SANDIA REPORT

SAND2003-1781
Unlimited Release
Printed June 2003

GILA User's Manual

Mark A. Christon

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2003-1781
Unlimited Release
Printed June 2003

GILA User's Manual[1]

Mark A. Christon
Computational Physics R & D Department
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-0819

Revised May 29, 2003

Abstract

GILA is a finite element code that has been developed specifically to attack the class of transient, incompressible, viscous, fluid dynamics problems that are predominant in the world that surrounds us. The purpose for this document is to provide sufficient information for an experienced analyst to use GILA in an effective way. The GILA User's Manual presents a technical outline of the governing equations for time-dependent incompressible flow, and the explicit and semi-implicit projection methods used in GILA to solve the equations. This manual also presents a brief overview of some of GILA's capabilities along with the keyword input syntax and sample problems.

[1]Version 1.1 - NS-SS

Acknowledgments

I wish to thank Phil Gresho for his help and encouragement over the many years of research that are embodied in GILA. I would also like to acknowledge Greg Tipton and Tom Voth for their helpful comments during the preparation of this report.

Contents

1 Introduction **15**

1.1 Guide to the GILA User's Manual 16

1.2 History of GILA Development 16

1.3 GILA Capabilities 17

1.3.1 Fluid Models 17

1.3.2 Turbulence Models 18

1.3.3 Pre-Processor Interfaces 18

1.3.4 Post-Processor Interfaces 18

2 Governing Equations **21**

2.1 Momentum Conservation 21

2.2 Mass Conservation 22

2.3 Energy Conservation 23

2.4 Boundary and Initial Conditions 23

3 Explicit Time Integration **27**

3.1 Modified Finite Element Formulation 31

3.1.1 Reduced-Integration Operators 31

3.2 Domain-Decomposition/Message-Passing 36

3.2.1 Domain-Decomposition 37

3.2.2 Parallel Assembly via Message Passing 37

3.2.3 The Parallel Explicit Algorithm 40

3.2.4 Communication Costs 41

4 The Semi-Implicit Projection Method **43**

4.1 Projection Properties 44

4.2 Time Integration Method 45

5 Boundary Conditions and Source Terms **51**

5.1 Node and Side Sets 51

5.2 Nodal Boundary Conditions 51

5.3 Traction Boundary Conditions 53

5.3.1 Pressure Boundary Conditions 53

5.4 Outflow Boundary Conditions 54
5.5 Body Forces 54
5.6 Flux Boundary Conditions 55
5.7 Pressure Levels 56

6 Running GILA 57
6.1 Execution 57
6.2 Restarts 58

7 GILA Control File 59
7.1 Analysis Title (title) 59
7.2 Analysis Parameters (analyze – end) 61
7.2.1 solve 61
7.2.2 temp 62
7.2.3 species 62
7.2.4 react 62
7.2.5 nstep 62
7.2.6 term 63
7.2.7 deltat 63
7.2.8 average – ; 63
7.2.9 cfl 64
7.2.10 dtchk 65
7.2.11 thetak 65
7.2.12 thetab 65
7.2.13 thetaa 65
7.2.14 thetaf 65
7.2.15 hglass 66
7.2.16 epshg 66
7.2.17 epsdt 66
7.2.18 divu 66
7.2.19 mass 66
7.2.20 btd 67
7.2.21 fct 67
7.2.22 icset 67
7.2.23 pltype 69
7.2.24 plti 69
7.2.25 prtlev 69
7.2.26 prti 69
7.2.27 ttyi 70
7.2.28 thtype 70
7.2.29 vortics 70
7.2.30 icwrt 70

7.2.31 dump 71
7.3 Fluid Models (material – end) 71
7.3.1 model 71
7.3.2 rho 72
7.3.3 mu 72
7.3.4 k 72
7.3.5 diff1 72
7.3.6 beta 72
7.3.7 t_ref 72
7.3.8 gx 72
7.3.9 gy 73
7.3.10 gz 73
7.3.11 p0 73
7.3.12 a 73
7.3.13 e 73
7.3.14 r 73
7.3.15 cp 73
7.3.16 gamma 74
7.3.17 q0 74
7.4 Momentum Equation Solver (momsol – end) 74
7.4.1 itmax 75
7.4.2 itchk 75
7.4.3 eps 75
7.4.4 wrt 75
7.4.5 hist 76
7.5 Pressure Equation Solvers (ppesol – end) 76
7.5.1 itmax 77
7.5.2 itchk 77
7.5.3 novec 77
7.5.4 eps 78
7.5.5 wrt 78
7.5.6 hist 78
7.5.7 omega 78
7.5.8 stab 78
7.5.9 beta 79
7.6 Nodal Time History Blocks (ndhist – end) 79
7.6.1 st 79
7.6.2 en 80
7.6.3 nstep 80
7.7 Turbulence Models 80
7.7.1 smagc 80
7.8 Nodal Boundary Conditions 81

7.8.1 nodeset . . . 81
7.8.2 ubc – end . . . 81
7.8.3 vbc – end . . . 81
7.8.4 wbc – end . . . 82
7.8.5 tbc – end . . . 82
7.8.6 rbc – end . . . 82
7.8.7 kbc – end . . . 82
7.8.8 ebc – end . . . 82
7.8.9 abc – end . . . 82
7.8.10 zbc – end . . . 82
7.8.11 ybc – end . . . 83
7.9 Outflow Boundary Conditions . . . 83
7.9.1 sideset . . . 83
7.10 Pressure Boundary Condition (pbc – end) . . . 83
7.11 Pressure Levels (ppebc – end) . . . 83
7.12 Initial Conditions . . . 84
7.12.1 Initial Condition Functions (icfn – end) . . . 84
7.12.2 Initial Condition Input (icfile – end) . . . 85
7.13 Derived Force Computation (dforce – end) . . . 85
7.13.1 sideset . . . 86
7.14 Load Curve (lcurve – end) . . . 86
7.15 Parallel File Sytem (pfs – end) . . . 86
7.15.1 ndisk . . . 87
7.15.2 start . . . 87

8 Example Problems **89**
8.1 Entrance Region in a 2-D Duct . . . 89
8.2 Backward Facing Step . . . 95
8.3 Vortex Shedding . . . 99
8.4 Momentum-Driven Jet . . . 103
8.5 Post & Plate . . . 107

A GILA Mesh File **111**
A.1 Mesh Description Line . . . 112
A.2 Header Block . . . 112
A.3 Nodal Coordinates . . . 113
A.4 Element Connectivity . . . 113
A.5 Node Sets . . . 114
A.6 Side Sets . . . 115
A.7 Sample ASCII Mesh File . . . 119

B Data Visualization Interfaces **123**

B.1 GRIZ - Binary Plot Files 123

List of Figures

2.1 Flow domain for conservation equations. 24

3.1 Velocity mesh with two degrees-of-freedom (DOF) per node, and the PPE dual grid with one DOF per element. 28
3.2 Grid parameters for: a) Two-dimensional element with centroid velocity and characteristic element dimensions h_ξ and h_η, b) Three-dimensional element with centroid velocity and characteristic dimensions h_ξ, h_η, and h_ζ. 30
3.3 Four processor spatial domain decomposition for a two-dimensional vortex-shedding mesh. 38
3.4 Parallel assembly procedure: a) Sequential Mesh, b) 2-Processor sub-domain mapping, c) Parallel assembly mapping. 39

5.1 Flow domain with inflow Γ_i, outflow Γ_o, and no-slip, no-penetration Γ_s boundaries. 52

7.1 A Sample GILA Control File 60

8.1 Mesh for the two-dimensional duct entrance region. 90
8.2 Control file for two-dimensional duct entrance region. 90
8.3 Divergence diagnostics for the initial conditions before and after the div-free projection. 91
8.4 Screen report of grid parameters showing the minimum and maximum grid Reynolds and CFL numbers and and associated element numbers. 92
8.5 Screen output of minimum and maximum velocity values at every 10 time steps. 93
8.6 Nodal time history plots for a) x-velocity, b) y-velocity, c) kinetic energy, d) outlet x-velocity profile at t=10 time units. 93
8.7 Snapshot of a) x-velocity, b) y-velocity, and c) pressure at $t = 10$ time units. 94
8.8 Coarse-grid mesh (case-B) for the Re=800 backward facing step. 96
8.9 Control file for the backward facing step. 96

8.10 Backward facing step at 400 time units: a) and b) Pressure field, c) and d) Vorticity field. 97
8.11 Coarse mesh for cylinder vortex shedding. 100
8.12 Control file for the $Re = 100$ vortex shedding problem. 101
8.13 Kinetic energy time history for the coarse mesh. 101
8.14 $Re = 100$ flow past a circular cylinder at 150 time units: a) vorticity field, b) pressure field. 102
8.15 Momentum-driven jet mesh with 11250 elements and 11466 nodes. 104
8.16 Control file for the momentum-driven jet. 105
8.17 Snapshot of a) temperature field, b) z-vorticity field and c) pressure for the momentum driven jet. (The temperature, vorticity and pressure fields have been reflected about the vertical centerline for presentation purposes.) 106
8.18 Post and plate Mesh. 108
8.19 Post and plate control file. 109
8.20 $Re = 100$ flow past post and plate at 300 time units. a) Pressure, b) X-Vorticity (isosurface values ± 0.75), c) Helicity (isosurface values ± 0.15), d) Y-Vorticity, (isosurface values ± 0.75), e) Enstrophy, and f) Z-Vorticity field. 110

A.1 Example description line and header block for ASCII mesh file. 113
A.2 Example node set section of the ASCII mesh file. 116
A.3 Canonical node and side numbering for the a) 2-D quadrilateral element, and b) the 3-D hexahedral element. 117
A.4 Example side set section of the ASCII mesh file. 118

List of Tables

3.1 Memory requirements and operations counts for a matrix-vector multiply for various element integration rules, stabilization operators, and storage schemes. (Nel: number of elements, Nnp: number of nodes, EBE: element-by-element) . . 36

8.1 Separation lengths for the backward facing step, $Re_H = 800$. Note that the element side lengths are for the uniform region of the mesh. (l_1 – length from the step face to the lower re-attachment point, l_2 – length from the step face to upper separation point, l_3 – length of the upper separation bubble. The element size is measured at the step inlet.) 98

8.2 Vortex shedding mesh parameters. (Δr_i: radial mesh spacing at the cylinder surface, Δr_o: radial mesh spacing at $x = \pm 4$.) 100

B.1 2-D GILA State and Time History Variables 124

B.2 3-D GILA State and Time History Variables 125

B.3 GILA Global Time History Variables 125

Preface

The goal for GILA has been to achieve a high level of computational performance across a spectrum of supercomputer architectures without sacrificing any of the aspects of the finite element method that make it flexible and capable for a broad class of fluid dynamics problems. GILA is based, in part, on the research code, HYDRA,[12] developed by the author at Lawrence Livermore National Laboratory (LLNL). GILA has been designed to take advantage of advanced solution algorithms for distributed-memory, parallel supercomputer architectures.

The development of GILA has drawn, in part, upon over ten years of research in computational fluid dynamics as well as from the research efforts of Phil Gresho, Stevens Chan and their colleagues at LLNL. Like HYDRA, GILA has also drawn upon the many years of finite element expertise embodied in DYNA3D[66] and NIKE3D.[50] Certain key architectural ideas from both DYNA3D and NIKE3D have been adopted and further refined to fit the advanced dynamic memory management and cache-based data structures implemented in GILA.

GILA, in its implementation, reflects, my training and experience with a broad array of supercomputers ranging from vector machines to massively-parallel distributed-memory machines such as the ASCI TFLOPs machine at Sandia National Laboratories. My philosophy for GILA has been to always focus on the most advanced solution algorithms for time-dependent incompressible flow and the concomitant mapping to parallel supercomputer architectures for the sole purpose of solving large scale problems – a philosophy I learned from Pat Burns and Dan Pryor while a graduate student at Colorado State University.

Chapter 1

Introduction

The simulation of time-dependent flow about vehicles and in turbomachinery remains a computational grand challenge despite the rapid increases in computing power observed over the past decade. An example of this class of computational fluid dynamics problem is the transient simulation of flow around a submarine or an automobile. In order to compute the flow around such a vehicle, $O(10^6)$ elements are required just to capture the geometry and the largest flow features in regions of separated flow. In addition to the high degree of spatial discretization, the temporal resolution for this class of problem is also demanding, ultimately requiring the effective mapping of flow-solution algorithms to current supercomputer architectures.

GILA is a finite element code that solves the transient, incompressible, viscous, Navier-Stokes equations, and is based, in part upon the work of Gresho, et al.[24,25,28,29] GILA makes use of multiple advanced solution algorithms for both semi-implicit and explicit time integration. The explicit solution algorithm[28,29] sacrifices some phase accuracy, but decouples the momentum equations and minimizes the memory requirements. While both the diffusive and Courant-Freidrichs-Levy (CFL) stability limits must be respected in the explicit algorithm, balancing tensor diffusivity ameliorates the restrictive diffusive stability limit and raises the order of accuracy of the time integration scheme. The explicit algorithm, in combination with single point integration and hourglass stabilization, has proven to be both simple and efficient in a computational sense. Because of this, the explicit algorithm was the focus of early parallelization efforts in GILA.

In the second-order projection algorithm,[24,25] a consistent-mass predictor in conjunction with a lumped mass corrector legitimately decouples the velocity and pressure fields thereby reducing both memory and CPU requirements relative to more traditional fully-coupled solution strategies for the Navier-Stokes equations. The consistent mass predictor retains phase speed accuracy, while the lumped mass corrector (a projection to a divergence-free subspace) maintains a div-free velocity field. Both the predictor and the corrector steps are amenable to solution via direct or preconditioned iterative techniques making it possible to tune the algorithm to the computing platform, i.e., parallel, vector or shared-memory. The second-order projection algorithm can accurately track shed vortices, and is amenable to the incorporation of either simple or complex (multi-equation) turbulence sub-models appropriate for a broad spectrum of applications.

1.1 Guide to the GILA User's Manual

The purpose for this document is to provide sufficient information for an experienced analyst to use GILA in an effective way. The assumption is that the user is somewhat familiar with modern supercomputers, large scale computing, common CFD practices, and to a certain degree, the current CFD literature. This manual provides sufficient references to the literature to permit the interested reader to pursue the technical details of GILA.

In this document, an attempt is made to adhere to the convention that all keywords and defaults for input data appear in a **boldface** type, and sample computer input/output appears in a `typewriter` font. All other keywords, parameters and variables are defined in the context they are used.

In Chapter 2, an overview of the theoretical background for GILA is presented. Chapters 3 and 4 present information on how to execute GILA in a UNIX environment and the necessary input data for GILA. Several sample calculations are presented in Chapter 5 which can be used as benchmark problems for the first time GILA user.

1.2 History of GILA Development

The original idea behind GILA was to develop a next-generation CFD code using the most current algorithmic ideas for incompressible flow coupled with a single-program multiple-data (SPMD) programming model and aggressive on-processor performance programming strategies. GILA has drawn on my experiences participating in the early ParaDyn team's efforts to parallelize DYNA3D[16,43] as well as the research efforts with HYDRA.[12] GILA code development efforts started essentially from scratch to avoid problems with inherited sequential code and with antiquated memory management schemes, although the requisite linear algebra has come from the ITLIB package originally developed by the author at Colorado State University.

GILA was developed using standard tools for source configuration, and control. Research efforts with HYDRA at LLNL on the Thinking Machines CM-200 and CM-5 were only partially successful due to the very "long-vector" characteristics of these architectures. This experience has pushed GILA development away from a data-parallel implementation and towards a more portable and scalable domain-decomposition message-passing model. The most recent algorithm mapping efforts with GILA have been directed towards the ASCI TFLOPs because of its superior network bandwidth, fast scalar speed, and large number of processors. However, the use of the domain-decomposition message-passing paradigm permits scalability from PC's to networks of workstations and ultimately machines like the ASCI TFLOPs platform.

Over the past several years, GILA has been under *spare-time* development, and has acted as a test bed not only for investigating the issues involved in mapping incompressible flow solution algorithms to parallel architectures, but also for the study of optimal solution methods for the pressure Poisson equation, and for the study of advanced Navier-Stokes solution algorithms. Presently, an experimental model for Large Eddy Simulation is available in GILA, and more traditional turbulence models are under development.

1.3 GILA Capabilities

GILA provides multiple analysis options for both 2-D and 3-D transient, viscous, incompressible flow problems. Of course, the analysis of problems with thermal convection is a subset of the 2-D and 3-D analysis options. In addition to the implicit and explicit algorithms for solving the transient Navier-Stokes equations, GILA also provides both implicit and explicit algorithms for solving the time-dependent scalar advection-diffusion equation.

1.3.1 Fluid Models

Fluid constitutive models in GILA may be broadly classified into two groups generally referred to as material models. The first category consists of the definition of constant properties, i.e., fluid density, kinematic viscosity, thermal diffusivity, etc. For flow problems that require only one material definition, GILA provides a simplified input format for specification of the fluid properties.

The second category of material model involves the definition of a relation between fluid properties and dependent variables such as velocity and temperature. The internal architecture of GILA permits the use of this class of material models but, at this time user access to this type of material model in GILA is restricted. In the future, user access for alternative constitutive models will be provided.

1.3.2 Turbulence Models

GILA has provided a test-bed for a variety of time-dependent Reynolds-averaged Navier Stokes (RANS) models. Currently, an experimental Smagorinsky subgrid-scale model is available for performing Large Eddy Simulations (LES). At this time, all of the turbulence models are considered experimental.

1.3.3 Pre-Processor Interfaces

At this time, mesh generation support for GILA is provided in several forms. First, meshes generated in the EXODUS II format[59] may be translated to a GILA compatible format using the EX2MSH utility. The EX2MSH utility relies on the EXODUS II I/O library. Therefore, EX2MSH is currently only available at Sandia National Laboratories. Second, the TRUEGRID mesh generator[67] produces "native" GILA mesh files. Because GILA can make use of a simplified ASCII mesh file format, it is very easy to adapt existing neutral files from other commercial mesh generators. Appendix A provides a brief description of GILA's ASCII file format.

1.3.4 Post-Processor Interfaces

GILA can output several forms of graphics files. The default file format uses the binary, familied graphics data format for the plot and time history files that are compatible with GRIZ[18] and THUG[61] respectively. GRIZ is used for visualizing snapshots of the entire flow-field (state data) or generating animations of the time varying flow-field data, while THUG[61] is used for interrogating time history data at a moderate number of mesh points.

GRIZ and THUG are general purpose scientific visualization tools for finite element codes, and they support analysis codes for both computational fluid dynamics and computational mechanics. Because of this, there is a translation from the primitive variables that GILA writes to the graphics datafiles to variables which can be displayed in GRIZ and THUG. The details on how GILA variables map to GRIZ and THUG variables may be found in Appendix B.

GILA also provides the ability to output state graphics using the EXODUS[59] and PXI parallel I/O libraries. The PXI library has been designed for large scale parallel simulations to avoid I/O bottlenecks and is based on the PDS/PIO[62] libraries which provide lightweight collective parallel I/O facilities. The PXI state graphics files are compatible with Sandia's MUSTAFA visualizer. The EXODUS state graphics files are compatible with any visualizers that read the EXODUS file format, e.g., BLOT, MUSTAFA, or SPYVIEW.

For time-history data, an optional output interface is provided for the HISPLT[64] time-history software for use internally at Sandia.

Chapter 2

Governing Equations

This chapter presents the basic forms of the partial differential equations that GILA solves, and Chapters 3 – 4 provide a general description of the methodologies employed in their solution. The interested reader may pursue the references included in these chapters for details on the incompressible flow solution algorithms used in GILA and their implementation.

In the ensuing discussion, the invariant bold-face vector notation of Gibbs is used with **boldface** symbols representing vector/tensor quantities. The reader may refer to Gresho and Sani[32] pp. 357-359 for an outline of notation for the Navier-Stokes equations.

2.1 Momentum Conservation

To begin, the conservation of linear momentum is

$$\rho \left\{ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right\} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{f}, \tag{2.1}$$

where $\mathbf{u} = (u, v, w)$ is the velocity, $\boldsymbol{\sigma}$ is the stress tensor, ρ is the mass density, and $\mathbf{f}$ is the body force. The body force contribution $\rho\mathbf{f}$ typically accounts for buoyancy forces with $\mathbf{f}$ representing the acceleration due to gravity.

The stress may be written in terms of the fluid pressure and the deviatoric stress tensor as

$$\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\tau}, \tag{2.2}$$

where p is the pressure, $\mathbf{I}$ is the identity tensor , and $\boldsymbol{\tau}$ is the deviatoric stress tensor.

A constitutive equation relates the deviatoric stress and the strain rate, e.g.,

$$\boldsymbol{\tau} = 2\boldsymbol{\mu}\mathbf{S}. \tag{2.3}$$

Here, the dynamic viscosity, $\boldsymbol{\mu}$, is a second-rank tensor. Frequently, the fluid viscosity is only available as an isotropic viscosity, in which case, the constitutive relation becomes

$$\boldsymbol{\tau} = 2\mu\mathbf{S}. \tag{2.4}$$

The strain-rate tensor is written in terms of the velocity gradients as

$$\mathbf{S} = \frac{1}{2}\left\{\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right\}. \tag{2.5}$$

Remark. In GILA, the viscosity, thermal conductivity, and mass diffusivities are treated as second-rank tensors even though these properties may only be available as scalar quantities for some fluids. In the limiting case of a scalar material property such as viscosity, the internal code-representation assumes that the viscosity is $\mu_{ij} = \delta_{ij}\hat{\mu}$ where $\hat{\mu}$ is the user-input scalar viscosity.

2.2 Mass Conservation

The mass conservation principle in divergence form is

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\nabla\mathbf{u}) = 0. \tag{2.6}$$

In the incompressible limit, the velocity field is solenoidal,

$$\nabla\cdot\mathbf{u} = 0, \tag{2.7}$$

which implies a mass density transport equation,

$$\frac{\partial\rho}{\partial t} + \mathbf{u}\cdot\nabla\rho = 0. \tag{2.8}$$

For constant density, Eq. (2.8) is neglected with Eq. (2.7) remaining as a constraint on the velocity field.

GILA provides the capability to transport up to 10 species represented by the mass concentration $Z_1, Z_2, ..., Z_{10}$. In order to simplify the presentation, a single mass fraction is presented representing a binary mixture. In order to account for the change in mass concentration, mass conservation applied to the individual species yields for Z_1

$$\rho\frac{\partial Z_1}{\partial t} + \rho\mathbf{u}\cdot\nabla Z_1 = -\nabla\mathbf{J}_1 + \dot{m}_1, \tag{2.9}$$

where $\mathbf{J}_1$ is the diffusional mass flux rate, and $\dot{m}_1$ is a volumetric mass source. The mass source may include the injection of mass concentration from a boundary or the source/sink terms from chemical reactions.

The diffusional mass flux rate is based on Fick's law of diffusion,

$$\mathbf{J}_1 = -\rho \mathbf{D}_1 \nabla Z_1, \tag{2.10}$$

where $\mathbf{D}$ is a tensorial mass diffusivity. Typically mass diffusivities are only available as scalars so that

$$\mathbf{J}_1 = -\rho \mathcal{D}_1 \nabla Z_1. \tag{2.11}$$

In the most general form, the species concentration transport equations are

$$\rho \frac{\partial Z_I}{\partial t} + \rho \mathbf{u} \cdot \nabla Z_I = -\nabla \cdot \mathbf{J}_I + \dot{m}_I, \tag{2.12}$$

where I indicates the species mass concentration, i.e., $I = 1, 2, ..., 10$.

2.3 Energy Conservation

The conservation of energy is expressed in terms of temperature, T, as

$$\rho C_p \left\{ \frac{\partial T}{\partial t} + \mathbf{u} \nabla T \right\} = -\nabla \cdot \mathbf{q} + Q \tag{2.13}$$

where C_p is the specific heat at constant pressure, $\mathbf{q}$ is the diffusional heat flux rate, and Q represents volumetric heat sources and sinks, e.g., due to exothermic/endothermic chemical reactions

Fourier's law relates the heat flux rate to the temperature gradient and thermal conductivity,

$$\mathbf{q} = -\mathbf{k} \nabla T, \tag{2.14}$$

where $\mathbf{k}$ is the thermal conductivity tensor. In many cases, the fluid properties are only available as scalar quantities, i.e., the thermal conductivity is considered to be isotropic.

2.4 Boundary and Initial Conditions

The prescription of boundary conditions is based on a flow domain with boundaries that are either physical or implied for the purposes of performing a simulation. A simple flow domain is shown in Figure 2.1 where the boundary of the domain is $\Gamma = \Gamma_1 \cup \Gamma_2$.

The momentum equations, Eq. (2.1), are subject to boundary conditions that consist of specified velocity on Γ_1 as in Eq. (2.15), or traction boundary conditions on Γ_2 as in Eq. (2.17).

$$\mathbf{u}(\mathbf{x}, t) = \hat{\mathbf{u}}(\mathbf{x}, t) \ \mathit{on} \ \Gamma_1 \tag{2.15}$$

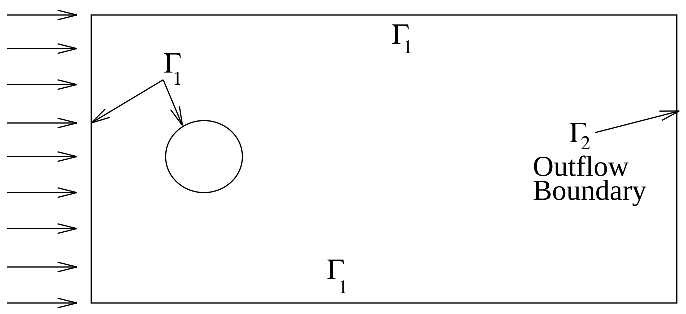


Figure 2.1: Flow domain for conservation equations.

In the case of a no-slip and no-penetration boundary, $\mathbf{u} = 0$ is the prescribed velocity boundary condition.

The prescribed traction boundary conditions are

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \hat{\mathbf{f}}(\mathbf{x}, t) \textit{ on } \Gamma_2, \tag{2.16}$$

where $\mathbf{n}$ is the outward normal for the domain boundary, and $\hat{\mathbf{f}}$ are the components of the prescribed traction. In terms of the pressure and strain-rate, the traction boundary conditions are

$$\{-p\mathbf{I} + 2\mu\mathbf{S}\} \cdot \mathbf{n} = \hat{\mathbf{f}}(\mathbf{x}, t) \textit{ on } \Gamma_2. \tag{2.17}$$

The traction and velocity boundary conditions can be mixed. In a two-dimensional sense, mixed boundary conditions can consist of a prescribed normal traction and a tangential velocity. For example, at the outflow boundary in Figure 2.1, a homogeneous normal traction and vertical velocity on Γ_2 constitutes a valid set of mixed boundary conditions. A detailed discussion of boundary conditions for the incompressible Navier-Stokes equations may be found in Gresho and Sani.[32]

Turning attention to the species transport equations, boundary conditions for Eq. (2.9) may consist of either a prescribed concentration or a mass flux rate. In the binary mixture example, the prescribed concentration is

$$Z_1(\mathbf{x}, t) = \hat{Z}_1(\mathbf{x}, t), \tag{2.18}$$

where $\hat{Z}_1$ is the known value of concentration for species 1. The prescribed mass flux rate is

$$-\rho\mathbf{D}_1\nabla Z_1 \cdot \mathbf{n} = \hat{J}_1(\mathbf{x}, t), \tag{2.19}$$

where $\hat{J}_1(x_i, t)$ is the known mass flux rate through the boundary with normal $\mathbf{n}$. The prescribed flux rate may also be specified in terms of a mass transfer coefficient as

$$-\rho\mathbf{D}_1\nabla Z_1 \cdot \mathbf{n} = h_{\mathcal{D}_\infty}(Z_1 - Z_{1_\infty}), \tag{2.20}$$

where $h_{\mathcal{D}_\infty}$ is the mass transfer coefficient and Z_{1_∞} is a reference species concentration.

