,ZL)
SETLIN(YC,YL+(YF-YL)*((1-LNJ)**PWRY))
DOMAIN(1,NX+1,1,NY-Y1+1,1,1)
SETLIN(ZC,ZF)
SETLIN(YC,YF+(YL-YF)*LNJ**PWRY)
DOMAIN(1,NX+1,1,NY+1,NZ/2+1,NZ/2+1)
SETLIN(YC,YF+(YL-YF)*LNJ)
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
MAGIC(T)
PI2 = 3.14159/2.
RAD = DIAM/2.
DO KK = 1, NZ+1
RKK = PI2*(1.-(1.-(KK-1.)/NZ)**PWRCYL)
ZC(1,1,:KK:) = RAD*SIN(RKK)
YC(1,1,:KK:) = RAD*COS(RKK)
ZC(2,1,:KK:) = RAD*SIN(RKK)
YC(2,1,:KK:) = RAD*COS(RKK)
ENDDO
DO KK = 1, NZ+1
RKK = PI2*((KK-1.)/NZ)**PWRCYL
ZC(1,NYP1,:KK:) = S2-RAD*COS(RKK)
YC(1,NYP1,:KK:) = S1/2.-RAD*SIN(RKK)
ZC(2,NYP1,:KK:) = S2-RAD*COS(RKK)
YC(2,NYP1,:KK:) = S1/2.-RAD*SIN(RKK)
ENDDO
DOMAIN(1,NX+1,1,NY-Y1+1,1,1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWRY)
DOMAIN(1,NX+1,NY-Y1+1,NY+1,1,1)
SETLIN(ZC,ZL+(ZF-ZL)*((1-LNJ)**PWRY))
DOMAIN(1,NX+1,1,Y1+1,NZ+1,NZ+1)
SETLIN(ZC,ZF+(ZL-ZF)*LNJ**PWRY)
DOMAIN(1,NX+1,Y1+1,NY+1,NZ+1,NZ+1)
SETLIN(YC,YL+(YF-YL)*((1-LNJ)**PWRY))
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
DOMAIN(1,NX+1,1,NY-Y1+1,Y1+1,Y1+1)
SETLIN(YC,YF+(YL-YF)*LNJ**PWRY)
SETLIN(ZC,ZF+(ZL-ZF)*LNJ**PWRY)
DOMAIN(1,NX+1,NY-Y1+1,NY+1,Y1+1,Y1+1)
SETLIN(YC,YL+(YF-YL)*((1-LNJ)**PWRY))
SETLIN(ZC,ZL+(ZF-ZL)*((1-LNJ)**PWRY))
DOMAIN(1,NX+1,1,NY/2+1,NZ/2+1,NZ/2+1)
```

```
SETLIN(YC,YF+(YL-YF)*LNJ**PWRcen)
SETLIN(ZC,ZF+(ZL-ZF)*LNJ**PWRcen)
DOMAIN(1,NX+1,NY/2+1,NY+1,NZ/2+1,NZ/2+1)
SETLIN(YC,YL-(YL-YF)*(1-LNJ)**PWRcen)
SETLIN(ZC,ZL-(ZL-ZF)*(1-LNJ)**PWRcen)
DOMAIN(1,NX+1,1,NY+1,1,NZ/2+1)
MAGIC(T)
DOMAIN(1,NX+1,1,NY+1,NZ/2+1,NZ+1)
MAGIC(T)
VIEW(I,1)
MSWP = 5
DO II = 1, NY/4
DOMAIN(1,NX+1,1,NY+1,1,NZ+1)
MAGIC(L)
DO JJ = 2, NY-Y1
YPOS = (JJ-1.)/Y1
RY1 = (YC(1,,:JJ,2)**2 + ZC(1,,:JJ,2)**2)**.5
RY2 = 1.5*YC(1,,:JJ,2) - .5*YC(1,,:JJ,3)
YC(1,,:JJ,1) = (1.-YPOS)*RY1 + YPOS*RY2
YC(2,,:JJ,1) = (1.-YPOS)*RY1 + YPOS*RY2
IY1 = NY+2-JJ
YC(1,IY1,NZP1) = S1/2. - YC(1,,:JJ,1)
YC(2,IY1,NZP1) = YC(1,IY1,NZP1)
ENDDO
DO JJ = 2, Y1
ZPOS = (JJ-1.)/(NY-Y1)
RZ1 = (YC(1,,:JJ,NZ)**2 + ZC(1,,:JJ,NZ)**2)**.5
RZ2 = 1.5*ZC(1,,:JJ,NZ) - .5*ZC(1,,:JJ,NZM1)
ZC(1,,:JJ,NZP1) = (1.-ZPOS)*RZ1 + ZPOS*RZ2
ZC(2,,:JJ,NZP1) = (1.-ZPOS)*RZ1 + ZPOS*RZ2
IY1 = NY+2-JJ
ZC(1,IY1,1) = S2 - ZC(1,,:JJ,NZP1)
ZC(2,IY1,1) = ZC(1,IY1,1)
ENDDO
VIEW(I,1)
ENDDO
STOP
GROUP 7. Variables stored, solved & named
GROUP 8. Terms (in differential equations) & devices
GROUP 9. Properties of the medium (or media)
GROUP 10. Inter-phase-transfer processes and properties
GROUP 11. Initialization of variable or porosity fields
GROUP 12. Convection and diffusion adjustments
GROUP 13. Boundary conditions and special sources
GROUP 14. Downstream pressure for PARAB=.TRUE.
GROUP 15. Termination of sweeps
GROUP 16. Termination of iterations
GROUP 17. Under-relaxation devices
GROUP 18. Limits on variables or increments to them
GROUP 19. Data communicated by satellite to GROUND
GROUP 20. Preliminary print-out
GROUP 21. Print-out of variables
GROUP 22. Spot-value print-out
GROUP 23. Field print-out and plot control
GROUP 24. Dumps for restarts
STOP
```

TALK=T;RUN(1, 1);VDU=X11-TERM

GROUP 1. Run title and other preliminaries
TEXT(BLOCK H-GRID GENERATOR)

REAL(S, S1, S2, DIAM, RD)
REAL (Z1, Z2, Y1, Y2, GEOFAC)
REAL (PWR1, PWR2, PWR3, PI, ANG1, ANG2)
REAL(YYY, ZZZ, YPOS)
INTEGER (NZCYL, NYCYL)
INTEGER (IZ1, IZ2, IY1, IY2)
INTEGER (NP, NPM, NPP)
DIAM = 1.
RD = DIAM/2.
S = 2.
S1 = S; S2 = S
GEOFAC = 1.1
PWR1 = 2.
PWR2 = 1.5
PWR3 = 1.25
NY = 20
NZ = 20
PI=3.14159
NYCYL= 2
ANG1 = (87./90.)*PI/2.
ANG2 = PI/2. - ANG1
Z1 = GEOFAC*NZ*3.*DIAM/(2.*S2 + DIAM)
NZCYL= Z1

GROUP 2. Transience time-step specification

GROUP 3. X-direction grid specification

GROUP 4. Y-direction grid specification

GRDPWR(Y, NY, S1/2., 1.0)

GROUP 5. Z-direction grid specification

GRDPWR(Z, NZ, S2/2, 1.0)

GROUP 6. Body-fitted coordinates or grid distortion

BFC=T

DOMAIN(1, NX+1, 1, NY+1, 1, NZ+1)

SETLIN(YC, YC-YF)

SETLIN(ZC, ZC-ZF)

MSWP=5

IY1 = NYCYL+1

IY2 = NY+1-NYCYL

IZ1 = NZ+1-NZCYL

IZ2 = NZCYL+1

DO KK = 1, NZCYL+1

ZZZ = (:KK:-1)/NZCYL

ZZZ = (1. - ZZZ)**PWR2

ZZZ = (1.-ZZZ)*ANG1

ZC(1, IY1, :KK:) = RD*SIN(ZZZ)

YC(1, IY1, :KK:) = RD*COS(ZZZ)

ZC(2, IY1, :KK:) = RD*SIN(ZZZ)

YC(2, IY1, :KK:) = RD*COS(ZZZ)

ENDDO

DO JJ = 1, NYCYL+1

YYY = (:JJ:-1.)/NYCYL

YYY = (1. - YYY)**PWR2

YYY = (1.-YYY)*ANG2


```
ZC(1,:JJ,IZ2) = RD*COS(YYY)
YC(1,:JJ,IZ2) = RD*SIN(YYY)
ZC(2,:JJ,IZ2) = RD*COS(YYY)
YC(2,:JJ,IZ2) = RD*SIN(YYY)
ENDDO
DOMAIN(1,NX+1,1,NYCYL+1,1,1)
SETLIN(YC,YF+LNJ*(YL-YF))
DOMAIN(1,NX+1,1,1,1,NZCYL+1)
SETLIN(ZC,ZF+LNK*(ZL-ZF))
DOMAIN(1,NX+1,1,NYCYL+1,1,NZCYL+1)
MAGIC(T)
DOMAIN(1,NX+1,NYCYL+1,NY+1,1,1)
SETLIN(YC,YF+(YL-YF)*(LNJ**PWR Y))
DOMAIN(1,NX+1,NYCYL+1,NY+1,NZCYL+1,NZCYL+1)
SETLIN(YC,YF+(YL-YF)*(LNJ**PWR Y))
SETLIN(ZC,ZF)
DOMAIN(1,NX+1,NY+1,NY+1,1,NZCYL+1)
SETLIN(ZC,ZF+(ZL-ZF)*LNK)
DOMAIN(1,NX+1,NYCYL+1,NY+1,1,NZCYL+1)
MAGIC(T)
DOMAIN(1,NX+1,1,1,NZCYL+1,NZ+1)
SETLIN(ZC,ZF+(ZL-ZF)*LNK)
DOMAIN(1,NX+1,NYCYL+1,NYCYL+1,NZCYL+1,NZ+1)
SETLIN(YC,YF)
DO KK = NZCYL+2, NZ
ZC(1,IY1,:KK) = ZC(1,1,:KK)
ENDDO
DOMAIN(1,NX+1,1,NYCYL+1,NZ+1,NZ+1)
SETLIN(YC,YF+(YL-YF)*LNJ)
DOMAIN(1,NX+1,1,NYCYL+1,NZCYL+1,NZ+1)
MAGIC(T)
DOMAIN(1,NX+1,NYCYL+1,NY+1,NZ+1,NZ+1)
SETLIN(YC,YF+(YL-YF)*(LNJ**PWR2))
DOMAIN(1,NX+1,NY+1,NY+1,NZCYL+1,NZ+1)
SETLIN(ZC,ZF+(ZL-ZF)*LNK)
DOMAIN(1,NX+1,NYCYL+1,NY+1,NZCYL+1,NZ+1)
MAGIC(T)
VIEW(I,1)
DOMAIN(1,NX+1,NY+1,NY+1,1,NZ+1)
SETLIN(ZC,ZL-(ZL-ZF)*(1-LNK)**PWRZ)
DOMAIN(1,NX+1,NY+1,NY+1,NZCYL+1,NZ+1)
SETLIN(ZC,ZL-(ZL-ZF)*(1-LNK))
DOMAIN(1,NX+1,NYCYL+1,NY+1,1,NZ+1)
MAGIC(T)
DO II = 1, NZ/4
SLIDN=T;SLIDH=F;SLIDL=F;SLIDS=F
LITXC=1;LITYC=20;LITZC=20
RELYC=.1;RELZC=.1
MSWP=NY/10
DO JJ = NYCYL+2, NY
YPOS = (JJ-NYCYL-1.)/(NY-NYCYL-1.)
Y1 = (YC(1,:JJ,2)**2 + ZC(1,:JJ,2)**2)**.5
Y2 = 1.5*YC(1,:JJ,2) - .5*YC(1,:JJ,3)
YC(1,:JJ,1) = (1.-YPOS)*Y1 + YPOS*Y2
YC(2,:JJ,1) = YC(1,:JJ,1)
ENDDO
DOMAIN(1,NX+1,NYCYL+1,NY+1,1,NZ+1)
```

```
MAGIC(L)
VIEW(I,1)
ENDDO
VIEW(I,1)
STOP
```

FORTTRAN UTILITY

```
C
C  PROGRAM GRID2D
C  2-D ELLIPTIC SOLVER FOR GRID GENERATION
C
  IMPLICIT NONE
  INTEGER NXMAX, NYMAX, NSPEC
  PARAMETER(NXMAX=102,NYMAX=102,NSPEC=4)
  REAL XM(2,NYMAX,NXMAX), YM(2,NYMAX,NXMAX), ZM(2,NYMAX,NXMAX)
  REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
  REAL AMASK(NXMAX,NYMAX)
  REAL G11E(NXMAX,NYMAX), G11N(NXMAX,NYMAX)
  REAL G22E(NXMAX,NYMAX), G22N(NXMAX,NYMAX)
  REAL G12E(NXMAX,NYMAX), G12N(NXMAX,NYMAX)
  REAL DELTA
  REAL GPSQRT(NXMAX,NYMAX)
  REAL DIAM, S, S1, S2
  REAL GESQRT, GNSQRT, GWSQRT, GSSQRT
  REAL DXSENE, DXSWSE, DXSWNW, DXNWNE
  REAL DYSENE, DYSWSE, DYSWNW, DYNWNE
  REAL AE(NXMAX,NYMAX), AN(NXMAX,NYMAX)
  REAL PHI(NXMAX), A(NXMAX), B(NXMAX), C(NXMAX), D(NXMAX)
  REAL SX(NXMAX,NYMAX), SY(NXMAX,NYMAX)
  REAL P(NXMAX,NYMAX), Q(NXMAX,NYMAX)
  REAL PP(NXMAX,NYMAX), QQ(NXMAX,NYMAX)
  REAL PV(NXMAX,NYMAX), QV(NXMAX,NYMAX)
  REAL RCYL, R, THETA
  REAL RELAX, RELAXB
  REAL SXX, SYY, RELAXS
  REAL AX(NSPEC), BX(NSPEC), CX(NSPEC), DX(NSPEC)
  REAL AY(NSPEC), BY(NSPEC), CY(NSPEC), DY(NSPEC)
  INTEGER NMIRROR
  REAL RVAL, RI, RJ, RIJ
  INTEGER IINNER, JINNER
  INTEGER I, J, K
  INTEGER NX, NY, NZ, NBLK
  INTEGER NXM_PLUS1, NYM_PLUS1, NZM_PLUS1
  INTEGER LITER, ISWEEP, LSWEEP
  INTEGER IPT(NSPEC), JPT(NSPEC)
  INTEGER INUM, IREF, ICOR, ICUR
  INTEGER II, IBEGIN, IEND
  LOGICAL L, ROT, TRI, NONORT, STAGGERED
  REAL Y, X1, X2, Y1, RR, SPPT, ANGFRG
  COMMON/DIMENS/ NX, NY
  COMMON/METRICS/G11E, G22E, G12E, G11N, G22N, G12N, GPSQRT
  COMMON/COORDS/XC, YC
  COMMON/BITMAP/AMASK
  COMMON/DOMAIN/DIAM,S,S1,S2, SPPT, STAGGERED, NBLK, ANGFRG
  COMMON/LINCO/AE, AN
  COMMON/SORS/P, Q, SX, SY, PP, QQ, PV, QV
```

```
DATA AX,BX,CX,DX,AY,BY,CY,DY/NSPEC*0.,NSPEC*0.,NSPEC*0.,NSPEC*0.,
$NSPEC*0.,NSPEC*0.,NSPEC*0.,NSPEC*0./
DATA IPT(1),JPT(1),IPT(2),JPT(2),ICUR/20,1,12,11,1/
C
C  INITIAL SETTINGS:
C
  NX = 41
  NY = 41
  DELTA = 1.E-12
  TRI = .FALSE.
  ROT = .TRUE.
  NONORT = .TRUE.
  STAGGERED = .TRUE.
  DIAM = 1.
  SPPT = 1.6
  S = 2.
  IF (TRI) THEN
    S1 = S
    S2 = S1*SQRT(3.)/2.
  ELSE IF (ROT) THEN
    S1 = S*SQRT(2.)
    S2 = S/SQRT(2.)
  ELSE
    S1 = S
    S2 = S
  END IF
  LSWEPT = 1
  LITER = 10
  RELAX = .1
  RELAXB = .1
  RELAXS = .1
  NBLK = 0
  ANGFRCT = .95
  BX(1) = 0
  DX(1) = 0
  BY(1) = 0
  DY(1) = 0
  BX(2) = 0
  DX(2) = 0
  BY(2) = 0
  DY(2) = 0
C
C  READ DATA FROM FILE OR CREATE INITIAL GRID
C
  CALL MASK(1,NX,1,NY,RELAX)
  CALL CLEAR_PARAMS
  CALL OPEN_GRAPHICS
  CALL INITIAL_GRID
C  CALL READ_GRID
  CALL DRAW_GRID
C
  IF (STAGGERED) THEN
    IPT(1) = INT(SPPT*(NX-1)*3.*DIAM/(4*S2+2.*DIAM))
    IPT(3) = INT(SPPT*(NX-1)*3.*DIAM/(4*S2+2.*DIAM))
  ELSE
    IPT(1) = INT(SPPT*(NX-1)*3.*DIAM/(2*S2+DIAM))
    IPT(3) = INT(SPPT*(NX-1)*3.*DIAM/(2*S2+DIAM))
```

```
END IF
IPT(2) = NX+1 - IPT(1)
IPT(4) = NX+1 - IPT(1)
JPT(1)= 1
JPT(2)= NY
JPT(3)= 1
JPT(4)= NY
C
C  MAIN LOOP
C
100 CONTINUE
C
901 FORMAT (' (1) SPECIAL POINT ',I3,' (2) I = ', I3,
$' (3) J = ', I3)
902 FORMAT (' (4) AX = ',G10.4,' (5) BX = ', G10.4,
$' (6) CX = ', G10.4,' (7) DX = ', G10.4)
903 FORMAT (' (8) AY = ',G10.4,' (9) BY = ', G10.4,
$' (10) CY = ', G10.4,' (11) DY = ', G10.4)
904 FORMAT (' (12) SWEEP= ',I3,' (13) NMIRROR = ', I3,
$' (14) SPPT = ', G10.4)
905 FORMAT (' (15) NX= ',I3,' (16) NY = ', I3,
$' (17) RELAX = ', G10.4)
906 FORMAT (' (18) RELAXB= ',G10.4,' (19) S= ',G10.4,
$'(20) NBLK = ', I2)
907 FORMAT (' (21) ANGFRG = ',G10.4)
908 FORMAT (' TO CHANGE A VALUE INPUT REFERENCE NO. AND VALUE AS
$A PAIR OF NUMBERS')
909 FORMAT (' INPUT 99 99 TO ACTIVATE SOLVER, 0 0 TO EXIT')
WRITE (6,901) ICUR, IPT(ICUR), JPT(ICUR)
WRITE (6,902) AX(ICUR), BX(ICUR), CX(ICUR), DX(ICUR)
WRITE (6,903) AY(ICUR), BY(ICUR), CY(ICUR), DY(ICUR)
WRITE (6,904) LSWEEP, NMIRROR, SPPT
WRITE (6,905) NX, NY, RELAX
WRITE (6,906) RELAXB, S, NBLK
WRITE (6,907) ANGFRG
WRITE (6,907) ANGFRG
WRITE (6,908)
WRITE (6,909)
READ(5,*) INUM, RVAL
IF (INUM.EQ.0) THEN
GO TO 300
ELSE IF (INUM.EQ.1) THEN
ICUR = RVAL
ELSE IF (INUM.EQ.2) THEN
IPT(ICUR) = RVAL
ELSE IF (INUM.EQ.3) THEN
JPT(ICUR) = RVAL
ELSE IF (INUM.EQ.4) THEN
AX(ICUR) = RVAL
ELSE IF (INUM.EQ.5) THEN
BX(ICUR) = RVAL
ELSE IF (INUM.EQ.6) THEN
CX(ICUR) = RVAL
ELSE IF (INUM.EQ.7) THEN
DX(ICUR) = RVAL
ELSE IF (INUM.EQ.8) THEN
AY(ICUR) = RVAL
```

```
ELSE IF (INUM.EQ.9) THEN
  BY(ICUR) = RVAL
ELSE IF (INUM.EQ.10) THEN
  CY(ICUR) = RVAL
ELSE IF (INUM.EQ.11) THEN
  DY(ICUR) = RVAL
ELSE IF (INUM.EQ.12) THEN
  LSWEEP = INT(RVAL)
ELSE IF (INUM.EQ.13) THEN
  NMIRROR = RVAL
ELSE IF (INUM.EQ.14) THEN
  SPPT = RVAL
ELSE IF (INUM.EQ.15) THEN
  NX = NINT(RVAL)
  CALL MASK(1,NX,1,NY,RELAX)
ELSE IF (INUM.EQ.16) THEN
  NY = NINT(RVAL)
  CALL MASK(1,NX,1,NY,RELAX)
ELSE IF (INUM.EQ.17) THEN
  RELAX = RVAL
  CALL MASK(1,NX,1,NY,RELAX)
ELSE IF (INUM.EQ.18) THEN
  RELAXB = RVAL
ELSE IF (INUM.EQ.19) THEN
  S = RVAL
  IF (TRI) THEN
    S1 = S
    S2 = S1*SQRT(3.)/2.
  ELSE IF (ROT) THEN
    S1 = S*SQRT(2.)
    S2 = S/SQRT(2.)
  ELSE
    S1 = S
    S2 = S
  END IF
  CALL CLOSE_GRAPHICS
  CALL OPEN_GRAPHICS
ELSE IF (INUM.EQ.20) THEN
  NBLK = NINT(RVAL)
  CALL MASK(1,NX,1,NBLK,0.)
  CALL MASK(1,NX,NY-NBLK+1,NY,0.)
  CALL MASK(1,NX,NBLK+1,NY-NBLK,RELAX)
ELSE IF (INUM.EQ.21) THEN
  ANGFRC = RVAL
END IF
IF (STAGGERED) THEN
  IPT(1) = INT(SPPT*(NX-1)*3.*DIAM/(4*S2+2.*DIAM))
  JPT(1)= NBLK+1
ELSE
  IPT(1) = INT(SPPT*(NX-1)*3.*DIAM/(2*S2+DIAM))
END IF
IPT(2) = NX+1 - IPT(1)
JPT(2)= NY-NBLK
IF (INUM.NE.99) GOTO 100
C
  CALL INITIAL_GRID
C  CALL READ_GRID
```

```
C
C  COMPUTE FORCING FUNCTION SOURCE-TERMS P AND Q
C
  DO 10 J = 1, NY
  DO 10 I = 1, NX
C
  PP(I,J) = 0.
  QQ(I,J) = 0.
C
  DO 10 IREF = 1, NSPEC
C
C  POINT AND LINE SOURCES/SINKS
C
  IF (IREF.EQ.1) THEN
    IBEGIN = 1
    IEND = IPT(IREF)
  ELSE IF (IREF.EQ.2) THEN
    IBEGIN = IPT(IREF)
    IEND = NX
  ELSE
    IBEGIN = IPT(IREF)
    IEND = IPT(IREF)
  END IF
C
  DO 10 II = IBEGIN, IEND
    RI = FLOAT(I-II)/FLOAT(NY)
    RJ = FLOAT(J-JPT(IREF))/FLOAT(NY)
    RIJ = FLOAT((I-II)**2
    $+ (J-JPT(IREF))**2)
    RIJ = SQRT(RIJ)/FLOAT(NX*NY)
    PP(I,J) = PP(I,J) -
    &AX(IREF)*SIGN(1.,RI)*EXP(-1.*CX(IREF)*ABS(RI))
    PP(I,J) = PP(I,J) -
    &BX(IREF)*SIGN(1.,RJ)*EXP(-1.*DX(IREF)*ABS(RIJ))
    QQ(I,J) = QQ(I,J) -
    &AY(IREF)*SIGN(1.,RJ)*EXP(-1.*CY(IREF)*ABS(RJ))
    QQ(I,J) = QQ(I,J) -
    &BY(IREF)*SIGN(1.,RJ)*EXP(-1.*DY(IREF)*ABS(RIJ))
    IF (NMIRROR.EQ.1) THEN
      IF (JPT(IREF).EQ.1) THEN
        RI = FLOAT(I-IPT(IREF))/FLOAT(NY)
        RJ = FLOAT(J-2*NY)/FLOAT(NY)
        RIJ = FLOAT((I-IPT(IREF))**2
        $+ (J-2*NY)**2)
        RIJ = SQRT(RIJ)/FLOAT(NX*NY)
        PP(I,J) = PP(I,J) -
        $AX(IREF)*SIGN(1.,RI)*EXP(-1.*CX(IREF)*ABS(RI))
        PP(I,J) = PP(I,J) -
        $BX(IREF)*SIGN(1.,RJ)*EXP(-1.*DX(IREF)*ABS(RIJ))
        QQ(I,J) = QQ(I,J) -
        $AY(IREF)*SIGN(1.,RJ)*EXP(-1.*CY(IREF)*ABS(RJ))
        QQ(I,J) = QQ(I,J) -
        $BY(IREF)*SIGN(1.,RJ)*EXP(-1.*DY(IREF)*ABS(RIJ))
      ELSE IF (JPT(IREF).EQ.NY) THEN
        RI = FLOAT(I-IPT(IREF))/FLOAT(NY)
        RJ = FLOAT(J+2*NY)/FLOAT(NY)
        RIJ = FLOAT((I-IPT(IREF))**2
```

```
$+ (J+2*NY)**2)
RIJ = SQRT(RIJ)/FLOAT(NX*NY)
PP(I,J) = PP(I,J) -
$AX(IREF)*SIGN(1.,RI)*EXP(-1.*CX(IREF)*ABS(RI))
PP(I,J) = PP(I,J) -
$BX(IREF)*SIGN(1.,RI)*EXP(-1.*DX(IREF)*ABS(RIJ))
QQ(I,J) = QQ(I,J) -
$AY(IREF)*SIGN(1.,RJ)*EXP(-1.*CY(IREF)*ABS(RJ))
QQ(I,J) = QQ(I,J) -
$BY(IREF)*SIGN(1.,RJ)*EXP(-1.*DY(IREF)*ABS(RIJ))
END IF
END IF
10 CONTINUE
C
C  MAIN SOLVER LOOP
C
DO 200 ISWEEP = 1, LSWEEP
CALL DRAW_GRID
C
C  WRITE (6,*) ' ISWEEP =', ISWEEP
C  WRITE (6,*) XC(NX/2,NY/2), YC(NX/2,NY/2)
C
DO 20 J = 1, NY
DO 20 I = 1, NX
P(I,J) = 0.
Q(I,J) = 0.
SX(I,J) = 0.
SY(I,J) = 0.
20 CONTINUE
C
C  COMPUTE LINEAR COEFFICIENTS AND BUILT-IN SOURCE TERMS
C
CALL METRIC(NONORT,1,NX,1+NBLK,NY-NBLK)
C
DO 30 J = 2, NY-1
DO 30 I = 1, NX-1
GESQRT = G11E(I,J)*G22E(I,J)-G12E(I,J)*G12E(I,J)
GESQRT = SQRT(AMAX1(DELTA,GESQRT))
AE(I,J) = G22E(I,J)/GESQRT
30 CONTINUE
DO 40 J = 1, NY-1
DO 40 I = 2, NX-1
GNSQRT = G11N(I,J)*G22N(I,J)-G12N(I,J)*G12N(I,J)
GNSQRT = SQRT(AMAX1(DELTA,GNSQRT))
AN(I,J) = G11N(I,J)/GNSQRT
40 CONTINUE
C
C  BUILT IN SOURCE-TERMS DUE TO NON-ORTHOGONALITY
C
IF (NONORT) THEN
DO 50 J = 2, NY-1
DO 50 I = 2, NX-1
GESQRT = G11E(I,J)*G22E(I,J)-G12E(I,J)*G12E(I,J)
GWSQRT = G11E(I-1,J)*G22E(I-1,J)-G12E(I-1,J)*G12E(I-1,J)
GNSQRT = G11N(I,J)*G22N(I,J)-G12N(I,J)*G12N(I,J)
GSSQRT = G11N(I,J-1)*G22N(I,J-1)-G12N(I,J-1)*G12N(I,J-1)
GESQRT = SQRT(GESQRT)
```

```
GWSQRT = SQRT(GWSQRT)
GNSQRT = SQRT(GNSQRT)
GSSQRT = SQRT(GSSQRT)
DXSENE = XC(I,J+1)+XC(I+1,J+1)-XC(I,J-1)-XC(I+1,J-1)
DXSWSE = XC(I+1,J)+XC(I+1,J-1)-XC(I-1,J)-XC(I-1,J-1)
DXSWNW = XC(I,J+1)+XC(I-1,J+1)-XC(I,J-1)-XC(I-1,J-1)
DXNWNE = XC(I+1,J+1)+XC(I+1,J)-XC(I-1,J+1)-XC(I-1,J)
DYSENE = YC(I,J+1)+YC(I+1,J+1)-YC(I,J-1)-YC(I+1,J-1)
DYSWSE = YC(I+1,J)+YC(I+1,J-1)-YC(I-1,J)-YC(I-1,J-1)
DYSWNW = YC(I,J+1)+YC(I-1,J+1)-YC(I,J-1)-YC(I-1,J-1)
DYNWNE = YC(I+1,J+1)+YC(I+1,J)-YC(I-1,J+1)-YC(I-1,J)
SXX = G12E(I-1,J)*DXSWNW/GWSQRT
$- G12E(I,J)*DXSENE/GESQRT
$+ G12N(I,J-1)*DXSWSE/GSSQRT
$- G12N(I,J)*DXNWNE/GNSQRT
SYY = G12E(I-1,J)*DYSWNW/GWSQRT
$- G12E(I,J)*DYSENE/GESQRT
$+ G12N(I,J-1)*DYSWSE/GSSQRT
$- G12N(I,J)*DYNWNE/GNSQRT
SX(I,J) = (1.-RELAXS)*SX(I,J) + RELAXS*SXX
SY(I,J) = (1.-RELAXS)*SY(I,J) + RELAXS*SYY
50 CONTINUE
END IF
DO 60 J = 2, NY-1
DO 60 I = 2, NX-1
P(I,J) = (1.-RELAXS)*P(I,J)+RELAXS*(PP(I,J))*GPSQRT(I,J)
Q(I,J) = (1.-RELAXS)*Q(I,J)+RELAXS*(QQ(I,J))*GPSQRT(I,J)
60 CONTINUE
CALL SOLVER(LITER,RELAXS)
CALL SLIDER(RELAXB,1,NX,1+NBLK,NY-NBLK)
IF (STAGGERED) CALL MIRROR(1,NX,1,NY)
200 CONTINUE
GO TO 100
C
C END OF MAIN SOLVER LOOP
C
300 CONTINUE
C
WRITE (6,*) ' WRITING OUTPUT DATA '
C
C 5. WRITE DATA
C
CALL WRITE_GRID
CALL CLOSE_GRAPHICS
STOP
END
C
SUBROUTINE TRANSFINITE(XC,YC,IX1,IX2,IY1,IY2)
C THIS SUBROUTINE PERFORMS TRANSFINITE INTERPOLATION
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX
INTEGER IX1, IX2, IY1, IY2
PARAMETER(NXMAX=102,NYMAX=102)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL X1(NXMAX,NYMAX), Y1(NXMAX,NYMAX)
REAL X2(NXMAX,NYMAX), Y2(NXMAX,NYMAX)
```



```
REAL RI1, RI2, RJ1, RJ2
INTEGER NY, NX
INTEGER I, J
COMMON/DIMENS/NX,NY
C
DO 10 J = IY1, IY2
DO 10 I = IX1, IX2
RI1 = FLOAT(I-IX1)/FLOAT(IX2-IX1)
RI2 = FLOAT(IX2-I)/FLOAT(IX2-IX1)
X1(I,J)=RI1*XC(IX2,J)+RI2*XC(IX1,J)
Y1(I,J)=RI1*YC(IX2,J)+RI2*YC(IX1,J)
10 CONTINUE
DO 20 J = IY1, IY2
DO 20 I = IX1, IX2
RJ1 = FLOAT(J-IY1)/FLOAT(IY2-IY1)
RJ2 = FLOAT(IY2-J)/FLOAT(IY2-IY1)
X2(I,J)=RJ1*(XC(I,IY2)-X1(I,IY2)) + RJ2*(XC(I,IY1)-X1(I,IY1))
Y2(I,J)=RJ1*(YC(I,IY2)-Y1(I,IY2)) + RJ2*(YC(I,IY1)-Y1(I,IY1))
20 CONTINUE
DO 30 J = IY1, IY2
DO 30 I = IX1, IX2
XC(I,J)=X1(I,J)+X2(I,J)
YC(I,J)=Y1(I,J)+Y2(I,J)
30 CONTINUE
RETURN
END
C
SUBROUTINE METRIC(NONORT,IX1,IX2,IY1,IY2)
C THIS SUBROUTINE COMPUTES THE INTERFACE AND THE CELL-CENTRE METRICS
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX, NSPEC
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL G11E(NXMAX,NYMAX), G11N(NXMAX,NYMAX)
REAL G22E(NXMAX,NYMAX), G22N(NXMAX,NYMAX)
REAL G12E(NXMAX,NYMAX), G12N(NXMAX,NYMAX)
REAL GPSQRT(NXMAX,NYMAX)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL GP11, GP22, GP12
REAL RELAX
LOGICAL NONORT
INTEGER NY, NX
INTEGER IX1, IX2, IY1, IY2
INTEGER I, J, N
COMMON/DIMENS/NX,NY
COMMON/METRICS/G11E, G22E, G12E, G11N, G22N, G12N, GPSQRT
COMMON/COORDS/XC, YC
RELAX = 1.
C
C COMPUTE METRICS AT EAST INTERFACE
C
DO 10 J = 1, NY
DO 10 I = 1, NX-1
G11E(I,J) = (1.-RELAX)*G11E(I,J) +
$RELAX*(XC(I+1,J)-XC(I,J))**2+(YC(I+1,J)-YC(I,J))**2
10 CONTINUE
DO 20 J = 2, NY-1
```

```
DO 20 I = 1, NX-1
  IF (J.LE.2) THEN
    G22E(I,J) = (1.-RELAX)*G22E(I,J) +
    $RELAX*((XC(I,J+1) + XC(I+1,J+1) - XC(I,J) - XC(I+1,J))**2
    $+ (YC(I,J+1) + YC(I+1,J+1) - YC(I,J) - YC(I+1,J))**2)/4.
  ELSE IF (J.LT.NY-1) THEN
    G22E(I,J) = (1.-RELAX)*G22E(I,J) +
    $RELAX*((XC(I,J+1) + XC(I+1,J+1) - XC(I,J-1) - XC(I+1,J-1))**2
    $+ (YC(I,J+1) + YC(I+1,J+1) - YC(I,J-1) - YC(I+1,J-1))**2)/16.
  ELSE
    G22E(I,J) = (1.-RELAX)*G22E(I,J) +
    $RELAX*((XC(I,J) + XC(I+1,J) - XC(I,J-1) - XC(I+1,J-1))**2
    $+ (YC(I,J) + YC(I+1,J) - YC(I,J-1) - YC(I+1,J-1))**2)/4.
  END IF
  IF (NONORT) G12E(I,J) = (1.-RELAX)*G12E(I,J) +
  $RELAX*((XC(I+1,J)-XC(I,J))*
  $(XC(I,J+1) + XC(I+1,J+1) - XC(I,J-1) - XC(I+1,J-1))
  $+ (YC(I+1,J)-YC(I,J))*
  $(YC(I,J+1) + YC(I+1,J+1) - YC(I,J-1) - YC(I+1,J-1)))/4.
20 CONTINUE
C
C  COMPUTE METRICS AT NORTH INTERFACE
C
  DO 30 J = 1, NY-1
    DO 30 I = 1, NX-1
      G22N(I,J) = (1.-RELAX)*G22N(I,J) +
      $RELAX*(XC(I,J+1)-XC(I,J))**2+(YC(I,J+1)-YC(I,J))**2
      IF (J.EQ.1) THEN
        G22N(I,J) = (1.-RELAX)*G22N(I,J+1) +
        $RELAX*(XC(I,J+2)-XC(I,J+1))**2+(YC(I,J+2)-YC(I,J+1))**2
      END IF
30 CONTINUE
    DO 40 J = 1, NY-1
      DO 40 I = 2, NX-1
        G11N(I,J) = (1.-RELAX)*G11N(I,J) +
        $RELAX*((XC(I+1,J) + XC(I+1,J+1) - XC(I-1,J) - XC(I-1,J+1))**2
        $+ (YC(I+1,J) + YC(I+1,J+1) - YC(I-1,J) - YC(I-1,J+1))**2)/16.
        IF (NONORT) G12N(I,J) = (1.-RELAX)*G12N(I,J) +
        $RELAX*((XC(I,J+1)-XC(I,J))*
        $(XC(I+1,J) + XC(I+1,J+1) - XC(I-1,J) - XC(I-1,J+1))
        $+ (YC(I,J+1)-YC(I,J))*
        $(YC(I+1,J) + YC(I+1,J+1) - YC(I-1,J) - YC(I-1,J+1)))/4.
40 CONTINUE
      C
      C  COMPUTE JACOBIAN AT NODE CENTRE
      C
        DO 50 J = 2, NY-1
          DO 50 I = 2, NX-1
            GP11 =
            $((XC(I+1,J)-XC(I-1,J))**2+(YC(I+1,J)-YC(I-1,J))**2)/4.
            GP22 =
            $((XC(I,J+1)-XC(I,J-1))**2+(YC(I,J+1)-YC(I,J-1))**2)/4.
            GP12 =
            $((XC(I+1,J)-XC(I-1,J))*(XC(I,J+1)-XC(I,J-1))
            $+ (YC(I+1,J)-YC(I-1,J))*(YC(I,J+1)-YC(I,J-1)))/4.
            IF (NONORT) THEN
              GPSQRT(I,J) = SQRT(GP11*GP22-GP12*GP12)
```

```
ELSE
GPSQRT(I,J) = SQRT(GP11*GP22)
END IF
50 CONTINUE
RETURN
END
C
SUBROUTINE TDMA(PHI, A, B, C, D, N)
C THIS SUBROUTINE SOLVES AN ARRAY USING THE TRI-DIAGONAL MATRIX ALGO-
RITHM.
C
IMPLICIT NONE
INTEGER NPHI
PARAMETER (NPHI=102)
REAL PHI(NPHI), A(NPHI), B(NPHI), C(NPHI), D(NPHI)
REAL P(NPHI), Q(NPHI)
INTEGER I, N
P(1) = B(1)/A(1)
Q(1) = D(1)/A(1)
DO 10 I = 2, N
P(I) = B(I)/(A(I) - C(I)*P(I-1))
Q(I) = (D(I) + C(I)*Q(I-1))/(A(I) - C(I)*P(I-1))
10 CONTINUE
PHI(N) = Q(N)
DO 20 I = N-1, 1, -1
PHI(I) = P(I)*PHI(I+1) + Q(I)
20 CONTINUE
RETURN
END
C
SUBROUTINE INITIAL_GRID
C
C THIS SUBROUTINE SETS UP AN INITIAL GRID BASED ON A BLOCK H-GRID
C STRUCTURE
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX, NSPEC
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL R, THETA, PWRY, PWRY2, PWRX, PWRCYL, PWRWAKE
REAL S1, S2, S, DIAM, RAD, PI2, SPPT
REAL ANG1, ANG2, ANGFRG
INTEGER I, J, K
INTEGER NX, NY, NZ, NBLK
INTEGER IX1, IX2, IY1, IY2
LOGICAL STAGGERED, LTEST
COMMON/COORDS/XC, YC
COMMON/DIMENS/ NX, NY
COMMON/DOMAIN/DIAM,S,S1,S2, SPPT, STAGGERED, NBLK, ANGFRG
C
LTEST = .FALSE.
IF (LTEST) THEN
DO 99 I = 1, NX
DO 99 J = 1, NY
XC(I,J) = (FLOAT(I)/FLOAT(NX))**1.
YC(I,J) = (FLOAT(J)/FLOAT(NY))**1.
99 CONTINUE
```

```
RETURN
END IF
RAD = DIAM/2.
IF (.NOT.STAGGERED) THEN
IX1 = INT(SPPT*(NX-1)*3.*DIAM/(2.*S2+DIAM))
ELSE
IX1 = INT(SPPT*(NX-1)*3.*DIAM/(4.*S2+2.*DIAM))
END IF
IX2 = NX+1 - IX1
IY1 = 1 + NBLK
IY2 = NY - NBLK
IF (NBLK.GT.0) THEN
CALL MASK(1,IX1,1,IY1,0.)
CALL MASK(IX2,NX,IY2,NY,0.)
CALL MASK(1,NX,1,IY1,0.)
CALL MASK(1,NX,IY2,NY,0.)
END IF
PWRY = 1.6
PWRY2 = 1.
PWRX = 1.6
PWRCYL = 1.
PWRWAKE = 1.
PWRCYL = .8
PI2 = 2.*ATAN(1.)
ANG1 = ANGFRN*PI2
ANG2 = PI2 - ANG1
IF (NBLK.EQ.0) ANG1 = PI2
IF (NBLK.EQ.0) ANG2 = 0.
XC(1,1) = 0.
YC(1,1) = DIAM/2.
XC(NX,1) = S2/2.
YC(NX,1) = 0.
XC(1,NY) = 0.
YC(1,NY) = S1/2.
XC(NX,NY) = S2/2.
YC(NX,NY) = S1/2.
```

C

```
IF (.NOT.STAGGERED) THEN
DO 10 I = 1, IX1
THETA = PI2*(1.-(1.-(FLOAT(I)-1)/(IX1-1))**PWRCYL)
XC(I,1) = RAD*SIN(THETA)
YC(I,1) = RAD*COS(THETA)
10 CONTINUE
DO 20 I = IX1, NX
R = FLOAT(I-IX1)/FLOAT(NX-IX1)
XC(I,1) = XC(IX1,1) + (XC(NX,1) - XC(IX1,1))*(R**PWRWAKE)
YC(I,1) = 0.
20 CONTINUE
DO 30 I = 1, NX
R = FLOAT(I-1)/FLOAT(NX-1)
XC(I,NY) = XC(NX,NY) - (XC(NX,NY) - XC(1,1))*((1.-R)**PWRX)
YC(I,NY) = S1/2.
30 CONTINUE
DO 40 I = IX1, NX
R = FLOAT(I-IX1)/FLOAT(NX-IX1)
XC(I,NY) = XC(IX1,NY) + (XC(NX,NY) - XC(IX1,NY))*R
YC(I,NY) = S1/2.
```

```
40 CONTINUE
  DO 50 J = 1, NY
    R = FLOAT(J-1)/FLOAT(NY-1)
    XC(1,J) = 0.
    YC(1,J) = YC(1,1) + (YC(1,NY) - YC(1,1))*(R**PWRY)
    XC(NX,J) = S2/2.
    YC(NX,J) = YC(NX,1) + (YC(NX,NY) - YC(NX,1))*(R**PWRY2)
50 CONTINUE
  ELSE
C   STAGGERED GRID
  DO 60 I = 1, IX1
    THETA = ANG1*(1.-(1.-(FLOAT(I)-1)/(IX1-1))**PWRCYL)
    XC(I,IY1) = RAD*SIN(THETA)
    YC(I,IY1) = RAD*COS(THETA)
    XC(NX+1-I,IY2) = S2 - XC(I,IY1)
    YC(NX+1-I,IY2) = S1/2 - YC(I,IY1)
60 CONTINUE
  IF (NBLK.GT.0) THEN
    DO 70 J = 1, IY1
      THETA = ANG2*(1.-(1.-(FLOAT(J)-1)/(IY1-1))**PWRCYL)
      XC(IX1,J) = RAD*COS(THETA)
      YC(IX1,J) = RAD*SIN(THETA)
      XC(IX2,NY+1-J) = S2 - XC(IX1,J)
      YC(IX2,NY+1-J) = S1/2 - YC(IX1,J)
70 CONTINUE
    DO 80 I = 1, IX1
      THETA = (FLOAT(I)-1.)/(IX1-1.)
      XC(I,1) = THETA*XC(IX1,1)
      YC(I,1) = 0.
      XC(NX+1-I,NY) = S2 - XC(I,1)
      YC(NX+1-I,NY) = S1/2 - YC(I,1)
80 CONTINUE
    DO 90 J = 1, IY1
      THETA = (FLOAT(J)-1.)/(IY1-1.)
      YC(1,J) = THETA*YC(1,IY1)
      XC(1,J) = 0.
      XC(NX,NY+1-J) = S2 - XC(1,J)
      YC(NX,NY+1-J) = S1/2 - YC(1,J)
      YC(NX,J) = YC(IX1,J)
      YC(1,NY+1-J) = S1/2 - YC(IX1,J)
90 CONTINUE
    CALL TRANSFINITE(XC,YC,1,IX1,1,IY1)
    CALL TRANSFINITE(XC,YC,IX2,NX,IY2,NY)
    END IF
    DO 100 I = IX1, NX
      R = FLOAT(I-IX1)/FLOAT(NX-IX1)
      XC(I,1) = XC(IX1,1) + (S2 - XC(IX1,1))*(R**PWRWAKE)
      YC(I,1) = 0.
      XC(NX+1-I,IY2) = S2 - XC(I,1)
      YC(NX+1-I,IY2) = S1/2 - YC(I,1)
100 CONTINUE
    IF (NBLK.GT.0) THEN
      DO 110 I = IX1, NX
        R = FLOAT(I-IX1)/FLOAT(NX-IX1)
        XC(I,IY1) = XC(IX1,IY1) + (S2 - XC(IX1,IY1))*(R**PWRWAKE)
        YC(I,IY1) = YC(IX1,IY1)
        XC(NX+1-I,IY2) = S2 - XC(I,IY1)
```

```
YC(NX+1-I,IY2) = S1/2 - YC(I,IY1)
110 CONTINUE
CALL TRANSFINITE(XC,YC,IX1,NX,1,IY1)
CALL TRANSFINITE(XC,YC,1,IX2,IY2,NY)
END IF
DO 120 J = IY1, (IY2-IY1)/2+IY1
R = FLOAT(J-IY1)/FLOAT((IY2-IY1)/2)
XC(1,J) = 0.
YC(1,J) = YC(1,IY1) + .5*(YC(1,IY2)-YC(1,IY1))*(R**PWRY)
XC(NX,NY+1-J) = S2 - XC(1,J)
YC(NX,NY+1-J) = S1/2. - YC(1,J)
120 CONTINUE
DO 125 J = (IY2-IY1)/2+IY1, IY2
R = FLOAT(J-(IY2-IY1)/2-IY1)/FLOAT(IY2-(IY2-IY1)/2-IY1)
XC(1,J) = 0.
YC(1,J) = YC(1,IY2) - .5*(YC(1,IY2)-YC(1,IY1))*((1.-R)**PWRY)
XC(NX,NY+1-J) = S2 - XC(1,J)
YC(NX,NY+1-J) = S1/2. - YC(1,J)
125 CONTINUE
END IF
CALL TRANSFINITE(XC,YC,1,NX,IY1,IY2)
CALL DRAW_GRID
RETURN
END
C
SUBROUTINE READ_GRID
C THIS SUBROUTINE READ A GRID IN PHOENICS FORMAT
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX, NSPEC
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL XM(2,NYMAX,NXMAX), YM(2,NYMAX,NXMAX), ZM(2,NYMAX,NXMAX)
LOGICAL FLAG
INTEGER I, J, K
INTEGER NX, NY, NZ
INTEGER NXM_PLUS1, NYM_PLUS1, NZM_PLUS1
COMMON/COORDS/XC, YC
COMMON/DIMENS/ NX, NY
DATA FLAG/.FALSE./
100 FORMAT(3I5)
200 FORMAT(5(1PE13.6))
IF (.NOT.FLAG) THEN
OPEN(1,FILE='GRID',FORM='FORMATTED',STATUS='UNKNOWN',
1ACCESS='SEQUENTIAL')
WRITE (6,*) ' DATA READ FROM INPUT FILE GRID'
READ (1,100) NXM_PLUS1, NYM_PLUS1, NZM_PLUS1
DO 10 K = 1, NZM_PLUS1
READ (1, 200) ((XM(I, J, K), J = 1, NYM_PLUS1), I = 1, NXM_PLUS1)
READ (1, 200) ((YM(I, J, K), J = 1, NYM_PLUS1), I = 1, NXM_PLUS1)
READ (1, 200) ((ZM(I, J, K), J = 1, NYM_PLUS1), I = 1, NXM_PLUS1)
10 CONTINUE
CLOSE(1,STATUS='KEEP')
FLAG = .TRUE.
END IF
C
C CONVERT DATA TO A 2-D X-Y ARRAY
```

```
C
  NX = NZM_PLUS1
  NY = NYM_PLUS1
  DO 20 J = 1, NY
  DO 20 I = 1, NX
  XC(I,J) = ZM(1, J, I)
  YC(I,J) = YM(1, J, I)
20 CONTINUE
  RETURN
  END

C
  SUBROUTINE WRITE_GRID
C   THIS SUBROUTINE WRITES A GRID IN PHOENICS FORMAT
C
  IMPLICIT NONE
  INTEGER NXMAX, NYMAX, NSPEC
  PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
  REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
  REAL XM(2,NYMAX,NXMAX), YM(2,NYMAX,NXMAX), ZM(2,NYMAX,NXMAX)
  INTEGER I, J, K
  INTEGER NX, NY, NZ
  INTEGER NXM_PLUS1, NYM_PLUS1, NZM_PLUS1
  COMMON/COORDS/XC, YC
  COMMON/DIMENS/NX, NY
100 FORMAT(3I5)
200 FORMAT(5(1PE13.6))
C
C   CONVERT DATA TO A 3-D Y-Z ARRAY
C
  NXM_PLUS1=2
  NYM_PLUS1=NY
  NZM_PLUS1=NX
  DO 10 J = 1, NY
  DO 10 I = 1, NX
  XM(1, J, I) = 0.
  YM(1, J, I) = YC(I,J)
  ZM(1, J, I) = XC(I,J)
  XM(2, J, I) = 1.
  YM(2, J, I) = YC(I,J)
  ZM(2, J, I) = XC(I,J)
10 CONTINUE
C
  OPEN(2,FILE='XYZ',FORM='FORMATTED',STATUS='UNKNOWN',
1ACCESS='SEQUENTIAL')
  WRITE (2,100) NXM_PLUS1, NYM_PLUS1, NZM_PLUS1
  DO 20 K = 1, NZM_PLUS1
  WRITE (2, 200) ((XM(I, J, K), J = 1, NYM_PLUS1), I = 1, NXM_PLUS1)
  WRITE (2, 200) ((YM(I, J, K), J = 1, NYM_PLUS1), I = 1, NXM_PLUS1)
  WRITE (2, 200) ((ZM(I, J, K), J = 1, NYM_PLUS1), I = 1, NXM_PLUS1)
20 CONTINUE
  CLOSE(2,STATUS='KEEP')
  WRITE (6,*) ' DATA WRITTEN TO OUTPUT FILE XYZ'
  RETURN
  END

C
  SUBROUTINE SOLVER(LITER,RELAXS)
C   THIS SUBROUTINE SOLVES THE PAIR OF LINEAR ALGEBRAIC EQUATIONS
```

```
C   FOR THE CARTESIAN COORDINATES
C
  IMPLICIT NONE
  INTEGER NXMAX, NYMAX, NSPEC
  PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
  REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
  REAL AMASK(NXMAX,NYMAX)
  REAL AE(NXMAX,NYMAX), AN(NXMAX,NYMAX)
  REAL PHI(NXMAX), A(NXMAX), B(NXMAX), C(NXMAX), D(NXMAX)
  REAL SX(NXMAX,NYMAX), SY(NXMAX,NYMAX)
  REAL P(NXMAX,NYMAX), Q(NXMAX,NYMAX)
  REAL PP(NXMAX,NYMAX), QQ(NXMAX,NYMAX)
  REAL PV(NXMAX,NYMAX), QV(NXMAX,NYMAX)
  REAL SXX, SYY, RELAXS
  LOGICAL LXPLUS, LXMINUS, LYPLUS, LYMINUS
  INTEGER I, J, K
  INTEGER NX, NY, NZ
  INTEGER LITER, ISWEEP, LSWEEP
  COMMON/DIMENS/ NX, NY
  COMMON/COORDS/XC, YC
  COMMON/BITMAP/AMASK
  COMMON/LINCO/AE, AN
  COMMON/SORS/P, Q, SX, SY, PP, QQ, PV, QV
  DATA LXPLUS, LXMINUS, LYPLUS, LYMINUS/
  $.TRUE.,.FALSE.,.TRUE.,.FALSE./
C
C   SOLVER INNER LOOP(S)
C
C   XC COORDINATES
C
  DO 900 K = 1, LITER
    IF (LXPLUS) THEN
      DO 20 I = 2, NX-1
        DO 10 J = 2, NY-1
          A(J) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
          B(J) = AN(I,J)
          C(J) = AN(I,J-1)
          D(J) = AE(I,J)*XC(I+1,J) + AE(I-1,J)*XC(I-1,J)
          $+ SX(I,J) + P(I,J) + PV(I,J)
10    CONTINUE
          A(1) = 1.
          B(1) = 0.
          C(1) = 0.
          D(1) = XC(I,1)
          A(NY) = 1.
          B(NY) = 0.
          C(NY) = 0.
          D(NY) = XC(I,NY)
          CALL TDMA(PHI, A, B, C, D, NY)
          DO 20 J = 2, NY-1
            XC(I,J) = (1.-AMASK(I,J))*XC(I,J) + AMASK(I,J)*PHI(J)
20    CONTINUE
          END IF
C
          IF (LXMINUS) THEN
            DO 40 I = NX-1, 2, -1
              DO 30 J = 2, NY-1
```



