UCRL-MA-121344

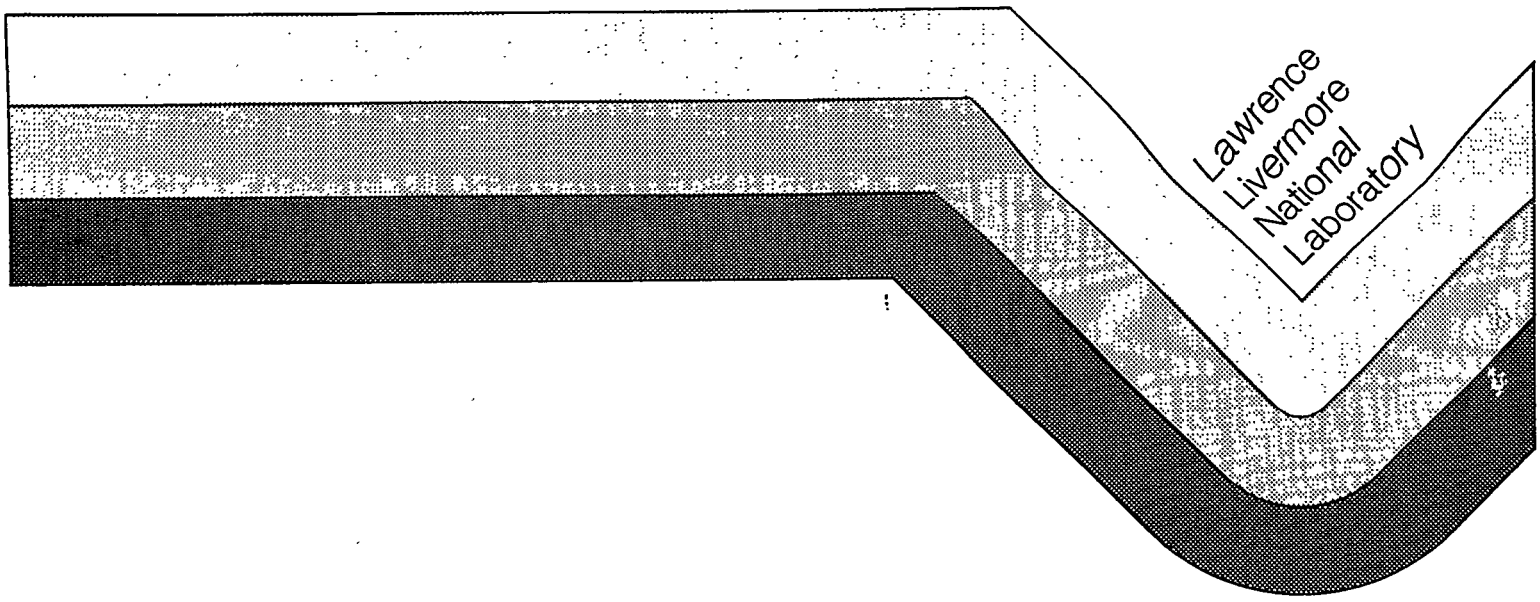
HYDRA

A Finite Element Computational Fluid Dynamics Code

User Manual

Mark A. Christon
Mechanical Engineering
University of California
Lawrence Livermore National Laboratory

June 1995



Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

HYDRA: A Finite Element Computational Fluid Dynamics Code

Mark A. Christon
Lawrence Livermore National Laboratory
Livermore, California 94551 USA

June, 1995

University of California
Lawrence Livermore
National Laboratory

Table of Contents

Preface

HYDRA is a finite element code which has been developed specifically to attack the class of transient, incompressible, viscous, computational fluid dynamics problems which are predominant in the world which surrounds us. The goal for HYDRA has been to achieve high performance across a spectrum of supercomputer architectures without sacrificing any of the aspects of the finite element method which make it so flexible and permit application to a broad class of problems. As supercomputer algorithms evolve, the continuing development of HYDRA will strive to achieve optimal mappings of the most advanced flow solution algorithms onto supercomputer architectures.

The development of HYDRA has drawn, in part, upon over ten years of research in computational fluid dynamics by Phil Gresho, Stevens Chan and their colleagues. Certainly, the work by Helmut Daniels with the PASTIS code (a research version of the Projection-II algorithm) has proven valuable, if in only identifying the obvious memory and performance issues to consider. HYDRA has also drawn upon the many years of finite element expertise constituted by DYNA3D[1] and NIKE3D[2] Certain key architectural ideas from both DYNA3D and NIKE3D have been adopted and further improved to fit the advanced dynamic memory management and data structures implemented in HYDRA.

HYDRA, in its implementation, reflects, to a certain degree, my training and experience with supercomputers beginning with the CYBER 205 and progressing through the CRAY UNICOS "friendly user" period at both the National Center for Supercomputing Applications and at the Pittsburgh Supercomputer Center to the ongoing parallel efforts with the Meiko CS-2 at LLNL. The philosophy for HYDRA is to focus on mapping flow algorithms to computer architectures to try and achieve a high level of performance, rather than just performing a port—a philosophy I adopted from Dan Pryor and Pat Burns during my days as a graduate student at Colorado State University.

I wish to thank Jerry Goudreau and Phil Gresho for their help, encouragement and periodic moral support during the initial development of HYDRA. I would also like to express my appreciation for the help which Brad Maker provided through his discussions with me regarding the Gibbs-Poole-Stockmeyer bandwidth minimization algorithm, the construction of dual grids for the pressure poisson equation, and the many algorithmic similarities and differences between solid and fluid mechanics.

I wish to give special thanks also to Lourdes Placeres and Cathe Forte for their time, patience and expertise in typesetting the draft version of this document and for their advice in defining the manual style. For the efforts of the early HYDRA collaborators, I wish to thank Barb Kornblum, Rose McCallen and Stevens Chan for their input on this document.

Chapter 1

Introduction

The simulation of flow fields about vehicles and in turbomachinery remains one of the computational *grand challenges*[3] for the 1990's. 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 simulate the flow around a vehicle, it is anticipated that more than one million elements will be required to resolve important flow-field features such as shed vortices from 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 optimal mapping of flow-solution algorithms to modern supercomputer architectures.

HYDRA is a finite element code which solves the transient, incompressible, viscous, Navier-Stokes equations, and is based, in part upon the work of Gresho, *et al.* [4,5,6,7]. HYDRA makes use of advanced solution algorithms for both implicit and explicit time integration. The explicit solution algorithm[4,5] introduces lagging phase error, 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 somewhat ameliorates the restrictive diffusive stability limit and raises the order of accuracy of the advective 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 has been the focus of early parallelization efforts with HYDRA.

In the second-order projection algorithm (P-II)[6,7], 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 high order phase speed accuracy, while the lumped mass corrector (a projection to a divergence-free subspace) maintains a divergence 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 super-scalar. The second-order projection algorithm can accurately track shed vortices, and is amenable to the incorporation of either simple or complex (multi-equation) turbulence submodels appropriate for a broad spectrum of applications. HYDRA provides several turbulence models which vary in complexity from a simple algebraic form[8,9] to more traditional multi-equation models[10]. However, because there is no single turbulence model which can solve all flow problems, HYDRA development efforts will continue to evaluate new turbulence models which are appropriate for large scale, complex geometry problems.

1.1 History of HYDRA Development

The early HYDRA development was driven in part by my participation in the early original efforts to parallelize DYNA3D[11,12]. The original idea was to develop the next generation CFD code using the most current algorithmic ideas for incompressible flow coupled with aggressive algorithm mapping for parallelization. The HYDRA code development effort started essentially from scratch to avoid problems with inherited sequential code and antiquated memory management schemes present in many existing

finite element codes. Over the past two years, HYDRA has been under continuous development, and has acted as a test bed not only for investigating the issues involved in mapping finite element codes 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.

HYDRA was developed using standard UNIX tools for source configuration, and source code control. HYDRA was originally constructed to permit the source code to be configured for compilation using either FORTRAN-77 or FORTRAN-90 compilers in an attempt to span multiple supercomputer architectures, e.g., traditional CRAY vector computers and machines like the Thinking Machines CM-5. FORTRAN-77 was used for the vectorized (CRAY) version of HYDRA, while FORTRAN-90 source configuration permitted coexistence and concurrent development of a data parallel version of HYDRA for the Thinking Machines CM-200 and CM-5.

While partially successful, the very long vector characteristics of the CM-200, and CM-5, have pushed HYDRA development away from a data parallel implementation and towards a more portable domain decomposition message passing model (DDMP) which relies upon the FORTRAN-77 part of HYDRA. Current algorithm mapping efforts with HYDRA are being directed towards the Meiko CS-2 because of its superior network bandwidth, fast scalar speed, vector processing abilities, and local disk. However, the general purpose nature of the DDMP approach will also permit future efforts to consider machines such as the CRAY T3D, and the INTEL Paragon.

1.2 HYDRA Capabilities

HYDRA 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, HYDRA also provides both implicit and explicit algorithms for solving the time-dependent scalar advection-diffusion equation.

1.2.1 Pre-Processor Interfaces

The primary mesh generator for HYDRA is currently INGRID[14]. At this time, there is no direct mesh generation support for HYDRA in INGRID. However, the input data for HYDRA has been designed to enable the use of the *dn3d* INGRID output option. One difficulty with this approach is that the user must do some manual editing of the HYDRA input files which can be a bit unwieldy where large grids are concerned. The use of the UNIX utility, *awk*, can simplify the conversion of DYNA3D boundary conditions to HYDRA boundary conditions. It is advisable to generate all 2-D HYDRA meshes in the *x-y* plane to simplify the task of converting the DYNA3D nodal coordinates and boundary conditions for HYDRA.

While INGRID is currently the primary mesh generator being used with HYDRA, the input data for HYDRA is quite straightforward, and nearly any finite element mesh generator could be used in place of INGRID. Alternative mesh generation tools such as those from the National Grid Project at Mississippi State University, the CUBIT project at

Sandia National Laboratories, or from the PMESH project at LLNL will hopefully provide an adequate interface for HYDRA in the near future.

1.2.2 Post-Processor Interfaces

HYDRA can output several forms of graphics files, but the primary file format is the Methods Development Group's binary,graphics data files and time history files which are compatible with GRIZ[15] and THUG[16]. 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 is used for interrogating time history data at a moderate number of mesh points. Typically, the state data are 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 solid and structural mechanics. The use of these general purpose data visualizers requires translation from the primitive variables which HYDRA writes to the graphics data files to variables which can be displayed in GRIZ and THUG. Table 1.1 shows the mapping from HYDRA's 2-D primitive variables to GRIZ and THUG variables. Table 1.2 shows the mapping from 3-D HYDRA variables to GRIZ variables.

In GRIZ, the character strings associated with certain variables may have to be reset to reflect the correct HYDRA variable, e.g., the x-acceleration variable in GRIZ is actually the x-component of vorticity in a 3-D HYDRA database. For 2-D HYDRA 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 HYDRA and the variable mapping is handled automatically. However, THUG currently requires the use of the default variable names for the display of HYDRA variables. Table 1.3 shows the relationship between HYDRA global time history variables and THUG global time history variables. THUG currently provides the *divu* and *ke* global variable commands for the display of the rms divergence and total kinetic energy for HYDRA time history data.

Future work on interfaces to alternative visualization tools will be based, in part, upon user requirements and the functionality provided by alternative visualization tools for large scale CFD problems using unstructured grids.

HYDRA State Variables	GRIZ State Variables	HYDRA 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-vorticity	x-acceleration	unused	unused
stream function	y-acceleration	unused	unused
temperature	temperature	temperature	temperature
pressure	pressure	pressure	N/A
turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)
dissipation rate (ε)	dissipation rate (ε)	dissipation rate (ε)	dissipation rate (ε)
stress invariant (A_2)	stress invariant (A_2)	stress invariant (A_2)	stress invariant (A_2)

Table 1.1: 2-D HYDRA State and Time History Variables

HYDRA State Variables	GRIZ State Variables	HYDRA 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	temperature
pressure	pressure	pressure	unused
turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)	turbulent kinetic energy (k)
dissipation rate (ε)	dissipation rate (ε)	dissipation rate (ε)	dissipation rate (ε)
stress invariant (A_2)	stress invariant (A_2)	stress invariant (A_2)	stress invariant (A_2)

Table 1.2: 3-D HYDRA State and Time History Variables

HYDRA Global Time History Variables	THUG Global Time History Commands
RMS Divergence Error: $\sqrt{\frac{\left(C^T \underset{\sim}{u}\right)\bullet\left(C^T \underset{\sim}{u}\right)}{Nel}}$	*divu*
Total Kinetic Energy: $\frac{1}{2}\underset{\sim}{u}^T M_L \underset{\sim}{u}$	*ke*

Table 1.3: HYDRA Global Time History Variables

1.2.3 Material Models

Material models in HYDRA may be broadly classified into two groups. The first material model consists of the definition of a fluid density, kinematic viscosity, and thermal diffusivity for a problem involving only fluid flow. For flow problems which require only one material definition, HYDRA provides a simplified input format for specification of the fluid properties.

Included in this class of material definition is the Smagorinsky[8] model which currently is treated as just an added viscosity, albeit a turbulent viscosity based upon the local strain-rate tensor. Because the input data for this model are minimal, the data format is presented in a simplified form in the HYDRA control file (see Chapter 4).

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

1.3 Guide to the HYDRA User's Manual

The purpose for this document is to provide sufficient information for an experienced analyst to use HYDRA 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 HYDRA.

In this document, an attempt is made to adhere to the convention that all 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 HYDRA is presented. Chapters 3 and 4 present information on how to execute HYDRA in a UNIX environment and the necessary input data for HYDRA. Several sample calculations are presented in Chapter 5 which can be used as benchmark problems for the first time HYDRA user.

Chapter 2

Theoretical Overview

This chapter presents a brief overview of the theoretical foundation for HYDRA. As an overview, this chapter is not intended to be a complete technical reference. Instead, it simply presents the basic forms of the partial differential equations which HYDRA treats, and a general description of the methodologies employed in their solution. The interested reader may pursue the references included in this chapter for details on the algorithms used in HYDRA and their implementation.

To begin, a brief introduction to the incompressible Navier-Stokes equations is presented. This is followed by the advection-diffusion equation, the semi-discrete form of the conservation equations, and a discussion of the pressure poisson equation. Finally, a brief description of the currently implemented turbulence models is presented.

2.1 The Navier-Stokes Equations

The conservation equations for isothermal, time-dependent, laminar, incompressible, viscous flow are:

$$\frac{\partial \underset{\sim}{u}}{\partial t} + \underset{\sim}{u} \cdot \nabla \underset{\sim}{u} = -\nabla P + \upsilon \nabla^2 \underset{\sim}{u} + \underset{\sim}{f} \tag{2.1}$$

$$\nabla \cdot \underset{\sim}{u} = 0 \tag{2.2}$$

where $\underset{\sim}{u} = (u, v, w)$ is the velocity, $P = p/\rho, p$ is the pressure, ρ is the mass density, υ is the kinematic viscosity, and $\underset{\sim}{f}$ is the body force.

The system of equations above are subject to boundary conditions which consist of specified velocity as in Eq. (2.3), or pseudo-traction boundary conditions on Γ_2 as in Eqs. (2.4)–(2.5) and shown in Fig. (2.1).

$$\underset{\sim}{u} = \underset{\sim}{\hat{u}} \text{ on } \Gamma_1 \tag{2.3}$$

$$-P + \upsilon \frac{\partial u_n}{\partial n} = F_n \text{ on } \Gamma_2 \tag{2.4}$$

$$\upsilon \frac{\partial u_\tau}{\partial \tau} = F_\tau \text{ on } \Gamma_2 \tag{2.5}$$

Here, $\Gamma_1 \cup \Gamma_2 = \Gamma$ is the boundary of the domain, n represents the outward normal direction at the boundary, $u_n = \underset{\sim}{u} \cdot n$, $u_\tau = \underset{\sim}{u} \cdot \tau$, F_n and F_τ are the normal and tangential components of the boundary traction respectively. Homogeneous traction boundary conditions correspond to the well known natural boundary conditions in the finite element formulation which are typically applied at outflow boundaries.

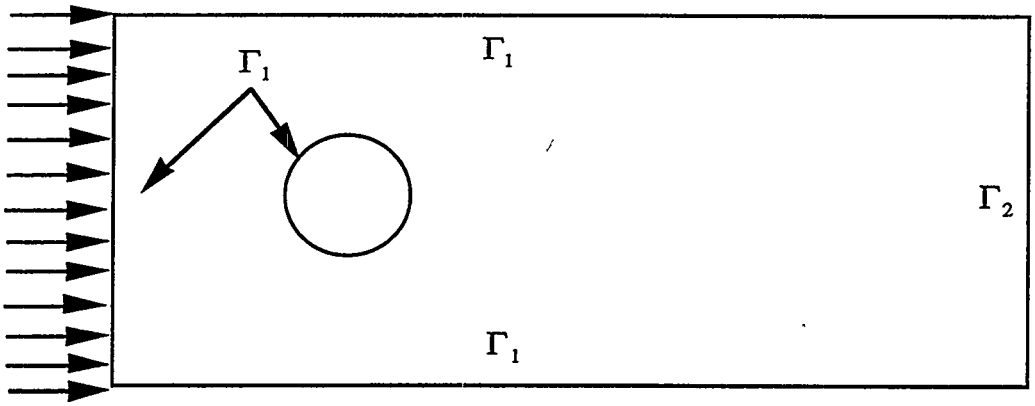


Figure 2.1: Flow domain for conservation equations.

The complete definition of a transient, incompressible flow problem requires the specification of initial conditions which also satisfy the divergence free constraint as shown in Eq. (2.7).

$$\underline{u}(x,0) = \hat{\underline{u}}(x,0) \tag{2.6}$$

$$\nabla \cdot \hat{\underline{u}} = 0 \tag{2.7}$$

$$n \cdot \underline{u}(x,0) = n \cdot \hat{\underline{u}}(x,0) \tag{2.8}$$

Equations (2.7)–(2.8) pose a solvability constraint on the flow problem[17]. That is, if either Eq. (2.7) or Eq. (2.8) are violated, then the flow problem is ill-posed. If $\Gamma_2 = 0$ (the null set), (e.g., enclosure flows with $\underline{u} \cdot n$ specified on all surfaces), then global mass conservation enters as an additional solvability constraint as:

$$\int_\Gamma n \cdot \hat{\underline{u}} \, d\Gamma = 0 \tag{2.9}$$

Remark

HYDRA always checks the initial conditions and boundary conditions and, if necessary, performs a divergence-free projection on the initial velocity field before the first time step is taken. This guarantees that the solvability constraints will always be met, even if the user input initial conditions violate the solvability constraints. The allowable error in the divergence $\left(\nabla \cdot \underline{u}\right)$ at start up may be specified using the *divu* analysis option (see Chapter 4).

2.2 The Advection-Diffusion Equation

The time-dependent, advection-diffusion equation is

$$\frac{\partial T}{\partial t} + \underset{\sim}{u} \cdot \nabla T = \frac{k}{\rho C_p} \nabla^2 T + \beta \tag{2.10}$$

Given the necessary initial and boundary conditions, HYDRA can solve the advection-diffusion equation with a prescribed velocity field ($\hat{\underset{\sim}{u}}$), or with a variable velocity field during the solution of the Navier-Stokes equations. Unlike the solvers for the incompressible Navier-Stokes equations, a divergence test on the prescribed velocity field is not performed when the advection-diffusion equation is solved by itself. It is the responsibility of the user to provide a divergence free velocity field. However, the *icwrt* command can be used to generate a file containing initial conditions from a Navier-Stokes computation which are suitable for HYDRA and may be used for advection-diffusion calculations (see the analysis commands in Chapter 4).

2.3 Spatial Discretization

The semi-discrete form of the conservation equations is the starting point for any discussion of time integration methods. The methods for obtaining the weak-form of the conservation equations are well known[18,19] and will not be repeated here. The semi-discrete form of the conservation equations are:

$$M \dot{\underset{\sim}{u}} + A\left(\underset{\sim}{u}\right)\underset{\sim}{u} + K\underset{\sim}{u} + CP = \underset{\sim}{f} \tag{2.11}$$

$$C^T \underset{\sim}{u} = 0 \tag{2.12}$$

$$M_T \dot{T} + A\left(\underset{\sim}{u}\right)T + K_T T = Q \tag{2.13}$$

Here, M is the unit mass matrix, $A(\underset{\sim}{u})$ is the advection operator, K the viscous diffusion, and f the body force. C is the gradient operator, and C^T is the divergence operator. In the energy conservation equation, Eq. (2.13), M_T is the mass matrix corresponding to the scalar valued advection-diffusion problem, with Q representing the discrete volumetric heat sources, and K_T being the thermal diffusivity operator.

The advection operator at the element level is:

$$A_{ij}^e\left(\underset{\sim}{u}\right) = \int_{\Omega_e} N_i \underset{\sim}{u}^h \cdot \nabla N_j \, d\Omega \tag{2.14}$$

where $\underset{\sim}{u}^h = \sum_{i=1}^{nnpe} N_i \underset{\sim}{u}_i$. Here, N_i is the element shape function, and *nnpe* is the number of nodes per element. In the evaluation of Eq. (2.14), an integral of triple products is required to generate the advection matrix. This very computationally intensive integral is

approximated by using an *ad hoc* modification known as the *centroid advection velocity.* This modification assumes that $\underline{u}^h$ in Eq. (2.14) may be approximated by:

$$\underline{u}^e = \sum_{i=1}^{nnpe} N_i(0,0,0)\underline{u}_i \tag{2.15}$$

This simplification results in an advection operator as shown in Eq. (2.16).

$$A_{ij}^e(u) = u_k^e \int_{\Omega_e} N_i \frac{\partial N_j}{\partial x_k} d\Omega \tag{2.16}$$

By following Gresho, *et al.*,[4,7], a consistent, discrete pressure-poisson equation may be constructed using the lumped mass matrix, M_L.

$$\left[C^T M_L^{-1} C\right] P = C^T M_L^{-1}\left[\mathbf{f} - K\underline{u} - A\left(\underline{u}\right)\underline{u}\right] \tag{2.17}$$

Equations (2.12)–(2.17) form the basis for a description of the time integration methods available in HYDRA. The following sections will discuss some of the modifications made to the basic element formulation for the sake of computational efficiency.

2.4 Element Technology

In HYDRA, the primary element for solving the Navier-Stokes equations is the so-called Q1/P0 element which provides C^0 continuity for the velocity and C^{-1} continuity for the pressure. Thus, the velocity support is trilinear in 3-D and bilinear in 2-D, while the pressure support is piecewise constant. Figures 2.2 and 2.3 show the canonical local node numbering in the natural coordinate systems for these elements.

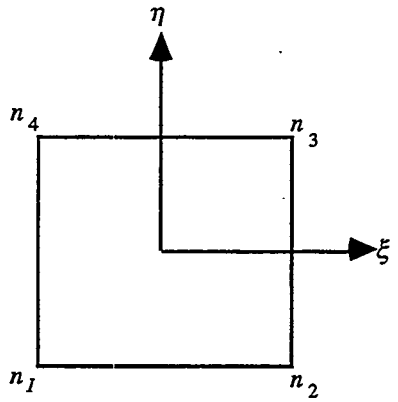


Figure 2.2: 2-D bilinear element.

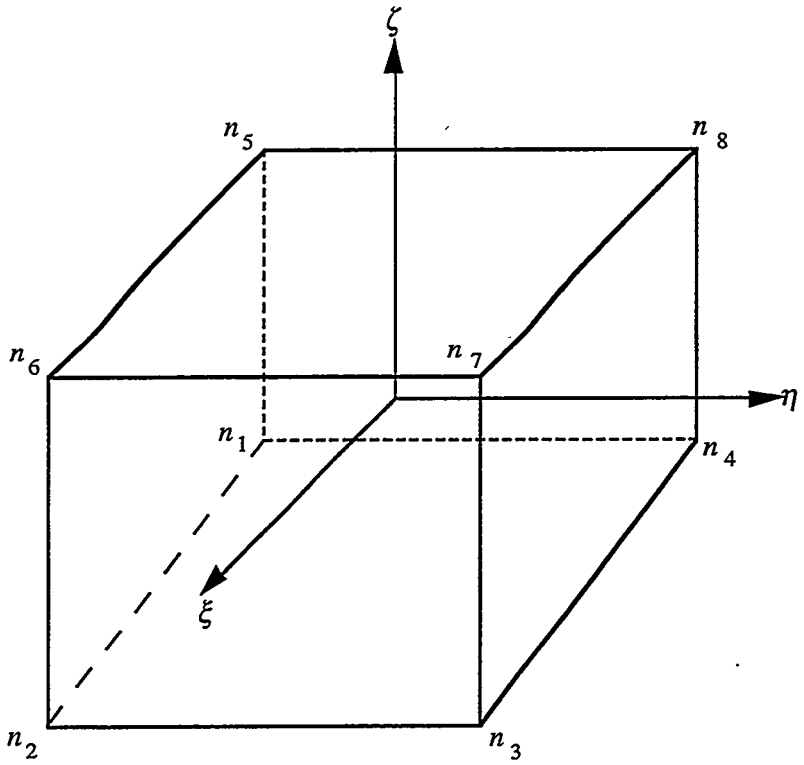


Figure 2.3: 3-D trilinear element.

In HYDRA, both "full" and "reduced" numerical integration are used. Full integration refers to the use of 2×2 quadrature points in the 2-D Gaussian quadrature, and 2×2×2 quadrature points in 3-D for all operators. In contrast, "reduced" or "one-point" integration means that a single quadrature point is used for all numerical integration. Table 2.1 (reproduced from Gresho, *et al.*[4]) shows the required quadrature rules for the evaluation of the element level operators in HYDRA.

Table 2.1 Number of Gauss points required in each coordinate direction to evaluate various element matrices (and element size, Ω^e) exactly

Matrices	Elements			
	Rectangle and parallelogram	General quadrilateral	Brick and parallelepiped	General hexahedron
M^e	1	2	1	2
K^e	2	>3*	2	>3*
A^e	2	2	2	2
C^e (and Ω^e)	1	1	1	2

* It is not generally possible to evaluate K^e exactly using Gauss quadrature.

Reduced Integration

Several of the solution options in HYDRA make use of reduced numerical integration for the sake of minimizing memory and floating point operations (FLOPs). Reduced integration has typically been employed in explicit time integration algorithms where every attempt is made to minimize the number of floating point operations per time step. The benefits of one-point integration are tremendous in nonlinear computational fluid dynamics problems because of the requisite sizes of meshes for interesting problems and the associated operation counts for performing constitutive evaluations and the concomitant operator formation and assembly.

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 operator, C. It has been demonstrated that the convergence rate of reduced integration elements is comparable to the fully integrated elements at a fraction of the computational cost. That is, second order convergence is maintained with reduced integration.[22]

Although there are practical benefits to using one-point integration, there are also some drawbacks. The primary difficulty with one-point integration is the possibility of a mesh instability referred to as *hourglassing* (sometimes referred to as *keystoning* in the finite difference community).

The reduced numerical quadrature of the diffusion term, K, in Eq. (2.11) leads to rank deficiency of the element level operator. The presence of an improper singular mode in the element level operator can also lead to singularity of the assembled global operators. In 2-D, there is only one improper singular mode as shown in Fig. 2.4. In 3-D, there are four improper singular modes which are shown in Fig. 2.5. When the hourglass modes are excited in a numerical solution, they remain undamped and can pollute the entire field. In 2-D, the presence of hourglass modes is most easily detected in surface or contour plots. In 3-D, the presence of hourglass modes is much more difficult to detect because the four modes rarely occur individually in a pure form.

In order to eliminate the singular modes, a stabilization operator is used as shown in Eq. 2.18 .

$$K^e = K^e_{1pt} + K^e_{stab} \tag{2.18}$$

The specifics of the stabilization operator in Eq. 2.18 may be found in references 20 - 24. In HYDRA, the default hourglass stabilization is based upon the work of Goudreau and Hallquist[24] and Gresho *et al.*[4]. This stabilization method is the so-called "h-stabilization" because the stabilization operator is formed from the outer-product of the hourglass vectors at the element level. This form of stabilization is sometimes referred to as trace stabilization.