The boundary conditions for the energy equation, Eq. (2.13), consist of a prescribed temperature or heat flux rate. The prescribed temperature is

$$T(\mathbf{x},t) = \hat{T}(\mathbf{x},t), \tag{2.21}$$

and the prescribed heat flux rate is

$$-\mathbf{k}\nabla T \cdot \mathbf{n} = \hat{q}(\mathbf{x},t), \tag{2.22}$$

where $\hat{q}$ is the known flux rate through the boundary with normal $\mathbf{n}$. The heat flux rate may also be prescribed in terms of a heat transfer coefficient,

$$-\mathbf{k}\nabla T \cdot \mathbf{n} = h(T - T_\infty), \tag{2.23}$$

where h is the heat transfer coefficient, and T_∞ is a reference temperature.

Initial conditions take on the form of prescribed velocity, species and temperature distributions at $t = 0$, i.e.,

$$\begin{aligned} \mathbf{u}(\mathbf{x},0) &= \mathbf{u}^0(\mathbf{x}), \\ Z_I(\mathbf{x},0) &= Z_I^0(\mathbf{x}), \\ T(\mathbf{x},0) &= T^0(\mathbf{x}). \end{aligned} \tag{2.24}$$

Remark. For a well-posed incompressible flow problem, the prescribed initial velocity field in equation (2.25) must satisfy equations (2.25) and (2.26) (see Gresho and Sani[31]). If $\Gamma_2 = 0$ (the null set, i.e., enclosure flows with $\vec{n} \cdot \mathbf{u}$ prescribed on all surfaces), then global mass conservation enters as an additional solvability constraint as shown in equation (2.27).

$$\nabla \cdot \mathbf{u}^0 = 0 \tag{2.25}$$

$$\mathbf{n} \cdot \mathbf{u}(\mathbf{x},0) = \mathbf{n} \cdot \mathbf{u}^o(\mathbf{x}) \tag{2.26}$$

$$\int_\Gamma \mathbf{n} \cdot \mathbf{u}^o d\Gamma = 0 \tag{2.27}$$

Chapter 3

Explicit Time Integration

This chapter presents the spatial discretization and explicit time-integration method for the incompressible Navier-Stokes equations. The spatial discretization is achieved using the $Q1Q0$ element with bilinear support for velocity and piecewise constant support for the pressure in two dimensions. In three dimensions, the velocity support is trilinear with piecewise constant support for pressure. The methods for obtaining the weak-form of the conservation equations are well known and will not be repeated here (see for example, Gresho, et al.,[30] Hughes,[44] and Zienkiewicz and Taylor[68]). The spatially discrete form of Eq. (2.1) and (2.7) are

$$M\dot{\mathbf{u}} + A(\mathbf{u})\mathbf{u} + K\mathbf{u} + Cp = \mathbf{F}, \tag{3.1}$$

and

$$C^T\mathbf{u} = 0. \tag{3.2}$$

M is the mass matrix, $A(\mathbf{u})$ and K are the the advection and the viscous diffusion operators respectively, and $\mathbf{F}$ is the body force. C is the gradient operator, and C^T is the divergence operator. Here, $\mathbf{u}$ and p are understood to be discrete approximations to the continuous velocity and pressure fields. Equations (3.1) and (3.2) constitute a differential-algebraic system of equations that precludes the direct application of time-marching algorithms due to the presence of the discrete incompressibility constraint.

Following Gresho, et al.,[28] a consistent, discrete pressure Poisson equation (PPE) is constructed using a row-sum lumped mass matrix, M_L.

$$[C^T M_L^{-1} C]p = C^T M_L^{-1}[\mathbf{F} - K\mathbf{u} - A(\mathbf{u})\mathbf{u}] \tag{3.3}$$

The PPE constitutes an algebraic system of equations that is solved for the element-centered pressure during the time-marching procedure. Figure 3.1 shows the dual, staggered grid associated with the pressure variables.

The PPE in Eq. (3.3) incorporates the effect of the essential velocity boundary conditions from Eq. (2.15), and automatically builds in the boundary conditions from Eq. (2.17) – see Gresho, et al.[31]

Equations (3.1) and (3.3) form the basis for a description of the explicit time integration algorithm. It is assumed that the explicit algorithm begins with a given divergence-free velocity field, $\mathbf{u}^0$, that satisfies the essential velocity boundary conditions, and an initial pressure, p^0. The explicit algorithm proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient, at time level n.

$$\tilde{\mathbf{a}}^n = M_L^{-1}\tilde{\mathbf{F}}^n \tag{3.4}$$

   where

$$\tilde{\mathbf{F}}^n = \mathbf{F}^n - K\mathbf{u}^n - A(\mathbf{u})\mathbf{u}^n \tag{3.5}$$

2. Solve the global PPE for the current pressure field.

$$[C^T M_L^{-1} C]p^n = C^T\tilde{\mathbf{a}}^n \tag{3.6}$$

3. Update the nodal velocities.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t[\tilde{\mathbf{a}}^n + M_L^{-1}Cp^n] \tag{3.7}$$

4. Repeat steps 1-3 until a maximum simulation time limit or maximum number of time steps is reached.

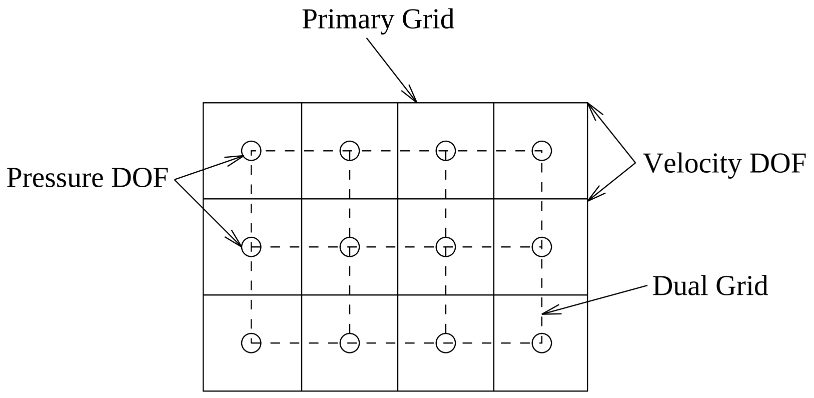


Figure 3.1: Velocity mesh with two degrees-of-freedom (DOF) per node, and the PPE dual grid with one DOF per element.

Remark. In GILA, the prescribed initial conditions and boundary conditions are tested and, if necessary, a projection to a divergence-free subspace is performed on the initial velocity field, $\mathbf{u}^0$. This guarantees that the flow problem is well-posed, even if the user prescribed initial conditions violate the conditions of Eq. (2.25) - (2.26) presented in Chapter 2.

In practice, the criterion for performing a projection onto a div-free subspace is based upon the RMS divergence error

$$\sqrt{\frac{(C^T\mathbf{u})\cdot(C^T\mathbf{u})}{Nel}} \leq \epsilon, \tag{3.8}$$

where Nel is the number of elements and ϵ is a user-specified tolerance typically 10^{-10} to 10^{-6}. If the RMS divergence error is greater than the specified tolerance for the initial candidate velocity field, $\hat{\mathbf{u}}^0$, then the PPE problem in Eq. (3.9) is solved for λ, and a mass-consistent projection performed using Eq. (3.10).

$$[C^T M_L^{-1} C]\lambda = C^T\hat{\mathbf{u}}^0 \tag{3.9}$$

$$\mathbf{u}^0 = \hat{\mathbf{u}}^0 - M_L^{-1}C\lambda \tag{3.10}$$

The explicit time-integration algorithm must respect both diffusive and convective stability limits. Although the analytical stability limits for the explicit time integration of the Navier-Stokes equations in multiple dimensions remain intractable,[28] approximate stability computations may be performed using local grid metrics.

In an unstructured grid, with variable element size, the calculation of the grid Re (Reynolds) and CFL (Courant-Freidrichs-Levy) numbers uses the element-local coordinates and centroid velocities. Figure 3.2 shows the canonical element-local node-numbering scheme, coordinate system and centroid velocity for the 2-D and 3-D elements.

The grid Re and CFL numbers are defined as

$$Re_i = \frac{|\overline{\mathbf{u}}\cdot\mathbf{h}_i|}{2\nu} \tag{3.11}$$

$$CFL_i = \frac{|\overline{\mathbf{u}}\cdot\mathbf{h}_i|\Delta t}{\|\mathbf{h}_i\|^2} \tag{3.12}$$

where $i = \xi$, η, ζ are the element-local coordinates. The grid Re and CFL numbers rely upon the projection of the centroid velocity onto the element-local coordinate directions that are oriented according to the canonical local node numbering scheme for each element type.[11,14]

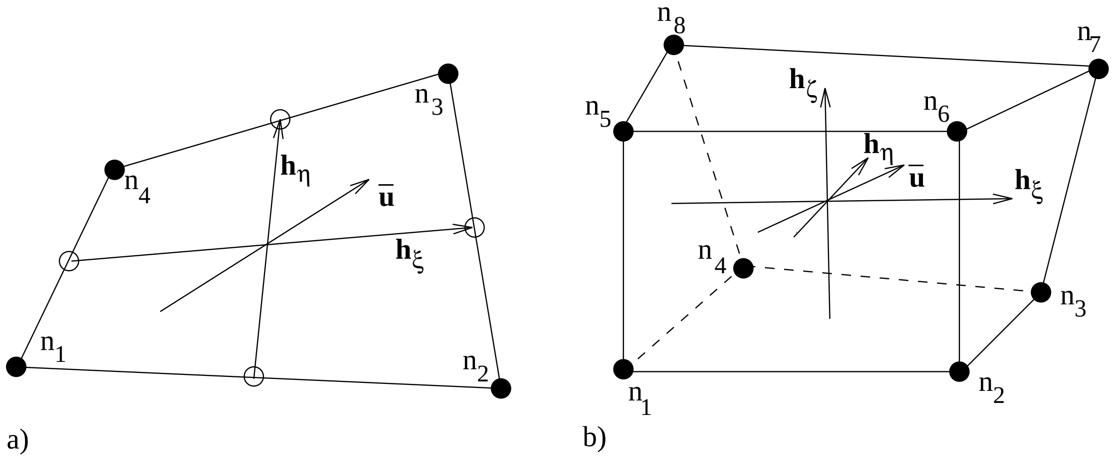


Figure 3.2: Grid parameters for: a) Two-dimensional element with centroid velocity and characteristic element dimensions h_ξ and h_η, b) Three-dimensional element with centroid velocity and characteristic dimensions h_ξ, h_η, and h_ζ.

In order to use Eq. (3.12) to estimate a stable time step, a unit vector for each element-local coordinate direction is defined as

$$\hat{\mathbf{e}}_i = \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|} \tag{3.13}$$

where $\hat{\mathbf{e}}_i$ denotes the unit vector for each of the (ξ, η, ζ) coordinate directions. Using the grid Re and the element size, $\mathbf{h}$, the advective-diffusive stability limit becomes

$$\Delta t_i \leq \nu \|\mathbf{h}_i\|^2 \left\{1 + \sqrt{1 + (Re_i)^2}\right\}^{-1} \tag{3.14}$$

where a minimum over all elements and all element-local coordinate directions establishes a global minimum time-step.

The advective stability limit is established in a similar manner using

$$\Delta t_i \leq \frac{CFL\|\mathbf{h}_i\|}{|\overline{\mathbf{u}} \cdot \mathbf{h}_i|}. \tag{3.15}$$

The stable time step is based upon the minimum time step derived from either Eq. (3.14) or (3.15). However, for meshes graded to resolve boundary layers, the advective-diffusive stability limit usually dictates the time step.

3.1 Modified Finite Element Formulation

Several ad-hoc modifications are made to the standard Galerkin finite element formulation for the explicit time integration algorithm. These modifications include the use of a row-sum lumped mass matrix, single point Gaussian quadrature, balancing tensor diffusivity (BTD), and hourglass stabilization to damp the spurious zero-energy modes known as *keystone* or *hourglass* modes. A detailed numerical analysis of these modifications is discussed in Gresho, et al.[28]

Before discussing the reduced integration operators, a brief overview of the element matrices associated with Eq. (3.2) and (3.1) is presented. The element level gradient, mass, advection and diffusion operators are presented in Eq. (3.16) – (3.19). Here, N_a is the element shape function, the subscripts a and b range from 1 to the number of nodes per element, $Nnpe$, and $1 \leq i, j \leq Ndim$.

$$C^e_{i_a} = -\int_{\Omega^e} \frac{\partial N_a}{\partial x_i} dV \tag{3.16}$$

$$M^e_{ab} = \int_{\Omega^e} N_a N_b dV \tag{3.17}$$

$$A^e_{ab}(\mathbf{u}) = \int_{\Omega^e} N_a \; u_i \frac{\partial N_b}{\partial x_i} dV \tag{3.18}$$

$$K^e_{ab} = \int_{\Omega^e} \frac{\partial N_a}{\partial x_i} \mu_{ij} \frac{\partial N_b}{\partial x_j} dV \tag{3.19}$$

3.1.1 Reduced-Integration Operators

From an examination of the explicit algorithm, it is clear that the bulk of the computational effort for a discrete time step consists of the formation and assembly of the right-hand-side in Eq. (3.5), and the subsequent PPE solution in Eq. (3.6). The right-hand-side vector is formed element-by-element using a right-to-left matrix-vector multiply with reduced integration operators for the diffusive and advective terms. In this context, reduced integration is considered synonymous with single-point Gaussian quadrature. Using single-point integration, the element area and gradient operators may be written strictly in terms of element-local nodal coordinates.

$$A^e = \frac{1}{2}[x_{31}y_{42} + x_{24}y_{32}] \tag{3.20}$$

$$C^e_x = \frac{1}{2A^e}[y_{24}, y_{31}, -y_{24}, -y_{31}]$$

$$C_y^e = \frac{1}{2A^e}[x_{42}, x_{13}, -x_{42}, -x_{13}] \tag{3.21}$$

In Eq. (3.20) and (3.21), $x_{ab} = x_a - x_b$, where the subscripts a and b identify the local node number and range from 1 to 4 for the two-dimensional bilinear element. The fact that $C_{x3} = -C_{x1}$, and $C_{x4} = -C_{x2}$, permits the storage of only the unique values in the gradient operator at the element level. In two-dimensions, this requires only 4 floating point values per element for C_x^e and C_y^e.

The computation of the element level gradient operators in three-dimensions is somewhat more involved. To begin, the element-local nodal coordinates in the referential domain are defined as follows:

$$\begin{aligned} \xi^T &= [-1, 1, -1, 1, 1, -1, 1, -1] \\ \eta^T &= [-1, 1, 1, -1, -1, 1, 1, -1] \\ \zeta^T &= [-1, -1, -1, -1, 1, 1, 1, 1]. \end{aligned} \tag{3.22}$$

The Jacobian, evaluated at the element centroid, or central Gauss point, is defined in terms of the element-local nodal coordinates $(\mathbf{x}_e, \mathbf{y}_e, \mathbf{z}_e)$ and the referential coordinates, (ξ, η, ζ) in Eq. (3.23).

$$J(0) = \frac{1}{8}\begin{bmatrix} \xi^T\mathbf{x}^e & \xi^T\mathbf{y}^e & \xi^T\mathbf{z}^e \\ \eta^T\mathbf{x}^e & \eta^T\mathbf{y}^e & \eta^T\mathbf{z}^e \\ \zeta^T\mathbf{x}^e & \zeta^T\mathbf{y}^e & \zeta^T\mathbf{z}^e \end{bmatrix} \tag{3.23}$$

The element volume is simply the determinant of the Jacobian in Eq. (3.24). Note that the element volume in three-dimensions may only be computed exactly with single-point integration for elements that are bricks or parallelepipeds.

$$V^e = det(J(0)) \tag{3.24}$$

The computation of the element level gradient operators proceeds by first evaluating the co-factors of the Jacobian, the inverse Jacobian in Eq. (3.25), and the gradient operators as shown in Eq. (3.26). Again, only the unique gradient operators must be stored, i.e., 12 words of storage per element are required for C_x^e, C_y^e, and C_z^e.[28]

$$\mathbf{D} = [\mathbf{D}_{ij}] = J(0)^{-1} \tag{3.25}$$

$$C_x^e = \frac{1}{8}[\mathbf{D}_{11}\xi + \mathbf{D}_{12}\eta + \mathbf{D}_{13}\zeta]$$

$$\begin{aligned} C^e_y &= \frac{1}{8}[\mathbf{D}_{21}\xi + \mathbf{D}_{22}\eta + \mathbf{D}_{23}\zeta] \\ C^e_z &= \frac{1}{8}[\mathbf{D}_{31}\xi + \mathbf{D}_{32}\eta + \mathbf{D}_{33}\zeta] \end{aligned} \tag{3.26}$$

The computation of the element level mass matrix using one-point quadrature and row-sum lumping yields the operator for 2-D in Eq. (3.27) and the operator for 3-D in Eq. (3.28).

$$M^e_{ab} = \delta_{ab}\frac{A^e}{4} \tag{3.27}$$

$$M^e_{ab} = \delta_{ab}\frac{V^e}{8} \tag{3.28}$$

The direct evaluation of the advection operator in Eq. (3.18) requires an integral of triple products that is very computationally intensive. Therefore, the advection operator is approximated using an *ad-hoc* modification known as the centroid advection velocity. This modification assumes that $\mathbf{u}$ in Eq. (3.18) may be approximated by

$$\overline{\mathbf{u}} = \sum_{a=1}^{Nnpe} N_a(\mathbf{0})\mathbf{u}_a \tag{3.29}$$

where $N_a(\mathbf{0})$ indicates evaluation of the shape functions at the origin of the referential coordinate system.

The application of single-point integration further simplifies the advection operator. In two-dimensions, α_1 and α_2 in Eq. (3.30) are used to form the advection operators in Eq. (3.31) where $u_{ab} = u_a - u_b$.

$$\begin{aligned} \alpha_1 &= \overline{u}C_{x1} + \overline{v}C_{y1} \\ \alpha_2 &= \overline{u}C_{x2} + \overline{v}C_{y2} \end{aligned} \tag{3.30}$$

$$\begin{aligned} A^e(\overline{\mathbf{u}})u^e &= [1,1,1,1]^T\frac{A^e}{4}[\alpha_1 u_{13} + \alpha_2 u_{24}] \\ A^e(\overline{\mathbf{u}})v^e &= [1,1,1,1]^T\frac{A^e}{4}[\alpha_1 v_{13} + \alpha_2 v_{24}] \end{aligned} \tag{3.31}$$

For the evaluation of the three-dimensional advection operators, $\beta_1 - \beta_4$ are defined as

$$\begin{aligned} \beta_1 &= \overline{u}C_{x1} + \overline{v}C_{y1} + \overline{w}C_{z1} \\ \beta_2 &= \overline{u}C_{x2} + \overline{v}C_{y2} + \overline{w}C_{z2} \\ \beta_3 &= \overline{u}C_{x3} + \overline{v}C_{y3} + \overline{w}C_{z3} \\ \beta_4 &= \overline{u}C_{x4} + \overline{v}C_{y4} + \overline{w}C_{z4}, \end{aligned} \tag{3.32}$$

and used in the element-level advection operators.

$$
\begin{aligned}
A^e(\overline{\mathbf{u}})u^e &= [1,1,1,1,1,1,1,1]^T \frac{V^e}{8}[\beta_1 u_{17} + \beta_2 u_{28} + \beta_3 u_{35} + \beta_4 u_{46}] \\
A^e(\overline{\mathbf{u}})v^e &= [1,1,1,1,1,1,1,1]^T \frac{V^e}{8}[\beta_1 v_{17} + \beta_2 v_{28} + \beta_3 v_{35} + \beta_4 v_{46}] \quad (3.33) \\
A^e(\overline{\mathbf{u}})w^e &= [1,1,1,1,1,1,1,1]^T \frac{V^e}{8}[\beta_1 w_{17} + \beta_2 w_{28} + \beta_3 w_{35} + \beta_4 w_{46}]
\end{aligned}
$$

The single-point diffusion operator may be stated simply as

$$K^e_{ab} = C_{i_a}\ \hat{\mu}_{ij} C_{j_b} V^e \tag{3.34}$$

where $\hat{\mu}_{ij}$ represents a tensorial viscosity. Here, i and j range from 1 to the number of spatial dimensions while a, b range from 1 to the number of nodes per element. Generation of the diffusion operator in Eq. (3.34) using single point integration leads to rank deficiency of the element level operator. The presence of an improper singular mode in the element level operator may also lead to singularity in the assembled global operator. In two-dimensions, there is only one improper singular mode, while in three-dimensions, there are four singular modes. These modes are commonly referred to as *hourglass* modes, and when excited in a numerical solution, they can remain undamped and pollute the field solution.

A detailed discussion of the hourglass stabilization methods used in GILA is beyond the scope of this chapter. However, for the sake of completeness, a brief overview of the so-called *h-stabilization* is presented. The term, h-stabilization, derives from the fact that the outer product of the element hourglass vectors is used to form the stabilization operator.

In two-dimensions, the single hourglass mode is

$$\Gamma^T = [1, -1, 1, -1]. \tag{3.35}$$

This mode, when excited, may be detected visually as a *w-mode* in isometric plots of the field variable from a numerical solution. Following Goudreau and Hallquist[23] and Gresho, et al.,[28] the element level stabilization operator is formed using the outer-product of the hourglass vector.

$$H^e_{ab} = \epsilon_{hg} \mu A^e \Gamma^T_a \Gamma_b \tag{3.36}$$

ϵ_{hg} is a non-dimensional parameter that, in practice, is unity by default for outer-product stabilization (see Gresho, et al.[28]).

In three-dimensions, the four hourglass vectors are

$$\Gamma^T_1 \ = \ [1, 1, -1, -1, -1, -1, 1, 1]$$

$$\begin{aligned} \Gamma_2^T &= [1,-1,-1,1,-1,1,1,-1] \\ \Gamma_3^T &= [1,-1,1,-1,1,-1,1,-1] \\ \Gamma_4^T &= [-1,1,-1,1,1,-1,1,-1]. \end{aligned} \tag{3.37}$$

The resulting 3-D hourglass stabilization operator is

$$H_{ab}^e = \epsilon_{hg}^e \mu [\Gamma_1 \ \Gamma_2 \ \Gamma_3 \ \Gamma_4] \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & C_3 & \\ & & & C_4 \end{bmatrix} \begin{Bmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \Gamma_4 \end{Bmatrix} \tag{3.38}$$

where $\epsilon_{hg}^e = 1.0$, $C_1 = C_2 = C_3 = C_4 = \overline{h}\sqrt[3]{V^e}$, $\overline{h} = (\sqrt[3]{V_{max}} - \sqrt[3]{V_{min}})/2$.

For the reduced integration element, γ-stabilization, see Belytschko, et al,[8, 48, 49] has also been investigated. γ-stabilization refers to the γ-vectors constructed from the hourglass modes for stabilization. While γ-stabilization is perhaps more robust than h-stabilization, this type of hourglass control also requires more operations and storage. It is the author's experience that it is relatively more difficult to excite the hourglass modes in an Eulerian computation than in a Lagrangian computation, e.g., a DYNA3D[66] simulation. However, γ-stabilization still requires fewer operations and less storage than the fully integrated two-dimensional bilinear element.

In three-dimensions, this is not the case. Table 3.1 shows the memory requirements and operations counts for a matrix-vector multiply (Ku) for a variety of element formulations. In 2-D, γ-stabilization requires nearly the same storage as the fully integrated element stored in either a compact, symmetric, element form, or in a global row-compressed form. However, this element requires 9 more operations to achieve the matrix-vector multiply when compared to the element-by-element matrix vector multiply with 2x2 quadrature. In 3-D, γ-stabilization is about 3 times more expensive to perform than the corresponding global row-compressed matrix-vector multiply.

There is one final modification to the finite element formulation that derives from the explicit treatment of the advective terms. For advection dominated flows, it is well known that the use of a backward-Euler treatment of the advective terms introduces excessive diffusion. Similarly, Gresho, et al.[28] have shown that forward-Euler treatment of the advective terms results in negative diffusivity, or an under-diffusive scheme. In order to remedy this problem, balancing-tensor diffusivity (BTD), derived from a Taylor series analysis to exactly balance the diffusivity deficit, is adopted. In the one-point quadrature element, the BTD term is simply added to the kinematic viscosity in Eq. (3.39) to form the tensorial diffusivity used in Eq. (3.34).

$$\hat{\mu}_{ij} = \mu_{ij} + \rho \frac{\Delta t}{2} u_i u_j \tag{3.39}$$

Table 3.1: Memory requirements and operations counts for a matrix-vector multiply for various element integration rules, stabilization operators, and storage schemes. (Nel: number of elements, Nnp: number of nodes, EBE: element-by-element)

Dimension	Element Formulation	Storage	$+$	$*$	Total Op.
2-D	1-pt., EBE	$4Nel$	$12Nel$	$16Nel$	$28Nel$
2-D	1-pt., h-stabilization, EBE	$4Nel$	$22Nel$	$18Nel$	$40Nel$
2-D	1-pt., γ-stabilization, EBE	$9Nel$	$19Nel$	$25Nel$	$44Nel$
2-D	2x2 Quadrature, EBE	$10Nel$	$16Nel$	$16Nel$	$32Nel$
2-D	2x2, ITPACKV[46]	$9Nnp$	$8Nnp$	$9Nnp$	$17Nnp$
3-D	1-pt., EBE	$12Nel$	$29Nel$	$28Nel$	$57Nel$
3-D	1-pt., h-stabilization, EBE	$12Nel$	$69Nel$	$46Nel$	$115Nel$
3-D	1-pt., γ-stabilization, EBE	$45Nel$	$61Nel$	$100Nel$	$161Nel$
3-D	2x2x2 Quadrature, EBE	$36Nel$	$64Nel$	$64Nel$	$128Nel$
3-D	2x2x2, ITPACKV[46]	$27Nnp$	$26Nnp$	$27Nnp$	$53Nnp$

In summary, the modifications made to the standard finite element formulation include the use of single-point integration, a row-sum lumped mass matrix, hourglass stabilization, and balancing tensor diffusivity. The benefits promised by one-point integration are tremendous in computational fluid dynamics problems because of the requisite mesh sizes for interesting problems and the concomitant memory requirements.

The reduction from 8 quadrature points to 1 in three dimensions reduces the computational load by a factor of about 6 to 7 and reduces memory requirements by over a factor of 2 for the basic gradient operators. Neglecting the storage costs associated with the PPE, the total storage requirements for the explicit algorithm is 60 words per 3-D element. Further, it has been demonstrated that the convergence rate of the one-point elements is comparable to the fully integrated elements at a fraction of the computational cost, see Liu.[49] With the element formulation defined, attention is now turned to the parallel aspects of the explicit algorithm when a domain-decomposition message-passing paradigm is used.