```
A(J) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
B(J) = AN(I,J)
C(J) = AN(I,J-1)
D(J) = AE(I,J)*XC(I+1,J) + AE(I-1,J)*XC(I-1,J)
$+ SX(I,J) + P(I,J) + PV(I,J)
30 CONTINUE
  A(1) = 1.
  B(1) = 0.
  C(1) = 0.
  D(1) = XC(I,1)
  A(NY) = 1.
  B(NY) = 0.
  C(NY) = 0.
  D(NY) = XC(I,NY)
  CALL TDMA(PHI, A, B, C, D, NY)
  DO 40 J = 2, NY-1
    XC(I,J) = (1.-AMASK(I,J))*XC(I,J) + AMASK(I,J)*PHI(J)
40 CONTINUE
  END IF
C
  IF (LYPLUS) THEN
    DO 60 J = 2, NY-1
    DO 50 I = 2, NX-1
      A(I) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
      B(I) = AE(I,J)
      C(I) = AE(I-1,J)
      D(I) = AN(I,J)*XC(I,J+1) + AN(I,J-1)*XC(I,J-1)
      $+ SX(I,J) + P(I,J) + PV(I,J)
50 CONTINUE
      A(1) = 1.
      B(1) = 0.
      C(1) = 0.
      D(1) = XC(1,J)
      A(NX) = 1.
      B(NX) = 0.
      C(NX) = 0.
      D(NX) = XC(NX,J)
      CALL TDMA(PHI, A, B, C, D, NX)
      DO 60 I = 2, NX-1
        XC(I,J) = (1.-AMASK(I,J))*XC(I,J) + AMASK(I,J)*PHI(I)
60 CONTINUE
      END IF
C
      IF (LYMINUS) THEN
        DO 80 J = NY-1, 2, -1
        DO 70 I = 2, NX-1
          A(I) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
          B(I) = AE(I,J)
          C(I) = AE(I-1,J)
          D(I) = AN(I,J)*XC(I,J+1) + AN(I,J-1)*XC(I,J-1)
          $+ SX(I,J) + P(I,J) + PV(I,J)
70 CONTINUE
          A(1) = 1.
          B(1) = 0.
          C(1) = 0.
          D(1) = XC(1,J)
          A(NX) = 1.
```

```
B(NX) = 0.
C(NX) = 0.
D(NX) = XC(NX,J)
CALL TDMA(PHI, A, B, C, D, NX)
DO 80 I = 2, NX-1
  XC(I,J) = (1.-AMASK(I,J))*XC(I,J) + AMASK(I,J)*PHI(I)
80 CONTINUE
  END IF
C
C  YC COORDINATES
C
  IF (LXPLUS) THEN
    DO 100 I = 2, NX-1
      DO 90 J = 2, NY-1
        A(J) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
        B(J) = AN(I,J)
        C(J) = AN(I,J-1)
        D(J) = AE(I,J)*YC(I+1,J) + AE(I-1,J)*YC(I-1,J)
        $+ SY(I,J) + Q(I,J) + QV(I,J)
      90 CONTINUE
        A(1) = 1.
        B(1) = 0.
        C(1) = 0.
        D(1) = YC(I,1)
        A(NY) = 1.
        B(NY) = 0.
        C(NY) = 0.
        D(NY) = YC(I,NY)
        CALL TDMA(PHI, A, B, C, D, NY)
        DO 100 J = 2, NY-1
          YC(I,J) = (1.-AMASK(I,J))*YC(I,J) + AMASK(I,J)*PHI(J)
        100 CONTINUE
      END IF
    C
    IF (LXMINUS) THEN
      DO 120 I = NX-1, 2, -1
        DO 110 J = 2, NY-1
          A(J) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
          B(J) = AN(I,J)
          C(J) = AN(I,J-1)
          D(J) = AE(I,J)*YC(I+1,J) + AE(I-1,J)*YC(I-1,J)
          $+ SY(I,J) + Q(I,J) + QV(I,J)
        110 CONTINUE
          A(1) = 1.
          B(1) = 0.
          C(1) = 0.
          D(1) = YC(I,1)
          A(NY) = 1.
          B(NY) = 0.
          C(NY) = 0.
          D(NY) = YC(I,NY)
          CALL TDMA(PHI, A, B, C, D, NY)
          DO 120 J = 2, NY-1
            YC(I,J) = (1.-AMASK(I,J))*YC(I,J) + AMASK(I,J)*PHI(J)
          120 CONTINUE
        END IF
      C
```

```
      IF (LYPLUS) THEN
      DO 140 J = 2, NY-1
      DO 130 I = 2, NX-1
      A(I) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
      B(I) = AE(I,J)
      C(I) = AE(I-1,J)
      D(I) = AN(I,J)*YC(I,J+1) + AN(I,J-1)*YC(I,J-1)
      $+ SY(I,J) + Q(I,J) + QV(I,J)
130  CONTINUE
      A(1) = 1.
      B(1) = 0.
      C(1) = 0.
      D(1) = YC(1,J)
      A(NX) = 1.
      B(NX) = 0.
      C(NX) = 0.
      D(NX) = YC(NX,J)
      CALL TDMA(PHI, A, B, C, D, NX)
      DO 140 I = 2, NX-1
      YC(I,J) = (1.-AMASK(I,J))*YC(I,J) + AMASK(I,J)*PHI(I)
140  CONTINUE
      END IF
C
      IF (LYMINUS) THEN
      DO 160 J = NY-1, 2, -1
      DO 150 I = 2, NX-1
      A(I) = AE(I,J)+AE(I-1,J)+AN(I,J)+AN(I,J-1)
      B(I) = AE(I,J)
      C(I) = AE(I-1,J)
      D(I) = AN(I,J)*YC(I,J+1) + AN(I,J-1)*YC(I,J-1)
      $+ SY(I,J) + Q(I,J) + QV(I,J)
150  CONTINUE
      A(1) = 1.
      B(1) = 0.
      C(1) = 0.
      D(1) = YC(1,J)
      A(NX) = 1.
      B(NX) = 0.
      C(NX) = 0.
      D(NX) = YC(NX,J)
      CALL TDMA(PHI, A, B, C, D, NX)
      DO 160 I = 2, NX-1
      YC(I,J) = (1.-AMASK(I,J))*YC(I,J) + AMASK(I,J)*PHI(I)
160  CONTINUE
      END IF
C
900  CONTINUE
      RETURN
      END
C
      SUBROUTINE MASK (IXF,IXL,IYF,IYL,VAL)
C   THIS SUBROUTINE MASKS A SUB-REGION OF THE DOMAIN WITH A SINGLE VALUE
C
      IMPLICIT NONE
      INTEGER NXMAX, NYMAX, NSPEC
      PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
      REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX), AMASK(NXMAX, NYMAX)
```

```
REAL VAL
INTEGER IXF, IXL, IYF, IYL
INTEGER I, J, K
INTEGER NX, NY, NZ
COMMON/COORDS/XC, YC
COMMON/BITMAP/AMASK
DO 10 I = IXF, IXL
DO 10 J = IYF, IYL
AMASK(I,J) = VAL
10 CONTINUE
RETURN
END
C
SUBROUTINE SLIDER(RELAX,IXF,IXL,IYF,IYL)
C THIS SUBROUTINE SLIDES VALUES AT VARIOUS BOUNDARIES
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX, NSPEC
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL SLOPE0, SLOPE1, SLOPEC, SLOPEL
REAL RCYL, R, THETA
REAL RELAX
REAL DIAM, S, S1, S2
REAL Y, X1, X2, Y1, RR, SPPT
INTEGER IXF, IXL, IYF, IYL
INTEGER IX1, IX2, IY1, IY2, NBLK
INTEGER I, J, K
INTEGER NX, NY, NZ
LOGICAL LSLIDE, LSLIDW, LSLIDS, LSLIDN,LOFF,LON
LOGICAL STAGGERED
C
COMMON/DIMENS/ NX, NY
COMMON/COORDS/XC, YC
COMMON/DOMAIN/DIAM,S,S1,S2, SPPT, STAGGERED, NBLK
C
DATA LSLIDE, LSLIDW, LSLIDS, LSLIDN, LON, LOFF
$/TRUE,.,TRUE,.,TRUE,.,TRUE,.,FALSE,.,TRUE./
C
SLOPE0(X1) = X1
SLOPE1(X1,X2) = 1.5*X1-.5*X2
SLOPEC(X1,X2) = SQRT(X1**2 + Y1**2)
SLOPEL(RR,X1,Y1,X2) = RR*(1.5*X1-.5*X2)+(1.-RR)*SQRT(X1**2+Y1**2)
C
IF (.NOT.STAGGERED) THEN
IX1 = INT(SPPT*(NX-1)*3.*DIAM/(2.*S2+DIAM))
ELSE
IX1 = INT(SPPT*(NX-1)*3.*DIAM/(4.*S2+2.*DIAM))
END IF
IX2 = NX+1 - IX1
C
C SLIDE BOUNDARY VALUES
C
IF (LSLIDW) THEN
DO 61 J = IYF+1, IYL-1
C WEST
RR = FLOAT(J-IYF)/FLOAT(IYL-IYF)
```

```
YC(1,J) = (1.-RELAX)*YC(1,J) +
$RELAX*SLOPEL(RR,YC(2,J),XC(2,J),YC(3,J))
61 CONTINUE
END IF
C
C EAST
C
IF (LSLIDE) THEN
DO 62 J = IYF+1,IYL-1
IF (STAGGERED) THEN
YC(NX,IYL-J+IYF) = S1/2.-YC(1,J)
END IF
62 CONTINUE
END IF
C
C SOUTH
C
IF (LSLIDS) THEN
DO 63 I = 1, NX-1
C ON-CYLINDER
IF (LON) THEN
IF (I.LT.IX1-2) THEN
THETA = ATAN(YC(I,IXF+1)/XC(I,IXF+1))
XC(I,IXF) = (1.-RELAX)*XC(I,IXF) + RELAX*DIAM*COS(THETA)/2.
YC(I,IXF) = (1.-RELAX)*YC(I,IXF) + RELAX*DIAM*SIN(THETA)/2.
END IF
END IF
C OFF-CYLINDER
IF (LOFF) THEN
IF (I.GT.IX1) THEN
RR = FLOAT(I-IX1)/FLOAT(NX-IX1)
XC(I,1) = (1.-RELAX)*XC(I,1) +
$RELAX*SLOPEL(RR,XC(I,2),YC(I,2),XC(I,3))
END IF
END IF
63 CONTINUE
END IF
C
C NORTH
C
IF (LSLIDN) THEN
DO 64 I = 1, NX-1
C ON-CYLINDER
IF (LON) THEN
IF (I.GT.IX2+2) THEN
THETA = ATAN((S1/2.-YC(I,IXL-1))/(S2-XC(I,IXL-1)))
XC(I,IXL) = (1.-RELAX)*XC(I,IXL)+
$RELAX*(S2 - DIAM*COS(THETA)/2.)
YC(I,IXL) = (1.-RELAX)*YC(I,IXL) +
$RELAX*(S1/2. - DIAM*SIN(THETA)/2.)
C OFF-CYLINDER
END IF
END IF
IF (LOFF) THEN
IF (I.LT.IX2) THEN
IF (STAGGERED) THEN
XC(I,NY) = S2 - XC(NX+1-I,1)
```

```
ELSE
XC(I,NY) = SLOPE0(XC(I,NY-1))
END IF
END IF
END IF
64 CONTINUE
END IF
RETURN
END
C
SUBROUTINE CLEAR_PARAMS
C THIS SUBROUTINE RESETS A NUMBER OF VARIABLES
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX, NSPEC
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL G11E(NXMAX,NYMAX), G11N(NXMAX,NYMAX)
REAL G22E(NXMAX,NYMAX), G22N(NXMAX,NYMAX)
REAL G12E(NXMAX,NYMAX), G12N(NXMAX,NYMAX)
REAL GPSQRT(NXMAX,NYMAX)
REAL AE(NXMAX,NYMAX), AN(NXMAX,NYMAX)
REAL SX(NXMAX,NYMAX), SY(NXMAX,NYMAX)
REAL P(NXMAX,NYMAX), Q(NXMAX,NYMAX)
REAL PP(NXMAX,NYMAX), QQ(NXMAX,NYMAX)
REAL PV(NXMAX,NYMAX), QV(NXMAX,NYMAX)
INTEGER NX, NY
INTEGER I, J
COMMON/DIMENS/ NX, NY
COMMON/METRICS/G11E, G22E, G12E, G11N, G22N, G12N, GPSQRT
COMMON/SORS/P, Q, SX, SY, PP, QQ, PV, QV
DO 10 I = 1, NX
DO 10 J = 1, NY
G11E(I,J) = 0.
G22E(I,J) = 0.
G12E(I,J) = 0.
G11N(I,J) = 0.
G22N(I,J) = 0.
G12N(I,J) = 0.
GPSQRT(I,J) = 0.
P(I,J) = 0.
Q(I,J) = 0.
PP(I,J) = 0.
SX(I,J) = 0.
SY(I,J) = 0.
PP(I,J) = 0.
QQ(I,J) = 0.
AE(I,J) = 0.
AN(I,J) = 0.
10 CONTINUE
RETURN
END
C
SUBROUTINE MIRROR(IXF,IXL,IYF,IYL)
C THIS SUBROUTINE MIRRORS VALUES FOR STAGGERED GRIDS
C
IMPLICIT NONE
INTEGER NXMAX, NYMAX, NSPEC
```

```
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL DIAM, S, S1, S2
REAL Y, X1, X2, Y1, RR, SPPT
LOGICAL STAGGERED
INTEGER IXF, IXL, IYF, IYL
INTEGER IX1, IX2, IY1, IY2, NBLK
INTEGER I, J, K
INTEGER NX, NY, NZ
C
COMMON/DIMENS/ NX, NY
COMMON/COORDS/XC, YC
COMMON/DOMAIN/DIAM,S,S1,S2, SPPT, STAGGERED, NBLK
DO 10 I = 1, (NX-1)/2
DO 10 J = 1, NY-1
YC(IXL-I+IXF,IYL-J+IYF) = S1/2.-YC(I,J)
XC(IXL-I+IXF,IYL-J+IYF) = S2-XC(I,J)
10 CONTINUE
RETURN
END
```

```
C
C GRAPHICS SUBROUTINES
C
SUBROUTINE OPEN_GRAPHICS
IMPLICIT NONE
INCLUDE "/usr/include/gl/fgl.h"
INTEGER NXMAX, NYMAX, NSPEC
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL DIAM, S1, S2
INTEGER WSID, NORMWIN, CONID, WSTYPE
INTEGER NX, NY
COMMON/DIMENS/ NX, NY
COMMON/COORDS/XC, YC
COMMON/DOMAIN/DIAM,S1,S2
INTEGER NWINDOW
COMMON /MYGL/ NWINDOW
INTEGER IXPIX, IYPIX, IXEDGE, IYEDGE
REAL SCALE, XEDGE, YEDGE, REDGE
DATA WSID,NORMWIN,CONID,WSTYPE/ 1,1,0,1/
REDGE = 0.1
IXPIX = 1280
IYPIX = 1024
SCALE = AMIN1 (IXPIX/S2, IYPIX/S1)
IXEDGE = S2 * SCALE * REDGE
IYEDGE = S1 * SCALE * REDGE
XEDGE = INT(S2 * REDGE) + 0.5
YEDGE = INT(S1 * REDGE) + 0.5
CALL PREFPO (IXEDGE, IXPIX - IXEDGE,
$IYEDGE, IYPIX - IYEDGE)
NWINDOW = WINOPE ('GRID2D', 6)
CALL ORTHO2 (-XEDGE, INT(S2 + 0.5) + XEDGE,
$-YEDGE, INT(S1 + 0.5) + YEDGE)
CALL DOUBLEBUFFER()
CALL GCONFI()
RETURN
```

END

C

```
SUBROUTINE CLOSE_GRAPHICS
IMPLICIT NONE
INCLUDE "/usr/include/gl/fgl.h"
INTEGER WSID
INTEGER NWINDOW
COMMON /MYGL/ NWINDOW
DATA WSID/1/
CALL WINCLO (NWINDOW)
RETURN
END
```

C

```
SUBROUTINE DRAW_GRID
IMPLICIT NONE
INCLUDE "/usr/include/gl/fgl.h"
INTEGER NXMAX, NYMAX, NSPEC, NXORY
PARAMETER(NXORY=102)
PARAMETER(NXMAX=102,NYMAX=102,NSPEC=2)
REAL XC(NXMAX,NYMAX), YC(NXMAX,NYMAX)
REAL VECTOR(2)
INTEGER WSID, NORMWIN
INTEGER NX, NY
INTEGER I, J
COMMON/DIMENS/ NX, NY
COMMON/COORDS/XC, YC
DATA WSID,NORMWIN/ 1,1 /
CALL COLOR(BLACK)
CALL CLEAR()
CALL COLOR(RED)
DO 20 J=1,NY
CALL BGNLIN()
DO 10 I=1,NX
VECTOR(1) = XC(I,J)
VECTOR(2) = YC(I,J)
CALL V2F(VECTOR)
10 CONTINUE
CALL ENDLIN()
20 CONTINUE
DO 40 I=1,NX
CALL BGNLIN()
DO 30 J=1,NY
VECTOR(1) = XC(I,J)
VECTOR(2) = YC(I,J)
CALL V2F(VECTOR)
30 CONTINUE
CALL ENDLIN()
40 CONTINUE
CALL SWAPBU()
RETURN
END
```


A3.3 Stream Functions

2-D Q1 FILE

TALK=F;RUN(1, 1);VDU=X11-TERM

GROUP 1. Run title and other preliminaries

TEXT(STREAM)

REAL(RE)

BOOLEAN(INLINE)

INLINE=F

GROUP 2. TRANSCIENCE TIME-STEP SPECIFICATION

STEADY=T

GROUP 3. X-DIRECTION GRID SPECIFICATION

NX = 80

GRDPWR(X,NX,1,1.)

GROUP 4. Y-DIRECTION GRID SPECIFICATION

NY = 161

GRDPWR(Y,NY,1,1.)

GROUP 5. Z-DIRECTION GRID SPECIFICATION

NZ = 1

BFC=T;NONORT=F;XCYIZ(1,NZ,F)

READCO(GRID)

GROUP 7. VARIABLES STORED, SOLVED & NAMED

STORE(P1,U1,V1,H1)

STORE(S1,SOR1)

SOLUTN(S1,Y,Y,Y,N,N,N)

GROUP 8. TERMS (IN DIFFERENTIAL EQUATIONS) & DEVICES

TERMS(S1,N,N,Y,N,Y,N)

GROUP 9. PROPERTIES OF THE MEDIUM (OR MEDIA)

RE = 300

ENUL = 1./RE;RHO1=1.0;PRNDTL(S1) = ENUL*RHO1

GROUP 10. INTER-PHASE-TRANSFER PROCESSES AND PROPERTIES

GROUP 11. INITIALIZATION OF VARIABLE OR POROSITY FIELDS

IF (INLINE) THEN

CONPOR(0.0,CELL, 39, 42, 1, 26, 1, NZ)

CONPOR(0.0,CELL, 39, 42, 55,106, 1, NZ)

CONPOR(0.0,CELL, 39, 42,135,186, 1, NZ)

ELSE

CONPOR(0.0,CELL, 1, 2, 1, 24, 1, NZ)

CONPOR(0.0,CELL, 79, 82, 1, 24, 1, NZ)

CONPOR(0.0,CELL, 39, 42, 17, 64, 1, NZ)

CONPOR(0.0,CELL, 1, 2, 57,104, 1, NZ)

CONPOR(0.0,CELL, 79, 82, 57,104, 1, NZ)

CONPOR(0.0,CELL, 39, 42, 97,144, 1, NZ)

CONPOR(0.0,CELL, 1, 2,137,184, 1, NZ)

CONPOR(0.0,CELL, 79, 82,137,184, 1, NZ)

ENDIF

FIINIT(SOR1)=0.

FIINIT(S1)= 0.

PATCH(DUM,CELL,1,NX,1,NY,1,NZ,1,LSTEP)

PATCH(FIXS,CELL,NX/2,NX/2,NY/4,NY/4,1,1,1,LSTEP)

COVAL(FIXS,S1,FIXVAL,0.)

PATCH(ALL,CELL,1,NX,1,NY,1,NZ,1,LSTEP)

COVAL(ALL,S1,FIXFLU,GRND5)

GROUP 15. Termination of sweeps
GROUP 16. Termination of iterations
LSWEEP = 200
ENDIT(S1)=0.
LITER(S1) = 20
RESREF(S1)=1.e-12
GROUP 17. Under relaxation devices
GROUP 18. Limits on variables or increments to them
GROUP 19. Data communicated by satellite to GROUND
USEGRX=F
GROUP 20. Preliminary print-out
GROUP 21. Print-out of variables
OUTPUT(P1,N,N,N,N,N,N)
OUTPUT(U1,N,N,N,N,N,N)
OUTPUT(V1,N,N,N,N,N,N)
OUTPUT(H1,N,N,N,N,N,N)
OUTPUT(UCRT,N,N,N,N,N,N)
OUTPUT(VCRT,N,N,N,N,N,N)
OUTPUT(EPOR,N,N,N,N,N,N)
OUTPUT(S1,Y,N,N,Y,Y,Y)
OUTPUT(SOR1,N,N,N,N,N,N)
GROUP 22. Spot-value print-out
NPRMON=LSWEEP;IXMON=NX/4;IYMON=(NY-1)/2;TSTSWP=LSWEEP/10
GROUP 23. Field print-out and plot control
GROUP 24. Dumps for restarts
RESTRT(P1,U1,V1,H1,UCRT,VCRT)
STOP

2-D GROUND FILE

C
C 2 User dimensions own arrays here:
PARAMETER (NXD = 80, NYD = 161, NZD = 1,NBUF = 500)
COMMON/COMPAS/IMYAE,IMYAW,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
C*****
C
C--- GROUP 1. Run title and other preliminaries
C
1 GO TO (1001,1002),ISC
1001 CONTINUE
C
C User may here change message transmitted to the VDU screen or
C batch-run log file.
IF(IGR.EQ.1.AND.ISC.EQ.1) THEN
CALL WRYT40('GROUND file is GROUND.FTN of: 200390 ')
CALL WRYT40('PHOENICS version number is : 1.5.3 ')
ENDIF
C
IMYAE = 1
IMYAW = 1
IMYAS = 1
IMYAN = 1
IMYAL = 0
IMYAH = 0
PI = 4.*ATAN(1.)
DIAM = 1.

```

CALL MAKE(EASP1)
CALL MAKE(EASP2)
CALL MAKE(EASP3)
CALL MAKE(EASP4)
CALL MAKE(EASP5)
CALL MAKE(EASP6)
CALL MAKE(EASP7)
CALL MAKE(EASP8)
CALL MAKE(EASP9)
CALL MAKE(EASP10)
CALL MAKE(DYG2D)
CALL MAKE(DYV2D)
CALL MAKE(DXG2D)
CALL MAKE(DXU2D)
CALL MAKE(AEAST)
CALL MAKE(ANORTH)
RETURN
1002 CONTINUE
RETURN
C*****
C
C--- GROUP 13. Boundary conditions and special sources
C          Index for Coefficient - CO
C          Index for Value    - VAL
  13 CONTINUE
    GO TO (130,131,132,133,134,135,136,137,138,139,1310,
    11311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321),ISC
  130 CONTINUE
C----- SECTION 1 ----- coefficient = GRND
  RETURN
  131 CONTINUE
C----- SECTION 2 ----- coefficient = GRND1
  RETURN
  132 CONTINUE
C----- SECTION 3 ----- coefficient = GRND2
  RETURN
  133 CONTINUE
C----- SECTION 4 ----- coefficient = GRND3
  RETURN
  134 CONTINUE
C----- SECTION 5 ----- coefficient = GRND4
  RETURN
  135 CONTINUE
C----- SECTION 6 ----- coefficient = GRND5
  RETURN
  136 CONTINUE
C----- SECTION 7 ----- coefficient = GRND6
  RETURN
  137 CONTINUE
C----- SECTION 8 ----- coefficient = GRND7
  RETURN
  138 CONTINUE
C----- SECTION 9 ----- coefficient = GRND8
  RETURN
  139 CONTINUE
C----- SECTION 10 ----- coefficient = GRND9
  RETURN

```

```

1310 CONTINUE
C----- SECTION 11 ----- coefficient = GRND10
      RETURN
1311 CONTINUE
C----- SECTION 12 ----- value = GRND
      RETURN
1312 CONTINUE
C----- SECTION 13 ----- value = GRND1
      RETURN
1313 CONTINUE
C----- SECTION 14 ----- value = GRND2
      RETURN
1314 CONTINUE
C----- SECTION 15 ----- value = GRND3
      RETURN
1315 CONTINUE
C----- SECTION 16 ----- value = GRND4
      RETURN
1316 CONTINUE
C----- SECTION 17 ----- value = GRND5
C   Get 2-D source term
C
      NSOR1 = INAME('SOR1')
      CALL FN0(VAL,NSOR1)
      RETURN
1317 CONTINUE
C----- SECTION 18 ----- value = GRND6
      RETURN
1318 CONTINUE
C----- SECTION 19 ----- value = GRND7
      RETURN
1319 CONTINUE
C----- SECTION 20 ----- value = GRND8
      RETURN
1320 CONTINUE
C----- SECTION 21 ----- value = GRND9
      RETURN
1321 CONTINUE
C----- SECTION 22 ----- value = GRND10
      RETURN
C*****
C
C--- GROUP 14. Downstream pressure for PARAB=.TRUE.
C
      14 CONTINUE
      RETURN
C*****
C
C--- GROUP 19. Special calls to GROUND from EARTH
C
      19 GO TO (191,192,193,194,195,196,197,198),ISC
      191 CONTINUE
C * ----- SECTION 1 ---- Start of time step.
      RETURN
      192 CONTINUE
C * ----- SECTION 2 ---- Start of sweep.
      RETURN

```

```
193 CONTINUE
C * ----- SECTION 3 ---- Start of iz slab.
  RETURN
194 CONTINUE
C * ----- SECTION 4 ---- Start of iteration.
  RETURN
195 CONTINUE
C * ----- SECTION 5 ---- Finish of iteration.
  RETURN
196 CONTINUE
C * ----- SECTION 6 ---- Finish of iz slab.

C   Calculate source term for 2-D stream function.
  IF (ISWEEP.EQ.1) THEN
    NS1 = INAME('S1')
    IF (SOLVE(NS1)) CALL FXS2D(NY,NX,BFC)
    END IF
  RETURN
197 CONTINUE
C * ----- SECTION 7 ---- Finish of sweep.
c
c
  RETURN
198 CONTINUE
C * ----- SECTION 8 ---- Finish of time step.

  RETURN
C*****
```

2-D and 3-D utilities

```
      SUBROUTINE FXS2D(NY,NX,BFC)
C
C   THIS SUBROUTINE COMPUTES THE SOURCE TERM FOR THE
C   2-D STREAM FUNCTION X-Y PLANE ONLY. CARTESIAN OR BFC GEOMETRY.
C
C   INCLUDE 'lp16/d_earth/GRDLOC'
C   INCLUDE 'lp16/d_earth/GRDEAR'
C   INCLUDE 'lp16/d_earth/GRDBFC'
C   COMMON/NAMFN/NAMFUN,NAMSUB
C   COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
C   INTEGER EPOR,HPOR, VPOR
C   LOGICAL BFC
C   CHARACTER*6 NAMFUN,NAMSUB
C
C   SOR1   SOURCE TERM FOR S1
C   EASP1   UCMP = U-VELOCITY COMPONENT
C   EASP2   VCMP = V-VELOCITY COMPONENT
C   EASP3   TXX = X-TANGENT VECTOR X-DIRECTION CARTESIAN COMPONENT
C   EASP4   TXY = X-TANGENT VECTOR Y-DIRECTION CARTESIAN COMPONENT
C   EASP5   TYX = Y-TANGENT VECTOR X-DIRECTION CARTESIAN COMPONENT
C   EASP6   TYY = Y-TANGENT VECTOR Y-DIRECTION CARTESIAN COMPONENT
C   EASP7   X CONTRIBUTION TO SOR1
C   EASP8   Y CONTRIBUTION TO SOR1
C   EASP9   TEMPORARY STORAGE
C   EASP10  TEMPORARY STORAGE
C
```

```
NAMSUB = 'FXS2D'
NSOR1 = INAME('SOR1')
LOAE = LOF(AEAST)
LOAN = LOF(ANORTH)
LOSOR1 = LOF(NSOR1)
LOEASP7 = LOF(EASP7)
LOEASP8 = LOF(EASP8)
NPOR = INAME('NPOR')
EPOR = INAME('EPOR')
VPOR = INAME('VPOR')
C
C 1. COMPUTE V
C RESULT IS STORED IN EASP1, EASP2
C
CALL FXVCMP2D(EASP1,EASP2,NY,NX,BFC)
C
C 2. X AND Y - DIRECTION SOURCE TERMS:
C RESULT IS STORED IN EASP7, EASP8
IF (.NOT.BFC) THEN
CALL FN0(EASP7,EASP2)
CALL FN25(EASP7,-1.)
CALL FN0(EASP8,EASP1)
IF(STORE(VPOR)) THEN
CALL FN26(EASP7,VPOR)
CALL FN26(EASP8,VPOR)
END IF
ELSE
C
C IN BFC'S NEED PERPENDICULAR VELOCITIES:
C X DIRECTION:
C RESULTS ARE STORED TEMPORARILY IN EASP3 AND EASP4 SUCH THAT
C  $TX = (PE + PW)/2 = TI*I + TJ*J = EASP3*I + EASP4*J$ 
C NORMAL DIRECTION IS JUST,
C  $NX = NI*I + NJ*J = EASP4*I - EASP3*J$ 
C
C Y DIRECTION:
C RESULTS ARE STORED TEMPORARILY IN EASP5 AND EASP6 SUCH THAT
C  $TY = (PN + PS)/2 = TI*I + TJ*J = EASP5*I + EASP6*J$ 
C NORMAL DIRECTION IS JUST,
C  $NY = NI*I + NJ*J = EASP6*I - EASP5*J$ 
C
CALL FNGBFC(EASP9,UNIVEE,IZ)
CALL FNGBFC(EASP10,UNIVEN,IZ)
CALL FNAV(EASP3,EASP9,'WEST')
CALL FNAV(EASP4,EASP10,'WEST')
CALL FN48(EASP9,EASP3,EASP3,EASP4,EASP4,1.,1.)
CALL FN30(EASP9)
CALL FN27(EASP3,EASP9)
CALL FN27(EASP4,EASP9)
CALL FNGBFC(EASP9,UNIVNE,IZ)
CALL FNGBFC(EASP10,UNIVNN,IZ)
CALL FNAV(EASP5,EASP9,'SOUTH')
CALL FNAV(EASP6,EASP10,'SOUTH')
CALL FN48(EASP9,EASP5,EASP5,EASP6,EASP6,1.,1.)
CALL FN30(EASP9)
CALL FN27(EASP5,EASP9)
CALL FN27(EASP6,EASP9)
```

```
C
C  X DIR. SOURCE = VCMP.NX IS STORED IN EASP7
C  Y DIR. SOURCE = UCMP.NY IS STORED IN EASP8
C
  CALL FN48(EASP7,EASP4,EASP5,EASP3,EASP6,1.,-1.)
  CALL FN48(EASP8,EASP6,EASP3,EASP5,EASP4,1.,-1.)
  CALL FN26(EASP7,EASP2)
  CALL FN26(EASP8,EASP1)
  IF(STORE(VPOR)) THEN
    CALL FN26(EASP7,VPOR)
    CALL FN26(EASP8,VPOR)
  END IF
  END IF
C
C  3. COMPUTE SOURCE TERMS FOR CELL P
C  RESULT IS STORED IN SOR1.
C
  DO 70 I = 1, NX
    DO 70 J = 1, NY
      IP = J + NY*(I-1)
      IW = J + NY*(I-2)
      IE = J + NY*I
      IS = J-1 + NY*(I-1)
      IN = J+1 + NY*(I-1)
C
      IF ((I.NE.1).AND.(IMYAW.NE.0))
        $SW = .5*(F(L0EASP7 + IP) + F(L0EASP7 + IW))
        SW = SW*F(L0AE + IW)
        IF ((I.NE.NX).AND.(IMYAE.NE.0))
          $SE = -.5*(F(L0EASP7 + IP) + F(L0EASP7 + IE))
          SE = SE*F(L0AE + IP)
          IF ((J.NE.1).AND.(IMYAS.NE.0))
            $SS = .5*(F(L0EASP8 + IP) + F(L0EASP8 + IS))
            SS = SS*F(L0AN + IS)
            IF ((J.NE.NY).AND.(IMYAN.NE.0))
              $SN = -.5*(F(L0EASP8 + IP) + F(L0EASP8 + IN))
              SN = SN*F(L0AN + IP)
C
          F(L0SOR1 + IP) = SW + SE + SS + SN
C
C  UNCOMMENT ONLY IF WALL VALUES OF STREAM FUNCTION
C  ARE PRESCRIBED IN Q1:
C
C  WEST
C  IF (I.EQ.1)
C  $   F(L0SOR1 + IP) = F(L0SOR1 + IP)
C  $   + .5*F(L0EASP7 + J + IP)*F(L0AE + IP)
C  EAST
C  IF (I.EQ.NX)
C  $   F(L0SOR1 + IP) = F(L0SOR1 + IP)
C  $   - .5*F(L0EASP7 + IP)*F(L0AE + IW)
C  SOUTH
C  IF (J.EQ.1)
C  $   F(L0SOR1 + IP) = F(L0SOR1 + IP)
C  $   + .5*F(L0EASP8 + IP)*F(L0AN + IP)
C  NORTH
C  IF (J.EQ.NY)
```

```
C  $  F(L0SOR1 + IP) = F(L0SOR1 + IP)
C  $  - .5*F(L0EASP8 + IP)*F(L0AN + IS)
C
70  CONTINUE
    RETURN
    END
C
SUBROUTINE FXVCMP2D(NUC,NVC,NY,NX,BFC)
C
C  THIS SUBROUTINE COMPUTES 2-D VECTOR COMPONENTS
C  LOCATED AT CELL CENTRES
C
C  NUC    UCMP = U-VELOCITY COMPONENT
C  NVC    VCMP = V-VELOCITY COMPONENT
C  EASP3   TXX = X-TANGENT VECTOR X-DIRECTION CARTESIAN COMPONENT
C  EASP4   TXY = X-TANGENT VECTOR Y-DIRECTION CARTESIAN COMPONENT
C  EASP5   TYX = Y-TANGENT VECTOR X-DIRECTION CARTESIAN COMPONENT
C  EASP6   TYY = Y-TANGENT VECTOR Y-DIRECTION CARTESIAN COMPONENT
C  EASP7   COSINE OF ANGLE TX.TY
C
INCLUDE 'lp16/d_earth/GRDLOC'
INCLUDE 'lp16/d_earth/GRDEAR'
INCLUDE 'lp16/d_earth/GRDBFC'
COMMON/NAMFN/NAMFUN,NAMSUB
COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
INTEGER EPOR,HPOR, VPOR
INTEGER UCRT, VCRT, WCRT
CHARACTER*6 NAMFUN,NAMSUB
LOGICAL BFC
NAMESUB = 'FXVCMP2D'
L0UC = L0F(NUC)
L0VC = L0F(NVC)
L0EASP3 = L0F(EASP3)
L0EASP4 = L0F(EASP4)
L0EASP5 = L0F(EASP5)
L0EASP6 = L0F(EASP6)
IF (STORE(U1)) L0U1 = L0F(U1)
IF (STORE(V1)) L0V1 = L0F(V1)
C
C  1. GET RESOLUTES
C
CALL FN0(NUC,U1)
CALL FN0(NVC,V1)
C
C  2. GET GEOMETRY AND DESTAGGER
C
CALL FNGBFC(EASP9,UNIVEE,IZ)
CALL FNGBFC(EASP10,UNIVEN,IZ)
CALL FNAV(EASP3,EASP9,'WEST')
CALL FNAV(EASP4,EASP10,'WEST')
CALL FN48(EASP9,EASP3,EASP3,EASP4,EASP4,1.,1.)
CALL FN30(EASP9)
CALL FN27(EASP3,EASP9)
CALL FN27(EASP4,EASP9)
CALL FNGBFC(EASP9,UNIVNE,IZ)
CALL FNGBFC(EASP10,UNIVNN,IZ)
CALL FNAV(EASP5,EASP9,'SOUTH')
```



```

CALL FNAV(EASP6,EASP10,'SOUTH')
CALL FN48(EASP9,EASP5,EASP5,EASP6,EASP6,1.,1.)
CALL FN30(EASP9)
CALL FN27(EASP5,EASP9)
CALL FN27(EASP6,EASP9)
C
C 3. EXTRAPOLATE EDGE VALUES
C
  IF (NX.GT.1) THEN
    DO 20 J = 1, NY
      IP = J + NY*(NX-1)
      IW = J + NY*(NX-2)
      F(NUC+IP) = F(NUC+IW)
20  CONTINUE
    END IF
    IF (NY.GT.1) THEN
      DO 10 I = 1, NX
        IP = NY + NY*(I-1)
        IS = NY - 1 + NY*(I-1)
        F(NVC+IP) = F(NVC+IS)
10  CONTINUE
      END IF
C
C 4. DESTAGGER RESOLUTE FIELD.
C
CALL FXDSTG(NUC,NVC,EASP9,EASP9,NZ,NY,NX)
C
C 5. CONVERT TO COMPONENTS
C
CALL FN48(EASP7,EASP3,EASP5,EASP4,EASP6,1.,1.)
CALL FN53(NUC,EASP7,NVC,-1.)
CALL FN53(NVC,EASP7,NUC,-1.)
CALL FN50(EASP7,2)
CALL FN25(EASP7,-1.)
CALL FN33(EASP7,1.)
CALL FN27(NUC,EASP7)
CALL FN27(NVC,EASP7)
RETURN
END
C
SUBROUTINE FXS3D(NZ,NY,NX,NPSI0,NZGRD,CONST,BFC)
C
C THIS SUBROUTINE COMPUTES THE SOURCE TERM FOR THE 2-D AND 3-D
C STREAM FUNCTIONS IN BOTH ORTHOGONAL BFC AND CARTESIAN GEOMETRY.
C
  INCLUDE 'lp16/d_earth/GRDLOC'
  INCLUDE 'lp16/d_earth/GRDEAR'
  COMMON/NAMFN/NAMFUN,NAMSUB
  COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
  INTEGER EPOR,HPOR,VPOR
  LOGICAL BFC
  CHARACTER*6 NAMFUN,NAMSUB
  NAMSUB = 'FXS3D'
  NS1 = INAME('S1')
  NS2 = INAME('S2')
  NSOR1 = INAME('SOR1')
  NSOR2 = INAME('SOR2')

```

```
L0ZGRD = L0F(NZGRD)
IF (IZ.GT.1) L0LZGRD = L0F(LOW(NZGRD))
IF (IZ.LT.NZ) L0HZGRD = L0F(HIGH(NZGRD))
IF (NX.GT.1) L0DXG2D = L0F(DXG2D)
IF (NY.GT.1) L0DYG2D = L0F(DYG2D)
IF (NX.GT.1) L0DXU2D = L0F(DXU2D)
IF (NY.GT.1) L0DYV2D = L0F(DYV2D)
L0AE = L0F(AEAST)
L0AN = L0F(ANORTH)
L0AH = L0F(AHIGH)
IF (IZ.GT.1) L0LAH = L0F(AHIGH)
L0EASP1 = L0F(EASP1)
L0EASP2 = L0F(EASP2)
L0EASP3 = L0F(EASP3)
L0EASP4 = L0F(EASP4)
L0EASP5 = L0F(EASP5)
L0EASP6 = L0F(EASP6)
L0EASP7 = L0F(EASP7)
L0EASP8 = L0F(EASP8)
L0EASP9 = L0F(EASP9)
L0PSI0 = L0F(NPSI0)
L0SOR1 = L0F(NSOR1)
L0SOR2 = L0F(NSOR2)
NPOR = INAME('NPOR')
HPOR = INAME('HPOR')
EPOR = INAME('EPOR')
VPOR = INAME('VPOR')
IF (STORE(EPOR)) L0EPOR = L0F(EPOR)
IF (STORE(NPOR)) L0NPOR = L0F(NPOR)
IF (STORE(HPOR)) L0HPOR = L0F(HPOR)
IF (STORE(VPOR)) L0VPOR = L0F(VPOR)
IF (IZ.GT.1) LPSI0 = L0F(LOW(NPSI0))
IF (STORE(HPOR).AND.(IZ.GT.1))
$LOLHPOR = L0F(LOW(HPOR))
IF (STORE(VPOR).AND.(IZ.GT.1))
$LOLVPOR = L0F(LOW(VPOR))
IF (STORE(VPOR).AND.(IZ.LT.NZ))
$LOHVPOR = L0F(HIGH(VPOR))
C
C 1. COMPUTE GRAD(PSI0)
C RESULT IS STORED IN EASP1, EASP2, EASP3
C
CALL FXGRAD3D(NPSI0,EASP1,EASP2,EASP3,EASP7,NZ,NY,NX,BFC)
C
C DE-STAGGER GRAD(PSI0)
c CALL FXDSTG(EASP1,EASP2,EASP3,EASP7,NZ,NY,NX)
C
C 2. COMPUTE NORMAL VELOCITY COMPONENTS
C RESULT IS STORED IN EASP4, EASP5, EASP6
C
CALL FXVCMPE(EASP4,EASP5,EASP6,EASP7,NZ,NY,NX,BFC)
C
C DE-STAGGER VELOCITY FIELD
c CALL FXDSTG(EASP4,EASP5,EASP6,EASP7,NZ,NY,NX)
C --> -->
C 3. COMPUTE DEL(PSI)/DR = {GRAD(PSI0) X V}/|GRAD(PSI0)|**2
C
```

```

C   A. COMPUTE GRAD(PSI0) X V
C   RESULT IS STORED IN EASP7, EASP8, EASP9
C
C   CALL FXCROSS3D(EASP1,EASP2,EASP3,EASP4,EASP5,EASP6,
$EASP7,EASP8,EASP9,NZ,NY,NX)
C
C   B. DIVIDE BY IGRAD(PSI0)**2
C   IGRAD(PSI0)**2 IS STORED IN EASP6
C   RESULT IS STORED IN EASP7, EASP8, EASP9
C
C   CALL FN1(EASP6,0.)
C   CALL FN49(EASP6,EASP1)
C   CALL FN49(EASP6,EASP2)
C   CALL FN49(EASP6,EASP3)
C   CALL FN27(EASP7,EASP6)
C   CALL FN27(EASP8,EASP6)
C   CALL FN27(EASP9,EASP6)
C
C   IF(STORE(VPOR)) THEN
C   CALL FN26(EASP7,VPOR)
C   CALL FN26(EASP8,VPOR)
C   CALL FN26(EASP9,VPOR)
C   END IF
C
C   STORE Z DIR VALUE IN FIELD VARIABLE NZGRD
C   CALL FN0(NZGRD,EASP9)
C
C   4. COMPUTE SOURCE TERMS FOR CELL P
C   RESULT IS STORED IN SOR1 OR SOR2 AS APPROPRIATE.
C
C   NB: WEST, SOUTH, AND LOW SOURCES ARE POSITIVE,
C   EAST, NORTH, AND HIGH ARE NEGATIVE.
C
C   INCLUDE 'sorwnd.f'
C   RETURN
C   END
C
C
C   SUBROUTINE FXGRAD3D(NPSI,NDPSIX,NDPSIY,NDPSIZ,NLDPSIZ,NZ,NY,
$NX,BFC)
C
C   THIS FUNCTION COMPUTES GRAD(PSI) ON A SLAB-WISE BASIS
C
C   INCLUDE 'lp16/d_earth/GRDLOC'
C   INCLUDE 'lp16/d_earth/GRDEAR'
C   INCLUDE 'lp16/d_earth/GRDBFC'
C   COMMON/NAMFN/NAMFUN,NAMSUB
C   COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
C   INTEGER EPOR,HPOR,VPOR
C   LOGICAL BFC
C   CHARACTER*6 NAMFUN,NAMSUB
C   NAMSUB = 'FXGRAD3D'
C   IF (NX.GT.1) L0DXG2D = L0F(DXG2D)
C   IF (NY.GT.1) L0DYG2D = L0F(DYG2D)
C   L0AE = L0F(AEAST)
C   L0AN = L0F(ANORTH)
C   L0PSI = L0F(NPSI)

```

```

L0DPSIX = L0F(NDPSIX)
L0DPSIY = L0F(NDPSIY)
NPOR = INAME('NPOR')
HPOR = INAME('HPOR')
EPOR = INAME('EPOR')
IF (STORE(EPOR)) L0EPOR = L0F(EPOR)
IF (STORE(NPOR)) L0NPOR = L0F(NPOR)
IF (STORE(HPOR)) L0HPOR = L0F(HPOR)
C
C  CHECK FOR 2-D FLOW
C
  IF (NX.LE.1) THEN
    CALL FN1(NDPSIX,1.)
    CALL FN1(NDPSIY,0.)
    CALL FN1(NDPSIZ,0.)
    RETURN
  ELSE IF (NY.LE.1) THEN
    CALL FN1(NDPSIX,0.)
    CALL FN1(NDPSIY,1.)
    CALL FN1(NDPSIZ,0.)
    RETURN
  ELSE IF (NZ.LE.1) THEN
    CALL FN1(NDPSIX,0.)
    CALL FN1(NDPSIY,0.)
    CALL FN1(NDPSIZ,1.)
    RETURN
  END IF
C
C  PROBLEM IS 3D
C
C  A. X DIRECTION.
C
  IZL = IZ-1
  DO 10 I = 1, NX
    DO 10 J = 1, NY
      IP = J + NY*(I-1)
      IW = J + NY*(I-2)
      IE = J + NY*I
      IS = J - 1 + NY*(I-1)
      IN = J + 1 + NY*(I-1)
      IF (I.LT.NX) THEN
        IF (STORE(EPOR)) THEN
          F(L0DPSIX + IP) =
            $(F(L0PSI + IE) - F(L0PSI + IP))*F(L0EPOR + IP)
          IF (ABS(F(L0DPSIX + IP)).GT.1.E-08)
            $F(L0DPSIX + IP) = F(L0DPSIX + IP)/
            $(F(L0DXG2D + IP)*F(L0EPOR + IP))
        ELSE
          F(L0DPSIX + IP) =
            $(F(L0PSI + IE) - F(L0PSI + IP))/
            $F(L0DXG2D + IP)
        END IF
      ELSE
        IF (STORE(EPOR)) THEN
          F(L0DPSIX + IP) =
            $(F(L0PSI + IP) - F(L0PSI + IW))*F(L0EPOR + IW)
          IF (ABS(F(L0DPSIX + IP)).GT.1.E-08)

```

```
$F(L0DPSIX + IP) = F(L0DPSIX + IP)/
$(F(L0DXG2D + IW)*F(L0EPOR + IW))
ELSE
F(L0DPSIX + IP) =
$(F(L0PSI + IP) - F(L0PSI + IW))/
$(F(L0DXG2D + IW))
END IF
END IF
C
C   B. Y DIRECTION.
C
IF (J.LT.NY) THEN
IF (STORE(NPOR)) THEN
F(L0DPSIY + IP) =
$(F(L0PSI + IN) - F(L0PSI + IP))*F(L0NPOR + IP)
IF (ABS(F(L0DPSIY + IP)),GT.1.E-08)
$F(L0DPSIY + IP) = F(L0DPSIY + IP)/
$(F(L0DYG2D + IP)*F(L0NPOR + IP))
ELSE
F(L0DPSIY + IP) =
$(F(L0PSI + IN) - F(L0PSI + IP))/
$(F(L0DYG2D + IP))
END IF
ELSE
IF (STORE(NPOR)) THEN
F(L0DPSIY + IP) =
$(F(L0PSI + IP) - F(L0PSI + IS))*F(L0NPOR + IS)
IF (ABS(F(L0DPSIY + IP)),GT.1.E-08)
$F(L0DPSIY + IP) = F(L0DPSIY + IP)/
$(F(L0DYG2D + IS)*F(L0NPOR + IS))
ELSE
F(L0DPSIY + IP) =
$(F(L0PSI + IP) - F(L0PSI + IS))/
$(F(L0DYG2D + IS))
END IF
END IF
10 CONTINUE
C
C   C. Z DIRECTION:
C
IF (.NOT.BFC) THEN
IF (IZ.LT.NZ) THEN
CALL FN10(NDPSIZ,HIGH(NPSI),NPSI,0.,1./DZG,-1./DZG)
IF (STORE(HPOR)) CALL FN26(NDPSIZ,HPOR)
ELSE
CALL FN10(NDPSIZ,NPSI,LOW(NPSI),0.,1./DZGL,-1./DZGL)
IF (STORE(HPOR)) CALL FN26(NDPSIZ,LOW(HPOR))
END IF
IF (IZ.GT.1) THEN
CALL FN10(NLDPSIZ,NPSI,LOW(NPSI),0.,1./DZGL,-1./DZGL)
IF (STORE(HPOR)) CALL FN26(NLDPSIZ,LOW(HPOR))
ELSE
CALL FN10(NLDPSIZ,HIGH(NPSI),NPSI,0.,1./DZG,-1./DZG)
IF (STORE(HPOR)) CALL FN26(NLDPSIZ,HPOR)
END IF
ELSE
IF (IZ.LT.NZ) THEN
```

```
CALL FN10(NDPSIZ,HIGH(NPSI),NPSI,0.,1.,-1.)
CALL FNGBFC(EASP4,DZGPH,IZ)
CALL FN27(NDPSIZ,EASP4)
IF (STORE(HPOR)) CALL FN26(NDPSIZ,HPOR)
ELSE
CALL FN10(NDPSIZ,NPSI,LOW(NPSI),0.,1.,-1.)
CALL FNGBFC(EASP4,DZGPH,IZL)
CALL FN27(NDPSIZ,EASP4)
IF (STORE(HPOR)) CALL FN26(NDPSIZ,LOW(HPOR))
END IF
IF (IZ.GT.1) THEN
CALL FN10(NLDPSIZ,NPSI,LOW(NPSI),0.,1.,-1.)
CALL FNGBFC(EASP4,DZGPH,IZL)
CALL FN27(NLDPSIZ,EASP4)
IF (STORE(HPOR)) CALL FN26(NLDPSIZ,LOW(HPOR))
ELSE
CALL FN10(NLDPSIZ,HIGH(NPSI),NPSI,0.,1.,-1.)
CALL FNGBFC(EASP4,DZGPH,IZ)
CALL FN27(NLDPSIZ,EASP4)
IF (STORE(HPOR)) CALL FN26(NLDPSIZ,HPOR)
END IF
END IF
RETURN
END
```

C

```
SUBROUTINE FXVCMP(NUC,NVC,NWC,NLWC,NZ,NY,NX,BFC)
```

C

C THIS SUBROUTINE COMPUTES A CELL CENTRED VELOCITY VECTOR

C