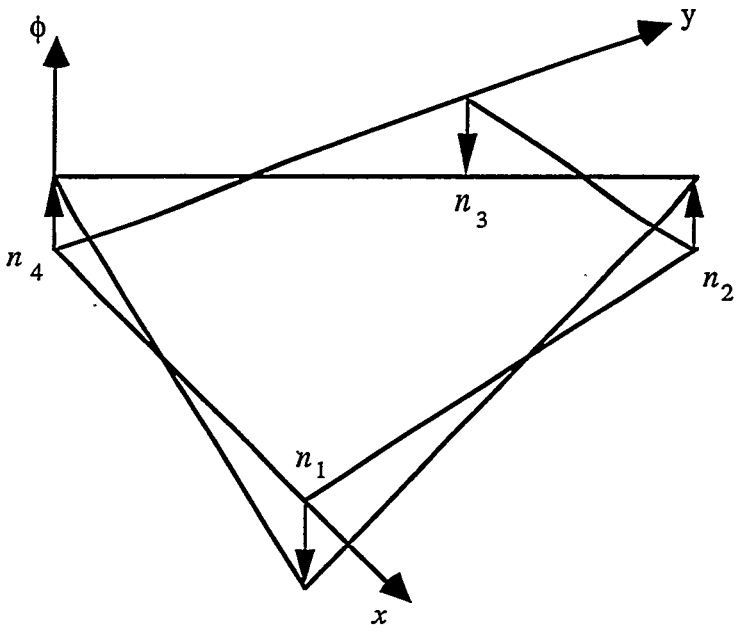


Figure 2.4: 2-D Hourglass mode shape.

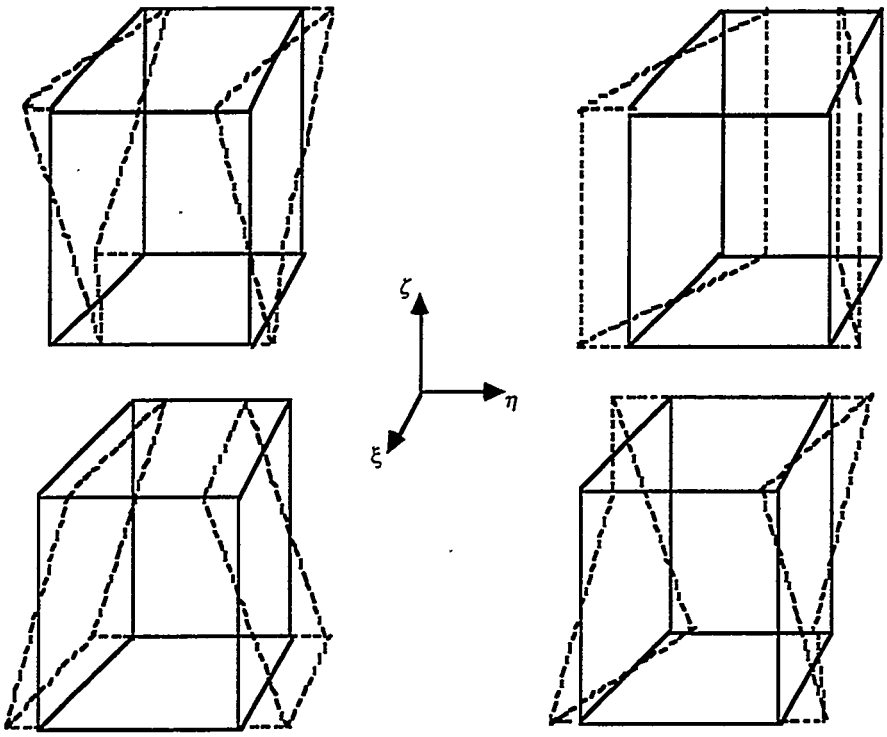


Figure 2.5: 3-D hourglass mode shapes.

In the 2-D, explicit time integration of the Navier-Stokes, there is an optional hourglass stabilization based upon the work of Belytschko and his colleagues[20,21,22,23]. This stabilization method is often referred to as γ-stabilization. γ-stabilization is perhaps more robust in structural mechanics applications, but also requires more operations and storage than h-stabilization.

It is the author's experience that it is relatively more difficult to excite the hourglass modes in an Eulerian computation than in a corresponding Lagrangian computation, e.g., a DYNA3D[1] computation. However, γ-stabilization still requires fewer operations and less storage than the fully integrated element in 2-D. In 3-D, this is not the case. Table 2.2 shows the memory requirements, and operations counts for a matrix-vector multiply ($K\underset{\sim}{u}$) for a variety of element formulations. In 2-D, γ-stabilization requires nearly the same storage as the fully integrated element, but takes 9 more operations to achieve the matrix-vector multiply. In 3-D, γ-stabilization is about 3 times more expensive to perform than the corresponding row-compressed global matrix-vector multiply. Thus, in both 2-D and in 3-D, h-stabilization is the default hourglass stabilization option.

Table 2.2: 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, Nbw = 1/2 bandwidth)

Dim.	Options	Storage	+	*	Total
2-D	1-pt. EBE	4Nel	12Nel	16Nel	28Nel
2-D	1-pt. h-stab	4Nel	22Nel	18Nel	40Nel
2-D	1-pt. γ-stab	9Nel	19Nel	25Nel	41Nel
2-D	2X2 quadrature	10Nel	16Nel	16Nel	32Nel
2-D	sparse/ITPACK	10Nel	8Nnp	9Nnp	17Nnp
3-D	1-pt. EBE	12Nel	29Nel	28Nel	57Nel
3-D	h-stab	12Nel	69Nel	46Nel	115Nel
3-D	γ-stab	45Nel	61Nel	100Nel	161Nel
3-D	sparse/ITPACK	54Nel	26Nel	23Npp	53Nnp
3-D	2X2X2,EBE	36Nel	64Nel	64Nel	128Nel
Global	Band Vector	NbwNnp	(2Nbw)Nnp	(2Nbw)Nnp	(4Nbw)Nnp

2.5 Grid Parameters

HYDRA computes grid Reynolds and CFL (Courant-Freidrichs-Levy) numbers and reports the minimum/maximum values at a user specified interval during time integration. The interval for the calculation and reporting of the grid parameters is controlled with the *dtchk* command which is described in Chapter 4. The grid parameters are computed according to the method presented in Christon.[43]

In an unstructured grid, with variable element size, the precise calculation of the grid *Re* and CFL numbers requires the computation of both grid metrics and contravariant velocities. This can be a relatively computationally intense calculation, and so HYDRA relies upon the use of element-local coordinates and centroid velocities for the estimation of the grid parameters.

The grid Re and *CFL* are defined in the canonical element-local coordinate system as shown in Figures 2.2 and 2.3. This coordinate system corresponds approximately to the element-local coordinates shown in Figures 2.6 and 2.7. In these figures, the location of the element-local coordinates is based upon the intersection of the vectors connecting the mid-sides of the elements (Fig. 2.6).

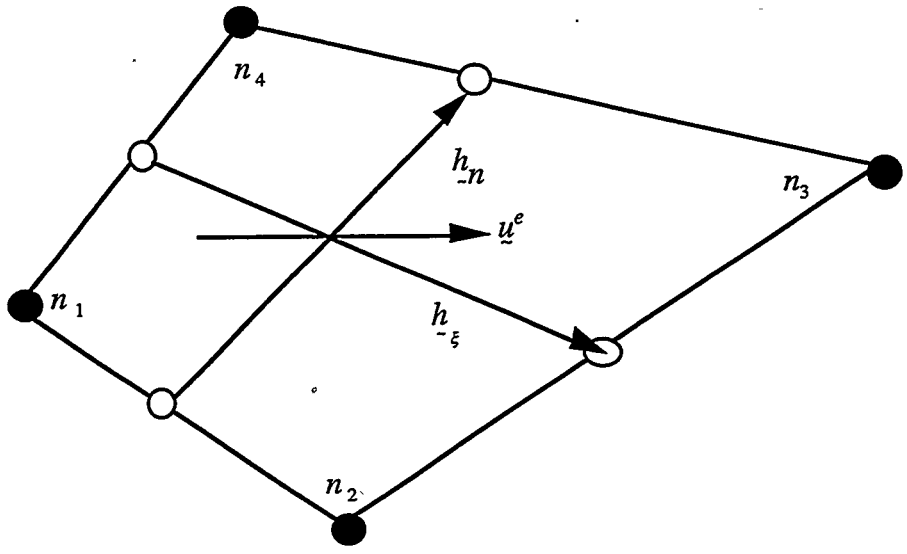


Figure 2.6: Element parameters for the grid Re and *CFL* estimation.

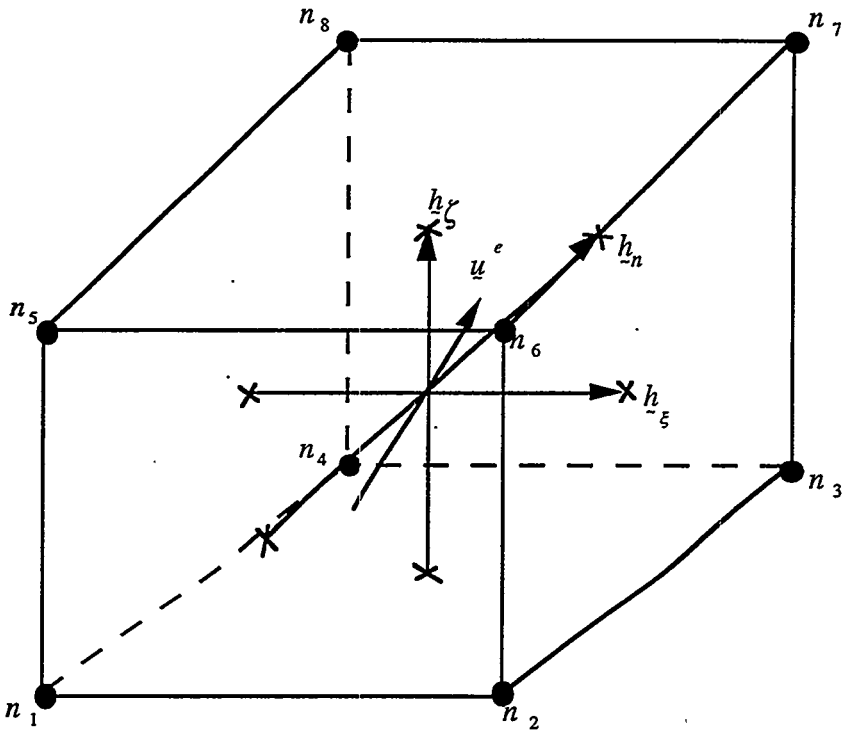


Figure 2.7: Element parameters for the grid Re and *CFL* estimation in 3-D.

The grid Reynolds numbers is defined as:

$$\mathrm{Re}_i = \frac{\left|\underset{\sim}{u}^e \cdot \underset{\sim}{h}_i\right|}{\upsilon} \tag{2.19}$$

and the grid CFL number as:

$$CFL_i = \frac{\left|\underset{\sim}{u}^e \cdot \underset{\sim}{h}_i\right| \Delta t}{\left\|h_i\right\|^2} \tag{2.20}$$

where $i = \xi,\ \eta,\ \zeta$ for the element-local coordinates.

Thus Reand *CFL* rely upon the projection of the centroid velocity onto the element-local coordinate directions. These quantities are used in the stability computations for the explicit time integration algorithms described below.

Figure 2.8 shows sample HYDRA output for the 2-D grid parameter estimation. The xi-grid, eta-grid, and psi-grid parameters correspond to the element-local ξ and η,coordinate directions based upon the canonical local node number system shown in Figures 2.2 and 2.3. Thus, interpretation of the element-local grid parameters requires a knowledge of element orientation which is normally available in a mesh plot.

```
xi-grid Reynolds Numbers:
=========================

Min. Element no ..............................          80
Min. xi-element dimension ....................1.7517E-02
Minimum xi-grid Reynolds Number ..............8.1128E-04
Max. Element no. .............................         201
Max. xi-element dimension ....................7.6475E-01
Maximum xi-grid Reynolds Number ..............7.6897E+01


eta-grid Reynolds Number:
=========================

Min. Element no. .............................        1080
Min. eta-element dimension ...................4.4444E-01
Minimum eta-grid Reynolds Number .............6.7829E-07
Max. Element no. .............................         212
Max. eta-element dimension ...................4.7355E-01
Maximum eta-grid Reynolds Number .............3.2396E+01


xi-grid CFL Numbers:
====================

Min. Element no. .............................          80
Min. xi-element dimension ....................1.7517E-02
Minimum xi-grid CFL Number ...................1.3220E-02
Max. Element no. .............................         424
Max. xi-element dimension ....................6.7045E-02
Maximum xi-grid CFL Number ...................1.3590E+01


eta-grid CFL Numbers:
=====================

Min. Element no. .............................        1080
Min. eta-element dimension ...................4.4444E-01
Minimum eta-grid CFL Number ..................1.7169E-08
Max. Element no. .............................         442
Max. eta-element dimension ...................4.6413E-02
Maximum eta-grid CFL Number  .................1.3777E+01
```

Figure 2.8: Sample HYDRA Output for Grid Parameters.

2.6 Explicit Time Integration

The explicit time integration algorithms for both the Navier-Stokes and the scalar advection-diffusion equation make use of the one-point stabilized element technology described in Section 2.4. The explicit algorithms provide very fast element cycle times, i.e., the cpu time required to advance one element one time step. Further, the explicit algorithms require minimal storage for the basic finite element operators.

The basic 2-D and 3-D explicit time integration follows the algorithm described in Gresho *et al.*[4,5]. The ad-hoc modifications to the standard Galerkin finite element method include the use of the reduced integration, stabilized elements, balancing tensor diffusivity, and a row-sum lumped mass matrix. The effect of balancing tensor diffusivity (BTD) is to provide second-order accuracy in time for the advection terms, and is a key ingredient in the explicit algorithm.

The explicit algorithm implemented in HYDRA begins with a given divergence-free velocity field, $\underset{\sim}{u}^o$, which satisfies the essential boundary conditions, and an initial pressure, P^o. The algorithm proceeds by first computing a partial acceleration as in Eq. 2.21. The pressure at level n is then computed using the partial acceleration to form the right-hand-side in Eq. 2.22. Given P^n, the velocity is updated according to Eq. 2.23 and the temperature according to Eq. 2.24.

$$\dot{\underset{\sim}{u}}^n = M_L^{-1}\left[\underset{\sim}{F}^n - A\underset{\sim}{u}^n - K\underset{\sim}{u}^n\right] \tag{2.21}$$

$$\left[C^T M_L^{-1} C\right]P^n = C^T \dot{\underset{\sim}{u}}^n \tag{2.22}$$

$$\underset{\sim}{u}^{n+1} = \underset{\sim}{u}^n + \Delta t\left[\dot{\underset{\sim}{u}}^n + M_L^{-1}\, CP^n\right] \tag{2.23}$$

$$T^{n+1} = T^n + \Delta t M_L^{-1}\left[Q^n - AT^n - K_T T^n\right] \tag{2.24}$$

The explicit algorithm must respect both the diffusive and the convective stability limits. The use of BTD is recommended for all simulations despite some of its drawbacks[7], because of its beneficial effects upon stability. While the analytical stability limits for the explicit time integration of the Navier-Stokes equations in multiple dimensions remains intractable[4], the stability computations in HYDRA rely upon the estimation of element grid size and the grid parameters described in section 2.5.

In order to make use of Equations 2.19 and 2.20, a unit normal for each element-local coordinate direction is defined as:

$$\hat{e}_i = \frac{\underset{\sim}{h}_i}{\left\|\underset{\sim}{h}_i\right\|} \tag{2.25}$$

Using Eq. 2.25, the advective-diffusive stability limit becomes

$$\Delta t_i \le \frac{\left\| \underset{\sim}{h}_i \right\|^2}{\gamma \left\{ 1 + \sqrt{1 + \left(\frac{\underset{\sim}{u}^e \cdot \underset{\sim}{h}_i}{\gamma} \right)^2} \right\}} \tag{2.26}$$

where $i = \xi,\ \eta,\ \zeta$. A minimum is taken over all elements and all element-local coordinates to establish a global mesh stability requirement.

The CFL condition is established in a similar manner using

$$\Delta t_i \le \frac{CFL \left\| \underset{\sim}{h}_i \right\|^2}{\left| \underset{\sim}{u}^e \cdot \underset{\sim}{h}_i \right|} \tag{2.27}$$

where *CFL* is the user specified acceptable *CFL* number which must be always be less than 1.0 for the explicit algorithm.

By default, the minimum constraining time step is used in the time integration scheme. However, the user may request that the grid parameters and stable time step be re-evaluated at a fixed interval using the *dtchk* command described in Chapter 4. In a rapidly changing flow field, this may avoid over-estimating or under-estimating the stable time step based upon the initial divergence-free velocity field.

2.7 The Semi-Implicit Projection Algorithm

The application of the explicit time integration scheme to the solution of the Navier-Stokes equations can lead to extremely small time steps based upon the diffusional stability limit. This is particularly true when an attempt is made to resolve very thin boundary layers. The semi-implicit projection algorithm treats the diffusional terms implicitly which removes the diffusional stability limit imposed in the explicit algorithm above. The implementation of the semi-implicit projection algorithm follows the description of the Projection-II (P-II) algorithm in Gresho and Chan.[7]

The P-II algorithm is a second-order accurate fractional step method[6,7] which uses a consistent-mass predictor in conjunction with a lumped mass corrector. In the P-II algorithm, the velocity and pressure fields are legitimately decoupled. This reduces both memory and cpu requirements relative to traditional fully coupled solution strategies. The consistent mass predictor retains phase speed accuracy, while the lumped mass corrector (projection) maintains a divergence 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 super-scalar. The P-II projection algorithm with consistent mass matrix delivers superconvergent fourth-order phase accuracy when a uniform mesh is used.

Given an initial divergence-free velocity field, u^o, which satisfies the essential boundary conditions, and the associated pressure field, P^o, the P-II algorithm proceeds as follows. First, an intermediate velocity field is predicted by solving Eq. 2.28 for $\tilde{u}^{n+1}$. The divergence of the intermediate velocity field is then used as a right-hand-side for the computation of a Lagrange multiplier as in Eq. 2.29. Finally, the predicted velocity field is projected to a divergence-free velocity field using Eq. 2.30, and the pressure field is updated using Eq. 2.31.

$$\left[M+\Delta t\left(\theta_k K+\theta_B K_B\right)\right]\tilde{\underline{u}}^{n+1}=\Delta t\left\{\theta_f f^{n+1}+\left(1-\theta_f\right)f^n-A\underline{u}^n-MM_L^{-1}CP^n\right\}+$$

$$\left[M-\Delta t\left[\left(1-\theta_k\right)K+\left(1-\theta_B\right)K_B\right]\right]\underline{u}^n \tag{2.28}$$

$$\left[C^T M_L^{-1}C\right]\lambda=C^T\tilde{\underline{u}}^{n+1} \tag{2.29}$$

$$\underline{u}^{n+1}=\tilde{\underline{u}}^{n+1}-M_L^{-1}C\lambda \tag{2.30}$$

$$P^{n+1}=P^n+\frac{\lambda}{\Delta t} \tag{2.31}$$

In Eq. 2.28, θ_K controls the time weighting of the viscous diffusion terms, θ_B controls the time weighting of the BTD terms, and θ_f controls the time weighting of any applied traction. For $\theta_i=\frac{1}{2}$, $(i=K,B,F)$ the time integration is second-order accurate, but is only conditionally stable. However, the implicit treatment of the BTD terms in Eq. 2.28 results in a relaxed *CFL* stability limit. Numerical experiments have shown that the *CFL* number can be as high as 15–20 when the limiting elements are small and in a region where advective transport does not influence the global flow field significantly. Typically, *CFL* numbers in the range of 2–10 produces stable simulations.

For $\theta_i=0$, $(i=K,B,F)$, the time integration is essentially an explicit algorithm corresponding to the algorithm described in section 2.4. For $\theta_i=1$, the time integration corresponds to backward Euler. For these two cases, the mass option described in Chapter 4 can be used to select a lumped mass—a good choice when using backward Euler time integration to time-march to a steady-state.

Solution of the Momentum Equations

Currently, the primary solution strategy for the P-II algorithm in HYDRA is a highly optimized Jacobi preconditioned conjugate gradient (JPCG) solver which is used to solve the linear systems which derive from the decoupled momentum equations at each time step. The JPCG solver makes use of the ITPACK row-compressed compact storage scheme[35,36]. This storage scheme is independent of the bandwidth of the linear system, and is very well suited to vector supercomputer architectures.

In the iterative solution of the momentum equations, two convergence criteria must be met as outlined in Gresho[44]. Viewing the system to be solved as $Ax=b$, Equations 2.32

and 2.33 define the stopping criteria. That is, both the normalized residual, and the change in the solution vector during the last iteration must be less than the user specified tolerance.

$$\frac{\|r^n\|}{\|b\|} \le \varepsilon \tag{2.32}$$

$$\frac{\|x^n - x^{n-1}\|}{\|x^n\|} \le \varepsilon \tag{2.33}$$

For the momentum equations, the user may specify the maximum number of iterations permitted, the interval to check the residual norms, and the allowable tolerance. Typically, $\varepsilon = 1.0e-5$, produces acceptable solutions in 20–50 iterations on most problems. Exceptionally tough problems may require more iterations, but the number of iterations should never exceed the number of nodal points in the mesh. For more detailed information on the iterative methods used in HYDRA, see references 25, 32, 33, 34, 38, 39, 41.

2.8 Start-up Procedures

The solvability constraints on the initial conditions in Eqs. 2.7–2.8 are tested during the initialization phase for a new calculation. This start-up procedure follows the algorithm given in Gresho[4]. The *rms* divergence of the initial velocity field is computed and compared to a user specified tolerance (see *divu* in Chapter 4).

If the divergence is greater than the specified tolerance, a mass consistent projection to a divergence-free subspace is performed. For an initial velocity field, $\tilde{u}^o$, Eq. 2.34 is solved for λ, and the orthogonal projection in Eq. 2.35 is performed.

$$\left[C^T M_L^{-1} C\right]\lambda = C^T \tilde{u}^o \tag{2.34}$$

$$u^o = \tilde{u}^o - M_L^{-1} C\lambda \tag{2.35}$$

Given initial conditions which satisfy the solvability constraint, an initial partial acceleration is computed according to Eq. 2.36. In the case of the explicit algorithm, a lumped mass matrix (M_L) is used, while for P-II, a consistent mass matrix is employed. A PPE is then solved for the pressure at t=0, using $\dot{u}^o$ from Eq. 2.36.

$$M\dot{u}^o = f^o - Ku^o - Au^o \tag{2.36}$$

$$\left[C^T M_L^{-1} C\right]P^o = C^T u^o \tag{2.37}$$

While the user is given control over the allowable divergence, it is not advisable to relax the divergence of the initial velocity field. This is particularly so with the explicit algorithm which relies heavily upon the initial velocity field being divergence-free.

2.9 The Pressure Poisson Equation

The efficient solution of the time-dependent Navier-Stokes equations relies heavily on robust and computationally efficient methods for solving the pressure Poisson equation (PPE). In HYDRA, the use of the Q1/P0 element implies that there is an underlying dual grid for the piece wise constant pressure elements as shown in Fig. 2.9.

Many codes rely upon a pre-processor in addition to a mesh generator for the construction of the PPE dual grid. HYDRA follows the algorithm in Christon[43] to construct the dual grid during the initialization phase. The simplest algorithms for constructing the dual grid are N_{el}^2 algorithms, while the dual grid construction algorithm in HYDRA only requires $O(N_{el})$ operations which permits the dual grid computation to be performed in HYDRA during the initialization phase.

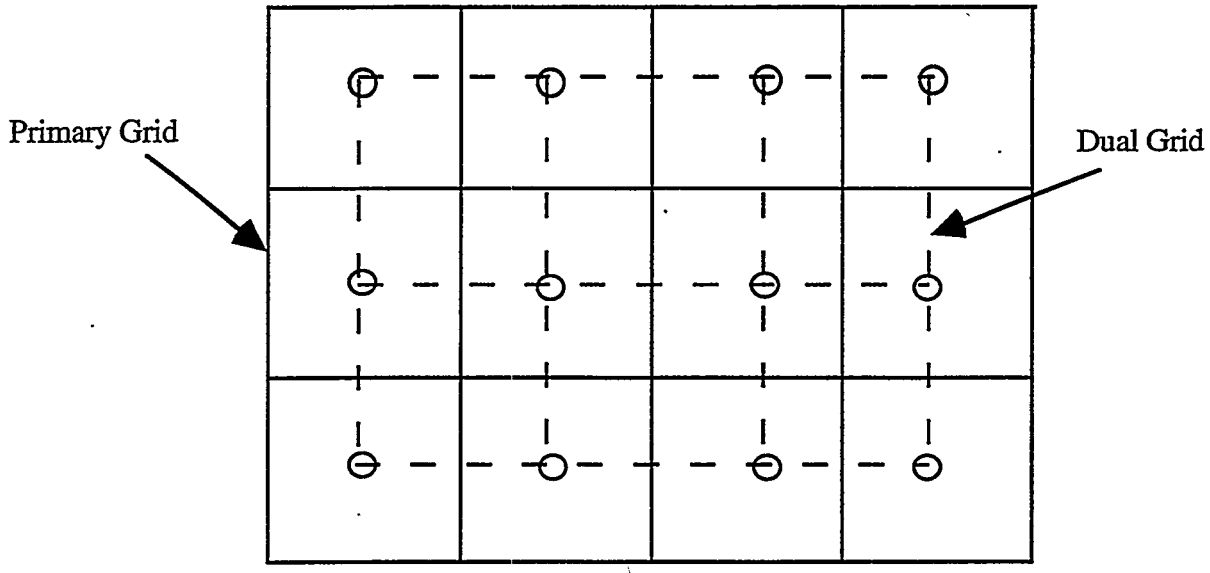


Figure 2.9: Mesh and PPE Dual Grid

The efficient solution of the PPE has been an ongoing research issue during the development of HYDRA, and is not a closed issue today. HYDRA provides a wide variety of solution strategies for the PPE, and the overall performance of the code is sensitive to the solution option chosen. In fact, the performance of the solution algorithms vary across computer architectures, and not all of the algorithms require the explicit construction of the dual grid.

For the iterative techniques, the convergence criteria applied to stop the iteration is identical to that used for the momentum equations (Eqs. 2.32–2.33). In general, all of the conjugate gradient based PPE solvers typically require $(0.01-0.1)Nel$ iterations to converge with a tolerance of $\varepsilon = 1.0e-5$. However, the iterative solvers require less memory than the direct solvers.

Of the direct solvers, the PVS[28, 29] solver is highly recommended when memory is not an issue. However, as is the case with all direct solvers, the PVS solver does not scale to very large problems because of round-off and memory limitations. Direct solvers are also very dependent on having minimum bandwidth (profile). All of the direct solvers in HYDRA automatically invoke the Gibbs-Poole-Stockmeyer[45] bandwidth minimization routines.

While there are no hard and fast rules for picking an optimal PPE solver, in general, the direct methods are robust and fast, requiring only a resolve of the pre-factored PPE at each time step. Table 2.3 shows the relative costs in memory and cpu time of a conjugate gradient solver, the PVS solver, and a fixed bandwidth Gaussian elimination solver.

The band Gaussian solver was used to normalize the memory requirements and cpu usage for a three dimensional problem with approximately 30,000 elements. The PVS solver is the clear winner with regards to the element cycle time of the explicit algorithm, but the EBE/JPCG solver requires no additional storage over the basic operators for the momentum equations. The algebraic multigrid solver (AMG) performed only marginally better than the conjugate gradient solver and required a good deal of memory to hold the coarse grids. While these findings are not absolute, hopefully they can guide the user in the selection of an appropriate solver for each problem. The following paragraphs provide a brief description of each PPE solver.

Table 2.3 PPE solver Comparison (All timings on a CRAY Y-MP single processor)

PPE Solver	Cycle Time [μs/El.$-\Delta t$]	Relative Cycle Time	Relative Memory Usage
Band Gaussian	22.0	1.00	1.00
PVS Cholesky	8.8	0.40	0.75
EBE/JPCG	34.5	1.57	0.00
AMG V-Cycle	31.90	1.45	1.30

EBE/JPCG

The element-by-element Jacobi preconditioned conjugate gradient (EBE/JPCG) solver was designed to make use of the C and M_L^{-1} operators without requiring any additional storage for the PPE matrix. Thus, this is a "matrix-free" method, because it performs the necessary matrix-vector multiplies for conjugate gradient in an element-by-element right-to-left order rather than using the traditional left-to-right ordering.

EBE/JPCG is highly vectorized, and is also a very parallel algorithm. The primary drawback in its current implementation is the relatively slow convergence rates. Future research will focus on better preconditioners for this solver because it has so much potential in parallel scalability.