3.2 Domain-Decomposition/Message-Passing

This section describes the domain-decomposition message-passing (DDMP) implementation of the explicit time integration algorithm for the the incompressible Navier-Stokes. A brief overview of domain-decomposition and the associated sub-domain to processor mapping is discussed first. Next, the parallel right-hand-side assembly procedure is presented. The parallel assembly procedure then sets the stage for a discussion of the parallel iterative solution methods applied to the PPE.

3.2.1 Domain-Decomposition

Domain-decomposition is the process of sub-dividing the spatial domain into sub-domains that can be assigned to individual processors for the subsequent solution process. There has been a great deal of work on the problem of spatial decomposition for unstructured grids over the last several years.[6, 20, 39–42, 53, 60] In general, the requirements for mesh decomposition is that the sub-domains be generated in such a way that the computational load is uniformly distributed across the available processors, and that the inter-processor communication is minimized.

In GILA, the decomposition of the finite element mesh into sub-domains is accomplished using the multilevel graph partitioning tools available in CHACO.[39–42] This type of spectral domain decomposition attempts to subdivide the computational domain in such a way that the computational load is uniform across processors while attempting to minimize the inter-processor communication. In part, CHACO was selected for this task because of its implicit weighting on the number of wires in a hypercube when using the spectral bisection and octasection methods.

In order to exploit the finite element assembly process[44] for parallelization, the dual graph of the finite element mesh, i.e., the connectivity of the dual grid shown in Figure 3.1, is used to perform a non-overlapping element-based domain decomposition. Note that the dual grid corresponds to the grid associated with the element centered pressure variables in the $Q1Q0$ element. Implicit in this choice of a domain decomposition strategy is the idea that elements are uniquely assigned to processors while the nodes at the sub-domain interfaces are stored redundantly in multiple processors. Figure 3.3 illustrates a non-overlapping domain decomposition for a simple two-dimensional vortex-shedding mesh partitioned for four processors. After obtaining the domain decomposition, the assignment of nodes, boundary conditions, and materials to individual processors is performed internally before parallel execution begins.

3.2.2 Parallel Assembly via Message Passing

The finite element assembly procedure is an integral part of any finite element code and consists of a global gather operation of nodal quantities, an add

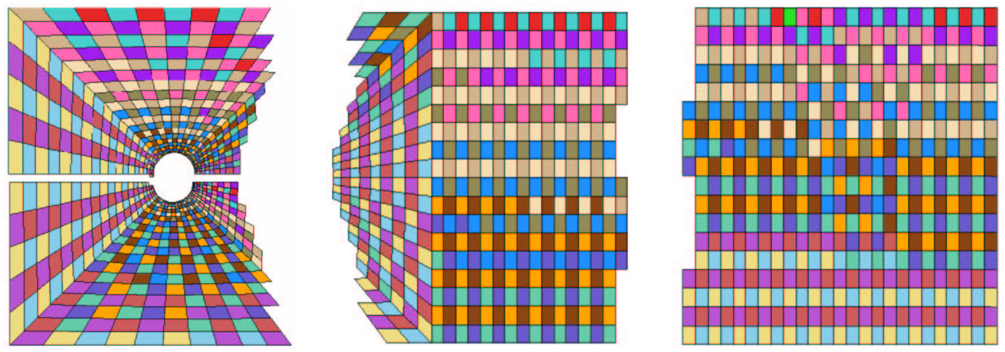


Figure 3.3: Four processor spatial domain decomposition for a two-dimensional vortex-shedding mesh.

operation, and a subsequent scatter back to the global memory locations. A complete description of the sequential assembly algorithm may be found in Hughes.[44] The assembly procedure is used to both form global coefficient matrices, right-hand-side vectors, and matrix-vector products in an element-by-element sense. In the case of the explicit time integration algorithm, the emphasis is upon the assembly of the element level contributions to the global right-hand-side vector

$$\hat{\mathbf{F}} = \mathcal{A}_{e=1}^{Nel}\{\mathbf{f}_e^n - K_e\mathbf{u}_e^n - A_e(\overline{\mathbf{u}})\mathbf{u}_e^n\} \tag{3.40}$$

where $\mathcal{A}$ is the assembly operator. Here, the diffusive and advective contributions are computed at the element-level using un-rolled right-to-left matrix-vector multiplies.

In order to exploit both register-to-register vector and cache-based processors, data-independent groups of elements are identified. This not only permits the vectorization of the assembly process, but also maximizes the number of element level operations with respect to the number of load-store operations. For cache based architectures, this permits all of the element data in a data-independent group to be loaded into cache once for the element-level operations, e.g., right-to-left matrix-vector multiplies. Thus, the assembly process using the vector/cache blocks proceeds block-by-block with all element level operations for a block being performed before completing the "add-scatter" portion of the gather-add-scatter assembly procedure.

The parallel assembly procedure may be viewed as a generalized form of the finite element assembly algorithm. However, inter-processor communication is an inherent part of the parallel assembly. As an example of a two-processor assembly, consider the sequential mesh and the assignment of the global nodes to two processors as shown in Figure 3.4. In Figure 3.4b, the local node numbers are enclosed in brackets to the right of the global node

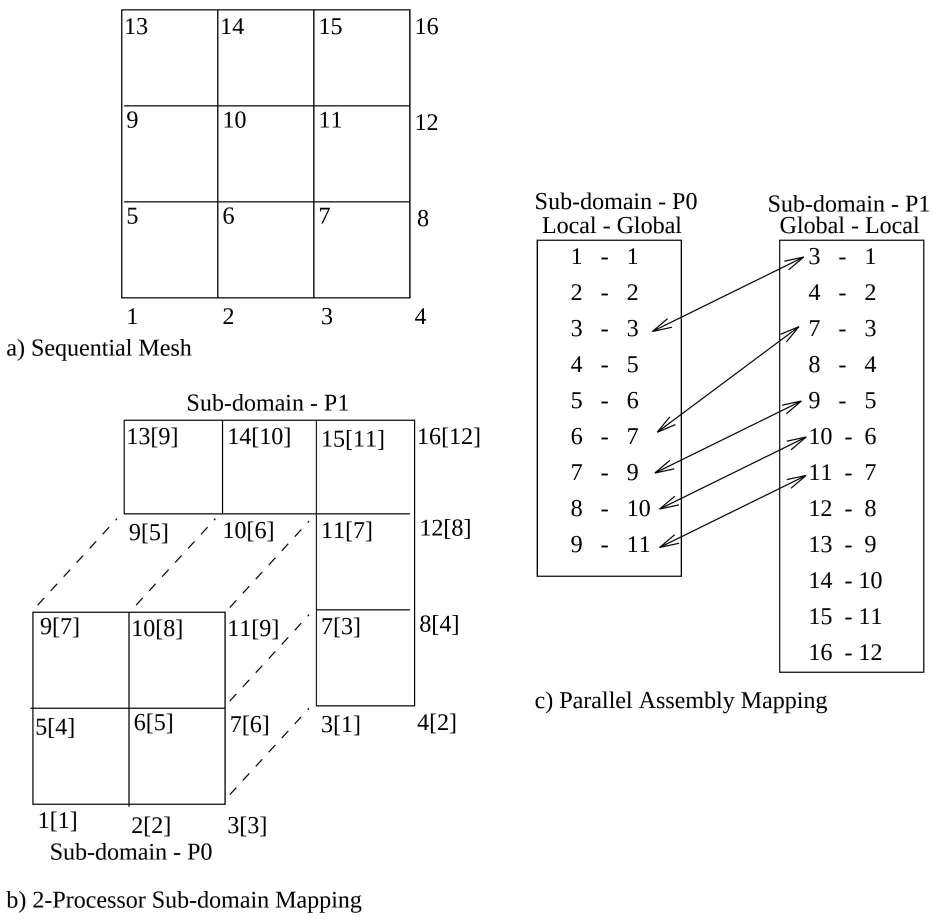


Figure 3.4: Parallel assembly procedure: a) Sequential Mesh, b) 2-Processor sub-domain mapping, c) Parallel assembly mapping.

number (global node 1 in sub-domain $P0$ is local node [1]). The sub-domain assignment of the global nodes is the consequence of the unique assignment of elements to processors, and reveals the existence of the nodes at the sub-domain boundaries on multiple processors.

Figure 3.4b shows the global inter-processor assembly of the sub-domain boundary nodes. Figure 3.4c illustrates the use of the local-to-global mapping required for the gather-add-scatter operation and the arrows between global node numbers identify a send-receive pair. Thus, the parallel assembly procedure induces communication in the form of a gather-send-receive-add-scatter process. The parallel right-hand-side assembly for Eq. (3.5) may be viewed as a generalized assembly with off-processor communication where first the on-process vector/cache blocked assembly is performed according to Eq. (3.41), and then the assembly at the sub-domain boundaries is performed according to Eq. (3.42).

$$\hat{\mathbf{F}}^p = \mathcal{A}_{e=1}^{Nel}\{\mathbf{f}_e^n - K_e\mathbf{u}_e^n - A_e(\overline{\mathbf{u}})\mathbf{u}_e^n\} \tag{3.41}$$

$$\hat{\mathbf{F}} = \mathcal{A}_{p=0}^{Np-1}\{\hat{\mathbf{F}}^p\} \tag{3.42}$$

There are several things to note about the parallel assembly procedure. First, the parallel assembly is simply a generalization of the sequential assembly procedure that includes inter-processor communication. Second, the algorithm only requires the communication of nodal data at the edges of adjacent sub-domains. Therefore, as the problem size increases, the communication overhead scales with number of surface nodes associated with sub-domain boundaries. Finally, this algorithm permits the implementation of vector-valued messages in order to avoid start-up issues associated with short messages, i.e., for the assembly shown in Eq. (3.42), the message length is proportional to the number of sub-domain boundary nodes, and the number of degrees-of-freedom per node. Finally, the use of non-overlapping grids implies that nodes in the finite element mesh that lie on sub-domain boundaries are stored in multiple processors as shown in Figure 3.4b. In contrast, over-lapping sub-domains would require the redundant storage of all the data associated with elements at the sub-domain boundaries.

3.2.3 The Parallel Explicit Algorithm

In this section, the issue of solving the PPE in parallel will not be addressed so that attention may be focused upon the solution process for the nodal variables. The DDMP version of the explicit time integration algorithm proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient, at time level n.

   i. All processors calculate the processor-local right-hand-side terms neglecting the pressure gradient according to Eq. (3.41).

   ii. Perform the inter-processor finite element assembly according to Eq. (3.42). In this step, processors that share a common sub-domain boundary exchange messages containing partially assembled right-hand side contributions, and accumulate the fully-summed terms at the sub-domain boundaries.

   iii. All processors compute the processor-local partial acceleration using a pre-calculated lumped mass matrix, i.e., the mass matrix has been fully summed for the nodes at the sub-domain boundaries.

$$\tilde{\mathbf{a}}^n = M_L^{-1}\hat{\mathbf{F}} \tag{3.43}$$

2. Solve the global PPE problem for the current pressure field.

$$[C^T M_L^{-1} C] P^n = C^T \tilde{\mathbf{a}}^n \tag{3.44}$$

3. Update the nodal velocities.

   i. The computation of the pressure gradient term in Eq. (3.45) consists of the parallel on-processor computation of the discrete pressure gradient, followed by an inter-processor assembly.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t[\tilde{\mathbf{a}}^n + M_L^{-1}\{\mathcal{A}_{p=0}^{Np-1}(CP^n)\}] \tag{3.45}$$

4. Repeat steps 1-3 until a maximum simulation time limit or maximum number of time steps is reached.

   Remark. For the explicit solution algorithm, the global row-sum lumped mass is accumulated at all nodes in the finite element mesh during the initialization phase. This requires the nodal assembly of the partial nodal mass at nodes on sub-domain boundaries as shown in Figure 3.4.

3.2.4 Communication Costs

This section outlines the communication costs associated with the explicit time integration algorithm. The communication costs may be broken down into the cost per time step for the momentum equations, and the cost per time step for solving the PPE. To begin, N_Γ is the number of nodes on the boundary of a sub-domain. In two-dimensions, the average number of nodes communicated to an adjacent processor is $N_\Gamma/4$, and in three-dimensions the average is $N_\Gamma/6$. Assuming that there are 8 bytes per floating point word, then the total number of bytes per processor to be communicated via a send is

$$N_c = 8 N_\Gamma N_{DOF}, \tag{3.46}$$

where N_{DOF} is the number of degrees-of-freedom per node.

The communication cost for a send operation may be broken into three parts: the time to initiate the message passing, $t_{startup}$, the cost per packet, t_{packet}, and the transmission time, $t_{transmit}$. Thus, the time to send a message is

$$t_{send} = t_{startup} + \frac{N_c}{N_{packet}} t_{packet} + N_c t_{transmit}, \tag{3.47}$$

where N_{packet} is the number of bytes per packet.

From Eq. (3.46) and (3.47), it is clear that the number of adjacent subdomains determines the total startup time for message passing. Next, the communication cost per time step is estimated for the explicit algorithm. For the momentum equations, there are two primary messaging steps. The first occurs during the distributed assembly of the right-hand-side in Eq. (3.41). The second occurs during the assembly of the pressure-gradient in Eq. (3.45). Thus, there are 2 vector-valued messaging steps. For the solution of the PPE, there is 1 vector-valued messaging step per iteration, where N_{IT} iterations are required in the conjugate gradient solution. From this, it is possible to estimate the communication cost per time step for the explicit algorithm on a per processor basis as

$$C_{step} = 8(2 + N_{IT})N_{DOF}N_{\Gamma}t_{send}. \tag{3.48}$$

Chapter 4

The Semi-Implicit Projection Method

The solution of the time-dependent incompressible Navier-Stokes equations poses several algorithmic issues due to the div-free constraint, and the concomitant spatial and temporal resolution required to perform time-accurate solutions particularly where complex geometry is involved. Although fully-coupled solution strategies are available, the cost of such methods is generally considered prohibitive for time-dependent simulations where high-resolution grids are required. The application of projection methods provides a computationally efficient alternative to fully-coupled solution methods.

A detailed review of projection methods is beyond the scope of this paper, but a partial list of relevant work is provided. Projection methods, also commonly referred to as fractional-step, pressure correction methods, or Chorin's method[10] have grown in popularity over the past 10 years due to the relative ease of implementation and computational performance. This is reflected by the volume of work published on the development of second-order accurate projection methods, see for example van Kan,[45] Bell, et al.,[7] Gresho, et al.,[24–27] Almgren, et al.,[1,2,4] Rider,[55–58] Minion,[51] Guermond and Quartapelle,[35] Puckett, et al.,[54] Sussman, et al.,[63] and Knio, et al.[47] The numerical performance of projection methods has been considered by Brown and Minion,[9,52] Wetton,[65] Guermond,[33,34] Guermond and Quartapelle,[36,37] and Almgren et al.[3]

As background, a brief review of Chorin's original projection method is presented before proceeding with the finite element form of the projection algorithm. The vector form of the momentum equations may be written as

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \nabla p = \mathcal{F}(\mathbf{u}), \tag{4.1}$$

where for a constant viscosity,

$$\mathcal{F}(\mathbf{u}) = \mathbf{f} + \mu \nabla^2 \mathbf{u} - \rho \mathbf{u} \cdot \nabla \mathbf{u}. \tag{4.2}$$

Now, $\mathcal{F}(\mathbf{u})$ may be decomposed into a div-free and curl-free part where the div-free part is

$$\nabla \cdot \left\{ \frac{\partial \mathbf{u}}{\partial t} \right\} = 0, \tag{4.3}$$

and the curl-free part is

$$\nabla \times \nabla p = 0. \tag{4.4}$$

Discretizing in space and time, the decomposition, neglecting the contribution of the pressure gradient, yields

$$\rho \frac{(\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n)}{\Delta t} = \mathcal{F}^h(\mathbf{u}), \tag{4.5}$$

where $\mathcal{F}^h(\mathbf{u})$ is the spatially discrete analogue of $\mathcal{F}$ in Eq. (4.2), and $\tilde{\mathbf{u}}^{n+1}$ is an approximate discrete velocity field at time $n+1$. Note that the discrete divergence of $\tilde{\mathbf{u}}^{n+1}$ is generally not zero, i.e. $G^T \tilde{\mathbf{u}}^{n+1} \neq 0$ where G^T is the discrete divergence operator. The functional dependence of $\mathcal{F}^h$ upon the discrete velocity, $\mathbf{u}$, depends upon whether the algorithm is implicit, explicit, or semi-implicit. However, the dependence on pressure, or rather pressure gradient, is explicit so that

$$\frac{(\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n)}{\Delta t} = \frac{(\mathbf{u}^{n+1} - \mathbf{u}^n)}{\Delta t} - \frac{1}{\rho} G p^{n+1}, \tag{4.6}$$

where G is the discrete gradient operator, and G^T is the discrete divergence operator. Applying the discrete divergence operator to Eq. (4.6) yields a Poisson equation for the pressure at time level $n+1$,

$$G^T \frac{1}{\rho} G p^{n+1} = \frac{1}{\Delta t} G^T \tilde{\mathbf{u}}^{n+1}. \tag{4.7}$$

By eliminating the velocity at time level n, Eq. (4.6) yields a relationship for the projected div-free velocity field.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \frac{1}{\rho} G p^{n+1}. \tag{4.8}$$

4.1 Projection Properties

The philosophy behind projection algorithms is to provide a legitimate way to decouple the pressure and velocity fields in the hope of providing an efficient computational method for transient, incompressible flow simulations. In practice, the action of the projection, $\mathcal{P}$, is to remove the part of the approximate velocity field that is not div-free, i.e., $\mathbf{u} = \mathcal{P}(\tilde{\mathbf{u}})$. In effect, the projection is achieved by decomposing the velocity field into div-free and curl-free components using a Helmholtz decomposition. The decomposition may be written as

$$\tilde{\mathbf{u}} = \mathbf{u} + \nabla\lambda, \tag{4.9}$$

where $\tilde{\mathbf{u}}$ is a non-solenoidal velocity field, $\mathbf{u}$ is its div-free counterpart, and $\nabla\lambda$ is the curl-free component, i.e., $\nabla \times \nabla\lambda = 0$.

Thus, given an approximate, non-solenoidal velocity field, $\tilde{\mathbf{u}}$, $\mathcal{F}(\tilde{\mathbf{u}})$ may be projected onto a divergence-free subspace such that

$$\rho\frac{\partial \mathbf{u}}{\partial t} = \mathcal{P}(\mathcal{F}(\tilde{\mathbf{u}})), \tag{4.10}$$

and

$$\nabla p = \mathcal{Q}(\mathcal{F}(\tilde{\mathbf{u}})). \tag{4.11}$$

Here, $\mathcal{P}$ and $\mathcal{Q}$ are the projection operators, and they have the following properties. $\mathcal{P}$ projects a velocity vector onto a div-free subspace, and $\mathcal{Q}$ projects a vector into a curl-free subspace. Both $\mathcal{P}$ and $\mathcal{Q}$ are idempotent, i.e., $\mathcal{P} = \mathcal{P}^2$ and $\mathcal{Q} = \mathcal{Q}^2$. Therefore, repeated application of the projection operators does not continue to modify the projected results. The projection operators are orthogonal, and commute, i.e., $\mathcal{P}\mathcal{Q} = \mathcal{Q}\mathcal{P} = 0$.

The explicit forms of the continuous projection operators are

$$\mathcal{P}(\cdot) = \left\{I - \nabla(\nabla^2)^{-1}\nabla\cdot\right\}(\cdot), \tag{4.12}$$

and

$$\mathcal{Q}(\cdot) = I - \mathcal{P} = \nabla(\nabla^2)^{-1}\nabla\cdot(\cdot). \tag{4.13}$$

It should be noted that $\mathcal{P}$, and $\mathcal{Q}$ have *built-in* all the appropriate physical boundary conditions. Further, The eigenvalues of $\mathcal{P}$ and $\mathcal{Q}$ are either 0 or 1 so that the projections are norm-reducing.

4.2 Time Integration Method

In GILA, the *optimal* Projection-2 (P2) method identified by Gresho[24] forms the starting point for a discussion of the finite element projection algorithms. Before proceeding with a description of the P2 algorithm, the semi-discrete Navier-Stokes equations are presented. The spatial discretization of the conservation equations is achieved using the $Q1Q0$ element with bilinear support for velocity and piecewise constant support for the pressure in two dimensions. In three dimensions, the velocity support is trilinear with piecewise constant support for pressure. The methods for obtaining the weak-form of the conservation equations are well known and will not be repeated here (see for example, Gresho, et al.,[30] Hughes,[44] and Zienkiewicz and Taylor[68]).

The spatially discrete form of momentum conservation Eq. (2.1) and the divergence constraint Eq. (2.7) are

$$M\dot{\mathbf{u}} + A(\mathbf{u})\mathbf{u} + K\mathbf{u} + MM_L^{-1}Cp = \mathbf{F}, \tag{4.14}$$

$$C^T\mathbf{u} = 0, \tag{4.15}$$

where M is the unit mass matrix, $A(\mathbf{u})$ and K are the advection and the viscous diffusion operators respectively, and $\mathbf{F}$ is the body force. C is the gradient operator, and C^T is the divergence operator, i.e., C and C^T are nearly the discrete finite element analogues of G and G^T discussed above. In order to simplify the nomenclature, $\mathbf{u}$ and p are understood to be discrete approximations to the continuous velocity and pressure fields.

Following the development of the projection method introduced above,

$$M\left(\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t}\right) = \mathcal{F}^h(\mathbf{u}, p), \tag{4.16}$$

where $\mathcal{F}^h(\mathbf{u}, p)$ may involve an explicit or implicit dependence upon $\mathbf{u}$ and is understood to be a discrete vector quantity. From this, it is clear that the intermediate velocity, $\tilde{\mathbf{u}}^{n+1}$, corresponds to an approximate velocity field that has not yet felt the influence of the current pressure field, and therefore is not necessarily solenoidal.

Incorporating the pressure gradient contribution, the approximation to $\mathcal{F}^h(\mathbf{u})$ is,

$$M\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t}\right) = M\left(\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t}\right) + MM_L^{-1}Cp^{n+1}. \tag{4.17}$$

Eliminating the velocity at time level n, yields the discrete statement of the Helmholtz decomposition with the concomitant div-free constraint, $C^T\mathbf{u}^{n+1} = 0$, viz.,

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{u}}^{n+1} \\ 0 \end{Bmatrix}. \tag{4.18}$$

Here, M_L is the row-sum lumped, i.e., diagonalized, mass matrix, and $\lambda = \Delta t(p^{n+1} - p^n)$

The system of equations in Eq. (4.18) is analogous to those introduced by Chorin.[10] Although Chorin suggested using a method of successive substitutions to obtain the velocity and pressure fields corresponding to the div-free state, the solution of Eq. (4.18) via a gradient-based iterative method and the solution of the Schur complement of Eq. (4.18) has proven more efficient.

The Schur complement of the *projection* operator may be formed explicitly from Eq. (4.18) yielding

$$[C^T M_L^{-1} C]\lambda = C^T \tilde{\mathbf{u}}^{n+1}, \tag{4.19}$$

where M_L is the diagonalized, i.e., row-sum lumped mass matrix and includes the prescription of essential velocity boundary conditions. Note that the term *projection* operator is used loosely here to describe Eq. (4.18) since its solution yields a div-free velocity field and the corresponding Lagrange multiplier. However, this operator is not to be confused with the continuous projection operators, $\mathcal{P}$ and $\mathcal{Q}$.

Eq. (4.18) is the consistent, discrete form of the elliptic operator for the projection algorithm. It represents an algebraic system of equations that is solved for the element-centered Lagrange multiplier during the time-marching procedure. Figure 3.1 shows the dual, staggered grid where λ and P are centered. The projection operator in Eq. (4.19) incorporates the effect of the essential velocity boundary conditions, and automatically builds in the boundary conditions from Eq. (2.15) and (2.17) – see Gresho, et al.[25]

The projection algorithm proceeds as follows. Given a div-free velocity, $\mathbf{u}^n$, and its corresponding pressure field, p^n, solve the momentum equations for an approximate velocity field at $n+1$.

1. Calculate the approximate velocity field $\tilde{\mathbf{u}}^{n+1}$

$$\begin{aligned}&[M + \Delta t \theta_K K]\tilde{\mathbf{u}}^{n+1} = \\ &\quad [M - \Delta t(1-\theta_K)K]\mathbf{u}^n + \\ &\quad \Delta t\{\theta_F \mathbf{F}^{n+1} + (1-\theta_F)\mathbf{F}^n - A\mathbf{u}^n - M M_L^{-1} C p^n\}.\end{aligned} \tag{4.20}$$

   Here, θ_K and θ_F control the time-weighting applied to the viscous and body force terms with $0 \le \theta_K, \theta_F \le 1$. For $\theta_K = 1$, the viscous terms are treated explicitly, while for $\theta_K = 0$, the viscous terms are treated implicitly, i.e., via backward-Euler, and for $\theta_K = 1/2$, the viscous treatment corresponds to a Crank-Nicolson integrator.

2. Given the approximate velocity, $\tilde{\mathbf{u}}^{n+1}$, solve Eq. (4.19) for λ.

3. Project the approximate velocity to a div-free subspace.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - M_L^{-1} C\lambda \tag{4.21}$$

4. After the velocity update, an updated pressure at time level $n+1$ is obtained via

$$p^{n+1} = p^n + \frac{\lambda}{\Delta t}. \tag{4.22}$$

Remark

In the projection algorithm, the start-up procedure to obtain the pressure, p^0, proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient, at time level n by solving the mass-matrix problem.

$$M\tilde{\mathbf{a}}^0 = \mathbf{F}^0 - K\mathbf{u}^0 - A(\mathbf{u})\mathbf{u}^0 \tag{4.23}$$

   where $\tilde{\mathbf{a}}^0$ is the instantaneous acceleration neglecting the pressure gradient.

2. Solve the global PPE for the current pressure field.

$$[C^T M_L^{-1} C]p^0 = C^T\tilde{\mathbf{a}}^0 \tag{4.24}$$

   In GILA, the prescribed initial conditions and boundary conditions are tested and, if necessary, a projection to a div-free subspace is performed on the initial velocity field, $\mathbf{u}^0$. This guarantees that the flow problem is well-posed, even if the user prescribed initial conditions violate the conditions of Eq. (2.25) – (2.26).

   In practice, the criterion for performing a div-free projection is based upon the RMS divergence error

$$\sqrt{\frac{(C^T\mathbf{u})\cdot(C^T\mathbf{u})}{Nel}} \leq \epsilon \tag{4.25}$$

   where Nel is the number of elements and ϵ is a user-specified tolerance typically 10^{-10} to 10^{-7}. If the RMS divergence error is greater than the specified tolerance for the initial candidate velocity field, $\tilde{\mathbf{u}}^0$, then the PPE problem in Eq. (4.26) is solved for λ, and a mass-consistent projection performed using Eq. (4.27).

$$[C^T M_L^{-1} C]\lambda = C^T\tilde{\mathbf{u}}^0 \tag{4.26}$$

$$\mathbf{u}^0 = \tilde{\mathbf{u}}^0 - M_L^{-1} C\lambda \tag{4.27}$$

The semi-implicit projection algorithm must respect a relaxed convective stability limit where $CFL \leq O(5-10)$ as described in Gresho and Chan.[25] Because of the semi-implicit treatment of viscous terms, there is no diffusive stability constraint. Computational experiments have demonstrated that $CFL \leq 5$ is a reasonable tradeoff between accuracy and computational cost.

In an unstructured grid, with variable element size, the calculation of the grid Re (Reynolds) and CFL (Courant-Freidrichs-Levy) numbers uses the element-local coordinates and centroid velocities as described in Chapter 3.