```
INCLUDE 'lp16/d_earth/GRDLOC'
INCLUDE 'lp16/d_earth/GRDEAR'
COMMON/NAMFN/NAMFUN,NAMSUB
COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
INTEGER EPOR,HPOR, VPOR
INTEGER UCRT, VCRT, WCRT
CHARACTER*6 NAMFUN,NAMSUB
LOGICAL BFC
NAMSUB = 'FXVCMP'
LOUC = LOF(NUC)
LOVC = LOF(NVC)
LOWC = LOF(NWC)
IF (STORE(U1)) LOU1 = LOF(U1)
IF (STORE(V1)) LOV1 = LOF(V1)
```

C

```
CALL FN1(NUC,0.)
CALL FN1(NVC,0.)
CALL FN1(NWC,0.)
```

C

C TANGENTIAL RESOLUTES ARE SIMPLE TRIPLETS

C

```
IF (NX.GT.1) THEN
CALL FN0(NUC,U1)
DO 20 J = 1, NY
IP = J + NY*(NX-1)
IW = J + NY*(NX-2)
F(NUC+IP) = F(NUC+IW)
```

20 CONTINUE

```

END IF
IF (NY.GT.1) THEN
CALL FN0(NVC,V1)
DO 10 I = 1, NX
IP = NY + NY*(I-1)
IS = NY - 1 + NY*(I-1)
F(NVC+IP) = F(NVC+IS)
10 CONTINUE
END IF
IF (NZ.GT.1) THEN
IF (IZ.EQ.1) THEN
CALL FN0(NWC,W1)
CALL FN0(NLWC,W1)
ELSE IF (IZ.LT.NZ) THEN
CALL FN0(NWC,W1)
CALL FN0(NLWC,LOW(W1))
ELSE
CALL FN0(NWC,LOW(W1))
CALL FN0(NLWC,LOW(W1))
END IF
END IF
RETURN
END
C
SUBROUTINE FXVNRM(IVNZ,IVNY,IVNX,IVTZ,IVTY,IVTX,NZ,NY,NX,BFC)
C
C THIS SUBROUTINE COMPUTES NORMAL RESOLUTES FROM TANGENTIAL VALUES.
C (NB: THE RESOLUTES ARE PER UNIT AREA NOT PER UNIT PROJECTED AREA)
C
C EASP11 IS REQUIRED FOR TEMPORARY STORAGE
C
INCLUDE 'lp16/d_earth/GRDLOC'
INCLUDE 'lp16/d_earth/GRDEAR'
INCLUDE 'lp16/d_earth/GRDBFC'
COMMON/NAMFN/NAMFUN,NAMSUB
COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
INTEGER EPOR,HPOR, VPOR
INTEGER UCRT, VCRT, WCRT
CHARACTER*6 NAMFUN,NAMSUB
LOGICAL BFC
C
NAMSUB = 'FXNRM'
C
CALL FN1(NX,0.)
IF (NX.GT.1) THEN
CALL FNGBFC(EASP11,MFCQEE,IZ)
CALL FN53(IVNX,EASP11,U1)
IF (NY.GT.1) CALL FNGBFC(EASP11,MFCQEN,IZ)
IF (NY.GT.1) CALL FN53(IVNY,EASP11,U1)
IF (NZ.GT.1) CALL FNGBFC(EASP11,MFCSHN,IZ)
IF (NZ.GT.1) CALL FN53(IVNZ,EASP11,U1)
END IF
IF (NY.GT.1) THEN
IF (NX.GT.1) CALL FNGBFC(EASP11,MFCRNE,IZ)
IF (NX.GT.1) CALL FN53(IVNX,EASP11,V1)
CALL FNGBFC(EASP11,MFCRNN,IZ)
CALL FN53(IVNY,EASP11,V1)

```

```
IF (NZ.GT.1) CALL FNGBFC(EASP11,MFCRNH,IZ)
IF (NZ.GT.1) CALL FN53(IVNZ,EASP11,W1)
END IF
IF (NZ.GT.1) THEN
IF (NX.GT.1) CALL FNGBFC(EASP11,MFCSHE,IZ)
IF (NX.GT.1) CALL FN53(IVNX,EASP11,W1)
IF (NY.GT.1) CALL FNGBFC(EASP11,MFCSHN,IZ)
IF (NY.GT.1) CALL FN53(IVNY,EASP11,W1)
CALL FNGBFC(EASP11,MFCSHH,IZ)
CALL FN53(IVNZ,EASP11,W1)
END IF
```

c

```
IF (NX.GT.1) THEN
CALL FNGBFC(EASP11,APROJE,IZ)
CALL FN22(EASP11,1.E-20)
CALL FN27(IVNX,EASP11)
CALL FNGBFC(EASP11,ASURFE,IZ)
CALL FN26(IVNX,EASP11)
END IF
IF (NY.GT.1) THEN
CALL FNGBFC(EASP11,APROJN,IZ)
CALL FN22(EASP11,1.E-20)
CALL FN27(IVNY,EASP11)
CALL FNGBFC(EASP11,ASURFN,IZ)
CALL FN26(IVNY,EASP11)
END IF
IF (NZ.GT.1) THEN
CALL FNGBFC(EASP11,APROJH,IZ)
CALL FN22(EASP11,1.E-20)
CALL FN27(IVNZ,EASP11)
CALL FNGBFC(EASP11,ASURFH,IZ)
CALL FN26(IVNZ,EASP11)
END IF
RETURN
END
```

C

```
SUBROUTINE FXCROSS3D(NAX,NAY,NAZ,NBX,NBY,NBZ,NCX,NCY,NCZ,NZ,
$NY,NX)
```

C

C THIS FUNCTION COMPUTES $C = AXB$ ON A SLAB-WISE BASIS

C

```
INCLUDE 'lp16/d_earth/GRDLOC'
INCLUDE 'lp16/d_earth/GRDEAR'
INCLUDE 'lp16/d_earth/GRDBFC'
COMMON/NAMFN/NAMFUN,NAMSUB
INTEGER EPOR,HPOR,VPOR
CHARACTER*6 NAMFUN,NAMSUB
```

C

```
NAMSUB = 'FXCROSS3D'
```

C

```
VPOR = INAME('VPOR')
```

C

```
L0AX = L0F(NAX)
L0AY = L0F(NAY)
L0AZ = L0F(NAZ)
L0BX = L0F(NBX)
L0BY = L0F(NBY)
L0BZ = L0F(NBZ)
```



```
L0CX = L0F(NCX)
L0CY = L0F(NCY)
L0CZ = L0F(NCZ)
DO 60 I = 1, NX
DO 60 J = 1, NY
IP = J + NY*(I-1)
F(L0CX + IP) =
$F(L0AY + IP)*F(L0BZ + IP) -
$F(L0AZ + IP)*F(L0BY + IP)
F(L0CY + IP) =
$F(L0AZ + IP)*F(L0BX + IP) -
$F(L0AX + IP)*F(L0BZ + IP)
F(L0CZ + IP) =
$F(L0AX + IP)*F(L0BY + IP) -
$F(L0AY + IP)*F(L0BX + IP)
60 CONTINUE
RETURN
END
C
c
SUBROUTINE FXDSTG(NUC, NVC, NWC, NLWC, NZ, NY, NX)
C
C THIS SUBROUTINE CONVERTS A SET OF STAGGERED QUNTITIES
C TO CELL-AVERAGED VALUES. CARTESIAN OR BFC GEOMETRY.
C
INCLUDE 'lp16/d_earth/GRDLOC'
COMMON/NAMFN/NAMFUN,NAMSUB
COMMON/IGE/IXF,IXL,IYF,IYL,IREG,NZSTEP,IGR,ISC,IRUN,
$IZSTEP,ITHYD,ISWEEP,ISTEP,INDVAR,VAL,CO,NDIREC,WALDIS,
$PATGEO,IGES20(6)
COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH,UREF,VREF,WREF
INTEGER VAL,CO,WALDIS,PATGEO,EPOR,HPOR
INTEGER VPOR
INTEGER UCRT, VCRT, WCRT
CHARACTER*6 NAMFUN,NAMSUB
LOGICAL BFC
C
NAMSUB = 'FXDSTG7'
C
L0UC = L0F(NUC)
L0VC = L0F(NVC)
L0WC = L0F(NWC)
L0LWC = L0F(NLWC)
C
DO 10 I = NX-1, 2, -1
DO 10 J = 1, NY
IP = J + NY*(I-1)
IW = J + NY*(I-2)
F(L0UC + IP) = (F(L0UC + IP) + F(L0UC + IW))/2.
10 CONTINUE
C
DO 20 J = NY-1, 2, -1
DO 20 I = 1, NX
IP = J + NY*(I-1)
IS = J-1 + NY*(I-1)
F(L0VC + IP) = (F(L0VC + IP) + F(L0VC + IS))/2.
20 CONTINUE
```

```

C
C  Z-DIRECTION IS NOT DE-STAGGERED
C
C  IF (NZ.LE.1) RETURN
C  CALL FN34(NWC,NLWC,1.)
C  CALL FN25(NWC,0.5)
  RETURN
  END
C
  SUBROUTINE FXSTCO(NZ,NY,NX,NPH)
C
C  THIS SUBROUTINE COMPUTES THE DIFFUSION COEFFICIENTS FOR THE 3-D
C  STREAM-FUNCTION IN BOTH BFC AND CARTESIAN GEOMETRY.
C
  INCLUDE 'lp16/d_earth/GRDLOC'
  COMMON/IGE/IXF,IXL,IYF,IYL,IREG,NZSTEP,IGR,ISC,IRUN,
  1IZSTEP,ITHYD,ISWEEP,ISTEP,INDVAR,VAL,CO,NDIREC,WALDIS,
  1PATGEO,IGES20(6)
  COMMON/COMPAS/IMYAW,IMYAE,IMYAS,IMYAN,IMYAL,IMYAH
  INTEGER VAL,CO,WALDIS,PATGEO,EPOR,HPOR
  COMMON/NAMFN/NAMFUN,NAMSUB
  CHARACTER*6 NAMFUN,NAMSUB
  LOGICAL LDBG, LFLG
C
  NPOR = INAME('NPOR')
  HPOR = INAME('HPOR')
  EPOR = INAME('EPOR')
  NS2 = INAME('S2')
  NAMSUB = 'FXSTCO'
  LDBG = .FALSE.
  LFLG = .TRUE.
C  IF (NS2.EQ.NPH) LFLG = .TRUE.
  IF (NDIREC.EQ.1) THEN
C  NORTH:
  CALL FN1(LAN,0.)
  IF (IMYAN.EQ.1) THEN
  L0LAN = L0F(LAN)
  CALL FN1(LAN,1.)
  IF (STORE(NPOR).AND.LFLG)
  $CALL FN26(LAN,NPOR)
  DO 10 I = 1, NX
    F(L0LAN+NY+NY*(I-1)) = 0.
10  CONTINUE
  END IF
  IF (LDBG) CALL PRN('LAN ',LAN)
C  SOUTH:
  CALL FN1(LAS,0.)
  IF (IMYAS.EQ.1) THEN
  L0LAS = L0F(LAS)
  CALL FN1(LAS,1.)
  IF (STORE(NPOR).AND.LFLG)
  $CALL FN26(LAS,SOUTH(NPOR))
  DO 20 I = 1, NX
    F(L0LAS+1+NY*(I-1)) = 0.
20  CONTINUE
  END IF
  IF (LDBG) CALL PRN('LAS ',LAS)

```

```
      ELSE IF (NDIREC.EQ.3) THEN
C   EAST:
      CALL FN1(LAE,0.)
      IF (IMYAE.EQ.1) THEN
        L0LAE = L0F(LAE)
        CALL FN1(LAE,1.)
        IF (STORE(EPOR).AND.LFLG)
          $CALL FN26(LAE,EPOR)
        DO 30 J = 1, NY
          F(L0LAE+J+NY*(NX-1)) = 0.
30    CONTINUE
      IF (LDBG) CALL PRN('LAE ',LAE)
      END IF
C   WEST:
      CALL FN1(LAW,0.)
      IF (IMYAW.EQ.1) THEN
        L0LAW = L0F(LAW)
        CALL FN1(LAW,1.)
        IF (STORE(EPOR).AND.LFLG)
          $CALL FN26(LAW,WEST(EPOR))
        DO 40 J = 1, NY
          F(L0LAW+J) = 0.
40    CONTINUE
      IF (LDBG) CALL PRN('LAW ',LAW)
      END IF
      ELSE IF (NDIREC.EQ.5) THEN
        CALL FN1(LD11,0.)
C   HIGH:
      IF ((IMYAH.EQ.1).AND.(IZ.NE.NZ)) CALL FN1(LD11,1.)
      IF (STORE(HPOR).AND.LFLG)
        $CALL FN26(LD11,HPOR)
      IF (LDBG) CALL PRN('LAH ',LD11)
      ELSE IF (NDIREC.EQ.6) THEN
C   LOW:
      CALL FN1(LD11,0.)
      IF ((IMYAL.EQ.1).AND.(IZ.NE.1)) CALL FN1(LD11,1.)
      IF (STORE(HPOR).AND.LFLG)
        $CALL FN26(LD11,LOW(HPOR))
      IF (LDBG) CALL PRN('LAL ',LD11)
      END IF
      RETURN
      END
```

A3.4 Steady two-dimensional flow

Q1 File

```
TALK=F;RUN( 1, 1);VDU=X11-TERM
GROUP 1. Run title and other preliminaries
TEXT(STEADY FLOW)
```

```
** S      = PITCH
** S1     = CROSS-WISE PITCH
** S2     = STREAM-WISE PITCH
** DIAM   = CYLINDER DIAMETER
** W0     = BULK VELOCITY
** RE     = PRESCRIBED REYNOLDS' NUMBER
** AMYCO   = PRESSURE COEFFICIENT
** AMYRELAX = PRESSURE RELAXATION FACTOR (IN GROUND)
** RESCALE = SCALE FACTOR USED TO COMPUTE RE
** TWALL   = WALL TEMPERATURE (CONSTANT T)
** QWALL   = WALL HEAT FLUX (CONSTANT QW)
** KFLUID  = FLUID THERMAL CONDUCTIVITY
** CFLUID  = FLUID SPECIFIC HEAT
** ALPHA   = FLUID THERMAL DIFFUSIVITY
** PRAN    = FLUID PRANDTL NUMBER
** FALST   = FALSE TIME STEP RELAXATION FACTOR
** GEOFAC  = GEOMETRIC FACTOR USED IN GRID GENERATION
** LHGRID  = T FOR HGRID
** LIGRID  = T FOR IGRID
** ICONFIG = T FOR IN-LINE GRID CONFIGURATION
** LDCONFIG = T FOR DIAGONAL GRID CONFIGURATION
** LINLINE = T FOR IN-LINE (FLOW) F FOR STAGGERED (FLOW)
```

```
REAL(S,S1, S2, DIAM, W0, RE)
REAL(Y1,Z1,Z2)
REAL(TBIN, TWALL, QWALL, DELH, KFLUID, CFLUID, ALPHA, PRAN)
REAL(FALST,AMYCO,AMYRELAX,RESCALE)
REAL (GEOFAC)
INTEGER(NZP1,NZM1,IZ1,IZ2)
INTEGER(IYF1,IYF2,IYL1,IYL2)
BOOLEAN(LHGRID,LIGRID,LPOR,LINLINE)
BOOLEAN(ICONFIG,DCONFIG,LCONSTT)
LHGRID = T;LIGRID = F
ICONFIG=F;DCONFIG=T
LINLINE = F;LCONSTT=F
LPOR = F
DIAM = 1.
RESCALE = 1
```

```
IF (LINLINE) THEN
S=2.00 S1 = S;S2 = S
GEOFAC = 1.0 for in-line grids
ELSE
ROTATED SQUARE
S=2.0 S1 = S*SQRT(2);S2 = S/SQRT(2)
GEOFAC = 1.
GEOFAC NOT RELEVANT FOR S = 1.25 (I-GRID)
```

```
GEOFAC = 1.6 when S = 1.5
GEOFAC = 1.5 when S = 2.
IF (S.LT.1.707) THEN
  RESCALE = (1.414*S-DIAM)/(2*(S-DIAM))
ENDIF
EQUILATERAL TRIANGLE
S=1.25;S1 = S;S2 = S1*(3**.5)/2.
GEOFAC = 1.5 GEOFAC = 1.5 when S = 1.25
GEOFAC = 1.3 when S = 1.5
GEOFAC = 1.5 when S = 2.0
GEOFAC=1.5 ENDIF
IF (ICONFIG) THEN RESCALE = S1/(S1-DIAM)
ENDIF
TBIN = 0.
NX=1;NY = 30;NZ = 50
NZ = NZ + 1
RE = 100.
W0 = 1.
AMYCO=1.E+06;AMYRELAX = .3
GROUP 2. Transience time-step specification
STEADY=T
GROUP 3. X-direction grid specification
GROUP 4. Y-direction grid specification
GROUP 5. Z-direction grid specification
GROUP 6. Body-fitted coordinates or grid distortion
BFC=T;NONORT=T
READCO(GRID)
GROUP 7. Variables stored, solved & named
FALST = 1.E+01*ZWLAST/(NZ*W0) STORE(P1,V1,W1,H1,THTA)
SOLUTN(P1,Y,Y,Y,N,N,N)
SOLUTN(V1,Y,Y,N,Y,N,N)
SOLUTN(W1,Y,Y,N,Y,N,N)
SOLUTN(H1,Y,Y,N,N,N,N)
GROUP 8. Terms (in differential equations) & devices
TERMS(H1,N,P,P,P,P,P)
GROUP 9. Properties of the medium (or media)
PRAN = 1.
ENUL = RESCALE*W0/RE;RHO1=1.0;
KFLUID=RESCALE*W0/(RE*PRAN);CFLUID=1.
ALPHA=KFLUID/(RHO1*CFLUID)
PRNDTL(H1) = ENUL/ALPHA
TWALL = 1.;QWALL = 1.
GROUP 10. Inter-phase-transfer processes and properties
GROUP 11. Initialization of variable or porosity fields
FIINT(V1)=0.;FIINT(P1)=0.;
IF (LCONSTT) THEN
  FIINT(H1)=CFLUID*(TBIN+TWALL)/2.
ELSE
  DELH = 3.14*QWALL*DIAM/(RHO1*W0*(S1-DIAM))
  PATCH(HINIT,LINVZ,1,NX,1,NY,1,NZ,1,LSTEP)
  INIT(HINIT,H1,DELH/S2,CFLUID*TBIN)
ENDIF
IF (LIGRID.OR.ICONFIG) THEN
  FIINT(W1)=S1/(S1-DIAM)*W0
ELSE
  FIINT(W1)=W0
ENDIF
```

FIINIT(W1)=0.

IF (LHGRID) THEN

IF (ICONFIG) THEN

Z1 = GEOFAC*(NZ-1)*3.*DIAM/(2.*(2.*S2 + DIAM))

IZ1 = Z1

Z1 = (NZ-1)/2+1-IZ1

ELSE

Z1 = GEOFAC*(NZ-1)*3.*DIAM/(2.*(2.*S2 + DIAM))

ENDIF

IZ1 = Z1

Z1 = IZ1

Z2 = NZ-Z1

IZ2 = Z2

ELSE IF (LIGRID)

IF (DCONFIG) THEN

Z1 = NZ

Z2 = NZ-Z1

CONPOR(0,CELL,1,NX,1,NY/2,NZ+1,NZ+1)

ELSE

Y1 = 1.5*NY*(S2-DIAM)/(S2-DIAM+S1)

CONPOR(0,CELL,1,NX,1,Y1,NZ,NZ)

ENDIF

LPOR = T

ENDIF

GROUP 12. Convection and diffusion adjustments

GROUP 13. Boundary conditions and special source

**** WALL BOUNDARY CONDITIONS**

IF (LHGRID) THEN

IF ((.NOT.(ICONFIG)).AND.(.NOT.DCONFIG)) THEN

PATCH(S1,SWALL,1,NX,1,1,1,Z1,1,LSTEP)

PATCH(S2,SWALL,1,NX,1,1,1,Z2,NZ,1,LSTEP)

PATCH(SH1,SOUTH,1,NX,1,1,1,Z1,1,LSTEP)

PATCH(SH2,SOUTH,1,NX,1,1,1,Z2,NZ,1,LSTEP)

ENDIF

IF (ICONFIG) THEN

PATCH(S1,SWALL,1,NX,1,1,1,Z1,NZ/2,1,LSTEP)

PATCH(S2,SWALL,1,NX,1,1,1,NZ/2+1,Z2,1,LSTEP)

PATCH(SH1,SOUTH,1,NX,1,1,1,Z1,NZ/2,1,LSTEP)

PATCH(SH2,SOUTH,1,NX,1,1,1,NZ/2+1,Z2,1,LSTEP)

ENDIF

IF (DCONFIG) THEN

PATCH(S1,SWALL,1,NX,1,1,1,Z1,1,LSTEP)

PATCH(S2,NWALL,1,NX,NY,NY,Z2,NZ,1,LSTEP)

PATCH(SH1,SOUTH,1,NX,1,1,1,Z1,1,LSTEP)

PATCH(SH2,NORTH,1,NX,NY,NY,Z2,NZ,1,LSTEP)

ENDIF

```
COVAL(S1,W1,1.,0.0)
COVAL(S2,W1,1.,0.0)
```

```
IF (LCONSTT) THEN
COVAL(S1,H1,1/PRNDTL(H1),CFLUID*TWALL)
COVAL(S2,H1,1/PRNDTL(H1),CFLUID*TWALL)
ELSE
COVAL(SH1,H1,FIXFLU,QWALL)
COVAL(SH2,H1,FIXFLU,QWALL)
ENDIF
```

```
ELSE
```

```
(LIGRID)
```

```
PATCH(S1,SWALL,1,NX,1,1,1,NZ-1,1,LSTEP)
COVAL(S1,W1,1.,0.)
```

```
IF (LCONSTT) THEN
COVAL(S1,H1,1/PRNDTL(H1),CFLUID*TWALL)
ELSE
PATCH(SH1,SOUTH,1,NX,1,1,1,NZ-1,1,LSTEP)
COVAL(SH1,H1,FIXFLU,QWALL)
ENDIF
```

```
IF (DCONFIG) THEN
PATCH(S2,NWALL,1,NX,NY,NY,1,NZ,1,LSTEP)
COVAL(S2,W1,1.,0.0)
```

```
IF (LCONSTT) THEN
COVAL(S2,H1,1/PRNDTL(H1),CFLUID*TWALL)
ELSE
PATCH(SH2,NORTH,1,NX,NY,NY,1,NZ,1,LSTEP)
COVAL(SH2,H1,FIXFLU,QWALL)
ENDIF
```

```
ENDIF
```

```
ENDIF
```

```
** INLET/EXIT BOUNDARY CONDITIONS
```

```
IYF1 = Y1+1
IYF2 = NY
IYL1 = Y1+1
IYL2 = NY
```

```
IF (LIGRID.AND.DCONFIG) THEN
IYF1 = 1
IYF2 = NY/2
IYL1 = NY/2+1
IYL2 = NY
ENDIF
```

```
PATCH(UPSTRM,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)
IF (LIGRID) THEN
IF (.NOT.DCONFIG) THEN
```

```
PATCH(UPSTRV,LOW,1,NX,IYF1-1,IYF2-1,1,1,1,LSTEP)
PATCH(DWNSTRV,HIGH,1,NX,IYL1-1,IYL2-1,NZ,NZ,1,LSTEP)
ELSE
PATCH(UPSTRV,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRV,HIGH,1,NX,IYL1,IYL2-1,NZ,NZ,1,LSTEP)
ENDIF
```

```
ELSE IF (LHGRID)
PATCH(UPSTRV,LOW,1,NX,IYF1,IYF2-1,1,1,1,LSTEP)
PATCH(DWNSTRV,HIGH,1,NX,IYL1,IYL2-1,NZ,NZ,1,LSTEP)
ENDIF
```

```
PATCH(UPSTRW,HIGH,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRM,LOW,1,NX,IYL1,IYL2,NZ,NZ,1,LSTEP)
PATCH(DWNSTRW,HIGH,1,NX,IYL1,IYL2,NZ-1,NZ-1,1,LSTEP)
```

```
COVAL(UPSTRM,P1,FIXFLU,GRND)
COVAL(UPSTRM,V1,ONLYMS,SAME)
COVAL(UPSTRM,W1,ONLYMS,SAME)
COVAL(UPSTRM,H1,GRND,GRND)
COVAL(UPSTRV,V1,GRND,GRND)
COVAL(UPSTRW,W1,GRND,GRND)
```

```
COVAL(DWNSTRM,P1,GRND1,GRND4)
COVAL(DWNSTRM,V1,ONLYMS,SAME)
COVAL(DWNSTRM,W1,ONLYMS,SAME)
COVAL(DWNSTRM,H1,ONLYMS,SAME)
COVAL(DWNSTRV,V1,FIXVAL,GRND)
```

```
COVAL(DWNSTRV,H1,FIXVAL,GRND)
COVAL(DWNSTRW,W1,GRND,GRND)
```

```
PATCH(VDUMMY,PHASEM,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(NDUMMY,NORTH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(SDUMMY,SOUTH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(HDUMMY,HIGH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(LDUMMY,LOW,1,NX,1,NY,1,NZ,1,LSTEP)
```

GROUP 14. Downstream pressure for PARAB=.TRUE.

GROUP 15. Termination of sweeps

LSWEEP = 600

RESREF(P1) = 1.E-12

RESREF(V1) = 1.E-12

RESREF(W1) = 1.E-12

RESREF(H1) = 1.E-12

GROUP 16. Termination of iterations

LITER(P1) = 20

ENDIT(P1) = 0.;ENDIT(V1)=0.;ENDIT(W1)=0.

GROUP 17. Under-relaxation devices

RELAX(P1,LINRLX,.3)

RELAX(V1,FALSDT,FALST)

RELAX(W1,FALSDT,FALST)

RELAX(H1,FALSDT,FALST)

GROUP 18. Limits on variables or increments to them

VARMIN(W1) = -1.*W0; VARMAX(W1) = 5.*W0

VARMIN(V1) = -2.*W0; VARMAX(V1) = 2.*W0

VARMIN(P1) = -10.*RHO1*W0*W0

VARMAX(P1) = 10.*RHO1*W0*W0


```
VARMIN(H1) = -10.*CFLUID*TWALL;VARMAX(H1) = 10.*CFLUID*TWALL
GROUP 19. Data communicated by satellite to GROUND
USEGRX=F
THESE DATA TRANSMITTED TO GROUND
```

```
LG(1) = LHGRID
LG(2) = LIGRID
LG(5) = LINLINE
LG(6) = ICONFIG
LG(8) = DCONFIG
LG(9) = LCONSTT
```

```
RG(1)=DIAM
RG(2)=S1
RG(3)=S2
RG(4)=W0
RG(5)=TBIN
IF (LCONSTT) THEN
RG(6)=TWALL
ELSE
RG(6)=QWALL
ENDIF
RG(7)=KFLUID
RG(8)=CFLUID
RG(9)= AMYCO
RG (10)=AMYRELAX
RG(12)= GEOFAC
GROUP 20. Preliminary print-out
ECHO=T
GROUP 21. Print-out of variables
```

```
OUTPUT(P1,Y,N,Y,Y,Y,Y)
OUTPUT(V1,Y,N,N,Y,Y,Y)
OUTPUT(W1,Y,N,N,Y,Y,Y)
OUTPUT(H1,Y,N,N,Y,Y,Y)
IF (LPOR) THEN
OUTPUT(VPOR,Y,N,N,Y,Y,Y)
OUTPUT(NPOR,Y,N,N,Y,Y,Y)
OUTPUT(HPOR,Y,N,N,Y,Y,Y)
ENDIF
GROUP 22. Spot-value print-out
NPRMON=LSWEEP;IYMON=NY/2;IZMON=NZ/2;TSTSWP=LSWEEP/10
GROUP 23. Field print-out and plot control
GROUP 24. Dumps for restarts
GROUP 25. Debug settings
RESTRT(ALL)
RESTRT(V1,W1,H1)
STOP
```

GROUND FILE

C 2 USER DIMENSIONS OWN ARRAYS HERE

PARAMETER (NXD = 1, NYD = 30, NZD = 51)
REAL CONOUT(NYD,NXD), CONIN(NYD,NXD)
REAL CONFRST(NYD,NXD), CONHALO(NYD,NXD)
REAL ASMIDL(NYD,NXD), AREF(NYD,NXD)
REAL ASFRST(NYD,NXD), APFRST(NYD,NXD), ASIN(NYD,NXD)
REAL APREF(NYD,NXD), ASREF(NYD,NXD)
REAL LZIN(NYD,NXD), LZOUT(NYD,NXD)
REAL LZFRST(NYD,NXD), LZLAST(NYD,NXD)
REAL HNHY(NYD,NXD), HNHZ(NYD,NXD), DHNH(NYD,NXD)
REAL HHHY(NYD,NXD), HHHZ(NYD,NXD), DHHH(NYD,NXD)
REAL PHHY(NYD,NXD), PHHZ(NYD,NXD), DPHH(NYD,NXD)
REAL LPY(NYD,NXD), LPZ(NYD,NXD)
REAL DPN(NYD,NXD), DPH(NYD,NXD), DLP(NYD,NXD)
REAL PHY(NYD,NXD), PHZ(NYD,NXD), PNY(NYD,NXD), PNZ(NYD,NXD)
REAL LNL(NYD,NXD), LNLZ(NYD,NXD)
REAL GWIN(NYD,NXD), HWIN(NYD,NXD), HVIN(NYD,NXD), GVIN(NYD,NXD)
REAL GWOUT(NYD,NXD), HWOUT(NYD,NXD)
REAL GMIN(NYD,NXD), HMIN(NYD,NXD)
REAL GMOUT(NYD,NXD), HMOUT(NYD,NXD)
REAL GMFRST(NYD,NXD), HMFRST(NYD,NXD)

C

REAL P1FRST(NYD,NXD), V1FRST(NYD,NXD)
REAL W1FRST(NYD,NXD), VALU(NYD,NXD)
REAL U1LAST(NYD,NXD), V1LAST(NYD,NXD), W1LAST(NYD,NXD)
REAL H1FRST(NYD,NXD)
REAL H1LAST(NYD,NXD), P1DOWN(NYD,NXD), P1HALO(NYD,NXD)
REAL W1MIDL(NYD,NXD), H1MIDL(NYD,NXD), PVAL(NYD,NXD)
REAL W1IN(NYD,NXD), W1OUT(NYD,NXD)
REAL V1IN(NYD,NXD), V1OUT(NYD,NXD)
REAL V1HALO(NYD,NXD)
REAL H1IN(NYD,NXD), H1OUT(NYD,NXD)

C

REAL AVEC1(3,NZD), AVEC2(3,NZD)
REAL SP1(NZD,NXD), W1WALL1(NZD,NXD)
REAL YWALL1(NZD,NXD), PWALL1(NZD,NXD)
REAL H1WALL1(NZD,NXD), AWALL1(NZD,NXD)
REAL PHI1(NZD,NXD), NUF1(NZD,NXD), CF1(NZD,NXD)
REAL NULM1(NZD,NXD), NUM1(NZD,NXD)
REAL NUUM1(NZD,NXD), NUMN1(NZD,NXD)
REAL CP1(NZD,NXD), CFZ1(NZD,NXD)

c

REAL SP2(NZD,NXD), W1WALL2(NZD,NXD)
REAL YWALL2(NZD,NXD), PWALL2(NZD,NXD)
REAL H1WALL2(NZD,NXD), AWALL2(NZD,NXD)
REAL PHI2(NZD,NXD), NUF2(NZD,NXD), CF2(NZD,NXD)
REAL NULM2(NZD,NXD), NUM2(NZD,NXD)
REAL NUUM2(NZD,NXD), NUMN2(NZD,NXD)
REAL CP2(NZD,NXD), CFZ2(NZD,NXD)

c

REAL PT1(3), PT2(3), PT3(3), PT4(3)
REAL XVEC(3), YVEC(3), ZVEC(3), VEC(3)
REAL PN(3), PH(3), LNL(3), NRML(3), TANG(3)
REAL LH(3), LP(3), HNH(3), HHH(3), PHH(3)

C

```
REAL DIAM, S1, S2, A, B, ATOT, ACYCL
REAL W0, WB, WMIDL
REAL RE, EU, NUBAR, PLAST, CDRP
REAL CPREF, NUFBAR, NUMBAR, NULMBAR, NUUMBAR, NUMNBAR
REAL TMIDL, TFU, TBIN, TWALL, TBLAST, LTITO, DTLM
REAL CFLUID, KFLUID
REAL PREF, PP, ST
REAL PI, COSPHI, SINPHI
REAL CMASS, C1HEAT, C2HEAT
REAL V1LST, W1LST, W1LH
REAL DIFCO
REAL CSGMMA, DTHTA, HQ, HR, V1UP, ZETA
INTEGER IZ1, IZ2
INTEGER I, J, KK
INTEGER LOCO, LOVAL, LOMH
LOGICAL LHGRID, LIGRID, LSTAGH
LOGICAL LINLINE, LCONSTT
LOGICAL LICONFIG, LSHIFT, LDCONFIG
LOGICAL LZONE1, LZONE2, LZONE3
C
C 3 USER PLACES HIS DATA STATEMENTS HERE:
  DATA ACYL,NUFBAR,NUMBAR,NULMBAR,NUMNBAR,
  $CDRP,CDRF/7*0/
  COMMON/GRIDTYPE/IGRID,IYF1,IYF2
C*****
C
C--- GROUP 1. RUN TITLE AND OTHER PRELIMINARIES
C
  1 GO TO (1001,1002),ISC
1001 CONTINUE
  PI = 4.*ATAN(1.0)
C
C USER MAY HERE CHANGE MESSAGE TRANSMITTED TO THE VDU SCREEN OR
C BATCH-RUN LOG FILE.
  IF(IGR.EQ.1.AND.ISC.EQ.1) THEN
    CALL WRYT40('GROUND file is GROUND.FTN of: 110191 ')
    CALL WRYT40('PHOENICS version number is : 1.6 ')
  ENDIF
C
  RETURN
1002 CONTINUE
  RETURN
C*****
C
C--- GROUP 13. BOUNDARY CONDITIONS AND SPECIAL SOURCES
C
  13 CONTINUE
  GO TO (130,131,132,133,134,135,136,137,138,139,1310,
  11311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321),ISC
130 CONTINUE
C----- SECTION 1 ----- COEFFICIENT = GRND
C
C COEFFICIENT BASED ON HYBRID FORMULATION.
C
  LOCO = LOF(CO)
  DIFCO = RHO1*ENUL*PRNDTL(INDVAR)
  DO 1301 J = IYF, IYL
```

```

DO 1301 I = IXF, IXL
IJ = L0CO + J + NY*(I-1)
IF(NPATCH(1:3).EQ.'UPS') THEN
IF (INDVAR.EQ.W1) THEN
WW = .5*(CONFRST(J,I)+CONIN(J,I))
DW = DIFCO/LZFRST(J,I)
F(L0CO + J + NY*(I-1)) =
$AMAX1(0., DW - DIFCUT*ABS(WW))
$+ AMAX1(0.,WW)
ELSE IF (INDVAR.EQ.V1) THEN
IF (J.LT.IYF1) THEN
WW = .5*CONIN(IYF1,I)*ASIN(IYF1,I)/ASIN(J,I)
DV = .5*DIFCO*ASIN(IYF1,I)/(ASIN(J,I)*LZIN(IYF1,I))
ELSE IF (J.LT.IYF2) THEN
WW = .5*(CONIN(J,I)+CONIN(J+1,I))
DV = .5*(DIFCO/LZIN(J,I)+DIFCO/LZIN(J+1,I))
ELSE
WV = CONIN(IYF2,I)
DV = DIFCO/LZIN(IYF2,I)
END IF
IF (J.LT.IYF2) THEN
F(IJ) = (AMAX1(0., DV - DIFCUT*ABS(WV))
$+ AMAX1(0.,WV))
ELSE
c  STAGGERED I-GRIDS
F(IJ) =
$.5*(AMAX1(0., DV - DIFCUT*ABS(WV))
$+ .5*AMAX1(0.,WV))
END IF
ELSE
DD = DIFCO/LZIN(J,I)
F(IJ) = 0.
$+AMAX1(0., DD - DIFCUT*ABS(CONIN(J,I)))
END IF
ELSE IF(NPATCH(1:3).EQ.'DWN') THEN
IF (INDVAR.EQ.W1) THEN
WW = .5*(CONHALO(J,I)+CONOUT(J,I))
DW = DIFCO/LZLAST(J,I)
F(IJ) =
$AMAX1(0.,DW - DIFCUT*ABS(WW))
$+ AMAX1(0.,-1.*WW)
END IF
END IF
1301 CONTINUE
RETURN
131 CONTINUE
C----- SECTION 2 ----- COEFFICIENT = GRND1
C
C  CO IS PROPORTIONAL TO 1/VI
C
L0CO = L0F(CO)
DO 1302 J = IYF, IYL
DO 1302 I = IXF, IXL
IF (ABS(W1LAST(J,I)).LT.1.E-12) THEN
F(L0CO + J + NY*(I-1)) = AMYCO/1.E-12
WRITE (6,*) ' WARNING W1LAST < 1.E-12 AT', I, J
WRITE (14,*) ' WARNING W1LAST < 1.E-12 AT', I, J

```

```
ELSE
  F(LOCO + J + NY*(I-1)) = AMYCO/ABS(W1LAST(J,I))
END IF
1302 CONTINUE
  RETURN
132 CONTINUE
C----- SECTION 3 ----- COEFFICIENT = GRND2
  RETURN
133 CONTINUE
C----- SECTION 4 ----- COEFFICIENT = GRND3
  RETURN
134 CONTINUE
C----- SECTION 5 ----- COEFFICIENT = GRND4
  RETURN
135 CONTINUE
C----- SECTION 6 ----- COEFFICIENT = GRND5
  RETURN
136 CONTINUE
C----- SECTION 7 ----- COEFFICIENT = GRND6
  RETURN
137 CONTINUE
C----- SECTION 8 ----- COEFFICIENT = GRND7
  RETURN
138 CONTINUE
C----- SECTION 9 ----- COEFFICIENT = GRND8
  RETURN
139 CONTINUE
C----- SECTION 10 ----- COEFFICIENT = GRND9
  RETURN
1310 CONTINUE
C----- SECTION 11 ----- COEFFICIENT = GRND10
  RETURN
1311 CONTINUE
C----- SECTION 12 ----- VALUE = GRND
C
C  INLET BOUNDARY CONDITIONS.
C
  LOVAL = LOF(VAL)
  IF(NPATCH(1:2).EQ.'UP') THEN
    IF (INDVAR.EQ.P1) THEN
      DO 1303 J = IYF, IYL
        DO 1303 I = IXF, IXL
          IF (J.GE.IYF1) THEN
            F(LOVAL + J + NY*(I-1)) = CMASS*RHO1*CONIN(J,I)
          ELSE
            F(LOVAL + J + NY*(I-1)) = 0.
          END IF
        1303 CONTINUE
      ELSE IF (INDVAR.EQ.V1) THEN
        DO 1304 J = IYF, IYL
          DO 1304 I = IXF, IXL
            IF (J.LE.IYF2) THEN
              F(LOVAL + J + NY*(I-1)) = GVIN(J,I)*V1IN(J,I)
              $+ HVIN(J,I)*W1IN(J,I)
            ELSE
              F(LOVAL + J + NY*(I-1)) = HVIN(IYF1,I)*
              $W1IN(IYF1,I)
```

```

      END IF
1304 CONTINUE
      ELSE IF (INDVAR.EQ.W1) THEN
        DO 1305 J = IYF, IYL
          DO 1305 I = IXF, IXL
            IF (J.EQ.IYL1) THEN
              V1UP = V1IN(J,I)
            ELSE IF (J.EQ.NY) THEN
              V1UP = V1IN(NY-1,I)
            ELSE
              V1UP = AMIN1(0.,V1IN(J,I)) + AMAX1(0.,V1IN(J-1,I))
            END IF
            F(L0VAL + J + NY*(I-1)) = GWIN(J,I)*V1UP
            $+ HWIN(J,I)*W1IN(J,I)
1305 CONTINUE
            ELSE IF (INDVAR.EQ.H1) THEN
              DO 1306 J = IYF, IYL
                DO 1306 I = IXF, IXL
                  F(L0VAL + J + NY*(I-1)) = H1IN(J,I)
1306 CONTINUE
            END IF
C
C   OUTLET BOUNDARY CONDITIONS
C
            ELSE IF (NPATCH(1:3).EQ.'DWN') THEN
              IF (INDVAR.EQ.V1) THEN
                DO 1307 J = IYF, IYL
                  DO 1307 I = IXF, IXL
                    IF (J.GE.IYL1) THEN
                      F(L0VAL + J + NY*(I-1)) = V1OUT(J,I)
                    ELSE
                      F(L0VAL + J + NY*(I-1)) = DOTZ*W1OUT(IYL1,I)
                    END IF
1307 CONTINUE
                    ELSE IF (INDVAR.EQ.W1) THEN
                      DO 1308 J = IYF, IYL
                        DO 1308 I = IXF, IXL
                          IF (J.EQ.IYL1) THEN
                            V1UP = V1OUT(IYL1-1,I)
                          ELSE IF (J.EQ.NY) THEN
                            V1UP = V1OUT(NY-1,I)
                          ELSE
                            V1UP = AMIN1(0.,V1OUT(J,I)) +
                              $AMAX1(0.,V1OUT(J-1,I))
                          END IF
                          F(L0VAL + J + NY*(I-1)) = GWOUT(J,I)*V1UP
                          $+ HWOUT(J,I)*W1OUT(J,I)
1308 CONTINUE
                          ELSE IF (INDVAR.EQ.H1) THEN
                            DO 1309 J = IYF, IYL
                              DO 1309 I = IXF, IXL
                                F(L0VAL + J + NY*(I-1)) = H1OUT(J,I)
1309 CONTINUE
                              END IF
                              END IF
                              RETURN
1312 CONTINUE

```

```

C----- SECTION 13 ----- value = GRND1
  RETURN
1313 CONTINUE
C----- SECTION 14 ----- value = GRND2
  RETURN
1314 CONTINUE
C----- SECTION 15 ----- value = GRND3
  RETURN
1315 CONTINUE
C----- SECTION 16 ----- value = GRND4
C
C  DOWNSTREAM PRESSURE VALUE
C
  LOVAL = LOF(VAL)
  DO 1322 J = IYF, IYL
  DO 1322 I = IXF, IXL
    F(LOVAL + J + NY*(I-1)) = PVAL(J,I)
1322 CONTINUE
  RETURN
1316 CONTINUE
C----- SECTION 17 ----- value = GRND5
  RETURN
1317 CONTINUE
C----- SECTION 18 ----- value = GRND6
  RETURN
1318 CONTINUE
C----- SECTION 19 ----- value = GRND7
  RETURN
1319 CONTINUE
C----- SECTION 20 ----- value = GRND8
  RETURN
1320 CONTINUE
C----- SECTION 21 ----- value = GRND9
  RETURN
1321 CONTINUE
C----- SECTION 22 ----- value = GRND10
  RETURN
C*****
C
C--- GROUP 19. Special calls to GROUND from EARTH
C
  19 GO TO (191,192,193,194,195,196,197,198),ISC
191 CONTINUE
C * ----- SECTION 1 ---- Start of time step.
  OPEN(93,FILE='PCO',FORM='FORMATTED',STATUS='UNKNOWN',
    $ACCESS='SEQUENTIAL')
  WRITE (93,*) ' SWEEP  PRESS'
  OPEN(94,FILE='MONFIL',FORM='FORMATTED',STATUS='UNKNOWN',
    $ACCESS='SEQUENTIAL')
  OPEN(98,FILE='PLOT',FORM='FORMATTED',STATUS='UNKNOWN',
    $ACCESS='SEQUENTIAL')
C
C  DATA TRANSMITTED FROM SATELLITE
C
  LHGRID = LG(1)
  LIGRID = LG(2)
  LINLINE = LG(3)

```

```
LICONFIG = LG(4)
LDCONFIG = LG(5)
LCONSTT = LG(6)
LSHIFT = .FALSE.
IF (LIGRID.AND.(.NOT.LDCONFIG)) LSHIFT = .TRUE.
DIAM = RG(1)
S1 = RG(2)
S2 = RG(3)
W0 = RG(4)
TBIN = RG(5)
IF (.NOT.LCONSTT) QWALL = RG(6)
IF (LCONSTT) TWALL = RG(6)
KFLUID = RG(7)
CFLUID = RG(8)
AMYCO = RG(9)
AMYRLX = RG(10)
RPOR = RG(11)
SCLGRD = RG(11)
```

```
C
C   GEOMETRIC PARAMETERS RELATED TO GRID TYPE
C
```

```
RESCALE = 1.
A = S1/DIAM
B = S2/DIAM
IF (LIGRID) THEN
  ACTOT = S1/2.
  IF (LDCONFIG) THEN
    ACTOT = (S1-DIAM)/2.
  IZ1 = 1
  IZ2 = NZ-1
  IYF1 = 1
  IYF2 = NY/2
  IYL1 = NY/2+1
  IYL2 = NY
  IF (A.GE.(2*B*B-.5)) THEN
    RESCALE = (A-1)/(SQRT(A*A+4.*B*B)-2.)
  ELSE
    RESCALE = 1.
  END IF
  ELSE
    IZ1 = 1
    IZ2 = NZ-1
    IYF1 = INT(1.5*NY*(S2-DIAM)/(S2-DIAM+S1))+1
    IYF2 = NY
    IYL1 = IYF1
    IYL2 = IYF2
    RESCALE = A/(A-1.)
  ENDIF
  IZFRST = 1
  IZLAST = NZ-1
  IZHALO = NZ
c
  ELSE IF (LHGRID) THEN
    ACTOT = (S1-DIAM)/2.
    IZ1 = INT(SCLGRD*(NZ-1)*3*DIAM/(4.*S2 + 2.*DIAM))
    IZ2 = NZ - IZ1
    IF (LDCONFIG)
```



```
$IZ1 = INT(SCLGRD*(NZ-1)*3*DIAM/(4.*S2 + 2.*DIAM))-1
IZ2 = NZ - IZ1
IYF1 = 1
IYF2 = NY
IYL1 = 1
IYL2 = NY
IZFRST = 1
IZLAST = NZ-1
IZHALO = NZ
RESCALE = 1.
IF (A.GE.(2*B*B-.5)) THEN
  RESCALE = (A-1)/(SQRT(A*A+4.*B*B)-2.)
END IF
IF (LICONFIG) THEN
  ACTOT = S1/2.
  IZ1 = INT(SCLGRD*(NZ - 1.)*3.*DIAM/(4.*S2 + 2.*DIAM))-1
  IZ1 = NZ/2-IZ1
  IZ2 = NZ - IZ1
  IYF1 = 1
  IYF2 = NY
  IYL1 = 1
  IYL2 = NY
  IZFRST = 1
  IZLAST = NZ-1
  IZHALO = NZ
  RESCALE = A/(A-1.)
END IF
END IF
IF (LHGRID.AND.(.NOT.LICONFIG).AND.(.NOT.LINLINE))
  $LSTAGH = .TRUE.
C
C  INITIALIZE DATA VALUES
C
  WB = RESCALE*W0
  CMASS = 1.
  C1HEAT = 1.
  IF (LCONSTT) C2HEAT = CFLUID*TWALL*(1. - C1HEAT)
  IF (.NOT.LCONSTT) C2HEAT = 0.
  DO 1901 J=1,NY
  DO 1901 I=1,NX
    W1LAST(J,I) = RESCALE*W0
    W1FRST(J,I) = RESCALE*W0
    W1IN(J,I) = RESCALE*W0
    W1OUT(J,I) = RESCALE*W0
    CONIN(J,I) = RESCALE*W0
    V1LAST(J,I)= 0.0
    V1FRST(J,I)= 0.0
    V1IN(J,I)= 0.0
    V1OUT(J,I)= 0.0
    H1IN(J,I) = CFLUID*TBIN
    H1OUT(J,I) = CFLUID*TBIN
    H1FRST(J,I) = CFLUID*TBIN
    H1LAST(J,I) = CFLUID*TBIN
    PVAL(J,I) = 0.
    P1HALO(J,I) = 0.
    P1DOWN(J,I) = 0.
  1901 CONTINUE
```

IZMIDL = (NZ-1)/2

C

C THIS SECTION IS USED TO CALCULATE AREAS AND DISTANCES
C USED IN THE ITERATIVE PROCEDURE

C

C ASIN = LOW CELL AREA AT THE FIRST CELL.

C ASFRST = HIGH CELL AREA AT THE FIRST CELL.

C APFRST = HIGH CELL PROJECTED AREA AT THE FIRST CELL.

C ASMIDL = CELL AREA AT THE MIDDLE CELL.

C AREF = REFERENCE AREA USED TO CALCULATE BULK VELOCITY ETC.

C ASREF = REFERENCE AREA

C APREF = PROJECTED REFERENCE AREA

C

C LZIN = DISTANCE FROM UPSTREAM TO THE FIRST CELL

C LZFRST = DISTANCE ACROSS THE LAST CELL

C LZLAST = DISTANCE ACROSS THE LAST CELL

C LZOUT = DISTANCE FROM THE LAST CELL TO THE HALO CELL

C

CALL GTIZYX(ASURFL,IZFRST,ASIN,NYD,NXD)

CALL GTIZYX(ASURFH,IZFRST,ASFRST,NYD,NXD)

CALL GTIZYX(APROJH,IZFRST,APFRST,NYD,NXD)

CALL GTIZYX(ASURFL,IZMIDL,ASMIDL,NYD,NXD)

CALL GTIZYX(ASURFH,IZLAST,AREF,NYD,NXD)

CALL GTIZYX(ASURFH,IZLAST,ASREF,NYD,NXD)

CALL GTIZYX(APROJH,IZLAST,APREF,NYD,NXD)

CALL GTIZYX(DHZPH,IZFRST,LZFRST,NYD,NXD)

CALL GTIZYX(DHZPH,IZLAST,LZLAST,NYD,NXD)

CALL GTIZYX(DZGPH,IZLAST,LZOUT,NYD,NXD)

c

ATOT = 0.0

DO 1902 J= IYL1, IYL2

DO 1902 I= 1, NX

LZLAST(J,I) = 2.*LZLAST(J,I)

LZFRST(J,I) = 2.*LZFRST(J,I)

ATOT=ATOT+AREF(J,I)

1902 CONTINUE

c

IF (LINLINE) THEN

CALL SETCXY(LZIN,LZOUT,NYD,NXD,1.,0.,NY)

ELSE

CALL SWCHY(LZIN,LZOUT,NYD,NXD,NY,NX)

IF (LSHIFT) CALL SHFTY(LZIN,IYF1,NYD,NXD,NY,NX)

END IF

c

CALL VECTR(XVEC,1.,0.,0.)

CALL VECTR(YVEC,0.,1.,0.)

CALL VECTR(ZVEC,0.,0.,1.)

C

C THIS SECTION IS WHERE THE CONVECTION FLUX

C FOR NON-ORTHOGONAL GRIDS IS COMPUTED

C

C HQ AND HR CONVERT TANGENTIAL RESOLUTES TO COMPONENTS.

C GM AND HM CONVERT TANGENTIAL RESOLUTES TO NORMAL RESOLUTES

C

IF (NONORT) THEN

CALL GTIZYX(MFCRNH,NZ-1,GMOUT,NYD,NXD)

CALL GTIZYX(MFCSHH,NZ-1,HMOUT,NYD,NXD)