UDU Band Gaussian

The UDU solver was written by Erik Thompson at Colorado State University, and is included as a reference solver for benchmarking purposes. The storage format is based upon storing the upper symmetric bands of the coefficient matrix in the columns of the PPE array. This solver requires bandwidth minimization, but still is relatively costly because of wasted storage and unnecessary operations associated with zero entries in the PPE .

PVS solver

The PVS[46] solver also requires bandwidth minimization, but uses a compact row storage scheme to avoid storing entries which are zero. The PVS solver is highly vectorized, and makes extensive use of loop unrolling to achieve a high level of performance on vector and cache based architectures.

AMG solver

The algebraic multigrid solver (AMG)[27] is a black-box multigrid solver which is capable of treating the linear systems generated for unstructured grids. This is an experimental solver in HYDRA, and will not be supported. For the brave users who wish to experiment with AMG, additional information on the multigrid method may be found in Briggs[26,40].

JPCG

The Jacobi preconditioned conjugate gradient solver is also used for the momentum equations in the P-II algorithm, and may also be invoked for the PPE. Details of the data structures used may be found in the ITPACK User's guide[35] and the NSPCG User's Guide[36].

SSOR-PCG

The symmetric successive over-relaxation preconditioned conjugate gradient algorithm requires no additional storage with respect to the JPCG solver for the PPE. Currently, this solver may only be invoked for the solution of the PPE. HYDRA's *SSOR-PCG* solver is based in part upon the work in references 30, 31, 42.

2.10 Turbulence Models

Turbulence modeling is currently the pacing issue in computational fluid dynamics. The ability to treat turbulent flow with complex geometry where the flow is time-dependent and possibly separated remains an unanswered challenge to date. Indeed, there is no universal turbulence model which can be applied to a given flow problem with an arbitrary level of geometrical complexity. Wilcox[9] sums it up best with his statement about the ideal turbulence model.

> *"... an ideal model should introduce the minimum of complexity while capturing the essence of the relevant physics."*

Wilcox also suggests that the complexity of a turbulence model is dictated by the level of detail required in a simulation. There is a plethora of turbulence models available in the literature which range in complexity from algebraic models which employ the Boussinesq eddy-viscosity approximation to full second-moment models which require seven additional transport equations. Turbulence research using HYDRA as a vehicle is an ongoing effort with two major concentrations. The first involves the use of a simple Smagorinsky model[8,9] for performing Large Eddy Simulations (LES). The second effort is focused on implementing multi-equation models which can capture the effects of the anisotropic Reynolds stress induced corner vortices.

Large Eddy Simulation

The basic idea behind LES is to resolve the larger eddies in a turbulent flow while employing a model for the smallest eddies which cannot be resolved by the grid.

The LES model implemented in HYDRA follows the development in Wilcox[9], and makes use of a simple Smagorinsky sub-grid scale (SGS) model. Currently, this model is available only in the 3-D, explicit, time-integration option of HYDRA. The Smagorinsky eddy viscosity is based upon the element level strain-rate tensor as defined in Eqs. 2.38–2.39.

$$\upsilon_T = \left(C_s \Delta\right)^2 \sqrt{S_{ij} S_{ij}} \tag{2.38}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.39}$$

In Eq. 2.38, Δ is based upon the local element diameter corresponding to the element volume. The Smagorinsky constant, C_s, varies from flow-to-flow, but its typical values are:

$$0.1 \leq C_s \leq 0.24 \tag{2.40}$$

The Smagorinsky model presented here represents the simplest type of turbulence model which can be employed. While the implementation in HYDRA will permit the use of a dynamic sub-grid scale model, however, it does not employ any type of explicit wall model. It should be noted that the one of the primary reasons for the relative success of the Smagorinsky model is that is provides enough additional diffusion to stabilize most simulations. Further, the large eddies (grid resolvable) yield statistics which are not influenced by the SGS model.

The scalability of this turbulence model is an issue. For a direct numerical simulation, the number of grid points for a simple channel flow scales as $Re^{9/4}$. The application of an LES model scales approximately as Re^2 which is still extremely demanding.

2.11 Derived Variables

The simulation of a flow problem is essentially meaningless without suitable visualization of the results. This requires that derived variables be provided for the visualization tool so that the user may interrogate all aspects of the flow field.

In HYDRA, the one-point elements are essentially constant gradient elements, e.g., velocity and temperature gradients are constant across an element. While this is true for all of the primitive variables with the one-point elements in HYDRA, certain derived variables, such as vorticity are projected to the nodes of the elements for output in the graphics database. For the computation of the vorticity, an inverse-area weighted projection is used. In 2-D, the computation of the stream function on unstructured grid is based upon the algorithm detailed in Christon[43].

It is important to note that the derived variable computations are performed only when a state dump to the graphics database is required. Therefore, derived variables such as the vorticity are not available in the time-history graphics database.

2.12 Vectorization, Parallelization, and Performance

The goal for HYDRA is to provide nearly optimal performance as possible over a wide variety of computer architectures as possible. The philosophy for HYDRA has been to map the most advanced incompressible flow algorithms which are available to existing high performance computer architectures. This is quite a different idea than performing a port (which often consists of just a re-compile) of an existing implementation to various computers. The primary difference being that the optimal implementation of an algorithm for a register-to-register vector supercomputer can be considerably different from the best implementation for a RISC machine.

Ultimately, for HYDRA to achieve its performance goals, it was necessary to design a code which is reconfigurable. That is, given a target architecture, it is necessary to select the machine specific parameters and routines which are most optimal for the computer.

HYDRA was developed using standard UNIX tools and employs a memory package which provides true dynamic memory management (unlike DYNA3D and NIKE3D which

rely upon the dated strategy of an expandable blank common). This means that HYDRA can both allocate and free memory as it needs during the execution cycle of a problem. This is a must for providing a scalable computational tool, i.e., one that can accommodate very large meshes.

Element Groups

One of the key vectorization ideas in DYNA3D and NIKE3D is the element group. HYDRA has enhanced this basic concept making it a flexible, hierarchical mechanism which is used in configuring the code for a target architecture. The basic idea behind the element groups is to group elements according to a certain criteria. For register-to-register vector supercomputers, the criteria is that all elements in a group and their associated data structures should be completely independent to permit full vectorization. Thus, the goal is to avoid any vectorization inhibiting data dependencies.

In HYDRA, the size of the element groups is a configurable parameter, and is set based upon the size and number of vector registers on a vector computer, or according to the size of cache on a RISC machine. The topological domain decomposition used to group elements for vectorization is a well known algorithm and is described in Hughes[18].

In the case of most parallel machines, the element groups form a second level in a hierarchical topological domain decomposition scheme. The coarse grained decomposition is achieved via a technique such as recursive spectral bisection (RSB), and then the elements are grouped in data independent clusters. This is the technique applied to machines such as the Meiko CS-2—a parallel vector architecture.

The element grouping idea has been an essential tool for mapping HYDRA to very-long-vector SIMD architectures such as the CM-2 and CM-5 where the element groups need not be data independent. The element groups need not be data independent because of the capability of these machines to perform data routing with data reduction to gracefully handle data dependencies. Given the size of machine (number of physical processors), the element groups may be quite large. For the CM-5 with vector units, the typical group size is 8192.

Vectorization

All of the algorithms in HYDRA have been vectorized. This is a common starting point even for parallel architectures because the process of vectorization is well understood, can be easily tested, and usually reveals the key aspects of finite element based algorithms. Some of the concepts used in DYNA3D for achieving its high level of vector performance have been studied and enhanced for HYDRA. Where it has been possible, standard techniques such as loop-unrolling, scalar promotion, and vector reuse have been taken into account in the implementation of HYDRA. However, there is always room for improvement.

Parallelization

The early development of HYDRA began on the CM-200 at the Advanced Computing Laboratory at Los Alamos National Laboratory. The work on the CM-200 then moved to the CM-5 at the Army High Performance Computing Research Center at the University of Minnesota. The early versions of HYDRA could be configured as FORTRAN-90 or FORTRAN-77 to permit the rapid transition from a CRAY environment to the Connection Machine.

While this was successful, the delivered performance of the CM-5 on unstructured grids was quite disappointing. The best performance was approximately eight times slower than the best timings on the CRAY Y-MP. The ideas of Tezduyar, *et al.* seem to lead to very good performance on the CM-5, but require a code architecture which is inflexible. This fact in conjunction with the shortcomings of FORTRAN-90 have led the author away from this architecture and away from FORTRAN-90 for the time being.

The movement back to FORTRAN-77 was also driven by the availability of machines such as the Meiko CS-2, and workstations clusters as well as symmetric multiprocessors. The flexibility afforded by FORTRAN-77 in combination with the DDMP model has led to a more portable version of HYDRA than FORTRAN-90 could provide.

The current parallelization efforts are focused upon the DDMP model not only for the Meiko, but for workstation clusters and the CRAY T3D. It is anticipated that future versions of HYDRA will even be available for heterogeneous clusters of workstations.

Chapter 3

Running HYDRA

HYDRA has been exercised on computers ranging from UNIX workstations to traditional CRAY supercomputers, Thinking Machines CM-200 and CM-5 computers, and the vector-parallel Meiko CS-2. The common thread for all of these machines is a UNIX (or UNIX like) environment. HYDRA 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.

3.1 Execution

HYDRA may be executed with the following command line options:

hydra -i *inf* -c *cntl* -o *out* -p *plot* -h *hist* -g *glob* -d *dump*

HYDRA Command Line Arguments

Keyword	**Meaning**
-i *inf*	HYDRA mesh file (*default:* **mesh**)
-c *cntl*	HYDRA control file (*default:* **cntl**)
-o *out*	Human readable output file (*default:* **out**)
-p *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 restarts (*default:* **dump**)

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.

```
hydra -i \~/plate/flow1.msh -c../cntl1 -o/home/joe/plate.out
```

As the 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 which changes to a three digit family number after 100 files have been generated.

3.2 Restarts

HYDRA will automatically write a binary check-point file which contains all of the data necessary to restart a computation. If the *dump* file exists when HYDRA starts execution, then HYDRA will allow the user to choose between a restart and starting a new calculation. HYDRA re-writes the check-point file in place each time a simulation is terminated normally. Therefore, the user must first save a copy of the check-point file before restarting if the original check-point file is to be preserved. This mode of operation derives from the fact that most supercomputer installations provide archival storage which may be used to preserve the check-point file during a restart run and subsequent check-point operations.

The state and time history plot files are preserved when a restart is performed. Upon restart, HYDRA will begin writing new state and time history data in the next family member of the plot files. Similarly, the global output data is simply concatenated to the existing *glob* file when a restart is performed. However, the human output file is re-written when a restart is performed.

HYDRA 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 at this time. However, changing material properties, the number of time steps, plot intervals, etc. is acceptable.

Chapter 4

HYDRA Input Data

The necessary input data for a HYDRA simulation is split into two files. The first file contains all of the control information for the problem, e.g., mesh parameters, analysis type, equation solver options, etc. The second file contains the nodal spatial coordinates, element connectivity, boundary conditions, etc. The following sections describe the necessary input data in the control and mesh files.

4.1 HYDRA Control File

The HYDRA control file uses a keyword-parameter syntax to define the mesh parameters, solver control variables, and analysis parameters. It is assumed that the first line of the control file is an 80 character description of the analysis. With the exception of the first line, the data contained in the control file is order independent and delimited by keywords for the mesh definition, solver controls, and analysis options. Comments in the control file must be preceded by a "C" and a blank space, or may be enclosed in a pair of braces "{ }". All input in the control file is case insensitive. Figure 4.1 shows the typical format of a HYDRA control file. In this section, default and allowable values for parameters are denoted by a **boldface** type.

4.1.1 Mesh Parameters

The mesh parameters are used to define the number of nodes and elements in the mesh, as well as, the number of materials, and problem dimension. Currently, only single material analyses are active in HYDRA.

Mesh Parameters

Keyword	Variable & Meaning
mesh	Mesh parameter starting delimiter.
nnp	Number of nodes (*no default*).
nel	Number of elements (*no default*).
nmat	Number of materials (*default:* **nmat=1**).
ndim	Number of dimensions (*default:* **ndim=3**).
end	Terminate mesh parameters.

```
Analysis Title {80 characters or less}
C A "C" followed by a blank space starts a comment line
{Comments may be enclosed in braces as well}
C The mesh-end block describes the mesh parameters
mesh
…
end
C The analyze-end block describes the analysis
parameters
analyze
…
end
C The momsol-end block defines the momentum equation
solver
momsol
…
end
C The ppesol-end block defines the PPE equation solver
ppesol
…
end
C The ndhist-end block defines the time-history nodes
ndhist
…
end

end
```

Figure 4.1: A Sample HYDRA Control File

4.1.2 Analysis Parameters

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 the number and type of boundary conditions, algorithm specific options such as hourglass stabilization, the number of times steps to take, termination time, output intervals, etc.

Analysis Parameters

Keyword	Variable & Meaning
analyze	Analysis parameter definition starting delimiter.
solve	**1**: Transient, incompressible, Navier-Stokes using 1-point quadrature, lumped mass, and Forward Euler time integration (*default.* **1**)
	3: Transient, incompressible, Navier-Stokes using full quadrature and P-II.
	101: Transient advection-diffusion using Forward Euler with a prescribed velocity field ($\hat{\underline{u}}$). This option makes use of 1-point integration.
	102: Transient advection diffusion using semi-implicit implicit time integration with a prescribed velocity field ($\hat{\underline{u}}$). This option makes use of fully integrated elements.
temp	**0/1**: Solve scalar advection-diffusion with momentum equations (*default:* **0**)
term	Simulation termination time (*default:* **1.0**).
nstep	Number of time steps (*default:* **10**).
deltat	time step size (*default:* **0.01**).
dtscal	time step scale factor (*default:* **1.0**).
dtchk	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, i.e., forces the grid parameters to be reported (*default:* **10**).
mass	**1**: Lumped Mass (*default:* **mass=1** for **solve=1,101**) **2**: Consistent Mass (*default:* **mass=2** for **solve=2,3,102**)

Analysis Parameters (cont.)

Keyword	Variable & Meaning
nubc	Number of prescribed x-velocity boundary conditions.
nvbc	Number of prescribed y-velocity boundary conditions.
nwbc	Number of prescribed z-velocity boundary conditions.
npbc	Number of prescribed pressure boundary conditions.
ntbc	Number of prescribed temperature boundary conditions.
nxbc	Number of prescribed x-traction boundary conditions.
nybc	Number of prescribed y-traction boundary conditions.
nzbc	Number of prescribed z-traction boundary conditions.
nkbc	Number of prescribed k essential boundary conditions.
nknb	Number of prescribed k natural boundary conditions
nebc	Number of prescribed ε essential boundary conditions.
nenb	Number of prescribed ε natural boundary conditions
nabc	Number of prescribed A_2 essential boundary conditions.
nanb	Number of prescribed A_2 natural boundary conditions
nper	Number of master/slave nodes for periodic boundary conditions. The periodic boundary conditions permit the anti-symmetry using the following optional parameters.
u_sym	**+/-1:** set anti-symmetry in the x or normal velocity
v_sym	**+/-1:** set anti-symmetry in the y or tangential velocity
w_sym	**+/-1:** set anti-symmetry in the w or tangential velocity
;	Required to terminate the nper command
hglass	**1**: h-stabilization hourglass control (*default:* **1**)
	2: γ-stabilization hourglass control (2-D only).
epshg	Hourglass coefficient (*default:* **1.0**)
btd	**0/1**: Balancing tensor diffusivity (BTD) (*default:* **1**).
divu	RMS divergence tolerance (*default:* **1.0e-10**).

Analysis Parameters (cont.)

Keyword	Variable & Meaning
nu	Fluid kinematic viscosity (*default:* **1.0**).
alpha	Fluid thermal diffusivity (*default:* **1.0**).
thetak	Time weight for viscous terms (*default:* **0.5**).
thetab	Time weight for BTD (*default::* **0.5**).
thetaa	Time weight for advection (*default:* **0.0**).
thetaf	Time weight for BC's (*default:* **0.5**).
icset	xxxxxxx: icset requires a 7-digit parameter and a ";" to terminate the command. The individual values of the 7-digit icset parameter are: **0**: Asumes initial conditions are 0.0 (*default:* **0**). **1**: Assumes initial conditions are 1.0 unless otherwise specified. **2**: Read in prescribed intitial conditions from mesh file.
	In the icset-; sequence, the following commands may be used to specify the values of initial conditions which all nodal points will inherit.
set_u	**U** : Sets the x-velocity initial condition to **U**.
set_v	**V** : Sets the y-velocity initial condition to **V**.
set_w	**U** : Sets the w-velocity initial condition to **W**.
set_t	**T** : Sets the t-velocity initial condition to **T**.
set_k	**k** : Sets the k-velocity initial condition to **k**.
set_e	**E** : Sets the ε-velocity initial condition to **E**.
set_a2	**A2** : Sets the A_2-velocity initial condition to **A2**.
;	Required to terminate the icset command
icwrt	**1**: Write the final velocity field to a file which may later be used to specify initial conditions for HYDRA. The name of the initial conditions file is just the name of the output file with ".ics" appended. **0**: Don't write an initial condition file (*default:* **0**).

Analysis Parameters (cont.)

Keyword	Variable & Meaning
prti	Hard copy print interval (*default*: **prti=10**).
prtlev	**0**: Print only analysis parameters to the output file (*default:* **0**).
	1: Print only the analysis parameters and result quantities to the output file.
	2: Print the analysis parameters, mesh coordinates, boundary conditions, etc., and the result quantities to the output file.
plti	Plot state output interval (*default*: **plti=10**).
ttyi	Interval to write min/max values to the screen (*default:* **ttyi=10**)
end	Terminate analysis parameters.

4.1.3 Momentum Equation—Solver Parameters

The solution of the Navier-Stokes equations using the P-II algorithm, or the solution of advection-diffusion equation with the semi-implicit algorithm requires the selection of an equation solver and the necessary associated 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 error norms. See Chapter 3 for definitions of the error norms used in HYDRA's iterative solvers.

Momentum Equation—Solver Parameters

Keyword	**Variable Meaning**
momsol *type*	Momentum equation solver parameter starting delimiter where *type* may be one of the following: **1**: Jacobi pre-conditioned conjugate gradient.
momsol=1 solver options:	
itmax	Iteration limit (*default*: **itmax=10**).
itchk	Interval to check convergence (*default*: **itchk=5**).
eps	Convergence tolerance (*default*: **eps=10^{-10}**).
wrt	**0**: Suppress diagnostic information from solver (*default*). **1**: Write diagnostic information from solver.
hist	**0**: Suppress writing an ASCII convergence history file (*default*). **1**: Write an ASCII convergence history file.
end	Terminate momentum equation solver parameters.

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 which can perform a variety of time integration schemes, it is implied that the user select appropriate convergence criteria and limits. The selection of *solve 3* or *solve 102* implicitly requires the user to set the iteration limit and convergence criteria. If the iteration limit and convergence criteria are not specified, the default values will be used.

4.1.4 Pressure Poisson Equation—Solver Parameters

In the solution of the Navier-Stokes equations using either the explicit, implicit (P-II) 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.

Pressure Poisson Equation (PPE)—Solver Parameters

Keyword	Variable Meaning
ppesol *type*	PPE solver parameter starting delimiter where *type* may be one of the following (*default:* **type=1**): **1**: Element-by-element, Jacobi-preconditioned conjugate gradient (EBE/JPCG). **11**: UDU storage format Gaussian elimination. **12**: UDU storage format ITLIB solver options (UDU/JPCG). **41**: Parallel-vector (PVS) row solver. **51**: Algebraic multigrid solver (AMG). **61**: compact storage ITPACK Jacobi-preconditioned conjugate gradient (JPCG). **62**: compact storage ITPACK Jacobi-preconditioned conjugate gradient (JPCG) for the stabilized pressure element. **64**: compact storage ITPACK symmetric successive over-relaxation preconditioned conjugate gradient (SSOR-PCG).
ppesol=1 EBE/JCG solver options:	
itmax	Iteration limit (*default*: **itmax=10**).
itchk	Interval to check convergence (*default:* **itchk=5**).
eps	Convergence tolerance (*default*: **eps=1.0e-10**).
wrt	**0**: Suppress diagnostic information from solver (*default).* **1**: Write diagnostic information from solver.
hist	**0**: Suppress writing an ASCII convergence history file (*default).* **1**: Write an ASCII convergence history file.
ppesol=12 UDU/JCG solver options:	
itmax	Iteration limit (*default:* **itmax=10**).
itchk	Interval to check convergence (*default:* **itchk=5**).
eps	Convergence tolerance (*default:* **eps=1.0e-10**).
wrt	**0**: Suppress diagnostic information from solver (*default).* **1:** Write diagnostic information from solver.
hist	**0**: Suppress writing an ASCII convergence history file (*default).* **1**: Write an ASCII convergence history file.

Pressure Poisson Equation (PPE)—Solver Parameters (cont.)

Keyword	Variable & Meaning
ppesol=51 AMG solver options:	
iout	AMG output options (default: **iout=10**). 1st digit of iout: not used; has to be non-zero. 2nd digit of iout: 0: no output (except for messages). 1: residual before and after solution process. 2: statistics on cp-times and storage requirements. 3: residual after each amg-cycle.
levelx	AMG grid level control must be greater than 1 (*default*: **levelx=5**).
ifirst	AMG initial guess control (*default*: **ifirst=13**). 1st digit of ifirst: not used; has to be non-zero. 2nd digit of ifirst—itypu: 0: no setting of first approximation, 1: first approximation constant to zero, 2: first approximation constant to one, 3: first approximation is random function with the concrete random sequence being determined by the following digits. rest of ifirst—rndu: determines the concrete random sequence used in the case **itypu=3**. (**ifirst=13** is equivalent to **ifirst=1372815**)
ncyc	AMG cycle (*default:* **ncyc=10110**). ncyc is an integer parameter describing the type of cycle to be used and the number of cycles to be performed. 1st digit of ncyc defines cycle: 1: v -cycle, 2: v*-cycle, 3: f -cycle, 4: w -cycle. if ncyc is negative, then the approximation of the problem on the second finest grid is computed by igam v-cycles on that particular grid. 2nd digit of ncyc: 0: no conjugate gradient, 1: conjugate gradient (only first step of cg),

Pressure Poisson Equation (PPE)—Solver Parameters (cont.)

Keyword	Variable & Meaning
	3rd digit of ncyc—iconv: convergence criterion for the user-defined & problem (finest grid): 1: perform a fixed number of cycles as given by ncycle (see below) 2: stop, if $\|r^n\| \le \varepsilon$ 3: stop, if $\|r^n\| \le \varepsilon\|b\|$ 4: stop, if $\|r^n\| \le \varepsilon\|x^n\|\,\|diagonal\|$ with $\|r^n\|L_2$ norm of residual, ε (see input parameter ε) $\|f\|$ = supremum norm of right hand side, $\|u\|$ = supremum norm of solution $\|diag\|$ = maximal diagonal entry in matrix . Note that in any case the solution process stops after at most ncycle cycles. Rest of ncyc—ncycle: maximal number of cycles to be performed (> 0) or ncycle=0: no cycling.
mad	AMG adaptivity options (*default:* **mad=27**). mad is an integer value specifying the choice of coarsest grid in cycling: 1st digit of **mad**: 1: in cycling, all grids constructed in the setup phase are used without check. 2: the number of grids is automatically reduced if the convergence factor on the coarser grids is found to be larger than a given value fac (see below). rest of mad—fac; the rest of mad defines the fractional part of a real number fac between 0.1 and 0.99, e.g. mad=258 means msel=2 and fac=0.58. If mad consists of only one digit, fac is set to 0.7 by default.
nrd	AMG downward relaxation option (*default:* **nrd=1131**). Parameter describing relaxation (downwards): 1st digit of nrd: not used; has to be non-zero. 2nd digit of nrd—nrdx: actual number of smoothing steps to be performed the type of which is given by the following digits following digits— array nrdtyp: 1: relaxation over the f-points only 2: full GS sweep 3: relaxation over the c-points only 4: full more color sweep, highest color first

Pressure Poisson Equation (PPE)—Solver Parameters (cont.)

Keyword	Variable & Meaning
nsolco	AMG coarsest grid solver option (*default*: **nsolco=110**). Parameter controlling the solution on coarsest grid: 1st digit—nsc: 1: gauss-seidel method 2: direct solver (yale smp) rest of nsolco—nrcx: (only if nsc=1) number of GS sweeps on coarsest grid (≥ 0). If nrcx=0, then as many GS sweeps are performed as are needed to reduce the residual by two orders of magnitude. (maximal 100 relaxation sweeps)
nru	AMG upward relaxation option (*default*: **nru=1131**). Parameter describing relaxation (downwards): 1st digit of nrd: not used; has to be non-zero. 2nd digit of nrd—nrdx: actual number of smoothing steps to be performed the type of which is given by the following digits following digits—array nrdtyp: 1: relaxation over the f-points only 2: full GS sweep 3: relaxation over the c-points only 4: full more color sweep, highest color first
eps	AMG convergence tolerance (*default*: **eps=1.0e-10**).
ppesol=61 JPCG solver options:	
itmax	Iteration limit (*default:* **itmax=10**).
itchk	Interval to check convergence (*default*: **itchk=5**).
eps	Convergence tolerance (*default:* **eps=1.0e-10**).
wrt	**0**: Suppress diagnostic information from solver (*default).* **1**: Write diagnostic information from solver.
hist	**0**: Suppress writing an ASCII convergence history file (*default).* **1**: Write an ASCII convergence history file.
ppesol=62 JPCG solver options - stabilized element:	
	Same as for solver 61 above

Pressure Poisson Equation (PPE)—Solver Parameters (cont.)

Keyword	Variable & Meaning
ppesol=64 SSOR-PCG solver options:	
itmax	Iteration limit (*default:* **itmax=10**).
itchk	Interval to check convergence (*default*: **itchk=5**).
eps	Convergence tolerance (*default:* **eps=1.0e-10**).
omega	Relaxation parameter (*default:* **omega=1.0**).
wrt	**0**: Suppress diagnostic information from solver (*default).*
	1: Write diagnostic information from solver.
hist	**0**: Suppress writing an ASCII convergence history file (*default).*
	1: Write an ASCII convergence history file.
end	Terminate PPE solver parameters.

4.1.5 Time History Blocks

Time history nodes may be defined to track primitive nodal variables at a small number of nodes where the interval that the data is recorded at is much smaller than for state data. This type of data is useful for detecting steady-state conditions, and for the evaluation of periodic behavior.

Time History Blocks

Keyword	Variable & Meaning
ndhist *n*	Specify n time history nodes (*default*: **n=0**).
st	Starting node number in block.
en	Ending node number in block.
nstep	Time history output interval (*default:* **nstep=1**).
end	Terminate time history block

4.1.6 Turbulence Models

Turbulence models and their associated parameters are specified in the HYDRA control file. For all two equation models and second-order closure models, the default for *ndof* is automatically set when the user activates the turbulence model.

Turbulence Models

Keyword	Variable & Meaning
turb *n*	Activate turbulence model # *n* (*default:* **n=0**). 1: Smagorinsky subgrid scale model without dynamic subgrid scale. 102: Lein and Leschziner[13] k-ε model - cartesian grids only. 103: Shuga, Craft, Launder[10] k-ε–A_2 model (not available yet).
n=1 Smagorinsky Model:	
smagc	Set the value of the Smagorinsky constant (*default:* **smagc=0.1**).
end	Terminate turbulence model block

4.2 HYDRA Mesh File

The HYDRA mesh file contains an 80 character comment line, the spatial coordinates of the nodes, nodal connectivity, all specified initial conditions, and boundary conditions. The 80 character comment line must be the first line in the input file and is echoed in the human readable output file. Lines may be commented out by using a "C" followed by a blank space, or by using #, *, $, or enclosing a region of the input file in braces, { }. Because the data in the mesh file is usually generated by an automatic mesh generator, the data in this file can't be input in a format-free style as in the case of the HYDRA control file. However, all input in the mesh file is case insensitive.

For many problems, load curves for essential boundary conditions are not necessary, e.g., no-slip boundaries or boundaries with prescribed velocity are the most common. Therefore, a blank field for the load curve number associated with a boundary condition indicates that the amplitude of the boundary condition should be used as the boundary condition. That is, the boundary condition does not vary with time. This applies to all prescribed boundary conditions in HYDRA.

4.2.1 Nodal Coordinates

nnp nodal coordinates are required in this section of the mesh file. In the case of a 2-D analysis, the z-coordinate may be left blank.

