LA-UR-12-23181

Approved for public release; distribution is unlimited.

Title: Hydra-TH User's Manual, Version: LA-CC-11120, Dated: December 1, 2011

Author(s): Christon, Mark A.
Bakosi, Jozsef
Lowrie, Robert B.

Intended for: Report

Hydra-TH User's Manual
Version: LA-CC-11-120
Dated: December 1, 2011

Mark A. Christon, Jozsef Bakosi, and Robert B. Lowrie

Computational Physics and Methods, CCS-2
Los Alamos National Laboratory
Box 1663, MS D413
Los Alamos, NM 87545

July 17, 2012

Abstract

Hydra-TH is a hybrid finite-element/finite-volume code built using the Hydra toolkit specifically to attack a broad class of incompressible, viscous fluid dynamics problems prevalent in the thermal-hydraulics community. The purpose for this manual is provide sufficient information for an experience analyst to use Hydra-TH in an effective way. The Hydra-TH User's Manual present a brief overview of capabilities and visualization interfaces. The execution and restart models are described before turning to the detailed description of keyword input. Finally, a series of example problems are presented with sufficient data to permit the user to verify the local installation of Hydra-TH, and to permit a convenient starting point for more detailed and complex analyses.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Hydra-TH is a hybrid finite-element/finite-volume incompressible/low-Mach flow solver built using the Hydra toolkit. Hydra-TH is one of a number of virtual physics using the Hydra multiphysics toolkit. The Hydra toolkit is written in C++ and provides a rich suite of components that permits rapid application development, supports multiple discretization techniques, provides I/O interfaces to permit reading/writing multiple file formats for meshes, plot data, time-history, surface-based and restart output. Data registration is used to provide the ability to register variables at appropriate locations (e.g., node, element, dual-edge, etc), and provides integrated and automatic output and restart capabilities along with memory management. The toolkit also provides run-time parallel domain decomposition with data-migration for both static and dynamic load-balancing. Linear algebra is handled through an abstract virtual interface that makes it possible to use popular libraries such as PETSc and Trilinos. The use of output delegates provides the ability to develop lightweight physics-specific output kernels with minimal memory overhead that can be tailored to a specific physics, e.g., computation of vorticity, helicity, enstrophy for large-eddy simulations.

The Hydra-TH theory manual [8] presents the theoretical background for the hybrid finite-element/finite-volume incompressible/low-Mach flow solver based on the Hydra toolkit. By design, Hydra-TH was built with the idea of handling both single and multi-component flows. Although not currently used in Hydra-TH, the Hydra toolkit provides a number of interfaces for using interface reconstruction for volume-tracking, front-tracking (via FronTier [15]), and it is anticipated that these will be used in the future for CASL applications.

Hydra-TH uses a hybrid finite-element/finite-volume discretization for the incompressible/low-Mach Navier-Stokes equations. All transport variables are cell-centered and treated with a conservative discretization that includes a high-resolution monotonicity-preserving advection algorithm. The spatial discretization is formally derived using a discontinous-Galerkin framework that, in the limit, reduces to a locally-conservative finite-volume method. The high-resolution advection algorithm is designed to permit both implicit and explicit advection with the explicit advection targeted primarily at volume-tracking with interface reconstruction. The time-integration methods include backward-Euler and the neutrally-dissipative trapezoidal method. The coding for an optional BDF2 time-integrator has also been provided, but is not currently used in Hydra-TH. The implicit advective treatment delivers unconditional stability for the scalar transport equations, and conditional stability for the momentum transport equations. A sharp stability estimate for the momentum equations is not tractable, but operational experience shows that the algorithm is stable for $20 \leq CFL \leq 40$. For steady-state problems, backward-Euler provides additional damping that, in conjunction with $20 \leq CFL \leq 40$, provides a computationally efficient solution method. For URANS and LES computations, the trapezoid rule is neutrally dissipative, and delivers optimal performance for the more moderate CFL requirements for transient flow.