```
CALL GTIZYX(MFCRNH,1,GMFRST,NYD,NXD)
CALL GTIZYX(MFCSHH,1,HMFRST,NYD,NXD)
ELSE
DO 1903 I = 1, NX
DO 1903 J = 1, NY
GMOUT(J,I) = 0.
HMOUT(J,I) = 1.
GMFRST(J,I) = 0.
HMFRST(J,I) = 1.
1903 CONTINUE
END IF
C
IF (LINLINE) THEN
CALL SETCXY(GMIN,GMOUT,NYD,NXD,1.,0.,NY)
CALL SETCXY(HMIN,HMOUT,NYD,NXD,1.,0.,NY)
ELSE
CALL GTIZYX(UNIVNN,NZ-1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,NZ-1,PNZ,NYD,NXD)
CALL GTIZYX(UNIVHN,NZ-1,PHY,NYD,NXD)
CALL GTIZYX(UNIVHH,NZ-1,PHZ,NYD,NXD)
IF (LSHIFT) THEN
PNYTOP = PNY(IYL1-1,1)
PNZTOP = PNZ(IYL1-1,1)
END IF
CALL SETCXY(VALU,PNY,NYD,NXD,1.,0.,NY-1)
CALL SWCHY(PNY,VALU,NYD,NXD,NY-1,NX)
IF (LSHIFT) CALL SHFTY(PNY,IYL1,NYD,NXD,NY-1,NX)
CALL SETCXY(VALU,PNZ,NYD,NXD,-1.,0.,NY-1)
CALL SWCHY(PNZ,VALU,NYD,NXD,NY-1,NX)
IF (LSHIFT) CALL SHFTY(PNZ,IYL1,NYD,NXD,NY-1,NX)
CALL SETCXY(VALU,PHY,NYD,NXD,-1.,0.,NY)
CALL SWCHY(PHY,VALU,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(PHY,IYL1,NYD,NXD,NY,NX)
CALL SETCXY(VALU,PHZ,NYD,NXD,1.,0.,NY)
CALL SWCHY(PHZ,VALU,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(PHY,IYL1,NYD,NXD,NY,NX)
PNY(NY,1) = PNY(NY-1,1)
PNZ(NY,1) = PNZ(NY-1,1)
IF (LSHIFT) THEN
PNY(NY,1) = PNYTOP
PNZ(NY,1) = -1.*PNZTOP
END IF
DO 1904 J = 1, NY
DO 1904 I = 1, NX
CALL GETPT(1,J+1,1,PT1(1),PT1(2),PT1(3))
CALL GETPT(1,J,1,PT2(1),PT2(2),PT2(3))
CALL GETPT(2,J,1,PT3(1),PT3(2),PT3(3))
CALL GETPT(2,J+1,1,PT4(1),PT4(2),PT4(3))
CALL NORM2(PT1,PT2,PT3,PT4,VEC)
CALL VECTR(PN, 0., PNY(J,I), PNZ(J,I))
CALL VECTR(PH, 0., PHY(J,I), PHZ(J,I))
CSGMMA = DOT(PH,PN)
ZETAY = DOT(VEC,PN)
ZETAZ = DOT(VEC,PH)
HQ = -1.*CSGMMA/(1. - CSGMMA*CSGMMA)
HR = 1./(1. - CSGMMA*CSGMMA)
GMIN(J,I) = HQ*ZETAZ + HR*ZETAY
```

```
HMIN(J,I) = HR*ZETAZ + HQ*ZETAY
1904 CONTINUE
END IF
C
C THIS SECTION IS WHERE THE W-MOMENTUM
C CURVATURE FACTORS ARE COMPUTED.
C
C PHY AND PHZ = COMPONENTS OF P-H. (BETWEEN IZ = 1 AND 2)
C PNY AND PNZ = COMPONENTS OF P-N. (AT IZ = 1)
C NRML = OUTWARD UNIT NORMAL AT P.
C CSGMMA = COSINE OF THE ANGLE BETWEEN W-P AND P-N.
C HQ AND HR = FACTORS FOR CONVERSION FROM RESOLUTES TO COMPONENTS.
C DTHTA = CURVATURE X ARC LENGTH.
C GWUP AND HWUP = MULTIPLIERS FOR V.
C
C A) INLET
C
CALL GTIZYX(UNIVNN,1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,1,PNZ,NYD,NXD)
CALL GTIZYX(DYGPN,1,DPN,NYD,NXD)
CALL GTIZYX(UNIVHN,1,PHY,NYD,NXD)
CALL GTIZYX(UNIVHH,1,PHZ,NYD,NXD)
CALL GTIZYX(DZGPH,1,DPH,NYD,NXD)
CALL GTIZYX(UNIVHN,NZ-1,LPY,NYD,NXD)
CALL GTIZYX(UNIVHH,NZ-1,LPZ,NYD,NXD)
CALL GTIZYX(DZGPH,NZ-1,DLP,NYD,NXD)
IF (.NOT.LINLINE) THEN
IF (.NOT.(LIGRID.AND.LDCONFIG)) THEN
CALL SETCXY(VALU,LPY,NYD,NXD,-1.,0.,NY)
CALL SETCXY(LPY,VALU,NYD,NXD,1.,0.,NY)
ELSE
CALL SETCXY(VALU,LPY,NYD,NXD,-1.,0.,NY)
CALL SWCHY(LPY,VALU,NYD,NXD,NY,NX)
CALL SETCXY(VALU,LPZ,NYD,NXD,1.,0.,NY)
CALL SWCHY(LPZ,VALU,NYD,NXD,NY,NX)
CALL SETCXY(VALU,DLP,NYD,NXD,1.,0.,NY)
CALL SWCHY(DLP,VALU,NYD,NXD,NY,NX)
END IF
END IF
DO 1905 I = 1, NX
PNY(NY,I) = PNY(NY-1,I)
PNZ(NY,I) = PNZ(NY-1,I)
DPN(NY,I) = DPN(NY-1,I)
DO 1905 J = 1, NY
CALL VECTR(LP, 0., DLP(J,I)*LPY(J,I), DLP(J,I)*LPZ(J,I))
CALL VECTR(PN, 0., DPN(J,I)*PNY(J,I), DPN(J,I)*PNZ(J,I))
CALL VECTR(PH, 0., DPH(J,I)*PHY(J,I), DPH(J,I)*PHZ(J,I))
CALL ADDVEC(LP,PH,LH)
CALL NORM(LP)
CALL NORM(PH)
CALL NORM(PN)
CALL NORM(LH)
CALL CROSS(LP,LH,VEC)
CALL CROSS(LH,VEC,NRML)
CALL NORM(NRML)
CSGMMA = DOT(LP,PN)
DELTA = ACOS(DOT(LP,PH))
```

```

DTHTA = 2.*SIN(DELTA/2.)
HQ = CSGMMA/(1. - CSGMMA*CSGMMA)
HR = -1./(1. - CSGMMA*CSGMMA)
GWIN(J,I) = DTHTA*(HR*DOT(NRML,PN) + HQ*DOT(NRML,LP))
HWIN(J,I) = 1. + DTHTA*(HQ*DOT(NRML,PN) + HR*DOT(NRML,LP))
1905 CONTINUE
C
C   B) OUTLET
C
  IF (LINLINE) THEN
    CALL SETCX(Y,GWOUT,GWIN,NYD,NXD,1.,0.,NY)
    CALL SETCX(Y,HWOUT,HWIN,NYD,NXD,1.,0.,NY)
  ELSE
    IF (LSHIFT) THEN
      CALL GTIZYX(UNIVNN,NZ-1,HNHY,NYD,NXD)
      CALL GTIZYX(UNIVNH,NZ-1,HNHZ,NYD,NXD)
      CALL GTIZYX(DYGPN,NZ-1,DHNNH,NYD,NXD)
      HNHYTOP = HNHY(IYL1-1,1)
      HNHZTOP = HNHZ(IYL1-1,1)
      DHNNHTOP = DHNNH(IYL1-1,1)
    ENDIF
    CALL GTIZYX(UNIVNN,NZ,HNHY,NYD,NXD)
    CALL GTIZYX(UNIVNH,NZ,HNHZ,NYD,NXD)
    CALL GTIZYX(DYGPN,NZ,DHNNH,NYD,NXD)
    CALL GTIZYX(UNIVHN,NZ-1,PHY,NYD,NXD)
    CALL GTIZYX(UNIVHH,NZ-1,PHZ,NYD,NXD)
    CALL GTIZYX(DZGPH,NZ-1,DPH,NYD,NXD)
    CALL GTIZYX(UNIVHN,1,HHHY,NYD,NXD)
    CALL GTIZYX(UNIVHH,1,HHHZ,NYD,NXD)
    CALL GTIZYX(DZGPH,1,DHHH,NYD,NXD)
    CALL SETCX(Y,VALU,HHHY,NYD,NXD,-1.,0.,NY)
    CALL SWCHY(HHHY,VALU,NYD,NXD,NY,NX)
    CALL SETCX(Y,VALU,HHHZ,NYD,NXD,1.,0.,NY)
    CALL SWCHY(HHHZ,VALU,NYD,NXD,NY,NX)
    CALL SETCX(Y,VALU,DHHH,NYD,NXD,1.,0.,NY)
    CALL SWCHY(DHHH,VALU,NYD,NXD,NY,NX)
    IF (LSHIFT) CALL SHFTY(HHHY,IYL1,NYD,NXD,NY,NX)
    IF (LSHIFT) CALL SHFTY(HHHZ,IYL1,NYD,NXD,NY,NX)
    IF (LSHIFT) CALL SHFTY(DHHH,IYL1,NYD,NXD,NY,NX)
    DO 1906 I = 1, NX
      HNHY(NY,I) = HNHY(NY-1,I)
      HNHZ(NY,I) = HNHZ(NY-1,I)
      DHNNH(NY,I) = DHNNH(NY-1,I)
      IF (LSHIFT) THEN
        HNHY(NY,I) = HNHYTOP
        HNHZ(NY,I) = HNHZTOP
        DHNNH(NY,I) = DHNNHTOP
      END IF
    DO 1906 J = 1, NY
      CALL VECTR(PH,0.,DPH(J,I)*PHY(J,I),DPH(J,I)*PHZ(J,I))
      CALL VECTR(HNNH,0.,DHNNH(J,I)*HNHY(J,I),DHNNH(J,I)*HNHZ(J,I))
      CALL VECTR(HHH,0.,DHHH(J,I)*HHHY(J,I),DHHH(J,I)*HHHZ(J,I))
      CALL ADDVEC(PH,HHH,PHH)
      CALL NORM(PH)
      CALL NORM(HNNH)
      CALL NORM(HHH)
      CALL NORM(PHH)

```

```

CALL CROSS(PH,PHH,VEC)
CALL CROSS(PHH,VEC,NRML)
CALL NORM(NRML)
CSGMMA = DOT(HHH,HNH)
DELTA = ACOS(DOT(PH,HHH))
DTHTA = 2.*SIN(DELTA/2.)
HQ = CSGMMA/(1. - CSGMMA*CSGMMA)
HR = -1./(1. - CSGMMA*CSGMMA)
GWOUT(J,I) = DTHTA*(HR*DOT(NRML,HNH) + HQ*DOT(NRML,HHH))
HWOUT(J,I) = 1. + DTHTA*(HQ*DOT(NRML,HNH) + HR*DOT(NRML,HHH))
IF ((LIGRID.AND.LDCONFIG).AND.(J.EQ.IYL1)) THEN
  GWOUT(J,I) = 0.
  HWOUT(J,I) = DOT(PH,HHH)
END IF
1906 CONTINUE
END IF
C
C THIS SECTION RELATES TO GEOMETRY USED TO ACCOUNT FOR
C DIVERGENCE OF THE GRID AT THE BOUNDARY.
C
C ASSIGNMENT OF V_LOW AT IZ = 1
C
C PNY AND PNZ = COMPONENTS OF P-N.
C LNLY AND LNLZ = COMPONENTS OF L-NL
C CSGMMA = COSINE OF THE ANGLE BETWEEN L-NL AND L-P
C HQ AND HR = FACTORS FOR CONVERSION FROM RESOLUTES TO COMPONENTS.
C HVIN AND GVIN = MULTIPLIERS FOR V RESPECTIVELY (UPSTREAM)
C
CALL GTIZYX(UNIVNN,1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,1,PNZ,NYD,NXD)
CALL GTIZYX(UNIVNN,NZ-1,LNLY,NYD,NXD)
CALL GTIZYX(UNIVNH,NZ-1,LNLZ,NYD,NXD)
CALL GTIZYX(UNIVHN,NZ-1,LPY,NYD,NXD)
CALL GTIZYX(UNIVHH,NZ-1,LPZ,NYD,NXD)
IF (.NOT.LINLINE) THEN
  CALL SETCXV(VALU,LNLY,NYD,NXD,1.,0.,NY-1)
  CALL SWCHY(LNLY,VALU,NYD,NXD,NY-1,NX)
  IF (LSHIFT) CALL SHFTY(LNLY,IYL1,NYD,NXD,NY-1,NX)
  CALL SETCXV(VALU,LNLZ,NYD,NXD,-1.,0.,NY-1)
  CALL SWCHY(LNLZ,VALU,NYD,NXD,NY-1,NX)
  IF (LSHIFT) CALL SHFTY(LNLZ,IYL1,NYD,NXD,NY-1,NX)
  CALL SETCXV(VALU,LPY,NYD,NXD,-1.,0.,NY)
  CALL SWCHY(LPY,VALU,NYD,NXD,NY,NX)
  IF (LSHIFT) CALL SHFTY(LPY,IYL1,NYD,NXD,NY,NX)
  CALL SETCXV(VALU,LPZ,NYD,NXD,1.,0.,NY)
  CALL SWCHY(LPZ,VALU,NYD,NXD,NY,NX)
  IF (LSHIFT) CALL SHFTY(LPZ,IYL1,NYD,NXD,NY,NX)
  IF (LSHIFT) THEN
    LNLY(IYF1-1,1) = LNLY(IYF1,1)
    LNLZ(IYF1-1,1) = LNLZ(IYF1,1)
    LPY(IYF1-1,1) = LPY(IYF1,1)
    LPZ(IYF1-1,1) = LPZ(IYF1,1)
  END IF
END IF
DO 1907 J = 1, NY
DO 1907 I = 1, NX
CALL VECTR(PN, 0., PNY(J,I), PNZ(J,I))

```

```

CALL VECTR(LNL, 0., LNLX(J,I), LNLZ(J,I))
CALL VECTR(LP, 0., LPY(J,I), LPZ(J,I))
CSGMMA = DOT(LP,LNL)
HQ = -1.*CSGMMA/(1. - CSGMMA*CSGMMA)
HR = 1./(1. - CSGMMA*CSGMMA)
HVIN(J,I) = HQ*DOT(LNL,PN) + HR*DOT(PN,LP)
GVIN(J,I) = HR*DOT(LNL,PN) + HQ*DOT(PN,LP)
IF ((LIGRID.AND.LDCONFIG).AND.(J.EQ.IYF2)) THEN
CALL GTIZYX(YP,1,VALU,NYD,NXD)
DELTAY = S1/2. - VALU(IYF2,I)
PNDOTZ = DOT(PN,ZVEC)
END IF
IF ((LIGRID.AND.LICONFIG).AND.(J.LE.IYF1-1)) THEN
GVIN(J,I) = 0.
HVIN(J,I) = DOT(PN,ZVEC)
PNDOTZ = DOT(PN,ZVEC)
END IF
1907 CONTINUE
C
IF (LIGRID) THEN
CALL GTIZYX(UNIVNN,NZ-1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,NZ-1,PNZ,NYD,NXD)
CALL VECTR(PN,0.,PNY(IYL1-1,1),PNZ(IYL1-1,1))
HVOUTX = DOT(ZVEC,PN)
END IF
C
C COMPUTE COEFFICIENTS PER UNIT SURFACE (NOT PROJECTED) AREA
C
DO 1908 J = 1, NY
DO 1908 I = 1, NX
GMFRST(J,I) = GMFRST(J,I)*APFRST(J,I)/ASFRST(J,I)
HMFRST(J,I) = HMFRST(J,I)*APFRST(J,I)/ASFRST(J,I)
GMOUT(J,I) = GMOUT(J,I)*APREF(J,I)/ASREF(J,I)
HMOUT(J,I) = HMOUT(J,I)*APREF(J,I)/ASREF(J,I)
1908 CONTINUE
C
IF (IYF1.GT.1) THEN
DO 1910 I = 1, NX
DO 1909 J = 1, IYF1-2
HMIN(J,I) = 0.
GMIN(J,I) = 0.
HWIN(J,I) = 0.
GWIN(J,I) = 0.
HVIN(J,I) = 0.
GVIN(J,I) = 0.
LZFRST(J,I) = LZFRST(IYF1,I)
LZIN(J,I) = LZIN(IYF1,I)
1909 CONTINUE
HMIN(IYF1-1,I) = 0.
GMIN(IYF1-1,I) = 0.
HWIN(IYF1-1,I) = 0.
GWIN(IYF1-1,I) = 0.
LZFRST(IYF1-1,I) = LZFRST(IYF1,I)
LZIN(IYF1-1,I) = LZIN(IYF1,I)
IF (.NOT.LIGRID) HVIN(IYF1-1,I) = 0
IF (.NOT.LIGRID) GVIN(IYF1-1,I) = 0
1910 CONTINUE

```

```

END IF
IF (IYF2.LT.NY) THEN
DO 1912 I = 1, NX
DO 1911 J = IYF2+1, NY
HMIN(J,I) = 0.
GMIN(J,I) = 0.
HWIN(J,I) = 0.
GWIN(J,I) = 0.
HVIN(J,I) = 0.
GVIN(J,I) = 0.
1911 CONTINUE
1912 CONTINUE
END IF
c
IF (IYL1.GT.1) THEN
DO 1913 I = 1, NX
DO 1913 J = 1, IYL1-1
AREF(J,I) = 0.
HMOUT(J,I) = 0.
GMOUT(J,I) = 0.
HWOUT(J,I) = 0.
GWOUT(J,I) = 0.
LZOUT(J,I) = LZOUT(IYL1,I)
LZLAST(J,I) = LZLAST(IYL1,I)
1913 CONTINUE
END IF
IF (IYL2.LT.NY) THEN
DO 1914 I = 1, NX
DO 1914 J = IYL2+1, NY
ASREF(J,I) = 1.E-20
APREF(J,I) = 1.E-20
AREF(J,I) = 1.E-20
HMOUT(J,I) = 0.
GMOUT(J,I) = 0.
LZOUT(J,I) = LZOUT(IYL2,I)
LZLAST(J,I) = LZLAST(IYL2,I)
1914 CONTINUE
END IF

```

```

C
C THIS SECTION IS WHERE GEOMETRIC FACTORS USED TO CALCULATE LOCAL
C COEFFICIENTS AROUND A TEST CYLINDER ARE COMPUTED.
C
C PHI(I) = ANGLE FROM FRONT LEADING EDGE OF CYLINDER (DEG)
C AWALL(I) = AREA OF CYLINDER WALL
C YWALL(I) = PERPENDICULAR DISTANCE FROM OUTSIDE CELL TO WALL
C
C FOR IN-LINE CONFIGURED GRIDS ONLY VALUES AT IY = 1 REQUIRE TO
C BE COMPUTED AND STORED. FOR DIAGONALLY CONFIGURED GRIDS VALUES
C AT IY = NY ALSO NEED TO BE STORED.
C
C PT4(1) = 0.
C PT4(2) = 0.
C PT4(3) = 0.
c
DO 1915 II = 1, NX
DO 1915 KK = 1, NZ

```



```
NJWALL = 1
CALL GTIZYX(DHYPN, KK, VALU, NYD, NXD)
YWALL1(KK, II) = VALU(NJWALL, II)
IF (NJWALL.EQ.1) CALL GTIZYX(ASURFS, KK, VALU, NYD, NXD)
AWALL1(KK, II) = VALU(NJWALL, II)
CALL GETPT(1, NJWALL, KK, PT1(1), PT1(2), PT1(3))
CALL GETPT(1, NJWALL, KK+1, PT2(1), PT2(2), PT2(3))
PT3(1) = 0.
PT3(2) = .5*(PT1(2) + PT2(2)) - PT4(2)
PT3(3) = .5*(PT1(3) + PT2(3)) - PT4(3)
CALL VECTR(VEC, 0., PT3(2), PT3(3))
CALL NORM(VEC)
AVEC1(1, KK) = VEC(1)
AVEC1(2, KK) = VEC(2)
AVEC1(3, KK) = VEC(3)
COSPFI = -1.*DOT(VEC, ZVEC)
PHI1(KK, II) = 180.*ACOS(COSPFI)/PI
1915 CONTINUE
c
  IF (LDCONFIG) THEN
    PT4(1) = 0.
    PT4(2) = S1/2.
    PT4(3) = S2
  ELSE IF (LHGRID.AND.(.NOT.LICONFIG)) THEN
    PT4(1) = 0.
    PT4(2) = 0.
    PT4(3) = S2
  END IF
c
DO 1916 II = 1, NX
DO 1916 KK = 1, NZ
  NJWALL = 1
  IF (LDCONFIG) NJWALL = NY
  CALL GTIZYX(DHYPN, KK, VALU, NYD, NXD)
  YWALL2(KK, II) = VALU(NJWALL, II)
  IF (NJWALL.EQ.1) CALL GTIZYX(ASURFS, KK, VALU, NYD, NXD)
  IF (NJWALL.EQ.NY) CALL GTIZYX(ASURFN, KK, VALU, NYD, NXD)
  AWALL2(KK, II) = VALU(NJWALL, II)
  IF (LDCONFIG) NJWALL = NY+1
  CALL GETPT(1, NJWALL, KK, PT1(1), PT1(2), PT1(3))
  CALL GETPT(1, NJWALL, KK+1, PT2(1), PT2(2), PT2(3))
  PT3(1) = 0.
  PT3(2) = .5*(PT1(2) + PT2(2)) - PT4(2)
  PT3(3) = .5*(PT1(3) + PT2(3)) - PT4(3)
  CALL VECTR(VEC, 0., PT3(2), PT3(3))
  CALL NORM(VEC)
  AVEC2(1, KK) = VEC(1)
  AVEC2(2, KK) = VEC(2)
  AVEC2(3, KK) = VEC(3)
  COSPFI = -1.*DOT(VEC, ZVEC)
  PHI2(KK, II) = 180.*ACOS(COSPFI)/PI
1916 CONTINUE
c
c  WRITE INITIAL DATA TO VDU AND PHIDA FILE
c
914 FORMAT (/,'          A = ',G11.4,'    B = ',G11.4)
951 FORMAT ('          IZ1 = ',I3,7X,'    IZ2 = ',I3,/)

```

```
915 FORMAT (' CONSTANT WALL HEAT FLUX')
918 FORMAT (' CONSTANT WALL TEMPERATURE')
949 FORMAT(' SWEEP EU  NU  RE ')
  WRITE (6,914) A, B
  WRITE (6,951) IZ1,IZ2
  WRITE (14,914) A, B
  WRITE (14,951) IZ1,IZ2
  IF (LCONSTT) THEN
    WRITE (6,918)
    WRITE (14,918)
  ELSE
    WRITE (6,915)
    WRITE (14,915)
  END IF
  WRITE (94,949)
  RETURN
192 CONTINUE
C * ----- SECTION 2 ---- START OF SWEEP.
  RETURN
193 CONTINUE
C * ----- SECTION 3 ---- START OF IZ SLAB.
  RETURN
194 CONTINUE
C * ----- SECTION 4 ---- START OF ITERATION.
  RETURN
195 CONTINUE
C * ----- SECTION 5 ---- FINISH OF ITERATION.
  RETURN
196 CONTINUE
C * ----- SECTION 6 ---- FINISH OF IZ SLAB.
C
C  GET VALUES AT FIRST CELL
C
  IF (IZ.EQ.IZFRST) THEN
    CALL GETYX(P1,P1FRST,NYD,NXD)
    CALL GETYX(V1,V1FRST,NYD,NXD)
    V1FRST(NY,1) = V1FRST(NY-1,1)
    CALL GETYX(H1,H1FRST,NYD,NXD)
    CALL GETYX(W1,W1FRST,NYD,NXD)
C
C  CALCULATE DOWNSTREAM VALUES.
C
  IF (.NOT.LINLINE) THEN
    CALL SETCX(Y,VALU,P1FRST,NYD,NXD,1.,-1.*PFRST,NY)
    CALL SWCHY(P1DOWN,VALU,NYD,NXD,NY,NX)
    CALL SWCHY(W1OUT,W1FRST,NYD,NXD,NY,NX)
    CALL SWCHY(VALU,H1FRST,NYD,NXD,NY,NX)
    CALL SETCX(Y,H1OUT,VALU,NYD,NXD,1./C1HEAT,
$-1.*C2HEAT/C1HEAT,NY)
    IF (.NOT.LSHIFT) THEN
      CALL SETCX(Y,VALU,V1FRST,NYD,NXD,-1.,0.,NY-1)
      CALL SWCHY(V1OUT,VALU,NYD,NXD,NY-1,NX)
    ELSE
      CALL SETCX(Y,VALU,V1FRST,NYD,NXD,-1.,0.,NY)
      CALL SWCHY(V1OUT,VALU,NYD,NXD,NY,NX)
      CALL SHFTY(V1OUT,IYL1-1,NYD,NXD,NY,NX)
      CALL SHFTY(H1OUT,IYL1,NYD,NXD,NY,NX)
```

```

CALL SHFTY(W1OUT,IYL1,NYD,NXD,NY,NX)
CALL SHFTY(P1DOWN,IYL1,NYD,NXD,NY,NX)
END IF
ELSE
CALL SETCX(Y,VALU,V1FRST,NYD,NXD,-1.,0.,NY)
CALL SWCHY(V1OUT,VALU,NYD,NXD,NY,NX)
CALL SETCX(Y,P1DOWN,P1FRST,NYD,NXD,1.,-1.*PFRST,NY)
CALL SETCX(Y,V1OUT,V1FRST,NYD,NXD,1.,0.,NY-1)
CALL SETCX(Y,W1OUT,W1FRST,NYD,NXD,1.,0.,NY)
CALL SETCX(Y,H1OUT,H1FRST,NYD,NXD,
$1./C1HEAT,-1.*C2HEAT/C1HEAT,NY)
END IF
C
C   COMPUTE REFERENCE PRESSURE FOR PRESSURE COEFFICIENT (I-GRIDS)
C
IF (LIGRID) THEN
IYREF = 1
IF (LDCONFIG) IYREF = NY
IXREF = 1
CPREF = P1FRST(IYREF,IXREF)
CPREF = 2.*CPREF/(RHO1*WB*WB)
WMIDL = 0.
TMIDL = 0.
C
C   COMPUTE VELOCITY AT MINIMUM CROSS-SECTION (DIAGONAL I-GRIDS ONLY)
C
DO 1917 J = IYL1, IYL2
DO 1917 I = 1, NX
WMIDL = WMIDL + AREF(J,I)*W1LAST(J,I)
TMIDL = TMIDL + AREF(J,I)*W1LAST(J,I)*H1FRST(J,I)
1917 CONTINUE
IF (ABS(WMIDL).GT.1.E-06) TMIDL = TMIDL/(CFLUID*WMIDL)
IF (TMIDL.EQ.TWALL) TMIDL = 1.E-06
END IF
END IF
IF (.NOT.LINLINE) TMIDL = (TBIN+TBLAST)/2.
C
C   CALCULATE BULK VELOCITY AND TEMPERATURE AT NZ/2
C
IF (IZ.EQ.(NZ-1)/2) THEN
CALL GETYX(W1,W1MIDL,NYD,NXD)
IF (STORE(H1)) CALL GETYX(H1,H1MIDL,NYD,NXD)
WMIDL = 0.
TMIDL = 0.
DO 1918 J = 1, NY
DO 1918 I = 1, NX
WMIDL = WMIDL + ASMIDL(J,I)*W1MIDL(J,I)
TMIDL = TMIDL + ASMIDL(J,I)*W1MIDL(J,I)*H1MIDL(J,I)
1918 CONTINUE
IF (ABS(WMIDL).GT.1.E-06) TMIDL = TMIDL/(CFLUID*WMIDL)
IF (TMIDL.EQ.TWALL) TMIDL = 1.E-06
END IF
IF (.NOT.LINLINE) TMIDL = (TBIN+TBLAST)/2.
C
C   COMPUTE REFERENCE PRESSURE FOR PRESSURE COEFFICIENT (H-GRIDS)
C
IF (LHGRID) THEN

```

```

      IF (((.NOT.LICONFIG).AND.(IZ.EQ.IZ2)).
$OR.(LICONFIG.AND.(IZ.EQ.IZ1))) THEN
      IYREF = 1
      IF (LSTAGH) IYREF = NY
      IXREF = NX
      IF (STORE(P1)) CALL GETONE(P1,CPREF,IYREF,IXREF)
      CPREF = 2.*CPREF/(RHO1*WB*WB)
      END IF
      END IF
C
C   GET VALUES AT LAST CELL
C
      IF (IZ.EQ.IZLAST) THEN
      IF (NX.GT.1) CALL GETYX(U1,U1LAST,NYD,NXD)
      IF (NY.GT.1) CALL GETYX(V1,V1LAST,NYD,NXD)
      V1LAST(NY,1) = V1LAST(NY-1,1)
      IF (NZ.GT.1) CALL GETYX(W1,W1LAST,NYD,NXD)
      IF (STORE(H1)) CALL GETYX(H1,H1LAST,NYD,NXD)
C
C   CALCULATE INLET BOUNDARY VALUES
C
      IF (.NOT.LINLINE) THEN
      CALL SWCHY(W1IN,W1LAST,NYD,NXD,NY,NX)
      CALL SWCHY(VALU,H1LAST,NYD,NXD,NY,NX)
      CALL SETCX(Y(H1IN,VALU,NYD,NXD,C1HEAT,C2HEAT,NY)
      IF (.NOT.LSHIFT) THEN
      CALL SETCX(Y(VALU,V1LAST,NYD,NXD,-1.,0.,NY-1)
      CALL SWCHY(V1IN,VALU,NYD,NXD,NY-1,NX)
      ELSE
      CALL SETCX(Y(VALU,V1LAST,NYD,NXD,-1.,0.,NY)
      CALL SWCHY(V1IN,VALU,NYD,NXD,NY,NX)
      CALL SHFTY(V1IN,IYL1-1,NYD,NXD,NY,NX)
      CALL SHFTY(W1IN,IYL1,NYD,NXD,NY,NX)
      CALL SHFTY(H1IN,IYL1,NYD,NXD,NY,NX)
      END IF
      ELSE
      CALL SETCX(Y(V1IN,V1LAST,NYD,NXD,1.,0.,NY-1)
      CALL SETCX(Y(W1IN,W1LAST,NYD,NXD,1.,0.,NY)
      CALL SETCX(Y(H1IN,H1LAST,NYD,NXD,C1HEAT,C2HEAT,NY)
      END IF
      IF (IYF1.GT.1) THEN
      DO 1919 II = 1, NX
      DO 1919 JJ = 1, IYF1-1
      W1IN(JJ,II) = CONIN(IYF1,II)
1919 CONTINUE
      END IF
C
C   CALCULATE BULK VELOCITY AND TEMPERATURE BASED ON EXIT VALUES
C
C   CONOUT      = MASS FLUX OUT OF LAST CELL
C   CONFRST     = MASS FLUX OUT OF FIRST CELL
C   CONFRST     = MASS FLUX OUT OF FIRST CELL
C   CONHALO     = MASS FLUX OUT OF HALO CELL
C
C   WB          = BULK VELOCITY AT CURRENT SWEEP
C   TBLAST      = BULK TEMPERATURE AT LAST CELL
C

```

```
WB = 0.
TBLAST = 0.0
DO 1920 J = IYL1, IYL2
DO 1920 I = 1, NX
CONOUT(J,I) = GMOUT(J,I)*V1LAST(J,I)
1+ HMOUT(J,I)*W1LAST(J,I)
CONFRST(J,I) = GMFRST(J,I)*V1FRST(J,I)
1+ HMFRST(J,I)*W1FRST(J,I)
WB = WB + AREF(J,I)*CONOUT(J,I)
TBLAST = TBLAST+AREF(J,I)*CONOUT(J,I)*H1LAST(J,I)
1920 CONTINUE
WB = WB/ACTOT
TBLAST = TBLAST/(CFLUID*WB*ACTOT)
CMASS = W0/WB
WB = WB*RESCALE
TBMN = (TBIN + TBLAST)/2.
C
C  COMPUTE INLET CONVECTION FLUX
C
IF (.NOT.LINLINE) THEN
CALL SWCHY(CONIN,CONOUT,NYD,NXD,NY,NX)
CALL SWCHY(CONHALO,CONFRST,NYD,NXD,NY,NX)
IF (LSHIFT) THEN
CALL SHFTY(CONIN,IYF1,NYD,NXD,NY,NX)
CALL SHFTY(CONHALO,IYL1,NYD,NXD,NY,NX)
END IF
ELSE
CALL SETCXY(CONIN,CONOUT,NYD,NXD,1.,0.,NY)
CALL SETCXY(CONHALO,CONFRST,NYD,NXD,1.,0.,NY)
END IF
C
C  COMPUTE ENTHALPY COEFFICENTS
C
IF (LCONSTT) THEN
IF (ABS(TBLAST - TWALL).LT.1.E-06) THEN
TBLAST = TWALL + 1.E-06*(TBIN - TWALL)
C1HEAT = TBIN/(TWALL + 1.E-06)
C2HEAT = 1.e-06
ELSE
C1HEAT = (TBIN - TWALL)/(TBLAST - TWALL)
C2HEAT = CFLUID*TWALL*(1. - C1HEAT)
END IF
TFU = TWALL + (TBIN - TWALL)*(TMIDL - TWALL)/(TBLAST - TWALL)
ELSE
C1HEAT = 1.
C2HEAT = CFLUID*(TBIN - TBLAST)
TFU = TMIDL - QWALL*PI/(RHO1*(A-1))
END IF
C
END IF
C
C  GET VALUES AT HALO CELL
C
IF (IZ.EQ.IZHALO) THEN
CALL GETYX(P1,P1HALO,NYD,NXD)
CALL GETYX(V1,V1HALO,NYD,NXD)
C
```

```
C  COMPUTE MEAN UPSTREAM AND DOWNSTREAM PRESSURES
C
  PFRST = 0.
  PLAST = 0.
  RSUM = FLOAT((IYL2+1-IYL1)*NX)
  IF (LINLINE) THEN
    DO 1921 J = IYL1, IYL2
    DO 1921 I = 1, NX
      PFRST = PFRST + P1FRST(J,I)
      PLAST = PLAST + P1HALO(J,I)
    1921 CONTINUE
    PFRST = PFRST/RSUM
    PLAST = PLAST/RSUM
  ELSE
    DO 1922 J = IYF1, IYF2
    DO 1922 I = 1, NX
      PFRST = PFRST + P1FRST(J,I)
      PLAST = PLAST + P1HALO(J,I)
    1922 CONTINUE
    PFRST = PFRST/RSUM
    PLAST = PLAST/RSUM
  END IF
  END IF

C
C  ----- THIS SECTION EXECUTED ONLY FOR ISWEEP = LSWEEP -----
C  FOR ALL VALUES OF IZ
C
  IF (ISWEEP.EQ.LSWEEP) THEN
C
  DHBAR = CFLUID*(TBLAST-TBIN)
  ITHTA = INAME('THTA')
  IF (STORE(ITHTA))
    $CALL FN2(ITHTA,H1,-1.*CFLUID*TBIN/DHBAR,1./DHBAR)
C
C  CALCULATE LOCAL PRESSURE COEFFICIENT IN Z AND ZCRT DIRECTIONS.
c
  DO 1923 II = 1, NX
    IF (STORE(P1)) CALL GETONE(P1,PWALL1(IZ,II),1,II)
    IF (NZ.GT.1) CALL GETONE(W1,W1WALL1(IZ,II),1,II)
    IF (STORE(H1)) CALL GETONE(H1,H1WALL1(IZ,II),1,II)
    IF (LDCONFIG) THEN
      IF (STORE(P1)) CALL GETONE(P1,PWALL2(IZ,II),NY,II)
      IF (NZ.GT.1) CALL GETONE(W1,W1WALL2(IZ,II),NY,II)
      IF (STORE(H1)) CALL GETONE(H1,H1WALL2(IZ,II),NY,II)
    ELSE
      IF (STORE(P1)) CALL GETONE(P1,PWALL2(IZ,II),1,II)
      IF (NZ.GT.1) CALL GETONE(W1,W1WALL2(IZ,II),1,II)
      IF (STORE(H1)) CALL GETONE(H1,H1WALL2(IZ,II),II,1)
    END IF
    IF (IZ.EQ.NZ) W1WALL1(IZ,II) = 0.
    IF (IZ.EQ.NZ) W1WALL2(IZ,II) = 0.
    IF ((IZ.EQ.NZ-1).AND.LIGRID) W1WALL1(IZ,II) = 0.
  1923 CONTINUE
C
C  CALCULATE LOCAL COEFFICIENTS
C
C  THREE DIFFERENT ZONES ARE CONSIDERED DEPENDING ON THE GRID
```

```

C  TYPE AND REFERENCE FLUID VALUES (BULK VELOCITY ETC)
C
C  ZONE1 = T FOR H-GRIDS AND STAGGERED I-GRIDS
C  WHEN THERE IS SOLID WALL AT IY = 1 (90 < PHI < 180)
C  ZONE2 = T FOR STAGGERED H- AND I-GRIDS
C  WHEN THERE IS SOLID WALL AT IY = NY (0 < PHI < 90)
C  ZONE3 = T FOR IN-LINE I-GRIDS
C  WHEN THERE IS SOLID WALL AT IY =1 (0 < PHI < 180)
C
C  LOCAL COEFFICIENTS ARE STORED IN CP1, CF1, ETC.
C  IF LZONE1 = T OR LZONE2 = T
C  OR ELSE IN CP2, CF2, ETC. IF LZONE3 = T
C
C  ALL COEFFICIENTS ARE CALCULATED AND OUTPUT FOR BOTH
C  CONSTANT QWALL AND CONSTANT TWALL (REGARDLESS OF WHETHER
C  OR NOT THEY ARE VALID)
C
C  PHI      = ANGLE FROM LEADING EDGE OF CYLINDER
C  TWALL    = WALL TEMPERATURE
C  QWALL    = WALL HEAT FLUX
C  CPREF    = PRESSURE AT REFERENCE CELL
C
C  CP1 (CP2 = LOCAL PRESSURE COEFFICIENT
C  CF1      = LOCAL FRICTION COEFFICIENT
C  CFZ1     = STREAM-WISE COMPONENT OF LOCAL FRICTION COEFFICIENT
C  CDRP     = PRESSURE COMPONENT OF DRAG COEFFICIENT
C  CDRF     = FRICTION COMPONENT OF DRAG COEFFICIENT
C  NUF1     = LOCAL NUSSELT NUMBER BASED ON TF
C  NULM1    = LOCAL NUSSELT NUMBER BASED ON TF (CONST TWALL ONLY)
C  NUM1     = LOCAL NUSSELT NUMBER BASED ON TM
C  NUUM1    = LOCAL NUSSELT NUMBER BASED ON TUM
C  NUMN1    = LOCAL NUSSELT NUMBER BASED IN (TUM + TDN)/2
C
C
C  LZONE1 = .FALSE.
C  LZONE2 = .FALSE.
C  LZONE3 = .FALSE.
c
C  IF (LHGRID) THEN
C  IF (LICONFIG) THEN
C  IF ((IZ.GE.IZ1).AND.(IZ.LE.IZ2)) LZONE3 = .TRUE.
C  ELSE
C  IF (IZ.LE.IZ1) LZONE1 = .TRUE.
C  IF ((IZ.GE.IZ2).AND.(IZ.LT.NZ)) LZONE2 = .TRUE.
C  END IF
C  ELSE IF (LIGRID) THEN
C  IF (LDCONFIG) THEN
C  LZONE1 = .TRUE.
C  LZONE2 = .TRUE.
C  ELSE
C  IF ((IZ.GE.IZ1).AND.(IZ.LE.IZ2)) LZONE3 = .TRUE.
C  END IF
C  END IF
c
C  IF (IZ.LT.NZ) THEN
C  IF (LZONE1) THEN
C  CALL VECTR(VEC,AVEC1(1,IZ),AVEC1(2,IZ),AVEC1(3,IZ))

```

```

COSPFI = DOT(VEC,ZVEC)
SINPHI = DOT(VEC,YVEC)
DO 1924 II = 1, NX
IF (.NOT.LCONSTT) THEN
TWALL = QWALL*YWALL1(IZ,II)/KFLUID + H1WALL1(IZ,II)/CFLUID
END IF
CP1(IZ,II) = 2.*PWALL1(IZ,II)/(RHO1*WB*WB) - EU
CP1(IZ,II) = 1. - CPREF + CP1(IZ,II)
CF1(IZ,II) = 2.*ENUL*W1WALL1(IZ,II)/(YWALL1(IZ,II)*WB*WB)
CFZ1(IZ,II) = CF1(IZ,II)*SINPHI
CDRP = CDRP - PWALL1(IZ,II)*COSPFI*AWALL1(IZ,II)
CDRF = CDRF + AWALL1(IZ,II)*SINPHI*RHO1*ENUL
$*W1WALL1(IZ,II)/(YWALL1(IZ,II))
TERM = DIAM*(TWALL - H1WALL1(IZ,II)/CFLUID)/YWALL1(IZ,II)
NUF1(IZ,II) = TERM/(TWALL - TFU)
NULM1(IZ,II) = TERM/DTLM
NUM1(IZ,II) = TERM/(TWALL - TBIN)
NUUM1(IZ,II) = TERM/(TWALL - TBIN)
NUMN1(IZ,II) = TERM/(TWALL - TBMN)
IF (STORE(NNUSS)) F(L0NUSS + 1 + NY*(II-1)) = NUF1(IZ,II)
IF (STORE(NCP)) F(L0NCP + 1 + NY*(II-1)) = CP1(IZ,II)
NUFBAR = NUFBAR + AWALL1(IZ,II)*NUF1(IZ,II)
NULMBAR = NULMBAR + AWALL1(IZ,II)*NULM1(IZ,II)
NUMBAR = NUMBAR + AWALL1(IZ,II)*NUM1(IZ,II)
NUUMBAR = NUUMBAR + AWALL1(IZ,II)*NUUM1(IZ,II)
NUMNBAR = NUMNBAR + AWALL1(IZ,II)*NUMN1(IZ,II)
ACYL = ACYL + AWALL1(IZ,II)
1924 CONTINUE
END IF
IF (LZONE2) THEN
CALL VECTR(VEC,AVEC2(1,IZ),AVEC2(2,IZ),AVEC2(3,IZ))
COSPFI = DOT(VEC,ZVEC)
SINPHI = DOT(VEC,YVEC)
DO 1925 II = 1, NX
IF (.NOT.LCONSTT) THEN
TWALL = QWALL*YWALL2(IZ,II)/KFLUID + H1WALL2(IZ,II)/CFLUID
END IF
CP2(IZ,II) = 2.*PWALL2(IZ,II)/(RHO1*WB*WB)
CP2(IZ,II) = 1. - CPREF + CP2(IZ,II)
CF2(IZ,II) = 2.*ENUL*W1WALL2(IZ,II)/(YWALL2(IZ,II)*WB*WB)
CFZ2(IZ,II) = CF2(IZ,II)*SINPHI
CDRP = CDRP - PWALL2(IZ,II)*COSPFI*AWALL2(IZ,II)
CDRF = CDRF + AWALL2(IZ,II)*SINPHI*RHO1*ENUL
$*W1WALL2(IZ,II)/(YWALL2(IZ,II))
TERM = DIAM*(TWALL - H1WALL2(IZ,II)/CFLUID)/YWALL2(IZ,II)
NUF2(IZ,II) = TERM/(TWALL - TMIDL)
NULM2(IZ,II) = TERM/DTLM
NUM2(IZ,II) = TERM/(TWALL - TBLAST)
NUUM2(IZ,II) = TERM/(TWALL - TBIN)
NUMN2(IZ,II) = TERM/(TWALL - TBMN)
IF (STORE(NNUSS)) F(L0NUSS + NY + NY*(II-1)) = NUF2(IZ,II)
IF (STORE(NCP)) F(L0NCP + NY + NY*(II-1)) = CP2(IZ,II)
NUFBAR = NUFBAR + AWALL2(IZ,II)*NUF2(IZ,II)
NULMBAR = NULMBAR + AWALL2(IZ,II)*NULM2(IZ,II)
NUMBAR = NUMBAR + AWALL2(IZ,II)*NUM2(IZ,II)
NUUMBAR = NUUMBAR + AWALL2(IZ,II)*NUUM2(IZ,II)
NUMNBAR = NUMNBAR + AWALL2(IZ,II)*NUMN2(IZ,II)

```



```
      ACYL = ACYL + AWALL2(IZ,II)
1925 CONTINUE
      END IF
      IF (LZONE3) THEN
        CALL VECTR(VEC,AVEC1(1,IZ),AVEC1(2,IZ),AVEC1(3,IZ))
        COSPHI = DOT(VEC,ZVEC)
        SINPHI = DOT(VEC,YVEC)
        DO 1926 II = 1, NX
          IF (.NOT.LCONSTT) THEN
            TWALL = QWALL*YWALL1(IZ,II)/KFLUID + H1WALL1(IZ,II)/CFLUID
          END IF
          CP1(IZ,II) = 2.*PWALL1(IZ,II)/(RHO1*WB*WB)
          CP1(IZ,II) = 1. - CPREF + CP1(IZ,II)
          CF1(IZ,II) = 2.*ENUL*W1WALL1(IZ,II)/(YWALL1(IZ,II)*WB*WB)
          CFZ1(IZ,II) = CF1(IZ,II)*SINPHI
          CDRP = CDRP - PWALL1(IZ,II)*COSPHI*AWALL1(IZ,II)
          CDRF = CDRF + AWALL1(IZ,II)*SINPHI*RHO1*ENUL
          $*W1WALL1(IZ,II)/(YWALL1(IZ,II))
          TERM = DIAM*(TWALL - H1WALL1(IZ,II)/CFLUID)/YWALL1(IZ,II)
          NUF1(IZ,II) = TERM/(TWALL - TBIN)
          NULM1(IZ,II) = TERM/DTLM
          NUM1(IZ,II) = TERM/(TWALL - TMIDL)
          NUMN1(IZ,II) = TERM/(TWALL - TBMN)
          NUUM1(IZ,II) = TERM/(TWALL - TMIDL)
          IF (STORE(NNUSS)) F(LONUSS + NY + NY*(II-1)) = NUF1(IZ,II)
          IF (STORE(NCP)) F(LONCP + NY + NY*(II-1)) = CP1(IZ,II)
          NUFBAR = NUFBAR + AWALL1(IZ,II)*NUF1(IZ,II)
          NULMBAR = NULMBAR + AWALL1(IZ,II)*NULM1(IZ,II)
          NUMBAR = NUMBAR + AWALL1(IZ,II)*NUM1(IZ,II)
          NUUMBAR = NUUMBAR + AWALL1(IZ,II)*NUUM1(IZ,II)
          NUMNBAR = NUMNBAR + AWALL1(IZ,II)*NUMN1(IZ,II)
          ACYL = ACYL + AWALL1(IZ,II)
        1926 CONTINUE
      END IF
      END IF
      RETURN
197 CONTINUE
C   * ----- SECTION 7 ---- Finish of sweep.
C
C   COMPUTE DOWNSTREAM PRESSURE FIELD USING LINEAR RELAXATION
C
      DO 1927 I = 1, NX
        DO 1927 J = IYL1, IYL2
          PVAL(J,I) = (1.-AMYRLX)*P1HALO(J,I)
          $+ AMYRLX*P1DOWN(J,I)
991 FORMAT (I10,G13.4)
1927 CONTINUE
      WRITE (93,*) ISWEEP, PVAL(NY,1)
      IDUM = INT(ISWEEP/TSTSWP)
      RDUM = FLOAT(ISWEEP)/FLOAT(TSTSWP)
      IF (RDUM.EQ.FLOAT(IDUM)) THEN
C
C   CALCULATE RE, EU, NUBAR
C
      RE = WB*DIAM/ENUL
      EU = 2.0*PFRST/(RHO1*WB*WB)
```

```
      IF (ABS(TBLAST-TWALL).GT.0.)
      $LTITO = (TBIN-TWALL)/(TBLAST-TWALL)
      IF(ABS(LTITO).GT.0) THEN
      LTITO = LOG(LTITO)
      IF(ABS(LTITO).GT.0) THEN
      DTLM = (TBLAST - TBIN)/LTITO
      NUBAR = RE*PRNDTL(H1)*(A-1.)*LTITO/PI
      END IF
      ELSE
      NUBAR = 1.E-20
      LTITO = 1.E-20
      DTLM = 1.E-20
      END IF
C
C  OUTPUT DATA TO SCREEN AND RESULT FILE.
C
910 FORMAT (/,' ISWEEP = ',I4)
913 FORMAT (' RE = ',G11.4,' RESCALE =', G11.4,
1' PR      = ',G11.4)
916 FORMAT (' EU = ',G11.4,
1' NUBAR   = ',G11.4,/)
953 FORMAT (' PFRST = ',G11.4,
1'PLAST = ',G11.4,' TBO - TBI = ',G11.4)
952 FORMAT (' LOG(TW-TI)/(TW-TBLAST) = ',G11.4,
1' DTLM   = ',G11.4)
954 FORMAT (' A*   = ',G11.4,' CMASS = ',G11.4,/)
955 FORMAT (' TFU = ',G11.4,
1' TMIDL = ',G11.4)
956 FORMAT (I4,2X,3(G11.4,1X))
      WRITE(6,910) ISWEEP
      WRITE(6,913) RE, RESCALE, PRNDTL(H1)
      WRITE(6,916) EU, NUBAR
      WRITE(6,953) PFRST, PLAST, TBLAST-TBIN
      WRITE(6,952) LTITO, DTLM
      WRITE(6,954) ATOT,CMASS
      WRITE(6,955) TFU, TMIDL
      WRITE(14,910) ISWEEP
      WRITE(14,913) RE, RESCALE, PRNDTL(H1)
      WRITE(14,916) EU, NUBAR
      WRITE(14,953) PFRST, PLAST, TBLAST-TBIN
      WRITE(14,952) LTITO, DTLM
      WRITE(14,954) ATOT, CMASS
      WRITE(14,955) TFU, TMIDL
      WRITE(94,956) ISWEEP, EU, NUBAR, RE
      END IF
      RETURN
198 CONTINUE
C  * ----- SECTION 8 --- Finish of time step.
C
C  AVERAGE COEFFICIENTS COMPUTED FROM LOCAL COEFFICIENTS
C
      NUFBAR = NUFBAR/ACYL
      NUMBAR = NUMBAR/ACYL
      NUUMBAR = NUUMBAR/ACYL
      NULMBAR = NULMBAR/ACYL
      NUMNBAR = NUMNBAR/ACYL
      CDRP = 4.*CDRP/(DIAM*RHO1*WB*WB)
```

```
CDRF = 4.*CDRF/(DIAM*RHO1*WB*WB)
CDR = CDRP + CDRF
C
C  OUTPUT LOCAL COEFFICIENTS TO VDU AND RESULT FILE
C
935 FORMAT(//,' LOCAL COEFFICIENTS',//,' CDRP =',
$G10.4,' CDRF =',G10.4,' CDR =',G10.4)
933 FORMAT(///,' TLM-EBAL',9X,'TF ',6X,'TM ',8X,
$'TLM',6X,'TUUM',6X,'TMEAN')
944 FORMAT(A,6(1X,E10.4))
WRITE (6,935) CDRP, CDRF, CDR
WRITE (14,935) CDRP, CDRF, CDR
WRITE (6,933)
WRITE (14,933)
WRITE (6,944) 'NU', NUBAR, NUFBAR, NUMBAR,
$NULMBAR, NUUMBAR, NUMNBAR
WRITE (14,944) 'NU', NUBAR, NUFBAR, NUMBAR,
$NULMBAR, NUUMBAR, NUMNBAR
PP = 1/(RE*PRNDTL(H1))
C  STANTON NUMBERS
WRITE (6,944) 'ST',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR,PP*NUMNBAR
WRITE (14,944) 'ST',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
C  COLBURN J FACTORS
PP = PRNDTL(H1)**(2./3.)*PP
WRITE (6,944) 'J ',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
WRITE (14,944) 'J ',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
C  BERGELIN B FACTORS
PP = PRNDTL(H1)**(-1./3.)
WRITE (6,944) 'B ',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
WRITE (14,944) 'B ',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
C  ZUKAUSKAS K FACTORS
PP = PRNDTL(H1)**(-.36)
WRITE (6,944) 'K ',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
WRITE (14,944) 'K ',PP*NUBAR, PP*NUFBAR, PP*NUMBAR,
$PP*NULMBAR, PP*NUUMBAR, PP*NUMNBAR
C
C  WRITE LOCAL COEFFICIENTS IN AUTO PLOT FORMAT TO FILE "PLOT"
C
1972 FORMAT(' DATA SAVED ON PLOT')
934 FORMAT(1X,I4,7(1X,E10.4))
WRITE(98,*) 'IZ PHI CF CP CFZ NULM NUF NUM'
IF (LHGRID) THEN
IF (.NOT.LICONFIG) THEN
DO 1928 II = 1, NX
DO 1928 KK = IZ2, NZ-1
WRITE(98,934) KK, PHI2(KK,II), CF2(KK,II), CP2(KK,II),
$CFZ2(KK,II), NULM2(KK,II)/NULMBAR, NUF2(KK,II)/NUFBAR,
$NUM2(KK,II)/NUMNBAR
1928 CONTINUE
DO 1929 II = 1, NX
```