The second-order semi-implicit projection method uses one additional modification to the finite element formulation that derives from the explicit treatment of the advective terms. For advection dominated flows, it is well known that the use of a backward-Euler treatment of the advective terms introduces excessive diffusion. Similarly, Gresho, et al.[28] have shown that forward-Euler treatment of the advective terms results in negative diffusivity, or an under-diffusive scheme. In order to remedy this problem, balancing-tensor diffusivity (BTD), derived from a Taylor series analysis to exactly balance the diffusivity deficit, is adopted. In the one-point quadrature element, the BTD term is simply added to the kinematic viscosity in Eq. (4.28) to form the tensorial diffusivity.

$$\hat{\mu}_{ij} = \mu_{ij} + \rho \frac{\Delta t}{2} u_i u_j \tag{4.28}$$

Chapter 5

Boundary Conditions and Source Terms

This chapter summarizes the boundary conditions that are available in GILA. The keywords that may be used to prescribe boundary conditions are summarized in Chapter 7.

5.1 Node and Side Sets

Sets provide a generalized concept for grouping nodes and element faces (or sides). In GILA, node sets and side sets provide the basic entities for the prescription of boundary conditions. Node sets consist of an arbitrary list of nodes that are treated as one entity for the application of nodal boundary conditions. Side sets consist of an arbitrary list of quadrilateral edges in two dimensions and a list of hexahedral faces in three dimensions. Figure 5.1 shows a node set associated with inflow conditions and a side set associated with traction boundary conditions for an external flow problem.

5.2 Nodal Boundary Conditions

The prescription of nodal boundary conditions in GILA encompasses all nodal degrees-of-freedom, e.g., velocity, temperature, mass concentration species, and turbulent kinetic energy. These boundary conditions are typically referred to as essential or Dirichlet boundary conditions and fix the nodal values of the field variable according to a prescribed value or function of time.

As an example of nodal boundary conditions, consider the prescription of boundary conditions based on the flow domain shown in Figure 5.1. Inflow boundary velocity conditions that emulate free-stream conditions are

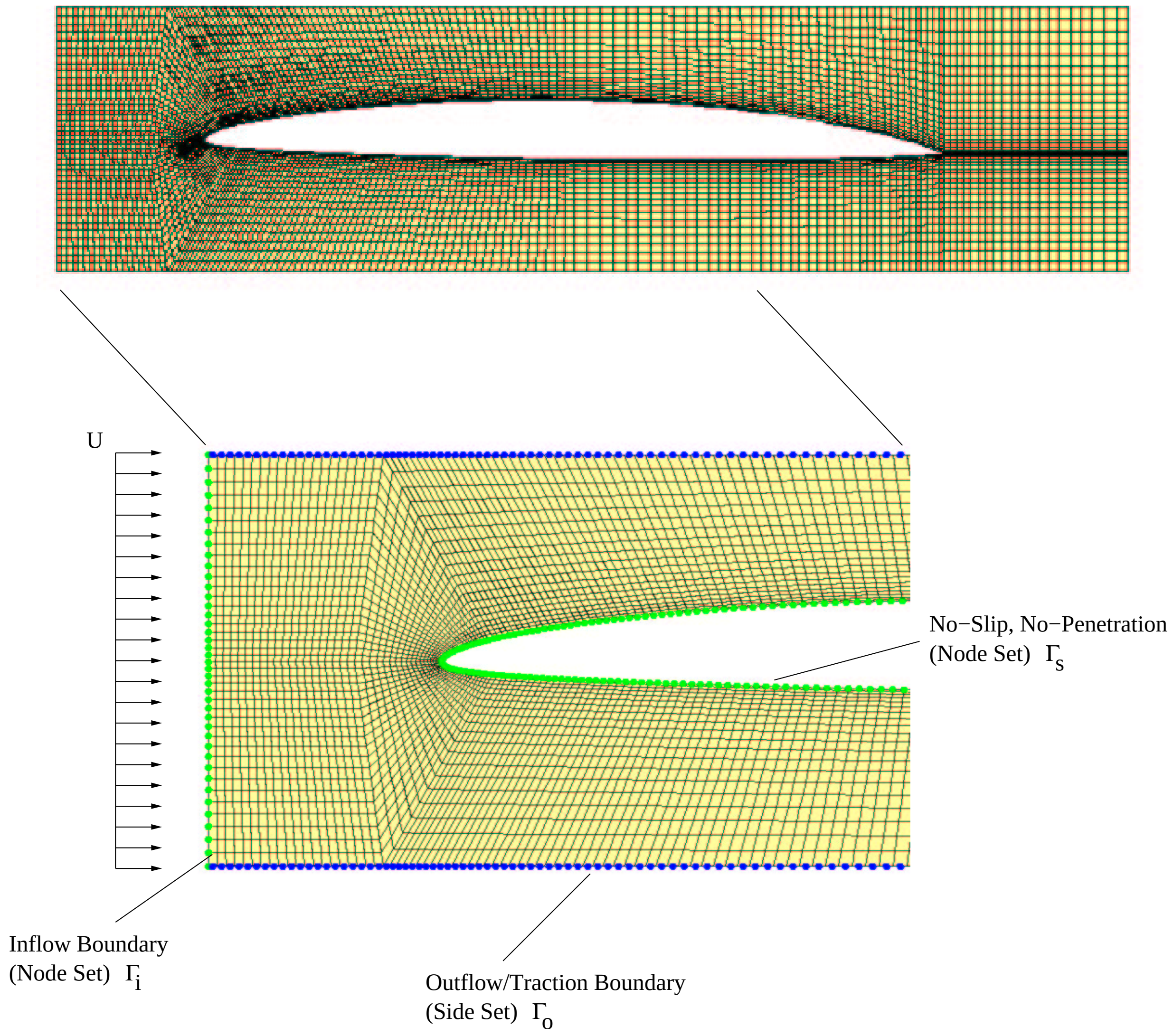


Figure 5.1: Flow domain with inflow Γ_i, outflow Γ_o, and no-slip, no-penetration Γ_s boundaries.

prescribed as $u_1 = U$ and $u_2 = 0$ on Γ_i. No-slip and no-penetration velocity boundary conditions are prescribed at the cylinder wall, $(u, v) = (0, 0)$ on Γ_s.

5.3 Traction Boundary Conditions

The formulation of the weak-form of the momentum equations Eq. (2.1) yields traction boundary conditions also known as natural boundary conditions. In terms of the shape functions N_a, the traction boundary conditions may be computed as

$$F^e_{i_a} = \int^e_\Gamma N_a \; \hat{f}_i(x_i, t) \; d\Gamma, \tag{5.1}$$

where $\hat{f}_i$ are the components of the prescribed traction. Here, $1 \leq i \leq Ndim$ and $1 \leq a \leq Nnpe$ where $Ndim$ is the number of space dimensions and $Nnpe$ is the number of nodes per element.

Alternatively, the boundary conditions may be written in terms of the stress,

$$F^e_{i_a} = \int^e_\Gamma N_a \; \hat{\sigma}_{ij} n_j \; d\Gamma, \tag{5.2}$$

where $\hat{\sigma}_{ij}$ is the prescribed stress and n_j is the outward normal for the domain boundary.

In terms of the pressure and strain-rate, the traction boundary conditions are

$$F^e_{i_a} = \int^e_\Gamma N_a \; \{-\hat{p}\delta_{ij} + 2\mu \; \hat{\varepsilon}_{ij}\} \, n_j. \tag{5.3}$$

By default, homogenous traction boundary conditions are applied unless other boundary conditions are prescribed, i.e., homogeneous traction (natural) boundary conditions are the *do-nothing* boundary conditions. The traction and velocity boundary conditions can be mixed. In a two-dimensional sense, mixed boundary conditions can consist of a prescribed normal traction and a tangential velocity. For example, at the outflow boundary in Figure 2.1, a homogeneous normal traction and vertical velocity on Γ_2 constitutes a valid set of mixed boundary conditions. A detailed discussion of boundary conditions for the incompressible Navier-Stokes equations may be found in Gresho and Sani.[32] Note that this boundary condition provides an important component for the coupling between fluid and structural components for coupled problems and is computed internally for this class of problems.

5.3.1 Pressure Boundary Conditions

In many practical situations, the viscous contributions in Eq. (5.3) may be neglected. This is typically the case in situations where the viscosity is small, i.e., the Reynolds number is relatively large. In this case, the viscous terms are ignored leaving only the pressure contribution to the traction boundary conditions,

$$F^e_{i_a} = \int_\Gamma^e N_a \ \{-\hat{p}\}\, n_i, \tag{5.4}$$

where $\hat{p}$ is the prescribed pressure.

Due to the common occurrence of this type of limiting traction boundary condition, GILA refers to this as a *pressure boundary condition* which reflects its use in finite element models. The **pbc** $-$ **end** keyword block is used to prescribe the pressure as a function of time, $\hat{p}(t)$ on a boundary side set.

5.4 Outflow Boundary Conditions

In many problems, it is necessary to truncate the physical domain resulting in an artificial outflow boundary. In the presence of strong vortical structures the homogeneous natural (*do-nothing*) boundary conditions can result in a global pressure response that is physically unrealistic. Heinrich, et al.[38] present a summary of this problem and suggest several remedies.

In GILA, *outflow* boundary conditions that are similar to the pressure boundary condition in Eq. (5.4) are used. However, the outflow boundary conditions use a known pressure distribution that has been computed during the solution procedure. This boundary condition results in an equilibrating force applied on the outflow boundary,

$$F^e_{i_a} = \int_\Gamma^e N_a \ \{-p^n\}\, n_i, \tag{5.5}$$

where p^n is the pressure field computed at time t^n.

The outflow boundary condition is prescribed using the **outflow** $-$ **end** keyword block which may be used to apply the outflow boundary conditions on a list of side sets.

5.5 Body Forces

The presence of body forces in the momentum equations results in element-level force contributions. At the element-level, the forces are

$$F^e_{i_a} = \int_\Omega N_a \ \rho f_i \ d\Omega, \tag{5.6}$$

where f_i represents the body-force per unit volume.

In thermal-convection problems where a Boussinesq fluid is appropriate, the body-force due to buoyancy is

$$F^e_{i_a} = \int_\Omega N_a \ \rho g_i \ \beta(T - T_{ref}) \ d\Omega. \tag{5.7}$$

where, g_i is the acceleration due to gravity, β is the coefficient of thermal expansion, and T_{ref} is a reference temperature. The specification of the gravity vector, reference temperature and fluid properties is achieved with the **material − end** keyword block.

Remark

> By default, the fluid temperature is initialized to the reference temperature T_{ref} specified in the **material − end** keyword block. Each fluid is initialized material-by-material resulting in *built-in* temperature initial conditions.

5.6 Flux Boundary Conditions

The heat flux rate at a boundary is prescribed as

$$Q_a^e = \int_\Gamma N_a \; \hat{q}_i(x_i, t) \; n_i \; d\Gamma \tag{5.8}$$

where $\hat{q}_i$ is the known flux rate through the boundary with normal n_i. Using the constitutive relationship between temperature and heat-flux is

$$\hat{q}_i(x_i, t) = -k_{ij} \frac{\partial T}{\partial x_j}, \tag{5.9}$$

where k_{ij} is the thermal conductivity tensor. The homogeneous form of the heat-flux boundary condition is the default and represents a perfectly insulated, i.e., zero heat-flux boundary.

The heat flux rate may also be prescribed in terms of a convective heat transfer coefficient,

$$\hat{q}_i(x_i, t) \; n_i = h(T - T_\infty), \tag{5.10}$$

where h is the heat transfer coefficient, and T_∞ is a reference temperature.

For the species transport equations, the prescribed mass flux rate for species-1 is

$$\dot{m}_a^e = \int_\Gamma N_a \; \hat{J}_{1_i} \; n_i \; d\Gamma, \tag{5.11}$$

where $\hat{J}_{1_i}$ is the known flux rate through the boundary with normal n_i. (For simplicity in the presentation, only the boundary conditions for species-1 are presented here.) Using the constitutive relation between species concentration and mass-flux rate,

$$\hat{J}_1(x_i, t) = -\rho \mathcal{D}_{1_{ij}} \frac{\partial Z_1}{\partial x_j}, \tag{5.12}$$

where $\mathcal{D}_{1_{ij}}$ is the tensorial mass diffusivity. The homogeneous form of the mass-flux boundary condition is the default and represents a boundary where the gradient of the species in the boundary normal direction is zero.

The prescribed flux rate may also be specified in terms of a convective mass transfer coefficient as

$$\hat{J}_{1_i}\ n_i = h_{\mathcal{D}\infty}(Z_1 - Z_{1_\infty}), \tag{5.13}$$

where $h_{\mathcal{D}_\infty}$ is the mass transfer coefficient and Z_{1_∞} is a reference species concentration.

5.7 Pressure Levels

In incompressible flow, the pressure is typically only known up to an additive constant – the hydrostatic pressure level. The prescription of a specific hydrostatic pressure level is achieved through the **ppebc − end** keyword block. For the $Q1Q0$ element technology, setting the hydrostatic pressure level requires a shell-set in 2-D or a solid-set in 3-D to identify those elements for which the pressure level will be fixed.

Chapter 6

Running GILA

GILA has been developed to permit rapid configuration making it adaptable to many computer architectures. GILA has been exercised on computers ranging from SUN, SGI and Linux workstations to networks of workstations, CRAY vector supercomputers, the Meiko CS-2, the Intel Pargon, Sandia's CPLANT and ASCI TFLOPs machines. The common thread for all of these machines is a UNIX (or UNIX like) environment. Thus, GILA provides a single command line interface which functions in the fashion which most common UNIX commands operate, i.e, a single command followed by a list of command line arguments.

6.1 Execution

GILA may be executed with the following command line options:

```
gila -i mesh -c cntl -o out -s plot -h hist
     -g glob -d dump -r restart -x part
```

GILA Command Line Arguments

Keyword	**Meaning**
-i *mesh*	GILA mesh file
-c *cntl*	GILA control file
-o *out*	Human readable output file (*default*: **out**)
-s *plot*	binary state database for graphics (*default*: **plot**)
-h *hist*	time history database (*default*: **hist**)
-g *glob*	ASCII global time history data (*default*: **glob**)
-d *dump*	Check-point file for writing restarts (*default*: **dump**)
-r *restart*	Check-point file for reading restarts (*no default*)
-x *part*	Parallel partition file (*default*: **part**)

All of the file names may include a path name as well. For example, the following command line makes use of the automatic expansion of the user's login directory and both absolute and relative paths for files.

```
gila -i /scr/plate/flow1.msh -c../cntl1 -o/home/joe/plate.out
```

There are several notable exceptions where file names may not include a path. GILA relies upon the "Sacket" I/O library for the generation of the GRIZ and THUG data files. For the state plot file names (-s plot) and the time history file names (-h hist), it is assumed that a six character root file name will be used. As the familied graphics files are generated during a simulation, the root file name and subsequent family members will be written to disk with the family member file names being the root file name concatenated with a two digit family number. Hopefully, future graphical data file formats will not be so limited.

6.2 Restarts

GILA will write a binary check-point file which contains all of the data necessary to restart a computation at intervals specified in the control file. An existing check-point file can be used to restart gila using the following command-line syntax.

```
gila -i flow1.msh -c cntl1 -r old_dump -d new_dump
```

Note that the **dump** keyword must be used in the **analyze** – **end** keyword block to activate restarts (see Chapter 7).

The state and time history plot files are preserved when a restart is performed. Similarly, the global output data is simply concatenated to the existing *glob* file when a restart is performed. However, the human output file is not preserved, i.e., it is over-written when a restart is performed unless a different output file is specified.

GILA will permit the user to change only a limited number of analysis parameters when a check-point file is used to restart a computation. For example, changing mesh parameters such as the number of nodes and elements is not possible. However, changing material properties, the number of time steps, plot intervals, etc. is acceptable.

Chapter 7

GILA Control File

The necessary input data for a GILA simulation is split into multiple files. The first file, the control file, contains all of the control information for the problem, e.g., analysis type, solver options, fluid properties, etc. The mesh file contains the nodal spatial coordinates, connectivity, node-set and side-set data, etc.

In the ensuing description, **bold** text denotes keywords, while *italic* text identifies keyword parameters or optional data in the control file. Primary sections of the control file are delimited by a **keyword** – **end** sequence that may contain a series of **keyword** – *parameter* sequences. The presence of a keyword and a parameter implies that the parameter is expected as input. Where possible, default values have been identified in order to minimize the number of keywords that are necessary in an input file.

Every attempt has been made to eliminate order-dependence in the control file, however it is necessary for the **analyze** − **end** block to occur before any boundary condition blocks. Comments in the control file must be preceded by a "$" symbol or a "#" symbol, or may be enclosed in a pair of braces "{ }". All input in the control file is case insensitive. Figure 7.1 shows the typical format of a GILA control file.

7.1 Analysis Title (title)

title
80-character analysis title

The analysis title may be specified in the input file using the **title** keyword. This keyword assumes that the following line of the input file contains an 80-character title. Comment characters are ignored in the title character string. The title character string is echoed to the screen and to the *out* file at execution time with the 80-character mesh title. An example of how the

```
title
Analysis Title {80 characters or less}

# starts a comment line

$ starts a comment line

{ Comments may be enclosed in braces as well }

# The analyze-end block describes the analysis parameters
analyze
...
end

# The material-end block defines material properties
material 1
...
end

# The turb-end block defines the turbulence model
turb
...
end

# The momsol-end block defines the momentum equation solver
momsol
...
end

# The ppesol-end block defines the PPE equation solver
ppesol
...
end

# The ndhist-end block defines the time-history nodes
ndhist
...
end

end
```

Figure 7.1: A Sample GILA Control File

title keyword appears in the input file is shown below and in Figure 7.1.

```
title
An 80-character string follows the ``title'' keyword
```

7.2 Analysis Parameters (analyze – end)

analyze – Starts the **analyze** – **end** block.

end – Terminates the analysis block.

The analysis parameters define the method of solution, i.e., the type of analysis to be performed and the solution algorithm to be used. These parameters also define algorithm specific options such as hourglass stabilization, the number of times steps to take, termination time, output intervals, etc. The analysis parameters are specified in the **analyze** – **end** block in the control file.

An example of how the **analyze** – **end** block would appear in the control file is shown below.

```
analyze
   solve 3
   nstep 10
   plti  10
   ...
end
```

Remark. In order to properly map nodal degrees-of-freedom at run-time, the **analyze** – **end** block must appear in the control file before any boundary condition blocks. This is the only order dependence that must be respected in the control file.

7.2.1 solve

solve *solver_id*

GILA provides multiple solver options for a variety of flow-related physics. The selection of the physics and related solver is obtained with the **solve** keyword in the **analyze - end** block. The **solve** command requires one argument that defines the physics and the underlying assumptions for solving the associated partial differential equations. Currently, there are 4 valid values for *solver_id*.

1: Transient, incompressible, Navier-Stokes using single-point quadrature, lumped mass, and Forward Euler time integration (*default*).

3: Transient, incompressible, Navier-Stokes using full quadrature and P2.

101: Transient advection-diffusion using Forward Euler with a prescribed velocity field ($\hat{\mathbf{u}}$). This option makes use of single-point integration.

102: Transient advection diffusion using semi-implicit implicit time integration with a prescribed velocity field ($\hat{\mathbf{u}}$). This option makes use of fully integrated elements.

7.2.2 temp

temp *flag*

Activate (**temp=1**) the solution of the temperature equation with the Navier-Stokes. (*default:* **temp=0**).

7.2.3 species

species *flag*

Activate (**species=1**) the solution of the species transport equations with the Navier-Stokes. (*default:* **species=0**).

7.2.4 react

react *flag*

Activate (**react=1**) chemical reactions with the transport of species and temperature. This option requires that **temp=1** and **species=1**. (*default:* **react=0**).

Time Step and Time Integration Options

The following keywords are provided to set/modify the parameters associated with the time step and the associated time integration methods.

7.2.5 nstep

nstep N_{step}

Define the number of time steps, N_{step}, to be taken during a single simulation. (*default:* **nstep=10**).

7.2.6 term

term τ_f

Define the simulation termination time, τ_f, in units consistent with the problem definition. (*default:* **term=1.0**).

> **Remark.** The **term** keyword and **nstep** keywords both affect the the length of simulated time in GILA. If the number of time steps specified using the **nstep** keyword would yield a simulation time greater than the termination time, the number of time steps is reduced to terminate the calculation according to the **term** command. Thus, the **term** keyword places a ceiling on the simulation termination time regardless of how many time steps have been specified by the **nstep** keyword.

7.2.7 deltat

deltat Δt

Define the time step size, Δt, to be used. This value may be over-ridden during the stable time step computation. (*default:* **deltat=0.01**).

7.2.8 average – ;

average

stats	*level*
start	*time*
avgi	N_{avg}

;

The **average – ;** keyword sequence activates time-averaging by setting the statistics *level* to a non-zero value of 10, 20 or 30, and prescribing a time to start accumulating statistics. Activating the time averaging results in an additional plot file containing the mean field values, correlations, skewness and flatness generated at the end of a simulation. The time-average plot file is the plot file name with `.avg` appended, e.g., `plot.avg`. By default, the time averaging parameters are: **stats**=0, **start**=0.0, and **avgi**=100.

> **Remark.** Note that time averaging is only available for analyses that solve the time-dependent Navier-Stokes equations, and is not fully operational for the time-dependent advection-diffusion solvers. Currently, the accumulated time-averages are only available with the EXODUS and PXI state graphics databases. The

derivation of additional statics require the use of the statistics post-processor, LESTATS.[15]

stats *level*

Set the level of statistics to gather during the calculation.

stats=10**:** Mean quantities are computed during the computation and output at the selected time averaging interval. The mean quantities generated at this level include the velocity, $< u_i >$, pressure, $< p >$, vorticity $< \omega_i >$, enstrophy, $1/2 < \omega_i > \cdot < \omega_i >$, and helicity, $< \omega_i > \cdot < u_i >$. For thermal problems, the mean quantities also include the temperature, $< T >$, and for species transport, the mean species concentration, $< Z_j >$.

stats=20**:** At this level of statistics, second moments are computed during the computation and output at in the time-average dump interval. In addition to the mean quantities, for **stats**=10, the turbulent stress tensor and scalar flux vectors are computed and output. That is, $< u_i u_j >$, $< u_i T >$, $< u_i Z_j >$, $< u_i p >$ are added to the time average database.

stats=30**:** In addition to the mean and derived flux quantities, the higher-order averages are generated that include $< u_i^3 >$ and $< u_i^4 >$.

start t_{start}

Set the simulation time to start collecting statistics.

avgi N_{avg}

Set the interval to write plot files containing the time-averaged statistics to N_{avg}. The statistics are re-initialized and collection of new statistics starts after the time-averaged data is written to the plot file.

7.2.9 cfl

cfl CFL_{max}

Specify the maximum CFL number, CFL_{max}, for the problem. The maximum CFL number is used in the time-step calculation for explicit algorithm only when it controls the time step. In the projection algorithm, the maximum CFL number is used to compute the time step in conjunction when the **dtchk** keyword is activated. (*default:* **cfl=1.0**).

7.2.10 dtchk

dtchk $N_{\Delta t}$

Interval to check the stable time step size and report grid parameters. A negative value for the interval causes the time step to be checked but not changed at the specified interval, i.e., it forces the grid parameters (Re^h, CFL^h) to be reported (*default:* **dtchk=10**).

7.2.11 thetak

thetak θ_K

Time weight for viscous terms: $0 \leq$ thetak ≤ 1 (*default:* **thetak=0.5**).

7.2.12 thetab

thetab θ_B

Time weight for balancing tensor diffusivity (BTD): $0 \leq$ thetab ≤ 1 (*default:* **thetab=0.5**).

7.2.13 thetaa

thetaa θ_A

Time weight for advection: $0 \leq$ thetaa ≤ 1 (*default:* **thetaa=0.0**). **This option is currently parsed, but ignored.**

7.2.14 thetaf

thetaf θ_F

Time weight for body forces: $0 \leq$ thetaf ≤ 1 (*default:* **thetaf=0.5**).

Hourglass Control Options

The following keywords are provided to set/modify the parameters associated with the hourglass stabilization. These keywords are effective only for the explicit time integration algorithms that make use of the reduced-integration element.

7.2.15 hglass

hglass *type*

Specify the hourglass *type* to use. (*default:* **hglass=1**). Valid options for *type* are:

1: Specifies h-type hourglass control.

2: Specifies that the Γ-stabilization should be used. Currently, Γ-stabilization is only available for two dimensional calculations.

7.2.16 epshg

epshg ϵ_{hg}

Set the hourglass coefficient. (*default:* **epshg=1.0**).

Miscellaneous Options

7.2.17 epsdt

epsdt $\epsilon_{\Delta t}$

Set the accuracy measure for sub-cycling the pressure solves. This is applicable only to the explicit time-integration algorithm. (*default:* **epsdt=1.0e-10**).

7.2.18 divu

divu $\|\nabla \cdot \mathbf{u}\|_{RMS}$

Set the RMS divergence tolerance. (*default:* **divu**=$1.0e-10$).

7.2.19 mass

mass *type*

Set the *type* of mass matrix. In GILA, the mass matrix can take on the form of a lumped mass matrix, M_l, a consistent mass matrix, M_c, or the higher-order mass matrix. The higher order mass matrix is defined as $M_{ho} = \frac{1}{2}\ M_l + \frac{1}{2}\ M_c$. Valid options for mass matrix *type* are:

mass=1: Sets the lumped Mass.

mass=2: Sets the consistent mass matrix.

mass=3: Sets the "higher-order" mass matrix.

(*default:* **mass=1**) for **solve=1,101**, *default:* **mass=2** for **solve=3,103**.)

Advection Options

There are currently three advection schemes in GILA. All of the advection schemes are explicit in time and rely on a basic underlying second-order central advection scheme. For low Reynolds number flows, balancing tensor diffusivity (BTD) preserves stability by promoting the advection scheme to second-order in time. The flux-limiting algorithm (FCT) is monotonicity preserving and can work in conjunction with BTD to yield very accurate high-Reynolds number simulations. The third option is an experimental scheme that relies upon a gradient-based metric.

7.2.20 btd

btd *flag*

Toggle the balancing tensor diffusivity *flag*. **btd=1** turns balancing tensor diffusivity (BTD) on, while **btd=0** turns BTD off. (*default:* **btd=1**).

7.2.21 fct

fct *flag*

Toggle the flux-limiting advection scheme on or off. **fct=1** turns the flux-limiting algorithm on, while **fct=0** turns it off. (*default:* **fct=1**).

7.2.22 icset

icset

set_u	u_0
set_v	v_0
set_w	w_0
set_T	T_0
set_z1	$Z1_0$
set_k	k_0
set_e	ϵ_0
set_a2	$A2_0$

;

The **icset** – **;** sequence is used to control the initialization of the velocity, temperature, etc. The complete command syntax is provided with the explanation of each parameter below. The keyword, **icset** starts the command block, and the '**;**' terminates the block.

set_u u_0

Set the initial x-velocity to u_0.

set_v v_0

Set the initial y-velocity to v_0.

set_w w_0

Set the initial z-velocity to w_0.

set_t T_0

Set the initial temperature to T_0.

set_z1 $Z1_0$

Set the initial mass fraction to $Z1_0$.

set_k k_0

Set the initial turbulent kinetic energy to k_0.

set_e ϵ_0

Set the initial turbulent dissipation rate to ϵ_0.

set_a2 $A2_0$

Set the initial second invariant of the Reynolds stress tensor to $A2_0$.