The solution algorithm used in Hydra-TH is based on a second-order incremental projection algorithm. Projection methods are the most computationally efficient solution method available for solving the time-dependent Navier-Stokes equations. Over the past 20+ years, projection methods have enjoyed widespread adoption and have been applied to complex flow problems ranging from mold filling (volume-tracking) to atmospheric dispersion, chemically reacting flows, and exterior aerodynamics (see for example [3, 4, 10, 24, 17, 18, 19, 20, 21, 23, 16, 6, 9, 7, 5, 2, 22, 25, 26, 27]). The projection method also permits treating the momentum equations in a coupled manner (see for example [25, 26]. Although not currently used, Hydra-TH provides the underlying coding to couple arbitrary degrees-of-freedom in multiple transport equations. In addition, Hydra-TH has been implemented to permit rapid conversion to SIMPLE-based solution methods for steady-state problems. Extension to fully-coupled Newton-Krylov solution strategies can also easily be incorporated using inheritance with the virtual physics hierarchy.

In order to address fluid-structure problems, Hydra-TH uses an arbitrary Lagrangian-Eulerian (ALE) formulation and provides a mesh-deformation interface that can support multiple different mesh smoothing algorithms. Details on the ALE formulation may be found in the Hydra-TH theory manual [8]. The added-mass terms are computed for the structural coupling and can be exported for any structural solver. For explicit coupling, Hydra-TH provides a pressure-stabilized algorithm based on Nitche's varational method that circumvents the stability limitations associated with highly flexible structures and near unity fluid/solid density ratios. For conjugate heat transfer, there are multiple alternatives available in Hydra-TH that include explicit coupling with third-party heat conduction solver, internal coupling using the existing heat conduction solver supported by the Hydra toolkit and the multiphysics manager, or direct integration (with continuous meshing). The calculation of exported fields for both fluid-structure interaction and conjugate heat transfer are implemented for explicit coupling methods, and can be easily extended for use in tightly-coupled solution strategy. It is anticipated that driving CASL applications will determine the most suitable FSI/CHT solution strategy.

The linear algebra interface in Hydra-TH provides a number of linear algebra options that include both native solvers for testing/evaluation, and a rich set of solvers provided by PETSc. These solvers include the conjugate-gradient method (CG), bi-conjugate gradient squared (BCGS), generalized minimum residual (GMRES) and it's flexible counter part (FGMRES). For the pressure equation, the Trilinos ML preconditioner is used with CG while the transport equations typically use either Jacobi or ILU(0) preconditioning and FGMRES/GMRES.

The Hydra-TH flow solver was developed to make use of hybrid meshes and uses hex, tet, pyramid and wedge elements to permit meshing extremely complex geometries. All boundary and initial conditions are implemented using node, side (surface), and element sets permitting flexibility in the development of complex models. Material sets provide a simple and natural way to prescribe initial material interfaces for multi-fluid problems, and for prescribing material properties in conjugate heat transfer problems.

Hydra-TH builds on the Hydra toolkit to provide a number of non-Newtonian viscosities as well as the ability to handle temperature-dependent properites (see the Hydra-TH theory manual [8]). In addition, the use of output delegates permits Hydra-TH to provide a rich suite of output variables that are automatically tied to user input. Field output data may be requested as either element-centered or node-centered. In addition to the solution variables (e.g., velocity, temperature, turbulent kinetic energy, dissipation rate, etc.), Hydra-TH currently provides the following field output: displacements (ALE calculations only), vorticity, helicity, enstropy, the second-invariant of velocity gradient, i.e, the 'Q' criteria, processor ID (MPI rank), and turbulent eddy viscosity. Because output delegates are registered at run-time, it is trivial to add additional field output for visualization and debugging purposes. Surface field output variables include total traction, shear traction, normal traction, wall shear force, $y+$ and $y*$, heat flux, and the normal heat flux. Time history output variables include the primitive variables and turbulent eddy viscosity, enstrophy, vorticity, helicity, average pressure, surface area, average velocity, force, pressure forces, viscous forces, mass flow, volume Flow, heat flow, and average temperature.

1.1 Guide to the Hydra-TH User's Manual

The purpose for this document is to provide sufficient information for an experienced analyst to use Hydra-TH in an effective way. The assumption is that the user is somewhat familiar with modern supercomputers, large scale computing, common practices in computational solid mechanics and fluid dynamics. This manual provides sufficient references to the literature to permit the interested reader to pursue the technical details of Hydra-TH.