Nodal Coordinates

Columns	Format	Description
14-33	E20.0	x-coordinate.
34-53	E20.0	y-coordinate.
54-73	E20.0	z-coordinate. (Ignored for 2-D)

4.2.2 Connectivity—Q1/P0 elements

The node numbers and material numbers associated with *nel* elements are required. For 2-D simulations, HYDRA ignores the last 4 nodal numbers in the connectivity.

Connectivity—Q1/P0 Elements

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 2-D)

4.2.3 Velocity Initial Conditions

This section may be skipped if the first three digits of the *icset* parameter has been set to 0 or 1. If any of the first three digits has a value of 2, then HYDRA will read *Nnp* velocity initial conditions and discard those velocity components which have been set through the *icset* parameter. The z-velocity field is ignored for 2-D analyses.

Velocity Initial Conditions

Columns	Format	Description
1-10	I10	Node number.
11-30	E20.0	Specified initial x-velocity.
31-50	E20.0	Specified initial y-velocity.
51-70	E20.0	Specified initial z-velocity. (Ignored for 2-D)

4.2.4 Thermal Initial Conditions

If *temp=1* and the *icset* paramter has been specified to be *xxx2xxx* in the control file, then *Nnp* temperature values must be input for the thermal initial conditions. For any other values of the *icset* parameter, this section of input may be skipped.

Thermal Initial Conditions

Columns	Format	Description
1-10	I10	Node number.
11-30	E20.0	Specified initial temperature.

4.2.5 Prescribed Turbulence Model Initial Conditions

This section may be skipped if the last three digits of the *icset* parameter has been set to 0 or 1. If any of the last three digits has a value of 2, then HYDRA will read *Nnp* turbulence initial conditions and discard those values which have been set through the *icset* parameter. The A_2 field is ignored for analyses with *turb=102*.

Turbulence Model Initial Conditions

Columns	Format	Description
1-10	I10	Node number.
11-30	E20.0	Specified initial turbulent kinetic energy (k).
31-50	E20.0	Specified initial dissipation rate (ε).
51-70	E20.0	Specified initial Reynolds Stress 2nd Invariant.

4.2.6 Periodic Boundary Conditions

nper pairs of master/slave node numbers must be input for the specification of periodic boundary conditions. It is up to the user (or the user's mesh generator) to verify that master nodes and slave nodes and their associated element align spatially. If the spatial alignment of nodes is not respected, HYDRA may produce unexpected results. Note that multiple levels of indirection may not be used to define periodic boundary conditions. That is, a master node may not also be a slave node and vice versa.

Periodic Boundary Conditions

Columns	Format	Description
1-10	I10	Master node number.
11-20	I10	Slave node number.

4.2.7 Prescribed Boundary Conditions

The format used for the input of all specified essential and natural boundary conditions is shown in the table below. For non-time dependent boundary conditions, the load curve may be left blank (effectively zero), and the amplitude of the prescribed prescribed boundary condition will be taken as the value of the boundary condition.

Prescribed Boundary Conditions

Columns	Format	Description
1-8	I8	Node number.
10-19	E10.0	Amplitude of prescribed boundary condition.
20-24	I5	Load curve number.

The order of the required essential and natural boundary conditions in the mesh file is shown in the table below with the corresponding analysis parameter from the control file and its associated mesh input requirements. In the table which follows, EBC refers to an essential boundary condition, e.g., specified value of velocity. In contrast, NBC refers to a natural boundary condition, e.g., specified tractions.

Analysis Parameters & Boundary Conditions

Variable	Parameter	Description of Input
X-Velocity (u)	nubc	nubc values of specified x-velocity.
Y-Velocity (v)	nvbc	nvbc values of specified y-velocity.
Z-Velocity (w)	nwbc	nwbc values of specified z-velocity
Pressure (P)	npbc	npbc values of the element level pressure*.
Temperature (T)	ntbc	ntbc values of the specified temperature.
Turbulent ke (k)	nkbc	nkbc values of the turbulent kinetic energy.
Dissipation rate (ε)	nebc	nebc values of the specified dissipation rate.
Reynolds Invariant	nabc	nabc values of the Reynolds stress 2nd invariant.
X-Traction (f_X)	nxbc	nxbc values of the x-traction component.
Y-Traction (f_Y)	nybc	nybc values of the y-traction component.
Z-Traction (f_Z)	nzbc	nzbc values of the z-traction component.
k - NBC's	nknb	nknb NBC's for the turbulent kinetic energy.
ε - NBC's	nenb	nenb NBC's for the dissipation rate.
A_2 - NBC's	nanb	nanb NBC's of the Reynolds stress 2nd invariant.

*Note that the element level pressures are not boundary conditions in the strict sense of the term. Rather, this input to HYDRA simply fixes element level pressures according to the input value. This feature is typically used to remove spurious modes from the pressure-poisson equation when they are present, e.g., in a lid-driven cavity calculation. Specification of element level pressures in this sense is drastically different from the specification of a pressure boundary condition via a non-zero boundary traction.

4.3 The Advection-Diffusion Equation

The following initial condition and boundary condition specifications apply only to the solution of the scalar advection-diffusion equation. In this case, it is assumed that the prescribed initial velocity distribution is fixed in time.

4.3.1 Prescribed Velocity Field

nnp values of nodal velocity must be provided if any one of the first three digits of the *icset* parameters is 2 in the control file. Otherwise, *icset* controls the value of the prescribed velocity field. For 2-D problems, the z-velocity component is ignored.

Prescribed Velocity Field

Columns	Format	Description
1-10	I10	Node number.
11-30	E20.0	Prescribed initial x-velocity.
31-50	E20.0	Prescribed initial y-velocity.
51-70	E20.0	Prescribed initial z-velocity. (Ignored for 2-D.)

4.3.2 Temperature Initial Conditions

nnp values of nodal temperature must be provided if the *icset* parameter has been specified as *xxx2xxx* in the control file. Otherwise, *icset* controls the value of the prescribed initial temperature field using the *set_t* command.

Temperature Initial Conditions

Columns	Format	Description
1-10	I10	Node number.
11-30	E20.0	Prescribed initial temperature.

4.3.3 Temperature Boundary Conditions

ntbc values of specified temperature must be input for the advection diffusion equation. See the table in section 4.2.7 for additional information.

Temperature Boundary Conditions

Columns	Format	Description
1-8	I8	Node number.
10-19	E10.0	Prescribed temperature.
20-24	I5	Load curve number.

Chapter 5

Example Problems

This chapter presents several 2-D and 3-D HYDRA calculations. These computation are provided for the first time user who wishes to perform several benchmark computations for comparison before embarking on a detailed analysis. For this reason, the control files are replicated here with representative results which can be compared to for a rough validation of the local HYDRA installation. These problems are not intended to be a comprehensive benchmark suite. Instead, the calculations presented here are simply intended to provide the first time user with a set of test cases which can be used for code familiarization.

For this reason, most of the sample problems use relatively coarse meshes to minimize run times and provide a starting point for the user who wishes to experiment with code options before attempting any significant calculations. All of the sample calculations are isothermal, but they would require minimal changes to activate the scalar advection-diffusion equation.

5.1 Entrance Region in a 2-D Duct

The entrance region sample problem consists of a 2-D duct with a 5:1 aspect ratio, and a plug inlet velocity profile at $x=0$. Natural boundary conditions are applied at the outflow boundary, and no-slip boundary conditions are prescribed on the top and bottom surfaces. The vertical velocity is prescribed to be zero on the inlet plane. Fig. 5.1 shows the mesh for the entrance region.

The HYDRA control file is shown in Fig. 5.2 for the case when the Reynolds number based on the channel height is 100. In this example, a time-accurate, P-II computation is performed for the 200 element inlet duct mesh.

A segment of the screen output for this calculation is shown in Fig. 5.3. The divergence for the specified initial conditions failed the divergence test, so a mass-consistent projection to a divergence-free subspace was performed on the initial velocity data. The resulting velocity field yielded an rms divergence of 1.0474E-17. The maximum grid CFL number was 0.53391 at t=0 for this problem.

The HYDRA simulation for this problem was performed in two parts. Initially, the computation was carried out to only 5 time units. At this point in the simulation, the velocities were not changing noticeably, however, the kinetic energy still had a positive slope. Thus, a restart was performed, and the simulation was extended another 5 time units.

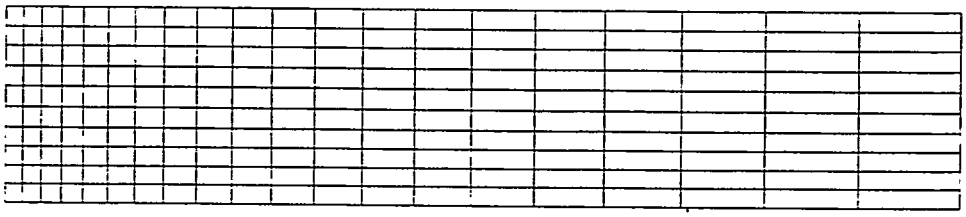


Figure 5.1: Mesh for 2-D Entrance Region.

```
2-D Duct Entrance Region, Re = 100

{Define mesh parameters}
mesh
 nnp 231 nel 200 nnpe 4
 nmat 1 ndim 2
end

{ Define analysis parameters }
analyze
 solve    3
 nubc    53
 nvbc    53
 nstep  100
 plti    10
 prti    50
 pltype   1
 divu   1.0e-10
 icset    2220000 ;
 term   10.00
 deltat  0.05
 dtchk   -1
 dtscal     1.0
 rho     1.0
 nu      1.0e-2
 thetak  0.5
 thetab  0.5
end

{ pvs direct solver for the PPE }
ppesol 41 end

{ Momentum eq. solver parameters }
momsol  1
 itmax 20
 itchk 5
 eps  1.0e-5
 wrt  0
end

{ Select time history blocks }
ndhist 3
 nstep 1
 st 12 en 22
 st 144 en 154
 st 221 en 231
end

end
```

Figure 5.2: 2-D Entrance Region Control File.

```
D i v e r g e n c e E r r o r
 Allowable Divergence ..................... 1.0000E-10
 Initial Divergence .......................   1.1379E-08
 Projected Divergence ..................... 1.0474E-17
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. ..........................          1
 Min. xi-element dimension ................. 8.7298E-02
 Minimum xi-grid Reynolds Number ........... 4.3649E+00
 Max. Element no. ..........................        140
 Max. xi-element dimension ................. 5.3391E-01
 Maximum xi-grid Reynolds Number ........... 5.3391E+01

 eta-grid Reynolds Number:
 =========================
 Min. Element no. ..........................        158
 Min. eta-element dimension ................ 1.0000E-01
 Minimum eta-grid Reynolds Number .......... 2.1019E-09
 Max. Element no. ..........................         79
 Max. eta-element dimension ................ 1.0000E-01
 Maximum eta-grid Reynolds Number .......... 1.1933E-04

 xi-grid CFL Numbers:
 ====================
 Min. Element no. ..........................        200
 Min. xi-element dimension ................. 5.3391E-01
 Minimum xi-grid CFL Number ................ 4.6825E-02
 Max. Element no. ..........................        161
 Max. xi-element dimension ................. 8.7298E-02
 Maximum xi-grid CFL Number ................ 5.7275E-01

 eta-grid CFL Numbers:
 =====================
 Min. Element no. ..........................        158
 Min. eta-element dimension ................ 1.0000E-01
 Minimum eta-grid CFL Number ............... 1.0510E-10
 Max. Element no. ..........................         79
 Max. eta-element dimension ................ 1.0000E-01
 Maximum eta-grid CFL Number ............... 5.9665E-06

  Step #      Time       U-Min.     U-Max.     V-Min.     V-Max.
========== ========== ========== ========== ========== ==========
     0     0.0000E+00 0.0000E+00 0.1000E+01 -.4104E-06 0.3512E-06
    10     0.5000E+00 -.5100E-25 0.1114E+01 -.7581E-01 0.7581E-01
     ...     .
    90     0.4500E+01 -.5099E-25 0.1362E+01 -.7912E-01 0.7912E-01
   100     0.5000E+01 -.5099E-25 0.1365E+01 -.7912E-01 0.7912E-01
... Restart performed ...
 Step #       Time       U-Min.     U-Max.     V-Min.     V-Max.
========== ========== ========== ========== ========== ==========
     0     0.5000E+01 -.5099E-25 0.1365E+01 -.7912E-01 0.7912E-01
    10     0.5500E+01 -.5099E-25 0.1366E+01 -.7912E-01 0.7912E-01
     ...
    90     0.9500E+01 -.5099E-25 0.1367E+01 -.7912E-01 0.7912E-01
   100     0.1000E+02 -.5099E-25 0.1367E+01 -.7912E-01 0.7912E-01
```

Figure 5.3: Grid Parameters and Velocity Min/Max Values.

The stream function contours for the velocity field are shown in Fig. 5.4 at 100 time units. The corresponding pressure field contours are shown in Fig. 5.5. Velocity time history plots are shown in Fig. 5.6 for the x and y-velocity components along the duct centerline (nodes 17, 149 and 226). The outlet x-velocity profile in Fig. 5.7 shows the expected parabolic velocity distribution for hydrodynamically fully developed flow. The kinetic energy for this calculation is also shown in Fig. 5.8.

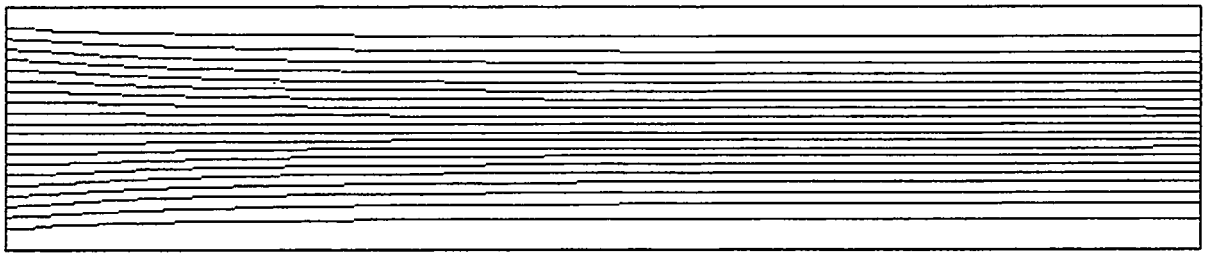


Figure 5.4: Stream function contours after 10 time units.

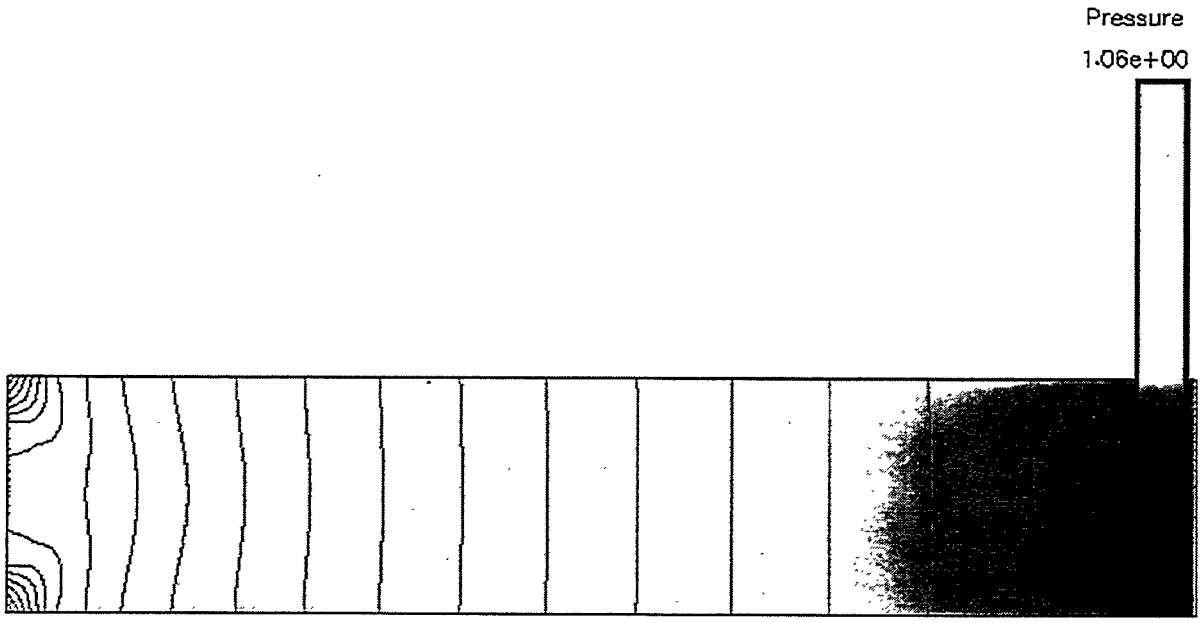
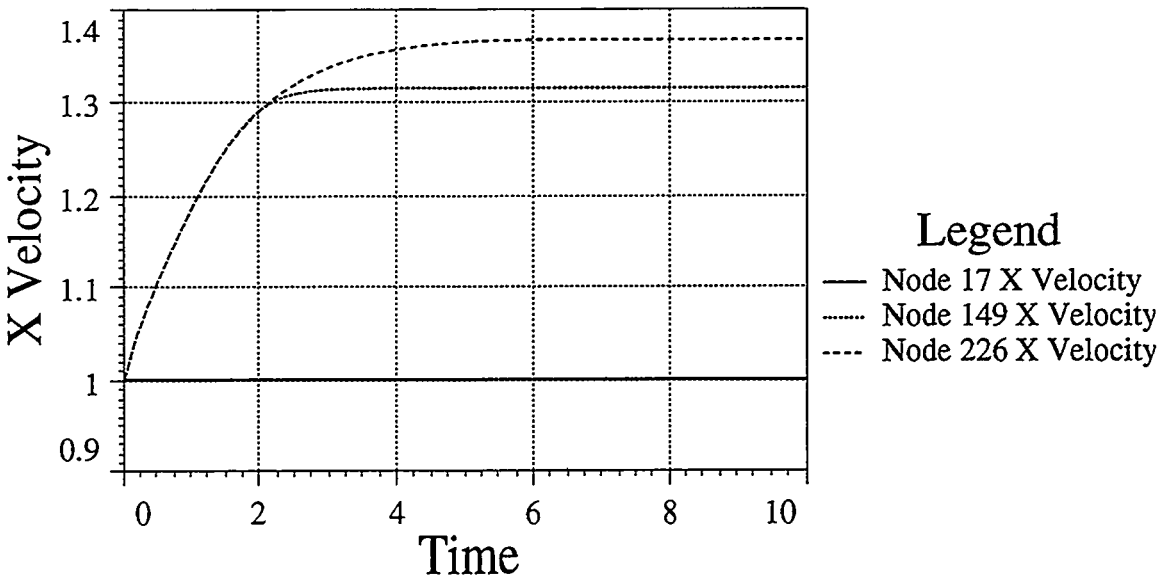
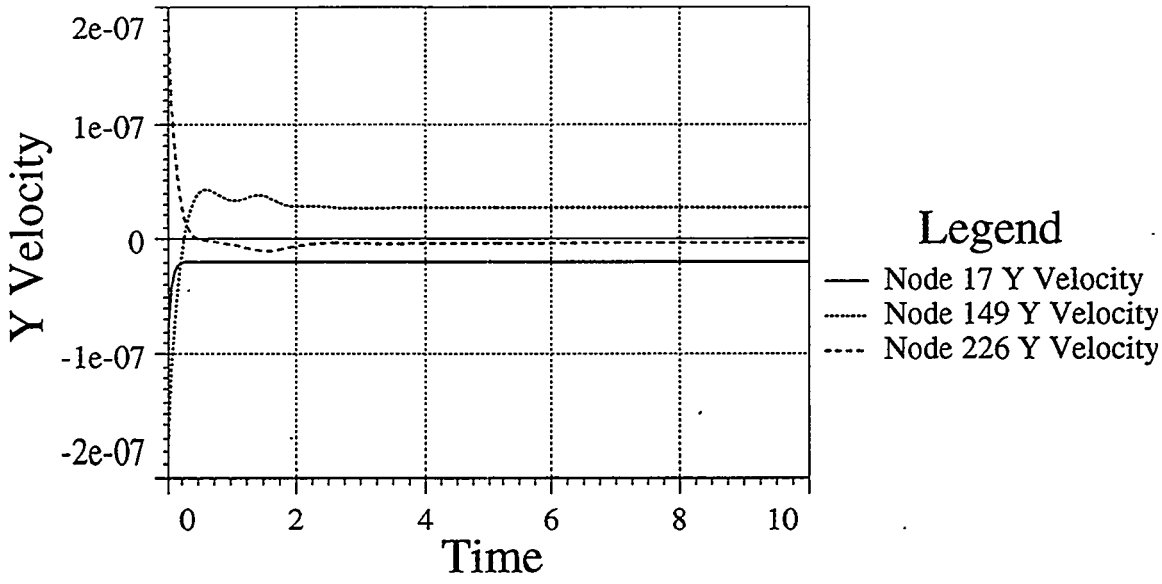


Figure 5.5: Pressure contours after 10 time units.



a) *x*-velocity time history.



b) y-velocity time history.

Figure 5.6: Velocity time history plot for 2-D duct.

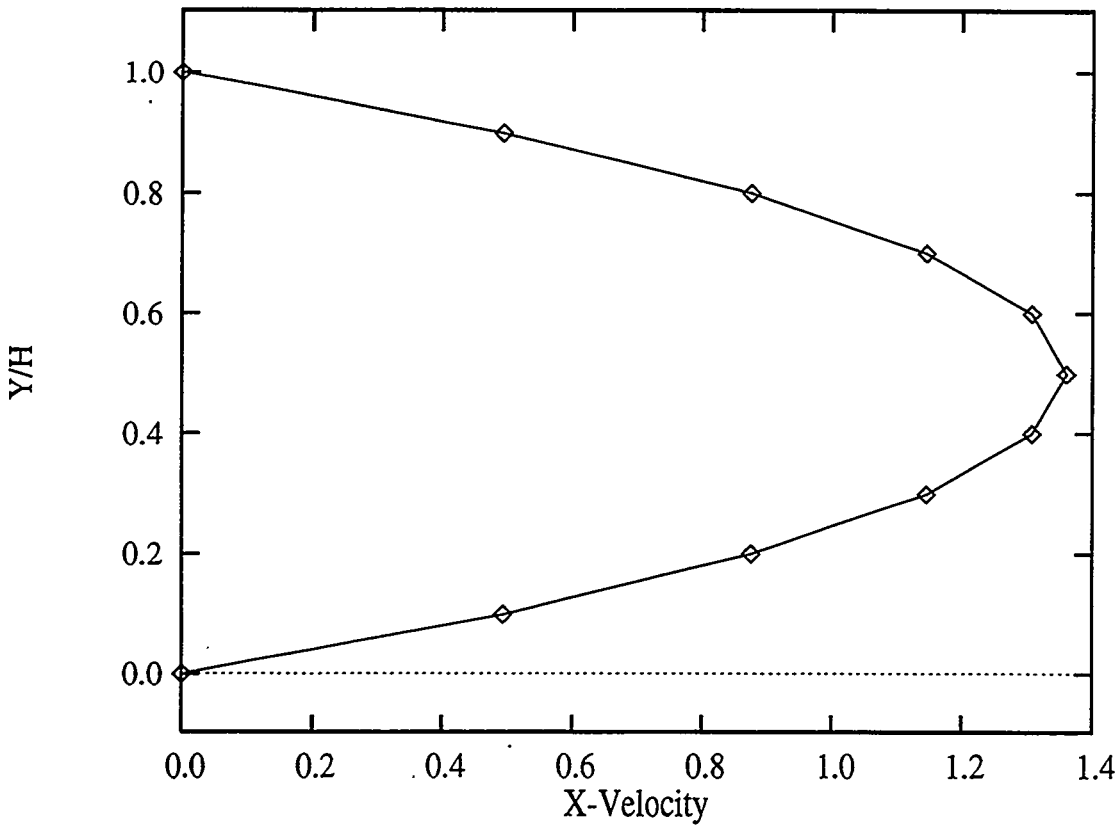


Figure 5.7: x-velocity profile at outlet plane.

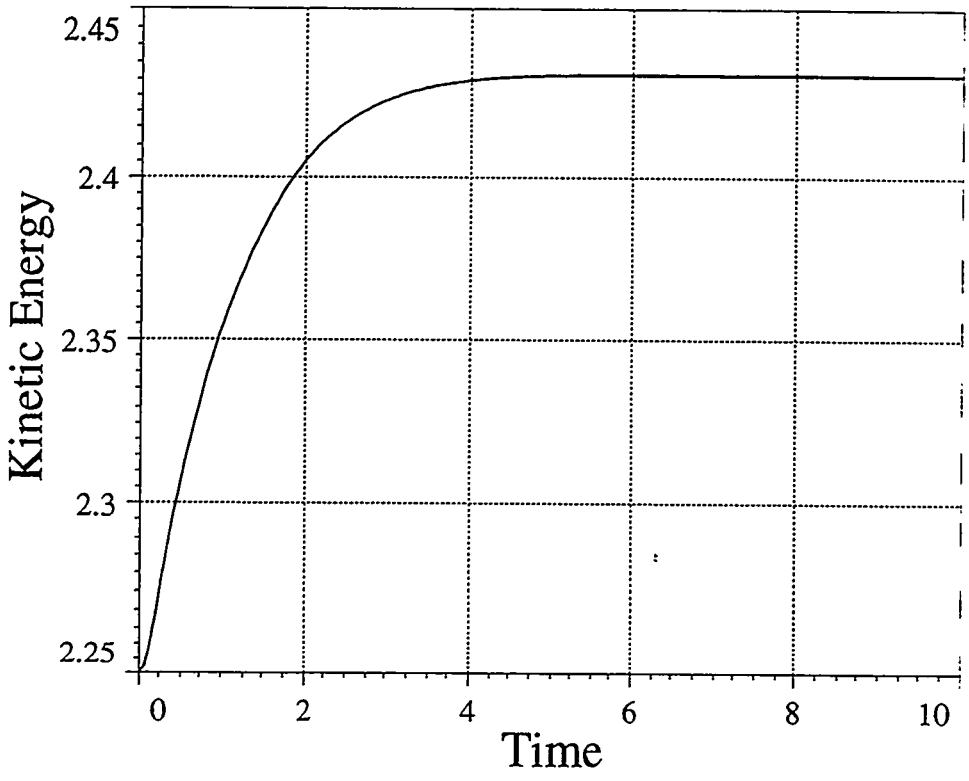


Figure 5.8: Kinetic energy time history.

5.2 Lid Driven Cavity

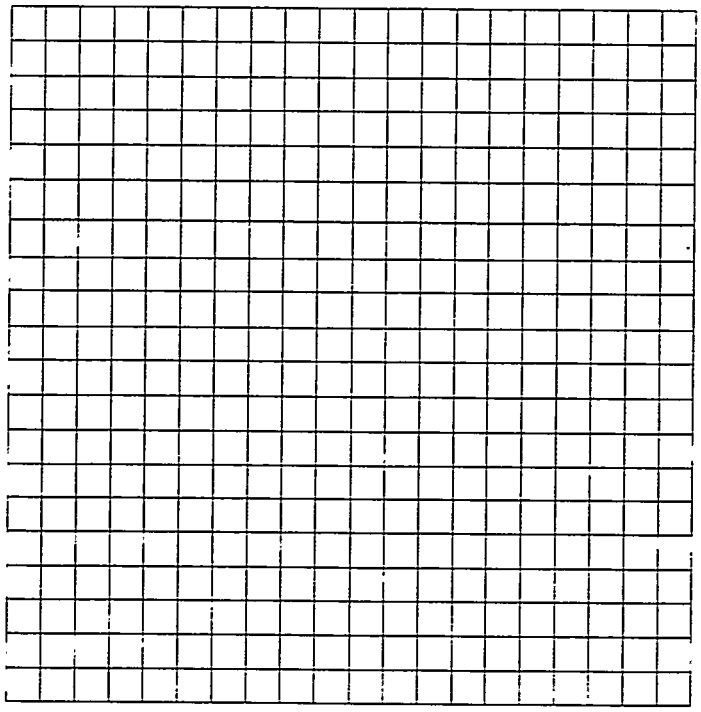
A standard benchmark problem is the computation of the flow in a lid driven cavity. The lid driven cavity problem consists of a square, unit domain discretized with a 20 x 20 element mesh as shown in Fig. 5.9. All boundaries are no-slip boundaries with a unit x-velocity specified at the top of the cavity. The computations for this problem were performed with a non-leaky lid. That is, the corner nodes at the lid had u=v=0, and while the first node in from each corner of the lid had u=0.5, v=0.