Output Options

The following commands are used to control the type of ASCII output, plot files, and the intervals at which data is written to each type of file.

7.2.23 pltype

pltype *type*

Select the type of state graphics database to be generated. Valid options for *type* are:

pltype=0: disables all graphics state output.

pltype=1: generates GRIZ[18] compatible plot files. Note that if this file option is selected for state databases, the time history database defaults to the THUG[61] database.

pltype=10: generates EXODUS state and time-average output files. Note that if this file option is selected for state databases, the time history database defaults to the HISPLT[64] database.

pltype=20: generates PXI state and time-average output files. Note that if this file option is selected for state databases, the time history database defaults to the HISPLT[64] database.

7.2.24 plti

plti N_{plot}

Set the plot state output interval, N_{plot}. (*default:* **plti=10**).

7.2.25 prtlev

prtlev *level*

Set the print *level* to control the amount of data written to the ASCII (human-readable) output file. **level=0** suppresses all output except for the primary code settings. **level=1** produces a data echo of the mesh coordinates and topology, while **level=2** produces a data echo of the primitive variables every N_{print} time steps. (*default:* **prtlev=0**).

7.2.26 prti

prti N_{print}

Set the hard copy print interval, N_{print}, for the output of primitive variables. This requires **prtlev=2**. (*default:* **prti=10**).

7.2.27 ttyi

ttyi N_{step}

Set the interval to report the min/max values of the velocity to N_{step}. The min/max values are normally written to the screen at run-time. (*default:* **ttyi=10**).

7.2.28 thtype

thtype *type*

Select the type of time-history database to be generated. *type* may take on one of the following values.

thtype=0: disables all graphics state output.

thtype=1: generates THUG[61] compatible time history files. Note that if this file option is selected for the time history database, the state database defaults to the GRIZ[18] database.

thtype=20: generates time history files compatible with HISPLT.[64]

7.2.29 vortics

vortics *flag*

This command activates the output of the voriticity field associated with the velocity field at the final time-step of the simulation to a file to be used as initial conditions in a subsequent vorticity-based calculation. Any non-zero value for *flag* activates this option. The velocity initial conditions are written to the output file name with a '.vor' appended to the name. This provides the ability to dump a vorticity field in a “frozen” state for later use with a vorticity-based flow solver. (*default:* **vortics=0**).

7.2.30 icwrt

icwrt *flag*

This command activates the output of primitive variables at the final time-step of a simulation to a file to be used as initial conditions in a subsequent calculation. Any non-zero value for *flag* activates this option. The velocity initial conditions are written to the output file name with a '.ics' appended to the name. This provides the ability to dump a velocity field in a “frozen” state for later use with the advection-diffusion options in GILA. (*default:* **icwrt=0**).

7.2.31 dump

dump N_{dump}

Activate checkpoint-restart files where a restart file will be written every N_{dump} time steps. In addition, a restart file will be written at the termination of the calculation. (*default:* **dump=0**).

7.3 Fluid Models (material – end)

material *id* – Start the material block with the material identifier, *id*.

end – Terminate the material block.

The specification of a material model occurs in the **material** – **end** block in the control file. In the specification of the material, an integer identification, *id*, is required. The material *id* must match the material (element block) numbers produced during mesh generation for each material (element block) in the mesh.

An example of how the **material – end** block would appear in the input file is shown below.

```
material 1
   model 10
   rho   1000.0
   mu    1.0e-3
   ...
end
```

Material Parameters

The following keyword – parameter commands are provided to specify fluid properties.

7.3.1 model

model *type*

Set the model that is to be identified with the material/block *id*. The table below shows the models that may be selected. (*default:* **model=1**).

Model Number	**Description**
1	Constant properties (no EOS)
10	Reacting flow - ideal gas EOS

7.3.2 rho

rho ρ

Set the reference fluid density, ρ. (*default:* **rho=1.0**).

7.3.3 mu

mu μ

Set the reference fluid dynamic viscosity, μ. (*default:* **mu=1.0**).

7.3.4 k

k k

Set the reference thermal conductivity, k. (*default:* **k=1.0**).

7.3.5 diff1

diff1 $\mathcal{D}_1$

Set the reference mass diffusivity $\mathcal{D}_1$. (*default:* **diff1=1.0**).

7.3.6 beta

beta β

Set the volumetric expansion coefficient, β. (*default:* **beta=0.0**).

7.3.7 t_ref

t_ref T_∞

Set the reference temperature, T_∞. (*default:* **t_ref=0.0**).

7.3.8 gx

gx g_x

Set the x-component of the gravity to g_x. (*default:* **gx=0.0**).

7.3.9 gy

gy g_y

Set the y-component of the gravity to g_y. (*default:* **gy=0.0**).

7.3.10 gz

gz g_z

Set the z-component of the gravity to g_z. (*default:* **gz=0.0**).

7.3.11 p0

p0 p_0

Set the reference thermodynamic pressure to p_0. This pressure is used as the initial condition for the evolution of the thermodynamic pressure in the low-Mach number formulation (*default:* **p0=0.0**).

Arrenhius Reaction Constants

7.3.12 a

a a

Set the Arrenhius constant, a.

7.3.13 e

e E

Set the activation energy, E.

7.3.14 r

r R

Set the gas constant, R.

7.3.15 cp

cp C_p

Set the constant-pressure specific heat, C_p.

7.3.16 gamma

gamma γ

Set the ratio of the specific heats, γ. (*default:* **gamma=1.4**).

7.3.17 q0

q0 q_0

Set the heat release coefficient, q_0.

7.4 Momentum Equation Solver (momsol – end)

The solution of the Navier-Stokes equations using the P2 algorithm, or the solution of advection-diffusion equation with the semi-implicit algorithm requires the selection of an linear equation solver and the concomitant solver parameters. For example, selection of an iterative solver requires the specification of a convergence criteria, the maximum number of iterations, and the interval to check the residual norms. See Chapter 3 for definitions of error norms used in the iterative solvers.

momsol *solver* Specify the momentum solver, *solver* to be used. Valid parameters for the momentum *solver* are:

1: Jacobi pre-conditioned conjugate gradient solver.

end Terminate the momentum solver block.

An example of how the **momsol** – **end** block would appear in the input file is shown below.

```
momsol    1
   itmax  100
   itchk  10
   eps    1.0e-5
   wrt    0
   hist   0
end
```

Remark. In the selection of an analysis option, the selection of a momentum equation solver is implied. For example, *solve 1* selects the explicit time integration scheme which requires no momentum solver. Similarly, *solve 101* selects the explicit solver for the transient advection-diffusion equation and does not require any solver options to be set.

By selecting an analysis option that can perform a variety of time integration schemes, it is implied that the user select appropriate convergence criteria and limits. For example, *solve 2* selects the P2 option which employs an element-by-element conjugate gradient solver for the momentum equations, and will use the default iteration limits and convergence criteria unless otherwise specified by the user. In a similar fashion, the selection of *solve = 3* or *solve = 102* implicitly requires the user to set the iteration limit and convergence criteria.

Momentum Equation Solver Parameters

The following keyword-parameter pairs may be used to control the linear equation solver used for the momentum equations.

7.4.1 itmax

itmax N_{itmax}

Set the iteration limit, N_{itmax}. (*default:* **itmax=10**).

7.4.2 itchk

itchk N_{check}

Set the interval to check convergence, N_{check}. (*default:* **itchk=5**).

7.4.3 eps

eps ϵ

Convergence tolerance. (*default*: **eps**=$1.0e-10$).

7.4.4 wrt

wrt *flag*

Toggle the output of diagnostic information to the screen. **wrt=0** suppresses diagnostic information from the solver, while **wrt=1** activates diagnostic information from solver. (*default*: **wrt=0**).

7.4.5 hist

hist *flag*

Toggle the output of the ASCII convergence history file called *history.mom*. **hist=0** suppresses writing the ASCII convergence history file, while **hist=1** activates the ASCII convergence history file. (*default*: **hist=0**).

7.5 Pressure Equation Solvers (ppesol – end)

In the solution of the Navier-Stokes equations using either the explicit, or the implicit (P2) algorithms, a pressure equation (PPE) must be solved. The selection of either a direct or an iterative solver requires the specification of parameters such as the convergence criteria, the maximum number of iterations, and the interval to check the residual norms.

There are a variety of options that may be enabled to control the behavior of the pressure solvers. However, each solver can have its own unique control keywords and parameters. Thus, each solver is presented with its complete set of keywords. Note that the direct solvers do not require any parameters to be set other than the *solver_id*. In the list of valid solver options, each solver is identified as either (MP) indicating that the solver may be used in parallel, or (Serial) indicating that the solver is only available for serial/vector platforms.

ppesol *solver_id*

solver_id may be one of the following pressure solvers.

1: Element-by-element, Jacobi-preconditioned conjugate gradient with pressure extrapolation (MP).

2: Saddle-point solver (MP).

11: UDU format Gaussian elimination (Serial).

12: UDU format ITLIB solver options (Serial).

41: Parallel-vector (PVS) row solver (Serial).

42: Parallel-vector (PVS) preconditioner with EBE solver (MP).

61: compact storage ITPACK Jacobi-preconditioned conjugate gradient (MP).

62: row-sum stabilized ITPACK Jacobi-preconditioned conjugate gradient (MP).

63: compact storage ITPACK SSOR-preconditioned conjugate gradient (MP). In MP mode, this preconditioner is effectively a subdomain preconditioner.

64: compact storage ITPACK SSOR-preconditioned conjugate gradient with Eisenstat transformation.

end Terminate the pressure solver block.

An example of how the **ppesol** − **end** block would appear in the input file is shown below.

```
ppesol   1
   itmax  100
   itchk  10
   eps    1.0e-5
   wrt    0
   hist   0
end
```

Pressure Equation Solver Parameters

The following keyword-parameter pairs may be used to set to control the linear equation solver used for the momentum equations.

7.5.1 itmax

itmax N_{itmax}

Set the iteration limit, N_{itmax}. (*default:* **itmax=10**).

7.5.2 itchk

itchk N_{check}

Set the interval to check convergence, N_{check}. (*default:* **itchk=5**).

7.5.3 novec

novec N_{vec}

Set the number of vectors used in the orthogonal projection conjugate gradient solvers. This option is valid for **ppesol=1,61,62,63,64**. (*default:* **novec=0**).

7.5.4 eps

eps ϵ

Convergence tolerance. (*default*: **eps**=$1.0e-10$).

7.5.5 wrt

wrt *flag*

Toggle the output of diagnostic information to the screen. **wrt=0** suppresses diagnostic information from the solver, while **wrt=1** activates diagnostic information from solver. (*default*: **wrt=0**).

7.5.6 hist

hist *flag*

Toggle the output of the ASCII convergence history file called *history.ppe*. **hist=0** suppresses writing an ASCII convergence history file, while **hist=1** activates the ASCII convergence history file. (*default*: **hist=0**).

7.5.7 omega

omega ω

SSOR over-relaxation parameter, $1 \le \omega \le 2$, applies only to solvers **63** – **64**. (*default*: **omega=1.0**).

7.5.8 stab

stab *flag*

Activate the stabilization for the Pressure Poisson equation. The stabilization options below apply only to solvers **61** − **64**. A complete description of these stabilization techniques may be found in Christon.[13] Valid values for the stabilization *flag* are:

10: Simple row-sum stabilization.

20: Global jump stabilization.

30: Local jump stabilization.

7.5.9 beta

beta β

Associated with the stabilization option is a stabilization parameter β. The **beta** keyword sets the global jump stabilization parameter to β. (*default*: **beta=0.0**)

7.6 Nodal Time History Blocks (ndhist – end)

Time history nodes may be defined to track primitive variables at a small number of nodes where the interval that the nodal data is recorded at is much smaller than for state data. The time history data is recorded in the time history database specified as a command line execution option. All nodal time history parameters are specified in a **ndhist** – **end** block in the control file.

ndhist ***n*** Specify n time history nodes. (*default*: **n=0**).

end Terminate the nodal time history block.

An example of how the **ndhist** – **end** block would appear in the control file is shown below.

```
ndhist    10
   st     100
   en     110
   ...
   nstep    2
end
```

Nodal Time History Parameters

The following keywords may be used to set the nodal time history parameters.

7.6.1 st

st *start_node*

Set the starting node number in the block.

7.6.2 en

en *end_node*

Set the ending node number in the block.

7.6.3 nstep

nstep N_{step}

Set the time history output interval. (*default*: **nstep=1**).

7.7 Turbulence Models

Turbulence models and their associated parameters are specified in **turb − end** block in the the control file.

turb ***model*** Specify the turbulence model, *model* to be used. Valid parameters for the turbulence *model* are:

- **1:** Smagorinsky model without dynamic subgrid scale.
- **100:** Baseline $k - \epsilon$ model.
- **102:** Experimental version of Lien – Leschziner $k - \epsilon$ model.
- **103:** Experimental version of Launder's $k - \epsilon - a_2$ model.

end Terminate the turbulence model block.

An example of how the **turb − end** block would appear in the input file is shown below.

```
turb     1
   smagc  0.1
   ...
end
```

Turbulence Model Parameters

The following keywords may be used to set the turbulence model parameters.

7.7.1 smagc

smagc C_s

Set the value of the Smagorinsky constant, C_s. (*default*: **smagc=0.1**).

7.8 Nodal Boundary Conditions

Prescribed nodal boundary conditions are input in a block in the control file that is initiated with a boundary condition keyword and terminated with **end**. For example, to prescribe boundary values for the x-velocity the **ubc** – **end** block would be used with a **nodeset** keyword included in the block. The arguments for the **nodeset** command consist of the nodeset *id*, the amplitude for the load curve, *amp*, and the load curve identifier, *lc* as shown below.

ubc
  nodeset *id amp lc*
end

7.8.1 nodeset

nodeset *id amp lc*

Specify the nodeset identifier, *id*, the nodeset amplitude, *amp*, and the nodeset load curve, *lc*. Any valid load curve id greater than 0 may be used for *lc*. In addition, pre-programmed analytic load curves may be specified with the following values for the load curve id:

0: Specifies a fixed amplitude essential boundary condition.

-1: Prescribes a steady, parabolic distribution for the velocity.

-2: Prescribes a time-dependent, parabolic distribution for the velocity

7.8.2 ubc – end

The **ubc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the x-velocity

7.8.3 vbc – end

The **vbc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the y-velocity

7.8.4 wbc – end

The **wbc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the z-velocity

7.8.5 tbc – end

The **tbc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the temperature field.

7.8.6 rbc – end

The **rbc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the mass density, ρ.

7.8.7 kbc – end

The **kbc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the turbulent kinetic energy.

7.8.8 ebc – end

The **ubc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the turbulent rate of dissipation, ϵ.

7.8.9 abc – end

The **abc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the second invariant of the Reynolds stress tensor, A_2.

7.8.10 zbc – end

The **zbc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the mass fraction, Z_1.

7.8.11 ybc – end

The **ybc – end** block is used with the **nodeset** keyword to prescribe the nodeset *id*, the nodeset amplitude, *amp*, and the associated load curve, *lc*, for the mass fraction, Z_2.

7.9 Outflow Boundary Conditions

outflow
        sideset *id*
end

Outflow boundary conditions are input in a **outflow – end** block in the control file. Outflow boundaries are identified in the **outflow – end** block with the **sideset** keyword. The arguments for the **sideset** command consist of the sideset *id* as shown below.

7.9.1 sideset

sideset *id*

Specify the sideset identifier, *id*. Valid values for *id* are restricted to the sideset id's produced during the mesh generation process.

7.10 Pressure Boundary Condition (pbc – end)

The **pbc – end** block is used to apply a pressure load over each face in a sideset . This keyword activates what is commonly referred to as the pressure boundary condition according to §5.3.1.

7.11 Pressure Levels (ppebc – end)

Set the number of pressure values, N_{pbc} to be pegged during the pressure solve. In the **ppebc – end** block, N_{pbc} pressure levels, P_{id} may be associated with each element *id* in the list of N_{pbc} elements.

ppebc N_{pbc}
        el *id* P_{id}
end

7.12 Initial Conditions

Initial conditions may be input in GILA in a variety of ways. The simplest method is to use the **icset** command in the **analyze − end** section of the control file. In addition, there are a variety of specialized functions for initializing the velocity field. Finally, an existing velocity field may be read from a PXI file using the **icfile** command.

7.12.1 Initial Condition Functions (icfn – end)

Option 1:

Option 1 of the **icfn − end** command initializes the temperature field to T in a spherical/circular region of radius, R centered at (X,Y,Z). If the **frac** option is specified then the mass fraction is set to F in the spherical region.

```
icfn  1
      x       X
      y       Y
      z       Z
      radius  R
      temp    T
      frac    F
end
```

Option 2:

Option 2 of the **icfn − end** command initializes a fixed velocity field with amplitude V in the entire domain.

```
icfn  2
      vel  V
end
```

Option 3:

Option 3 of the **icfn − end** command initializes a velocity field with counter-rotating cells. The amplitude of the velocity field, V, and the number of cells, $Ncell$, in a single spatial dimension may be specified.

```
icfn  3
      vel    V
      ncell  N_cell
end
```

Option 10:

Option 10 of the **icfn – end** command initializes a velocity field by reading an ASCII file generated by a previous run using the **icwrt** command.

```
icfn  10
      file  filename
end
```

7.12.2 Initial Condition Input (icfile – end)

```
icfile
      file    filename
      index   id
      type    filetype
end
```

The **icfile – end** section of the input file identifies a local filename that contains the velocity field to be read and used as initial conditions. The filename may be up to 80 characters long and can contain relative path information. Currently, only velocity values are read from the file.

Valid file types are either **ASCII** or **pxi**. For **type=pxi**, the *id* is the time index in the PXI database associated with the velocity data. The **icwrt** command is used to generate an ascii file that is compatible with the **type = ascii** initial condition file.

7.13 Derived Force Computation (dforce – end)

```
dforce
      sideset  id
end
```

The computation of derived forces is selected with the **dforce – end** block in the control file. The computation of derived forces yields the Cartesian components of the resultant forces on each prescribed sideset surface. Currently, the derived forces are output to the global results file, *glob*.

The surfaces to be used for the derived force computations are identified in the **dforce – end** block with the **sideset** keyword. The arguments for the **sideset** command consist of the sideset *id* as shown below.

7.13.1 sideset

sideset *id*

Specify the sideset identifier, *id*. Valid values for *id* are restricted to the sideset id's produced during the mesh generation process.

7.14 Load Curve (lcurve – end)

lcurve	*id*	*Npts*
	t_1	v_1
	t_2	v_2
	t_3	v_3
	...	
	t_{Npts}	v_{Npts}
end		

The input of load-curve data for time-dependent boundary conditions requires the specification of a load curve *id*, the number of points associated with the load curve, *Npts*, and the data associated with the load curve. The load curve *id* must be positive and non-zero. The time – value (t_i, v_i) pairs are expected to be in a space-delimited contiguous list.

7.15 Parallel File Sytem (pfs – end)

The parallel file system block activates the use of the parallel file system of the ASCI Red TFLOPs machine, or similar system that provides an array of parallel file systems. The available options for using a parallel file system are listed below and control the output of graphics state databases only when **pltype** = 1 in the **analyze** – **end** block. Otherwise, the **pfs** options apply to the ASCII output, time history databases, and checkpoint-restart files.

pfs	*id*	
	ndisk	N_{disk}
	start	*starting_disk*
end		

Valid values for *id* are:

1: Sets the root portion of the path for files written on the raid system to : `/raid/io_`.

2: Sets the root portion of the path for files written on the pfs to : `/pfs/io_`.

3: Sets the root portion of the path for files written on the pfs to : `/pfs/tmp_`.

> **Remark.** The use of the **pfs** control block assumes that disks on the parallel file system are numbered sequentially from some starting disk number. This model also assumes that one file per processor will be written in a "round-robin" fashion on N_{disk} **pfs** disks beginning with the *starting_disk*. Currently, this option applies only to the parallel file systems on Sandia's Intel Paragon and ASCI Red machines. Here, a user id must be provided on the command-line options in order for GILA to properly open and write the files across the parallel file system. For example, `-o /machris/out`, where `machris` is the user id, is required as a command line option in order to properly place the output files on the parallel file system.

Parallel File System Options

The following keywords may be used to set the **pfs** options.

7.15.1 ndisk

ndisk N_{disk}

Set the number of disks in the parallel file system to use. Typically, this number also corresponds to the number of I/O nodes to be used.

7.15.2 start

start *starting_disk*

Set the starting disk number to be used. For example, if N_{disk} **pfs** disks is specified with a starting disk number of 10, and *id = 1*, then the state, time-history and checkpoint-restart files will be written in parallel directories starting with `/raid/io_10/`.

Chapter 8

Example Problems

This chapter presents several GILA calculations for the first-time user who wishes to perform simple computations for comparison before embarking on a full-blown analysis. For this reason, the control files are replicated here with representative results that can be compared for a rough verification of the local GILA installation. In addition, most of the sample problems use relatively coarse grids to minimize run times and provide a starting point for the user who wishes to experiment with code options before attempting any significant calculations.

8.1 Entrance Region in a 2-D Duct

This sample problem consists of the entrance region to a two-dimensional duct. At the duct inlet, a plug inlet velocity profile ($u = 1$) is prescribed for the x-velocity along with homogeneous natural boundary conditions in the y-direction. This permits a vertical "slip" velocity along the inlet plane. No-slip and no-penetration conditions ($u = 0$, $v = 0$) are prescribed along the top and bottom duct walls. Natural boundary conditions are applied at the outflow boundary. The computaional grid contains 1000 elements, and has a $20 : 1$ aspect ratio as shown in Figure 8.1. The boundary conditions have been designed to result in a pressure singularity at the corners of the inlet to the duct.

The GILA control file is shown in Figure 8.2 for a Reynolds number of 100 based on the channel height. In this example, the time-accurate, second-order projection algorithm for incompressible flow is used along with the PVS direct solver for the pressure.

A segment of the screen output that reports the RMS divergence for the initial conditions before and after the div-free projection is shown in Figure 8.3. In this calculation, the divergence for the specified initial conditions failed the divergence test, so a mass-consistent projection to a divergence-

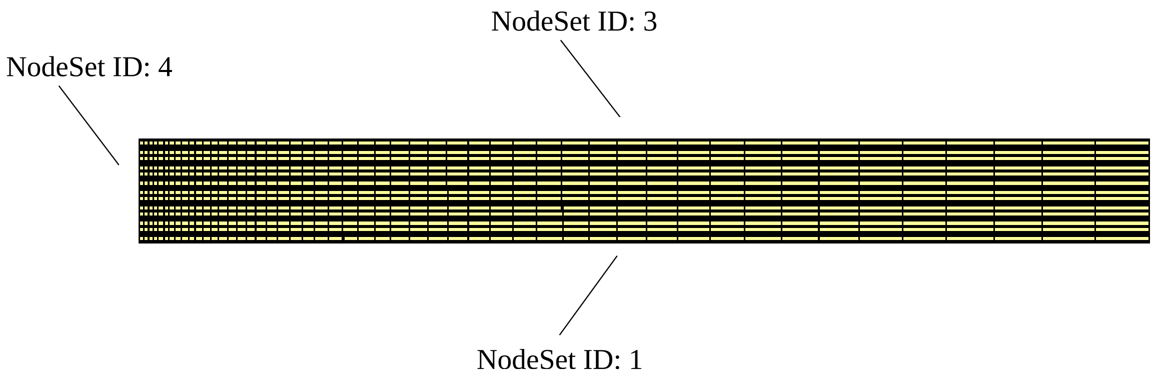


Figure 8.1: Mesh for the two-dimensional duct entrance region.

```
title
Duct Entrance Region

{ Analysis paramters }
analyze
  solve      3
  nstep    250
  pltype    10
  plti      10
  prtlev     0
  prti       1
  term     100.00
  deltat     0.10
  dtchk     -1
  divu       1.0e-18
  thetak     1.0
  thetab     1.0
  mass       1
  icset
    set_u 0.0
    set_v 0.0
  ;
end

material 1
  model      1
  rho        1.0
  mu         1.0e-2
end

{ Prescribed x-velocity }
ubc
  nodeset  4  1.0475 0
  nodeset  1  0.0    0
  nodeset  3  0.0    0
end

{ Prescribed y-velocity }
vbc
  nodeset  1  0.0 0
  nodeset  3  0.0 0
end

{ Pressure solver }
ppesol 41 end
```

```

{ Momentum solver }
momsol     1
  itmax  50
  itchk   2
  eps     1.0e-8
  wrt     0
end

{ Time history nodes
  along the outlet
  plane }
ndhist 6
  nstep  1
  st 312 en 312
  st 516 en 516
  st 720 en 720
end

exit
```

Figure 8.2: Control file for two-dimensional duct entrance region.

```
D i v e r g e n c e   E r r o r

     Allowable Divergence ........................ 1.0000E-18

     Initial   Divergence ........................ 7.1238E-03

     Projected Divergence ........................ 1.3261E-14
```

Figure 8.3: Divergence diagnostics for the initial conditions before and after the div-free projection.

free subspace was performed on the initial velocity data in order to produce boundary conditions and intial conditions that are compatible. The resulting velocity field yielded an RMS divergence of $\|\nabla \cdot \mathbf{u}\|_{RMS} = 1.3261E - 14$.

GILA also reports the grid CFL and Reynolds numbers at the interval specified by the **dtchk** keyword. Sample screen output of the grid parameters are shown in Figure 8.4. The grid CFL and Reynolds numbers are reported in terms of the element local coordinates (ξ,η). For this calculation, the maximum grid CFL number was 1.1937 at $t = 0$ in element 001 as indicated by the grid parameters. In addition to the grid parameters, GILA reports the minimum and maximum velocities at intervals specified by the **ttyi** keyword as shown in Figure 8.5.