In this document, all keywords and defaults for input data appear in a **boldface** type, and sample computer input/output appears in a `typewriter` font. All other keywords, parameters and variables are defined in the context they are used. All keywords are optional unless otherwise specified.

Chapter 2 explains how to run Hydra-TH using command line arguments, and how to perform restarts. Chapter 3 presents the general analysis keywords, and chapter 4 presents the physics-specific keywords for Hydra-TH. A series of sample problems are presented in chapter 5.

1.2 Hydra-TH Capabilities

The Hydra toolkit that Hydra-TH is built on was parallel design and scalable across platforms and applications. The underlying infrastructure design was based on scaling to mesh sizes with greater than 10^9 elements. The C++ design is interface centric and component based. The underlying concept for the Hydra toolkit is to permit the discretization and solution methods that are optimal for the specific application space. For Hydra-TH, the discretization makes use of edge-based finite-volume techniques that provide local conservation and monotonicity-preserving advection methods, while permitting the use of hybrid meshes that include tet, hex, pyramid, wedge and polyhedral elements.

Hydra-TH makes provides run-time parallel load balancing with dynamic data migration. The I/O interfaces are provided with "plug 'n play' multi-reader/multi-writer options for meshes and output. Physics-centric output delegates provide a rich set of output variables for state, derived, statistical and time-history data. A general virtual linear algebra interface provides access to a broad suite of Krylov solvers and advance preconditioners. Error handling is provided for both exceptional and cumulative errors. This permits the ability to distinguish between "soft" errors that need not terminate a calculation, and those that require immediate termination of the code.

In addition, some of the features that Hydra-TH provides include the following

- Energy equation may be solved in temperature or enthalpy form
- Multiple turbulence models that include implicit large-eddy simulation (ILES), detached-eddy simulation (DES) Spalart-Allmaras, RNG $k - \epsilon$. In addition, prototype versions of the $k - \omega$ and k-sgs are under development.
- Porous media flow
- Time-dependent boundary and source terms
- Generalized body forces
- Treatment of hybrid meshes with tet, hex, wedge pyramid elements
- Monotonicity-preserving implicit advection
- Automatic time-step control
- Eulerian or ALE with deforming mesh
- Passive outflow boundary conditions
- Coupling interface for use with third-party codes for conjugate heat transfer and fluid-solid interaction

1.2.1 Visualization Interfaces

Hydra-TH can output several types of files for visualization that include both field data and time-history data at varying time-intervals. By default, the Exodus-II file format is used for field data and time-history data is output an ASCII format. In addition, global data, e.g., kinetic energy, is output in an ASCII format.

The file formats for field data are compatible with a number of popular visualization tools that include EnSight, ParaView, and VisIT. (Note: At this time, VisIT does not support the use of hybrid meshes in the Exodus-II file format.) The field formats for time-history data may be used with a number of tools ranging from GNUplot to MatLab. Additional details on the specific output options for Hydra-TH may be found in chapter 4.

Chapter 2

Running Hydra-TH

Hydra-TH has been developed to permit rapid configuration making it adaptable to many computer architectures. Hydra-TH has been exercised on computers ranging from laptops to leadership class massively parallel supercomputers. Hydra-TH provides a single command line interface for that is the same regardless of the computer platform. The command line functions in the fashion in which most common UNIX commands operate, i.e, a single command followed by a list of command line arguments. The command line is invariant with respect to the physics or multiphysics being solved with Hydra-TH.

2.1 Execution

Hydra-TH may be executed with the following command line options:

`hydra -i mesh -c cntl -o [out] -p [plot] -g [glob] -d [dump] -r restart`

Hydra-TH Command Line Arguments

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

All 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 /scratch/plate/flow1.msh -c ../cntl1 -o /home/joe/plate.out
```

2.2 Restarts

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

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

Note that the **dump** keyword must be used in the **physics** − **end** keyword block to activate restarts (see Chapter 3).

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

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

Chapter 3

General Analysis Keywords

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

In this document, **bold** text denotes keywords, while *italic* text identifies keyword parameters or optional data in the control file. Primary sections of the control file are delimited by a **keyword** – **end** sequence that may contain a series of **keyword** – *parameter* sequences. The presence of a keyword and a parameter implies that the parameter is expected as input. Where possible, default values have been identified in order to minimize the number of keywords that are necessary in an input file. Comments in the control file must be preceded by a "$", "#" or "∗" symbol, or may be enclosed in a pair of braces "{ }". All input in the control file is case insensitive. Every attempt has been made to eliminate order-dependence in the control file. Figure 3.1 shows the typical format of a Hydra control file.

Keyword input in Hydra provides the ability to use aliases for keywords. In the ensuing discussion of keyword input, the primary keyword syntax is presented. Where aliases for a keyword are available, they are listed after the primary keyword syntax. Default values, where appropriate, are indicated by *default*. All keywords are considered to be optional unless otherwise specified.

This section outlines the general analysis keywords that apply to all Hydra physics. These keywords control the termination conditions, output frequency, restarts, load curves, and parallel load-balancing.

3.1 title

title
80-character analysis title

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

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

# starts a comment line
* starts a comment line
$ starts a comment line

{ Comments may be enclosed in braces as well }

# The physics-end block describes the analysis parameters
cc_navierstokes
  ...

  # The material - end block defines material properties
  # The material -- end block is required input
  material
    ...
  end

  # The turbulence keyword activates a turbulence model
  turbulence spalart_allmaras

  # The momsolsolver - end block defines the momentum equation solver
  momentumsolver
    ...
  end

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

  ...

  end # Ends the physics - end block

end # End keyword input
```

Figure 3.1: A Sample Hydra Control File.

appears in the input file is shown below and in Figure 3.1.

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

3.2 load_balance

Hydra provides a number of parallel run-time load-balancing options. The **load_balance** block specifies the load balancing method to be used and the level of diagnostic output.

Syntax:

> **load_balance**
>   **method** *type*
>   **verbose** *level*
> **end**

Aliases: loadbalance

Parameter Description:

method *type* (string, default=`sfc_and_hg`) Specifies the type load balancing. The value must be one of the following:

- `rcb` Use recursive coordinate bisection for domain decomposition.
- `rib` Use recursive inertial bisection for domain decomposition.
- `sfc` Use a space-filling curve for domain decomposition.
- `hg` Use hypergraph domain decomposition.
- `sfc_and_hg` Use a space-filling curve for the initial domain decomposition, followed by a hypergraph partitioning algorithm.

verbose *level* (string, default=`silent`) Specify the verbosity of load balance output. Valid values are the following:

- `silent` Minimal output from load balancer.

3.3 load_curve

The input of load-curve data allows for the specification of time-dependent boundary conditions.

Syntax:

> **load_curve**
>   **Id** *LoadCurveId*
> t_1 v_1
> t_2 v_2
> t_3 v_3
> $\ldots$
> t_{Npts} v_{Npts}
> **end**

Aliases: lcurve

Parameter Description:

Id *LoadCurveId* (integer, required) Specifies an integer identifier for this load curve. The load curve is defined as a list of time and factor pairs:

t_i The time value

v_i The load factor

3.4 Material Models (Required)

This section describes the required input for material model definition and material set definition. A material model contains the necessary material parameters to describe material behavior for a range of physics. Material sets describe what materials appear in various regions of the physical domain, i.e., what parts of the computational mesh are contained in a material set.

For the initial delivery of Hydra-TH for use in VERA-CFD, a simplified material model interface has been provided. Subsequent releases will provide a more comprehensive material model interface. In the discussion that follows, only those material parameters required for a specific physics need to be included in the **material** – **end** keyword block.

3.4.1 material

Syntax:

material
  id *matId*
  rho ρ_{ref}
  Cp C_p
  Cv C_v
  k11 k_{11}
  k12 k_{12}
  k13 k_{13}
  k22 k_{22}
  k23 k_{23}
  k33 k_{33}
  mu μ
  gamma γ
  beta β
  Tref T_{ref}
end

Aliases: *NONE*

Parameter Description:

id *matId* (integer, required) Specifies the integer Id for the material.

rho ρ_{ref} (float, default=`1.0`) Specify the material mass density ρ_{ref}.