The HYDRA control file for the mesh with 441 nodes is shown in Fig. 5.10. The Reynolds based upon the cavity dimension is 100, and based upon this discretization, the maximum grid Reynolds and CFL numbers were 2.5. A time accurate P-II computation was performed for a duration of 25 time units at which a steady-state velocity field was achieved.

The non-leaky lid driven cavity problem results in both a hydrostatic and a checkerboard mode in the pressure field. Thus, *npbc 2* is specified in the control file, and the two pressures at the bottom, center of the cavity are pegged at 0.0.

The pressure, stream function, and vorticity contours at t=25 units are shown in Figures 5.11-5.13. In comparison to the results of Hughes, et al., the results for these quantities are qualitatively similar. Figure 5.14 shows the velocity vectors for the problem after 25 time units. Examination of the kinetic energy time history in Fig. 5.15 shows that problem is essentially steady-state after only 10 time units.

Figure 5.9: 20 x 20 lid driven cavity mesh.



```
Lid driven cavity Re = 100

mesh
 nnp   441
 nel   400
 nnpe   4
 nmat   1
 ndim   2
end

analyze
 solve   3
 nubc   84
 nvbc   84
 npbc   2
 nstep 100
 plti   20
 prti   100
 pltype   1
 icset   2220000 ;
 divu   1.0e-10
 deltat 0.25
 dtchk -1
 term   1000.00
 rho   1.0
 nu    1.0e-2
end

ndhist 4
 nstep 1
 st 25 en 25
 st 149 en 149
 st 419 en 419
 st 311 en 311
end

ppesol 41 end

momsol 1
 itmax 400
 itchk 10
 eps 1.0e-5
 wrt 0
end
```

Figure 5.10: Lid driven cavity control file.

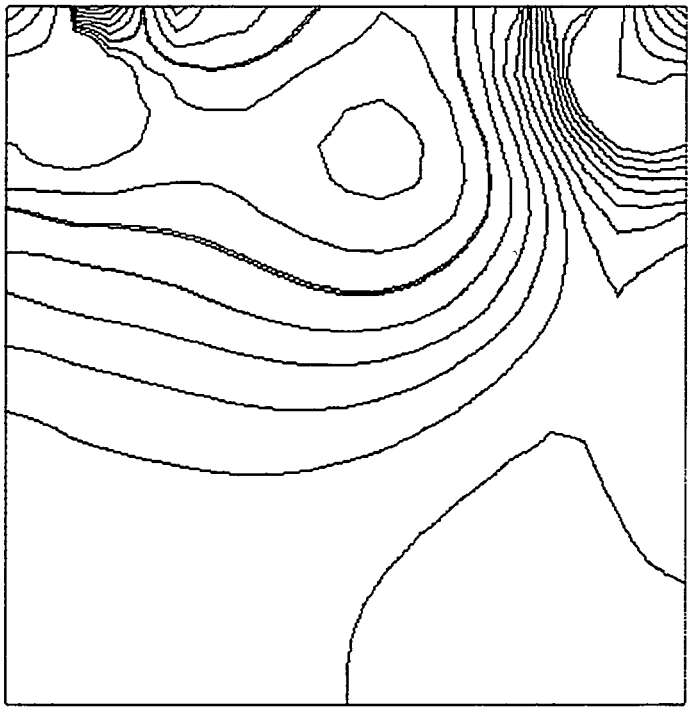


Figure 5.11: Lid driven cavity pressure contours.

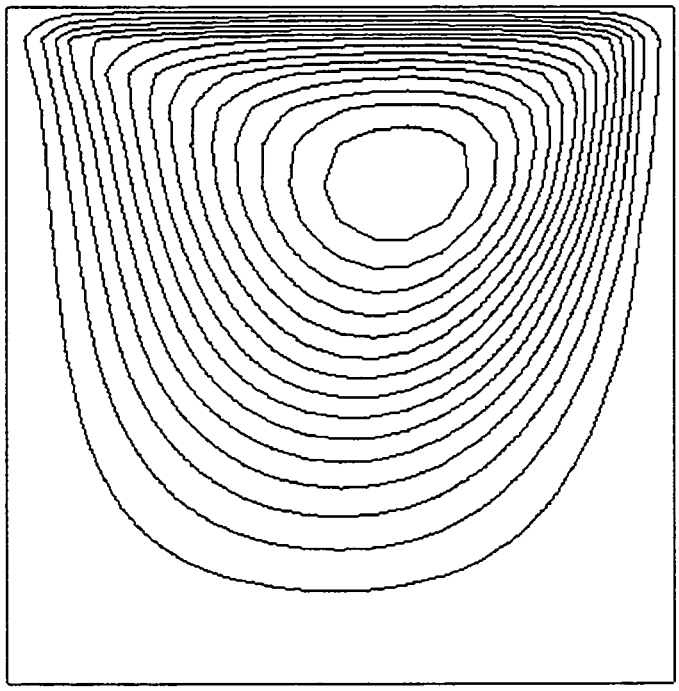


Figure 5.12: Lid driven cavity stream function countours.

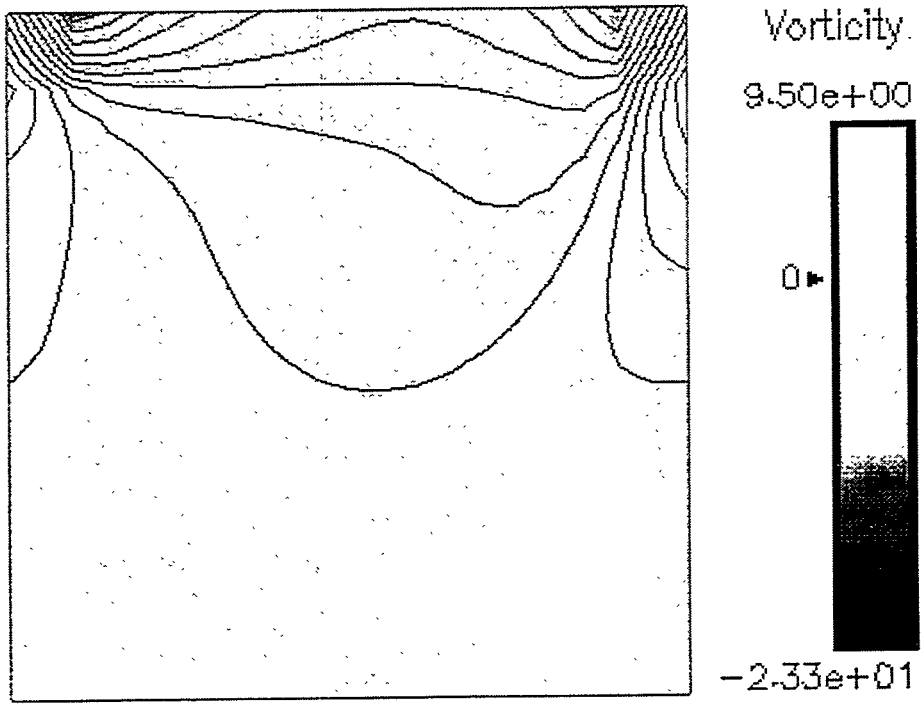


Figure 5.13: Lid driven cavity vorticity contours.

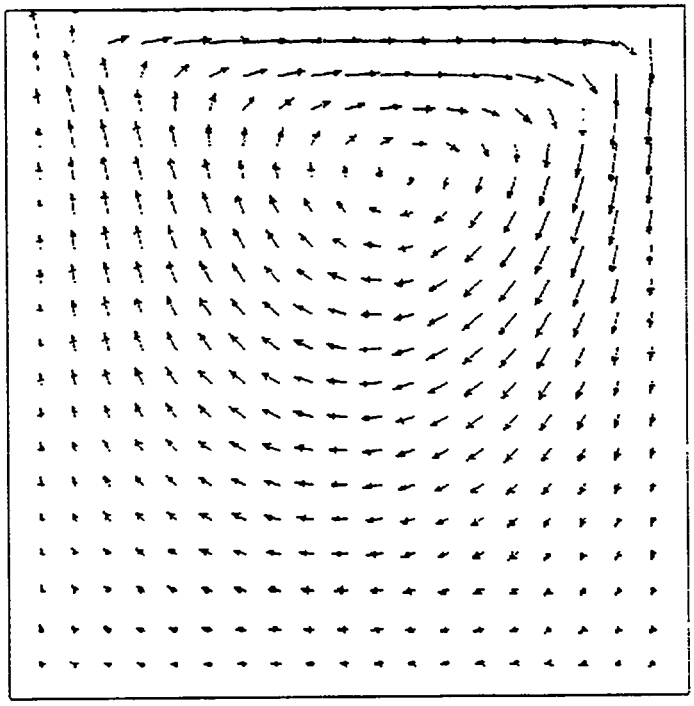


Figure 5.14: Lid driven cavity velocity vectors.

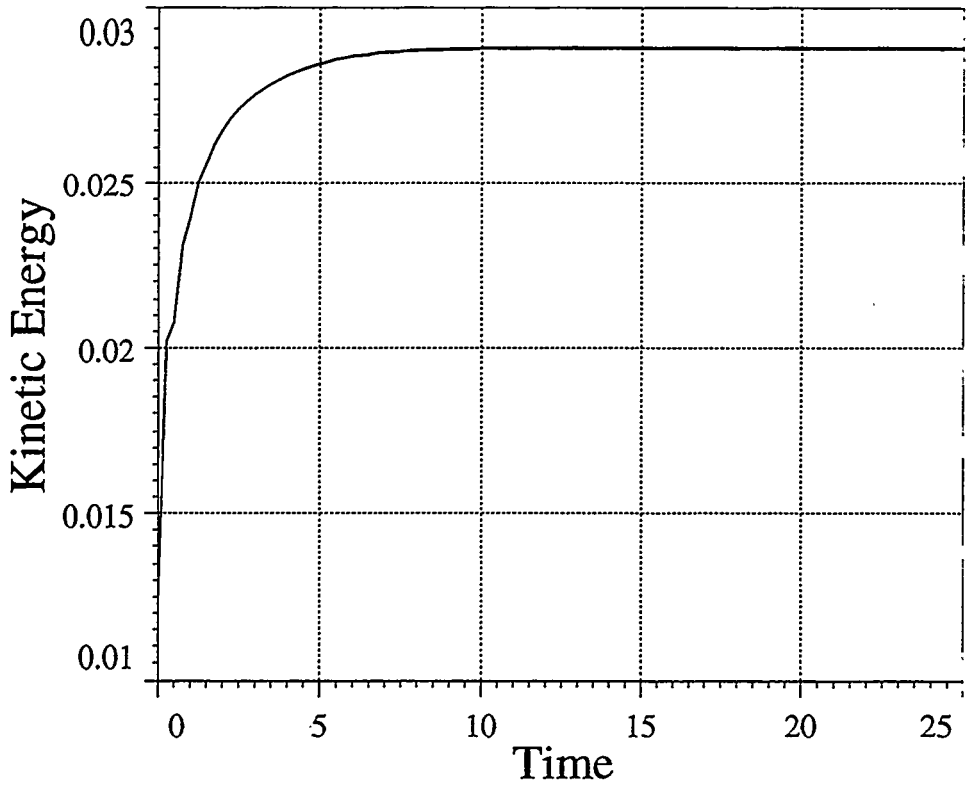


Figure 5.15: Lid driven cavity kinetic energy time history.

5.3 Vortex Shedding

The flow past a circular cylinder is also a common benchmark problem used to test CFD codes. This sample problem is essentially a duplicate of the vortex shedding problem discussed in Gresho, *et al.*[5].

The mesh in Fig. 5.16 consists of 1760 elements and 1852 nodes and is a close approximation to that of Gresho[5]. Boundary conditions consist of no-slip at the cylinder wall, and tow-tank conditions ($u=1$, $v=0$) on the inlet and edges of the domain. Natural boundary conditions are applied at the outflow boundary of the domain.

For this grid, CFL=6.8884 and Re=76.897 after the initial divergence-free projection. After a quasi-steady flow development of approximately 125 time units, the flow transitions to a steady, periodic vortex shedding cycle. Fig. 5.18 shows a snapshot of the stream function countours after 250 time units. The vorticity is also shown in Fig. 5.19 at this time in the simulation. Both the stream function contours and the vorticity plot clearly reveal the presence of the well-known Karman vortex street downstream of the cylinder.

Nodal time history plots of the x and y-velocity are shown in Fig. 5.20. The nodal time history points are located at the edge of the boundary layer around the cylinder, and just downstream of the cylinder. These plots clearly reveal the relatively long quasi-steady portion of the flow simulation in which two standing, symmetric eddies exist downstream of the cylinder. From these plots, the Strouhal number can be estimated for the periodic portion of the flow. Based upon the y-velocity, St=0.175 which compares favorable with the results in Gresho, *et al.*[5]. The kinetic energy time history is presented in Fig. 5.21.

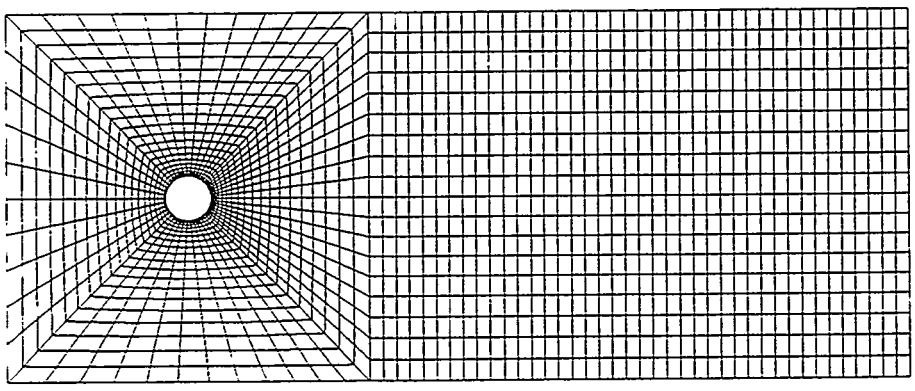


Figure 5.16: Coarse mesh for cylinder vortex shedding.

```
Coarse mesh vortex shedding - Re=100

mesh
 nnp   1852
 nel   1760
 nnpe   4
 nmat   1
 ndim   2
end

{ Max. deltat 0.025 for explicit time integration }
analyze
 solve   3
 nubc   175
 nvbc   175
 nstep 1000
 plti   50
 prti   500
 pltype   1
 icset   2220000 ;
 btd    1
 divu   1.0e-10
 deltat 0.250
 dtchk   -1
 dtscal   1
 term 500.00
 rho   1.0
 nu    1.0e-2
end

ndhist 5
 nstep 1
 st 441 en 441
 st 447 en 447
 st 653 en 673
 st 844 en 844
 st 899 en 899
end

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

ppesol 41 end


 end
```

Figure 5.17: Control file for vortex shedding.

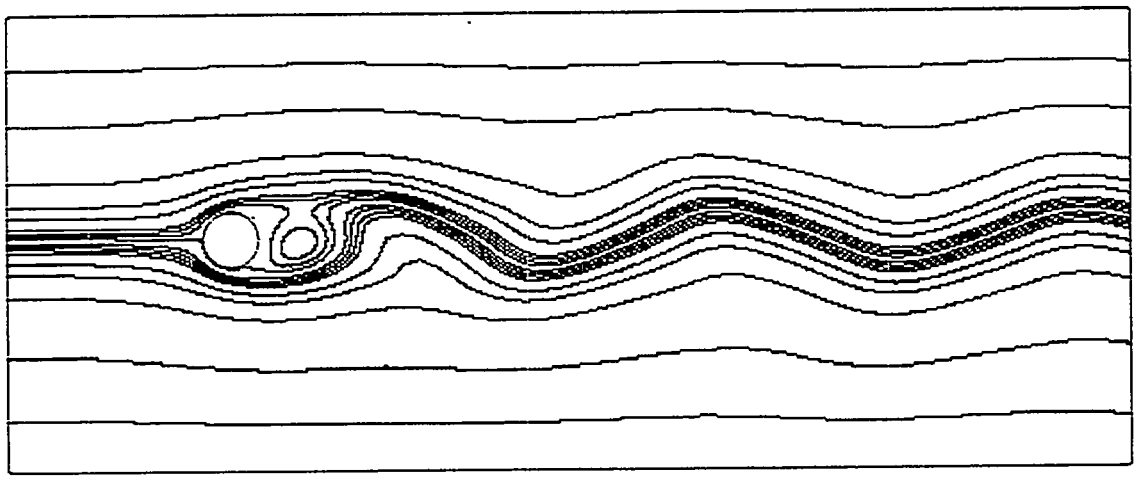


Figure 5.18: Stream function contours. ψ = +/- 0.05, 0.1, 0.15, 0.2, 0.4, 0.6, 1.0, 2.0, 3.0 at $\tau = 250$ time units.

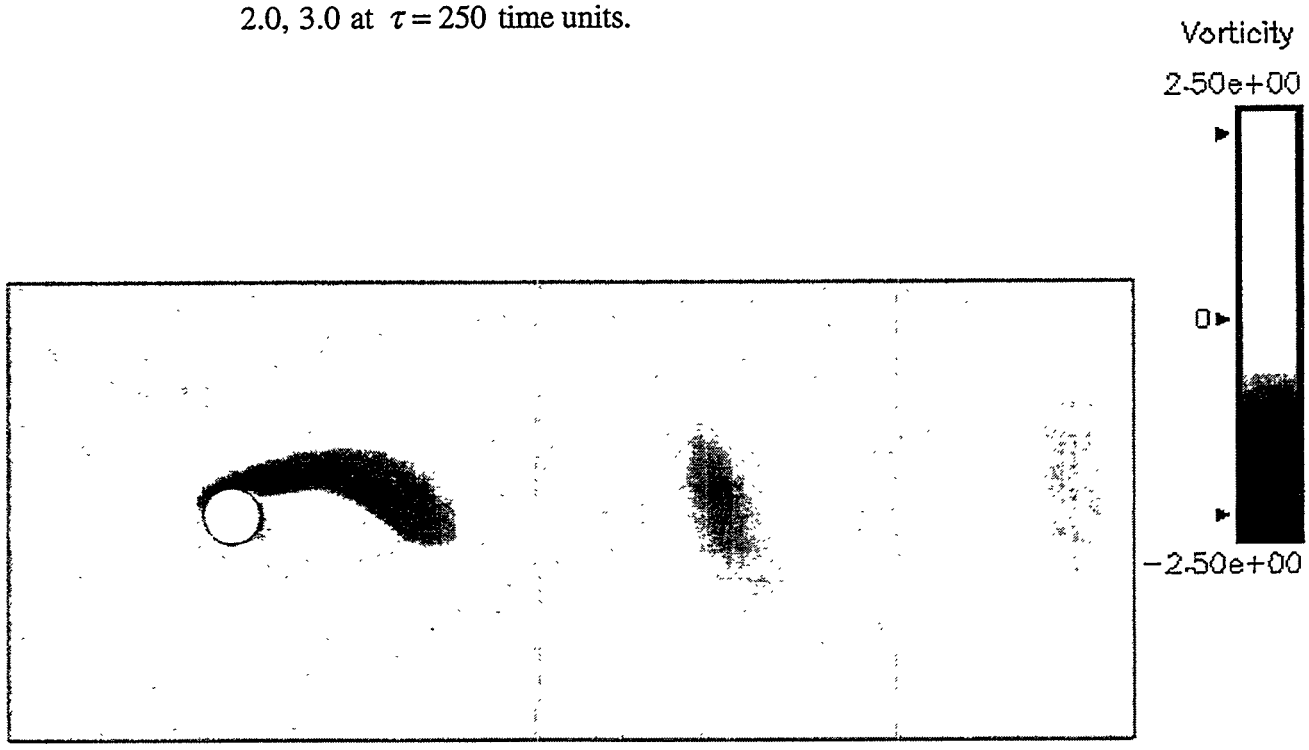
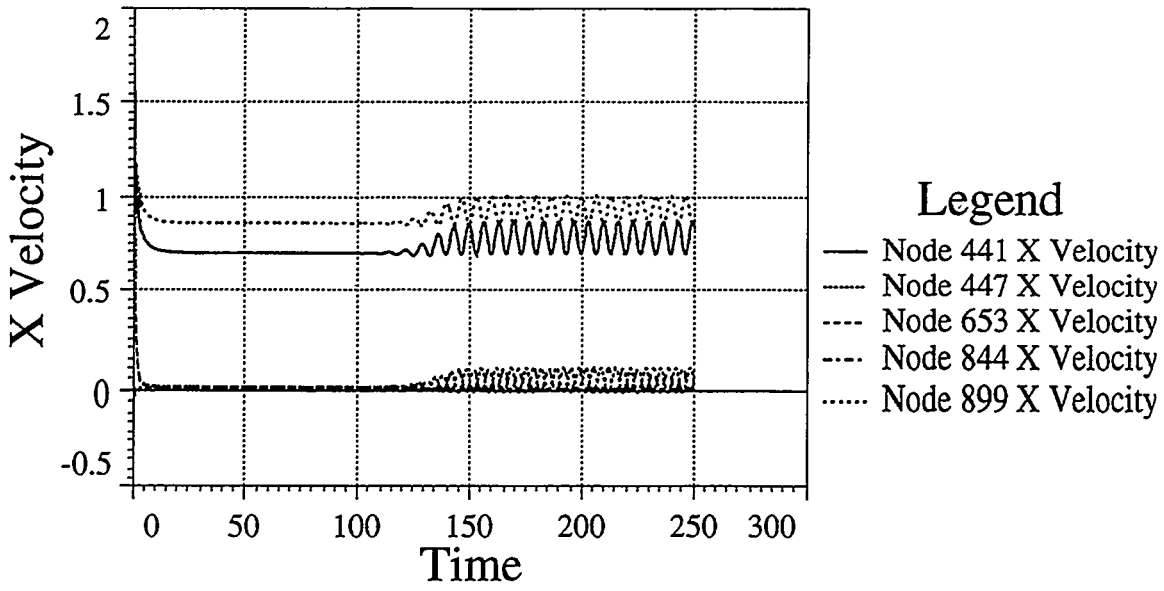
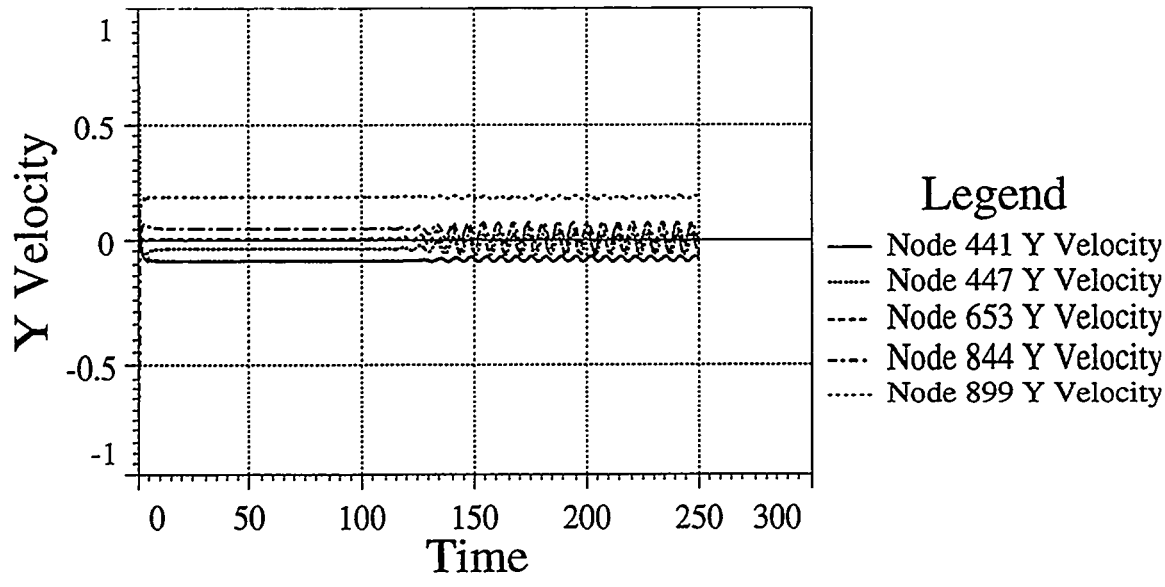


Figure 5.19: Vorticity snapshot at $\tau = 250$ time units.



a) x-velocity time histories.



b) y-velocity time histories.

Figure 5.20: Velocity time history plots at nodes 441, 447, 653, 844, and 899.

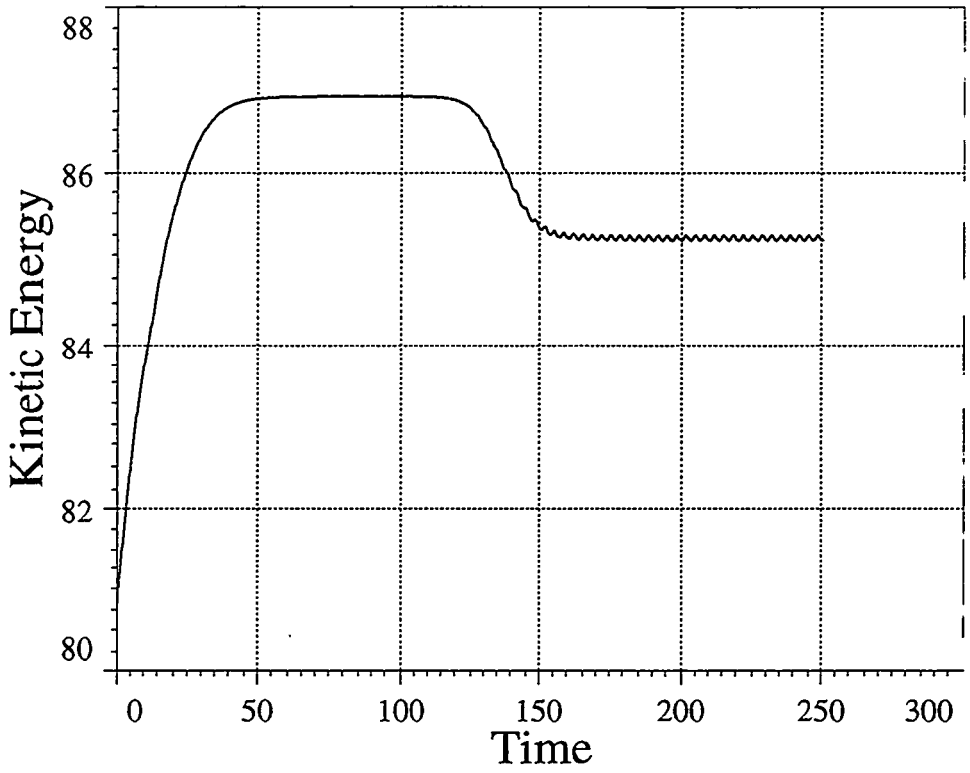


Figure 5.21: Vortex shedding kinetic energy time history.

5.4. Backward Facing Step

The inclusion of at least one backward facing step example seems appropriate given the attention which this problem has received as a CFD benchmark in the past few years. Once again, this problem is not included to provide a validation test for HYDRA. Instead, it is intended to be a sample problem for code familiarization.

The backward facing step consists of a domain with a 15:1 aspect ratio (channel length to height). The mesh consists of 2000 elements and is graded as shown in Fig. 5.22. The inlet x-velocity above the step is a fully developed parabolic profile, while the y-velocity is specified as zero. No-slip boundary conditions are applied at all walls, and natural boundary conditions are prescribed at the outflow boundary.

For $Re=800$, a steady-state solution results. The stream function and vorticity contours at $t=250$ time units are shown in Fig. 5.24–5.25 respectively. The reattachment point on the lower boundary is approximately 5.40 units downstream from the step. In comparison to the results of Gartling, this is a 11.48 percent error, but with comparably few elements.

The velocity time history plots in Fig. 5.26 show that the velocity field is still changing, although the kinetic energy time history in Fig. 5.27 appears to be steady. (The time history nodes were selected at $x=5.866$, $y=0.1, 0.5, 0.9$ according.)