The duct calculation is carried out until a steady-state is essentially achieved as indicated in Figure 8.6. The time-history points were placed along the duct cross section at node 312 located at $(0.488, 0.3)$, node 516 at $(0.488, 0.5)$, and node 720 at $(0.488, 0.7)$. The outlet x-velocity profile in Figure 8.6 d) is plotted as points over the the exact solution for Poiseuille flow. Poiseuille flow in a channel is characterized by a balance between a constant pressure gradient and viscous shear forces, i.e., for steady-flow,

$$\frac{1}{Re}\frac{\partial^2 u}{\partial y^2} = \frac{\partial p}{\partial x}. \tag{8.1}$$

For the duct flow, the exact pressure gradient is $\partial p/\partial x = -0.12$, and at $Re = 100$, the resulting parabolic velocity profile has a maximum velocity of $u_{max} = 1.5$, and average velocity of $u_{avg} = 1.0$. As shown in Figure 8.6 d), the computed velocity profile essentially interpolates the exact velocity profile at the duct outlet. Figure 8.7 shows snapshots of the flow fields (velocity and pressure) at $t = 10$ time units.

```
G r i d   R e y n o l d s   &   C F L   N u m b e r s

     xi-grid   Reynolds Numbers:
     ===========================
     Min. Element no. ..........................         951
     Min. xi-element  dimension ................ 8.7748E-02
     Minimum xi-grid  Reynolds Number .......... 2.2979E+00
     Max. Element no. ..........................         950
     Max. xi-element  dimension ................ 1.0833E+00
     Maximum xi-grid  Reynolds Number .......... 5.6738E+01

     eta-grid  Reynolds Number:
     ==========================
     Min. Element no. ..........................           1
     Min. eta-element dimension ................ 5.0000E-02
     Minimum eta-grid Reynolds Number .......... 6.5742E-05
     Max. Element no. ..........................         785
     Max. eta-element dimension ................ 5.0000E-02
     Maximum eta-grid Reynolds Number .......... 8.0298E-03

     xi-grid   CFL Numbers:
     ======================
     Min. Element no. ..........................        1000
     Min. xi-element  dimension ................ 1.0833E+00
     Minimum xi-grid  CFL Number ............... 4.8345E-02
     Max. Element no. ..........................         901
     Max. xi-element  dimension ................ 8.7752E-02
     Maximum xi-grid  CFL Number ............... 1.1937E+00

     eta-grid  CFL Numbers:
     ======================
     Min. Element no. ..........................           1
     Min. eta-element dimension ................ 5.0000E-02
     Minimum eta-grid CFL Number ............... 5.2594E-05
     Max. Element no. ..........................         785
     Max. eta-element dimension ................ 5.0000E-02
     Maximum eta-grid CFL Number ............... 6.4238E-03
```

Figure 8.4: Screen report of grid parameters showing the minimum and maximum grid Reynolds and CFL numbers and and associated element numbers.

```
Step  #      Time       U-Min.      U-Max.      V-Min.      V-Max.
========  ==========  ==========  ==========  ==========  ==========
       0  0.0000E+00  0.0000E+00  0.1048E+01  -.1452E-03  0.6165E-04
      10  1.0000E+00  0.0000E+00  0.1271E+01  -.1791E+00  0.1791E+00
      20  2.0000E+00  0.0000E+00  0.1385E+01  -.1801E+00  0.1801E+00
      30  3.0000E+00  0.0000E+00  0.1445E+01  -.1801E+00  0.1801E+00
      40  4.0000E+00  0.0000E+00  0.1474E+01  -.1800E+00  0.1800E+00
      50  5.0000E+00  0.0000E+00  0.1487E+01  -.1800E+00  0.1800E+00
          . . .
     250  2.5000E+01  0.0000E+00  0.1499E+01  -.1800E+00  0.1800E+00
```

Figure 8.5: Screen output of minimum and maximum velocity values at every 10 time steps.

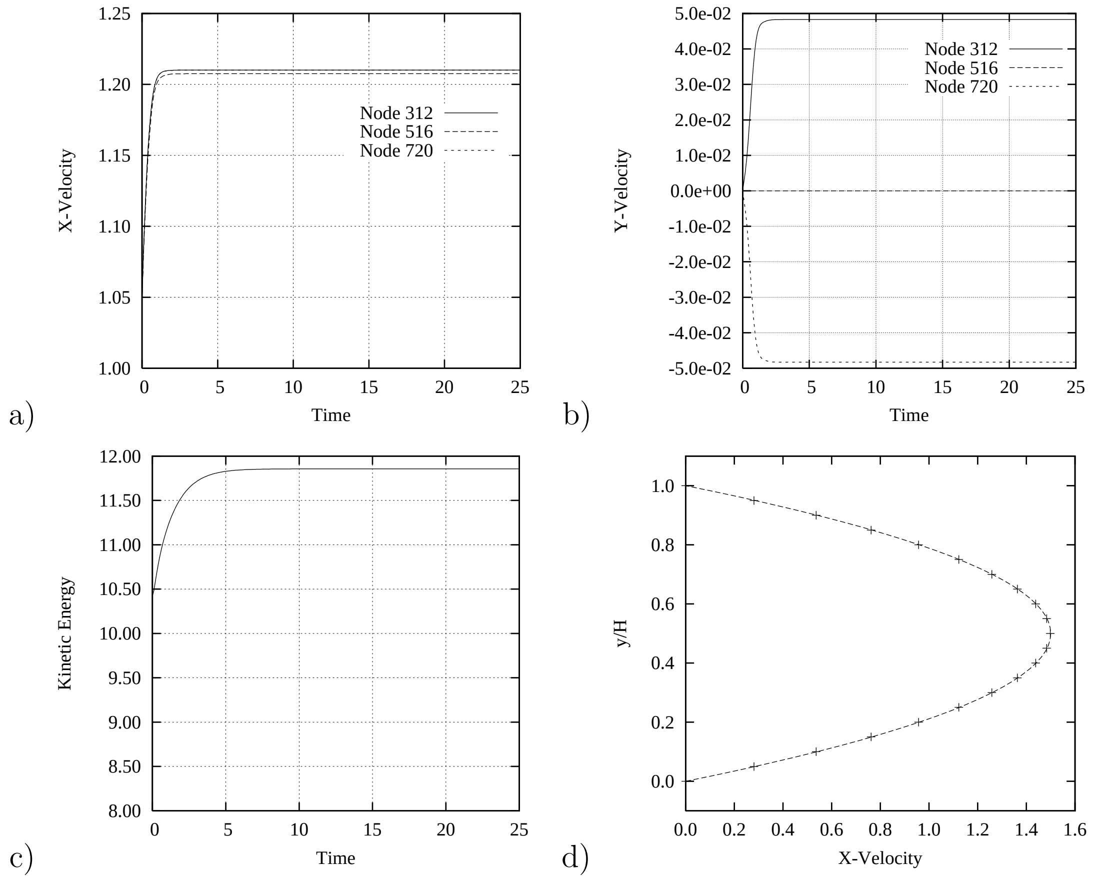
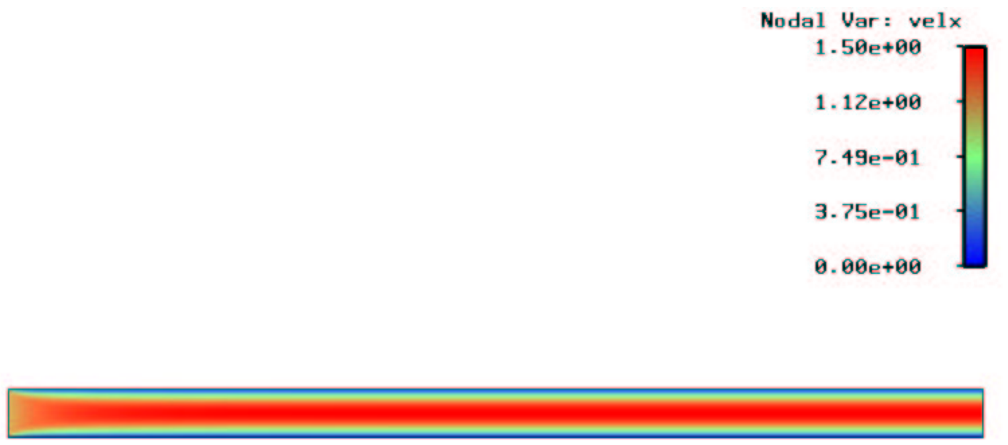
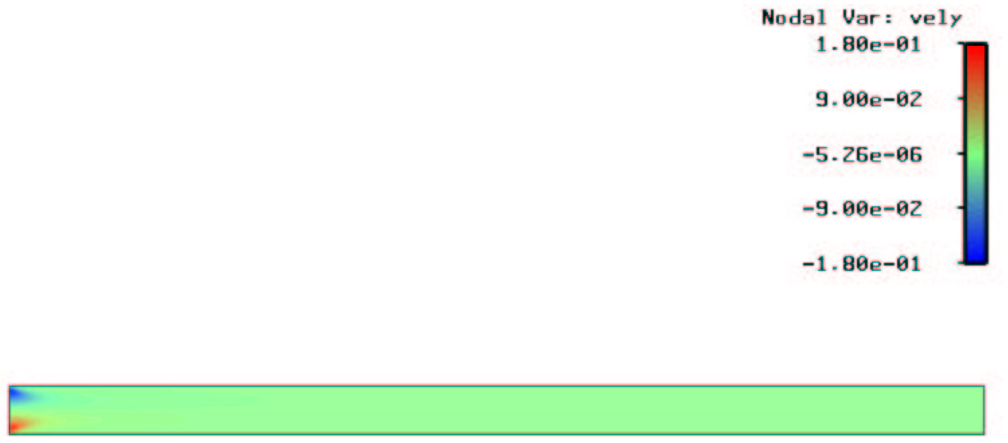


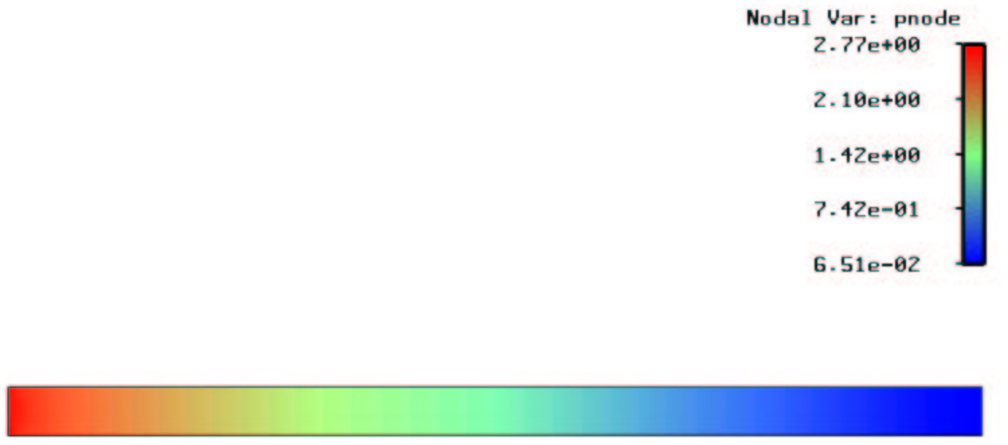
Figure 8.6: Nodal time history plots for a) x-velocity, b) y-velocity, c) kinetic energy, d) outlet x-velocity profile at t=10 time units.



a)



b)



c)

Figure 8.7: Snapshot of a) x-velocity, b) y-velocity, and c) pressure at $t = 10$ time units.

8.2 Backward Facing Step

Although motivated in part by Freitas,[21] the computations presented here for the backward facing step are based upon the $Re = 800$ backward facing step benchmark performed by Gartling.[22] This benchmark was chosen, in part, because it presents a demanding test for solution strategies applied to the PPE due to the pressure singularity at the corner of the step.

The backward facing step height is taken as $H/2$, the total channel height is H, and the length of the computational domain is $L = 30H$. A parabolic inlet velocity profile is specified above the step with Reynolds number defined as: $Re = U_{avg}H/\nu$. The isothermal flow solution is integrated in time from initial divergence-free conditions.

Following Gartling,[22] four grids with identical boundary conditions were constructed for the backward facing step. The grids are labeled B through E in Table 8.1 to correspond to those used by Gartling. However, the grid spacing used here has been doubled to account for the fact that the computations use the Q1Q0 element rather than the bi-quadratic velocity, linear discontinuous pressure elements employed by Gartling. On mesh E, this yields $387,362$ degrees-of-freedom (DOF) in the problem corresponding to the $355,362$ reported by Gartling.

The coarse-grid mesh used for case-B is shown in Figure 8.8. No-slip and no-penetration boundary conditions are prescribed using Nodesets 1, 2, and 4 which correspond to the upper/lower walls and downstream face of the step. At the inlet, Nodeset 3, a parabolic x-velocity profile was prescribed with zero vertical velocity ($v = 0$).

The GILA control file for the backward-facing step problem is shown in Figure 8.9. Here, a lumped-mass matrix (`mass 1`) and backward-Euler time-integrator (`thetak 1.0`, `thetab 1.0`) have been selected with the the PVS direct solver for the pressure. Note that a negative boundary condition load-curve id (-1) has been used with nodeset 3 to prescribe an inline computation of the parabolic x-velocity profile at the step inlet.

All the backward facing step calculations have been carried out to 400 time units, where the flow field has essentially established a steady-state condition, e.g., time history plots of the kinetic energy, and velocity components indicate that the solution is no longer changing in time. The separation and re-attachment lengths shown in Table 8.1 are for a snapshot at 400 time units. l_1, and l_3 are the re-attachment points on the upper and lower walls respectively, and l_2 is the separation point on the upper wall. The computed separation/re-attachment lengths are within about 1.25% of the benchmark results presented by Gartling, while the primary re-attachment point, l_1, is within 14% of the length estimated from Armaly, et al.[5]

Figure 8.10 shows snapshots of the pressure and vorticity fields for the

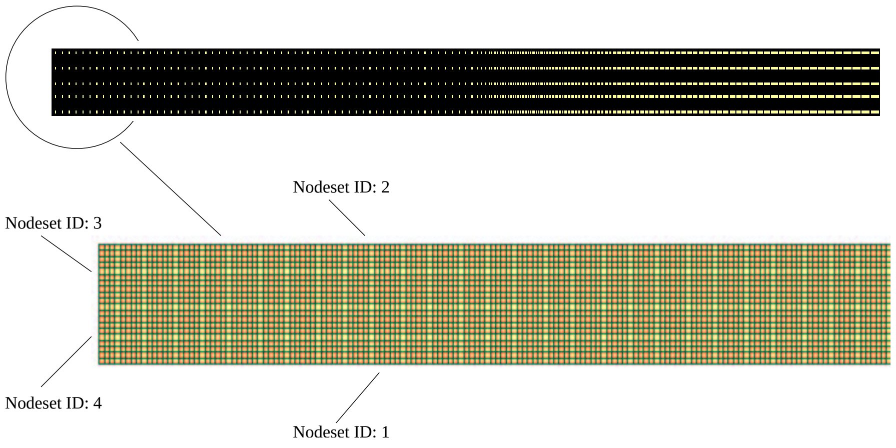


Figure 8.8: Coarse-grid mesh (case-B) for the Re=800 backward facing step.

```
title
Backward Facing Step

{Analysis parameters}
analyze
  solve  3
  nstep  10000
  plti     500
  prti   10000
  pltype  1
  dump    0
  prtlev  0
  ttyi    25
  icset
     set_u 1.0
     set_v 0.0
  ;
  mass    1
  btd     1
  divu    1.0e-10
  deltat  1.0e-1
  dtchk  -1
  dtscal  1
  thetak  1.0
  thetab  1.0
  term    400.00
end

material   1
  model   1
  rho     1.0
  mu      1.25e-3
end

ubc
  nodeset 3 1.5 -1
  nodeset 1 0.0  0
  nodeset 2 0.0  0
  nodeset 4 0.0  0
end

vbc
nodeset  1 0.0 0
nodeset  2 0.0 0
nodeset  3 0.0 0
nodeset  4 0.0 0
end
```

```

{PPE solver}
ppesol      64
  itmax   850
  itchk     1
  novec    15
  omega   1.0
  eps     1.0e-6
  wrt     0
end

{Momentum solver}
momsol    1
  itmax   50
  itchk    5
  omega   1.0
  eps     1.0e-5
  wrt     0
end

exit
{End-of-control file}
```

Figure 8.9: Control file for the backward facing step.

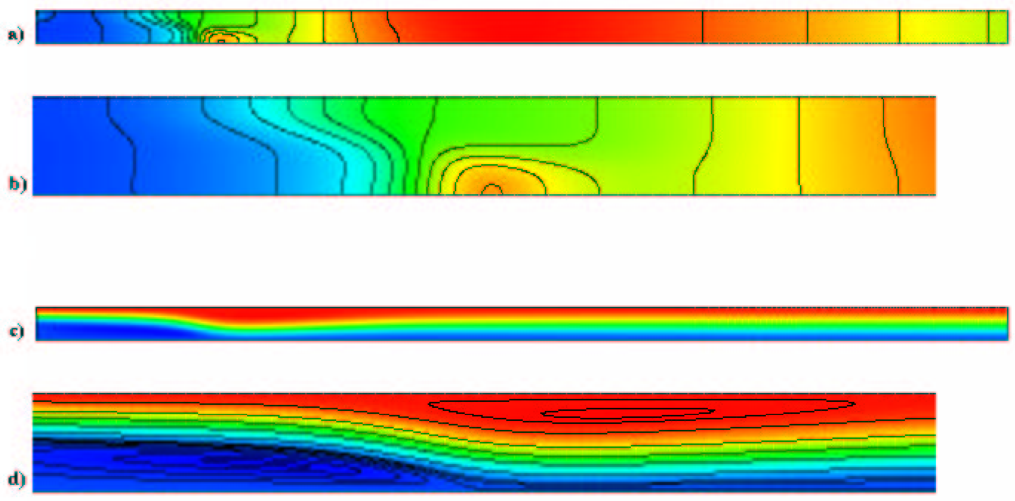


Figure 8.10: Backward facing step at 400 time units: a) and b) Pressure field, c) and d) Vorticity field.

backward facing step calculation at 400 time units. Figure 8.10a shows the entire pressure field, while Figure 8.10b shows the pressure field for $0 \leq x \leq 10$. Similarly, Figure 8.10c and d show the vorticity field for the entire domain and for $0 \leq x \leq 10$ respectively. Although not an exhaustive comparison, this calculation compares well with the published benchmark results[22] in terms of the steady-state field, and the separation/re-attachment lengths.

Table 8.1: Separation lengths for the backward facing step, $Re_H = 800$. Note that the element side lengths are for the uniform region of the mesh. (l_1 – length from the step face to the lower re-attachment point, l_2 – length from the step face to upper separation point, l_3 – length of the upper separation bubble. The element size is measured at the step inlet.)

Case	No. of Elements	No. of Unknowns	Element Size $(\Delta x, \Delta y)$	l_1	l_2	l_3
B	8000	24842	0.100	5.41	4.71	9.87
C	32000	97682	0.050	5.89	4.89	10.24
D	72000	218522	0.033	6.03	4.91	10.34
E	128000	387362	0.025	6.03	4.90	10.37
Gartling[22]	32000	355362	0.025	6.10	4.85	10.48
Armaly[5]	n/a	n/a	n/a	7.00	5.30	9.40

8.3 Vortex Shedding

The simulation of the von Karman vortex street behind a circular cylinder is a long-standing and well-known CFD benchmark for evaluating time-dependent solution algorithms for the incompressible Navier-Stokes equations. The results presented here are for a mesh modeled after that used for the benchmark calculation of Engleman and Jamnia.[19] The computational domain for the vortex shedding calculations consists of an interior square region $-4 \leq x \leq 4$, $-4 \leq y \leq 4$ surrounding the unit diameter cylinder. This core region is surrounded by a grid that spans $-8 \leq x \leq 24$, $-8 \leq y \leq 8$. Table 8.2 shows the radial grid spacing associated with each of the three meshes of coarse, medium and fine resolution. Here, the *coarse* mesh shown in Figure 8.11 is considered the base discretization, while the *medium* mesh corresponds to a 2×2 refinement, and the *fine* mesh constitutes a 3×3 refinement relative to the coarse mesh.

"Tow-tank" boundary conditions consisting of $u = 1, v = 0$ are applied at the upstream, top and bottom computational boundaries using nodeset 1, while outflow conditions i.e., homogeneous natural boundary conditions, are applied downstream. No-slip and no-penetration conditions are prescribed at the cylinder wall using nodeset 2. The Reynolds number for the vortex shedding calculations is $Re = 100$ based upon the cylinder diameter. The GILA control file for this problem is shown in Figure 8.11.

All of the flow computations are started from initial conditions that are divergence-free. After startup, the flow field evolves through a quasi-steady vortex stretching phase before the buildup of numerical round-off trips the vortex shedding. The number of degrees-of-freedom and the time steps associated with each of the three meshes are shown in Table 8.2. The kinetic energy time-history for the coarse mesh indicates that a "steady" periodic flow is achieved after approximately 160 time units – similar behavior is observed with the refined meshes. For the coarse mesh, approximately 142 time steps were taken per vortex shedding cycle, while the time step for fine mesh resulted in about 457 time steps per cycle. The measured Strouhal number for the fine mesh computation is 0.175 based upon the shedding period measured over 25 shedding cycles. This is consistent with the published numerical results of $0.172 - 0.173$.[19]

Figure 8.14 shows snapshots of the vorticity and pressure fields on the fine mesh at approximately 150 time units. The close-up of the instantaneous vorticity field shows two distinct secondary attached vortices in the wake of the cylinder. The color map for the vorticity was chosen specifically to delineate positive and negative vorticity as illustrated by the region of essentially zero vorticity that separates the upper and lower regions of the vortex street. The pressure field shows the regions of low pressure that cor-

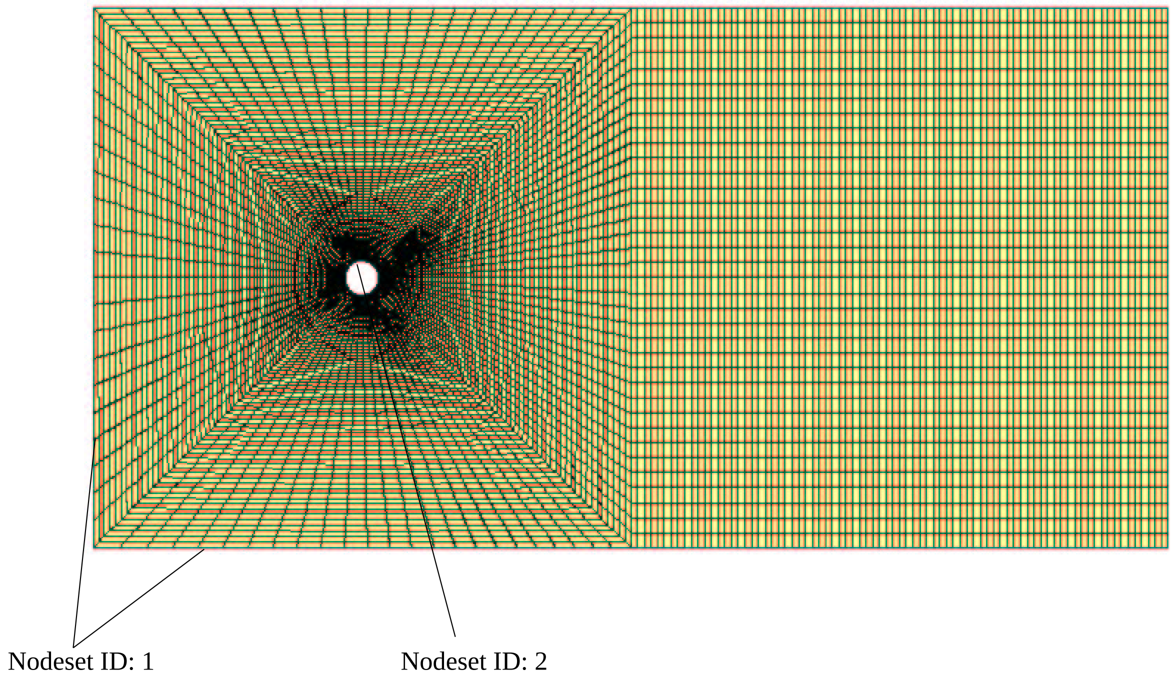


Figure 8.11: Coarse mesh for cylinder vortex shedding.

respond to the regions of high vorticity. The decay in vortex strength due to viscous diffusion is clear in the downstream section of the vortex street. The low-pressure regions associated with the vortex centers also exhibit a reduced amplitude in the downstream wake.

Table 8.2: Vortex shedding mesh parameters. (Δr_i: radial mesh spacing at the cylinder surface, Δr_o: radial mesh spacing at $x = \pm 4$.)

Case	No. of Elements	No. of Unknowns	Δr_i	Δr_o	Time Step
Coarse	2800	8584	0.0375	0.3375	0.0400
Medium	11200	33968	0.0188	0.1688	0.0200
Fine	25200	76152	0.0100	0.1125	0.0125

```
title
Vortex shedding - Re=100

analyze
   solve     3
   temp      0
   species   0
   react     0
   nstep     2000
   plti      100
   prti      10000
   pltype    0
   dump      1000
   ttyi      1
   icset
      set_u  1.0
      set_v  0.0
   ;
   btd       1
   fct       1
   divu      1.0e-10
   deltat    5.00e-2
   dtchk       -1
   cfl        1.0
   term    500.00
end

material 1
   rho        1.0
   mu         1.0e-2
end

ubc
  nodeset    1 1.0  0
  nodeset    2 0.0  0
end

vbc
  nodeset    1 0.0 0
  nodeset    2 0.0 0
end

ppesol    64
   itmax 500
   itchk   1
   eps   1.0e-5
   wrt     1
   hist    0
   stab   20
   beta   0.025
   novec  10
end
```

```
momsol 1
   itmax 20
   itchk  2
   eps   1.0e-5
   wrt   0
end

ndhist 1
   nstep  1
   st 2546 en 2570
end

exit
```

Figure 8.12: Control file for the $Re = 100$ vortex shedding problem.

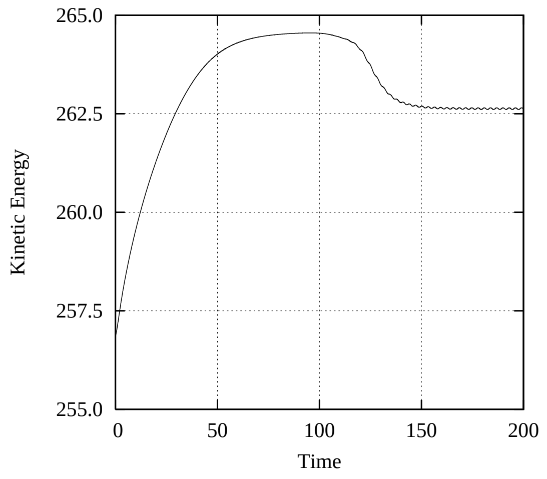
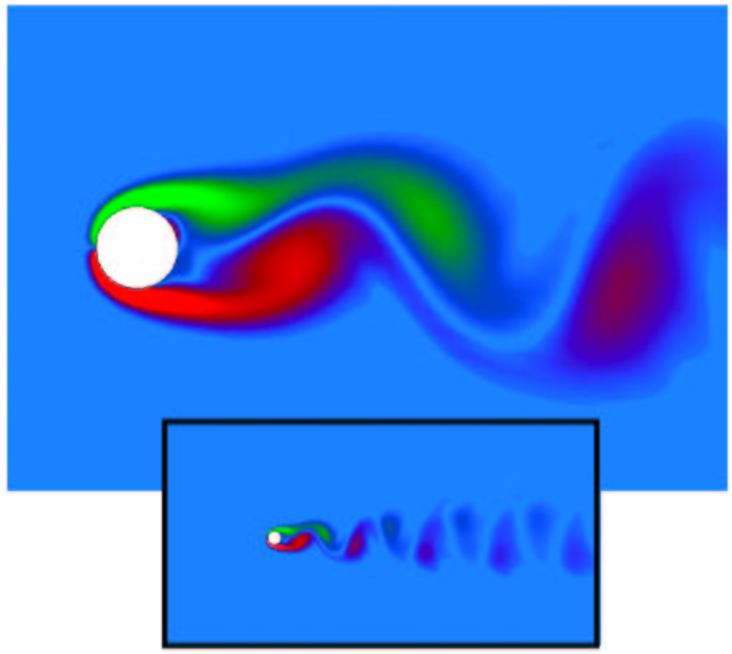


Figure 8.13: Kinetic energy time history for the coarse mesh.


a)

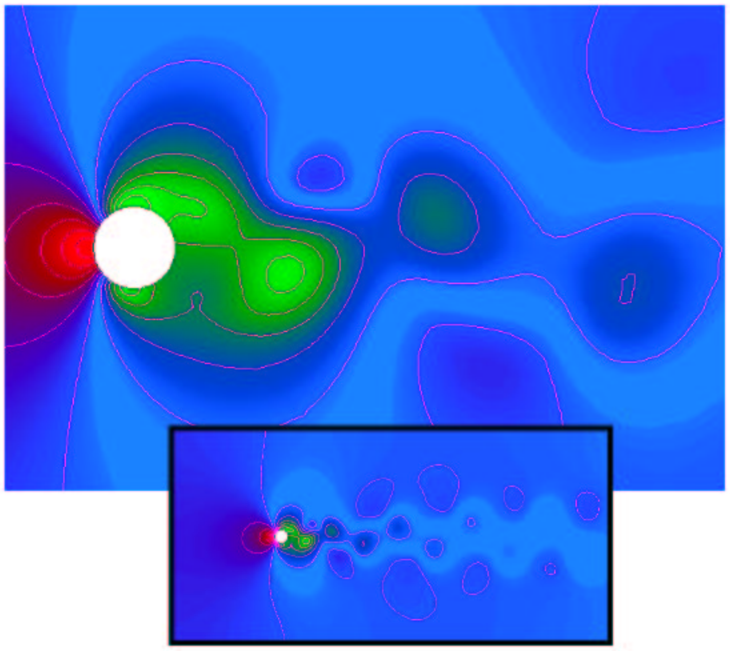

b)

Figure 8.14: $Re = 100$ flow past a circular cylinder at 150 time units: a) vorticity field, b) pressure field.