Cp C_p (float, default=`1.0`) Specify the constant-pressure specific heat C_p.
Cv C_v (float, default=`1.0`) Specify the constant-volume specific heat C_v.
k11 k_{11} (float, default=`1.0`) Specify k_{11} in the thermal conductivity tensor. For fluid dynamics problems, this is the only thermal conductivity that is required.
k12 k_{12} (float, default=`0.0`) Specify k_{12} in the thermal conductivity tensor.
k13 k_{13} (float, default=`0.0`) Specify k_{13} in the thermal conductivity tensor.
k22 k_{22} (float, default=`1.0`) Specify k_{22} in the thermal conductivity tensor.
k23 k_{23} (float, default=`0.0`) Specify k_{23} in the thermal conductivity tensor.
k33 k_{33} (float, default=`1.0`) Specify k_{33} in the thermal conductivity tensor.
mu μ (float, default=`1.0`) Specify the molecular viscosity.
Tref T_{ref} (float, default=`0.0`) Specify the material reference temperature.
beta β (float, default=`0.0`) Specify the material coefficient of thermal expansion.

3.4.2 materialset

The **materialset** – **end** keyword block relates material definitions to the element sets (or element blocks). For fluid dynamics. Multiple material sets may be used to define regions where a common single material exists in the model.

Syntax:

> **materialset**
>   **id** *setId*
>   **block** *blockId*
>   **material** *matId*
>   **type** *setType*
> **end**

Aliases: matset

Parameter Description:

id *setId* (integer, required) Specifies the integer material set Id.
block *blockId* (integer, required) Specify an element set or block Id for this material set. This keyword may be repeated as many times as required to describe all element sets contained in a material set.
type *setType* (string, default=`Eulerian`) Specify the type of material set. *setType* must be one of the following
- `Eulerian` Use Eulerian coordinates for this material set.
- `Lagrangian` Use Lagrangian coordinates for this material set.
- `ALE` Use an arbitrary Lagrangian-Eulerian description for this material set.

3.5 Output Keywords

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

3.5.1 dump

Activates checkpoint-restart files, where a restart file will be written every N_{dump} time steps. In addition, a restart file will be written at the termination of the calculation.

A restart file may be read on a subsequent invocation of HYDRA with the "**-r**" command-line option.

Syntax:

dump N_{dump}

Aliases: *NONE*

Parameter Description:

N_{dump} (integer, default=0) Output dump files every N_{dump} time steps.

3.5.2 filetype

For parallel computations, the **filetype** keyword selects the use of either distributed or serial plot files.

Syntax:

filetype *type*

Aliases: *NONE*

Parameter Description:

type (string, default=`serial`) Specifies the type of plot files. Valid values are the following:

`serial` This file type is the default for serial calculations. For parallel calculations, this file type requires the serialization of data using a parallel fan-in procedure, but results in a single plot file.

`distributed` This file type results in one plot file per processor and is more scalable for problems where a large number of processors are used.

3.5.3 histvar

The **histvar** block specifies time history output for element and node-centered field data. In addition, integral data associated with side-sets may be requested. Each physics provides the set of node, element and side-set variables that can be requested for time-history output.

Syntax:

histvar
  elem Id_1 var_1 # Element time-history output
  elem Id_2 var_2
  ...
  node Id_1 var_1 # Node time-history output

node Id_2 var_2
...
side Id_1 var_1 # Side-set time-history output
side Id_2 var_2
...
end

Aliases: *NONE*

Parameter Description:

elem Id (integer, required) Element index
var (string, required) Variable name
node Id (integer, required) Node index
var (string, required) Variable name
side Id (integer, required) Side-set index
var (string, required) Variable name

3.5.4 plotvar

The **plotvar** block specifies the plot output variables for element and node-centered field data. In addition, face-centered data associated with side-sets may be requested. Each physics provides the set of node, element and side-set variables that can be requested for plot output.

Syntax:

plotvar
elem var_1 # Element field plot output
elem var_2
...
node var_1 # Node field plot output
node var_2
...
side Id_1 var_1 # Side-set field output
side Id_2 var_2
...
end

Aliases: *NONE*

Parameter Description:

elem var (string, required) Variable name
node var (string, required) Variable name
side Id (integer, required) Side-set index
var (string, required) Variable name

3.5.5 pltype

The **pltype** keyword selects the file format used for field output which includes both instantaneous state variables, and time-averaged statistical quantities. At this time, surface field variables associated with model side-sets is only available in the `exodusii` and `exodusii_hdf5` formats.