Entrance
Region

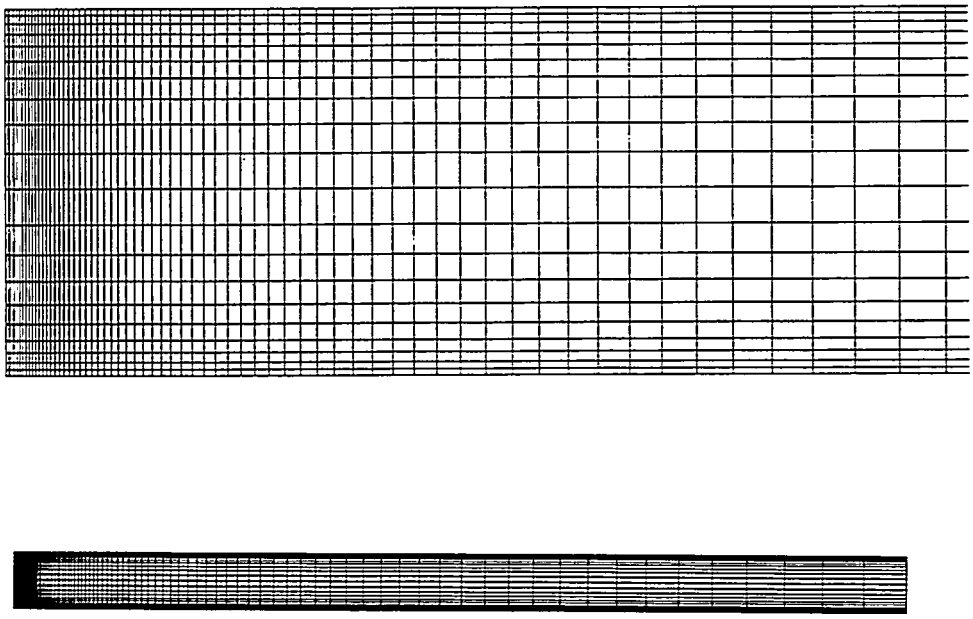


Figure 5.22: Mesh for Re=800 backward facing step.

```
BFS #4 - Re=800, parabolic inflow
mesh
 nnp   2121
 nel   2000
 nnpe   4
 nmat   1
 ndim   2
end

analyze
 solve   3
 nubc   222
 nvbc   224
 nstep 2500
 plti   50
 prti   250
 pltype 1
 btd   1
 icset   0000000 ;
 divu   1.0e-10
 deltat 0.100
 dtchk   -1
 dtscal   1
 term 500.00
 rho   1.0
 nu    1.25e-3
end
ndhist 8
 nstep 1
 st 902 en 902
 st 896 en 896
 st 1927 en 1927
 st 1050 en 1050
 st 1056 en 1056
 st 2067 en 2067
 st 528 en 528
 st 672 en 672
end
momsol 1
 itmax 250
 itchk 20
 eps 1.0e-6
 wrt 0
end

ppesol 41 end



end
```

Figure 5.23: Control file for backward facing step.

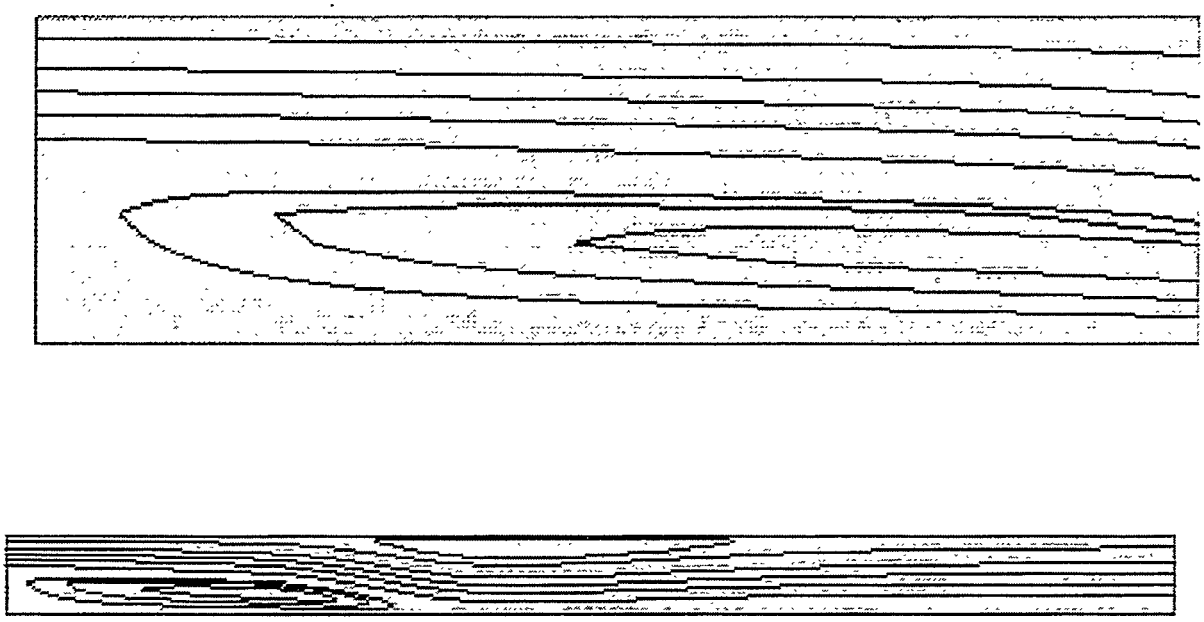


Figure 5.24: Stream function contours. psi = 0.0, +/- 0.1,+/-0.2, 0.225, -0.275, -0.285, -0.295 at $\tau = 250$ time units.

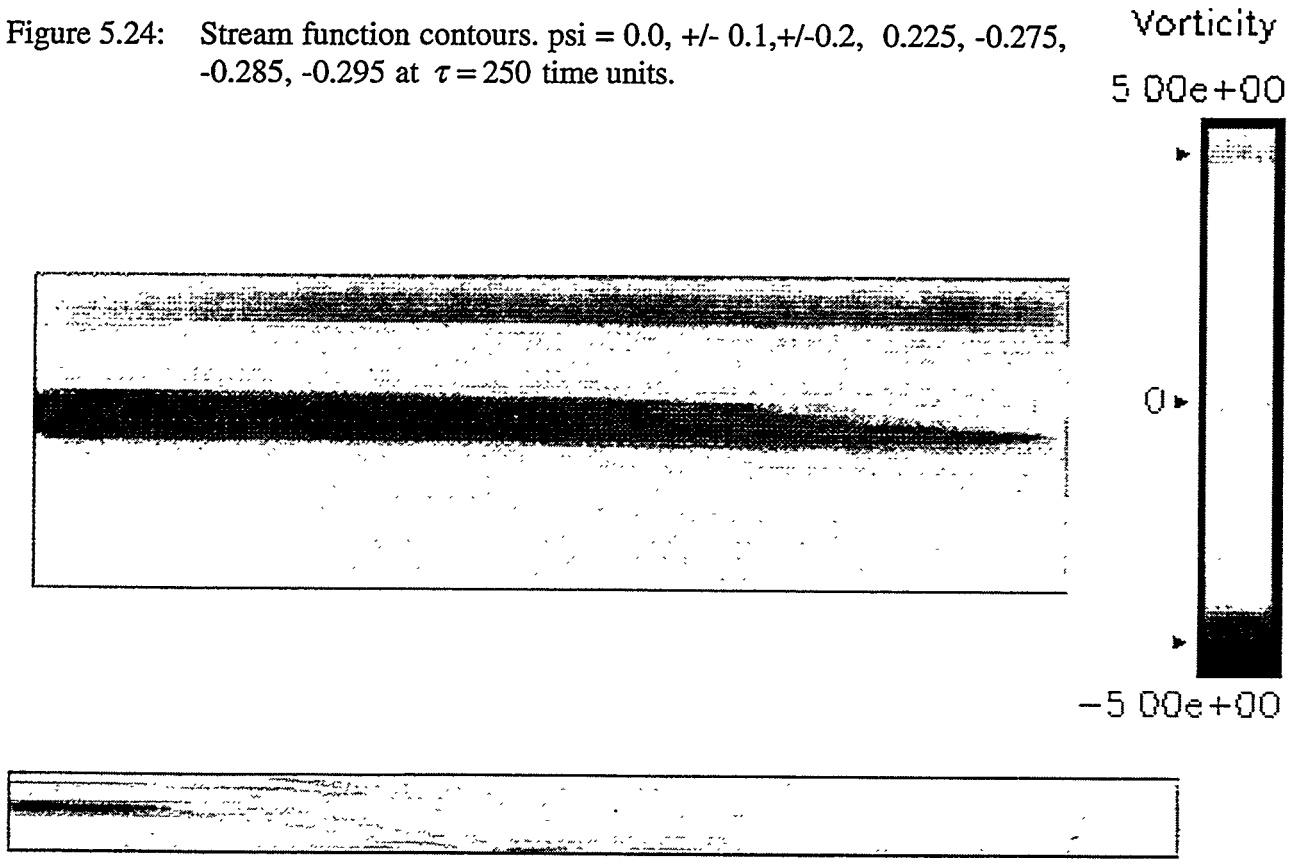
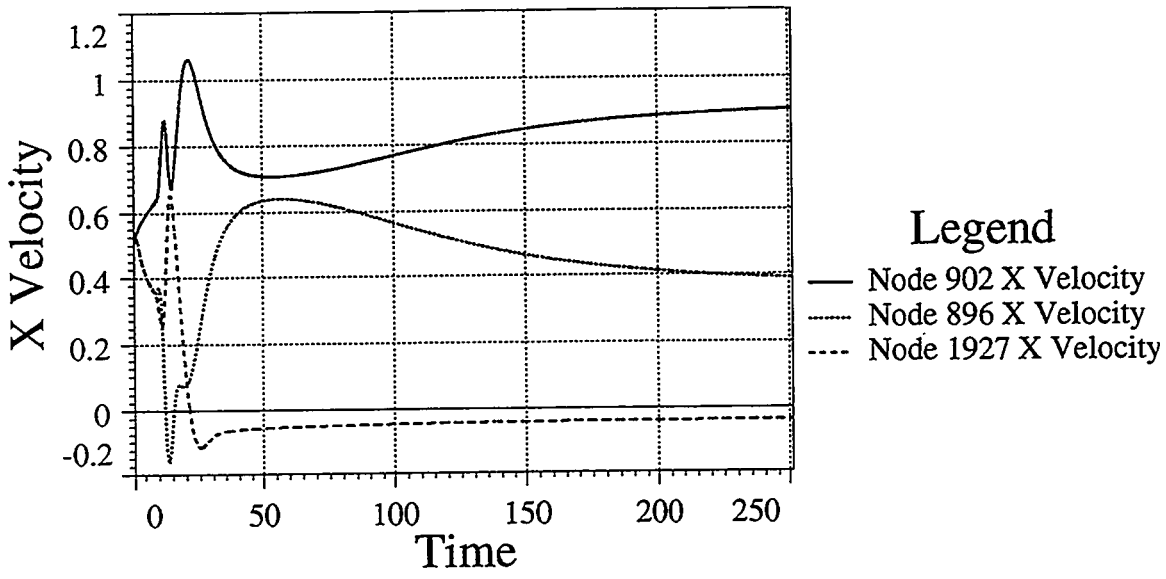
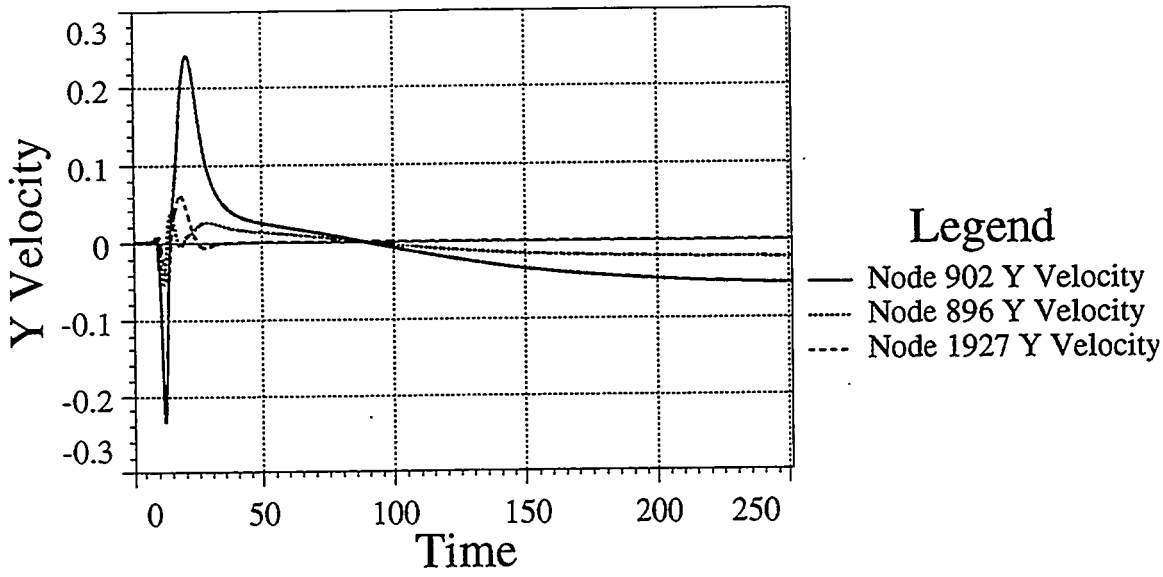


Figure 5.25: Vorticity contours at $\tau = 250$ time unites.



a) x-velocity time history plot.



b) y-velocity time history plot.

Figure 5.26: Velocity time history plots at nodes 896, 902, 1927.

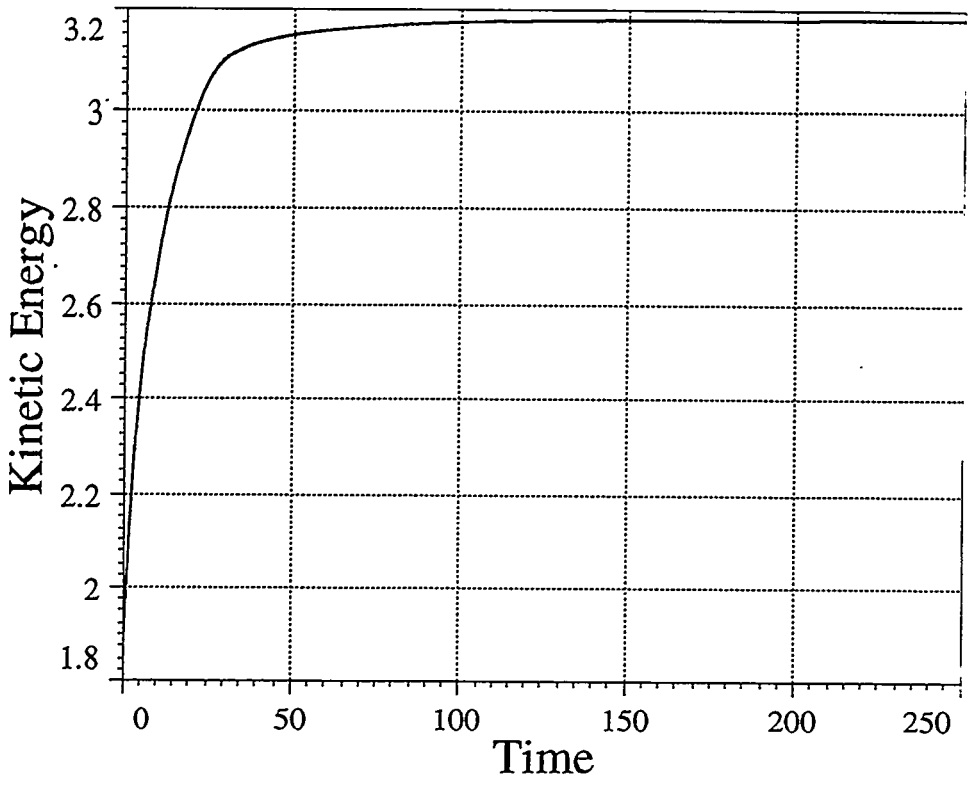


Figure 5.27: Kinetic energy time history.

5.5 Circular Duct Entrance Region

This example is the 3-D counterpart to the example presented in section 5.1. The circular duct example makes use of symmetry, but relies on a 3-D computation for the axis-symmetric flow solution. The 3300 element mesh for this problem is shown in Fig. 5.28.

At the wall, no-slip boundary conditions are prescribed. A plug-flow inlet velocity profile is prescribed, and natural boundary conditions are applied at the outflow boundary. Two planes of symmetry are uses, i.e., the x-z plane requires $w=0$ and the y-z plane requires $u=0$. The Reynolds number is 100 based upon the inlet velocity and duct diameter.

A steady-state solution is achieved after approximately 4 time units. The pressure contours for $y=100$ are shown for the inlet region in Fig. 5.30, and the z-velocity contours are shown in Fig. 5.31. Time history plots for the y-velocity, z-velocity (axial velocity), and kinetic energy are shown in Figures 5.32 and 5.33.

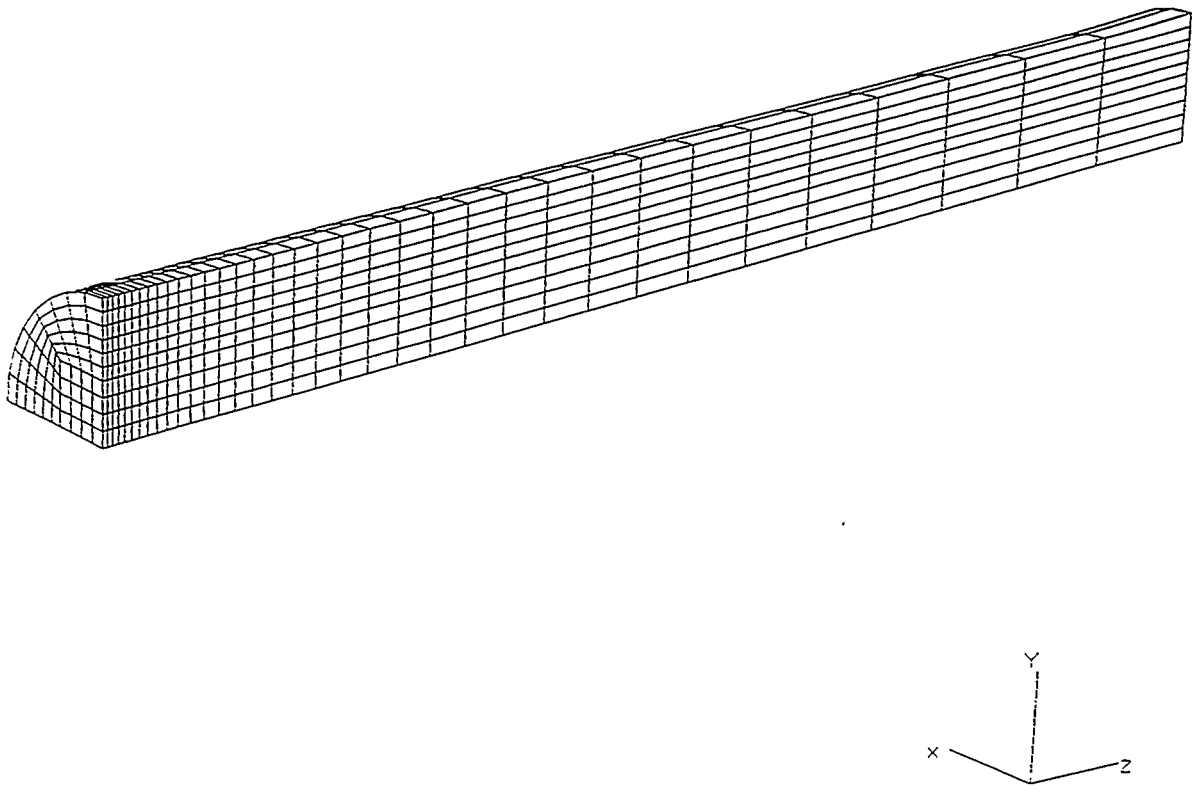


Figure 5.28: Circular duct entrance region mesh.

```
Duct entrance region Re=100
mesh
 nnp   3978
 nel   3300
 nnpe   8
 nmat   1
 ndim   3
end
analyze
 solve   3
 nubc   915
 nvbc   915
 nwbc   525
 nstep   100
 plti   10
 prti   100
 pltype   1
 icset   2220000 ;
 divu   1.0e-10
 btd   1
 dtchk   -1
 dtscal  1.0
 deltat  1.0e-1
 term   10.00
 rho   1.0
 nu    1.0e-2
end
ndhist 11
 nstep 1
 st 36 en 36
 st 72 en 72
 st 108 en 108
 st 144 en 144
 st 180 en 180
 st 936 en 936
 st 972 en 972
 st 1008 en 1008
 st 1044 en 1044
 st 1080 en 1080
 st 1116 en 1116
end
momsol 1
 itmax 50
 itchk 5
 eps 1.0e-5
 wrt 0
end

ppesol 41 end    end
```

Figure 5.29: Circular duct entrance region control file.

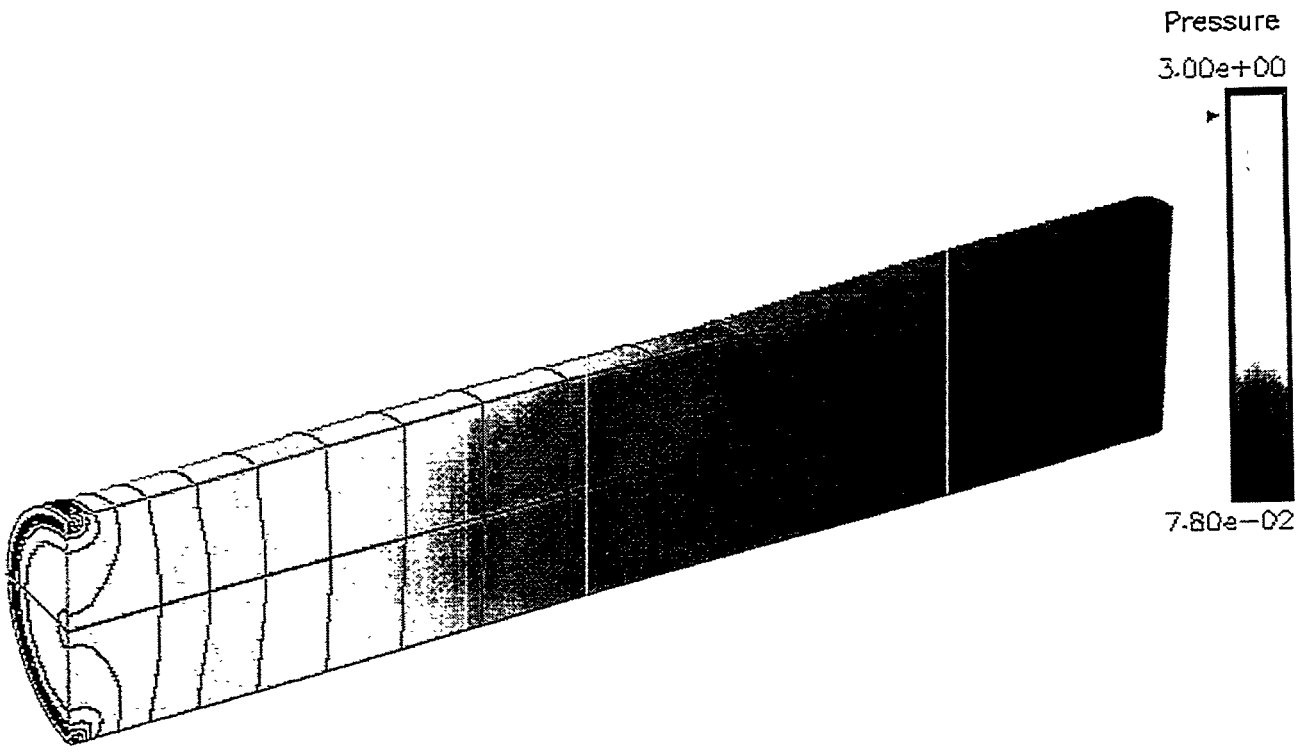


Figure 5.30: Pressure contours.

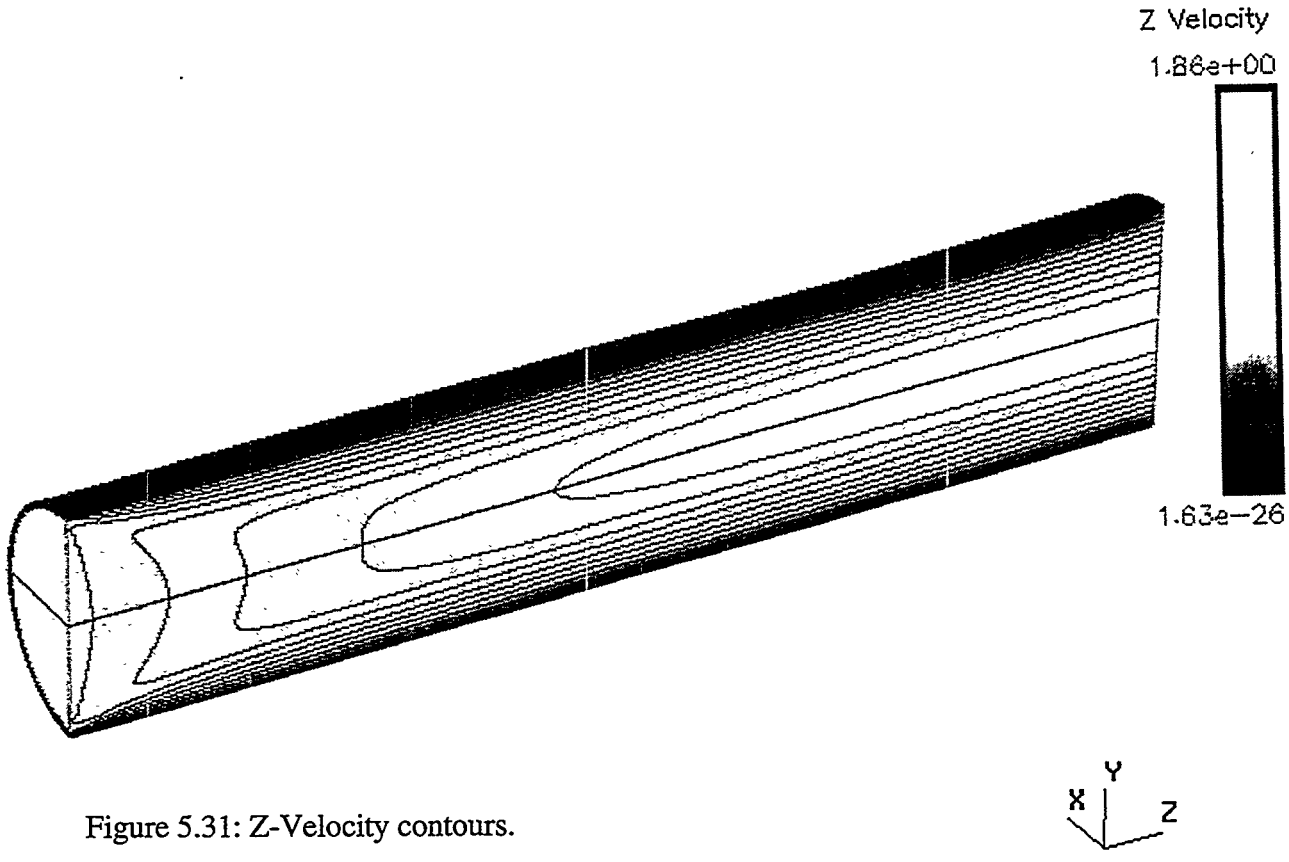
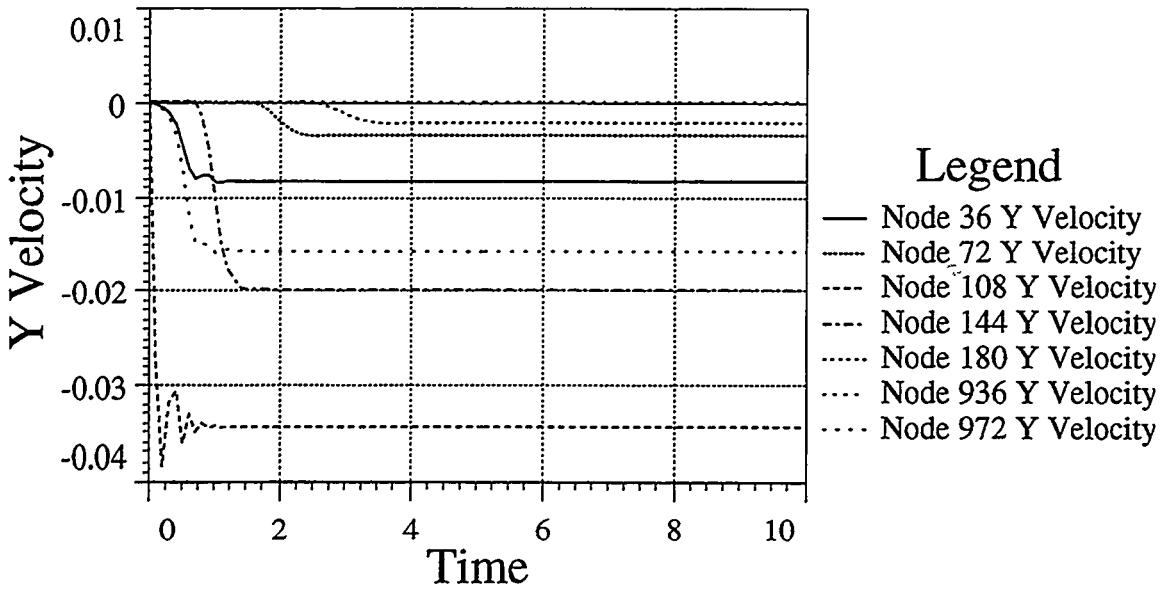
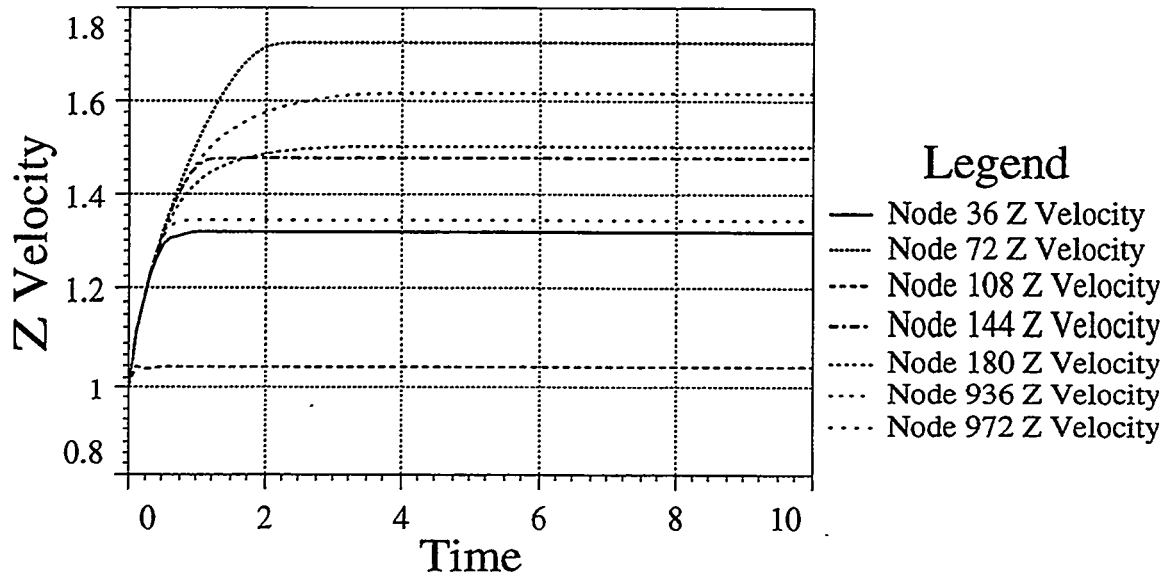


Figure 5.31: Z-Velocity contours.



a) y-velocity time history plot.



b) z-velocity time history plot.