8.4 Momentum-Driven Jet

This example computation shows the starting vortical structure associated with a heated slot jet entering a relatively cold, quiescent fluid and the resulting classical shear instability phenomena known as the Kelvin-Helmoltz instability. The parameters for the momentum-driven jet are prescribed so that the Reynolds number is $Re = 3561$ based on a 15 mm slot width and $Fr = 326$ ($Fr = v^2/g\beta\Delta T L_c$ where $g = 9.81\ m/s^2$, $\beta\Delta T = 1$, $L_c = 15\ mm$, and $v = 4\ m/s$). Here, the relatively large Froude number indicates that the influence of buoyancy forces is small compared to the inertial forces, i.e., a momentum driven jet.

The grid for the momentum-driven jet is shown in Figure 8.15 where a single plane of symmetry at $x = 0$ is used with the jet half-width $H/2$. In this computation, an energy equation is solved in conjunction with the Navier-Stokes equations using a Boussinesq fluid, i.e., air. The initial conditions consist of an initial div-free velocity field with a free-field temperature of $300K$ and an inlet air jet temperature of $400K$. The working fluid is air with a near-unit Prandtl number $Pr = 0.71$ resulting in a Peclet number $Pe = 2528$ where $Pe = RePr$.

The GILA control file is shown in Figure 8.16. A time-accurate simulation is of interest here, so the default second-order time-weighting parameters (`thetak 0.5`) are used. The sharp gradients in the temperature and velocity fields requires the use of advective flux limiters (`fct 1`) for the momentum and scalar transport equations. The pressure is solved with the SSOR preconditioned conjugate-gradient method and 5 A-conjugate projection vectors (`nvec 5`). Figure 8.17 shows snapshots of the temperature, vorticity and pressure fields.

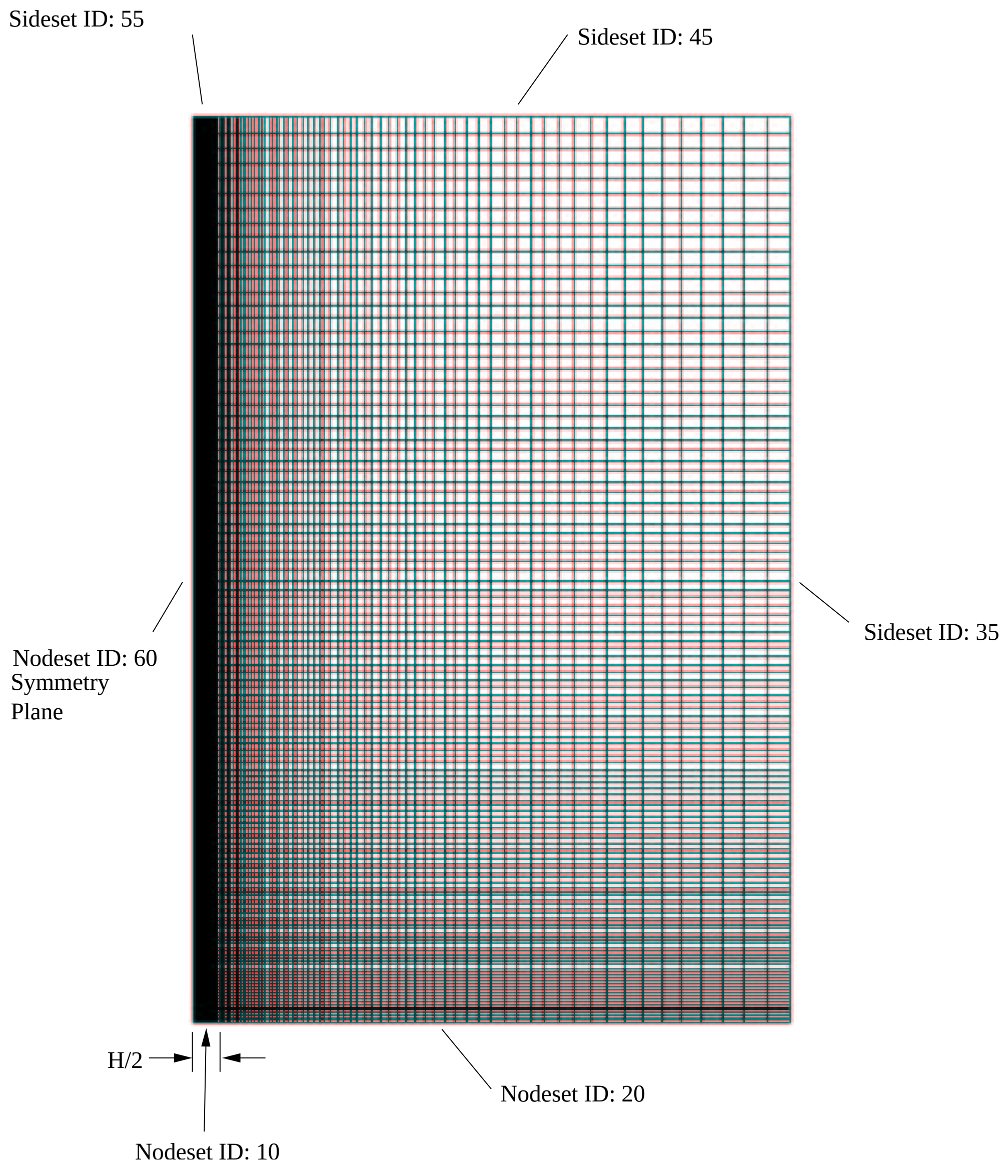


Figure 8.15: Momentum-driven jet mesh with 11250 elements and 11466 nodes.

```
title
2-D hot momentum jet

{ Analysis parameters: }
analyze
   solve     3
   temp      1
   nstep  1000
   plti    100
   prti   9999
   pltype    1
   prtlev    0
   icset
      set_u  0.0
      set_v  0.0
      set_t  300.0
   ;
   ttyi    1
   btd     1
   fct     1
   divu    1.0e-10
   deltat  5.0e-4
   dtchk  -1
   dtscal  1
   thetak  0.5
   thetab  0.5
   term    10.00
end

{ Ideal gas }
material   1
   model   1
   rho     1.177
   mu      1.685e-5
   alpha   1.685e-5
   diff1   1.685e-5
   t_ref   300.0
   beta    3.33e-3
   gx      0.0
   gy     -9.81
   a       100.0
   k       1.4
   e       4.0e+3
   r       0.287
  q0       1.0
end

{ u-velocity BC's: }
ubc
  nodeset    10 0.0 0
  nodeset    60 0.0 0
  nodeset    20 0.0 0
end
```

```

{ v-velocity BC's: }
vbc
  nodeset    10 1.0 0
  nodeset    20 0.0 0
end

{ Temperature BC's: }
tbc
  nodeset    10 400.0 0
  nodeset    20 300.0 0
end

{ Outflow BC's: }
outflow
  sideset    55
  sideset    45
  sideset    35
end

{ PPE solver: }
ppesol      64
   itmax  1000
   itchk     1
   novec     5
   wrt       1
   stab     30
   beta   0.1000
   eps    1.0e-5
end

{ Momentum solver: }
momsol      1
   itmax   50
   itchk    2
   wrt      0
   eps     1.0e-5
end

{ End-of-control file }
exit
```

Figure 8.16: Control file for the momentum-driven jet.

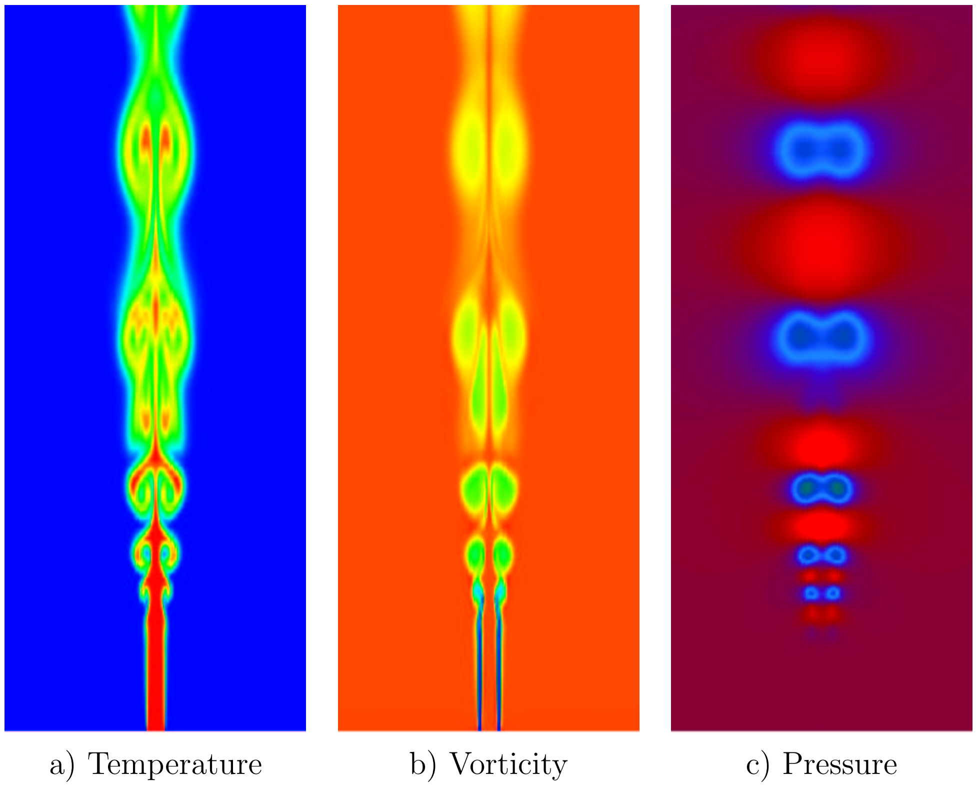


Figure 8.17: Snapshot of a) temperature field, b) z-vorticity field and c) pressure for the momentum driven jet. (The temperature, vorticity and pressure fields have been reflected about the vertical centerline for presentation purposes.)

8.5 Post & Plate

Juncture flows are of interest in numerous industrial applications, and in particular, exterior flows around vehicles. The presence of geometrical junctures can lead to the generation of longitudinal vortices that pass on each side of the juncture in counter-rotating directions. These types of flow configurations occur in turbomachinery, blade and end-wall flows, aircraft wing - body junctures, and ship/hull appendages. A recent experimental investigation of this type of flow configuration may be found in Devnport and Simpson.[17]

The flow past a plate with an attached cylinder is considered here as a simplified prototype juncture configuration for a Reynolds number of 100 based upon the cylinder diameter. The computational domain is chosen such that $-4.5 \leq x \leq 6.5$, $-2.5 \leq y \leq 2.5$, and $0 \leq z \leq 2$ with a cylinder of unit diameter at the origin oriented in the z-direction, and a plate in the $x-y$ plane at $z = 0$. The boundary conditions are specified as $u = 1, v = 0, w = 0$ at the upstream inlet, with no-slip and no-penetration boundary conditions on the plate and post. A symmetry condition is imposed at the upper boundary ($z = 2$) so that only the x and y-velocity components vary in the $x-y$ plane. Homogeneous natural boundary conditions are used on the $y - z$ planes of the domain. For this example, a coarse mesh with 9120 nodes and 7840 elemensts as shown in Figure 8.18 is used. The GILA control file for this problem is shown in Figure 8.19.

In Figure 8.20, snapshots of the pressure, helicity ($\omega \cdot \mathbf{u}$), and enstrophy ($\omega \cdot \omega/2$) at 300 time units are shown in the left column for the medium mesh resolution. In the right column of Figure 8.20, the three vorticity fields are shown. The flow field in the symmetry plane at $z = 2$ exhibits behavior strikingly similar to the two-dimensional von Karman vortex street discussed above. The pressure isosurfaces illustrate the three-dimensional nature of the flow field and highlight the stagnation regions at the leading edge of both the plate and post.

The presence of the juncture as well as the plate itself, leads to a strongly three-dimensional vorticity field particularly in the downstream section of the flow. The isosurfaces of x-vorticity illustrate the orientation of the vorticity with the primary flow direction at the post-plate juncture. Inspection of the helicity field confirms that the flow field contains significant three-dimensional structure, and identifies the regions of the juncture flow where the longitudinal vortices are aligned with the velocity field. The blue and gold helicity isosurfaces on either side of the cylinder-plate juncture clearly identify the presence of counter-rotating longitudinal vortices oriented with the primary flow direction. The fact that the helicity changes sign on the upstream side of the juncture suggests that the flow field is subjected to extreme strain-rates in this region.

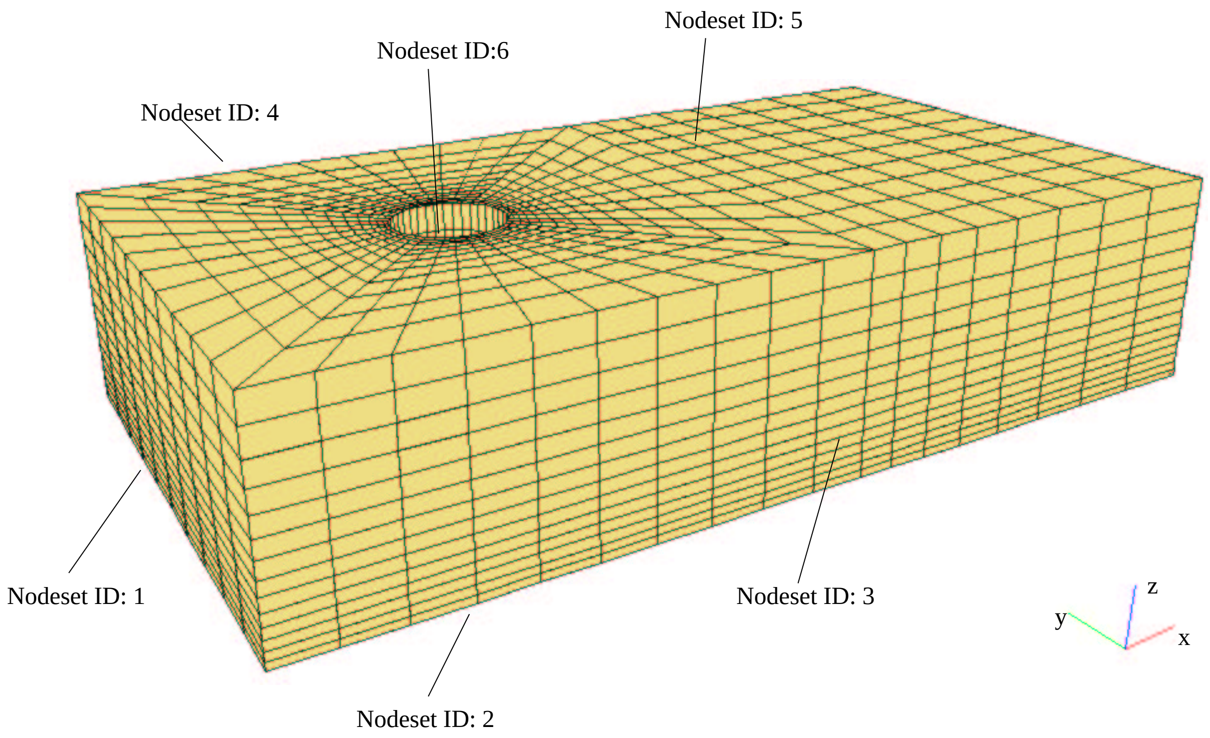


Figure 8.18: Post and plate Mesh.

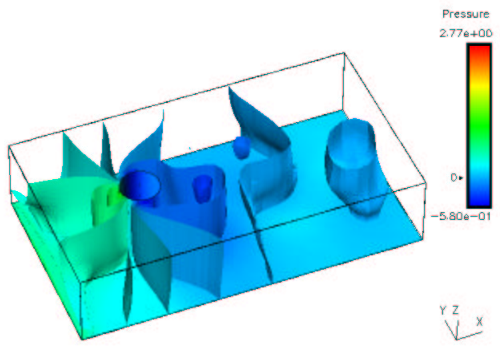
The y-vorticity isosurfaces correlate to the development of the boundary layer from the leading edge of the plate, and clearly identify the wake region near the plate downstream of the cylinder. The y-vorticity values on the upstream surface of the cylinder also indicates the presence of several counter-rotating vortices centered along the stagnation line of the cylinder. In contrast, to the strongly three-dimensional x and y-vorticity fields, the z-vorticity isosurfaces are relatively two-dimensional except near the plate surface where the influence of the boundary layer is significant.

Isosurfaces of the enstrophy field are also shown in Figure 8.20. The enstrophy isosurfaces show a three-dimensional trough in the downstream region of the post that corresponds to the trough in the y-vorticity isosurfaces. The measured Strouhal number (based upon the y-velocity fluctuations in the symmetry plane) is 0.17 which agrees well with the two-dimensional vortex shedding calculations.

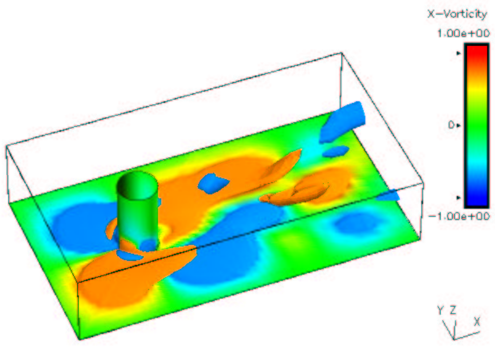
```
title
Post w. Re=100

analyze
   solve      3
   nstep   2000
   plti      50
   prti    9999
   pltype     1
   ttyi       1
   dump    1000
   icset
      set_u 1.0
      set_v 0.0
      set_w 0.0
   ;
   btd      1
   fct      1
   divu     1.0e-8
   term     1.000000E+03
   deltat   0.100000E+00
   dtscal   1.0
   dtchk     -1
end

material    1
   model    1
   rho      1.0
   cp       1.0
   mu       1.0e-2
   k        1.0e-2
   t_ref    0.0
   beta     0.5
   gx       0.0
   gy       0.0
   gz       0.0
end

ubc
  nodeset 1 1.0 0
  nodeset 3 1.0 0
  nodeset 4 1.0 0
  nodeset 2 0.0 0
  nodeset 6 0.0 0
end
```

```
vbc
  nodeset 1 0.0 0
  nodeset 2 0.0 0
  nodeset 3 0.0 0
  nodeset 4 0.0 0
  nodeset 5 0.0 0
  nodeset 6 0.0 0
end

wbc
  nodeset 1 0.0 0
  nodeset 2 0.0 0
  nodeset 5 0.0 0
  nodeset 6 0.0 0
end

ppesol          64
   itmax      300
   itchk        2
   novec       10
   stab        30
   beta       1.0e-3
   omega      1.00
   eps        1.0e-5
   wrt        1
end

momsol        1
   itmax      100
   itchk        2
   eps        1.0e-5
   wrt        1
end

exit
```

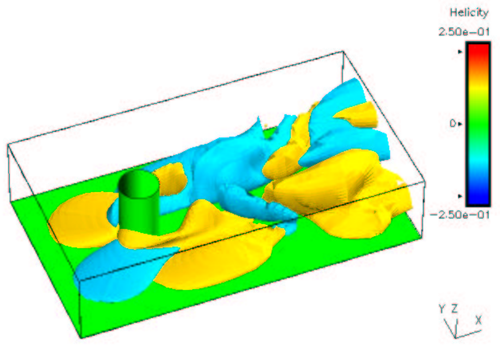
Figure 8.19: Post and plate control file.



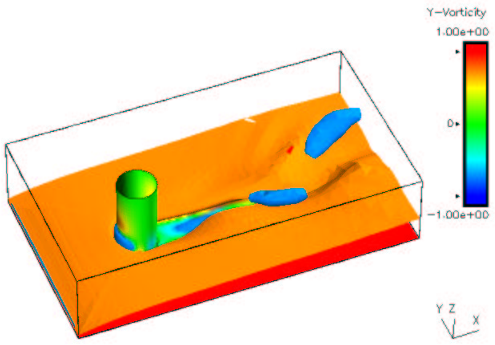
a)



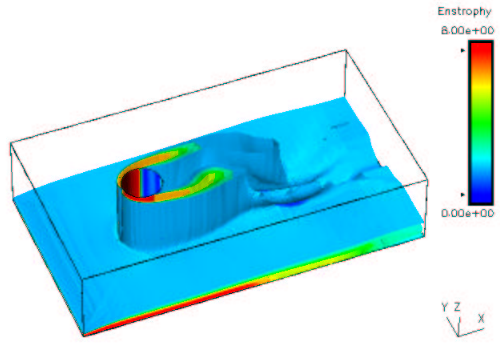
b)



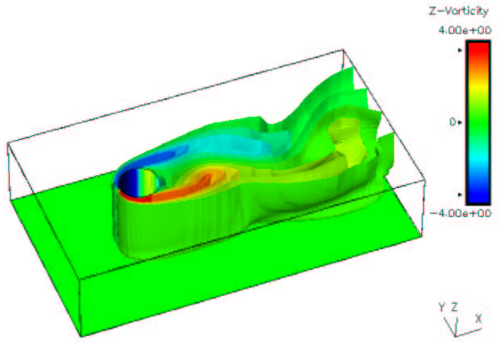
c)



d)



e)



f)

Figure 8.20: $Re = 100$ flow past post and plate at 300 time units. a) Pressure, b) X-Vorticity (isosurface values ± 0.75), c) Helicity (isosurface values ± 0.15), d) Y-Vorticity, (isosurface values ± 0.75), e) Enstrophy, and f) Z-Vorticity field.

Appendix A

GILA Mesh File

The GILA ASCII mesh file is intended to provide a simplified, machine independent alternative to EXODUS II. The ASCII mesh file permits GILA to interface with a number of commercial mesh generation tools. For this reason, the ASCII mesh file has been intentionally simplified to permit easy adaptation of existing neutral file formats produced by commercial mesh generators.

The GILA ASCII mesh file contains a mesh description line, the spatial coordinates of the nodes, element connectivity, and all node and side sets. The mesh description line is limited to 80 characters, must be the first line in the mesh file, and is echoed at run-time to the screen and the human readable output file. Lines in the mesh file may be commented out by using a "C" followed by a blank space, by using #, *, $, or enclosing a region of the mesh file in braces, { }. Because the data in the mesh file is usually generated by an automatic mesh generator, the data in this file is not accepted in a format-free style as in the case of the GILA control file. However, all input in the mesh file is case insensitive.

The mesh file consists of separate sections that proceed in the following order.

1. Mesh description line (80 characters maximum).
2. Header block containing control information.
3. Nodal coordinates.
4. Element connectivity.
5. Node set data.
6. Side set data.

Typically, each section of the mesh file contains a short series of comments describing the contents of the section.

A.1 Mesh Description Line

The first line of the GILA ASCII mesh file is expected to contain the mesh description. A blank line is acceptable, and there are no restrictions on special characters. However, the mesh description line is limited to 80 characters in length. Comment characters, "C", #, *, $, { }, in the description line are treated as a part of the 80 character description. There can be no comment lines before the description line in the mesh file.

A.2 Header Block

The header block follows the mesh description line and consists of a sequence of comment lines that contain control information describing the mesh. The information required in the header block consists of the number of nodes, elements, element blocks, node sets and side sets in the mesh as shown in the table below. All keywords are case insensitive and order-independent with the exception of the *end* keyword that terminates the header block. An example of the header block is shown in Figure A.1.

ASCII Mesh File Header Block

Keyword	**Variable**	**Meaning**
Nnp	Nnp	Specify the number of nodes, Nnp.
Nel	Nel	Specify the number of elements, Nel.
Nnpe	$Nnpe$	Specify the number of nodes-per-element, $Nnpe$.
Ndim	$Ndim$	Specify the number of dimensions, $Ndim$.
Nel_blk	Nel_blk	Specify the number of element blocks, Nel_blk.
Nnd_sets	Nnd_sets	Specify the number of node sets, Nnd_sets.
Nsd_sets	Nsd_sets	Specify the number of side sets, Nsd_sets.
end		Terminate the header block

```
The 80-character description line comes first in the mesh file.
#
# Nnp      1852
# Nel      1760
# Nnpe     4
# Ndim     2
# Nel_blk  1
# Nnd_sets 3
# Nsd_sets 3
# end
```

Figure A.1: Example description line and header block for ASCII mesh file.

A.3 Nodal Coordinates

Nnp nodal coordinates are required in this section of the mesh file. In the case of a 2-D analysis ($Ndim = 2$), the z-coordinate is ignored in the mesh file. The format specifications for the nodal coordinates are shown in the table below.

Nodal Coordinates

Columns	Format	Description
1-8	I8	node number
14-33	E20.0	x-coordinate.
34-53	E20.0	y-coordinate.
54-73	E20.0	z-coordinate. (Ignored for 2-D, $Ndim = 2$)

A.4 Element Connectivity

The node numbers and material numbers associated with Nel elements are required in this section of the mesh file. For 2-D calculations, $Ndim = 2$, GILA ignores the last 4 nodes numbers if they are present in the connectivity. The table below shows the format specifications for the element connectivity.

Element Connectivity

Columns	Format	Description
9-13	I5	material number.
14-21	I8	local node #1.
22-29	I8	local node #2.
30-37	I8	local node #3.
38-45	I8	local node #4.
...		...
70-77	I8	local node #8.
		(Nodes 5-8 are ignored for $Ndim = 2$)

A.5 Node Sets

The node set section of the mesh file consists of three parts that describe the number of node sets in the mesh, the number of nodes in each node set, and the node lists for each node set. The following tables outline the formats required for each part of the node set section of the mesh file. In Part 1, the number of node sets, Nnd_sets is specified. Immediately following, in Part 2, is a list containing Nnd_sets lines of input that contain the node set id or node set number, and the number of nodes associated with each node set id. In Part 3, Nnd_sets lists of input follows. Each list contains the local node counter and the node numbers associated with the node set id's listed in Part 2. A short sample of this section of the input file is shown in Figure A.2. In this example, comments are used to delineate the three sections of the input data.

Node Sets - Part 1

Columns	Format	Description
1-10	I10	Number of node sets in the mesh file.

Node Sets - Part 2

Columns	Format	Description
1-10	I10	Integer node set identifier for the node set.
11-20	I10	Number of nodes in the node set.

Node Sets - Part 3

Columns	Format	Description
1-10	I10	Node counter of the current node.
11-20	I10	Node number for the current node.