```
DO 1929 KK = 1, IZ1
WRITE(98,934) KK, PHI1(KK,II), CF1(KK,II), CP1(KK,II),
$CFZ1(KK,II), NULM1(KK,II)/NULMBAR, NUF1(KK,II)/NUFBAR,
$NUM1(KK,II)/NUMBAR
1929 CONTINUE
ELSE
DO 1930 II = 1, NX
DO 1930 KK = IZ1, IZ2
WRITE(98,934) KK, PHI1(KK,II), CF1(KK,II), CP1(KK,II),
$CFZ1(KK,II), NULM1(KK,II)/NULMBAR, NUF1(KK,II)/NUFBAR,
$NUM1(KK,II)/NUMBAR
1930 CONTINUE
END IF
ELSE IF (LIGRID) THEN
IF (LDCONFIG) THEN
DO 1931 II = 1, NX
DO 1931 KK = 1, NZ-1
WRITE(98,934) KK, PHI2(KK,II), CF2(KK,II), CP2(KK,II),
$CFZ2(KK,II), NULM2(KK,II)/NULMBAR, NUF2(KK,II)/NUFBAR,
$NUM2(KK,II)/NUMBAR
1931 CONTINUE
DO 1932 II = 1, NX
DO 1932 KK = 1, NZ-1
WRITE(98,934) KK, PHI1(KK,II), CF1(KK,II), CP1(KK,II),
$CFZ1(KK,II), NULM1(KK,II)/NULMBAR, NUF1(KK,II)/NUFBAR,
$NUM1(KK,II)/NUMBAR
1932 CONTINUE
ELSE
DO 1933 II = 1, NX
DO 1933 KK = IZ1, IZ2
WRITE(98,934) KK, PHI1(KK,II), CF1(KK,II), CP1(KK,II),
$CFZ1(KK,II), NULM1(KK,II)/NULMBAR, NUF1(KK,II)/NUFBAR,
$NUM1(KK,II)/NUMBAR
1933 CONTINUE
END IF
END IF
CLOSE(93,STATUS='KEEP')
CLOSE(94,STATUS='KEEP')
CLOSE(98,STATUS='KEEP')
WRITE (6,1972)
RETURN
C*****
SUBROUTINE SHFTY(A,JSTART,NY,NX,IYLAST,IXLAST)
REAL A(NY,NX)
INTEGER NY, NX, I, J, IYLAST, IXLAST
c
DO 10 I = 1, IXLAST
DO 10 J = IYLAST,JSTART,-1
A(J,I) = A(J-JSTART+1,I)
10 CONTINUE
IF (JSTART.GT.1) THEN
DO 20 I = 1, IXLAST
DO 20 J = JSTART-1,1,-1
A(J,I) = 1.E-20
20 CONTINUE
END IF
IF (IYLAST.LT.NY) THEN
```

```
      DO 30 I = 1, IXLAST
      DO 30 J = NY,IYLAST+1, -1
      A(J,I) = A(IYLAST,I)
30  CONTINUE
      END IF
      RETURN
      END
C
      SUBROUTINE SWCHY(AUP,ADOWN,NY,NX,IYLAST,IXLAST)
      REAL ADOWN(NY,NX), AUP(NY,NX)
      INTEGER NY, NX, I, J, IYLAST, IXLAST
C
      DO 10 J = 1, IYLAST
      DO 10 I = 1, NX
      AUP(IYLAST-J+1,I) = ADOWN(J,I)
10  CONTINUE
      IF (IYLAST.LT.NY) THEN
      DO 20 J = IYLAST+1, NY
      DO 20 I = 1, NX
      AUP(J,I) = AUP(IYLAST,I)
20  CONTINUE
      END IF
      RETURN
      END
C
      SUBROUTINE SETCX(Y,X,NY,NX,M,C,IYLAST)
C   Y = MX + C
      REAL Y(NY,NX), X(NY,NX)
      REAL M, C
      INTEGER NY, NX, I, J
C
      DO 10 J = 1, IYLAST
      DO 10 I = 1, NX
      Y(J,I) = M*X(J,I) + C
10  CONTINUE
      DO 20 J = IYLAST+1,NY
      DO 20 I = 1, NX
      Y(J,I) = Y(IYLAST,I)
20  CONTINUE
      RETURN
      END
C
      SUBROUTINE VECTR(V,P,Q,R)
      REAL V(3)
      V(1) = P
      V(2) = Q
      V(3) = R
      RETURN
      END
C
      SUBROUTINE NORM2(PT1,PT2,PT3,PT4,VEC)
      REAL PT1(3),PT2(3),PT3(3),PT4(3),VEC(3)
      REAL V1(3), V2(3)
      DO 10 I = 1, 3
      V1(I) = PT2(I) - PT1(I)
      V2(I) = PT4(I) - PT1(I)
10  CONTINUE
```

```
CALL CROSS(V1,V2,VEC)
CALL NORM(VEC)
RETURN
END
```

A3.5 Unsteady flow

Q1 FILE

```
TALK=F;RUN( 1, 1);VDU=X11-TERM
```

GROUP 1. Run title and other preliminaries

```
TEXT(UNSTEADY FLOW)
```

```
** NCYCS  = TOTAL NUMBER OF CYCLES AT SH = SH0
** IFDUMP = FIRST TIME STEP PHIDA FILE CREATED
** NDUMP  = FREQUENCY PHIDA FILES CREATED
** IT0    = NUMBER OF INITIAL TIME STEPS
** IXLAT  = LOCATION OF INITIAL DISTURBANCE
** V0     = BULK VELOCITY
** RE     = PRESCRIBED REYNOLDS' NUMBER
** SH0    = ASSUMED STROUHAL NUMBER
** DIAM   = CYLINDER DIAMETER
** FREQ   = ASSUMED FREQUENCY
** AMP    = APPLIED DISTURBANCE AT CURRENT TIME STEP
** ARG    = ARGUMENT OF SINUSOIDAL DISTURBANCE
** FALST  = INERTIAL RELAXATION FACTOR
```

```
INTEGER(NCYCS,IFDUMP,NDUMP,IT0, IXLAT)
REAL(V0,RE,SH0,DIAM,FREQ,PI,AMP,ARG)
REAL(FALST)
BOOLEAN(INLINE, LOCOSAV)
LOCOSAV = F
PI = 3.14159;DIAM=1;V0 = 1.
INLINE=T;RE = 300.
SH0 = .2
NX = 80;NY=161
IXLAT = NX
IF (INLINE) THEN
SH0 = .1
IXLAT = NX/2
ENDIF
FREQ = SH0*V0/DIAM
GROUP 2. TRANSCIENCE TIME-STEP SPECIFICATION
STEADY=F
LSTEP=640;NCYCS = 8
IF (INLINE) THEN
LSTEP=960;NCYCS = 12
ENDIF
GRDPWR(T,LSTEP,NCYCS*DIAM/(V0*SH0),1.)
IT0 = 3*LSTEP/NCYCS
GROUP 3. X-DIRECTION GRID SPECIFICATION
```

GRDPWR(X,NX,1,1.)
GROUP 4. Y-DIRECTION GRID SPECIFICATION
GRDPWR(Y,NY,1,1.)
GROUP 5. Z-DIRECTION GRID SPECIFICATION
GRDPWR(Z,1,1,1.)
GROUP 6. Body-fitted coordinates or grid distortion
BFC=T;NONORT=T;
READCO(GRID)
XCYIZ(1,NZ,T)
GROUP 7. VARIABLES STORED, SOLVED & NAMED
SOLVE(P1,U1,V1,H1)
SOLUTN(P1,Y,Y,Y,N,N,N)
SOLUTN(V1,Y,Y,N,Y,N,N)
SOLUTN(U1,Y,Y,N,Y,N,N)
SOLUTN(H1,Y,Y,N,Y,N,N)
** S1 AND SOR1 ARE USED TO COMPUTE STREAM FUNCTIONS
STORE(S1,SOR1)
GROUP 8. TERMS (IN DIFFERENTIAL EQUATIONS) & DEVICES
TERMS(H1,N,P,P,P,P,P)
GROUP 9. PROPERTIES OF THE MEDIUM (OR MEDIA)
ENUL = 1./RE;RHO1=1.0
GROUP 10. INTER-PHASE-TRANSFER PROCESSES AND PROPERTIES
GROUP 11. INITIALIZATION OF VARIABLE OR POROSITY FIELDS

IF (INLINE) THEN

CONPOR(0.0,CELL,-39,-42,-1,-26,1,NZ)
CONPOR(0.0,CELL,-39,-42,-55,-106,1,NZ)
CONPOR(0.0,CELL,-39,-42,-135,-186,1,NZ)
ELSE
CONPOR(0.0,CELL,-1,-2,-1,-24,1,NZ)
CONPOR(0.0,CELL,-79,-82,-1,-24,1,NZ)
CONPOR(0.0,CELL,-39,-42,-17,-64,1,NZ)
CONPOR(0.0,CELL,-1,-2,-57,-104,1,NZ)
CONPOR(0.0,CELL,-79,-82,-57,-104,1,NZ)
CONPOR(0.0,CELL,-39,-42,-97,-144,1,NZ)
CONPOR(0.0,CELL,-1,-2,-137,-184,1,NZ)
CONPOR(0.0,CELL,-79,-82,-137,-184,1,NZ)
ENDIF

FIINIT(P1)=0.
FIINIT(V1)=V0
FIINIT(U1)=0.
FIINIT(H1)=0.

GROUP 12. Convection and diffusion adjustments
GROUP 13. Boundary conditions and special sources

** DUMMY PATCHES (USED TO ENSURE ALLOCATION OF BFC QUANTITIES IN EARTH)

PATCH(DUM,CELL,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(DUME,EAST,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(DUMW,WEST,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(DUMN,NORTH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(DUMS,SOUTH,1,NX,1,NY,1,NZ,1,LSTEP)

** UPSTREAM BOUNDARY CONDITIONS

```
PATCH(UPSTRM1,SOUTH,1,NX,1,1,1,NZ,1,IT0)
COVAL(UPSTRM1,P1,FIXFLU,RHO1*V0)
COVAL(UPSTRM1,V1,ONLYMS,V0)
COVAL(UPSTRM1,U1,ONLYMS,0.)
COVAL(UPSTRM1,H1,ONLYMS,0.)
IF (.NOT.STEADY) THEN
PATCH(UPSTRM2,SOUTH,1,NX,1,1,1,NZ,IT0+1,LSTEP)
PATCH(UPSTRM3,SOUTH,1,NX,1,1,1,NZ-1,IT0+1,LSTEP)
COVAL(UPSTRM2,P1,FIXFLU,GRND)
COVAL(UPSTRM2,V1,ONLYMS,GRND)
COVAL(UPSTRM3,U1,FIXVAL,GRND)
COVAL(UPSTRM1,H1,ONLYMS,GRND)
ENDIF
```

**** DOWNSTREAM BOUNDARY CONDITIONS**

```
PATCH(DWNP1,SOUTH,1,NX,NY,NY,1,NZ,1,LSTEP)
COVAL(DWNP1,P1,-2.*RHO1,0.)
COVAL(DWNP1,V1,ONLYMS,SAME)
```

**** LATERAL BOUNDARY CONDITIONS**

```
IF (.NOT.STEADY) THEN
DO II = 1, IT0
ARG= 2.*PI*FREQ*TFRAC(II)*TLAST
AMP = .2*V0*SIN(ARG)
PATCH(LAT:II,CELL,IXLAT,IXLAT,1,NY/2,1,NZ,:II,:II)
COVAL(LAT:II,U1,FIXVAL,AMP)
ENDDO
PATCH(LAT,CELL,IXLAT,IXLAT,1,NY/2,1,NZ,IT0+1,LSTEP)
COVAL(LAT,U1,FIXVAL,GRND)
ENDIF
```

**** WALL BOUNDARY CONDITIONS**

```
IF (INLINE) THEN
```

```
PATCH (WW1,WEST,43,43,1,26,1,NZ,1,LSTEP)
COVAL(WW1,H1,FIXFLU,1.)
PATCH (SS1,SOUTH,39,42,27,27,1,NZ,1,LSTEP)
COVAL(SS1,H1,FIXFLU,1.)
PATCH (EE1,EAST,38,38,1,26,1,NZ,1,LSTEP)
COVAL(EE1,H1,FIXFLU,1.)
PATCH (WW2,WEST,43,43,55,106,1,NZ,1,LSTEP)
COVAL(WW2,H1,FIXFLU,1.)
PATCH (SS3,SOUTH,39,42,107,107,1,NZ,1,LSTEP)
COVAL(SS3,H1,FIXFLU,1.)
PATCH (NN1,NORTH,39,42,54,54,1,NZ,1,LSTEP)
COVAL(NN1,H1,FIXFLU,1.)
PATCH (EE2,EAST,38,38,55,106,1,NZ,1,LSTEP)
COVAL(EE2,H1,FIXFLU,1.)
PATCH (WW3,WEST,43,43,135,186,1,NZ,1,LSTEP)
COVAL(WW3,H1,FIXFLU,1.)
PATCH (NN3,NORTH,39,42,134,134,1,NZ,1,LSTEP)
COVAL(NN3,H1,FIXFLU,1.)
PATCH (EE3,EAST,38,38,135,186,1,NZ,1,LSTEP)
```


COVAL(EF3,H1,FXFLU,1.)

ELSE

PATCH (WW1,WEFT,3,3,1,24,1,NZ,1,LSTEP)
COVAL(WW1,H1,FXFLU,1.)
PATCH (SS1,SOUTH,1,2,25,25,1,NZ,1,LSTEP)
COVAL(SS1,H1,FXFLU,1.)
PATCH (EE1,EAST,78,78,1,24,1,NZ,1,LSTEP)
COVAL(EE1,H1,FXFLU,1.)
PATCH (SS2,SOUTH,79,82,25,25,1,NZ,1,LSTEP)
COVAL(SS2,H1,FXFLU,1.)
PATCH (WW2,WEFT,43,43,17,64,1,NZ,1,LSTEP)
COVAL(WW2,H1,FXFLU,1.)
PATCH (EE2,EAST,38,38,17,64,1,NZ,1,LSTEP)
COVAL(EE2,H1,FXFLU,1.)
PATCH (SS3,SOUTH,39,42,65,65,1,NZ,1,LSTEP)
COVAL(SS3,H1,FXFLU,1.)
PATCH (NN1,NORTH,39,42,16,16,1,NZ,1,LSTEP)
COVAL(NN1,H1,FXFLU,1.)
PATCH (WW3,WEFT,3,3,57,104,1,NZ,1,LSTEP)
COVAL(WW3,H1,FXFLU,1.)
PATCH (SS4,SOUTH,1,2,105,105,1,NZ,1,LSTEP)
COVAL(SS4,H1,FXFLU,1.)
PATCH (NN2,NORTH,1,2,56,56,1,NZ,1,LSTEP)
COVAL(NN2,H1,FXFLU,1.)
PATCH (EE3,EAST,78,78,57,104,1,NZ,1,LSTEP)
COVAL(EE3,H1,FXFLU,1.)
PATCH (SS5,SOUTH,79,82,105,105,1,NZ,1,LSTEP)
COVAL(SS5,H1,FXFLU,1.)
PATCH (NN3,NORTH,79,82,56,56,1,NZ,1,LSTEP)
COVAL(NN3,H1,FXFLU,1.)
PATCH (WW4,WEFT,43,43,97,144,1,NZ,1,LSTEP)
COVAL(WW4,H1,FXFLU,1.)
PATCH (EE4,EAST,38,38,97,144,1,NZ,1,LSTEP)
COVAL(EE4,H1,FXFLU,1.)
PATCH (SS6,SOUTH,39,42,145,145,1,NZ,1,LSTEP)
COVAL(SS6,H1,FXFLU,1.)
PATCH (NN4,NORTH,39,42,96,96,1,NZ,1,LSTEP)
COVAL(NN4,H1,FXFLU,1.)
PATCH (WW5,WEFT,3,3,137,184,1,NZ,1,LSTEP)
COVAL(WW5,H1,FXFLU,1.)
PATCH (NN5,NORTH,1,2,136,136,1,NZ,1,LSTEP)
COVAL(NN5,H1,FXFLU,1.)
PATCH (EE5,EAST,78,78,137,184,1,NZ,1,LSTEP)
COVAL(EE5,H1,FXFLU,1.)
PATCH (NN6,NORTH,79,82,136,136,1,NZ,1,LSTEP)
COVAL(NN6,H1,FXFLU,1.)

ENDIF

GROUP 14. Downstream pressure for PARAB=.TRUE.

GROUP 15. Termination of sweeps

GROUP 16. Termination of iterations

LITER(P1) = 20

LSWEEP = 40

IF (STEADY) THEN

LSWEEP = 100

ENDIF

GROUP 17. Under-relaxation devices

RELAX(P1,LINRLX,,5)

FALST = 1.E+00*YVLAST/(NY*V0)

IF (INLINE) THEN

FALST = 1.E+01*YVLAST/(NY*V0)

ENDIF

RELAX(U1,FALSDT,FALST)

RELAX(V1,FALSDT,FALST)

RELAX(H1,FALSDT,FALST)

GROUP 18. Limits on variables or increments to them

VARMIN(U1) = -1.*V0;VARMAX(U1) = 1.*V0

VARMIN(V1) = -1.*V0;VARMAX(V1) = 2.*V0

GROUP 19. Data communicated by satellite to GROUND

USEGRX=F

NDUMP = 1;IFDUMP = 1

IG(1) = IT0

IG(2) = NDUMP

IG(3) = IFDUMP

LG(1) = INLINE

LG(2) = LOCOSAV

RG(1) = V0

GROUP 20. Preliminary print-out

NTPRIN=1

GROUP 21. Print-out of variables

OUTPUT(P1,N,N,N,N,Y,Y)

OUTPUT(U1,N,N,N,N,Y,Y)

OUTPUT(V1,N,N,N,N,Y,Y)

OUTPUT(H1,N,N,N,N,Y,Y)

OUTPUT(UCRT,N,N,N,N,N,N)

OUTPUT(VCRT,N,N,N,N,N,N)

OUTPUT(EPOR,N,N,N,N,N,N)

OUTPUT(S1,N,N,N,N,N,N)

OUTPUT(SOR1,N,N,N,N,N,N)

GROUP 22. Spot-value print-out

IXMON=3*NX/4;IYMON=3*(NY-1)/4

NPRMON=LSWEEP;TSTSWP=LSWEEP

GROUP 23. Field print-out and plot control

GROUP 24. Dumps for restarts

IF (.NOT.STEADY) THEN

RESTRT(ALL)

ENDIF

STOP

GROUND FILE

C THE INDEXES 1 - 5 REFER TO THE FOLLOWING LOCATIONS

```

C
C
C  |---O---5          O---5---O
C  |   |             |   |
C  |  4  |           |  O  4
C  |   |             |   |
C  |  O  3   ^      O  3  O   1 = INLET
C  |   |             |   |   2 = UPSTREAM OF TEST CYLINDER
C  |  2  |           |  O  2   3 = MIDWAY
C  |   |             |   |   4 = DOWNSTREAM OF TEST CYLINDER
C  |---O---1 FLOW  0----1---O   5 = EXIT

```

C IN-LINE STAGGERED

C 2 User dimensions own arrays here, for example:

```

C
C  PARAMETER (NXD = 80, NYD = 161, NZD = 1, NL = 112, NBUF = 250)
C  COMMON/NOFIL/NOFILE,LFIL/INFX/INFX(20)
C  REAL UT(NXD), VT(NXD), HT(NXD)
C  REAL AREF(NXD), ARR(NYD,NXD)
C  REAL UXT(NYD)
C  REAL UBUF(NXD,NBUF)
C  REAL HBUF(NXD,NBUF), VBUF(NXD,NBUF)
C  REAL UXBUF(NYD,NBUF)
C  REAL AWALL(NL), PHI(NL), YWALL(NL)
C  REAL VWALL(NL), PWALL(NL), H1WALL(NL)
C  REAL NF(NL), CF(NL), CP(NL)
C  REAL CDRP(NL), CDRF(NL), CLFP(NL), CLFF(NL)
C  REAL PT(3), PT0(3), PT1(3), PT2(3)
C  REAL KFLUID
C  CHARACTER*7 LOCOFIL
C  CHARACTER*3 CSTEP
C  LOGICAL INLINE, LOCOSAV

```

C*****

C--- GROUP 1. Run title and other preliminaries

```

C
C  1 GO TO (1001,1002),ISC
C  1001 CONTINUE

```

```

C
C  User may here change message transmitted to the VDU screen or
C  batch-run log file.
C  IF(IGR.EQ.1.AND.ISC.EQ.1) THEN
C    CALL WRYT40('GROUND file is GROUND.FTN of:  110191 ')
C    CALL WRYT40('PHOENICS version number is :  1.6  ')
C  ENDIF

```

```

C
C  CALL MAKE(AEAST)
C  PI = 4.*ATAN(1.)
C  IBUF = 0
C  IT0 = 30
C  CMASS = 1.
C  DIAM = 1.
C  QWALL = 1.

```

```

A = 2.
TRFAV = QWALL*PI/(2.*RHO1*(A-1.))
PRAN = 1.
CFLUID = 1.
KFLUID = RHO1*ENUL/PRAN
C
IF (STEADY) RETURN
C
C OPEN FILES AND WRITE HEADERS
C
C REF PLOT CONTAINS DATA FROM SEVERAL MONITOR
C POINTS AT EACH TIME STEP
C
OPEN(90,FILE= 'REFPLOT',ACCESS='SEQUENTIAL',STATUS='UNKNOWN'
1,FORM='FORMATTED',IOSTAT=IOS)
IF (IOS.NE.0) THEN
WRITE (14,*) ' ERROR OPENING FILE REFPLOT'
WRITE (6,*) ' ERROR OPENING FILE REFPLOT'
STOP
END IF
WRITE (90,*) ' RUN T U1 U2 U3 U4 U5 V1 V2 V3 V4 V5 P2 P4 HB2 HB3'
CLOSE(90,STATUS='KEEP')
c
C
C TIMPLOT CONTAINS DATA FROM EACH TIME STEP
C
OPEN(91,FILE= 'TIMPLOT',ACCESS='SEQUENTIAL',STATUS='UNKNOWN'
1,FORM='FORMATTED',IOSTAT=IOS)
IF (IOS.NE.0) THEN
WRITE (14,*) ' ERROR OPENING FILE TIMPLOT'
WRITE (6,*) ' ERROR OPENING FILE TIMPLOT'
STOP
END IF
WRITE (91,*) ' RUN T SH2 SH4 PHSE EU CD CL NUF'
CLOSE(91,STATUS='KEEP')
C
C CUM PLOT CONTAINS CUMMULATIVE AVERAGES
C
OPEN(92,FILE= 'CUMPLOT',ACCESS='SEQUENTIAL',STATUS='UNKNOWN'
1,FORM='FORMATTED',IOSTAT=IOS)
IF (IOS.NE.0) THEN
WRITE (14,*) ' ERROR OPENING FILE CUMPLOT'
WRITE (6,*) ' ERROR OPENING FILE CUMPLOT'
STOP
END IF
WRITE (92,*) ' RUN T SH2 SH4 EU CPRF CD CL HB2 HB3 NUF'
CLOSE(92,STATUS='KEEP')
RETURN
1002 CONTINUE
RETURN
C*****
C
C--- GROUP 13. Boundary conditions and special sources
C
C Index for Coefficient - CO
C
C Index for Value - VAL
13 CONTINUE
GO TO (130,131,132,133,134,135,136,137,138,139,1310,

```

```
11311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321),ISC
130 CONTINUE
C----- SECTION 1 ----- coefficient = GRND
  RETURN
131 CONTINUE
C----- SECTION 2 ----- coefficient = GRND1
  RETURN
132 CONTINUE
C----- SECTION 3 ----- coefficient = GRND2
  RETURN
133 CONTINUE
C----- SECTION 4 ----- coefficient = GRND3
  RETURN
134 CONTINUE
C----- SECTION 5 ----- coefficient = GRND4
  RETURN
135 CONTINUE
C----- SECTION 6 ----- coefficient = GRND5
  RETURN
136 CONTINUE
C----- SECTION 7 ----- coefficient = GRND6
  RETURN
137 CONTINUE
C----- SECTION 8 ----- coefficient = GRND7
  RETURN
138 CONTINUE
C----- SECTION 9 ----- coefficient = GRND8
  RETURN
139 CONTINUE
C----- SECTION 10 ----- coefficient = GRND9
  RETURN
1310 CONTINUE
C----- SECTION 11 ----- coefficient = GRND10
  RETURN
1311 CONTINUE
C----- SECTION 12 ----- value = GRND
C
C  UPSTREAM AND LATERAL VALUES ARE PRESCRIBED USING VALUES
C  PREVIOUSLY STORED IN RING BUFFER.
C
C  NBUF = MAXIMUM DIMENSION OF THE RING BUFFER
C  IT0 = SIZE OF THE RING BUFFER (SAME AS INITIAL TIME PERIOD)
C  IBUF = POINTER TO CURRENT ADDRESS IN RING
C  IOFF = POINTER TO IBUF - IT0
C
  LOVAL = LOF(VAL)
  IOFF = IBUF - IT0
  IF (IOFF.LE.0) IOFF = IOFF + NBUF
  IF(NPATCH(1:3).EQ.'UPS') THEN
    IF (INDVAR.EQ.P1) THEN
      DO 1301 J = IYF, IYL
        DO 1301 I = IXF, IXL
          F(LOVAL+J+NY*(I-1)) = RHO1*VBUF(I,IOFF)
1301 CONTINUE
        ELSE IF (INDVAR.EQ.U1) THEN
          DO 1302 J = IYF, IYL
            DO 1302 I = IXF, IXL
```

```
F(L0VAL+J+NY*(I-1)) = UBUF(I,IOFF)
1302 CONTINUE
  ELSE IF (INDVAR.EQ.V1) THEN
    DO 1303 J = IYF, IYL
      DO 1303 I = IXF, IXL
        F(L0VAL+J+NY*(I-1)) = VBUF(I,IOFF)
1303 CONTINUE
      ELSE IF (INDVAR.EQ.H1) THEN
        DO 1304 J = IYF, IYL
          DO 1304 I = IXF, IXL
            F(L0VAL+J+NY*(I-1)) = HBUF(I,IOFF)
1304 CONTINUE
        END IF
      ELSE
        DO 1305 J = IYF, IYL
          DO 1305 I = IXF, IXL
            F(L0VAL+J+NY*(I-1)) = UXBUF(J+(NY-1)/2,IOFF)
1305 CONTINUE
        END IF
      RETURN
1312 CONTINUE
C----- SECTION 13 ----- value = GRND1
  RETURN
1313 CONTINUE
C----- SECTION 14 ----- value = GRND2
  RETURN
1314 CONTINUE
C----- SECTION 15 ----- value = GRND3
  RETURN
1315 CONTINUE
C----- SECTION 16 ----- value = GRND4
  RETURN
1316 CONTINUE
C----- SECTION 17 ----- value = GRND5
  RETURN
1317 CONTINUE
C----- SECTION 18 ----- value = GRND6
  RETURN
1318 CONTINUE
C----- SECTION 19 ----- value = GRND7
  RETURN
1319 CONTINUE
C----- SECTION 20 ----- value = GRND8
  RETURN
1320 CONTINUE
C----- SECTION 21 ----- value = GRND9
  RETURN
1321 CONTINUE
C----- SECTION 22 ----- value = GRND10
  RETURN
C*****
C
C--- GROUP 19. Special calls to GROUND from EARTH
C
  19 GO TO (191,192,193,194,195,196,197,198),ISC
  191 CONTINUE
C * ----- SECTION 1 ---- Start of time step.
```

```
C
C INCREMENT RING BUFFER INDEX IBUF (POINTS TO CURRENT STEP)
C AT START OF EVERY TIME STEP
C
  IBUF=IBUF+1
  IF (IBUF.GT.NBUF) IBUF = 1
C
  IF (ISTEP.EQ.1) THEN
C
    1. GET PARAMETERS FROM Q1 FILE AT START OF TIME STEP
C
    NDUMP = FREQUENCY PHIDA FILES CREATED
    IT0 = INITIAL TIME PERIOD
    IDUMP1 = TIME STEP BEFORE WHICH NO PHIDA FILES ARE CREATED
    V0 = BULK VELOCITY
C
    IT0 = IG(1)
    NDUMP = IG(2)
    IDUMP1 = IG(3)
    INLINE = LG(1)
    LOCOSAV = LG(2)
    V0 = RG(1)
c
    IF (INLINE) THEN
      NXC = 4
      NYC = 52
      IX0 = NX/2+1
      IY0 = (NY-1)/2+1
      IYU = (NY-1)/4
      IYD = 3*(NY-1)/4
      ELSE
      NXC = 4
      NYC = 48
      IX0 = NX/2+1
      IY0 = (NY-1)/4+1
      IYU = (NY-1)/4
      IYD = 3*(NY-1)/4
      END IF
      NLOCO = 2*NXC + 2*NYC
c
      IZ0 = 1
      IX1 = IX0 - NXC/2
      IX2 = IX0 + NXC/2
      IY1 = IY0 - NYC/2
      IY2 = IY0 + NYC/2
C
C AREF IS CROSS-SECTION AT INLET/EXIT
C
  CALL GTIZYX(APROJN,IZ,ARR,NYD,NXD)
  ATOT = 0.0
  DO 1901 I= 1, NX
    AREF(I) = ARR(NY-1,I)
    ATOT=ATOT+AREF(I)
  1901 CONTINUE
C
C
C THIS SECTION IS WHERE GEOMETRIC FACTORS USED TO CALCULATE LOCAL
```

```

C   COEFFICIENTS AROUND A TEST CYLINDER ARE COMPUTED. THE CENTRE OF THE
C   CYLINDER IS LOCATED AT (IX0,IY0) AND THE PERIMETER BOUNDED BY THE LINES
C   IX1 = CONSTANT, IY1 + CONSTANT, IX2 = CONSTANT, IY2 = CONSTANT
C   IN COMPUTATIONAL SPACE. THE COMPUTATION PROCEEDS FROM THE FRONT LEAD-
C   ING
C   EDGE 0 DEG TO 360 DEG IN A CLOCK-WISE FASHION.
C   THE FOLLOWING QUANTITIES ARE COMPUTED:
C
C   PHI(I) = ANGLE FROM FRONT LEADING EDGE OF CYLINDER (DEG)
C   AWALL(I) = AREA OF CYLINDER WALL (M**2)
C   YWALL(I) = PERPENDICULAR DISTANCE FROM OUTSIDE CELL TO WALL (M)
C
      CALL GETPT(IX0,IY0,IZ0,PT0(1),PT0(2),PT0(3))
c
c   SOUTH
c
      ACYL = 0.
      I = 0
      DO 1902 II = IX0-1, IX1, -1
      I = I + 1
      CALL GTIZYX(DHYPN,IZ,ARR,NYD,NXD)
      YWALL(I) = ARR(IY1-1,II)
      CALL GTIZYX(ASURFN,IZ,ARR,NYD,NXD)
      AWALL(I) = ARR(IY1-1,II)
      ACYL = ACYL + AWALL(I)
      CALL GETPT(II,IY1,IZ,PT1(1),PT1(2),PT1(3))
      CALL GETPT(II+1,IY1,IZ,PT2(1),PT2(2),PT2(3))
      PT(1) = .5*(PT1(1) + PT2(1)) - PT0(1)
      PT(2) = .5*(PT1(2) + PT2(2)) - PT0(2)
      PHI(I) = 180.*(1.+ATAN2(PT(1),PT(2)))/PI
1902 CONTINUE
c
c   WEST
c
      DO 1903 II = IY1, IY2-1, 1
      I = I + 1
      CALL GTIZYX(DHXPE,IZ,ARR,NYD,NXD)
      YWALL(I) = ARR(II,IX1-1)
      CALL GTIZYX(ASURFE,IZ,ARR,NYD,NXD)
      AWALL(I) = ARR(II,IX1-1)
      ACYL = ACYL + AWALL(I)
      CALL GETPT(IX1,II,IZ,PT1(1),PT1(2),PT1(3))
      CALL GETPT(IX1,II+1,IZ,PT2(1),PT2(2),PT2(3))
      PT(1) = .5*(PT1(1) + PT2(1)) - PT0(1)
      PT(2) = .5*(PT1(2) + PT2(2)) - PT0(2)
      PHI(I) = 180.*(1.+ATAN2(PT(1),PT(2)))/PI
1903 CONTINUE
c
c   NORTH
c
      DO 1904 II = IX1, IX2-1, +1
      I = I + 1
      CALL GTIZYX(DHYPN,IZ,ARR,NYD,NXD)
      YWALL(I) = ARR(IY2,II)
      CALL GTIZYX(ASURFS,IZ,ARR,NYD,NXD)
      AWALL(I) = ARR(IY2,II)

```



```

      ACYL = ACYL + AWALL(I)
      CALL GETPT(II+1,IY2,IZ,PT1(1),PT1(2),PT1(3))
      CALL GETPT(II,IY2,IZ,PT2(1),PT2(2),PT2(3))
      PT(1) = .5*(PT1(1) + PT2(1)) - PT0(1)
      PT(2) = .5*(PT1(2) + PT2(2)) - PT0(2)
      PHI(I) = 180.*(1.+ATAN2(PT(1),PT(2)))/PI
1904 CONTINUE
c
c  EAST
c
      DO 1905 II = IY2-1, IY1, -1
      I = I + 1
      CALL GTIZYX(DHXPE,IZ,ARR,NYD,NXD)
      YWALL(I) = ARR(II,IX2)
      CALL GTIZYX(ASURFW,IZ,ARR,NYD,NXD)
      AWALL(I) = ARR(II,IX2)
      ACYL = ACYL + AWALL(I)
      CALL GETPT(IX2,II,IZ,PT1(1),PT1(2),PT1(3))
      CALL GETPT(IX2,II+1,IZ,PT2(1),PT2(2),PT2(3))
      PT(1) = .5*(PT1(1) + PT2(1)) - PT0(1)
      PT(2) = .5*(PT1(2) + PT2(2)) - PT0(2)
      PHI(I) = 180.*(1.+ATAN2(PT(1),PT(2)))/PI
1905 CONTINUE
c
c  SOUTH
c
      DO 1906 II = IX2-1, IX0, -1
      I = I + 1
      CALL GTIZYX(DHYPN,IZ,ARR,NYD,NXD)
      YWALL(I) = ARR(IY1-1,II)
      CALL GTIZYX(ASURFN,IZ,ARR,NYD,NXD)
      AWALL(I) = ARR(IY1-1,II)
      ACYL = ACYL + AWALL(I)
      CALL GETPT(II,IY1,IZ,PT1(1),PT1(2),PT1(3))
      CALL GETPT(II+1,IY1,IZ,PT2(1),PT2(2),PT2(3))
      PT(1) = .5*(PT1(1) + PT2(1)) - PT0(1)
      PT(2) = .5*(PT1(2) + PT2(2)) - PT0(2)
      PHI(I) = 180.*(1.+ATAN2(PT(1),PT(2)))/PI
1906 CONTINUE
C
C  INITIALIZE USER VARIABLES
C
      DO 1907 I = 1, NX
      UT(I) = 0.
      VT(I) = V0
      HT(I) = 0.5
1907 CONTINUE
      DO 1908 J = 1, NY
      UXT(J) = 0.
1908 CONTINUE
      IXMON = NX/2
      UREF = 0.
      VREF = V0
      VB3 = V0
      VB3OFF = V0
      END IF
      RETURN
192 CONTINUE
```

```
C * ----- SECTION 2 ---- Start of sweep.
  RETURN
193 CONTINUE
C * ----- SECTION 3 ---- Start of iz slab.

  RETURN
194 CONTINUE
C * ----- SECTION 4 ---- Start of iteration.
  RETURN
195 CONTINUE
C * ----- SECTION 5 ---- Finish of iteration.
  RETURN
196 CONTINUE
C * ----- SECTION 6 ---- Finish of iz slab.
  RETURN
197 CONTINUE
C * ----- SECTION 7 ---- Finish of sweep.
C
C  VALUES AT CURRENT TIME STEP ARE AS FOLLOWS
C
C  UT(I) = CROSS-WISE VELOCITY AT (NY-1)/2
C  VT(I) = STREAM-WISE VELOCITY AT (NY-1)/2
C  HT(I) = ENTHALPY AT (NY-1)/2
C  UXT  = CROSS-WISE VELOCITY
c  PBAR2 = AVERAGE PRESSURE UPSTREAM  OF TEST CYLINDER
c  PBAR4 = AVERAGE PRESSURE DOWNSTREAM OF TEST CYLINDER
c  VB3   = BULK VELOCITY AT MINIMUM CROSS-SECTION AT (NY-1)/2
C  HB3   = BULK ENTHALPY AT MINIMUM CROSS-SECTION AT (NY-1)/2
C
C  IF (ISWEEP.GE.LSWEEP) THEN
c
c  LOP1 = LOF(P1)
c  LOU1 = LOF(U1)
c  LOV1 = LOF(V1)
c  LOH1 = LOF(H1)
c  LOVPOR = LOF(VPOR)
c  LOAEAST = LOF(AEAST)
c  VB2 = 0.
c  VB3 = 0.
c  HB2 = 0.
c  HB3 = 0.
c  PBAR2 = 0.
c  PBAR4 = 0.
c  NKOUNT = 0
C
C  DO 1909 I = 1, NX
C    UT(I) = F(LOU1 + (NY-1)/2 + 1 + NY*(I-1))
C    VT(I) = F(LOV1 + (NY-1)/2 + NY*(I-1))
C    HT(I) = F(LOH1 + (NY-1)/2 + NY*(I-1))
C    VB3 = VB3 + VT(I)*AREF(I)
C    VB2 = F(LOV1 + IYU + NY*(I-1))*F(AEAST + IYU + NY*(I-1))
C    HB3 = HB3 + VT(I)*HT(I)*AREF(I)
C    HB2 = HB2 + F(LOH1 + IYU + NY*(I-1))*
C    $F(LOV1 + IYU + NY*(I-1))*F(LOAEAST + IYU + NY*(I-1))
C    IF (F(LOVPOR + IYU + NY*(I-1)).GT.0.5) THEN
C      PBAR2 = PBAR2 + F(LOP1 + IYU + NY*(I-1))
C      PBAR4 = PBAR4 + F(LOP1 + IYD + NY*(I-1))
```

```
NKOUNT = NKOUNT + 1
END IF
1909 CONTINUE
C
DO 1910 J = 1, NY
  IF (INLINE) THEN
    UXT(J) = F(L0U1 + J + NY*(NX/2-1))
  ELSE
    UXT(J) = F(L0U1 + J + NY*(NX-1))
  END IF
1910 CONTINUE
C
PBAR2 = PBAR2/FLOAT(NKOUNT)
PBAR4 = PBAR4/FLOAT(NKOUNT)
HB3 = HB3/VB3
HB2 = HB2/VB2
VB3 = VB3/ATOT
CMASS = V0/VB3
EU = 2.*(PBAR2 - PBAR4)/(RHO1*V0*V0 + 1.E-10)
IF (.NOT.INLINE) EU = EU/2.
C
C THE FOLLOWING SECTION COMPUTES REFERENCE VALUES OF VELOCITY AND PRES-
SURE
C WHICH ARE OUTPUT TO FILE "REFPLOT".
C
IF (.NOT.INLINE) THEN
  PRF2 = F(L0P1 + (NY-1)/4 + NY*(NX-1))
  PRF4 = F(L0P1 + 3*(NY-1)/4 + NY*(NX-1))
  URF1 = F(L0U1 + 1 + NY*(NX/2-1))
  URF2 = F(L0U1 + (NY-1)/4 + NY*(NX-1))
  URF3 = F(L0U1 + (NY-1)/2 + 1 + NY*(NX/2-1))
  URF4 = F(L0U1 + 3*(NY-1)/4 + NY*(NX-1))
  URF5 = F(L0U1 + (NY-1) + NY*(NX/2-1))
  VRF1 = F(L0V1 + 1 + NY*(NX/2-1))
  VRF2 = F(L0V1 + (NY-1)/4 + NY*(NX-1))
  VRF3 = F(L0V1 + (NY-1)/2 + NY*(NX/2-1))
  VRF4 = F(L0V1 + 3*(NY-1)/4 + NY*(NX-1))
  VRF5 = F(L0V1 + (NY-1) + NY*(NX/2-1))
ELSE
  PRF2 = F(L0P1 + (NY-1)/4 + NY*(NX/2-1))
  PRF4 = F(L0P1 + 3*(NY-1)/4 + NY*(NX/2-1))
  URF1 = F(L0U1 + 1 + NY*(NX-1))
  URF2 = F(L0U1 + (NY-1)/4 + NY*(NX/2-1))
  URF3 = F(L0U1 + (NY-1)/2 + 1 + NY*(NX-1))
  URF4 = F(L0U1 + 3*(NY-1)/4 + NY*(NX/2-1))
  URF5 = F(L0U1 + (NY-1) + NY*(NX-1))
  VRF1 = F(L0V1 + 1 + NY*(NX-1))
  VRF2 = F(L0V1 + (NY-1)/4 + NY*(NX/2-1))
  VRF3 = F(L0V1 + (NY-1)/2 + NY*(NX-1))
  VRF4 = F(L0V1 + 3*(NY-1)/4 + NY*(NX/2-1))
  VRF5 = F(L0V1 + (NY-1) + NY*(NX-1))
END IF
C
C CP(I) = PRESSURE COEFFICIENT
C CF(I) = FRICTION COEFFICIENT
C NF(I) = LOCAL NUSSELT NUMBER REFERRED TO T2
C
```

C CDP = PRESSURE COMPONENT OF DRAG COEFFICIENT
C CDF = FRICTION COMPONENT OF DRAG COEFFICIENT
C CLP = PRESSURE COMPONENT OF LIFT COEFFICIENT
C CLF = FRICTION COMPONENT OF LIFT COEFFICIENT
C CPRF = PRESSURE AT REFERENCE CELL

C

CDP = 0.
CDF = 0.
CLP = 0.
CLF = 0.
RNUF = 0.
IF (STORE(P1)) CALL GETONE(P1,XX,IY1-1,IX0)
IF (STORE(P1)) CALL GETONE(P1,YY,IY1-1,IX0-1)
CPRF = 1. - (XX+YY)/(RHO1*V0*V0)
IF (STEADY) CPRFAV=CPRF

C

C SOUTH

C

I = 0
DO 1911 II = IX0-1, IX1, -1
I = I + 1
IF (STORE(P1)) CALL GETONE(P1,PWALL(I),IY1-1,II)
IF (STORE(V1)) CALL GETONE(V1,VWALL(I),IY1-1,II-1)
IF (STORE(H1)) CALL GETONE(H1,H1WALL(I),IY1-1,II)
COSPFI = COS(PI*PHI(I)/180.)
SINPHI = SIN(PI*PHI(I)/180.)
TWALL = QWALL*YWALL(I)/KFLUID + H1WALL(I)/CFLUID
CP(I) = CPRFAV + 2.*PWALL(I)/(RHO1*V0*V0)
CF(I) = -1.*RHO1*ENUL*VWALL(I)/(YWALL(I))
CDP = CDP + PWALL(I)*AWALL(I)*COSPFI
CDF = CDF + CF(I)*AWALL(I)*SINPHI
CLP = CLP - PWALL(I)*AWALL(I)*SINPHI
CLF = CLF + CF(I)*AWALL(I)*COSPFI

C

CDRP(I) = +1.*PWALL(I)*AWALL(I)*COSPFI
CDRF(I) = +1.*CF(I)*AWALL(I)*SINPHI
CLFP(I) = -1.*PWALL(I)*AWALL(I)*SINPHI
CLFF(I) = +1.*CF(I)*AWALL(I)*COSPFI

C

CF(I) = 2.*CF(I)/(RHO1*V0*V0)
TERM = DIAM*(TWALL - H1WALL(I)/CFLUID)/YWALL(I)
NF(I) = TERM/(TWALL - TRFAV)
RNUF = RNUF + AWALL(I)*NF(I)

1911 CONTINUE

C

C WEST

C

DO 1912 II = IY1, IY2-1, 1
I = I + 1
IF (STORE(P1)) CALL GETONE(P1,PWALL(I),II,IX1-1)
IF (STORE(V1)) CALL GETONE(V1,VWALL(I),II,IX1-1)
IF (STORE(H1)) CALL GETONE(H1,H1WALL(I),II,IX1-1)
COSPFI = COS(PI*PHI(I)/180.)
SINPHI = SIN(PI*PHI(I)/180.)
TWALL = QWALL*YWALL(I)/KFLUID + H1WALL(I)/CFLUID
CP(I) = CPRFAV + 2.*PWALL(I)/(RHO1*V0*V0)
CF(I) = RHO1*ENUL*VWALL(I)/(YWALL(I))

```
CDP = CDP + PWALL(I)*AWALL(I)*COSPFI
CDF = CDF + CF(I)*AWALL(I)*SINPHI
CLP = CLP - PWALL(I)*AWALL(I)*SINPHI
CLF = CLF + CF(I)*AWALL(I)*COSPFI
c
CDRP(I) = +1.*PWALL(I)*AWALL(I)*COSPFI
CDRF(I) = +1.*CF(I)*AWALL(I)*SINPHI
CLFP(I) = -1.*PWALL(I)*AWALL(I)*SINPHI
CLFF(I) = +1.*CF(I)*AWALL(I)*COSPFI
c
CF(I) = 2.*CF(I)/(RHO1*V0*V0)
TERM = DIAM*(TWALL - H1WALL(I)/CFLUID)/YWALL(I)
NF(I) = TERM/(TWALL - TRFAV)
RNUF = RNUF + AWALL(I)*NF(I)
1912 CONTINUE
c
c  NORTH

DO 1913 II = IX1, IX0-1, +1
  I = I + 1
  IF (STORE(P1)) CALL GETONE(P1,PWALL(I),IY2,II)
  IF (STORE(V1)) CALL GETONE(U1,VWALL(I),IY2,II)
  IF (STORE(H1)) CALL GETONE(H1,H1WALL(I),IY2,II)
  COSPHI = COS(PI*PHI(I)/180.)
  SINPHI = SIN(PI*PHI(I)/180.)
  TWALL = QWALL*YWALL(I)/KFLUID + H1WALL(I)/CFLUID
  CP(I) = CPRFAV + 2.*PWALL(I)/(RHO1*V0*V0)
  CF(I) = RHO1*ENUL*VWALL(I)/(YWALL(I))
  CDP = CDP + PWALL(I)*AWALL(I)*COSPFI
  CDF = CDF + CF(I)*AWALL(I)*SINPHI
  CLP = CLP - PWALL(I)*AWALL(I)*SINPHI
  CLF = CLF + CF(I)*AWALL(I)*COSPFI
c
CDRP(I) = +1.*PWALL(I)*AWALL(I)*COSPFI
CDRF(I) = +1.*CF(I)*AWALL(I)*SINPHI
CLFP(I) = -1.*PWALL(I)*AWALL(I)*SINPHI
CLFF(I) = +1.*CF(I)*AWALL(I)*COSPFI
c
CF(I) = 2.*CF(I)/(RHO1*V0*V0)
TERM = DIAM*(TWALL - H1WALL(I)/CFLUID)/YWALL(I)
NF(I) = TERM/(TWALL - TRFAV)
RNUF = RNUF + AWALL(I)*NF(I)
1913 CONTINUE

c  NORTH
c
DO 1914 II = IX0, IX2-1, +1
  I = I + 1
  IF (STORE(P1)) CALL GETONE(P1,PWALL(I),IY2,II)
  IF (STORE(V1)) CALL GETONE(U1,VWALL(I),IY2,II)
  IF (STORE(H1)) CALL GETONE(H1,H1WALL(I),IY2,II)
  COSPHI = COS(PI*PHI(I)/180.)
  SINPHI = SIN(PI*PHI(I)/180.)
  TWALL = QWALL*YWALL(I)/KFLUID + H1WALL(I)/CFLUID
  CP(I) = CPRFAV + 2.*PWALL(I)/(RHO1*V0*V0)
  CF(I) = -1.*RHO1*ENUL*VWALL(I)/(YWALL(I))
  CDP = CDP + PWALL(I)*AWALL(I)*COSPFI
```

```
CDF = CDF - CF(I)*AWALL(I)*SINPHI
CLP = CLP - PWALL(I)*AWALL(I)*SINPHI
CLF = CLF - CF(I)*AWALL(I)*COSPHI
c
CDRP(I) = +1.*PWALL(I)*AWALL(I)*COSPHI
CDRF(I) = -1.*CF(I)*AWALL(I)*SINPHI
CLFP(I) = -1.*PWALL(I)*AWALL(I)*SINPHI
CLFF(I) = -1.*CF(I)*AWALL(I)*COSPHI
c
CF(I) = 2.*CF(I)/(RHO1*V0*V0)
TERM = DIAM*(TWALL - H1WALL(I)/CFLUID)/YWALL(I)
NF(I) = TERM/(TWALL - TRFAV)
RNUF = RNUF + AWALL(I)*NF(I)
1914 CONTINUE
c
c  EAST
c
DO 1915 II = IY2-1, IY1, -1
I = I + 1
IF (STORE(P1)) CALL GETONE(P1,PWALL(I),II,IX2)
IF (STORE(V1)) CALL GETONE(V1,VWALL(I),II,IX2)
IF (STORE(H1)) CALL GETONE(H1,H1WALL(I),II,IX2)
COSPHI = COS(PI*PHI(I)/180.)
SINPHI = SIN(PI*PHI(I)/180.)
TWALL = QWALL*YWALL(I)/KFLUID + H1WALL(I)/CFLUID
CP(I) = CPRFAV + 2.*PWALL(I)/(RHO1*V0*V0)
CF(I) = RHO1*ENUL*VWALL(I)/(YWALL(I))
CDP = CDP + PWALL(I)*AWALL(I)*COSPHI
CDF = CDF - CF(I)*AWALL(I)*SINPHI
CLP = CLP - PWALL(I)*AWALL(I)*SINPHI
CLF = CLF - CF(I)*AWALL(I)*COSPHI
c
CDRP(I) = +1.*PWALL(I)*AWALL(I)*COSPHI
CDRF(I) = -1.*CF(I)*AWALL(I)*SINPHI
CLFP(I) = -1.*PWALL(I)*AWALL(I)*SINPHI
CLFF(I) = -1.*CF(I)*AWALL(I)*COSPHI
c
CF(I) = 2.*CF(I)/(RHO1*V0*V0)
TERM = DIAM*(TWALL - H1WALL(I)/CFLUID)/YWALL(I)
NF(I) = TERM/(TWALL - TRFAV)
RNUF = RNUF + AWALL(I)*NF(I)
1915 CONTINUE
c
c  SOUTH
c
DO 1916 II = IX2-1, IX0, -1
I = I + 1
IF (STORE(P1)) CALL GETONE(P1,PWALL(I),IY1-1,II)
IF (STORE(V1)) CALL GETONE(U1,VWALL(I),IY1-1,II-1)
IF (STORE(H1)) CALL GETONE(H1,H1WALL(I),IY1-1,II)
COSPHI = COS(PI*PHI(I)/180.)
SINPHI = SIN(PI*PHI(I)/180.)
TWALL = QWALL*YWALL(I)/KFLUID + H1WALL(I)/CFLUID
CP(I) = CPRFAV + 2.*PWALL(I)/(RHO1*V0*V0)
CF(I) = +1.*RHO1*ENUL*VWALL(I)/(YWALL(I))
CDP = CDP + PWALL(I)*AWALL(I)*COSPHI
CDF = CDF - CF(I)*AWALL(I)*SINPHI
```