Figure 5.32: Velocity time history plots.

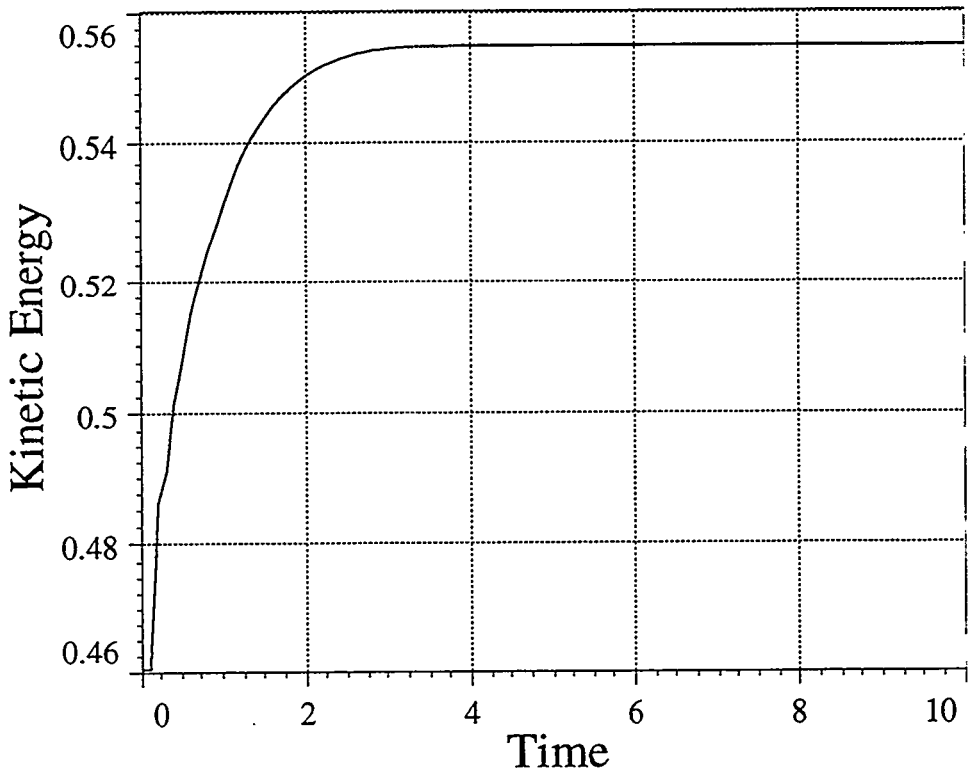


Figure 5.33: Kinetic energy time history plot.

5.6 Vortex Shedding

This example is the 3-D analogue to the 2-D cylinder vortex shedding calculation in section 5.3. All of the problem parameters are identical to the 2-D calculation. However, a 3-D calculation is performed by extruding the mesh out 1-element thick in the cross-flow direction as shown in Fig. 5.34. The control file for this problem is shown in Fig. 5.35.

Symmetry conditions are applied on the x-z planes of the domain. That is, $v=0$ on the symmetry planes, but u and w are allowed to vary. These boundary conditions result in a flow field identical to the that computed in section 5.3, but it is prismatic.

Snapshots for pressure isosurfaces are shown in Fig. 5.36, and the y-vorticity is shown in Fig. 5.37. The vorticity plot is essentially identical to the plot shown in Fig. 5.19, although it is only a single vorticity component in 3-D.

The time history nodes in the 3-D mesh were selected to correspond spatially to the time history points used in the 2-D vortex shedding example. As shown in Fig. 5.38, the velocity field goes through approximately 125 time units of a quasi-steady state in which there are two symmetric, standing eddies downstream of the cylinder. As expected, the velocity time histories and the kinetic energy time history (Fig. 5.39) are essentially identical to those for the 2-D vortex shedding calculation.

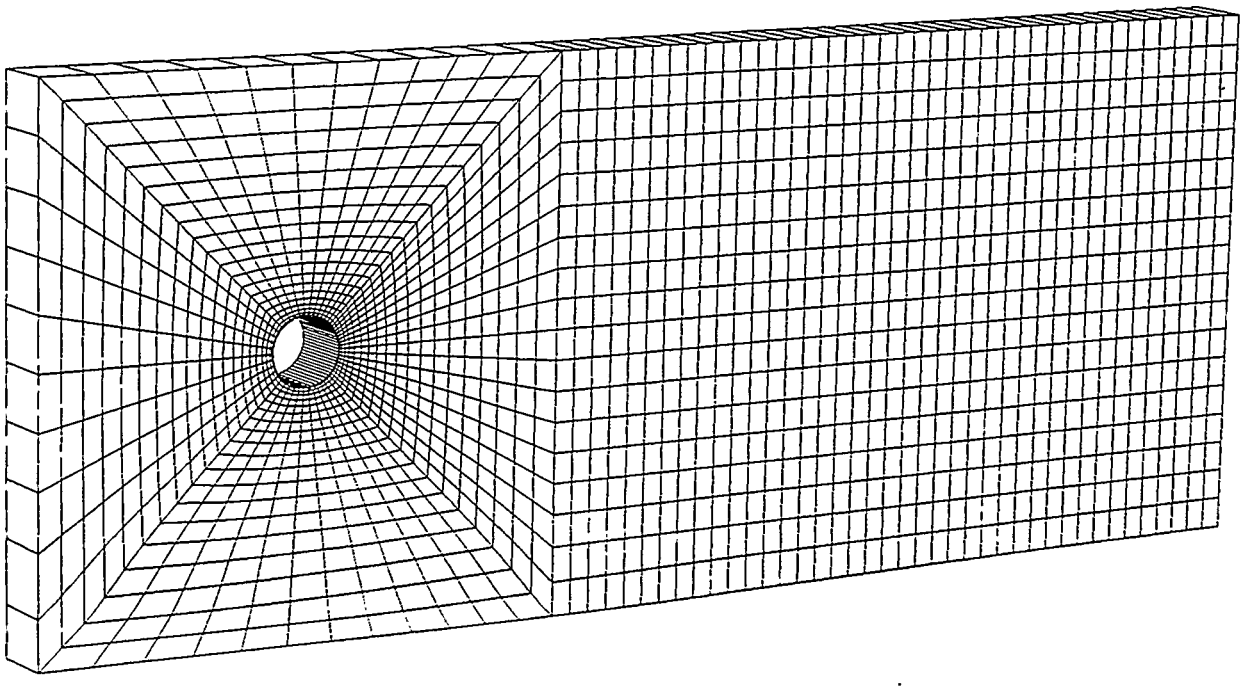


Figure 5.34: Mesh for 3-D vortex shedding.

```
Coarse mesh vortex shedding - Re=100

mesh
 nnp 3704
 nel 1760
 nnpe  8
 nmat  1
 ndim  3
end

analyze
 solve  3
 nubc  350
 nvbc 3704
 nwbc  350
 nstep 1000
 plti  50
 prti  500
 pltype  1
 icset  2220000 ;
 divu  1.0e-10
 pltype  1
 term  500.0
 deltat 0.25
 dtscal 1.0
 dtchk  -1
 rho  1.000000E+00
 nu  1.000000E-02
end

ppesol 41 end

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

ndhist 5
 nstep 1
 st 671 en 671
 st 678 en 678
 st 1135 en 1135
 st 1327 en 1327
 st 1760 en 1760
end
end
```

Figure 5.35: Control file for vortex shedding.

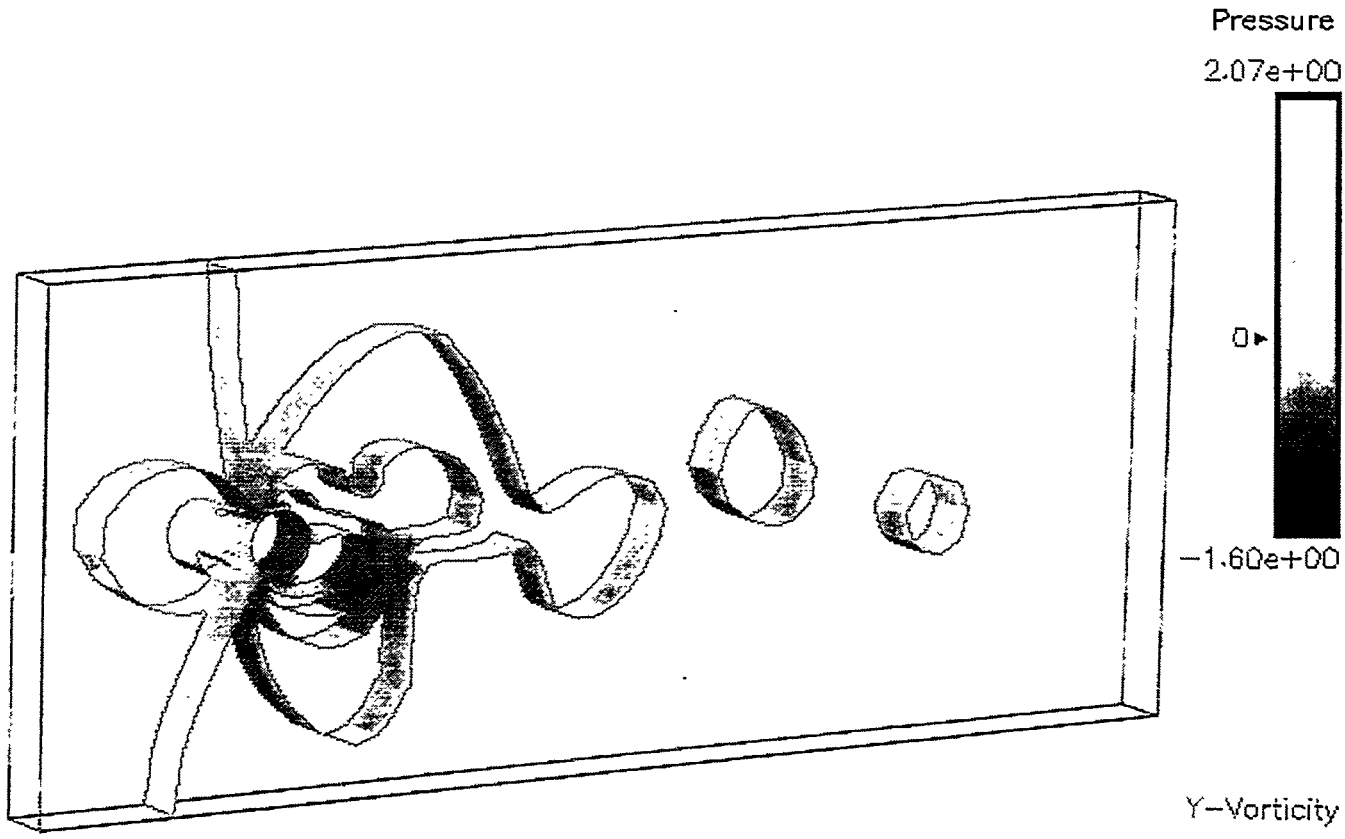


Figure 5.36: Pressure isosurfaces at $\tau = 250$.

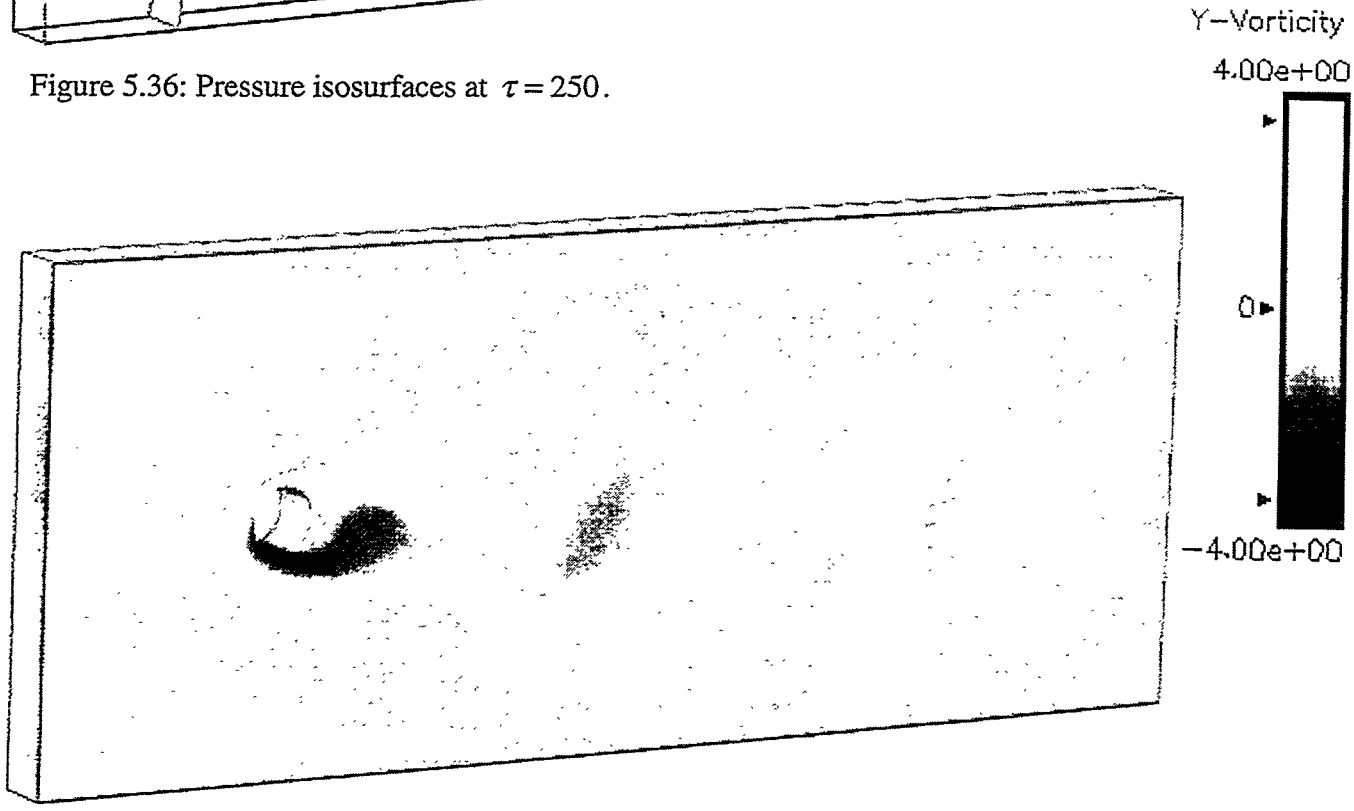
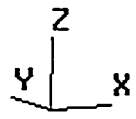
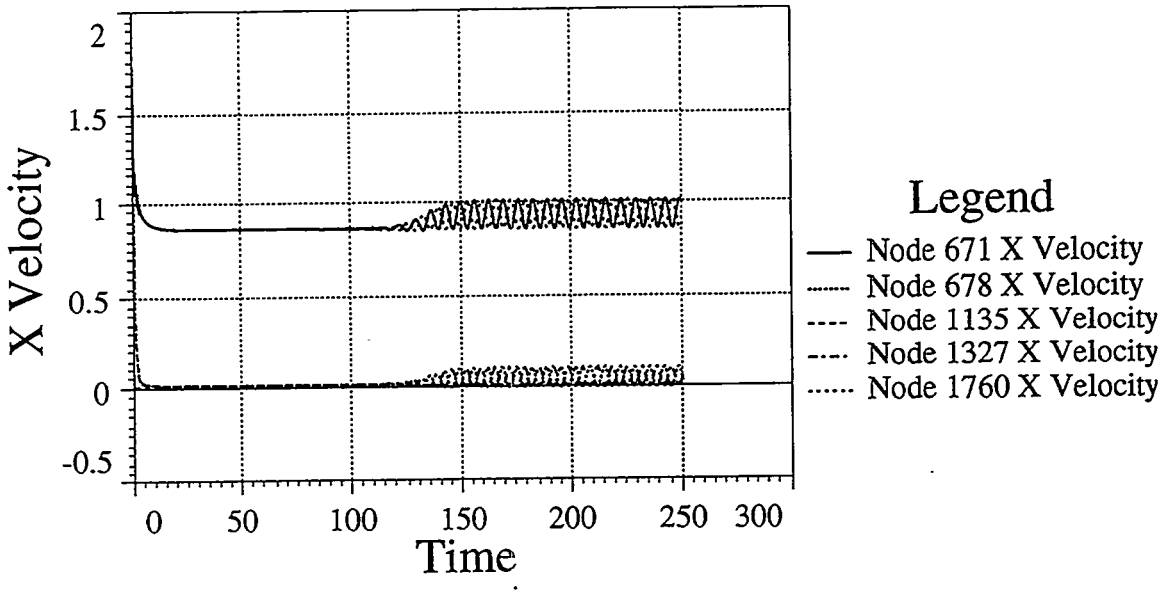
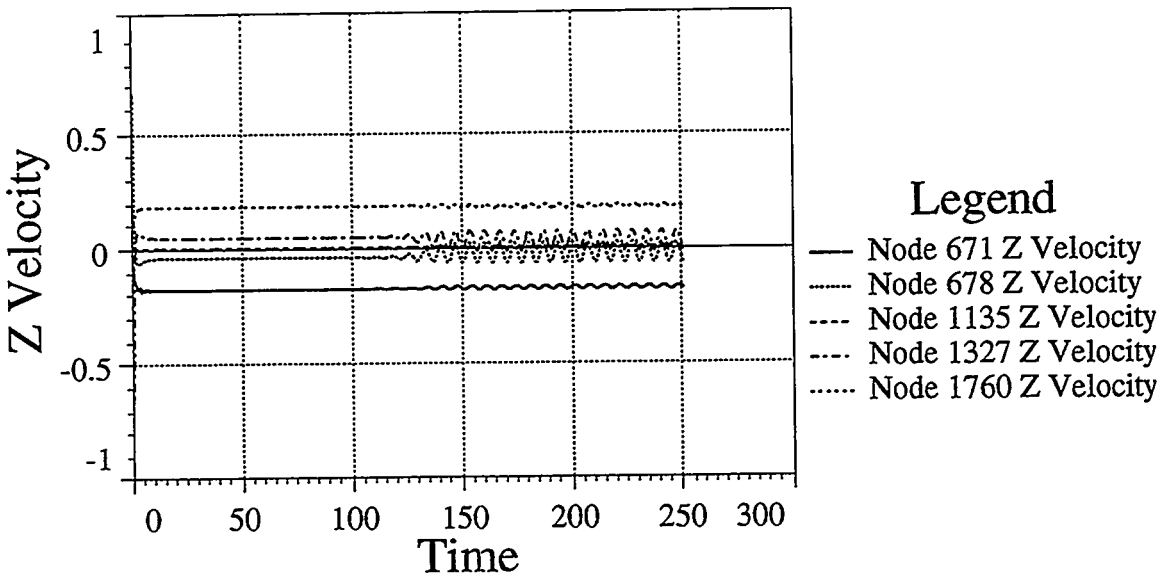


Figure 5.37: z-vorticity contours at $\tau = 250$.





a) x-velocity time history plots



b) z-velocity time history plots

Figure 5.38: Nodal velocity time histories.

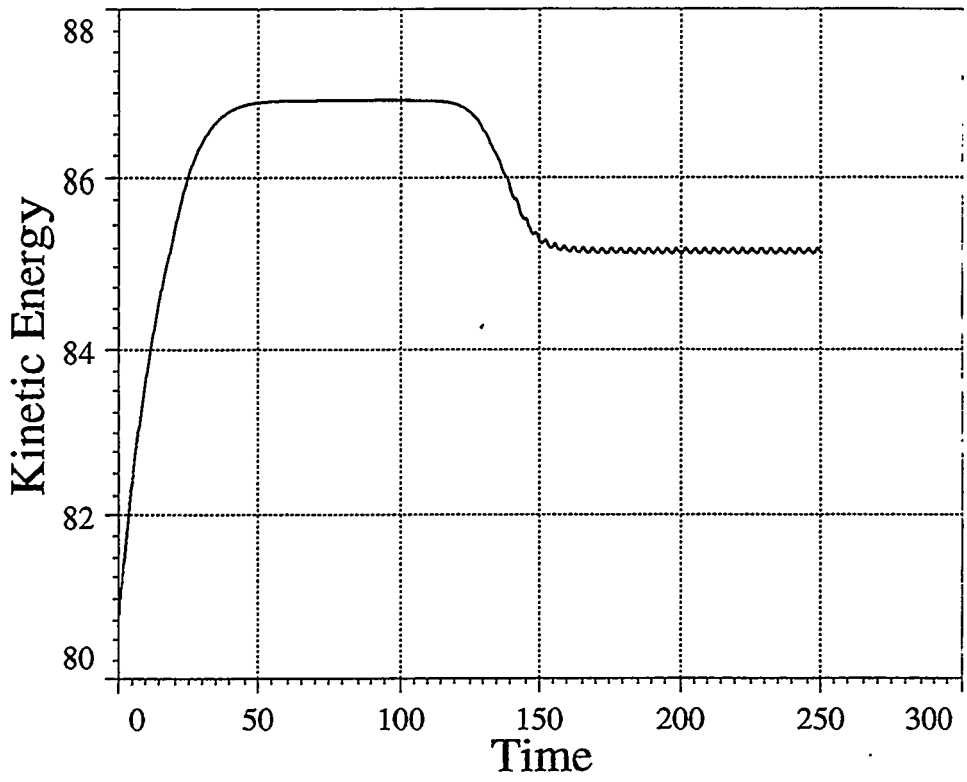


Figure 5.39: Kinetic energy time history.

5.7 Post & Plate

The final example problem consists of the flow past a flat plate with an attached cylindrical post. The mesh for this problem is shown in Fig. 5.40. The upstream conditions are a uniform velocity field ($u=1, v=0, w=0$), with no-slip boundary conditions applied at the plate and cylinder wall. The Reynolds number is 100 based upon the cylinder diameter. A symmetry plane is used at 2 diameters above the plate surface. Figure 5.41 shows the control file for this problem

In effect, this computation demonstrates several phenomena shown in earlier examples. As the simulation progresses, two symmetric, standing eddies appear just downstream of the post at the symmetry plane. After about 200 time units, vortex shedding begins which is very similar in character to the 2-D shedding problem.

The interaction of the plate boundary layer and the post results in longitudinal vortices being shed from the root of the post in the downstream direction. Figure 5.42 shows a snapshot of pressure isosurfaces during a vortex shedding cycle. Figure 5.43 shows isosurfaces of the z-vorticity also at 380 time units. The helicity ($\underline{u} \cdot \underline{\omega}$) is shown in Fig. 5.44.

Several velocity time history plots are shown in Fig. 5.45. The time history node was placed n the symmetry plane, and are directly downstream of the post. Figure 5.46 shows the kinetic energy time history for the calculation.

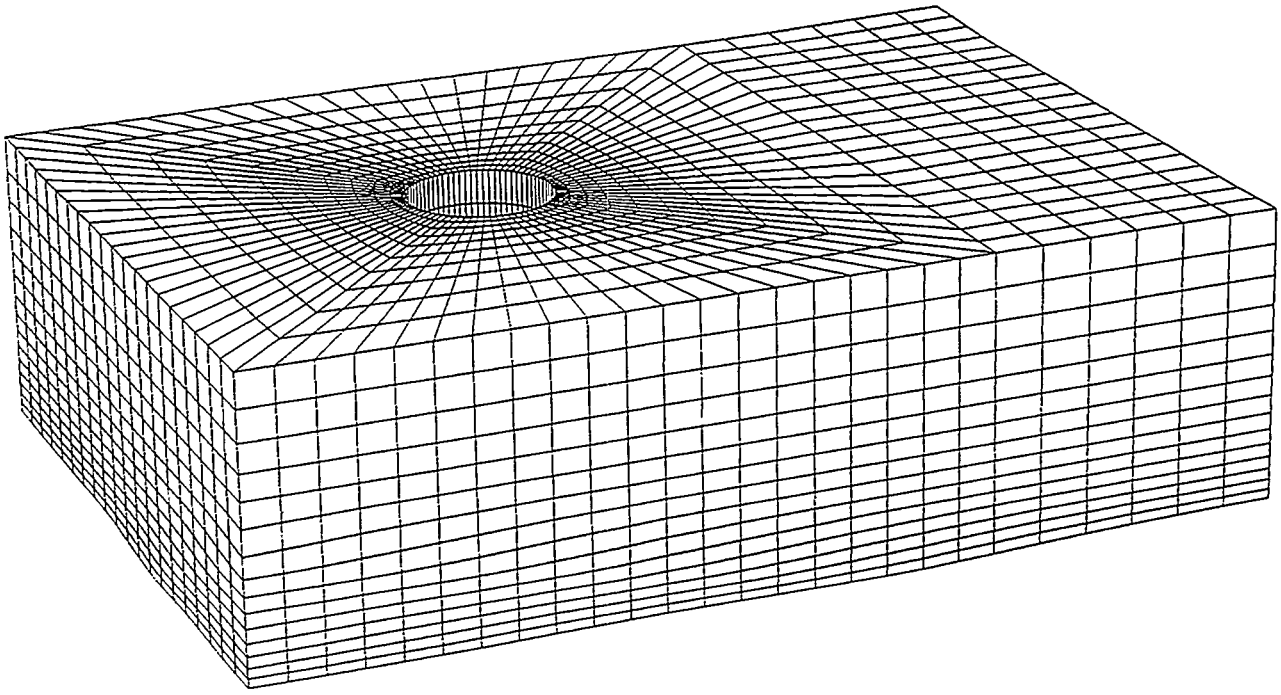


Figure 5.40: Post & Plate Mesh.

```
Post w. Re=100

mesh
 nnp 16096
 nel 13800
 nnpe  8
 nmat  1
 ndim  3
end

analyze
 solve  3
 npbc  0
 nubc 2747
 nvbc 3643
 nwbc 3814
 nstep  1
 plti  1
 prti  10
 pltype  1
 icset  2220000 ;
 divu  1.0e-10
 term  1.000000E+03
 deltat 0.500000E+00
 dtscal 1.0
 dtchk  -1
 rho  1.000000E+00
 nu  1.000000E-02
end

ppesol 41 end

ndhist 1
 st 1 en 10
 nstep 1
end

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

end
```

Figure 5.41: Post & plate control file.