A.6 Side Sets

The input section for side sets also consists of three parts that describe the number of side sets, the number of segments in each side set, and the side lists for each side set. In this section, the canonical, finite element side-ordering is used to identify element sides. The following tables outline the formats required for each part of the side set section of the mesh file. In Part 1, the number of side sets, Nsd_sets is specified. Immediately following, in Part 2, is a list containing Nsd_sets lines of input that contain the side set id or number, and the number of side segments associated with each side set id. In Part 3, Nsd_sets lists of input follows. Each side set list contains the element number and element side number associated with the side set id's listed in Part 2 of the side set data.

Side Sets - Part 1

Columns	**Format**	**Description**
1-10	I10	Number of side sets in the mesh file.

Side Sets - Part 2

Columns	**Format**	**Description**
1-10	I10	Integer side set identifier for the side set.
11-20	I10	Number of elements in the side set.

Side Sets - Part 3

Columns	**Format**	**Description**
1-10	I10	Element number for the current side set segment.
11-20	I10	Side number for the current segment.

The canonical local node numbering scheme is shown with the side numbering in Figure A.3. The following tables show the local node ordering corresponding to each side number for the 2-D quadrilateral and 3-D hexahedral element. A sample side set section of the mesh file is shown in Figure A.4. Note that for each side set segment, the segment lists consist of the element number and the associated side number based upon the canonical local node ordering.

```
#
# 3 Node-sets
         3
#
# Node set   Number of Nodes
         1       118
         2        56
         3       175
#
# Node Set Number : 1
# No. of Nodes    : 118
#
         1       211
         2       190
       ...
       118       861
#
# Node Set Number : 2
# No. of Nodes    : 56
#
         1        21
         2        42
       ...
        56       841
#
# Node Set Number : 3
# No. of Nodes    : 175
#
         1       211
         2       190
       ...
       175       861
```

Figure A.2: Example node set section of the ASCII mesh file.

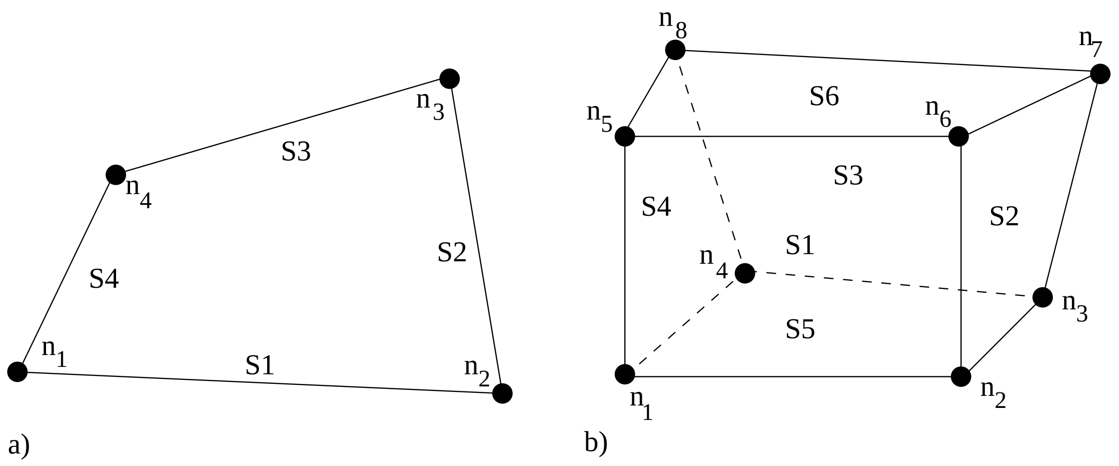


Figure A.3: Canonical node and side numbering for the a) 2-D quadrilateral element, and b) the 3-D hexahedral element.

Side Numbers - 2-D Quadrilateral

Side	Node-1	Node-2
Side-1 (S1)	1	2
Side-2 (S2)	2	3
Side-3 (S3)	3	4
Side-4 (S4)	4	1

Side Numbers - 3-D Hexahedral Element

Side	Node-1	Node-2	Node-3	Node-4
Side-1 (S1)	1	2	6	5
Side-2 (S2)	2	3	7	6
Side-3 (S3)	3	4	8	7
Side-4 (S4)	4	1	5	8
Side-5 (S5)	1	4	3	2
Side-6 (S6)	5	6	7	8

```
#
# 2 Side-sets
         2
# Side Set  Number of Sides
        15          8
        65         50
#
# Side Set Number : 15
# No. of Segments : 8
#
         1          1
         2          1
       ...
        32          1
#
# Side Set Number : 65
# No. of Segments : 50
#
         1          4
         9          4
       ...
       393          4
```

Figure A.4: Example side set section of the ASCII mesh file.

A.7 Sample ASCII Mesh File

The following sample ASCII mesh file is provided to show the overall structure of the mesh file, the use of comments to delineate the sections of the mesh file, and the structure of the individual sections of the mesh file.

```
Sample GILA ASCII mesh file
#
# Nnp      1681
# Nel      1600
# Nnpe     4
# Ndim     2
# Nel_blk  1
# Nnd_sets 1
# Nsd_sets 6
# end
#
# ===== Nodal Coordinates =====
#
       1      5.0000000000000e-01-5.0000000000000e-01
       2      5.0000000000000e-01 5.0000000000000e-01
     ...
    1681     -4.7499999403954e-01 4.7499999403954e-01
#
# ===== Element Connectivity =====
#
            1       1       3     161     160
            1       3       4     162     161
     ...
            1    1681      81      42      83
#
# 1 Node-sets
         1
# Node Set  Number of Nodes
        10        41
#
# Node Set Number : 10
# No. of Nodes    : 41
#
         1       122
         2       123
       ...
        41         1
```

```
#
# 6 Side-sets
         6
# Side Set  Number of Sides
        15          8
        65         50
        55          8
        25         22
        35         50
        45         22
#
# Side Set Number : 15
# No. of Segments : 8
#
         1          1
         2          1
       ...
        32          1
#
# Side Set Number : 65
# No. of Segments : 50
#
         1          4
         9          4
       ...
       393          4
#
# Side Set Number : 55
# No. of Segments : 8
#
       393          3
       394          3
       ...
       400          3
#
# Side Set Number : 25
# No. of Segments : 22
#
       401          1
       402          1
       ...
       422          1
```

```
#
# Side Set Number : 35
# No. of Segments : 50
#
       422         2
       444         2
       ...
      1500         2
#
# Side Set Number : 45
# No. of Segments : 22
#
      1479         3
      1480         3
       ...
      1500         3
```

Appendix B

Data Visualization Interfaces

B.1 GRIZ - Binary Plot Files

GILA can output several forms of graphics files. One of the primary file formats is the binary, failed graphics data files and time history files that are compatible with GRIZ[18] and THUG.[61] GRIZ is used for visualizing snapshots of the entire flow-field (state data) or generating animations of the time varying flow-field data, while THUG[61] is used for interrogating time history data at a moderate number of mesh points. Typically, the state data is written at relatively large time intervals while the time history data is recorded at each time step.

GRIZ and THUG are general purpose scientific visualization tools for finite element codes, and they support analysis codes for both computational fluid dynamics and computational mechanics. Because of their general purpose nature there is a required translation from the primitive variables that GILA writes to the graphics data files to variables which can be displayed in GRIZ and THUG. Table B.1 shows the mapping from GILA's 2-D primitive variables to GRIZ and THUG variables. Table B.3 shows the mapping from 3-D GILA variables to GRIZ variables.

In GRIZ, the character strings associated with certain variables may have to be reset to reflect the correct GILA variable, e.g., the x-acceleration variable in GRIZ is actually the x-component of vorticity in a 3-D GILA database. For 2-D GILA state databases, the z-velocity and x-acceleration are omitted, and the z-vorticity and stream function are computed and output at the nodes instead.

THUG provides direct support for GILA and the variable mapping is handled automatically. However, THUG currently requires the use of the default variable names for GILA variables. For example, to display the global divergence error ($\nabla \cdot \mathbf{u}$), the global THUG variable, *rigid*, must be specified in the *plot* command in THUG. Table B.3 shows the relationship between

GILA global time history variables and THUG global time history variables.

GILA State Variables	GRIZ State Variables	GILA Time History Variables	THUG Time History Variables
unused	x-displacement	unused	x-displacement
unused	y-displacement	unused	y-displacement
unused	z-displacement	unused	z-displacement
x-velocity	x-velocity	x-velocity	x-velocity
y-velocity	y-velocity	y-velocity	y-velocity
unused	z-velocity	unused	z-velocity
unused	x-acceleration	unused	unused
z-vorticity	y-acceleration	unused	unused
stream function	z-acceleration	unused	unused
temperature	temperature	temperature	x-acceleration
pressure	pressure	pressure	N/A
turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)
dissipation rate	dissipation rate	dissipation rate	dissipation rate

Table B.1: 2-D GILA State and Time History Variables

GILA State Variables	GRIZ State Variables	GILA Time History Variables	THUG Time History Variables
unused	x-displacement	unused	x-displacement
unused	y-displacement	unused	y-displacement
unused	z-displacement	unused	z-displacement
x-velocity	x-velocity	x-velocity	x-velocity
y-velocity	y-velocity	y-velocity	y-velocity
z-velocity	z-velocity	z-velocity	z-velocity
x-vorticity	x-acceleration	unused	unused
y-vorticity	y-acceleration	unused	unused
z-vorticity	z-acceleration	unused	unused
temperature	temperature	temperature	x-acceleration
pressure	pressure	pressure	unused
turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)
dissipation rate	dissipation rate	dissipation rate	dissipation rate

Table B.2: 3-D GILA State and Time History Variables

GILA Global Time History Variables	THUG Global Time History Variables
RMS Divergence Error: $\sqrt{\frac{\sum_{i=1}^{nel}(C_i^T \mathbf{u}_i)^2}{nel}}$	Rigid Body x-displacement
Total Kinetic Energy: $\frac{1}{2}\mathbf{u}^T M_L \mathbf{u}$	Rigid Body y-displacement

Table B.3: GILA Global Time History Variables

Bibliography

[1] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell, *An adaptive projection method for the incompressible euler equations*, in Eleventh AIAA Computational Fluid Dynamics Conference, AIAA, 1993, pp. 530–539.

[2] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *A conservative adaptive projection method for the variable density incompressible navier-stokes equations*, Journal of Computational Physics, 142 (1998), pp. 1–46.

[3] A. S. Almgren, J. B. Bell, and W. Y. Crutchfield, *Approximate projection methods: Part i. inviscid analysis*, SIAM Journal on Scientific Computing, 22 (2000), pp. 1139–1159.

[4] A. S. Almgren, J. B. Bell, and W. G. Szymcyzk, *A numerical method for the incompressible navier-stokes equations based on an approximate projection*, SIAM Journal for Scientific Computing, 17 (1996), pp. 358–369.

[5] B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schonug, *Experimental and theoretical investigation of backward facing step flow*, Journal of Fluid Mechanics, 127 (1983), pp. 473–496.

[6] S. T. Barnard and H. D. Simon, *A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems*, in 6th SIAM Conference on Parallel Processing for Scientific Computing, 1993, pp. 711–718.

[7] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible navier-stokes equations*, Journal of Computational Physics, 85 (1989), pp. 257–283.

[8] T. Belytschko, J. S. Ong, W. K. Liu, and J. M. Kennedy, *Hourglass control in linear and nonlinear problems*, Computer Methods in Applied Mechanics and Engineering, 43 (1984), pp. 251–276.

[9] D. L. Brown and M. L. Minion, *Performance of under-resolved two-dimensional incompressible flow simulations*, Journal of Computational Physics, 122 (1995), pp. 165–183.

[10] A. J. Chorin, *Numerical solution of the navier-stokes equations*, Mathematics of Computations, 22 (1968), pp. 745–762.

[11] M. A. Christon, *Visualization methods for high-resolution, transient, 3-D, finite element simulations*, in International Workshop on Visualization, Paderborn, Germany, January 1994. (*invited paper*, LLNL UCRL-JC-119879).

[12] ——, *HYDRA: A finite element computational fluid dynamics code - User Manual*, Tech. Rep. UCRL-MA-121344, Lawrence Livermore National Laboratory, June 1995.

[13] ——, *Dealing with pressure – solution strategies for the time-dependent navier-stokes equations*, International Journal for Numerical Methods in Fluids, 38 (2002), pp. 1177–1198.

[14] M. A. Christon and T. E. Spelce, *Visualization of high resolution three-dimensional nonlinear finite element analyses*, in Visualization '92, Boston, MA, 1992, IEEE, pp. 324–331. (LLNL UCRL-JC-110110).

[15] M. A. Christon and T. E. Voth, *Lestats user's manual*, Tech. Rep. in preparation, Sandia National Laboratories, Albuquerque, New Mexico, January 2003.

[16] M. A. Christon and R. Whirley, *The 1991 MPCI Yearly Report*, Lawrence Livermore National Laboratory, UCRL-ID-10722 ed., 1991, ch. Explicit Structural Analysis in a Concurrent Computing Environment.

[17] W. J. Devenport and R. L. Simpson, *Time-dependent and time-averaged turbulence structure near the nose of a wing-body junction*, Journal of Fluid Mechanics, 210 (1990), pp. 23–55.

[18] D. J. Dovey, T. E. Spelce, and D. E. Speck, *Griz finite element analysis results visualization for unstructured grids*, Tech. Rep. UCRL-MA-115696 Rev. 1, Lawrence Livermore National Laboratory, Livermore, California, March 1996.

[19] M. S. Engelman and M.-A. Jamnia, *Transient flow past a circular cylinder: a benchmark solution*, International Journal for Numerical Methods in Fluids, 11 (1990), pp. 985–1000.

[20] C. Farhat and M. Lesoinne, *Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics*, International Journal for Numerical Methods in Engineering, 36 (1993), pp. 745–764.

[21] C. J. Freitas, *Perspective: Selected benchmarks from commercial cfd codes*, Journal of Fluids Engineering, 117 (1995), pp. 208–218.

[22] D. K. Gartling, *A test problem for outflow boundary conditions - flow over a backward-facing step*, International Journal for Numerical Methods in Fluids, 11 (1990), pp. 953–967.

[23] G. L. Goudreau and J. O. Halquist, *Recent developments in large-scale finite element lagrangian hydrocode technology*, Computer Methods in Applied Mechanics and Engineering, 33 (1982), p. 725.

[24] P. M. Gresho, *On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 1: Theory*, International Journal for Numerical Methods in Fluids, 11 (1990), pp. 587–620.

[25] P. M. Gresho and S. T. Chan, *On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 2: Implementation*, International Journal for Numerical Methods in Fluids, 11 (1990), pp. 621–659.

[26] ——, *Projection 2 goes turbulent – and fully implicit*, preprint International Journal for Computational Fluid Dynamics, (1996). (LLNL UCRL-JC-123727).

[27] P. M. Gresho, S. T. Chan, M. A. Christon, and A. C. Hindmarsh, *A little more on stabilized q1q1 for transient viscous incompressible flow*, International Journal for Numerical Methods in Fluids, 21 (1995), pp. 837–856.

[28] P. M. Gresho, S. T. Chan, R. L. Lee, and C. D. Upson, *A modified finite element method for solving the time-dependent, incompressible navier-stokes equations. part 1: Theory*, International Journal for Numerical Methods in Fluids, 4 (1984), pp. 557–598.

[29] ——, *A modified finite element method for solving the time-dependent, incompressible navier-stokes equations. part 2: Applications*, International Journal for Numerical Methods in Fluids, 4 (1984), pp. 619–640.

[30] P. M. Gresho, R. L. Lee, and R. Sani, *On the time-dependent solution of the Navier-Stokes equations in two and three dimensions*, vol. 1, Pineridge Press, Swansea, U.K, 1980, p. 27.

[31] P. M. Gresho and R. L. Sani, *On pressure boundary conditions for the incompressible navier-stokes equations*, International Journal for Numerical Methods in Fluids, 7 (1987), pp. 1111–1145.

[32] P. M. Gresho and R. L. Sani, *Incompressible flow and the finite element method, Advection-diffusion and isothermal laminar flow*, John Wiley & Sons, Chicester, England, 1998.

[33] J.-L. Guermond, *Some implementations of projection methods for navier-stokes equations*, Mathematical Modelling and Numerical Analysis, 30 (1996), pp. 637–667.

[34] ——, *A convergence result for the approximation of the navier-stokes equations by an incremental projection method*, C. R. Acad. Sci. Paris, 325 (1997), pp. 1329–1332.

[35] J.-L. Guermond and L. Quartapelle, *Calculation of incompressible viscous flow by an unconditionally stable projection fem*, Journal of Computational Physics, 132 (1997), pp. 12–23.

[36] J.-L. Guermond and L. Quartapelle, *On stability and convergence of projection methods based on pressure poisson equation*, International Journal for Numerical Methods in Fluids, 26 (1998), pp. 1039–1053.

[37] ——, *On the approximation of the unsteady navier-stokes equations by finite element projection methods*, Numerische Mathematik, 80 (1998), pp. 207–238.

[38] J. C. Heinrich, S. R. Idelsohn, E. Onate, and C. A. Vionnet, *Boundary conditions for finite element simulations of convective flows with artificial boundaries*, International Journal for Numerical Methods in Fluids, 39 (1996), pp. 1053–1071.

[39] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, Tech. Rep. SAND92-1460, Sandia National Laboratories Report, September 1992.

[40] ——, *The Chaco user's guide - version 1.0*, Tech. Rep. SAND93-2339, Sandia National Laboratories Report, October 1993.

[41] ——, *Multidimensional spectral load balancing*, Tech. Rep. SAND93-0074, Sandia National Laboratories Report, January 1993.

[42] ——, *A multilevel algorithm for partitioning graphs*, Tech. Rep. SAND93-1301, Sandia National Laboratories Report, October 1993.

[43] C. Hoover, M. A. Christon, R. Whirley, and A. J. De Groot, *The 1992 MPCI Yearly Report*, Lawrence Livermore National Laboratory, UCRL-ID-10722-92 ed., 1992, ch. Explicit Nonlinear Structural Dynamics Models for Massively Parallel Computers.

[44] T. J. R. Hughes, *The Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[45] J. V. Kan, *A second-order accurate pressure-correction scheme for viscous incompressible flow*, SIAM Journal for Scientific and Statistical Computing, 7 (1986), pp. 870–891.

[46] D. R. Kincaid, T. C. Oppe, and D. M. Young, *Itpackv 2d user's guide*, tech. rep., Center for Numerical Analysis, The University of Texas at Austin, May 1989.

[47] O. M. Knio, H. N. Najm, and P. S. Wyckoff, *A semi-implicit numerical scheme for reacting flow. ii. stiff operator-split formulation*, pre-print submitted to Journal of Computational Physics, (1999).

[48] W. K. Liu and T. Belytschko, *Efficient linear and nonlinear heat conduction with a quadrilateral element*, International Journal for Numerical Methods in Engineering, 20 (1984), pp. 931–948.

[49] W. K. Liu, J. S. Ong, and R. A. Uras, *Finite element stabilization matrices*, Computer Methods in Applied Mechanics and Engineering, 53 (1985), pp. 13–46.

[50] B. N. Maker, R. M. Ferencz, and J. O. Halquist, *Nike3d, a nonlinear, implicit, three-dimensional code for solid and structural mechanics - uers's manual*, Tech. Rep. UCRL-MA-105268, Lawrence Livermore National Laboratory, 1991.

[51] M. L. Minion, *A projection method for locally refined grids*, Journal of Computational Physics, 127 (1996), pp. 158–178.

[52] M. L. Minion and D. L. Brown, *Performance of under-resoloved two-dimensional incompressible flow simulations, ii*, Journal of Computational Physics, 138 (1997), pp. 734–765.

[53] A. Pothen, H. Simon, and K. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.

[54] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider, *A high-order projection method for tracking fluid interfaces in variable density incompressible flows*, Journal of Computational Physics, 130 (1997), pp. 269–282.

[55] W. J. Rider, *Filtering nonsolenoidal modes in numerical solutions of incompressible flows*, Tech. Rep. LA-UR-3014, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1994.

[56] ——, *The robust formulation of approximate projection methods for incompressible flows*, Tech. Rep. LA-UR-3015, Los Alamos National Laboratory, 1994.

[57] ——, *Approximate projection methods for incompressible flow: implementation, variants and robustness*, Tech. Rep. LA-UR-2000, Los Alamos National Laboratory, Los Alamos, New Mexico, July 1995.

[58] W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerutti, and J. I. Hochstein, *Accurate solution algorithms for incompressible multiphase flows*, Tech. Rep. AIAA-95-0699, AIAA, Reno, Nevada, January 1995.

[59] L. A. Schoof and V. R. Yarberry, *Exodus ii: A finite element data model*, Tech. Rep. SAND92-2137, Sandia National Laboratories, Albuqureque, New Mexico, September 1994.

[60] H. D. Simon, *Partitioning of unstructured problems for parallel processing*, Computing Systems in Engineering, 2 (1991), p. 135.

[61] D. E. Speck, *Thug*, Tech. Rep. UCRL-XXXX, Lawrence Livermore National Laboratory, Livermore, California, February 1994.

[62] J. E. Sturtevant, M. A. Christon, P. D. Heermann, and P.-C. Chen, *PDS/PIO: Lightweight libraries for collective parallel I/O*, in SC98 Proceedings, Orlando, Florida, November 1998, IEEE Supercomputing '98.

[63] M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *An adaptive level set approach for two-phase flows*, Journal of Computational Physics, 148 (1999), pp. 81–124.

[64] S. L. Thompson and L. N. Kmetyk, *Hisplt, a history graphics postprocessor*, Tech. Rep. SAND9101767, Sandia National Laboratories, Albuquerque, New Mexico, September 1991.

[65] B. B. Wetton, *Error analysis of pressure increment schemes*, submitted to SINUM, (1998).

[66] R. G. Whirley and B. E. Engelmann, *Dyna3d: A nonlinear, explicit, three-dimensional finite element code for solid and structural mechanics - user manual*, Tech. Rep. UCRL-MA-107254, Rev. 1, Lawrence Livermore National Laboratory, 1993.

[67] XYZ Scientific, Inc., *TrueGrid*.

[68] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, vol. 2, McGraw-Hill Book Company Limited, Berkshire, England, fourth ed., 1991.

Index

a, 73
abc – end, 82
Advection Options, 67
Analysis Parameters, 61
Analysis Title (title), 59
analyze – end, 61
Arrenhius Reaction Constants, 73
average, 63

Backward Facing Step, 95
beta, 72, 79
Boundary conditions, 23, 51
  flux, 55
  heat flux rate, 55
  mass flux rate, 55
  natural, 53
  nodal, 51
  node sets, 51
  outflow, 54
  pressure, 53, 83
  side sets, 51
  traction, 53
btd, 67

cfl, 64
cp, 73

deltat, 63
Derived Force Computation, 85
dforce – end, 85
diff1, 72
divu, 66
drag, 85
dtchk, 65
dump, 71

e, 73
ebc – end, 82
Element
  modified finite element formulation, 31
  reduced-integration operators, 31
en, 80
Entrance Region in a 2-D Duct, 89
eps, 75, 78
epsdt, 66
epshg, 66
Execution, 57

fct, 67
Fluid Models, 17, 71
forces, 85

gamma, 74
GILA Capabilities, 17
GILA Control File, 59
GILA Development, 16
Governing equations, 21
  energy conservation, 23
  mass conservation, 22
  momentum conservation, 21
Grid parameters
  CFL number, **29**
  Reynolds number, **29**
Guide to the GILA User's Manual, 16
gx, 72
gy, 73
gz, 73

hglass, 66
hist, 76, 78

Hourglass Control Options, 65

icfile – end, 85
icfn – end, 84
icset, 67
icwrt, 70
Initial Condition Functions, 84
Initial Condition Input, 85
Initial Conditions, 84
Initial conditions, 23
Introduction, 15
itchk, 75, 77
itmax, 75, 77

k, 72
kbc – end, 82

lcurve – end, 86
lift, 85

mass, 66
material – end, 71
Material Parameters, 71
model, 71
Momentum Equation Solver, 74
Momentum Equation Solver Parameters, 75
Momentum-Driven Jet, 103
momsol – end, 74
mu, 72

ndhist – end, 79
ndisk, 87
Nodal Boundary Conditions, 81
Nodal Time History Blocks, 79
Nodal Time History Parameters, 79
nodeset, 81
novec, 77
nstep, 62, 80

omega, 78
Outflow Boundary Conditions, 83
Output Options, 68

p0, 73
Parallel File System Options, 87
Parallel File Sytem, 86
Parallelism
    communication costs, 41
    domain-decomposition, 37
    domain-decomposition message-passing, 36
    parallel assembly via message passing, 37
    parallel explicit algorithm, 40
pfs – end, 86
plti, 69
pltype, 69
Post & Plate, 107
Post-Processor Interfaces, 18
ppebc – end, 83
ppesol – end, 76
Pre-Processor Interfaces, 18
Pressure Boundary Condition, 83
Pressure Equation Solver Parameters, 77
Pressure Equation Solvers, 76
Pressure Levels, 83
Pressure-Poisson equation
    pressure levels, 56
Pressure-Poisson equation (PPE), 27, 44, 48
Projection method, 43
    projection properties, 44
    semi-implicit projection method, 45
prti, 69
prtlev, 69

q0, 74

r, 73
rbc – end, 82
react, 62
Restarts, 58
rho, 72
Running GILA, 57

Sets
  node, 51
  shell, 51
  side, 51
  solid, 51
sideset, 83, 86
smagc, 80
solve, 61
Source terms, 51
  body forces, 54
  buoyancy forces, 54
species, 62
st, 79
stab, 78
start, 87
Start-up procedure
  explicit time integration, 29
  semi-implicit projection method, 48

t_ref, 72
tbc – end, 82
temp, 62
term, 63
thetaa, 65
thetab, 65
thetaf, 65
thetak, 65
thtype, 70
time averaging, 63
Time integration
  explicit time integration, 27
    domain-decomposition message-passing, 36
    modified finite element formulation, 31
    communication costs, 41
    parallel explicit algorithm, 40
    reduced-integration operators, 31
    stability, 29
    start-up, 29
  projection method, 43
    projection properties, 44
    semi-implicit projection method, 45
    start-up, 48
Time Step and Time Integration Options, 62
ttyi, 70
Turbulence Model Parameters, 80
Turbulence Models, 18, 80

ubc – end, 81

vbc – end, 81
Vortex Shedding, 99
vortics, 70

wbc – end, 82
write initial conditions, 70
write voriticity initial conditions, 70
write vorticity field, 70
wrt, 75, 78

ybc – end, 83

zbc – end, 82

External Distribution:

Prof. Patrick J. Burns
Mechanical Engineering Dept.
Colorado State University
Ft. Collins, CO 80523

Philip M. Gresho
Lawrence Livermore National Laboratory
P.O. Box 808, L-103
Livermore, California 94551

Internal Distribution:

1	MS 0321	W. J. Camp, 09200
1	MS 0310	P. Yarrington, 09230
1	MS 0847	S. Mitchell, 09211
1	MS 1110	D. Womble, 9214
1	MS 0820	P. Chavez, 09232
1	MS 0316	S. Dosanjh, 09233
1	MS 0819	R. Summers, 9231
5	MS 0819	M. Christon, 09231
1	MS 9018	Central Technical Files, 8945-1
2	MS 0899	Technical Library, 09616
1	MS 0612	Review & Approval Desk, 09612 for DOE/OSTI