```
CLP = CLP - PWALL(I)*AWALL(I)*SINPHI
CLF = CLF - CF(I)*AWALL(I)*COSPHI
C
CDRP(I) = +1.*PWALL(I)*AWALL(I)*COSPHI
CDRF(I) = -1.*CF(I)*AWALL(I)*SINPHI
CLFP(I) = -1.*PWALL(I)*AWALL(I)*SINPHI
CLFF(I) = -1.*CF(I)*AWALL(I)*COSPHI
C
CF(I) = 2.*CF(I)/(RHO1*V0*V0)
TERM = DIAM*(TWALL - H1WALL(I)/CFLUID)/YWALL(I)
NF(I) = TERM/(TWALL - TRFAV)
RNUF = RNUF + AWALL(I)*NF(I)
1916 CONTINUE
C
RNUF = RNUF/ACYL
CDP = 2.*CDP/(DIAM*RHO1*V0*V0)
CDF = 2.*CDF/(DIAM*RHO1*V0*V0)
CLP = 2.*CLP/(DIAM*RHO1*V0*V0)
CLF = 2.*CLF/(DIAM*RHO1*V0*V0)
CDRG = CDP + CDF
CLFT = CDL + CLF
C
END IF
C
RETURN
198 CONTINUE
C * ----- SECTION 8 ---- Finish of time step.
C
C CALCULATE CUMMULATIVE AVERAGES
C
C WHEN COMPUTING CPREF HBAV ETC. WHEN DOING AUTOPSY RUNS
C THE USER SHOULD CHANGE THESE VALUES TO THE APPROPRIATE VALUES
C
IF (ISTEP.LE.IT0) THEN
CPRFSUM = 0.
CDRGSUM = 0.
CLFTSUM = 0.
EUSUM = 0.
HB3SUM = 0.
HB2SUM = 0.
RNUFSUM = 0.
ELSE
CPRFSUM = CPRFSUM + CPRF
CDRGSUM = CDRGSUM + CDRG
CLFTSUM = CLFTSUM + CLFT
EUSUM = EUSUM + EU
HB3SUM = HB3SUM + HB3
HB2SUM = HB2SUM + HB2
RNUFSUM = RNUFSUM + RNUF
C
RT = FLOAT (ISTEP - IT0)
CPRFAV = CPRFSUM/RT
CDRGAV = CDRGSUM/RT
CLFTAV = CLFTSUM/RT
EUAV = EUSUM/RT
HB3AV = HB3SUM/RT
HB2AV = HB2SUM/RT
```

```
CPRFAV = CPRFSUM/RT
RNUFAV = RNUFSUM/RT
END IF
C
C FILE LOCONNN CONTAINS LOCAL COEFFICIENTS AS A FUNCTION OF PHI
C AT EACH TIME STEP IF LOCOSAV IS SET .TRUE.
C
LOCOSAV = .FALSE.
IF (LOCOSAV) THEN
  IF (STEADY) THEN
    LOCOFIL = 'LOCO'
  ELSE
    LOCOFIL = 'LOCO'//CSTEP
  END IF
  OPEN(91,FILE = LOCOFIL,ACCESS='SEQUENTIAL',STATUS='UNKNOWN'
$,FORM='FORMATTED',IOSTAT=IOS)
  IF (IOS.NE.0) THEN
    WRITE (14,*) 'ERROR OPENING FILE ', LOCOFIL
    WRITE (6,*) ' ERROR OPENING FILE ', LOCOFIL
  END IF
  WRITE (91,*) ' PHI NUF CF CP CDRP CDRF CLFP CLFF'
  DO 1917 I = 1, NLOCO
    WRITE (91,*) PHI(I), NF(I), CF(I), CP(I),
    $CDRP(I), CDRF(I), CLFP(I), CLFF(I)
1917 CONTINUE
    CLOSE(91,STATUS='KEEP')
  END IF
C
C STORE VALUES IN RING BUFFERS
C
C VBUF(I,IBUF) = STREAM-WISE VELOCITY
C UBUF(I,IBUF) = CROSS-WISE VELOCITY
C UXBUF(J,IBUF) = CROSS-WISE VELOCITY AT LATERAL BOUNDARY
C HBUF(I,IBUF) = SCALED ENTHALPY VALUES
C
C1 = 1.
C2 = -1.*HB3AV
DO 1918 I = 1, NX
  UBUF(I,IBUF) = UT(I)
  VBUF(I,IBUF) = VT(I)
  HBUF(I,IBUF) = C1*HT(I) + C2
1918 CONTINUE
  DO 1919 J = 1, NY
    UXBUF(J,IBUF) = UXT(J)
1919 CONTINUE
C
C COMPUTE PERIOD AND PHASE.
C
C URF2 = UPSTREAM REFERENCE VELOCITY USED TO COMPUTE PHASE
C UDR4 = DOWNSTREAM REFERENCE VELOCITY USED TO COMPUTE PHASE
C
C
C TAU2 = UPSTREAM PERIOD (SEC)
C TAU4 = UPSTREAM PERIOD (SEC)
C ITAU2 = INTEGER PERIOD (TIME STEPS)
C ITAU4 = INTEGER PERIOD (TIME STEPS)
C T2 = TIME AT WHICH URF2 = 0 (SEC)
```



```
C  T4    = TIME AT WHICH URF4 = 0 (SEC)
C  T2_LAST = LAST VALUE OF TC
C  T4_LAST = LAST VALUE OF T4
C  IPHASE  = PHASE TO NEAREST INTEGER (TIME STEPS)
C  RPHASE  = PHASE (SEC) BETWEEN 2 AND 4
C
  IF ((URF2_LAST.GT.0).AND.(URF2.LE.0)) THEN
    T2_LAST = T2
    RLAMDA = URF2_LAST/(URF2_LAST-URF2)
    T2 = FLOAT(ISTEP) - 1. + RLAMDA
    TAU2 = (T2 - T2_LAST)*DT
  END IF
  IF ((URF4_LAST.GT.0).AND.(URF4.LE.0)) THEN
    T4_LAST = T4
    RLAMDA = URF4_LAST/(URF4_LAST-URF4)
    T4 = FLOAT(ISTEP) - 1. + RLAMDA
    TAU4 = (T4 - T4_LAST)*DT
    IF (T4.GT.T2) THEN
      IPHASE = NINT(T4 - T2)
      RPHASE = (T4 - T2)*DT
    END IF
  END IF
C
C  CHECK THAT RING BUFFER IS LARGE ENOUGH
C
  IF (ITAU2.GT.NBUF) THEN
    WRITE (14,*) ' WARNING   !!! - RING BUFFER TOO SMALL'
  END IF

C
C  COMPUTE BULK VELOCITY AT NEXT TIME STEP MINUS REFERENCE TIME (IT0)
C
  IF (ITAU.GT.1) THEN
    UB = 0.
    VB3 = 0.
    IOFF = IBUF + 1 - IT0
    IF (IOFF.LE.0) IOFF = IOFF + NBUF
    DO 1920 I = 1, NX
      VB3OFF = VB3OFF + VBUF(I,IOFF)*AREF(I)
1920 CONTINUE
    VB3OFF = VB3OFF/ATOT
  END IF
C
C  COMPUTE STROUHAL NUMBER
C
  SH2 = DIAM/(V0*TAU2+1.E-30)
  SH4 = DIAM/(V0*TAU4+1.E-30)
  IF (ABS(SH2).GT.1000) SH2 = 0
  IF (ABS(SH4).GT.1000) SH4 = 0
C
C  OUTPUT DATA TO SCREEN
C
976 FORMAT (A,I4)
971 FORMAT (A,G11.4)
975 FORMAT (2(A,G11.4))
961 FORMAT (4(A,G11.4))
977 FORMAT ( ' ' )
```

```
972 FORMAT(' VB3 = ',G11.4,' VB3 = ',G11.4,' VB3OFF = ',G11.4)
979 FORMAT(A,G11.4,A,G11.4)
  WRITE(6,977)
  WRITE(14,977)
  WRITE(6,976) 'ISTEP = ',ISTEP
  WRITE(14,976) 'ISTEP = ',ISTEP
  WRITE(6,961) ' EU = ',EU,' CMASS = ',CMASS,
$' HB3 = ',HB3,' HB3AV = ',HB3AV
  WRITE(14,961) ' EU = ',EU,' CMASS = ',CMASS,
$' HB3 = ',HB3,' HB3AV = ',HB3AV
  WRITE(6,961) ' PRF2 = ',PRF2,' PRF4 = ',PRF4,
$' VB3 = ',VB3,' VB3OFF = ',VB3OFF
  WRITE(14,961) ' PRF2 = ',PRF2,' PRF4 = ',PRF4,
$' VB3 = ',VB3,' VB3OFF = ',VB3OFF
  WRITE(6,961) ' PBAR2= ',PBAR2,' PBAR4= ',PBAR4,
$' VB3 = ',VB3,' VB3OFF = ',VB3OFF
  WRITE(14,961) ' PBAR2= ',PBAR2,' PBAR4= ',PBAR4,
$' VB3 = ',VB3,' VB3OFF = ',VB3OFF
  WRITE(6,975) ' SH2 = ',SH2,'SH4 = ',SH4
  WRITE(14,975) ' SH2 = ',SH2,'SH4 = ',SH4
  WRITE(6,971) ' NUF = ',RNUF
  WRITE(14,971) ' NUF = ',RNUF
  WRITE(6,977)
  WRITE(14,977)
C
  URF_LAST = URF2
  URF2_LAST = URF2
  URF4_LAST = URF4
C
  IF (STEADY) RETURN
c
C  OUTPUT MONITOR DATA TO FILE "REFPLOT"
C
C
  OPEN(90,FILE='REFPLOT',ACCESS='SEQUENTIAL',STATUS='OLD'
1,FORM='FORMATTED',IOSTAT=IOS)
  IF (IOS.NE.0) THEN
    WRITE(14,*) ' ERROR OPENING FILE REFPLOT'
    WRITE(6,*) ' ERROR OPENING FILE REFPLOT'
    STOP
  ELSE
    WRITE(14,*) ' FILE REFPLOT OPENED'
    WRITE(6,*) ' FILE REFPLOT OPENED'
  END IF
  RT = (FLOAT(ISTEP))*DT
  WRITE(90,*) ISTEP, RT, URF1, URF2, URF3, URF4, URF5,
$VRF1, VRF2, VRF3, VRF4, VRF5, PRF2, PRF4, HB2, HB3
  CLOSE(90,STATUS='KEEP')
C
C  OUTPUT DATA TO FILE "TIMPLOT"
C
  OPEN(91,FILE='TIMPLOT',ACCESS='SEQUENTIAL',STATUS='OLD'
1,FORM='FORMATTED',IOSTAT=IOS)
  IF (IOS.NE.0) THEN
    WRITE(14,*) ' ERROR OPENING FILE TIMPLOT'
    WRITE(6,*) ' ERROR OPENING FILE TIMPLOT'
    STOP
```

```
ELSE
WRITE (14,*) ' FILE TIMPLOT OPENED'
WRITE (6,*) ' FILE TIMPLOT OPENED'
END IF
RT = (FLOAT(ISTEP))*DT
WRITE(91,*) ISTEP, RT, SH2, SH4, RPHASE, EU, CDRG, CLFT, RNUF
CLOSE(91,STATUS='KEEP')
C
C  OUTPUT CUMMULATIVE AVERAGES TO FILE "CUMPLOT"
C
OPEN(92,FILE= 'CUMPLOT',ACCESS='SEQUENTIAL',STATUS='OLD'
1,FORM='FORMATTED',IOSTAT=IOS)
IF (IOS.NE.0) THEN
WRITE (14,*) ' ERROR OPENING FILE CUMPLOT'
WRITE (6,*) ' ERROR OPENING FILE CUMPLOT'
STOP
ELSE
WRITE (14,*) ' FILE CUMPLOT OPENED'
WRITE (6,*) ' FILE CUMPLOT OPENED'
END IF
RT = (FLOAT(ISTEP))*DT
WRITE(92,*) ISTEP, RT, SH2AV, SH4AV, EUAV, CPRFAV, CDRGAV, CLFTAV,
$HB2AV, HB3AV, RNUFAV
CLOSE(92,STATUS='KEEP')
C
C  OUTPUT DATA TO PHIDANNN
C
IF (ISTEP.LT.IDUMP1) RETURN
IF (MOD((ISTEP),NDUMP).NE.0) RETURN
IDIG = 0
CSTEP = '000'
NDIGIT = 3
IFIL = ISTEP/NDUMP
DO 1921 IDEC = 1, NDIGIT
IREM = MOD(IFIL,10**IDEC) - MOD(IFIL,10**(IDEC-1))
IREM = IREM/(10**(IDEC-1))
INDX = NDIGIT - IDEC + 1
IF (IREM.EQ.0) CSTEP(INDX:INDX) = '0'
IF (IREM.EQ.1) CSTEP(INDX:INDX) = '1'
IF (IREM.EQ.2) CSTEP(INDX:INDX) = '2'
IF (IREM.EQ.3) CSTEP(INDX:INDX) = '3'
IF (IREM.EQ.4) CSTEP(INDX:INDX) = '4'
IF (IREM.EQ.5) CSTEP(INDX:INDX) = '5'
IF (IREM.EQ.6) CSTEP(INDX:INDX) = '6'
IF (IREM.EQ.7) CSTEP(INDX:INDX) = '7'
IF (IREM.EQ.8) CSTEP(INDX:INDX) = '8'
IF (IREM.EQ.9) CSTEP(INDX:INDX) = '9'
1921 CONTINUE
NMFIL(22) = 'PHI'//CSTEP
NMFIL(23) = 'PHIDA'//CSTEP
CALL DUMP
RETURN
```

A3.6 Fin and tube heat exchanger

Q1 FILE

TALK=F;RUN(1, 1);VDU=X11-TERM

GROUP 1. Run title and other preliminaries

TEXT(FIN AND TUBE HEAT EXCHANGER)

** S = PITCH
** S1 = CROSS-WISE PITCH
** S2 = STREAM-WISE PITCH
** RP = LONGITUDINAL PITCH
** DIAM = CYLINDER DIAMETER
** W0 = BULK VELOCITY
** RE = PRESCRIBED REYNOLDS' NUMBER
** AMYCO = PRESSURE COEFFICIENT
** AMYRELAX = PRESSURE RELAXATION FACTOR (IN GROUND)
** RESCALE = SCALE FACTOR USED TO COMPUTE RE
** TWALL = WALL TEMPERATURE (CONSTANT T)
** QWALL = WALL HEAT FLUX (CONSTANT QW)
** KFLUID = FLUID THERMAL CONDUCTIVITY
** CFLUID = FLUID SPECIFIC HEAT
** ALPHA = FLUID THERMAL DIFFUSIVITY
** PRAN = FLUID PRANDTL NUMBER
** FALST = FALSE TIME STEP RELAXATION FACTOR
** GEOFAC = GEOMETRIC FACTOR USED IN GRID GENERATION

REAL(S,S1, S2, DIAM, W0, RE)

REAL(Y1,Z1,Z2)

REAL(TBIN, TWALL, QWALL, DELH, KFLUID, CFLUID, ALPHA, PRAN)

REAL(FALST,AMYCO,AMYRELAX,RESCALE)

REAL(GEOFAC, RP)

INTEGER(NZP1,NZM1,IZ1,IZ2)

INTEGER(IYF1,IYF2,IYL1,IYL2)

BOOLEAN(LHGRID,LIGRID,LPOR,LINLINE)

BOOLEAN(ICONFIG,DCONFIG,LCONSTT)

LHGRID = T;LIGRID = F

ICONFIG=F;DCONFIG=T

LINLINE = F;LCONSTT=T

LPOR = F

DIAM = 1.

RP = 2.

RP = 2.

RP = .5

RESCALE = 1.

IF (LINLINE) THEN

S=1.5;S1 = S;S2 = S

GEOFAC = 1.

ELSE

S=2.0;S1 = S*SQRT(2);S2 = S/SQRT(2)

GEOFAC = 1.5

ENDIF

```

TBIN = 0.
NX=20;NY = 40;NZ = 30
NZ = NZ + 1
RE = 100.
W0 = 1.
AMYCO=1.E+06;AMYRELAX = .5
  GROUP 2. Transience time-step specification
STEADY=T
  GROUP 3. X-direction grid specification
  GROUP 4. Y-direction grid specification
  GROUP 5. Z-direction grid specification
  GROUP 6. Body-fitted coordinates or grid distortion
BFC=T;NONORT=T
READCO(GRID)
  GROUP 7. Variables stored, solved & named
FALST = 1.E+01*ZWLAST/(NZ*W0)
STORE(P1,U1,V1,W1,H1)
SOLUTN(P1,Y,Y,Y,N,N,N)
SOLUTN(U1,Y,Y,N,Y,N,N)
SOLUTN(V1,Y,Y,N,Y,N,N)
SOLUTN(W1,Y,Y,N,Y,N,N)
SOLUTN(H1,Y,Y,N,N,N,N)
  GROUP 8. Terms (in differential equations) & devices
TERMS(H1,N,P,P,P,P,P)
  GROUP 9. Properties of the medium (or media)
PRAN = 1.
ENUL = RESCALE*W0/RE;RHO1=1.0;
KFLUID=RESCALE*W0/(RE*PRAN);CFLUID=1.
ALPHA=KFLUID/(RHO1*CFLUID)
PRNDTL(H1) = ENUL/ALPHA
TWALL = 1.;QWALL = 1.
  GROUP 10. Inter-phase-transfer processes and properties
  GROUP 11. Initialization of variable or porosity fields
FIINIT(V1)=0.;FIINIT(P1)=0.;
IF (LCONSTT) THEN
FIINIT(H1)=CFLUID*(TBIN+TWALL)/2.
ELSE
DELH = 3.14*QWALL*DIAM/(RHO1*W0*(S1-DIAM))
PATCH(HINIT,LINVLZ,1,NX,1,NY,1,NZ,1,LSTEP)
INIT(HINIT,H1,DELH/S2,CFLUID*TBIN)
ENDIF
IF (LIGRID.OR.ICONFIG) THEN
FIINIT(W1)=S1/(S1-DIAM)*W0
ELSE
FIINIT(W1)=W0
ENDIF

IF (LHGRID) THEN

IF (ICONFIG) THEN
Z1 = GEOFAC*(NZ-1)*3.*DIAM/(2.*(2.*S2 + DIAM)) - 1
IZ1 = Z1
Z1 = (NZ-1)/2+1-IZ1
ELSE
Z1 = GEOFAC*(NZ-1)*3.*DIAM/(2.*(2.*S2 + DIAM)) - 1
ENDIF
IZ1 = Z1

```

```
Z1 = IZ1  
Z2 = NZ-Z1  
IZZ = Z2
```

```
ELSE IF (LIGRID)
```

```
IF (DCONFIG) THEN
```

```
Z1 = NZ  
Z2 = NZ-Z1  
CONPOR(0,CELL,1,NX,1,NY/2,NZ+1,NZ+1)  
ELSE  
Y1 = 1.5*NY*(S2-DIAM)/(S2-DIAM+S1)  
CONPOR(0,CELL,1,NX,1,Y1,NZ,NZ)  
ENDIF  
LPOR = T  
ENDIF
```

GROUP 12. Convection and diffusion adjustments
GROUP 13. Boundary conditions and special source

**** WALL BOUNDARY CONDITIONS**

```
IF (LHGRID) THEN
```

```
IF ((.NOT.(ICONFIG)).AND.(.NOT.DCONFIG)) THEN  
PATCH(S1,SWALL,1,NX,1,1,Z1,1,LSTEP)  
PATCH(S2,SWALL,1,NX,1,1,Z2,NZ+1,1,LSTEP)  
PATCH(SH1,SOUTH,1,NX,1,1,Z1,1,LSTEP)  
PATCH(SH2,SOUTH,1,NX,1,1,Z2,NZ+1,1,LSTEP)  
ENDIF
```

```
IF (ICONFIG) THEN  
PATCH(S1,SWALL,1,NX,1,1,Z1,NZ/2,1,LSTEP)  
PATCH(S2,SWALL,1,NX,1,1,NZ/2+1,Z2,1,LSTEP)  
PATCH(SH1,SOUTH,1,NX,1,1,Z1,NZ/2,1,LSTEP)  
PATCH(SH2,SOUTH,1,NX,1,1,NZ/2+1,Z2,1,LSTEP)  
ENDIF
```

```
IF (DCONFIG) THEN  
PATCH(S1,SWALL,1,NX,1,1,Z1,1,LSTEP)  
PATCH(S2,NWALL,1,NX,NY,NY,Z2,NZ+1,1,LSTEP)  
PATCH(SH1,SOUTH,1,NX,1,1,Z1,1,LSTEP)  
PATCH(SH2,NORTH,1,NX,NY,NY,Z2,NZ+1,1,LSTEP)  
ENDIF
```

```
COVAL(S1,W1,1,0.0)  
COVAL(S2,W1,1,0.0)
```

```
IF (LCONSTT) THEN  
COVAL(S1,H1,1/PRNDTL(H1),CFLUID*TWALL)  
COVAL(S2,H1,1/PRNDTL(H1),CFLUID*TWALL)  
ELSE  
COVAL(SH1,H1,FIXFLU,QWALL)  
COVAL(SH2,H1,FIXFLU,QWALL)  
ENDIF
```

ELSE

(IGRID)

PATCH(S1,SWALL,1,NX,1,1,1,NZ-1,1,LSTEP)
COVAL(S1,W1,1.,0.)

IF (LCONSTT) THEN
COVAL(S1,H1,1/PRNDTL(H1),CFLUID*TWALL)
ELSE
PATCH(SH1,SOUTH,1,NX,1,1,1,NZ-1,1,LSTEP)
COVAL(HS1,H1,FIXFLU,QWALL)
ENDIF

IF (DCONFIG) THEN
PATCH(S2,NWALL,1,NX,NY,NY,1,NZ,1,LSTEP)
COVAL(S2,W1,1.,0.0)

IF (LCONSTT) THEN
COVAL(S2,H1,1/PRNDTL(H1),CFLUID*TWALL)
ELSE
PATCH(SH2,NORTH,1,NX,NY,NY,1,NZ,1,LSTEP)
COVAL(S2,H1,FIXFLU,QWALL)
ENDIF

ENDIF
ENDIF

IF (NX.GT.1) THEN
COVAL(S1,U1,1.,0.)
COVAL(S2,U1,1.,0.)
PATCH(WPLTE,WWALL,1,1,1,NY,1,NZ,1,LSTEP)
** COMMENT THE NEXT LINE FOR ADIABATIC PLATE
COVAL(WPLTE,H1,1/PRNDTL(H1),CFLUID*TWALL)
COVAL(WPLTE,V1,1.,0.)
COVAL(WPLTE,W1,1.,0.)
ENDIF

** INLET/EXIT BOUNDARY CONDITIONS

IYF1 = Y1+1
IYF2 = NY
IYL1 = Y1+1
IYL2 = NY

IF (LIGRID.AND.DCONFIG) THEN
IYF1 = 1
IYF2 = NY/2
IYL1 = NY/2+1
IYL2 = NY
ENDIF

PATCH(UPSTRM,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)

IF (LIGRID) THEN
IF (.NOT.DCONFIG) THEN

```
IF (NX.GT.1) THEN
PATCH(UPSTRU,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRU,HIGH,1,NX,IYL1,IYL2,NZ,NZ,1,LSTEP)
ENDIF
PATCH(UPSTRV,LOW,1,NX,IYF1-1,IYF2-1,1,1,1,LSTEP)
PATCH(DWNSTRV,HIGH,1,NX,IYL1,IYL2-1,NZ,NZ,1,LSTEP)

ELSE
IF (NX.GT.1) THEN
PATCH(UPSTRU,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRU,HIGH,1,NX,IYL1,IYL2-1,NZ,NZ,1,LSTEP)
ENDIF
PATCH(UPSTRV,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRV,HIGH,1,NX,IYL1,IYL2-1,NZ,NZ,1,LSTEP)
ENDIF

ELSE IF (LHGRID)
IF (NX.GT.1) THEN
PATCH(UPSTRU,LOW,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRU,HIGH,1,NX,IYL1,IYL2,NZ,NZ,1,LSTEP)
ENDIF
PATCH(UPSTRV,LOW,1,NX,IYF1,IYF2-1,1,1,1,LSTEP)
PATCH(DWNSTRV,HIGH,1,NX,IYL1,IYL2-1,NZ,NZ,1,LSTEP)
ENDIF

PATCH(UPSTRW,HIGH,1,NX,IYF1,IYF2,1,1,1,LSTEP)
PATCH(DWNSTRM,HIGH,1,NX,IYL1,IYL2,NZ,NZ,1,LSTEP)
PATCH(DWNSTRW,HIGH,1,NX,IYL1,IYL2,NZ-1,NZ-1,1,LSTEP)

COVAL(UPSTRM,P1,FIXFLU,GRND)
COVAL(UPSTRM,V1,ONLYMS,SAME)
COVAL(UPSTRM,W1,ONLYMS,SAME)
COVAL(UPSTRM,H1,GRND,GRND)
COVAL(UPSTRV,V1,GRND,GRND)
COVAL(UPSTRW,W1,GRND,GRND)

IF (NX.GT.1) THEN
COVAL(UPSTRU,U1,ONLYMS,SAME)
COVAL(UPSTRU,U1,GRND,GRND)
ENDIF

COVAL(DWNSTRM,P1,GRND1,GRND4)
COVAL(DWNSTRM,V1,ONLYMS,SAME)
COVAL(DWNSTRM,W1,ONLYMS,SAME)
COVAL(DWNSTRM,H1,ONLYMS,SAME)
COVAL(DWNSTRV,V1,FIXVAL,GRND)

COVAL(DWNSTRV,H1,FIXVAL,GRND)
COVAL(DWNSTRW,W1,GRND,GRND)

PATCH(VDUMMY,PHASEM,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(NDUMMY,NORTH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(SDUMMY,SOUTH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(HDUMMY,HIGH,1,NX,1,NY,1,NZ,1,LSTEP)
PATCH(LDUMMY,LOW,1,NX,1,NY,1,NZ,1,LSTEP)
```


GROUP 14. Downstream pressure for PARAB=.TRUE.

GROUP 15. Termination of sweeps

LSWEEP = 400

RESREF(P1) = 1.E-12

RESREF(U1) = 1.E-12

RESREF(V1) = 1.E-12

RESREF(W1) = 1.E-12

RESREF(H1) = 1.E-12

GROUP 16. Termination of iterations

LITER(P1) = 10;ENDIT(P1) = 0.

ENDIT(V1)=0.;ENDIT(V1)=0.;ENDIT(W1)=0.

GROUP 17. Under-relaxation devices

RELAX(P1,LINRLX,.5)

RELAX(V1,FALSDT,FALST)

RELAX(W1,FALSDT,FALST)

RELAX(H1,FALSDT,FALST)

IF (NX.GT.1) THEN

RELAX(U1,FALSDT,FALST)

ENDIF

GROUP 18. Limits on variables or increments to them

IF (NX.GT.1) THEN

VARMIN(U1) = -2.*W0; VARMAX(U1) = 2.*W0

ENDIF

VARMIN(W1) = -2.*W0; VARMAX(W1) = 5.*W0

VARMIN(V1) = -2.*W0; VARMAX(V1) = 2.*W0

VARMIN(P1) = -100.*RHO1*W0*W0

VARMAX(P1) = 100.*RHO1*W0*W0

GROUP 19. Data communicated by satellite to GROUND

USEGRX=F

THESE DATA TRANSMITTED TO GROUND

LG(1) = 0 for H-grid, 1 for I-grid

LG(1) = LHGRID

LG(2) = LIGRID

LG(3) = LINLINE

LG(4) = ICONFIG

LG(5) = DCONFIG

LG(6) = LCONSTT

RG(1)=DIAM

RG(2)=S1

RG(3)=S2

RG(4)=W0

RG(5)=TBIN

IF (LCONSTT) THEN

RG(6)=TWALL

ELSE

RG(6)=QWALL

ENDIF

RG(7)=KFLUID

RG(8)=CFLUID

RG(9)= AMYCO

RG (10)=AMYRELAX

RG(11)= GEOFAC

RG(12)= RP

GROUP 20. Preliminary print-out

ECHO=T

GROUP 21. Print-out of variables

```
OUTPUT(P1,N,N,N,N,Y,Y)
OUTPUT(U1,N,N,N,N,Y,Y)
OUTPUT(V1,N,N,N,N,Y,Y)
OUTPUT(W1,N,N,N,N,Y,Y)
OUTPUT(H1,N,N,N,N,Y,Y)
  GROUP 22. Spot-value print-out
NPRMON=LSWEEP;IYMON=Y1+1;IZMON=1;TSTSWP=LSWEEP/10
  GROUP 23. Field print-out and plot control
  GROUP 24. Dumps for restarts
  GROUP 25. Debug settings
  RESTRT(P1,U1,V1,W1,H1)
STOP
```

GROUND FILE

C 2 USER DIMENSIONS OWN ARRAYS HERE:

```
PARAMETER (NXD = 20, NYD = 40, NZD = 31)
REAL CONOUT(NYD,NXD), CONIN(NYD,NXD)
REAL CONFRST(NYD,NXD), CONHALO(NYD,NXD)
REAL ASMIDL(NYD,NXD), AREF(NYD,NXD)
REAL ASFRST(NYD,NXD), APFRST(NYD,NXD), ASIN(NYD,NXD)
REAL APREF(NYD,NXD), ASREF(NYD,NXD)
REAL LZIN(NYD,NXD), LZOUT(NYD,NXD)
REAL LZFRST(NYD,NXD), LZLAST(NYD,NXD)
REAL HNHY(NYD,NXD), HNHZ(NYD,NXD), DHNH(NYD,NXD)
REAL HHHY(NYD,NXD), HHHZ(NYD,NXD), DHHH(NYD,NXD)
REAL PHHY(NYD,NXD), PHHZ(NYD,NXD), DPHH(NYD,NXD)
REAL LPY(NYD,NXD), LPZ(NYD,NXD)
REAL DPN(NYD,NXD), DPH(NYD,NXD), DLP(NYD,NXD)
REAL PHY(NYD,NXD), PHZ(NYD,NXD), PNY(NYD,NXD), PNZ(NYD,NXD)
REAL LNLV(NYD,NXD), LNLZ(NYD,NXD)
REAL GWIN(NYD,NXD), HWIN(NYD,NXD), HVIN(NYD,NXD), GVIN(NYD,NXD)
REAL GWOUT(NYD,NXD), HWOUT(NYD,NXD)
REAL GMIN(NYD,NXD), HMIN(NYD,NXD)
REAL GMOUT(NYD,NXD), HMOUT(NYD,NXD)
REAL GMFRST(NYD,NXD), HMFRST(NYD,NXD)
```

C

```
REAL PIFRST(NYD,NXD), VIFRST(NYD,NXD)
REAL WIFRST(NYD,NXD), VALU(NYD,NXD)
REAL UILAST(NYD,NXD), VILAST(NYD,NXD), WILAST(NYD,NXD)
REAL HIFRST(NYD,NXD), UIFRST(NYD,NXD)
REAL HILAST(NYD,NXD), PIDOWN(NYD,NXD), PIHALO(NYD,NXD)
REAL WIMIDL(NYD,NXD), HIMIDL(NYD,NXD), PVAL(NYD,NXD)
REAL WIIN(NYD,NXD), WIOUT(NYD,NXD)
REAL VIIN(NYD,NXD), VIOUT(NYD,NXD), UIIN(NYD,NXD)
REAL VIHALO(NYD,NXD), UIOUT(NYD,NXD)
REAL HIIN(NYD,NXD), HIOUT(NYD,NXD)
```

C

```
REAL AVEC1(3,NZD), AVEC2(3,NZD)
REAL SP1(NZD,NXD), W1WALL1(NZD,NXD), U1WALL1(NZD,NXD)
REAL YWALL1(NZD,NXD), PWALL1(NZD,NXD)
REAL H1WALL1(NZD,NXD), AWALL1(NZD,NXD)
REAL PHI1(NZD,NXD), NUF1(NZD,NXD), CF1(NZD,NXD)
REAL NULM1(NZD,NXD), NUM1(NZD,NXD)
```

```
REAL CP1(NZD,NXD), CFZ1(NZD,NXD)
REAL DIST(2*NZD, NXD), QVALU(2*NZD, NXD)
REAL Q(2*NZD, NXD, 4)
C
REAL SP2(NZD,NXD), W1WALL2(NZD,NXD), U1WALL2(NZD,NXD)
REAL YWALL2(NZD,NXD), PWALL2(NZD,NXD)
REAL H1WALL2(NZD,NXD), AWALL2(NZD,NXD)
REAL PHI2(NZD,NXD), NUF2(NZD,NXD), CF2(NZD,NXD)
REAL NULM2(NZD,NXD), NUM2(NZD,NXD)
REAL CP2(NZD,NXD), CFZ2(NZD,NXD)
C
REAL PT1(3), PT2(3), PT3(3), PT4(3)
REAL XVEC(3), YVEC(3), ZVEC(3), VEC(3)
REAL PN(3), PH(3), LNL(3), NRML(3), TANG(3)
REAL LH(3), LP(3), HNH(3), HHH(3), PHH(3)
C
REAL DIAM, S1, S2, A, B, ATOT, ACYCL
REAL W0, WB, WMIDL
REAL RE, EU, NUBAR, PLAST, CDRP
REAL CPREF, NUFBAR, NUMBAR, NULMBAR
REAL TMIDL, TFU, TBIN, TWALL, TBLAST, LTITO, DTLM
REAL CFLUID, KFLUID
REAL PREF, PP, ST
REAL PI, COSPHI, SINPHI
REAL CMASS, C1HEAT, C2HEAT
REAL V1LST, W1LST, W1LH
REAL DIFCO
REAL CSGMMA, DTHTA, HQ, HR, V1UP, ZETA
INTEGER IZ1, IZ2
INTEGER I, J, KK
INTEGER LOCO, LOVAL, LOMH
LOGICAL LHGRID, LIGRID, LSTAGH
LOGICAL LINLINE, ISVE, LCONSTT
LOGICAL LICONFIG, LSHIFT, LDCONFIG
LOGICAL LZONE1, LZONE2, LZONE3
C
C 3 USER PLACES HIS DATA STATEMENTS HERE
DATA ACYL,NUFBAR,NUMBAR,NULMBAR,CDRP,CDRF/6*0./
COMMON/GRIDTYPE/IGRID,IYF1,IYF2
C*****
C
C--- GROUP 1. RUN TITLE AND OTHER PRELIMINARIES
C
1 GO TO (1001,1002),ISC
1001 CONTINUE
PI = 4.*ATAN(1.0)
C USER MAY HERE CHANGE MESSAGE TRANSMITTED TO THE VDU SCREEN OR
C BATCH-RUN LOG FILE.
IF(IGR.EQ.1.AND.ISC.EQ.1) THEN
CALL WRYT40('GROUND file is GROUND.FTN of: 110191 ')
CALL WRYT40('PHOENICS version number is : 1.6 ')
ENDIF
C
RETURN
1002 CONTINUE
RETURN
C*****
```

```
C
C--- GROUP 13. BOUNDARY CONDITIONS AND SPECIAL SOURCES
C
  13 CONTINUE
    GO TO (130,131,132,133,134,135,136,137,138,139,1310,
    11311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321),ISC
  130 CONTINUE
C----- SECTION 1 ----- COEFFICIENT = GRND
C
C  CALCULATE COEFFICIENT BASED ON HYBRID FORMULATION.
C
  L0CO = L0F(CO)
  DIFCO = RHO1*ENUL*PRNDTL(INDVAR)
  DO 1301 J = IYF, IYL
    DO 1301 I = IXF, IXL
      IJ = L0CO + J + NY*(I-1)
      IF(NPATCH(1:3).EQ.'UPS') THEN
        IF (INDVAR.EQ.W1) THEN
          WW = .5*(CONFRST(J,I)+CONIN(J,I))
          DW = DIFCO/LZFRST(J,I)
          F(L0CO + J + NY*(I-1)) = AMAX1(0., DW - DIFCUT*ABS(WW))
          $+ AMAX1(0.,WW)
        ELSE IF (INDVAR.EQ.V1) THEN
          IF (J.LT.IYF1) THEN
            WV = .5*CONIN(IYF1,I)*ASIN(IYF1,I)/ASIN(J,I)
            DV = .5*DIFCO*ASIN(IYF1,I)/(ASIN(J,I)*LZIN(IYF1,I))
          ELSE IF (J.LT.IYF2) THEN
            WV = .5*(CONIN(J,I)+CONIN(J+1,I))
            DV = .5*(DIFCO/LZIN(J,I)+DIFCO/LZIN(J+1,I))
          ELSE
            WV = CONIN(IYF2,I)
            DV = DIFCO/LZIN(IYF2,I)
          END IF
          IF (J.LT.IYF2) THEN
            F(IJ) = AMAX1(0., DV - DIFCUT*ABS(WV))
            $+ AMAX1(0.,WV)
          ELSE
C  STAGGERED I-GRIDS
            F(IJ) = .5*(AMAX1(0., DV - DIFCUT*ABS(WV))
            $+ .5*AMAX1(0.,WV))
          END IF
        ELSE IF (INDVAR.EQ.U1) THEN
          IF (I.LT.NX) THEN
            WU = .5*(CONIN(J,I)+CONIN(J,I+1))
            DU = .5*(DIFCO/LZIN(J,I)+DIFCO/LZIN(J,I+1))
          ELSE
            WU = CONIN(J,NX)
            DU = DIFCO/LZIN(J,NX)
          END IF
          F(IJ) = AMAX1(0., DU - DIFCUT*ABS(WU))
          $+ AMAX1(0.,WU)
        ELSE
          DD = DIFCO/LZIN(J,I)
          F(IJ) = 0.
          1+AMAX1(0., DD - DIFCUT*ABS(CONIN(J,I)))
        END IF
      ELSE IF(NPATCH(1:3).EQ.'DWN') THEN
```

```
      IF (INDVAR.EQ.W1) THEN
      WW = .5*(CONHALO(J,I)+CONOUT(J,I))
      DW = DIFCO/LZLAST(J,I)
      F(I) = AMAX1(0.,DW - DIFCUT*ABS(WW))
      $+ AMAX1(0.,-1.*WW)
      END IF
      END IF
1301 CONTINUE
      RETURN
131 CONTINUE
C----- SECTION 2 ----- COEFFICIENT = GRND1
C
C   CO IS PROPORTIONAL TO 1/VI
C
      LOCO = LOF(CO)
      DO 1302 J = IYF, IYL
      DO 1302 I = IXF, IXL
      IF (ABS(W1LAST(J,I)).LT.1.E-12) THEN
      F(LOCO + J + NY*(I-1)) = AMYCO/1.E-12
      WRITE (6,*) ' WARNING W1LAST < 1.E-12 AT', I, J
      WRITE (14,*) ' WARNING W1LAST < 1.E-12 AT', I, J
      ELSE
      F(LOCO + J + NY*(I-1)) = AMYCO/ABS(W1LAST(J,I))
      END IF
1302 CONTINUE
      RETURN
132 CONTINUE
C----- SECTION 3 ----- COEFFICIENT = GRND2
      RETURN
133 CONTINUE
C----- SECTION 4 ----- COEFFICIENT = GRND3
      RETURN
134 CONTINUE
C----- SECTION 5 ----- COEFFICIENT = GRND4
      RETURN
135 CONTINUE
C----- SECTION 6 ----- COEFFICIENT = GRND5
      RETURN
136 CONTINUE
C----- SECTION 7 ----- COEFFICIENT = GRND6
      RETURN
137 CONTINUE
C----- SECTION 8 ----- COEFFICIENT = GRND7
      RETURN
138 CONTINUE
C----- SECTION 9 ----- COEFFICIENT = GRND8
      RETURN
139 CONTINUE
C----- SECTION 10 ----- COEFFICIENT = GRND9
      RETURN
1310 CONTINUE
C----- SECTION 11 ----- COEFFICIENT = GRND10
      RETURN
1311 CONTINUE
C----- SECTION 12 ----- VALUE = GRND
C
C   INLET BOUNDARY CONDITIONS.
```

C

```
L0VAL = L0F(VAL)
IF(NPATCH(1:2).EQ.'UP') THEN
  IF (INDVAR.EQ.P1) THEN
    DO 1303 J = IYF, IYL
    DO 1303 I = IXF, IXL
    IF (J.GE.IYF1) THEN
      F(L0VAL + J + NY*(I-1)) = CMASS*RHO1*CONIN(J,I)
    ELSE
      F(L0VAL + J + NY*(I-1)) = 0.
    END IF
  1303 CONTINUE
  ELSE IF (INDVAR.EQ.U1) THEN
    DO 1304 J = IYF, IYL
    DO 1304 I = IXF, IXL
    F(L0VAL + J + NY*(I-1)) = U1IN(J,I)
  1304 CONTINUE
  ELSE IF (INDVAR.EQ.V1) THEN
    DO 1305 J = IYF, IYL
    DO 1305 I = IXF, IXL
    IF (J.LE.IYF2) THEN
      F(L0VAL + J + NY*(I-1)) = GVIN(J,I)*V1IN(J,I)
    $+ HVIN(J,I)*W1IN(J,I)
    ELSE
      F(L0VAL + J + NY*(I-1)) = HVIN(IYF1,I)*W1IN(IYF1,I)
    END IF
  1305 CONTINUE
  ELSE IF (INDVAR.EQ.W1) THEN
    DO 1306 J = IYF, IYL
    DO 1306 I = IXF, IXL
    IF (J.EQ.IYL1) THEN
      V1UP = V1IN(J,I)
    ELSE IF (J.EQ.NY) THEN
      V1UP = V1IN(NY-1,I)
    ELSE
      V1UP = AMIN1(0.,V1IN(J,I)) + AMAX1(0.,V1IN(J-1,I))
    END IF
    F(L0VAL + J + NY*(I-1)) = GWIN(J,I)*V1UP
    $+ HWIN(J,I)*W1IN(J,I)
  1306 CONTINUE
  ELSE IF (INDVAR.EQ.H1) THEN
    DO 1307 J = IYF, IYL
    DO 1307 I = IXF, IXL
    F(L0VAL + J + NY*(I-1)) = H1IN(J,I)
  1307 CONTINUE
  END IF
  ELSE IF (NPATCH(1:3).EQ.'DWN') THEN
    IF (INDVAR.EQ.U1) THEN
      DO 1308 J = IYF, IYL
      DO 1308 I = IXF, IXL
      F(L0VAL + J + NY*(I-1)) = U1OUT(J,I)
    1308 CONTINUE
    ELSE IF (INDVAR.EQ.V1) THEN
      DO 1309 J = IYF, IYL
      DO 1309 I = IXF, IXL
      IF (J.GE.IYL1) THEN
        F(L0VAL + J + NY*(I-1)) = V1OUT(J,I)
```

```
ELSE
  F(L0VAL + J + NY*(I-1)) = DOTZ*W1OUT(IYL1,I)
END IF
1309 CONTINUE
  ELSE IF (INDVAR.EQ.W1) THEN
    DO 1322 J = IYF, IYL
      DO 1322 I = IXF, IXL
        IF (J.EQ.IYL1) THEN
          V1UP = V1OUT(IYL1-1,I)
        ELSE IF (J.EQ.NY) THEN
          V1UP = V1OUT(NY-1,I)
        ELSE
          V1UP = AMIN1(0.,V1OUT(J,I)) + AMAX1(0.,V1OUT(J-1,I))
        END IF
        F(L0VAL + J + NY*(I-1)) = GWOUT(J,I)*V1UP
        $+ HWOUT(J,I)*W1OUT(J,I)
      1322 CONTINUE
    ELSE IF (INDVAR.EQ.H1) THEN
      DO 1323 J = IYF, IYL
        DO 1323 I = IXF, IXL
          F(L0VAL + J + NY*(I-1)) = H1OUT(J,I)
        1323 CONTINUE
      END IF
    END IF
    RETURN
  1312 CONTINUE
C----- SECTION 13 ----- VALUE = GRND1
  RETURN
  1313 CONTINUE
C----- SECTION 14 ----- VALUE = GRND2
  RETURN
  1314 CONTINUE
C----- SECTION 15 ----- VALUE = GRND3
  RETURN
  1315 CONTINUE
C----- SECTION 16 ----- VALUE = GRND4
  L0VAL = L0F(VAL)
  DO 1324 J = IYF, IYL
    DO 1324 I = IXF, IXL
      F(L0VAL + J + NY*(I-1)) = PVAL(J,I)
    1324 CONTINUE
  RETURN
  1316 CONTINUE
C----- SECTION 17 ----- VALUE = GRND5
  RETURN
  1317 CONTINUE
C----- SECTION 18 ----- VALUE = GRND6
  RETURN
  1318 CONTINUE
C----- SECTION 19 ----- VALUE = GRND7
  RETURN
  1319 CONTINUE
C----- SECTION 20 ----- VALUE = GRND8
  RETURN
  1320 CONTINUE
C----- SECTION 21 ----- VALUE = GRND9
  RETURN
```

```
1321 CONTINUE
C----- SECTION 22 ----- VALUE = GRND10
      RETURN
C*****
C
C--- GROUP 19. SPECIAL CALLS TO GROUND FROM EARTH
C
      19 GO TO (191,192,193,194,195,196,197,198),ISC
191 CONTINUE
C * ----- SECTION 1 ---- START OF TIME STEP.
C
C   DATA TRANSMITTED FROM SATELLITE
C
      LHGRID = LG(1)
      LIGRID = LG(2)
      LINLINE = LG(3)
      LICONFIG = LG(4)
      LDCONFIG = LG(5)
      LCONSTT = LG(6)
      LSHIFT = .FALSE.
      IF (LIGRID.AND.(.NOT.LDCONFIG)) LSHIFT = .TRUE.
      DIAM = RG(1)
      S1 = RG(2)
      S2 = RG(3)
      W0 = RG(4)
      TBIN = RG(5)
      IF (.NOT.LCONSTT) QWALL = RG(6)
      IF (LCONSTT) TWALL = RG(6)
      KFLUID = RG(7)
      CFLUID = RG(8)
      AMYCO = RG(9)
      AMYRLX = RG(10)
      SCLGRD = RG(11)
      RP = RG(12)
C
C   GEOMETRIC PARAMETERS RELATED TO GRID TYPE
C
      RESCALE = 1.
      A = S1/DIAM
      B = S2/DIAM
      C = RP/DIAM
      IF (LIGRID) THEN
        ACTOT = S1*RP/4.
      IF (LDCONFIG) THEN
        IZ1 = 1
        IZ2 = NZ-1
        IYF1 = 1
        IYF2 = NY/2
        IYL1 = NY/2+1
        IYL2 = NY
        IF (A.GE.(2*B*B-.5)) THEN
          RESCALE = (A-1)/(SQRT(A*A/4.+B*B)-2.)
        ELSE
          RESCALE = 1.
        END IF
      ELSE
        IZ1 = 1
```



```

IZ2 = NZ-1
IYF1 = INT(1.5*NY*(S2-DIAM)/(S2-DIAM+S1))+1
IYF2 = NY
IYL1 = IYF1
IYL2 = IYF2
RESCALE = A/(A-1.)
ENDIF
IZFRST = 1
IZLAST = NZ-1
IZHALO = NZ
C
ELSE IF (LHGRID) THEN
ACTOT = (S1-DIAM)*RP/4.
IZ1 = INT(SCLGRD*(NZ-1)*3*DIAM/(4.*S2 + 2.*DIAM))
IZ2 = NZ - IZ1
IF (LDCONFIG)
$IZ1 = INT(SCLGRD*(NZ-1)*3*DIAM/(4.*S2 + 2.*DIAM)) - 1
IZ2 = NZ - IZ1
IYF1 = 1
IYF2 = NY
IYL1 = 1
IYL2 = NY
IZFRST = 1
IZLAST = NZ-1
IZHALO = NZ
RESCALE = 1.
IF (A.GE.(2*B*B-.5)) THEN
RESCALE = (A-1)/(SQRT(A*A+4.*B*B)-2.)
END IF
IF (LICONFIG) THEN
ACTOT = S1*RP/4.
IZ1 = INT(SCLGRD*(NZ - 1.)*3.*DIAM/(4.*S2 + 2.*DIAM))-1
IZ1 = NZ/2-IZ1
IZ2 = NZ - IZ1
IYF1 = 1
IYF2 = NY
IYL1 = 1
IYL2 = NY
IZFRST = 1
IZLAST = NZ-1
IZHALO = NZ
RESCALE = A/(A-1.)
END IF
END IF
IF (LHGRID.AND.(.NOT.LICONFIG).AND.(.NOT.LINLINE))
$LSTAGH = .TRUE.

```

C

C INITIALIZE DATA VALUES

C

```

WB = RESCALE*W0
CMASS = 1.
C1HEAT = 1.
IF (LCONSTT) C2HEAT = CFLUID*TWALL*(1. - C1HEAT)
IF (.NOT.LCONSTT) C2HEAT = 0.
DO 1901 J=1,NY
DO 1901 I=1,NX
W1LAST(J,I) = RESCALE*W0

```

```

W1FRST(J,I) = RESCALE*W0
W1IN(J,I) = RESCALE*W0
W1OUT(J,I) = RESCALE*W0
CONIN(J,I) = RESCALE*W0
U1LAST(J,I)= 0.0
U1FRST(J,I)= 0.0
U1IN(J,I)= 0.0
U1OUT(J,I)= 0.0
V1LAST(J,I)= 0.0
V1FRST(J,I)= 0.0
V1IN(J,I)= 0.0
V1OUT(J,I)= 0.0
H1IN(J,I) = CFLUID*TBIN
H1OUT(J,I) = CFLUID*TBIN
H1FRST(J,I) = CFLUID*TBIN
H1LAST(J,I) = CFLUID*TBIN
PVAL(J,I) = 0.
P1HALO(J,I) = 0.
P1DOWN(J,I) = 0.
1901 CONTINUE
    IZMIDL = (NZ-1)/2
C
C   THIS SECTION IS USED TO CALCULATE AREAS AND DISTANCES
C   USED IN THE ITERATIVE PROCEDURE
C
C   ASIN  = LOW CELL AREA AT THE FIRST CELL.
C   ASFRST = HIGH CELL AREA AT THE FIRST CELL.
C   APFRST = HIGH CELL PROJECTED AREA AT THE FIRST CELL.
C   ASMIDL = CELL AREA AT THE MIDDLE CELL.
C   AREF  = REFERENCE AREA USED TO CALCULATE BULK VELOCITY ETC.
C   ASREF = REFERENCE AREA
C   APREF = PROJECTED REFERENCE AREA
C
C   LZIN  = DISTANCE FROM UPSTREAM TO THE FIRST CELL
C   LZFRST = DISTANCE ACROSS THE LAST CELL
C   LZLAST = DISTANCE ACROSS THE LAST CELL
C   LZOUT = DISTANCE FROM THE LAST CELL TO THE HALO CELL
C
CALL GTIZYX(ASURFL,IZFRST,ASIN,NYD,NXD)
CALL GTIZYX(ASURFH,IZFRST,ASFRST,NYD,NXD)
CALL GTIZYX(APROJH,IZFRST,APFRST,NYD,NXD)
CALL GTIZYX(ASURFL,IZMIDL,ASMIDL,NYD,NXD)
CALL GTIZYX(ASURFH,IZLAST,AREF,NYD,NXD)
CALL GTIZYX(ASURFH,IZLAST,ASREF,NYD,NXD)
CALL GTIZYX(APROJH,IZLAST,APREF,NYD,NXD)
CALL GTIZYX(DHZPH,IZFRST,LZFRST,NYD,NXD)
CALL GTIZYX(DHZPH,IZLAST,LZLAST,NYD,NXD)
CALL GTIZYX(DZGPH,IZLAST,LZOUT,NYD,NXD)
c
ATOT = 0.0
DO 1902 J= IYL1, IYL2
DO 1902 I= 1, NX
    LZLAST(J,I) = 2.*LZLAST(J,I)
    LZFRST(J,I) = 2.*LZFRST(J,I)
    ATOT=ATOT+AREF(J,I)
1902 CONTINUE
c

```

```
IF (LINLINE) THEN
CALL SETCX(Y(LZIN,LZOUT,NYD,NXD,1.,0.,NY)
ELSE
CALL SWCHY(LZIN,LZOUT,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(LZIN,IYF1,NYD,NXD,NY,NX)
END IF

C
CALL VECTR(XVEC,1.,0.,0.)
CALL VECTR(YVEC,0.,1.,0.)
CALL VECTR(ZVEC,0.,0.,1.)