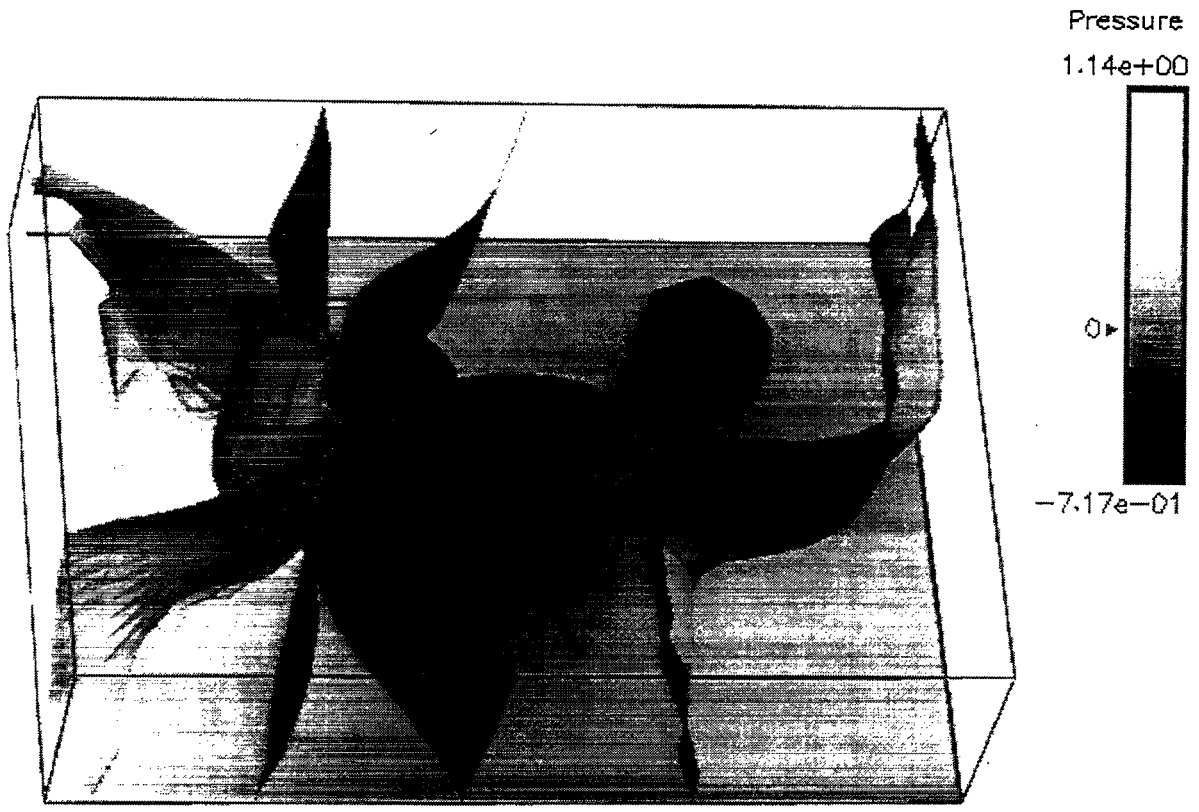


Figure 5.42: Pressure isosurfaces at $\tau = 380$ time units.

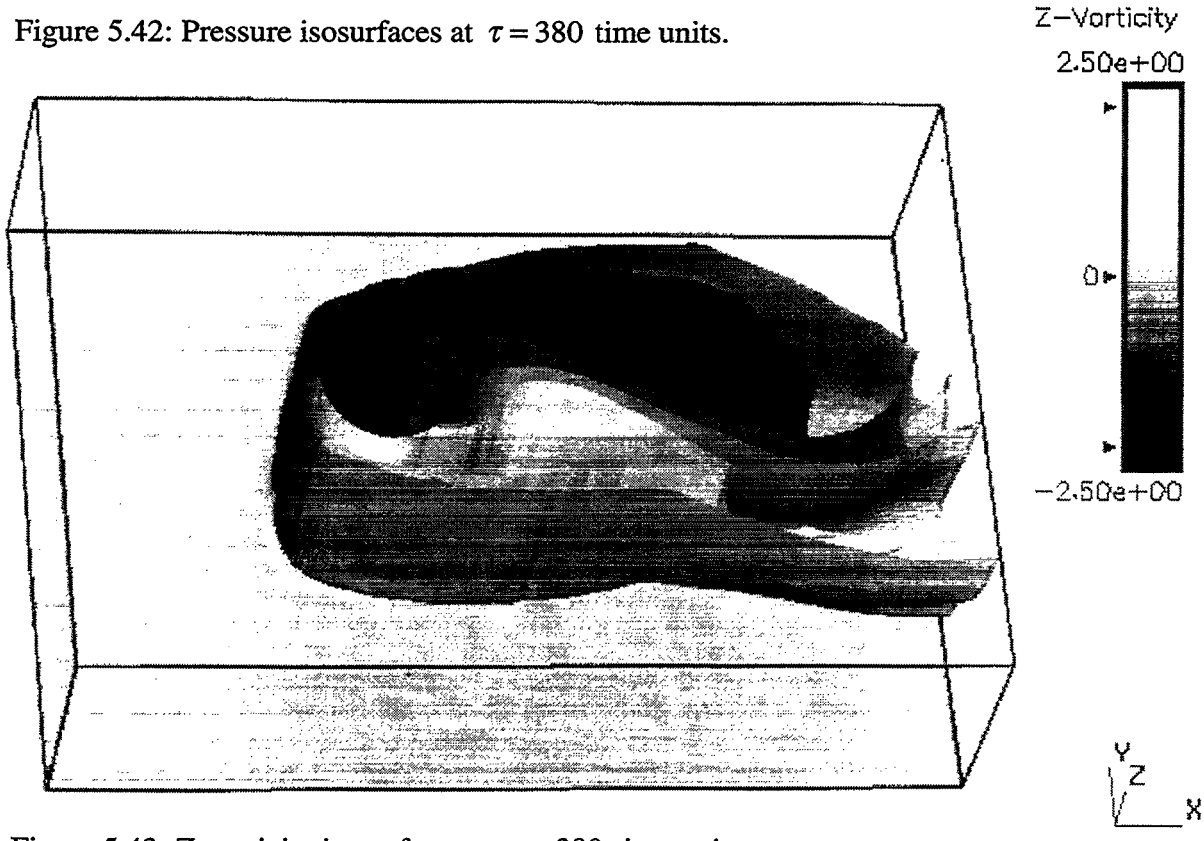


Figure 5.43: Z-vorticity isosurfaces at $\tau = 380$ time units.

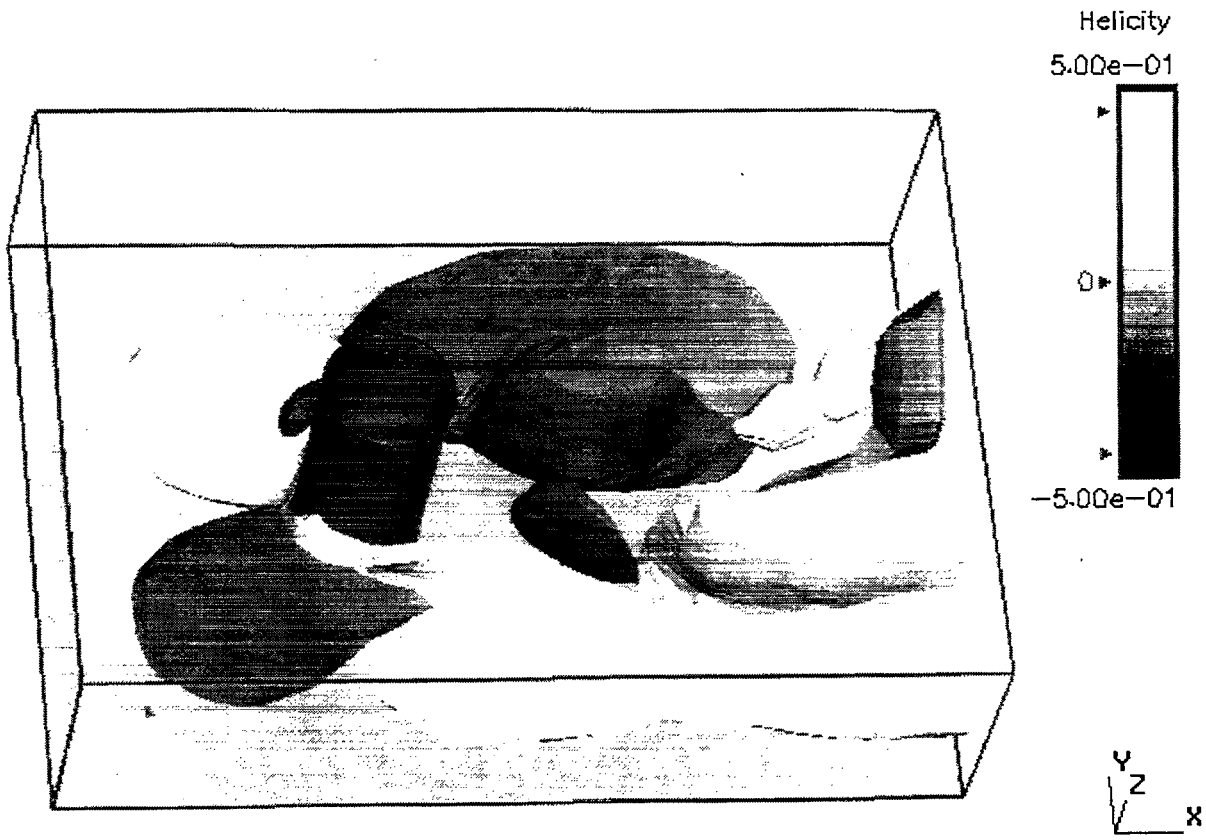
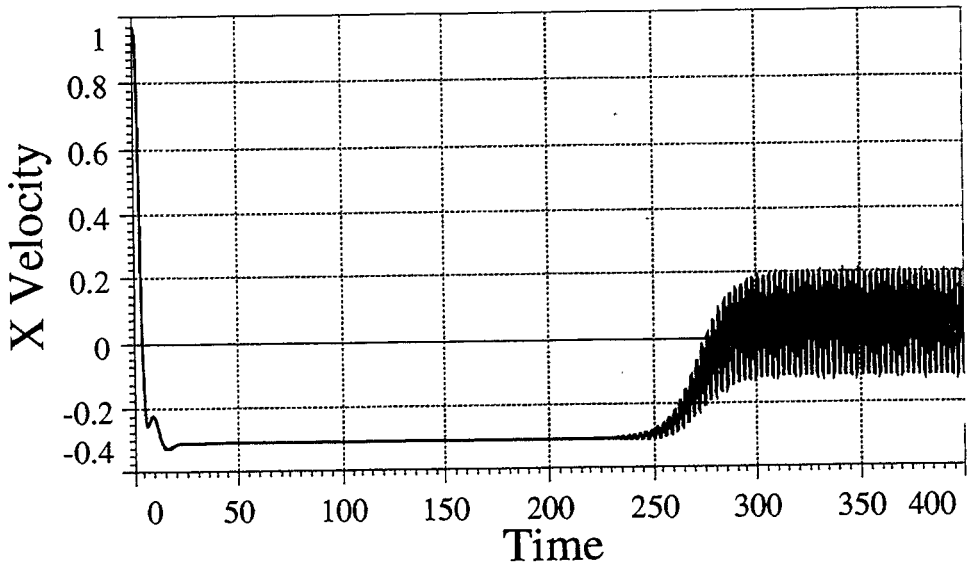
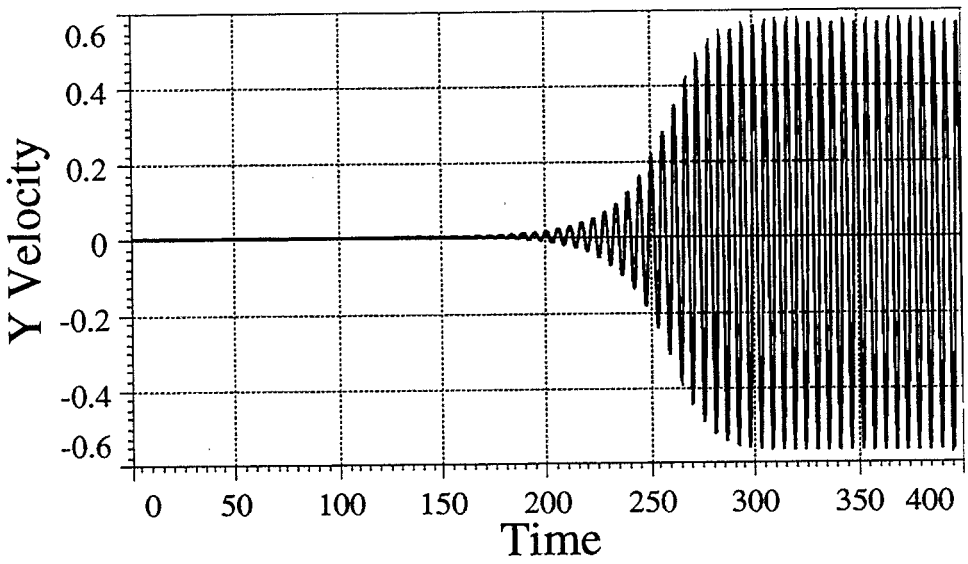


Figure 5.44: Helicity isosurfaces at $\tau = 380$ time units.

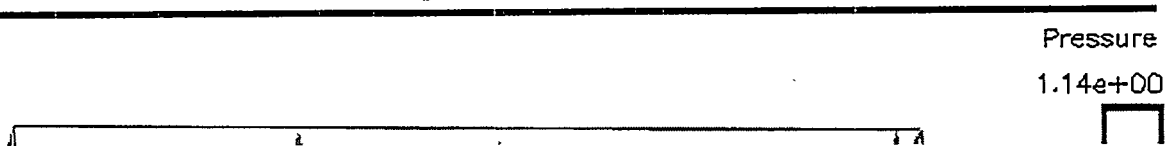


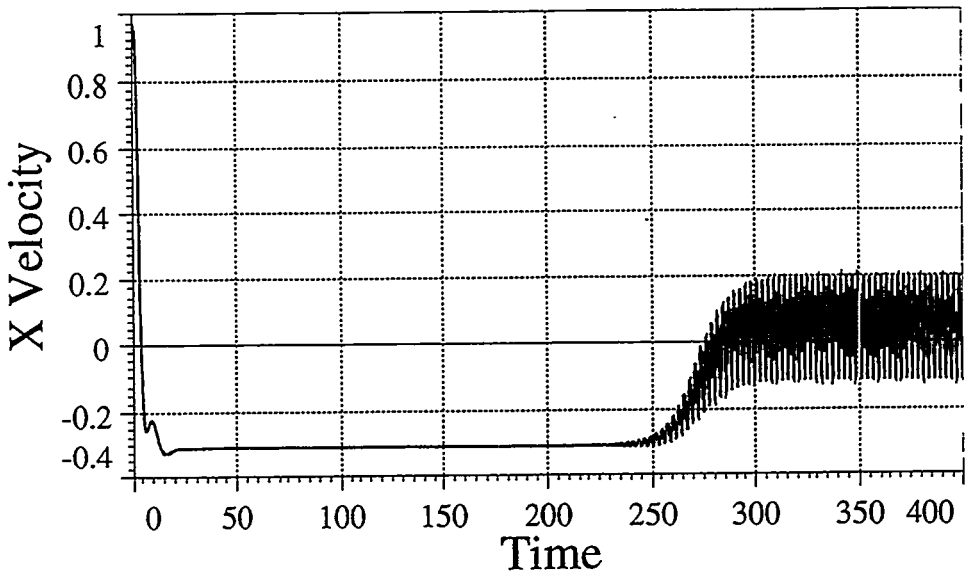
a) x-velocity time history plot.



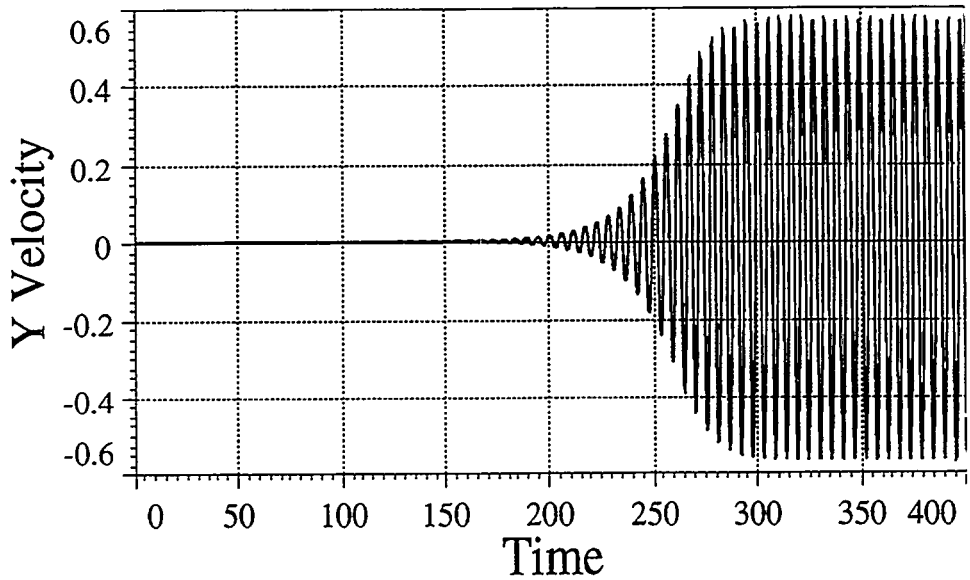
b) y-velocity time history plot

Figure 5.45: Velocity time history plots.





a) x-velocity time history plot.



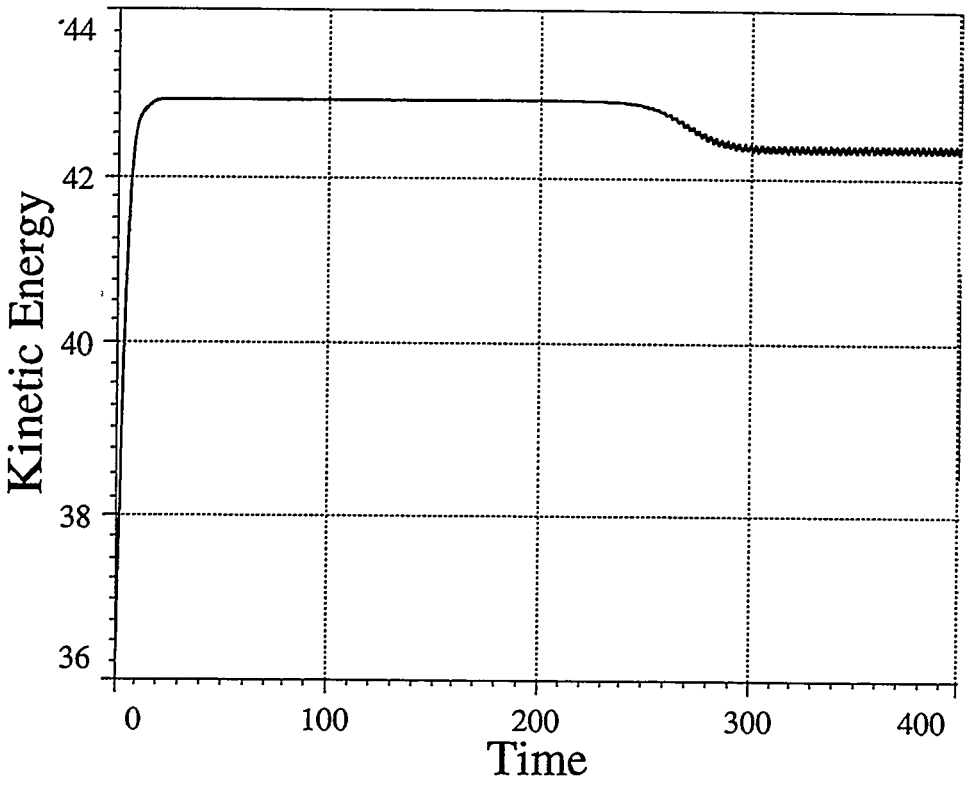


Figure 5.46: Post & plate kinetic energy time history.

Bibliography

1. R.G. Whirley, B.E. Engelmann, "DYNA3D: A Nonlinear, Explicit, Three Dimensional Finite Element Code for Solid and Structural Mechanics—User Manual," Lawrence Livermore National Laboratory, UCRL-MA-107254, Rev. 1, 1993.

2. B.N. Maker, R.M. Ferencz, J.O. Hallquist, "NIKE3D, A Nonlinear, Implicit, Three-Dimensional Finite element Code for Solid and Structural mechanics—User's Manual," Lawrence Livermore National Laboratory, UCRL-MA-105268, 1991.

3. *The Federal High Performance Computing Program*, Executive Office of the President, Office of Science and Technology, pp. 49–50, September 8, 1989.

4. 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," *Journal for Numerical Methods in Fluids* , Vol. **4**, pp. 557–598, 1984.

5. 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 2: Applications," *Int. Journal for Numerical Methods in Fluids*, Vol. **4**, pp. 619–640, 1984.

6. 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," *Int. Journal for Numerical Methods in Fluids*, Vol. **11**, pp. 587-620, 1990.

7. 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: Applications," *Int. Journal for Numerical Methods in Fluids*, Vol. **11**, pp. 587–620, 1990.

8. Smagorinsky, J., "General Circulation Experiments with the Primitive Equations," Mon. Weather Rev., Vol. 91, pp. 99–164, 1963.

9. Wilcox, D.W., Turbulence Modeling for CFD, DCW Industries, Inc., La Cañada, California, pp. 1–9, pp. 316–327, 1993.

10. K. Suga, T. J. Craft, and B. E. Launder, "Development of a k-ε-A_2 three equation model, notes from UMIST, January, 1995.

11. M. Christon, R. Whirley, "Explicit Structural Analysis in a Concurrent Computing Environment," *The 1991 MPCI Yearly Report*, Lawrence Livermore National Laboratory, UCRL-ID-10722-91, pp. 50–54, 1991.

12. C. Hoover, M. Christon, R. Whirley, J. DeGroot, "Explicit Nonlinear Structural Dynamics Models for Massively Parallel Computers," *The 1992 MPCI Yearly Report,* Lawrence Livermore National Laboratory, UCRL-ID-10722-92, pp. 83–88, 1992.

13. F. S. Lien and M. A. Leschziner, "Modelling the Flow in a Transition Duct with a Non-orthogonal FV Procedure and Low-Re Turbulence-Transport Models," *Advances in COmputational Methods in Fluid Dynamics,* ASME, Fed-vol. 196, 1994.

14. M.A. Christon, D. Dovey, and J.O. Hallquist, "INGRID A 3-D Mesh Generator for Modeling Nonlinear Systems," UCRL-MA-10970, September, 1992.

15. D.J. Dovey, and T.E. Spelce, "GRIZ Finite Element Analysis Results Visualization for Unstructured Grids," Draft manual—version 2, February, 1993.

16. D.E. Speck, "THUG: Time Histories from Unstructured Grids—User Manual," Lawrence Livermore National Laboratory, June, 1994.

17. P.M. Gresho and R. Sani, "On Pressure Boundary Conditions for the Incompressible Navier-Stokes Equations," *Int. J. Numer. Methods Fluids*, Vol. **7**, pg. 1111, 1987.

18. T.J.R. Hughes, *The Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 423–429, pp. 490–512, 1987.

19. O.C. Zienkiewicz, and R.L. Taylor, *The Finite Element Method*, Volume 2, Fourth edition, McGraw-Hill Book Company Limited, Berkshire, England, pp. 404–437, 1991.

20. T. Belytschko, J.S. Ong, W.K. Liu and J.M. Kennedy, "Hourglass Control in Linear and Nonlinear Problems," *Computational Methods in Applied Mechanics and Engineering*, Vol. **43**, pp. 251–276, 1984.

21. W.K. Liu and T. Belytschko, "Efficient Linear and Nonlinear Heat Conduction with a Quadrilateral Element," *International Journal for Numerical Methods in Engineering*, Vol. **20**, pp. 931–948, 1984.

22. W.K. Liu, J.S. Ong, and R.A. Uras, "Finite Element Stabilization Matrices," *Computer Methods in Applied Mechanics and Engineering*, Vol. **53**, pp. 13–46, 1985.

23. T. Belytschko and W.K. Liu, "On Reduced Matrix Inversion for Operator Spliiting Methods," *International Journal for Numerical Methods in Engineering*, Vol. **20**, pp. 385–390, 1984.

24. G.L. Goudreau, and J.O. Hallquist, "Recent Developments in Large-Scale Finite Element Lagrangian Hydrocode Technology," *Computational Methods in Applied Mechanics and Engineering*, Vol. **33**, pp. 725, 1982.

25. F.S. Beckman, "The Solution of Linear Equations by the Conjugate Gradient Method," *Mathematical Methods for Digital Computers*, Vol. **1**, pp. 62-72,1965.

26. W.L. Briggs, *Multigrid Tutorial*, Lancaster Press, Lancaster, PA, 1987.

27. J. W. Ruge and K. Stuben, "Algebraic Multigrid," *Multigrid Methods*, S. F. McCormick (Ed.), SIAM Series: Frontiers in Applied Mathematics, Philadelphia, Pennsylvania, 1987.

28. O.O. Storaasli, D.T. Nguyen, and T.K. Agarwal, "A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers," NASA Technical Memorandum 102614, April, 1990.

29. T.K. Agarwal, O.O. Storaasli, D.T. Nguyen, "A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers," AIAA 90-1149, April, 1990.

30. S. C. Eisenstat, "Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods," SIAM J. Sci Stat. Comput., Vol. 2, No. 1, pp. 1-4. March, 1981.

31. S. C. Eisenstat, J. M. Ortega. and C. T. Vaughan, "Efficieint Polynomial Preconditioning for the Conjugate Gradient Method, "SIAM J. Sci. Stat. Comput, VOl. 11, No. 5, pp. 859-872, Sept. 1990.

32. I. Fried, "A Gradient Computational Procedure for the Solution of Large Problems Arising from the Finite Element Discretization Method," *International Journal for Numerical Methods in Engineering*, Vol. **2**, pp. 477–494, 1970.

33. L.J. Hayes and P. Devloo, "A Vectorized Version of a Sparse Matrix-Vector Multiply," *International Journal for Numerical Methods in Engineering*, Vol. **23**, pp. 1043–1056, 1986.

34. R.M. Firencz, "Element-by-element Preconditioning Techniques for Large-Scale, Vectorized Finite Element Analysis in Nonlinear Solid and Structural Mechanics," Ph.D. Thesis, Stanford University, 1989.

35. D.R. Kincaid, T.C. Oppe, D.M. Young, *ITPACKV 2D User's Guide*, Center for Numerical Analysis, The University of Texas at Austin, May, 1989.

36. T.C. Oppe, W.D. Joubert, D.R. Kincaid, *NSPCG User's Guide, Version 1.0*, Center for Numerical Analysis, The University of Texas at Austin, April, 1988.

37. D.J. Tritton, *Physical Fluid Dynamics*, Second Edition, Oxford University Press, New York, 1988.

38. J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, Pennsylvania, 1991.

39. O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, New York, 1994.

40. S.F. McCormick, *Multilevel Projection Methods for Partial Differential Equations*, SIAM, Philadelphia, Pennsylvania, 1992.

41. R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, Pennsylvania, 1994.

42. O. Axelsson, "A Generalized SSOR Method," BIT, Vol 13, pp. 443-467, 1972.

43. Christon, M., *Some Useful Algorithms for Treating Unstructured Grids*, in preparation for *IJNME*, 1994.

44. Gresho, P.M., *Time Integration and Conjugate Gradient Methods for the Incompressible Navier-Stokes Equations*, LLNL, UCRL-9400, January 1986.

45. N.E. Gibbs, W.G. Poole, Jr., P.K. Stockmeyer, An Algorithm for Reducing the Bandwidth and Profile of a Spruce Matrix, SIAM J. Numer. Anul., Vol. 13, No. 2, pp. 236–250, 1976.

46. Storaasli, O.O., D.T. Nguyen, T.K. Agaruwal, "A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers," NASA Technical Memorandum 102614, April, 1990.