C
C THIS MODULE IS WHERE THE CONVECTION FLUX
C FOR NON-ORTHOGONAL GRIDS IS COMPUTED
C
C HQ AND HR CONVERT TANGENTIAL RESOLUTES TO COMPONENTS.
C GM AND HM CONVERT TANGENTIAL RESOLUTES TO NORMAL RESOLUTES
C
IF (NONORT) THEN
CALL GTIZYX(MFCRNH,NZ-1,GMOUT,NYD,NXD)
CALL GTIZYX(MFCSHH,NZ-1,HMOUT,NYD,NXD)
CALL GTIZYX(MFCRNH,1,GMFRST,NYD,NXD)
CALL GTIZYX(MFCSHH,1,HMFRST,NYD,NXD)
ELSE
DO 1903 I = 1, NX
DO 1903 J = 1, NY
GMOUT(J,I) = 0.
HMOUT(J,I) = 1.
GMFRST(J,I) = 0.
HMFRST(J,I) = 1.
1903 CONTINUE
END IF

C
IF (LINLINE) THEN
CALL SETCX(Y(GMIN,GMOUT,NYD,NXD,1.,0.,NY)
CALL SETCX(Y(HMIN,HMOUT,NYD,NXD,1.,0.,NY)
ELSE
CALL GTIZYX(UNIVNN,NZ-1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,NZ-1,PNZ,NYD,NXD)
CALL GTIZYX(UNIVHN,NZ-1,PHY,NYD,NXD)
CALL GTIZYX(UNIVHH,NZ-1,PHZ,NYD,NXD)
IF (LSHIFT) THEN
DO 1904 I = 1, NX
PNYTOP = PNY(IYL1-1,I)
PNZTOP = PNZ(IYL1-1,I)
1904 CONTINUE
END IF
CALL SETCX(Y(VALU,PNY,NYD,NXD,1.,0.,NY-1)
CALL SWCHY(PNY,VALU,NYD,NXD,NY-1,NX)
IF (LSHIFT) CALL SHFTY(PNY,IYL1,NYD,NXD,NY-1,NX)
CALL SETCX(Y(VALU,PNZ,NYD,NXD,-1.,0.,NY-1)
CALL SWCHY(PNZ,VALU,NYD,NXD,NY-1,NX)
IF (LSHIFT) CALL SHFTY(PNZ,IYL1,NYD,NXD,NY-1,NX)
CALL SETCX(Y(VALU,PHY,NYD,NXD,-1.,0.,NY)
CALL SWCHY(PHY,VALU,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(PHY,IYL1,NYD,NXD,NY,NX)
CALL SETCX(Y(VALU,PHZ,NYD,NXD,1.,0.,NY)
CALL SWCHY(PHZ,VALU,NYD,NXD,NY,NX)
```

```
IF (LSHIFT) CALL SHFTY(PHY,IYLI,NYD,NXD,NY,NX)
DO 1905 I = 1, NX
  PNY(NY,I) = PNY(NY-1,I)
  PNZ(NY,I) = PNZ(NY-1,I)
  IF (LSHIFT) THEN
    PNY(NY,I) = PNYTOP
    PNZ(NY,I) = -1.*PNZTOP
  END IF
1905 CONTINUE
  DO 1906 J = 1, NY
    DO 1906 I = 1, NX
      CALL GETPT(1,J+1,1,PT1(1),PT1(2),PT1(3))
      CALL GETPT(1,J,1,PT2(1),PT2(2),PT2(3))
      CALL GETPT(2,J,1,PT3(1),PT3(2),PT3(3))
      CALL GETPT(2,J+1,1,PT4(1),PT4(2),PT4(3))
      CALL NORM2(PT1,PT2,PT3,PT4,VEC)
      CALL VECTR(PN,0.,PNY(J,I),PNZ(J,I))
      CALL VECTR(PH,0.,PHY(J,I),PHZ(J,I))
      CSGMMA = DOT(PH,PN)
      ZETAY = DOT(VEC,PN)
      ZETAZ = DOT(VEC,PH)
      HQ = -1.*CSGMMA/(1. - CSGMMA*CSGMMA)
      HR = 1./(1. - CSGMMA*CSGMMA)
      GMIN(J,I) = HQ*ZETAZ + HR*ZETAY
      HMIN(J,I) = HR*ZETAZ + HQ*ZETAY
    1906 CONTINUE
  END IF
C
C   THIS SECTION IS WHERE THE W-MOMENTUM
C   CURVATURE FACTORS ARE COMPUTED.
C
C   PHY AND PHZ  = COMPONENTS OF P-H. (BETWEEN IZ = 1 AND 2)
C   PNY AND PNZ  = COMPONENTS OF P-N. (AT IZ = 1)
C   NRML        = OUTWARD UNIT NORMAL AT P.
C   CSGMMA       = COSINE OF THE ANGLE BETWEEN W-P AND P-N.
C   HQ AND HR    = FACTORS FOR CONVERSION FROM RESOLUTES TO COMPONENTS.
C   DTHTA        = CURVATURE X ARC LENGTH.
C   GWUP AND HWUP = MULTIPLIERS FOR V.
C
C
C   A) INLET
C
  CALL GTIZYX(UNIVNN,1,PNY,NYD,NXD)
  CALL GTIZYX(UNIVNH,1,PNZ,NYD,NXD)
  CALL GTIZYX(DYGPN,1,DPN,NYD,NXD)
  CALL GTIZYX(UNIVHN,1,PHY,NYD,NXD)
  CALL GTIZYX(UNIVHH,1,PHZ,NYD,NXD)
  CALL GTIZYX(DZGPH,1,DPH,NYD,NXD)
  CALL GTIZYX(UNIVHN,NZ-1,LPY,NYD,NXD)
  CALL GTIZYX(UNIVHH,NZ-1,LPZ,NYD,NXD)
  CALL GTIZYX(DZGPH,NZ-1,DLP,NYD,NXD)
  IF (.NOT.LINLINE) THEN
    IF (.NOT.(LIGRID.AND.LDCONFIG)) THEN
      CALL SETCXV(VALU,LPY,NYD,NXD,-1.,0.,NY)
      CALL SETCXV(LPZ,VALU,NYD,NXD,1.,0.,NY)
    ELSE
      CALL SETCXV(VALU,LPY,NYD,NXD,-1.,0.,NY)
```

```
CALL SWCHY(LPY,VALU,NYD,NXD,NY,NX)
CALL SETCXY(VALU,LPZ,NYD,NXD,1.,0.,NY)
CALL SWCHY(LPZ,VALU,NYD,NXD,NY,NX)
CALL SETCXY(VALU,DLP,NYD,NXD,1.,0.,NY)
CALL SWCHY(DLP,VALU,NYD,NXD,NY,NX)
END IF
END IF
DO 1907 I = 1, NX
  PNY(NY,I) = PNY(NY-1,I)
  PNZ(NY,I) = PNY(NY-1,I)
  DPN(NY,I) = DPN(NY-1,I)
  DO 1907 J = 1, NY
    CALL VECTR(LP, 0., DLP(J,I)*LPY(J,I), DLP(J,I)*LPZ(J,I))
    CALL VECTR(PN, 0., DPN(J,I)*PNY(J,I), DPN(J,I)*PNZ(J,I))
    CALL VECTR(PH, 0., DPH(J,I)*PHY(J,I), DPH(J,I)*PHZ(J,I))
    CALL ADDVEC(LP,PH,LH)
    CALL NORM(LP)
    CALL NORM(PH)
    CALL NORM(PN)
    CALL NORM(LH)
    CALL CROSS(LP,LH,VEC)
    CALL CROSS(LH,VEC,NRML)
    CALL NORM(NRML)
    CSGMMA = DOT(LP,PN)
    DELTA = ACOS(DOT(LP,PH))
    DTHTA = 2.*SIN(DELTA/2.)
    HQ = CSGMMA/(1. - CSGMMA*CSGMMA)
    HR = -1./(1. - CSGMMA*CSGMMA)
    GWIN(J,I) = DTHTA*(HR*DOT(NRML,PN) + HQ*DOT(NRML,LP))
    HWIN(J,I) = 1. + DTHTA*(HQ*DOT(NRML,PN) + HR*DOT(NRML,LP))
    IF ((LIGRID.AND.LDCONFIG).AND.(J.EQ.IYF2)) THEN
      END IF
1907 CONTINUE
C
C   B) OUTLET
C
  IF (LINLINE) THEN
    CALL SETCXY(GWOUT,GWIN,NYD,NXD,1.,0.,NY)
    CALL SETCXY(HWOUT,HWIN,NYD,NXD,1.,0.,NY)
  ELSE
    IF (LSHIFT) THEN
      CALL GTIZYX(UNIVNN,NZ-1,HNHY,NYD,NXD)
      CALL GTIZYX(UNIVNH,NZ-1,HNHZ,NYD,NXD)
      CALL GTIZYX(DYGPN,NZ-1,DHNNH,NYD,NXD)
      DO 1908 I = 1, NX
        HNHYTOP = HNHY(IYL1-1,I)
        HNHZTOP = HNHZ(IYL1-1,I)
        DHNNHTOP = DHNNH(IYL1-1,I)
1908 CONTINUE
      ENDIF
      CALL GTIZYX(UNIVNN,NZ,HNHY,NYD,NXD)
      CALL GTIZYX(UNIVNH,NZ,HNHZ,NYD,NXD)
      CALL GTIZYX(DYGPN,NZ,DHNNH,NYD,NXD)
      CALL GTIZYX(UNIVHN,NZ-1,PHY,NYD,NXD)
      CALL GTIZYX(UNIVHH,NZ-1,PHZ,NYD,NXD)
      CALL GTIZYX(DZGPH,NZ-1,DPH,NYD,NXD)
      CALL GTIZYX(UNIVHN,1,HHHY,NYD,NXD)
```

```

CALL GTIZYX(UNTVHH,1,HHHZ,NYD,NXD)
CALL GTIZYX(DZGPH,1,DHHH,NYD,NXD)
CALL SETCX(Y,VALU,HHHY,NYD,NXD,-1.,0.,NY)
CALL SWCHY(HHHY,VALU,NYD,NXD,NY,NX)
CALL SETCX(Y,VALU,HHHZ,NYD,NXD,1.,0.,NY)
CALL SWCHY(HHHZ,VALU,NYD,NXD,NY,NX)
CALL SETCX(Y,VALU,DHHH,NYD,NXD,1.,0.,NY)
CALL SWCHY(DHHH,VALU,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(HHHY,IYL1,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(HHHZ,IYL1,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(DHHH,IYL1,NYD,NXD,NY,NX)
DO 1909 I = 1, NX
  HNHY(NY,I) = HNHY(NY-1,I)
  HNHZ(NY,I) = HNHZ(NY-1,I)
  DHNH(NY,I) = DHNH(NY-1,I)
  IF (LSHIFT) THEN
    HNHY(NY,I) = HNHYTOP
    HNHZ(NY,I) = HNHZTOP
    DHNH(NY,I) = DHNHSTOP
  END IF
DO 1909 J = 1, NY
  CALL VECTR(PH,0.,DPH(J,I)*PHY(J,I),DPH(J,I)*PHZ(J,I))
  CALL VECTR(HNH,0.,DHNH(J,I)*HNHY(J,I),DHNH(J,I)*HNHZ(J,I))
  CALL VECTR(HHH,0.,DHHH(J,I)*HHHY(J,I),DHHH(J,I)*HHHZ(J,I))
  CALL ADDVEC(PH,HHH,PHH)
  CALL NORM(PH)
  CALL NORM(HNH)
  CALL NORM(HHH)
  CALL NORM(PHH)
  CALL CROSS(PH,PHH,VEC)
  CALL CROSS(PHH,VEC,NRML)
  CALL NORM(NRML)
  CSGMMA = DOT(HHH,HNH)
  DELTA = ACOS(DOT(PH,HHH))
  DTHTA = 2.*SIN(DELTA/2.)
  HQ = CSGMMA/(1. - CSGMMA*CSGMMA)
  HR = -1./(1. - CSGMMA*CSGMMA)
  GWOUT(J,I) = DTHTA*(HR*DOT(NRML,HNH) + HQ*DOT(NRML,HHH))
  HWOUT(J,I) = 1. + DTHTA*(HQ*DOT(NRML,HNH) + HR*DOT(NRML,HHH))
  IF ((LIGRID.AND.LDCONFIG).AND.(J.EQ.IYL1)) THEN
    GWOUT(J,I) = 0.
    HWOUT(J,I) = DOT(PH,HHH)
  END IF
1909 CONTINUE
END IF
C
C THIS SECTION RELATES TO GEOMETRY USED TO ACCOUNT FOR
C DIVERGENCE OF THE GRID AT THE BOUNDARY.
C
C ASSIGNMENT OF V_LOW AT IZ = 1
C
C PNY AND PNZ = COMPONENTS OF P-N.
C LNLV AND LNLZ = COMPONENTS OF L-NL
C CSGMMA = COSINE OF THE ANGLE BETWEEN L-NL AND L-P
C HQ AND HR = FACTORS FOR CONVERSION FROM RESOLUTES TO COMPONENTS.
C HVIN AND GVIN = MULTIPLIERS FOR V RESPECTIVELY (UPSTREAM)
C

```

```
CALL GTIZYX(UNIVNN,1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,1,PNZ,NYD,NXD)
CALL GTIZYX(UNIVNN,NZ-1,LNLY,NYD,NXD)
CALL GTIZYX(UNIVNH,NZ-1,LNLZ,NYD,NXD)
CALL GTIZYX(UNIVHN,NZ-1,LPY,NYD,NXD)
CALL GTIZYX(UNIVHH,NZ-1,LPZ,NYD,NXD)
IF (.NOT.LINLINE) THEN
CALL SETCX(Y,VALU,LNLY,NYD,NXD,1.,0.,NY-1)
CALL SWCHY(LNLY,VALU,NYD,NXD,NY-1,NX)
IF (LSHIFT) CALL SHFTY(LNLY,IYL1,NYD,NXD,NY-1,NX)
CALL SETCX(Y,VALU,LNLZ,NYD,NXD,-1.,0.,NY-1)
CALL SWCHY(LNLZ,VALU,NYD,NXD,NY-1,NX)
IF (LSHIFT) CALL SHFTY(LNLZ,IYL1,NYD,NXD,NY-1,NX)
CALL SETCX(Y,VALU,LPY,NYD,NXD,-1.,0.,NY)
CALL SWCHY(LPY,VALU,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(LPY,IYL1,NYD,NXD,NY,NX)
CALL SETCX(Y,VALU,LPZ,NYD,NXD,1.,0.,NY)
CALL SWCHY(LPZ,VALU,NYD,NXD,NY,NX)
IF (LSHIFT) CALL SHFTY(LPZ,IYL1,NYD,NXD,NY,NX)
IF (LSHIFT) THEN
DO 1910 I = 1, NX
LNLY(IYF1-1,I) = LNLY(IYF1,I)
LNLZ(IYF1-1,I) = LNLZ(IYF1,I)
LPY(IYF1-1,I) = LPY(IYF1,I)
LPZ(IYF1-1,I) = LPZ(IYF1,I)
1910 CONTINUE
END IF
END IF
DO 1911 J = 1, NY
DO 1911 I = 1, NX
CALL VECTR(PN, 0., PNY(J,I), PNZ(J,I))
CALL VECTR(LNL, 0., LNLY(J,I), LNLZ(J,I))
CALL VECTR(LP, 0., LPY(J,I), LPZ(J,I))
CSGMMA = DOT(LP,LNL)
HQ = -1.*CSGMMA/(1. - CSGMMA*CSGMMA)
HR = 1./(1. - CSGMMA*CSGMMA)
HVIN(J,I) = HQ*DOT(LNL,PN) + HR*DOT(PN,LP)
GVIN(J,I) = HR*DOT(LNL,PN) + HQ*DOT(PN,LP)
IF ((LIGRID.AND.LDCONFIG).AND.(J.EQ.IYF2)) THEN
CALL GTIZYX(YP,1,VALU,NYD,NXD)
DELTAY = S1/2. - VALU(IYF2,I)
PNDOTZ = DOT(PN,ZVEC)
END IF
IF ((LIGRID.AND.LICONFIG).AND.(J.LE.IYF1-1)) THEN
GVIN(J,I) = 0.
HVIN(J,I) = DOT(PN,ZVEC)
PNDOTZ = DOT(PN,ZVEC)
END IF
1911 CONTINUE
C
IF (LIGRID) THEN
CALL GTIZYX(UNIVNN,NZ-1,PNY,NYD,NXD)
CALL GTIZYX(UNIVNH,NZ-1,PNZ,NYD,NXD)
DO 1912 I = 1, NX
CALL VECTR(PN,0.,PNY(IYL1-1,I),PNZ(IYL1-1,I))
HVOUTX = DOT(ZVEC,PN)
1912 CONTINUE
```

```
END IF
C
C  COMPUTE COEFFICENTS PER UNIT SURFACE (NOT PROJECTED) AREA
C
DO 1913 J = 1, NY
DO 1913 I = 1, NX
GMFRST(J,I) = GMFRST(J,I)*APFRST(J,I)/ASFRST(J,I)
HMFRST(J,I) = HMFRST(J,I)*APFRST(J,I)/ASFRST(J,I)
GMOUT(J,I) = GMOUT(J,I)*APREF(J,I)/ASREF(J,I)
HMOUT(J,I) = HMOUT(J,I)*APREF(J,I)/ASREF(J,I)
1913 CONTINUE
C
IF (IYF1.GT.1) THEN
DO 1915 I = 1, NX
DO 1914 J = 1, IYF1-2
HMIN(J,I) = 0.
GMIN(J,I) = 0.
HWIN(J,I) = 0.
GWIN(J,I) = 0.
HVIN(J,I) = 0.
GVIN(J,I) = 0.
LZFRST(J,I) = LZFRST(IYF1,I)
LZIN(J,I) = LZIN(IYF1,I)
1914 CONTINUE
HMIN(IYF1-1,I) = 0.
GMIN(IYF1-1,I) = 0.
HWIN(IYF1-1,I) = 0.
GWIN(IYF1-1,I) = 0.
LZFRST(IYF1-1,I) = LZFRST(IYF1,I)
LZIN(IYF1-1,I) = LZIN(IYF1,I)
IF (.NOT.LIGRID) HVIN(IYF1-1,I) = 0
IF (.NOT.LIGRID) GVIN(IYF1-1,I) = 0
1915 CONTINUE
END IF
IF (IYF2.LT.NY) THEN
DO 1916 I = 1, NX
DO 1916 J = IYF2+1, NY
HMIN(J,I) = 0.
GMIN(J,I) = 0.
HWIN(J,I) = 0.
GWIN(J,I) = 0.
HVIN(J,I) = 0.
GVIN(J,I) = 0.
1916 CONTINUE
END IF
C
IF (IYL1.GT.1) THEN
DO 1917 I = 1, NX
DO 1917 J = 1, IYL1-1
AREF(J,I) = 0.
HMOUT(J,I) = 0.
GMOUT(J,I) = 0.
HWOUT(J,I) = 0.
GWOUT(J,I) = 0.
LZOUT(J,I) = LZOUT(IYL1,I)
LZLAST(J,I) = LZLAST(IYL1,I)
1917 CONTINUE
```



```

END IF
IF (IYL2.LT.NY) THEN
DO 1918 I = 1, NX
DO 1918 J = IYL2+1, NY
ASREF(J,I) = 1.E-20
APREF(J,I) = 1.E-20
AREF(J,I) = 1.E-20
HMOUT(J,I) = 0.
GMOUT(J,I) = 0.
LZOUT(J,I) = LZOUT(IYL2,I)
LZLAST(J,I) = LZLAST(IYL2,I)
1918 CONTINUE
END IF
C
C THIS SECTION IS WHERE GEOMETRIC FACTORS USED TO CALCULATE LOCAL
C COEFFICIENTS AROUND A TEST CYLINDER ARE COMPUTED.
C
C PHI(I) = ANGLE FROM FRONT LEADING EDGE OF CYLINDER (DEG)
C AWALL(I) = AREA OF CYLINDER WALL
C YWALL(I) = PERPENDICULAR DISTANCE FROM OUTSIDE CELL TO WALL
C
C FOR IN-LINE CONFIGURED GRIDS ONLY VALUES AT IY = 1 REQUIRE TO
C BE COMPUTED AND STORED. FOR DIAGONALLY CONFIGURED GRIDS VALUES
C AT IY = NY ALSO NEED TO BE STORED.
C
PT4(1) = 0.
PT4(2) = 0.
PT4(3) = 0.
DO 1919 II = 1, NX
DO 1919 KK = 1, 2*NZ
CALL GTIZYX(XP,1,VALU,NYD,NXD)
DIST(KK,II) = VALU(1,II)
1919 CONTINUE
DO 1920 II = 1, NX
DO 1920 KK = 1, NZ
NJWALL = 1
CALL GTIZYX(DHYPN,KK,VALU,NYD,NXD)
YWALL1(KK,II) = VALU(NJWALL,II)
IF (NJWALL.EQ.1) CALL GTIZYX(ASURFS,KK,VALU,NYD,NXD)
AWALL1(KK,II) = VALU(NJWALL,II)
CALL GETPT(1,NJWALL,KK,PT1(1),PT1(2),PT1(3))
CALL GETPT(1,NJWALL,KK+1,PT2(1),PT2(2),PT2(3))
PT3(1) = 0.
PT3(2) = .5*(PT1(2) + PT2(2)) - PT4(2)
PT3(3) = .5*(PT1(3) + PT2(3)) - PT4(3)
CALL VECTR(VEC,0.,PT3(2),PT3(3))
CALL NORM(VEC)
AVEC1(1,KK) = VEC(1)
AVEC1(2,KK) = VEC(2)
AVEC1(3,KK) = VEC(3)
COSPFI = -1.*DOT(VEC,ZVEC)
PHI1(KK,II) = 180.*ACOS(COSPFI)/PI
1920 CONTINUE
C
IF (LDCONFIG) THEN
PT4(1) = 0.
PT4(2) = S1/2.

```

```
PT4(3) = S2
ELSE IF (LHGRID.AND.(.NOT.LICONFIG)) THEN
PT4(1) = 0.
PT4(2) = 0.
PT4(3) = S2
END IF
```

C

```
DO 1921 II = 1, NX
DO 1921 KK = 1, NZ
  NJWALL = 1
  IF (LDCONFIG) NJWALL = NY
  CALL GTIZYX(DHYPN, KK, VALU, NYD, NXD)
  YWALL2(KK, II) = VALU(NJWALL, II)
  IF (NJWALL.EQ.1) CALL GTIZYX(ASURFS, KK, VALU, NYD, NXD)
  IF (NJWALL.EQ.NY) CALL GTIZYX(ASURFN, KK, VALU, NYD, NXD)
  AWALL2(KK, II) = VALU(NJWALL, II)
  IF (LDCONFIG) NJWALL = NY+1
  CALL GETPT(1, NJWALL, KK, PT1(1), PT1(2), PT1(3))
  CALL GETPT(1, NJWALL, KK+1, PT2(1), PT2(2), PT2(3))
  PT3(1) = 0.
  PT3(2) = .5*(PT1(2) + PT2(2)) - PT4(2)
  PT3(3) = .5*(PT1(3) + PT2(3)) - PT4(3)
  CALL VECTR(VEC, 0., PT3(2), PT3(3))
  CALL NORM(VEC)
  AVEC2(1, KK) = VEC(1)
  AVEC2(2, KK) = VEC(2)
  AVEC2(3, KK) = VEC(3)
  COSPHI = -1.*DOT(VEC, ZVEC)
  PHI2(KK, II) = 180.*ACOS(COSPHI)/PI
1921 CONTINUE
```

c

c WRITE INITIAL DATA TO VDU AND PHIDA FILE

C

```
914 FORMAT (/,'      A = ',G11.4,'    B = ',G11.4,
$'    C = ',G11.4)
951 FORMAT ('      IZ1 = ',I3,7X,'    IZ2 = ',I3,/)
915 FORMAT (' CONSTANT WALL HEAT FLUX')
918 FORMAT (' CONSTANT WALL TEMPERATURE')
949 FORMAT(' SWEEP EU  NU  RE ')
  WRITE (6,914) A, B, C
  WRITE (6,951) IZ1, IZ2
  WRITE (14,914) A, B, C
  WRITE (14,951) IZ1, IZ2
  IF (LCONSTT) THEN
    WRITE (6,918)
    WRITE (14,918)
  ELSE
    WRITE (6,915)
    WRITE (14,915)
  END IF
  RETURN
192 CONTINUE
C * ----- SECTION 2 ---- START OF SWEEP.
  RETURN
193 CONTINUE
C * ----- SECTION 3 ---- START OF IZ SLAB.
  RETURN
```

```
194 CONTINUE
C * ----- SECTION 4 ---- START OF ITERATION.
  RETURN
195 CONTINUE
C * ----- SECTION 5 ---- FINISH OF ITERATION.
  RETURN
196 CONTINUE
C * ----- SECTION 6 ---- FINISH OF IZ SLAB.
C
C   GET VALUES AT FIRST CELL
C
  IF (IZ.EQ.IZFRST) THEN
    CALL GETYX(P1,P1FRST,NYD,NXD)
    IF (NX.GT.1) CALL GETYX(U1,U1FRST,NYD,NXD)
    CALL GETYX(V1,V1FRST,NYD,NXD)
    CALL GETYX(H1,H1FRST,NYD,NXD)
    CALL GETYX(W1,W1FRST,NYD,NXD)
    DO 1922 I = 1, NX
      V1FRST(NY,I) = V1FRST(NY-1,I)
1922 CONTINUE
C
C   CALCULATE DOWNSTREAM VALUES.
C
  IF (.NOT.LINLINE) THEN
    CALL SETCX(Y,VALU,P1FRST,NYD,NXD,1,-1.*PFRST,NY)
    CALL SWCHY(P1DOWN,VALU,NYD,NXD,NY,NX)
    CALL SWCHY(U1OUT,U1FRST,NYD,NXD,NY,NX)
    CALL SWCHY(W1OUT,W1FRST,NYD,NXD,NY,NX)
    CALL SWCHY(VALU,H1FRST,NYD,NXD,NY,NX)
    CALL SETCX(Y,H1OUT,VALU,NYD,NXD,1./C1HEAT,
$-1.*C2HEAT/C1HEAT,NY)
    IF (.NOT.LSHIFT) THEN
      CALL SETCX(Y,VALU,V1FRST,NYD,NXD,-1.,0.,NY-1)
      CALL SWCHY(V1OUT,VALU,NYD,NXD,NY-1,NX)
    ELSE
      CALL SETCX(Y,VALU,V1FRST,NYD,NXD,-1.,0.,NY)
      CALL SWCHY(V1OUT,VALU,NYD,NXD,NY,NX)
      CALL SHFTY(V1OUT,IYL1-1,NYD,NXD,NY,NX)
      CALL SHFTY(U1OUT,IYL1,NYD,NXD,NY,NX)
      CALL SHFTY(W1OUT,IYL1,NYD,NXD,NY,NX)
      CALL SHFTY(H1OUT,IYL1,NYD,NXD,NY,NX)
      CALL SHFTY(P1DOWN,IYL1,NYD,NXD,NY,NX)
    END IF
    ELSE
      CALL SETCX(Y,VALU,V1FRST,NYD,NXD,-1.,0.,NY)
      CALL SWCHY(V1OUT,VALU,NYD,NXD,NY,NX)
      CALL SETCX(Y,P1DOWN,P1FRST,NYD,NXD,1,-1.*PFRST,NY)
      CALL SETCX(Y,U1OUT,U1FRST,NYD,NXD,1.,0,NY)
      CALL SETCX(Y,V1OUT,V1FRST,NYD,NXD,1.,0,NY-1)
      CALL SETCX(Y,W1OUT,W1FRST,NYD,NXD,1.,0.,NY)
      CALL SETCX(Y,H1OUT,H1FRST,NYD,NXD,
$1./C1HEAT,-1.*C2HEAT/C1HEAT,NY)
    END IF
C
C   COMPUTE REFERENCE PRESSURE FOR PRESSURE COEFFICIENT (I-GRIDS)
C
  IF (LIGRID) THEN
```

```
IYREF = 1
IF (LDCONFIG) IYREF = NY
IXREF = 1
CPREF = P1FRST(IYREF,IXREF)
CPREF = 2.*CPREF/(RHO1*WB*WB)
WMIDL = 0.
TMIDL = 0.
C
C  COMPUTE VELOCITY AT MINIMUM CROSS-SECTION (DIAGONAL I-GRIDS ONLY)
C
DO 1923 J = IYL1, IYL2
DO 1923 I = 1, NX
WMIDL = WMIDL+ AREF(J,I)*W1LAST(J,I)
TMIDL = TMIDL+ AREF(J,I)*W1LAST(J,I)*H1FRST(J,I)
1923 CONTINUE
IF (ABS(WMIDL).GT.1.E-06) TMIDL = TMIDL/(CFLUID*WMIDL)
IF (TMIDLEQ.TWALL) TMIDL = 1.E-06
END IF
END IF
IF (.NOT.LINLINE) TMIDL = (TBIN+TBLAST)/2.
C
C  CALCULATE BULK VELOCITY AND TEMPERATURE AT NZ/2
C
IF (IZ.EQ.(NZ-1)/2) THEN
CALL GETYX(W1,W1MIDL,NYD,NXD)
IF (STORE(H1)) CALL GETYX(H1,H1MIDL,NYD,NXD)
WMIDL = 0.
TMIDL = 0.
DO 1924 J = 1, NY
DO 1924 I = 1, NX
WMIDL = WMIDL+ ASMIDL(J,I)*W1MIDL(J,I)
TMIDL = TMIDL+ ASMIDL(J,I)*W1MIDL(J,I)*H1MIDL(J,I)
1924 CONTINUE
IF (ABS(WMIDL).GT.1.E-06) TMIDL = TMIDL/(CFLUID*WMIDL)
IF (TMIDLEQ.TWALL) TMIDL = 1.E-06
END IF
IF (.NOT.LINLINE) TMIDL = (TBIN+TBLAST)/2.
C
C  COMPUTE REFERENCE PRESSURE FOR PRESSURE COEFFICIENT (H-GRIDS)
C
IF (LHGRID) THEN
IF (((.NOT.LICONFIG).AND.(IZ.EQ.IZ2)).
$OR.(LICONFIG.AND.(IZ.EQ.IZ1))) THEN
IYREF = 1
IF (LSTAGH) IYREF = NY
IXREF = NX
IF (STORE(P1)) CALL GETONE(P1,CPREF,IYREF,IXREF)
CPREF = 2.*CPREF/(RHO1*WB*WB)
END IF
END IF
C
C  GET VALUES AT LAST CELL
C
IF (IZ.EQ.IZLAST) THEN
IF (NX.GT.1) CALL GETYX(U1,U1LAST,NYD,NXD)
IF (NY.GT.1) CALL GETYX(V1,V1LAST,NYD,NXD)
DO 1925 I = 1, NX
```

```
V1LAST(NY,I) = V1LAST(NY-1,I)
1925 CONTINUE
  IF (NZ.GT.1) CALL GETYX(W1,W1LAST,NYD,NXD)
  IF (STORE(H1)) CALL GETYX(H1,H1LAST,NYD,NXD)
C
C  CALCULATE INLET BOUNDARY VALUES.
C
  IF (.NOT.LINLINE) THEN
    CALL SWCHY(U1IN,U1LAST,NYD,NXD,NY,NX)
    CALL SWCHY(W1IN,W1LAST,NYD,NXD,NY,NX)
    CALL SWCHY(VALU,H1LAST,NYD,NXD,NY,NX)
    CALL SETCXY(H1IN,VALU,NYD,NXD,C1HEAT,C2HEAT,NY)
    IF (.NOT.LSHIFT) THEN
      CALL SETCXY(VALU,V1LAST,NYD,NXD,-1.,0.,NY-1)
      CALL SWCHY(V1IN,VALU,NYD,NXD,NY-1,NX)
    ELSE
      CALL SETCXY(VALU,V1LAST,NYD,NXD,-1.,0.,NY)
      CALL SWCHY(V1IN,VALU,NYD,NXD,NY,NX)
      CALL SHFTY(U1IN,IYL1,NYD,NXD,NY,NX)
      CALL SHFTY(V1IN,IYL1-1,NYD,NXD,NY,NX)
      CALL SHFTY(W1IN,IYL1,NYD,NXD,NY,NX)
      CALL SHFTY(H1IN,IYL1,NYD,NXD,NY,NX)
    END IF
    ELSE
      CALL SETCXY(U1IN,U1LAST,NYD,NXD,1.,0.,NY)
      CALL SETCXY(V1IN,V1LAST,NYD,NXD,1.,0.,NY-1)
      CALL SETCXY(W1IN,W1LAST,NYD,NXD,1.,0.,NY)
      CALL SETCXY(H1IN,H1LAST,NYD,NXD,C1HEAT,C2HEAT,NY)
    END IF
    IF (IYF1.GT.1) THEN
      DO 1926 II = 1, NX
        DO 1926 JJ = 1, IYF1-1
          W1IN(JJ,II) = CONIN(IYF1,II)
1926 CONTINUE
        END IF
C
C  CALCULATE BULK VELOCITY AND TEMPERATURE BASED ON EXIT VALUES
C
C  CONOUT      = MASS FLUX OUT OF LAST CELL
C  CONFRST     = MASS FLUX OUT OF FIRST CELL
C  CONFRST     = MASS FLUX OUT OF FIRST CELL
C  CONHALO     = MASS FLUX OUT OF HALO CELL
C
C  WB          = BULK VELOCITY AT CURRENT SWEEP
C  TBLAST      = BULK TEMPERATURE AT LAST CELL
C
    WB = 0.
    TBLAST = 0.0
    DO 1927 J = IYL1, IYL2
      DO 1927 I = 1, NX
        CONOUT(J,I) = GMOUT(J,I)*V1LAST(J,I)
        $+ HMOUT(J,I)*W1LAST(J,I)
        CONFRST(J,I) = GMFRST(J,I)*V1FRST(J,I)
        $+ HMFRST(J,I)*W1FRST(J,I)
        WB = WB + AREF(J,I)*CONOUT(J,I)
        TBLAST = TBLAST+AREF(J,I)*CONOUT(J,I)*H1LAST(J,I)
1927 CONTINUE
```

```
WB = WB/ACTOT
TBLAST = TBLAST/(CFLUID*WB*ACTOT)
CMASS = W0/WB
WB = WB*RESCALE
C
C  COMPUTE INLET CONVECTION FLUX
C
  IF (.NOT.LINLINE) THEN
    CALL SWCHY(CONIN,CONOUT,NYD,NXD,NY,NX)
    CALL SWCHY(CONHALO,CONFRST,NYD,NXD,NY,NX)
    IF (LSHIFT) THEN
      CALL SHFTY(CONIN,IYF1,NYD,NXD,NY,NX)
      CALL SHFTY(CONHALO,IYL1,NYD,NXD,NY,NX)
    END IF
    ELSE
      CALL SETCXY(CONIN,CONOUT,NYD,NXD,1.,0.,NY)
      CALL SETCXY(CONHALO,CONFRST,NYD,NXD,1.,0.,NY)
    END IF
C
C  COMPUTE ENTHALPY COEFFICENTS
C
  IF (LCONSTT) THEN
    IF (ABS(TBLAST - TWALL).LT.1.E-06) THEN
      TBLAST = TWALL + 1.E-06*(TBIN - TWALL)
      C1HEAT = TBIN/(TWALL + 1.E-06)
      C2HEAT = 1.E-06
    ELSE
      C1HEAT = (TBIN - TWALL)/(TBLAST - TWALL)
      C2HEAT = CFLUID*TWALL*(1. - C1HEAT)
    END IF
    TFU = TWALL + (TBIN - TWALL)*(TMIDL - TWALL)/(TBLAST - TWALL)
  ELSE
    C1HEAT = 1.
    C2HEAT = CFLUID*(TBIN - TBLAST)
    TFU = TMIDL - QWALL*PI/(RHO1*(A-1))
  END IF
C
  END IF
C
C  GET VALUES AT HALO CELL
C
  IF (IZ.EQ.IZHALO) THEN
    CALL GETYX(P1,P1HALO,NYD,NXD)
    CALL GETYX(V1,V1HALO,NYD,NXD)
C
C  COMPUTE MEAN UPSTREAM AND DOWNSTREAM PRESSURES
C
  PFRST = 0.
  PLAST = 0.
  RSUM = FLOAT((IYL2+1-IYL1)*NX)
  IF (LINLINE) THEN
    DO 1928 J = IYL1, IYL2
      DO 1928 I = 1, NX
        PFRST = PFRST + P1FRST(J,I)
        PLAST = PLAST + P1HALO(J,I)
      1928 CONTINUE
    PFRST = PFRST/RSUM
```

```

PLAST = PLAST/RSUM
ELSE
DO 1929 J = IYF1, IYF2
DO 1929 I = 1, NX
PFRST = PFRST + P1FRST(J,I)
PLAST = PLAST + P1HALO(J,I)
1929 CONTINUE
PFRST = PFRST/RSUM
PLAST = PLAST/RSUM
END IF
END IF

C
C ----- THIS SECTION EXECUTED ONLY FOR ISWEEP = LSWEEP -----
C      FOR ALL VALUES OF IZ
C
      IF((ISWEEP.EQ.LSWEEP).OR.(ENUF5W)) THEN
DHBAR = CFLUID*(TBIN-TBLAST)
ITHTA = INAME('THTA')
IF (STORE(ITHTA))
$CALL FN2(ITHTA,H1,-1.*CFLUID*TBIN/DHBAR,1/DHBAR)
C
C  CALCULATE LOCAL PRESSURE COEFFICIENT IN Z AND ZCRT DIRECTIONS.
C
DO 1930 II = 1, NX
IF (STORE(P1)) CALL GETONE(P1,PWALL1(IZ,II),1,II)
IF (NX.GT.1) CALL GETONE(U1,U1WALL1(IZ,II),1,II)
IF (NZ.GT.1) CALL GETONE(W1,W1WALL1(IZ,II),1,II)
IF (STORE(H1)) CALL GETONE(H1,H1WALL1(IZ,II),1,II)
IF (LDCONFIG) THEN
IF (STORE(P1)) CALL GETONE(P1,PWALL2(IZ,II),NY,II)
IF (NX.GT.1) CALL GETONE(U1,U1WALL2(IZ,II),NY,II)
IF (NZ.GT.1) CALL GETONE(W1,W1WALL2(IZ,II),NY,II)
IF (STORE(H1)) CALL GETONE(H1,H1WALL2(IZ,II),NY,II)
ELSE
IF (STORE(P1)) CALL GETONE(P1,PWALL2(IZ,II),1,II)
IF (NX.GT.1) CALL GETONE(U1,U1WALL2(IZ,II),1,II)
IF (NZ.GT.1) CALL GETONE(W1,W1WALL2(IZ,II),1,II)
IF (STORE(H1)) CALL GETONE(H1,H1WALL2(IZ,II),1,II)
END IF
IF (IZ.EQ.NZ) W1WALL1(IZ,II) = 0.
IF (IZ.EQ.NZ) W1WALL2(IZ,II) = 0.
IF ((IZ.EQ.NZ-1).AND.LIGRID) W1WALL1(IZ,II) = 0.
1930 CONTINUE
C
IF (STORE(NNUSS)) L0NUSS = L0F(NNUSS)
C
C  CALCULATE LOCAL COEFFICIENTS
C
C  THREE DIFFERENT ZONES ARE CONSIDERED DEPENDING ON THE GRID
C  TYPE AND REFERENCE FLUID VALUES (BULK VELOCITY ETC)
C
C  ZONE1 = T FOR H-GRIDS AND STAGGERED I-GRIDS
C  WHEN THERE IS SOLID WALL AT IY = 1 (90 < PHI < 180)
C  ZONE2 = T FOR STAGGERED H- AND I-GRIDS
C  WHEN THERE IS SOLID WALL AT IY = NY (0 < PHI < 90)
C  ZONE3 = T FOR IN-LINE I-GRIDS
C  WHEN THERE IS SOLID WALL AT IY = 1 (0 < PHI < 180)

```

```

C
C LOCAL COEFFICIENTS ARE STORED IN CP1, CF1, ETC.
C IF LZONE1 = T OR LZONE2 = T
C OR ELSE IN CP2, CF2, ETC. IF LZONE3 = T
C
C ALL COEFFICIENTS ARE CALCULATED AND OUTPUT FOR BOTH
C CONSTANT QWALL AND CONSTANT TWALL (REGARDLESS OF WHETHER
C OR NOT THEY ARE VALID)
C
C PHI      = ANGLE FROM LEADING EDGE OF CYLINDER
C TWALL    = WALL TEMPERATURE
C QWALL    = WALL HEAT FLUX
C CPREF    = PRESSURE AT REFERENCE CELL
C
C CP1 (CP2 = LOCAL PRESSURE COEFFICIENT
C CF1      = LOCAL FRICTION COEFFICIENT
C CFZ1     = STREAM-WISE COMPONENT OF LOCAL FRICTION COEFFICIENT
C CDRP     = PRESSURE COMPONENT OF DRAG COEFFICIENT
C CDRF     = FRICTION COMPONENT OF DRAG COEFFICIENT
C NULM1    = LOCAL NUSSELT NUMBER BASED ON TF (CONST TWALL ONLY)
C NUM1     = LOCAL NUSSELT NUMBER BASED ON TM
C
LZONE1 = .FALSE.
LZONE2 = .FALSE.
LZONE3 = .FALSE.
C
IF (LHGRID) THEN
IF (LICONFIG) THEN
IF ((IZ.GE.IZ1).AND.(IZ.LE.IZ2)) LZONE3 = .TRUE.
ELSE
IF (IZ.LE.IZ1) LZONE1 = .TRUE.
IF ((IZ.GE.IZ2).AND.(IZ.LT.NZ)) LZONE2 = .TRUE.
END IF
ELSE IF (LIGRID) THEN
IF (LDCONFIG) THEN
LZONE1 = .TRUE.
LZONE2 = .TRUE.
ELSE
IF ((IZ.GE.IZ1).AND.(IZ.LE.IZ2)) LZONE3 = .TRUE.
END IF
END IF
C
IF (IZ.LT.NZ) THEN
IF (LZONE1) THEN
CALL VECTR(VEC,AVEC1(1,IZ),AVEC1(2,IZ),AVEC1(3,IZ))
COSPFI = DOT(VEC,ZVEC)
SINPHI = DOT(VEC,YVEC)
DO 1931 II = 1, NX
IF (.NOT.LCONSTT)
$TWALL = QWALL*YWALL1(IZ,II)/KFLUID + H1WALL1(IZ,II)/CFLUID
CP1(IZ,II) = 2.*PWALL1(IZ,II)/(RHO1*WB*WB) - EU
CP1(IZ,II) = 1. - CPREF + CP1(IZ,II)
WTERM1 = W1WALL1(IZ,II)*W1WALL1(IZ,II) +
$U1WALL1(IZ,II)*U1WALL1(IZ,II)
WTERM1 = SQRT(WTERM1)
CF1(IZ,II) = 2.*ENUL*WTERM1/(YWALL1(IZ,II)*WB*WB)
CFZ1(IZ,II) = CF1(IZ,II)*SINPHI

```



```
CDRP = CDRP - PWALL1(IZ,II)*COSPFI*AWALL1(IZ,II)
CDRF = CDRF + AWALL1(IZ,II)*SINPHI*RHO1*ENUL
$*WTERM1/(YWALL1(IZ,II))
TERM = DIAM*(TWALL - H1WALL1(IZ,II)/CFLUID)/YWALL1(IZ,II)
NUF1(IZ,II) = TERM/(TWALL - TFU)
NULM1(IZ,II) = TERM/DTLM
NUM1(IZ,II) = TERM/(TWALL - TBIN)
IF (STORE(NNUSS)) F(LONUSS + 1 + NY*(II-1)) = NUF1(IZ,II)
NUFBAR = NUFBAR + AWALL1(IZ,II)*NUF1(IZ,II)
NULMBAR = NULMBAR + AWALL1(IZ,II)*NULM1(IZ,II)
NUMBAR = NUMBAR + AWALL1(IZ,II)*NUM1(IZ,II)
ACYL = ACYL + AWALL1(IZ,II)
1931 CONTINUE
END IF
IF (LZONE2) THEN
CALL VECTR(VEC,AVEC2(1,IZ),AVEC2(2,IZ),AVEC2(3,IZ))
COSPFI = DOT(VEC,ZVEC)
SINPHI = DOT(VEC,YVEC)
DO 1932 II = 1, NX
IF (.NOT.LCONSTT)
$TWALL = QWALL*YWALL2(IZ,II)/KFLUID + H1WALL2(IZ,II)/CFLUID
CP2(IZ,II) = 2.*PWALL2(IZ,II)/(RHO1*WB*WB)
CP2(IZ,II) = 1. - CPREF + CP2(IZ,II)
WTERM2 = W1WALL2(IZ,II)*W1WALL2(IZ,II) +
$U1WALL2(IZ,II)*U1WALL2(IZ,II)
WTERM2 = SQRT(WTERM2)
CF2(IZ,II) = 2.*ENUL*WTERM2/(YWALL2(IZ,II)*WB*WB)
CFZ2(IZ,II) = CF2(IZ,II)*SINPHI
CDRP = CDRP - PWALL2(IZ,II)*COSPFI*AWALL2(IZ,II)
CDRF = CDRF + AWALL2(IZ,II)*SINPHI*RHO1*ENUL
$*WTERM2/(YWALL2(IZ,II))
TERM = DIAM*(TWALL - H1WALL2(IZ,II)/CFLUID)/YWALL2(IZ,II)
NUF2(IZ,II) = TERM/(TWALL - TMIDL)
NULM2(IZ,II) = TERM/DTLM
NUM2(IZ,II) = TERM/(TWALL - TBLAST)
IF (STORE(NNUSS).AND.LINLINE)
$F(LONUSS + 1 + NY*(II-1)) = NUF2(IZ,II)
IF (STORE(NNUSS).AND..NOT.LINLINE)
$F(LONUSS + NY + NY*(II-1)) = NUF2(IZ,II)
NUFBAR = NUFBAR + AWALL2(IZ,II)*NUF2(IZ,II)
NULMBAR = NULMBAR + AWALL2(IZ,II)*NULM2(IZ,II)
NUMBAR = NUMBAR + AWALL2(IZ,II)*NUM2(IZ,II)
ACYL = ACYL + AWALL2(IZ,II)
1932 CONTINUE
END IF
IF (LZONE3) THEN
CALL VECTR(VEC,AVEC1(1,IZ),AVEC1(2,IZ),AVEC1(3,IZ))
COSPFI = DOT(VEC,ZVEC)
SINPHI = DOT(VEC,YVEC)
DO 1933 II = 1, NX
IF (.NOT.LCONSTT)
$TWALL = QWALL*YWALL1(IZ,II)/KFLUID + H1WALL1(IZ,II)/CFLUID
CP1(IZ,II) = 2.*PWALL1(IZ,II)/(RHO1*WB*WB)
CP1(IZ,II) = 1. - CPREF + CP1(IZ,II)
WTERM1 = W1WALL1(IZ,II)*W1WALL1(IZ,II) +
$U1WALL1(IZ,II)*U1WALL1(IZ,II)
WTERM1 = SQRT(WTERM1)
```

```

CF1(IZ,II) = 2.*ENUL*WTERM1/(YWALL1(IZ,II)*WB*WB)
CFZ1(IZ,II) = CF1(IZ,II)*SINPHI
CDRP = CDRP - PWALL1(IZ,II)*COSPHI*AWALL1(IZ,II)
CDRF = CDRF + AWALL1(IZ,II)*SINPHI*RHO1*ENUL
$*WTERM1/(YWALL1(IZ,II))
TERM = DIAM*(TWALL - H1WALL1(IZ,II)/CFLUID)/YWALL1(IZ,II)
NUF1(IZ,II) = TERM/(TWALL - TBIN)
NULM1(IZ,II) = TERM/DTLM
NUM1(IZ,II) = TERM/(TWALL - TMIDL)
IF (STORE(NNUSS).AND.LINLINE)
$F(LONUSS + 1 + NY*(II-1)) = NUF2(IZ,II)
IF (STORE(NNUSS).AND..NOT.LINLINE)
$F(LONUSS + NY + NY*(II-1)) = NUF2(IZ,II)
NUFBAR = NUFBAR + AWALL1(IZ,II)*NUF1(IZ,II)
NULMBAR = NULMBAR + AWALL1(IZ,II)*NULM1(IZ,II)
NUMBAR = NUMBAR + AWALL1(IZ,II)*NUM1(IZ,II)
ACYL = ACYL + AWALL1(IZ,II)
1933 CONTINUE
END IF
END IF
END IF
C
RETURN
197 CONTINUE
C * ----- SECTION 7 ---- Finish of sweep.
c
IF (LINLINE) THEN
CALL DSTGV(W1WALL1,U1WALL1,NZ,NX)
CALL DSTGV(W1WALL2,U1WALL2,NZ,NX)
END IF
c
DO 1934 I = 1, NX
DO 1934 J = IYL1, IYL2
PVAL(J,I) = (1.-AMYRLX)*P1HALO(J,I)
1+ AMYRLX*P1DOWN(J,I)
991 FORMAT (I10,G13.4)
1934 CONTINUE
C
IDUM = INT(ISWEEP/TSTSWP)
RDUM = FLOAT(ISWEEP)/FLOAT(TSTSWP)
IF (RDUM.EQ.FLOAT(IDUM)) THEN
C
C
C CALCULATE RE, EU, NU.
C
RE= WB*DIAM/ENUL
EU = 2.0*PFRST/(RHO1*WB*WB)
IF (ABS(TBLAST-TWALL).GT.0.)
$LTITO = (TBIN-TWALL)/(TBLAST-TWALL)
IF(ABS(LTITO).GT.0) THEN
LTITO = LOG(LTITO)
IF(ABS(LTITO).GT.0) THEN
DTLM = (TBLAST - TBIN)/LTITO
NUBAR = RE*PRNDTL(H1)*(A-1.)*LTITO/PI
END IF
ELSE
NUBAR = 1.E-20

```

```
      LTITO = 1.E-20
      DTLM = 1.E-20
      END IF
C
C   OUTPUT DATA TO SCREEN AND RESULT FILE.
C
910 FORMAT (/,' ISWEEP = ',I4)
913 FORMAT (' RE = ',G11.4,
      $' PR   = ',G11.4)
916 FORMAT (' EU = ',G11.4,
      $' NUBAR   = ',G11.4,/)
953 FORMAT (' PFRST = ',G11.4,
      $' PLAST = ',G11.4,' TBO - TBI = ',G11.4)
952 FORMAT (' LOG(TW-TI)/(TW-TBLAST) = ',G11.4,
      $' DTLM   = ',G11.4)
954 FORMAT (' A*   = ',G11.4,' CMASS = ',G11.4,/)
955 FORMAT (' TFU = ',G11.4,
      $' TMIDL = ',G11.4)
956 FORMAT (I4,2X,3(G11.4,1X))
      WRITE(6,910) ISWEEP
      WRITE(6,913) RE, PRNDTL(H1)
      WRITE(6,916) EU, NUBAR
      WRITE(6,953) PFRST, PLAST, TBLAST-TBIN
      WRITE(6,952) LTITO, DTLM
      WRITE(6,954) ATOT,CMASS
      WRITE(6,955) TFU, TMIDL
      WRITE(14,910) ISWEEP
      WRITE(14,913) RE, PRNDTL(H1)
      WRITE(14,916) EU, NUBAR
      WRITE(14,953) PFRST, PLAST, TBLAST-TBIN
      WRITE(14,952) LTITO, DTLM
      WRITE(14,954) ATOT, CMASS
      WRITE(14,955) TFU, TMIDL
C
C   DUMP DATA
C
      IF (LSWEEP.GT.200) CALL DUMP
      END IF
      RETURN
198 CONTINUE
C * ----- SECTION 8 ---- Finish of time step.
C
C   AVERAGE COEFFICIENTS COMPUTED FROM LOCAL COEFFICIENTS
C
      NUFBAR = NUFBAR/ACYL
      NUMBAR = NUMBAR/ACYL
      NULMBAR = NULMBAR/ACYL
      CDRP = 4.*CDRP/(DIAM*RHO1*WB*WB)
      CDRF = 4.*CDRF/(DIAM*RHO1*WB*WB)
      CDR = CDRP + CDRF
c
C   SAVE COEFFICIENT DATA IN PLOT3D FORMAT
C
C   1. OPEN FILES
C
      LGRID = 95
      LCODATA = 96
```

```
LVDATA = 97
OPEN(LGRID,FILE='cogrid.fmt',FORM='FORMATTED',
$STATUS='UNKNOWN',ACCESS='SEQUENTIAL',IOSTAT=IOS)
IF (IOS.NE.0) THEN
  WRITE (14,*) ' ERROR OPENING cogrid.fmt'
  WRITE (6,*) ' ERROR OPENING cogrid.fmt'
ELSE
  WRITE (14,*) ' cogrid.fmt SUCCESSFULLY OPENED'
  WRITE (6,*) ' cogrid.fmt SUCCESSFULLY OPENED'
END IF
OPEN(LCODATA,FILE='codata.fmt',FORM='FORMATTED',
$STATUS='UNKNOWN',ACCESS='SEQUENTIAL',IOSTAT=IOS)
IF (IOS.NE.0) THEN
  WRITE (14,*) ' ERROR OPENING codata.fmt'
  WRITE (6,*) ' ERROR OPENING codata.fmt'
ELSE
  WRITE (14,*) ' codata.fmt SUCCESSFULLY OPENED'
  WRITE (6,*) ' codata.fmt SUCCESSFULLY OPENED'
END IF
OPEN(LVDATA,FILE='vdata.fmt',FORM='FORMATTED',
$STATUS='UNKNOWN',ACCESS='SEQUENTIAL',IOSTAT=IOS)
IF (IOS.NE.0) THEN
  WRITE (14,*) ' ERROR OPENING vdata.fmt'
  WRITE (6,*) ' ERROR OPENING vdata.fmt'
ELSE
  WRITE (14,*) ' vdata.fmt SUCCESSFULLY OPENED'
  WRITE (6,*) ' vdata.fmt SUCCESSFULLY OPENED'
END IF
C
C 2. WRITE HEADER
C
  NZPLOT3D = 2*IZ1
  WRITE (LGRID,*) NZPLOT3D, NX
  WRITE (LCODATA,*) NZPLOT3D, NX
  WRITE (LCODATA,*) 0, 0, RE, 0
  WRITE (LVDATA,*) NZPLOT3D, NX
  WRITE (LVDATA,*) 0, 0, RE, 0
C
C 3. CONVERT AND WRITE DATA
C
C X - GRID IS  $\text{PHI} \times 2 \times \text{PI} \times \text{DIAM} / 180$ 
C Y - GRID IS HEIGHT
c
c
  CALL SHUFFLE(PHI1,PHI2,QVALU,NZD,NXD,IZ1,IZ2)
  DO 1935 KK = 1, NZPLOT3D
    DO 1935 II = 1, NX
      QVALU(KK, II) =  $\text{PI} \times \text{DIAM} \times \text{QVALU}(\text{KK}, \text{II}) / 360.$ 
1935 CONTINUE
  WRITE (LGRID,*) ((QVALU(KK,II), KK=1, NZPLOT3D), II=1, NX),
$(DIST(KK,II), KK=1, NZPLOT3D), II=1, NX)
c
  DO 1936 LL = 1, 4
    IF (LL.EQ.1) THEN
      CALL SHUFFLE(CP1,CP2,QVALU,NZD,NXD,IZ1,IZ2)
    ELSE IF (LL.EQ.2) THEN
      CALL SHUFFLE(CF1,CF2,QVALU,NZD,NXD,IZ1,IZ2)
```

```
ELSE IF (LL.EQ.3) THEN
CALL SHUFFLE(NUF1,NUF2,QVALU,NZD,NXD,IZ1,IZ2)
END IF
DO 1936 KK = 1, NZPLOT3D
DO 1936 II = 1, NX
IF (LL.LE.2) THEN
Q(KK,II,LL) = QVALU(KK,II)
ELSE IF (LL.LE.3) THEN
Q(KK,II,LL) = QVALU(KK,II)/NUFBAR
ELSE
Q(KK,II,LL) = 0.
END IF
1936 CONTINUE
c
WRITE (LCODATA,*)
$(((Q(KK,II,LL), KK=1, NZPLOT3D), II=1, NX), LL = 1, 4)
c
DO 1937 LL = 1, 4
IF (LL.EQ.2) THEN
CALL SHUFFLE(W1WALL1,W1WALL2,QVALU,NZD,NXD,IZ1,IZ2)
ELSE IF (LL.EQ.3) THEN
CALL SHUFFLE(U1WALL1,U1WALL2,QVALU,NZD,NXD,IZ1,IZ2)
END IF
DO 1937 KK = 1, NZPLOT3D
DO 1937 II = 1, NX
IF (LL.LE.1) THEN
Q(KK,II,LL) = RHO1
ELSE IF (LL.LE.3) THEN
Q(KK,II,LL) = QVALU(KK,II)
ELSE
Q(KK,II,LL) = 0.
END IF
1937 CONTINUE
c
WRITE (LVDATA,*)
$(((Q(KK,II,LL), KK=1, NZPLOT3D), II=1, NX), LL = 1, 4)
c
c 4. Close files
c
CLOSE(LGRID,STATUS='KEEP')
CLOSE(LCODATA,STATUS='KEEP')
CLOSE(LVDATA,STATUS='KEEP')
RETURN

C*****
c
SUBROUTINE SWCHY(AUP,ADOWN,NY,NX,IYLAST,IXLAST)
C
REAL ADOWN(NY,NX), AUP(NY,NX)
INTEGER NY, NX, I, J, IYLAST, IXLAST
C
DO 10 J = 1, IYLAST
DO 10 I = 1, NX
AUP(IYLAST-J+1,I) = ADOWN(J,I)
10 CONTINUE
IF (IYLAST.LT.NY) THEN
DO 20 J = IYLAST+1, NY
```

```
      DO 20 I = 1, NX
      AUP(J,I) = AUP(IYLAST,I)
20  CONTINUE
      END IF
      RETURN
      END

C
      SUBROUTINE SETCX(Y,X,NY,NX,M,C,IYLAST)
c   Y = MX + C
      REAL Y(NY,NX), X(NY,NX)
      REAL M, C
      INTEGER NY, NX, I, J
C
      DO 10 J = 1, IYLAST
      DO 10 I = 1, NX
      Y(J,I) = M*X(J,I) + C
10  CONTINUE
      DO 20 J = IYLAST+1,NY
      DO 20 I = 1, NX
      Y(J,I) = Y(IYLAST,I)
20  CONTINUE
      RETURN
      END

C
      SUBROUTINE VECTR(V,P,Q,R)
      REAL V(3)
      V(1) = P
      V(2) = Q
      V(3) = R
      RETURN
      END

c
      SUBROUTINE NORM2(PT1,PT2,PT3,PT4,VEC)
      REAL PT1(3),PT2(3),PT3(3),PT4(3),VEC(3)
      REAL V1(3), V2(3)
      DO 10 I = 1, 3
      V1(I) = PT2(I) - PT1(I)
      V2(I) = PT4(I) - PT1(I)
10  CONTINUE
      CALL CROSS(V1,V2,VEC)
      CALL NORM(VEC)
      RETURN
      END

c
      SUBROUTINE SHUFFLE(A1,A2,A,NZD,NXD,IZ1,IZ2)
      IMPLICIT NONE
      REAL A1(NZD,NXD), A2(NZD,NXD), A(2*NZD,NXD)
      INTEGER NZD, NXD, IZ1, IZ2, I, K
      DO 10 I = 1, NXD
      DO 10 K = 1, IZ1
      A(K,I) = A2(IZ2+K-1,I)
      A(IZ1+K,I) = A1(K,I)
10  CONTINUE
      RETURN
      END

c
      SUBROUTINE DSTGV(U,V,NX,NY)
```

```
REAL U(NX,NY), V(NX,NY), RJUNK
c
DO 15 J = 1, NY
  RJUNK = U(NX-1,J)
  DO 10 I = NX-1, 2, -1
    U(I,J) = (U(I,J)+U(I-1,J))/2.
10 CONTINUE
  U(1,J) = (U(1,J)+RJUNK)/2.
15 CONTINUE
  DO 20 J = NY-1, 2, -1
    DO 20 I = 1, NX-1
      V(I,J) = (V(I,J)+V(I,J-1))/2.
20 CONTINUE
  RETURN
END
```

A3.7 Offset fin heat exchanger

Q1 FILE

TALK=T;RUN(1, 1);VDU=DEC GKS

GROUP 1. Run title and other preliminaries
TEXT(3-D thick plate-fin hx.)
REAL(HDIAM, W0, TWALL, TBUP, RE)
REAL (SMALL, AMYCO)
REAL(B,LENG,PF,DELTA, XX, YY, A, AC, AF)
REAL (VFIN,VFLUID, PRFIN, PFLUID, KFIN, KFLUID,CFLUID)
INTEGER (NXFIN, NYFIN)
** Core 105
B = .0750;PF = .0415; LENG = 1./9.; DELTA = .004
HDIAM = .003966*12.; AF = .857
TWALL = 1.;TBUP = 0.
Metric Conversion:
B = B*.0254;PF = PF*.0254;LENG = LENG*.0254
DELTA = DELTA*.0254;HDIAM = HDIAM*.0254

GROUP 2. Transience; time-step specification
STEADY=T

GROUP 3. X-direction grid specification
CARTES=T; NX = 20
NX = 28
XULAST = B
NXFIN = 4
XFRAC(1) = -1.*(NXFIN+1); XFRAC(2) = DELTA/(B*NXFIN)
XFRAC(3) = (NX-2*NXFIN-2);
XX = (1 - 2*(NXFIN+1)*XFRAC(2))
XFRAC(4) = XX/(NX - 2*(NXFIN+1))
XFRAC(5) = NXFIN+1; XFRAC(6) = XFRAC(2)

GROUP 4. Y-direction grid specification
NY = 14
YVLAST = PF/2.

```

NYFIN = 2
YFRAC(1) = -1.*(NYFIN+1); YFRAC(2) = DELTA/(PF*NYFIN)
YFRAC(3) = (NY-2*(NYFIN+1));
YY = (1 - 2*(NYFIN+1)*YFRAC(2))
YFRAC(4) = YY/(NY - 2*(NYFIN+1))
YFRAC(5) = NYFIN+1; YFRAC(6) = YFRAC(2)
  GROUP 5. Z-direction grid specification
NZ = 40
ZWLAST = LENG
GRDPWR(Z,-1*NZ,ZWLAST,0.75)
NZ = NZ + 1
ZFRAC(NZ) = 1 + ZFRAC(1)
  GROUP 6. Body-fitted coordinates or grid distortion
  GROUP 7. Variables stored, solved & named
SOLVE(P1)
SOLUTN(U1,Y,Y,N,N,N,Y)
SOLUTN(V1,Y,Y,N,N,N,Y)
SOLUTN(W1,Y,Y,N,N,N,Y)
SOLUTN(H1,Y,Y,N,N,N,Y)
STORE(VISL)
  ** Store local heat transfer coefficients:
STORE(NULX,NULY,NULZ)
  GROUP 8. Terms (in differential equations) & devices
  ** Suppress laminar dissipation
TERMS(H1,N,P,P,P,P,P)
  GROUP 9. Properties of the medium (or media)
RHO1 = 0.9944;RE = 1.E+02
VFLUID = 2.0821E-05/RHO1;ENUL=GRND;PRNDTL(H1) = GRND
W0=VFLUID*RE/HDIAM
CFLUID = 1.016E+03;KFLUID = 0.03035
KFIN = 630.0;VFIN = 1.E+06
PFLUID = RHO1*CFLUID*VFLUID/KFLUID
PRFIN = RHO1*CFLUID*VFIN/KFIN
  GROUP 10. Inter-phase-transfer processes and properties
  GROUP 11. Initialization of variable or porosity fields
FIINIT(NULX)=0.0;FIINIT(NULY)=0.0;FIINIT(NULZ)=0.0;
FIINIT(P1) = 1.E-10;FIINIT(V1) = 1.E-10;FIINIT(W1) = W0
FIINIT(H1) = CFLUID*(TBUP+TWALL)/2.
INIADD = F
PATCH(FINH5,INIVAL,1,NX,1,NYFIN,1,(NZ-1)/2,1,LSTEP)
PATCH(FINH6,INIVAL,1,NX,NY-NYFIN,NY,(NZ-1)/2+1,NZ,1,LSTEP)
INIT(FINH5,W1,0.,0.)
INIT(FINH6,W1,0.,0.)
INIT(FINH5,H1,0.,CFLUID*TWALL)
INIT(FINH6,H1,0.,CFLUID*TWALL)
  GROUP 12. Convection and diffusion adjustments
  GROUP 13. Boundary conditions and special sources

  ** INLET BOUNDARY CONDITIONS

PATCH(UPSTRM,LOW,1,NX,1,NY,1,1,1,LSTEP)
COVAL(UPSTRM,P1,FIXFLU,GRND)
COVAL(UPSTRM,U1,ONLYMS,GRND)
COVAL(UPSTRM,V1,ONLYMS,GRND)
COVAL(UPSTRM,W1,ONLYMS,GRND)
COVAL(UPSTRM,H1,ONLYMS,GRND)

```


**** OUTLET BOUNDARY CONDITIONS**

PATCH(DWNSTRM,LOW,1,NX,1,NY,NZ,NZ,1,LSTEP)
COVAL(DWNSTRM,P1,GRND,GRND)
COVAL(DWNSTRM,H1,FIXVAL,GRND)

**** HORIZONTAL FIN BOUNDARY CONDITIONS**

PATCH(FINH5,SWALL,1,NX,1,1,1,(NZ-1)/2,1,LSTEP)
PATCH(FINH6,NWALL,1,NX,NY,NY,(NZ-1)/2+1,NZ,1,LSTEP)
COVAL(FINH5,W1,1.,0.0)
COVAL(FINH6,W1,1.,0.0)

**** PLATE (& VERTICAL FIN) BOUNDARY CONDITIONS**

PATCH(PLATEE,EWALL,NX,NX,NYFIN+1,NY,1,(NZ-1)/2,1,LSTEP)
PATCH(PLATEW,WWALL,1,1,1,NY-NYFIN,(NZ-1)/2+1,NZ-1,1,LSTEP)
PATCH(FINEDGE,EWALL,NX,NX,1,NYFIN,1,(NZ-1)/2,1,LSTEP)
PATCH(FINEDGW,WWALL,1,1,1,NY-NYFIN+1,NY,(NZ-1)/2+1,NZ-1,1,LSTEP)
PATCH(FINW,WWALL,1,1,1,NY,1,(NZ-1)/2,1,LSTEP)
PATCH(FINE,EWALL,NX,NX,1,NY,(NZ-1)/2+1,NZ-1,1,LSTEP)

COVAL(PLATEE,V1,1.0,0.0)
COVAL(PLATEE,W1,1.0,0.0)
COVAL(PLATEE,H1,1.0/PFLUID,CFLUID*TWALL)
COVAL(PLATEW,V1,1.0,0.0)
COVAL(PLATEW,W1,1.0,0.0)
COVAL(PLATEW,H1,1.0/PFLUID,CFLUID*TWALL)
COVAL(FINEDGE,V1,1.0,0.0)
COVAL(FINEDGE,W1,1.0,0.0)
COVAL(FINEDGE,H1,1.0/PRFIN,CFLUID*TWALL)
COVAL(FINEDGW,V1,1.0,0.0)
COVAL(FINEDGW,W1,1.0,0.0)
COVAL(FINEDGW,H1,1.0/PRFIN,CFLUID*TWALL)
COVAL(FINW,V1,1.0,0.0)
COVAL(FINW,W1,1.0,0.0)
COVAL(FINW,H1,1.0/PRFIN,CFLUID*TWALL)
COVAL(FINE,V1,1.0,0.0)
COVAL(FINE,W1,1.0,0.0)
COVAL(FINE,H1,FIXVAL,CFLUID*TWALL)

GROUP 14. Downstream pressure for PARAB=.TRUE.

GROUP 15. Termination of sweeps

LSWEEP = 2000

ISWC1 = LSWEEP/4

RESREF(P1) = 1.E-12

RESREF(U1) = 1.E-12

RESREF(V1) = 1.E-12

RESREF(W1) = 1.E-12

RESREF(H1) = 1.E-12

GROUP 16. Termination of iterations

GROUP 17. Under-relaxation devices

RELAX(P1,LINRLX,.5)

RELAX(U1,FALSDT,.01)

RELAX(V1,FALSDT,.01)

RELAX(W1,FALSDT,.01)

RELAX(H1,FALSDT,.01)

GROUP 18. Limits on variables or increments to them

VARMIN(P1) = -2.*RHO1*W0*W0*RE; VARMAX(P1) = 2.*RHO1*W0*W0*RE

VARMIN(U1) = -2.*W0; VARMAX(U1) = 2.*W0

VARMIN(V1) = -2.*W0; VARMAX(V1) = 2.*W0

VARMIN(W1) = -2.*W0; VARMAX(W1) = 5.*W0

GROUP 19. Data communicated by satellite to GROUND

USEGRX=F

IG(1) = NXFIN

IG(2) = NYFIN

RG(1) = TBUP

RG(2) = TWALL

RG(3) = W0

RG(4) = KFLUID

RG(5) = CFLUID

RG(6) = KFIN

RG(7) = VFLUID

AMYCO = 1.E+06

RG(8) = AMYCO

GROUP 20. Preliminary print-out

GROUP 21. Print-out of variables

ABSIZ=0.50

ORSIZ=0.40

OUTPUT(P1,N,N,N,N,Y,Y)

OUTPUT(V1,N,N,N,N,Y,Y)

OUTPUT(W1,N,N,N,N,Y,Y)

OUTPUT(H1,Y,Y,Y,N,Y,Y)

OUTPUT(VISL,N,N,N,N,N,N)

OUTPUT(NULX,N,N,N,N,N,N)

OUTPUT(NULY,N,N,N,N,N,N)

OUTPUT(NULZ,N,N,N,N,N,N)

GROUP 22. Spot-value print-out

IXMON = NXFIN + 1; IYMON = NYFIN + 1

TSTSWP = LSWEET/10

GROUP 23. Field print-out and plot control

GROUP 24. Dumps for restarts

RESTRT(ALL)

ECHO=F

INIFLD=F

STOP

GROUND FILE

C 2 User dimensions own arrays here, for example:

C DIMENSION GUH(10,10),GUC(10,10),GUX(10,10),GUZ(10)

C These arrays must have dimension (NY,NX)

C

REAL AAH(14,28), AAN(14,28), LY(14,28)

REAL PIFRST(14,28), PIDOWN(14,28)

REAL U1LAST(14,28), V1LAST(14,28)

REAL W1LAST(14,28), H1LAST(14,28)

REAL VALU(14,28)

C

C These arrays must have dimension NZ

C

REAL TB(41)

REAL SP(41), W1WALL(41), LZ(41), YWALL(41), PWALL(41)

REAL NUSLAB(41), AHSLAB(41), WBSLAB(41), TBSLAB(41), QSLAB(41)

C

REAL B, PF, L, DELTA, HDIAM

REAL YVLAST, ATOT, ASUM

REAL KFIN, KFLUID, KEFF, VFIN, VFLUID, PRFIN, PRFLUID

REAL W0, WB, PLBAR, PDBAR, TBUP, TWALL, HFLUX

REAL RE, FBAR, NUBAR, STBAR, JBAR, CFLUID, ANTU,ALMTD, NUSUM

REAL PP, PPLUS, ST

REAL CMASS, C1HEAT, C2HEAT

REAL A, M, D

REAL AC, AHT, AFIN, UOVER, HLOC

C

INTEGER I, J, KK

INTEGER LOVIST, LOCO, LODYV2D, LOVISL, LOPR

INTEGER LOES7, LOES8, LOES9, LOES10

INTEGER LOLD7, LOW1

INTEGER NXFIN, NYFIN

C

C 3 User places his data statements here:

C

DATA CMASS, C1HEAT, C2HEAT/ 1.0, 1.0, 0.0/

DATA KFIN, KFLUID, VFIN, VFLUID/630.,

1.03035E-03,1.E+06, 2.094E-05/

DATA B, PF, L, DELTA, HDIAM/.075, .0415, .11111111, .004,

1.003966/

C*****

C

C--- GROUP 1. Run title and other preliminaries

C

1 GO TO (1001,1002),ISC

1001 CONTINUE

CALL MAKE(EASP1)

CALL MAKE(EASP6)

CALL MAKE(EASP7)

CALL MAKE(EASP8)

CALL MAKE(EASP9)

CALL MAKE(EASP10)

CALL MAKE(YG2D)

CALL MAKE(DXG2D)

CALL MAKE(DXU2D)

CALL MAKE(DYG2D)

```
CALL MAKE(DYV2D)
CALL MAKE(AEAST)
CALL MAKE(AWEST)
CALL MAKE(AHIGH)
C
C User may here change message transmitted to the VDU screen or
C batch-run log file.
IF(IGR.EQ.1.AND.ISC.EQ.1) THEN
  CALL WRYT40('GROUND file is GROUND.FTN of: 200390 ')
  CALL WRYT40('PHOENICS version number is : 1.5.3 ')
ENDIF
C
  RETURN
1002 CONTINUE
  RETURN
C*****
C
C--- GROUP 9. Properties of the medium (or media)
C
C The sections in this group are arranged sequentially in their
C order of calling from EARTH. Thus, as can be seen from below,
C the temperature sections (10 and 11) precede the density
C sections (1 and 3); so, density formulae can refer to
C temperature stores already set.
  9 GO TO (91,92,93,94,95,96,97,98,99,900,901,902,903,904,905),ISC
C*****
  900 CONTINUE
C * ----- SECTION 10 -----
C For TMP1.LE.GRND----- phase-1 temperature Index TEMP1
  RETURN
  901 CONTINUE
C * ----- SECTION 11 -----
C For TMP2.LE.GRND----- phase-2 temperature Index TEMP2
  RETURN
  902 CONTINUE
C * ----- SECTION 12 -----
C For EL1.LE.GRND----- phase-1 length scale Index LEN1
  RETURN
  903 CONTINUE
C * ----- SECTION 13 -----
C For EL2.LE.GRND----- phase-2 length scale Index LEN2
  RETURN
  904 CONTINUE
C * ----- SECTION 14 -----
C For SOLVE(TEMP1)----- phase-1 specic heat
  RETURN
  905 CONTINUE
C * ----- SECTION 15 -----
C For SOLVE(TEMP2)----- phase-2 specic heat
  RETURN
  91 CONTINUE
C * ----- SECTION 1 -----
C For RHO1.LE.GRND--- density for phase 1 Index DEN1
  RETURN
  92 CONTINUE
C * ----- SECTION 2 -----
C For DRH1DP.LE.GRND--- D(LN(DEN))/DP for phase 1
```

```

C                               Index D1DP
  RETURN
93 CONTINUE
C * ----- SECTION 3 -----
C For RHO2.LE.GRND--- density for phase 2    Index DEN2
  RETURN
94 CONTINUE
C * ----- SECTION 4 -----
C For DRH2DP.LE.GRND--- D(LN(DEN))/DP for phase 2
C                               Index D2DP
  RETURN
95 CONTINUE
C * ----- SECTION 5 -----
C For ENUT.LE.GRND--- reference turbulent kinematic viscosity
C                               Index VIST
  RETURN
96 CONTINUE
C * ----- SECTION 6 -----
C For ENUL.LE.GRND--- reference laminar kinematic viscosity
C                               Index VISL
C
C
C
  L0VISL = L0F(AUX(VISL))
    IF (IZ.LE.(NZ-1)/2) THEN
      DO 961 J = 1, NY
        DO 961 I = 1, NXFIN
          F(L0VISL + J + NY*(I - 1)) = VFIN
961 CONTINUE
        DO 962 J = 1, NYFIN
          DO 962 I = 1, NX
            F(L0VISL + J + NY*(I - 1)) = VFIN
962 CONTINUE
        DO 963 J = NYFIN+1, NY
          DO 963 I = NXFIN+1, NX
            F(L0VISL + J + NY*(I - 1)) = VFLUID
963 CONTINUE
        ELSE
          DO 964 J = 1, NY - NYFIN
            DO 964 I = 1, NX - NXFIN
              F(L0VISL + J + NY*(I - 1)) = VFLUID
964 CONTINUE
          DO 965 J = NY - NYFIN + 1, NY
            DO 965 I = 1, NX
              F(L0VISL + J + NY*(I - 1)) = VFIN
965 CONTINUE
          DO 966 J = 1, NY
            DO 966 I = NX - NXFIN + 1, NX
              F(L0VISL + J + NY*(I - 1)) = VFIN
966 CONTINUE
        END IF
      RETURN
97 CONTINUE
C * ----- SECTION 7 -----
C For PRNDTL( ) .LE.GRND--- laminar PRANDTL nos., or diffusivity
C                               Index LAMPR
C
C
C

```

```

C  For PRNDTL( ).LE.GRND--- laminar PRANDTL nos., or diffusivity.
  LOPRL = L0F(LAMPR)
    IF (IZ.LE.(NZ-1)/2) THEN
      DO 971 J = 1, NY
        DO 971 I = 1, NXFIN
          F(LOPRL + J + NY*(I - 1)) = PRFIN
971  CONTINUE
        DO 972 J = 1, NYFIN
          DO 972 I = 1, NX
            F(LOPRL + J + NY*(I - 1)) = PRFIN
972  CONTINUE
        DO 973 J = NYFIN+1, NY
          DO 973 I = NXFIN+1, NX
            F(LOPRL + J + NY*(I - 1)) = PRFLUID
973  CONTINUE
        ELSE
          DO 974 J = 1, NY - NYFIN
            DO 974 I = 1, NX - NXFIN
              F(LOPRL + J + NY*(I - 1)) = PRFLUID
974  CONTINUE
          DO 975 J = NY - NYFIN + 1, NY
            DO 975 I = 1, NX
              F(LOPRL + J + NY*(I - 1)) = PRFIN
975  CONTINUE
          DO 976 J = 1, NY
            DO 976 I = NX - NXFIN + 1, NX
              F(LOPRL + J + NY*(I - 1)) = PRFIN
976  CONTINUE
        END IF
      RETURN
98  CONTINUE
C  * ----- SECTION 8 -----
C  For PHINT( ).LE.GRND--- interface value of first phase
C                                Index FII1
      RETURN
99  CONTINUE
C  * ----- SECTION 9 -----
C  For PHINT( ).LE.GRND--- interface value of second phase
C                                Index FII2
      RETURN
C*****
C
C--- GROUP 13. Boundary conditions and special sources
C                                Index for Coefficient - CO
C                                Index for Value      - VAL
13  CONTINUE
  GO TO (130,131,132,133,134,135,136,137,138,139,1310,
    11311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321),ISC
130  CONTINUE
C----- SECTION 1 ----- coefficient = GRND
  LOCO = L0F(CO)
  DO 1301 J = IYF, IYL
    DO 1301 I = IXF, IXL
      IF (W1LAST(J,I).LT.1.E-06) THEN
        F(LOCO + J + NY*(I-1)) = AMYCO/1.E-06
      ELSE
        F(LOCO + J + NY*(I-1)) = AMYCO/W1LAST(J,I)

```

```
END IF
1301 CONTINUE
RETURN
131 CONTINUE
C----- SECTION 2 ----- coefficient = GRND1
RETURN
132 CONTINUE
C----- SECTION 3 ----- coefficient = GRND2
RETURN
133 CONTINUE
C----- SECTION 4 ----- coefficient = GRND3
RETURN
134 CONTINUE
C----- SECTION 5 ----- coefficient = GRND4
RETURN
135 CONTINUE
C----- SECTION 6 ----- coefficient = GRND5
RETURN
136 CONTINUE
C----- SECTION 7 ----- coefficient = GRND6
RETURN
137 CONTINUE
C----- SECTION 8 ----- coefficient = GRND7
RETURN
138 CONTINUE
C----- SECTION 9 ----- coefficient = GRND8
RETURN
139 CONTINUE
C----- SECTION 10 ----- coefficient = GRND9
RETURN
1310 CONTINUE
C----- SECTION 11 ----- coefficient = GRND10
RETURN
1311 CONTINUE
C----- SECTION 12 ----- value = GRND
C
C Inlet boundary conditions.
C
      IF (NPATCH.EQ.'UPSTRM') THEN
        IF (INDVAR.EQ.P1) THEN
          CALL SWITCH(W1LAST,VALU,NY,NX,NY,NX)
          CALL SETYX(VAL,VALU,NY,NX)
          CALL FN25(VAL,RHO1*CMASS)
          CALL FN22(VAL,1.E-10)
        ELSE IF (INDVAR.EQ.U1) THEN
          CALL SWITCH(U1LAST,VALU,NY,NX,NY,(NX-1))
          CALL SETYX(VAL,VALU,NY,NX)
          CALL FN25(VAL,-1.)
        ELSE IF (INDVAR.EQ.V1) THEN
          CALL SWITCH(V1LAST,VALU,NY,NX,(NY-1),NX)
          CALL SETYX(VAL,VALU,NY,NX)
          CALL FN25(VAL,-1.)
        ELSE IF (INDVAR.EQ.W1) THEN
          CALL SWITCH(W1LAST,VALU,NY,NX,NY,NX)
          CALL SETYX(VAL,VALU,NY,NX)
          CALL FN22(VAL,1.E-10)
        ELSE IF (INDVAR.EQ.H1) THEN
```

```
CALL SWITCH(H1LAST,VALU,NY,NX,NY,NX)
CALL SETYX(VAL,VALU,NY,NX)
CALL FN25(VAL,C1HEAT)
CALL FN33(VAL,C2HEAT)
  END IF
C
C  Outlet boundary conditions.
C
      ELSE IF (NPATCH.EQ.'DWNSTRM') THEN
      LOVAL = LOF(VAL)
      DO 1302 J = IYF, IYL
      DO 1302 I = IXF, IXL
        IF (INDVAR.EQ.P1) THEN
          F(LOVAL + J + NY*(I-1)) = P1DOWN(J,I)
        ELSE IF(INDVAR.EQ.H1) THEN
          F(LOVAL + J + NY*(I-1)) = H1LAST(J,I)
        END IF
1302  CONTINUE
C
      END IF
      RETURN
1312 CONTINUE
C----- SECTION 13 ----- value = GRND1
      RETURN
1313 CONTINUE
C----- SECTION 14 ----- value = GRND2
      RETURN
1314 CONTINUE
C----- SECTION 15 ----- value = GRND3
      RETURN
1315 CONTINUE
C----- SECTION 16 ----- value = GRND4
      RETURN
1316 CONTINUE
C----- SECTION 17 ----- value = GRND5
      RETURN
1317 CONTINUE
C----- SECTION 18 ----- value = GRND6
      RETURN
1318 CONTINUE
C----- SECTION 19 ----- value = GRND7
      RETURN
1319 CONTINUE
C----- SECTION 20 ----- value = GRND8
      RETURN
1320 CONTINUE
C----- SECTION 21 ----- value = GRND9
      RETURN
1321 CONTINUE
C----- SECTION 22 ----- value = GRND10
      RETURN
C*****
C
C--- GROUP 19. Special calls to GROUND from EARTH
C
      19 GO TO (191,192,193,194,195,196,197,198),ISC
191 CONTINUE
```



```
C * ----- SECTION 1 ---- Start of time step.
C
  B = B*.0254
  PF = PF*.0254
  L = L*.0254
  DELTA = DELTA*.0254
  HDIAM = HDIAM*.0254*12.
  AC = (B - DELTA)*(PF - DELTA)
  AHT = 2.*L*(B-DELTA) + 2.*L*(PF-DELTA) + 2.*B*DELTA
  1 + (PF-DELTA)*DELTA
  AFIN = 2.*L*B + 2.*B*DELTA + 2.*PF*DELTA
  ALEFF = .5*B - DELTA
C
  NXFIN = IG(1)
  NYFIN = IG(2)
  TBUP = RG(1)
  TWALL = RG(2)
  W0 = RG(3)
  KFLUID = RG(4)
  CFLUID = RG(5)
  KFIN = RG(6)
  VFLUID = RG(7)
  AMYCO = RG(8)
  PRFLUID = RHO1*CFLUID*VFLUID/KFLUID
  PRFIN = RHO1*CFLUID*VFIN/KFIN
C
C   Get geometry values.
  CALL GETZ(DZGNZ,LZ,NZ)
C
C
C   Initialize variables
C
  CMASS = 1.
  WB = W0
  DO 911 J = 1, NY
  DO 911 I = 1, NX
    LY(J,I) = .1
    P1FRST(J,I) = 0.
    P1DOWN(J,I) = 0.
    U1LAST(J,I) = 0.
    V1LAST(J,I) = 0.
    IF ((J.LE.NY-NYFIN).AND.(I.LE.NX-NXFIN)) THEN
      W1LAST(J,I) = W0
    ELSE
      W1LAST(J,I) = 0.
    END IF
    H1LAST(J,I) = CFLUID*(TBUP + TWALL)/2.
  911 CONTINUE
    DO 1912 KK = 1, NZ
      TB(KK) = (TBUP + TWALL)/2.
  1912 CONTINUE
    CALL FN1(EASP7,1.)
    CALL FN1(EASP8,1.)
    CALL FN1(EASP9,1.)
    CALL FN1(EASP10,1.)
    ANUMIN = 0.0
    ANUMAX = 0.0
```

```

C
  RETURN
192 CONTINUE
C * ----- SECTION 2 ---- Start of sweep.
  RETURN
193 CONTINUE
C * ----- SECTION 3 ---- Start of iz slab.
C ----- This section executed only for ISWEEP = 1 -----
C
C
C  Get area of cell interfaces at exit/entrance to domain.
C
  IF (ISWEEP.EQ.1) THEN
    IF (IZ.EQ.NZ-1) THEN
      CALL GETYX(AHIGH,AAH,NY,NX)
      CALL GETYX(DYG2D,LY,NY,NX)
      ATOT = 0.0
      DO 912 J = 1, NY - NYFIN
        DO 912 I = 1, NX - NXFIN
          ATOT=ATOT+AAH(J,I)
612 CONTINUE
        END IF
C
C
C  Get area of north cell interfaces at IZ = NZ
  IF (IZ.EQ.(NZ-1)) THEN
    CALL GETYX(ANORTH,AAN,NY,NX)
    END IF
C
  END IF
  RETURN
194 CONTINUE
C * ----- SECTION 4 ---- Start of iteration.
  RETURN
195 CONTINUE
C * ----- SECTION 5 ---- Finish of iteration.
  RETURN
196 CONTINUE
C * ----- SECTION 6 ---- Finish of iz slab.
C
C ----- This section executed for all values of ISWEEP -----
C
C ----- This sub-section executed for IZ = 1 -----
C
C  Get field variables at first node.
C
  IF (IZ.EQ.1) THEN
    CALL GETYX(P1,P1FRST,NY,NX)
C
C  Calculate downstream pressures.
C
  DO 1915 J = 1, NY
    DO 1915 I = 1, NX
      VALU(J,I) = P1FRST(J,I) - P1FRST(NYFIN+1,NXFIN+1)
1915 CONTINUE
    CALL SWITCH(VALU,P1DOWN,NY,NX,NY,NX)
    END IF

```

```
C
C ----- This sub-section executed for IZ = NZ - 1 -----
C
C   Get exit values of w.
C   and values of p, u, v, h at "last" node.
C
C       IF (IZ.EQ.(NZ-1)) THEN
C       CALL GETYX(U1,U1LAST,NY,NX)
C       CALL GETYX(V1,V1LAST,NY,NX)
C       CALL GETYX(W1,W1LAST,NY,NX)
C       CALL GETYX(H1,H1LAST,NY,NX)
C
C   Calculate bulk exit velocity, and scale factor for mass flux.
C
C       WB = 0.0
C       DO 1902 J = 1, NY - NYFIN
C       DO 1902 I = 1, NX - NXFIN
C       WB = WB + AAH(J,I)*W1LAST(J,I)
1902 CONTINUE
C       WB = WB/ATOT
C       CMASS = W0/WB
C
C   Calculate bulk temperature and scale factors for enthalpy.
C
C       TB(NZ-1) = 0.0
C       DO 1905 J = 1, NY - NYFIN
C       DO 1905 I = 1, NX - NXFIN
C       TB(NZ-1) = TB(NZ-1) + AAH(J,I)*W1LAST(J,I)*H1LAST(J,I)
1905 CONTINUE
C       TB(NZ-1) = TB(NZ-1)/(CFLUID*ATOT*WB)
C       C1HEAT = (TBUP-TWALL)/(TB(NZ-1)-TWALL)
C       C2HEAT = CFLUID*TWALL*(1. - C1HEAT)
C
C       END IF
C
C ----- This section executed only for ISWEEP = LSWEEP -----
C       for all values of IZ
C
C       IF((ISWEEP.EQ.LSWEEP).OR.(ENUFWSW.EQ..TRUE.)) THEN
C
C ----- This sub-section computes local Nusselt number-----
C
C       INUX = INAME('NULX')
C       INUY = INAME('NULY')
C       INUZ = INAME('NULZ')
C       L0HX = L0F(INUX)
C       L0HY = L0F(INUY)
C       L0HZ = L0F(INUZ)
C       L0H1 = L0F(H1)
C       L0LOH1 = L0F(LOW(H1))
C       L0HIH1 = L0F(HIGH(H1))
C       L0DX = L0F(DXG2D)
C       L0DY = L0F(DYG2D)
C       L0AE = L0F(AEAST)
C       L0AN = L0F(ANORTH)
C       L0AH = L0F(AHIGH)
```

```

KEFF = 1./(0.5*(1./KFIN + 1./KFLUID))
C
    IF (IZ.LT.(NZ-1)/2) THEN
        IXF = IXFIN + 1
        IYF = IYFIN + 1
        IXL = NX
        IYL = NY
        ELSE IF (IZ.EQ.(NZ-1)/2) THEN
            IXF = IXFIN + 1
            IYF = IYFIN + 1
            IXL = NX - NXFIN
            IYL = NY - NYFIN
        ELSE IF (IZ.LT.NZ) THEN
            IXF = 1
            IYF = 1
            IXL = NX - NXFIN
            IYL = NY - NYFIN
        END IF
C
CALL FN1(EASP1,1.)
CALL FN20(WBSLAB(IZ),AHIGH,W1,EASP1)
CALL FN20(TBSLAB(IZ),AHIGH,H1,W1)
TBSLAB(IZ) = TBSLAB(IZ)/(CFLUID*WBSLAB(IZ))
WBSLAB(IZ) = WBSLAB(IZ)/ATOT
DELT = ABS(TWALL - TBSLAB(IZ))
IXF = 1
IYF = 1
IXL = NX
IYL = NY
C
C   Compute local Nusselt numbers:
C
C   NULX, NULY, NULY set below to local heat tr. coeff:
C    $h = k(dT/dn)/(T_{wall} - T_{bulk})$ 
C
NUSLAB(IZ) = 0.0
AHSLAB(IZ) = 0.0
C   First half of element,
C
    IF (IZ.LE.(NZ-1)/2) THEN
        DO 310 J = NYFIN+1, NY
C   West fin coefficient.
        TFIN = F(L0H1 + J + NY*(NXFIN - 1))/CFLUID
        TFLUID = F(L0H1 + J + NY*NXFIN)/CFLUID
        DN = F(L0DX + J + NY*(NXFIN-1))
        DA = F(L0AE + J + NY*(NXFIN-1))
        HLOC = KEFF*ABS((TFIN - TFLUID)/(DN*DELT))
        IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
        F(L0HX + J + NY*NXFIN) = HLOC
        NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
        AHSLAB(IZ) = AHSLAB(IZ) + DA
C   IF (J.GT.0) WRITE (6,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
C   IF (J.GT.0) WRITE (14,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
310    CONTINUE
        DO 320 I = NXFIN+1, NX
C   South fin coefficient.
        TFIN = F(L0H1 + NYFIN + NY*(I - 1))/CFLUID

```

```

TFLUID = F(L0H1 + NYFIN + 1 + NY*(I - 1))/CFLUID
DN = F(L0DY + NYFIN + NY*(I - 1))
DA = F(L0AN + NYFIN + NY*(I - 1))
HLOC = KEFF*ABS((TFIN - TFLUID)/(DN*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HY + NYFIN + 1 + NY*(I - 1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
C   IF (I.GT.0) WRITE (6,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
C   IF (I.GT.0) WRITE (14,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
320  CONTINUE
DO 330 J = NYFIN+1, NY
C   East plate coefficient.
TFLUID = F(L0H1 + J + NY*(NX-1))/CFLUID
DN = 0.5*F(L0DX + J + NY*(NX - 2))
DA = F(L0AE + J + NY*(NX - 2))
HLOC = KFLUID*ABS((TWALL - TFLUID)/(DN*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HX + J + NY*(NX - 1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
C   IF (J.GT.0) WRITE (6,*) TWALL, TFLUID, DN, DA, ANUMAX, HLOC
C   IF (J.GT.0) WRITE (14,*) TWALL, TFLUID, DN, DA, ANUMAX, HLOC
330  CONTINUE
C
C   Second half of element,
C
      ELSE IF (IZ.LE.(NZ-1)) THEN
DO 340 J = 1, NY - NYFIN
C   East fin coefficient.
TFIN = F(L0H1 + J + NY*(NX - NXFIN))/CFLUID
TFLUID = F(L0H1 + J + NY*(NX - NXFIN - 1))/CFLUID
DN = F(L0DX + J + NY*(NX - NXFIN - 1))
DA = F(L0AE + J + NY*(NX - NXFIN - 1))
HLOC = KEFF*ABS((TFIN - TFLUID)/(DN*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HX + J + NY*(NX - NXFIN - 1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
C   IF (J.GT.0) WRITE (6,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
C   IF (J.GT.0) WRITE (6,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
340  CONTINUE
DO 350 I = 1, NX - NXFIN
C   North fin coefficient.
TFIN = F(L0H1 + NY - NYFIN + 1 + NY*(I - 1))/CFLUID
TFLUID = F(L0H1 + NY - NYFIN + NY*(I - 1))/CFLUID
DN = F(L0DY + NY - NYFIN + NY*(I - 1))
DA = F(L0AN + NY - NYFIN + NY*(I - 1))
HLOC = KEFF*ABS((TFIN - TFLUID)/(DN*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HY + NY - NYFIN + NY*(I - 1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
C   IF (I.GT.0) WRITE (6,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
C   IF (I.GT.0) WRITE (14,*) TFIN, TFLUID, DN, DA, ANUMAX, HLOC
350  CONTINUE
C   West plate coefficient.

```

```
DO 360 J = 1, NY - NYFIN
TFLUID = F(L0H1 + J)/CFLUID
DN = 0.5*F(L0DX + J)
DA = F(L0AE + J)
HLOC = KFLUID*ABS((TWALL - TFLUID)/(DN*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HX + J) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
C   IF (J.GT.0) WRITE (6,*) TWALL, TFLUID, DN, DA, ANUMAX, HLOC
C   IF (J.GT.0) WRITE (14,*) TWALL, TFLUID, DN, DA, ANUMAX, HLOC
360  CONTINUE
      END IF
C
      IF(IZ.EQ.(NZ-1)/2) THEN
ZDST = DZG
DO 370 I = 1, NXFIN
DO 370 J = 1, NY
C   West fin coefficient.
TFLUID = F(L0HIH1 + J + NY*(I-1))/CFLUID
TFIN = F(L0H1 + J + NY*(I - 1))/CFLUID
DA = F(L0AH + J + NY*(I-1))
HLOC = KEFF*ABS((TFIN - TFLUID)/(ZDST*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HZ + J + NY*(I-1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
370  CONTINUE
DO 380 J = 1, NYFIN
DO 380 I = NXFIN+1, NX
C   South fin coefficient.
TFLUID = F(L0HIH1 + J + NY*(I-1))/CFLUID
TFIN = F(L0H1 + J + NY*(I - 1))/CFLUID
DA = F(L0AH + J + NY*(I-1))
HLOC = KEFF*ABS((TFIN - TFLUID)/(ZDST*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HZ + J + NY*(I-1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
380  CONTINUE
      ELSE IF (IZ.EQ.((NZ-1)/2+1)) THEN
DO 390 I = NY - NXFIN + 1, NY
DO 390 J = 1, NY
C   East fin coefficient.
TFLUID = F(L0LOH1 + J + NY*(I-1))/CFLUID
TFIN = F(L0H1 + J + NY*(I - 1))/CFLUID
DA = F(L0AH + J + NY*(I-1))
HLOC = KEFF*ABS((TFIN - TFLUID)/(ZDST*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HZ + J + NY*(I-1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ-1) = AHSLAB(IZ) + DA
390  CONTINUE
DO 400 J = NY - NYFIN + 1, NY
DO 400 I = NXFIN+1, NX
C   North fin coefficient.
TFLUID = F(L0LOH1 + J + NY*(I-1))/CFLUID
```

```
TFIN = F(L0H1 + J + NY*(I - 1))/CFLUID
HLOC = KEFF*ABS((TFIN - TFLUID)/(ZDST*DELT))
IF (HLOC.GT.ANUMAX) ANUMAX = HLOC
F(L0HZ + J + NY*(I-1)) = HLOC
NUSLAB(IZ) = NUSLAB(IZ) + HLOC*DA
AHSLAB(IZ) = AHSLAB(IZ) + DA
400  CONTINUE
      END IF
C
C  NULX, NULY, NULZ set below to
C  Nu = h*Dh/Kfluid
C
C  NUSLAB contains slab-size Nu
C
      CALL FN25(INUX,HDIAM/KFLUID)
      CALL FN25(INUY,HDIAM/KFLUID)
      CALL FN25(INUZ,HDIAM/KFLUID)
      CALL FN40(INUX)
      CALL FN40(INUY)
      CALL FN40(INUZ)
      IF (IZ.EQ.NZ) THEN
        ANUMIN = ANUMIN*HDIAM/KFLUID
        ANUMAX = ANUMAX*HDIAM/KFLUID
        F(L0HX + J) = ANUMIN
        F(L0HY + J) = ANUMIN
        F(L0HZ + J) = ANUMIN
        F(L0HX + J + NY*(NX-1)) = ANUMAX
        F(L0HY + J + NY*(NX-1)) = ANUMAX
        F(L0HZ + J + NY*(NX-1)) = ANUMAX
      ELSE
        QSLAB(IZ) = NUSLAB(IZ)*(TWALL - TBSLAB(IZ))
        NUSLAB(IZ) = NUSLAB(IZ)*HDIAM/(AHSLAB(IZ)*KFLUID)
        CALL PRN('NULX',INUX)
        CALL PRN('NULY',INUY)
        CALL PRN('NULZ',INUZ)
      END IF
      END IF
C
C  ----- End of local Nusselt number sub-section-----
C
      RETURN
197 CONTINUE
C  * ----- SECTION 7 ---- Finish of sweep.
C
      IF((FLOAT(ISWEEP)/FLOAT(TSTSWP)).EQ.INT(ISWEEP/TSTSWP))
      1.OR.(ENUFSW.EQ..TRUE.).OR.(ISWEEP.EQ.LSWEEP)) THEN
C
C  Calculate Reynolds number.
C
      RE = WB*HDIAM/VFLUID
C
C  Calculate average drag coefficient.
C
      FBAR = P1FRST(NYFIN+1,NXFIN+1)*HDIAM/(2.*L*RHO1*WB*WB)
C
C  Calculate average Nusselt Number and fin efficiency.
C
```

```
ANTU = (TBUP - TWALL)/(TB(NZ-1) - TWALL)
ALMTD = (TB(NZ-1) - TBUP)
IF(ANTU.GT.0) THEN
  ANTU=LOG(ANTU)
  ALMTD = ALMTD/ANTU
  STBAR = ANTU*AC/AHT
  NUBAR=RE*PRFLUID*STBAR
  JBAR = STBAR*(PRFLUID**(2./3.))
  UOVER = NUBAR*KFLUID/HDIAM
  ETA0 = .8
  HBAR = ETA0*UOVER
  DO 950 KK = 1, 10
    AM = SQRT(2.*ABS(HBAR)*(1. + DELTA/L)/(KFIN*DELTA))
    ETAF = (TANH(AM*ALEFF))/(AM*ALEFF)
    ETA0 = 1. - (AFIN/AHT)*(1. - ETAF)
    HBAR = UOVER/ETA0
  950 CONTINUE
  ELSE
    NUBAR = 999
  END IF
C
C   Output data to Screen and RESULT file.
C
990 FORMAT (' ISWEEP   = ',I4,8X,' DH       = ',G12.4)
889 FORMAT (' AC      = ',G12.4,' AHT      = ',G12.4)
991 FORMAT (' WB      = ',G12.4,' RE       = ',G12.4)
992 FORMAT (' PR      = ',G12.4,' PRT      = ',G12.4)
993 FORMAT (' F       = ',G12.4,' J        = ',G12.4)
994 FORMAT (' ST      = ',G12.4,' NU       = ',G12.4)
995 FORMAT (' CONST    = ',G12.4,' PDIF    = ',G12.4)
996 FORMAT (' TBLAST   = ',G12.4,' ALMTD   = ',G12.4)
997 FORMAT (' C1H      = ',G12.4,' C2H     = ',G12.4)
887 FORMAT (' ETA0     = ',G12.4,' ETAF    = ',G12.4)
  WRITE(6,990) ISWEEP, HDIAM
  WRITE(6,889) AC, AHT
  WRITE(6,991) WB, RE
  WRITE(6,992) PRFLUID, PRT(H1)
  WRITE(6,993) FBAR, JBAR
  WRITE(6,994) STBAR, NUBAR
  WRITE(6,995) CMASS, P1FRST(NYFIN+1,NXFIN+1)
  WRITE(6,996) TB(NZ-1), ALMTD
  WRITE(6,997) C1HEAT, C2HEAT
  WRITE(6,887) ETA0, ETAF
  WRITE(14,990) ISWEEP, HDIAM
  WRITE(14,889) AC, AHT
  WRITE(14,991) WB, RE
  WRITE(14,992) PRFLUID, PRT(H1)
  WRITE(14,993) FBAR, JBAR
  WRITE(14,994) STBAR, NUBAR
  WRITE(14,995) CMASS, P1FRST(NYFIN+1,NXFIN+1)
  WRITE(14,996) TB(NZ-1), ALMTD
  WRITE(14,997) C1HEAT, C2HEAT
  WRITE(14,887) ETA0, ETAF
998 CONTINUE
  WRITE(6,*)
  WRITE(14,*)
  END IF
```



```

      RETURN
198 CONTINUE
C * ----- SECTION 8 ---- Finish of time step.
C
C   Calculate average Nusselt Number from slab-average
C   Nusselt numbers.
C
906   FORMAT (//,' Local data',//, ' IZ ',
1     ' Nu      ', ' Aht      ',
2     ' Wb      ', ' Tb      ', ' Q      ')
907   FORMAT (I3,5(1X, G12.4))
      WRITE (6,906)
      WRITE (14,906)
      QSUM = 0.0
      HSUM = 0.0
      DO 734 KK = 1, NZ-1
      WRITE (6,907) KK, NUSLAB(KK), AHSLAB(KK), WBSLAB(KK),
1     TBSLAB(KK), QSLAB(KK)
      WRITE (14,907) KK, NUSLAB(KK), AHSLAB(KK), WBSLAB(KK),
1     TBSLAB(KK), QSLAB(KK)
      QSUM = QSUM + QSLAB(KK)
      ASUM = ASUM + AHSLAB(KK)
734   CONTINUE
      NUSUM = HDIAM*QSUM/(ASUM*KFLUID*ALMTD)

982   FORMAT (' NUSUM  =',G12.4,' ASUM  =',G12.4)
      WRITE(6,982) NUSUM, ASUM
      WRITE(14,982) NUSUM, ASUM
      RETURN
C*****
C
C   SUBROUTINE SWITCH(ADOWN,AUP,NY,NX,IYLAST,IXLAST)
C
C   REAL ADOWN(NY,NX), AUP(NY,NX)
C   REAL ADOWN(14,28), AUP(14,28)
C   INTEGER NY, NX, I, J, IYLAST, IXLAST
C
C   DO 2501 J = 1, IYLAST
C   DO 2501 I = 1, IXLAST
C   AUP(J,I) = ADOWN(J,I)
C   AUP(IYLAST-J+1,IXLAST-I+1) = ADOWN(J,I)
2501 CONTINUE
      IF (IYLAST.LT.NY) THEN
      DO 2502 J = IYLAST+1, NY
      DO 2502 I = 1, NX
      AUP(J,I) = 2.E-10
2502 CONTINUE
      END IF
      IF (IXLAST.LT.NX) THEN
      DO 2503 J = 1, NY
      DO 2503 I = IXLAST+1, NX
      AUP(J,I) = 3.E-10
2503 CONTINUE
      END IF
      RETURN
      END

```