Syntax:

pltype *type*

Aliases: *NONE*

Parameter Description:

type (string, default=`exodusii`) Specifies the plot file format. Valid values are the following:

`gmv_ascii`	generates ASCII GMV files suitable for use with the GMV visualization tool.
`exodusii`	generates Exodus-II (CDF) format plot files. Note that the Exodus-II (CFD) file format is limited to mesh sizes below approximately 10^8 elements.
`exodusii_hdf5`	generates Exodus-II (HDF5) plot files. This format allows for larger mesh sizes than the `exodusii` format.
`vtk_ascii`	generates ASCII VTK format plot files.

3.5.6 plti

Sets the output interval for plot files.

Syntax:

plti N_{plot}

Aliases: *NONE*

Parameter Description:

N_{plot} (integer, default=20) Output plot files every N_{plot} time steps.

3.5.7 prtlev

Controls the amount of data written to the ASCII (human-readable) output file.

Syntax:

prtlev *level*

Aliases: *NONE*

Parameter Description:

level (string, default=`param`) The amount of data to output. Valid values are the following:

`param`	suppresses all output except for a data echo of the primary code options.
`results`	produces data echo of the mesh coordinates and topology.
`verbose`	produces a data echo of the primitive variables every N_{print} time steps.

3.5.8 prti

Set the hard copy print interval, N_{print}, for the output of primitive variables. This option requires **prtlev**=`verbose`.

Syntax:

plti N_{print}

Aliases: *NONE*

Parameter Description:

N_{print} (integer, default=10) Output every N_{print} time steps.

3.5.9 thti

Set the interval to write time-history data to the time-history files.

Syntax:

thti N_{step}

Aliases: *NONE*

Parameter Description:

N_{step} (integer, default=1) Output time history every N_{step} time steps.

3.5.10 ttyi

Set the interval to report the min/max values of the velocity to N_{step}. The min/max values are normally written to the screen at run-time.

Syntax:

ttyi N_{step}

Aliases: *NONE*

Parameter Description:

N_{step} (integer, default=10) Output min/max values every N_{step} time steps.

3.6 Time Step and Time Integration Options

The keywords described in this section are provided to set/modify the parameters associated with the time step and the associated time integration methods. For some physics, more extensive time-step control options are available in addition to or in lieu of the basic control provided here.

3.6.1 nstep

Defines the maximum number of time steps to be taken during a single simulation. See also the **term** keyword, described in §3.6.2.

Syntax:

nstep N_{step}

Aliases: *NONE*

Parameter Description:

N_{step} (integer, default=10) Maximum number of time steps to take.

3.6.2 term

Define the simulation termination time, in units consistent with the problem definition. The **term** keyword and **nstep** (described in §3.6.1) keywords both affect the the length of simulated time in Hydra. If the number of time steps specified using the **nstep** keyword would yield a simulation time greater than the termination time, the number of time steps is reduced to terminate the calculation according to the **term** command. Thus, the **term** keyword places a ceiling on the simulation termination time regardless of how many time steps have been specified by the **nstep** keyword.

Syntax:

term τ_f

Aliases: termination

Parameter Description:

τ_f (float, default=1.0) Termination time.

3.6.3 deltat

Defines the time step size, Δt, to be used. This value may be over-ridden by physics specific constraints on the time step.

Syntax:

deltat Δt

Aliases: *NONE*

Parameter Description:

Δt (float, default=0.01) The time step value

3.7 Turbulence Statistics

This section describes keywords that used for collecting turbulence statistics, for physics that are relevant.

3.7.1 plotstatvar

The **plotstatvar** block specifies plot output for statistics data. Its syntax is identical to that of **plotvar**; see §3.5.4. See also the **statistics** keyword, described in §3.7.2.

3.7.2 statistics

Sets options for accumulating statistics of fluctuating variables, which are defined in block **plotstatvar** (described in §3.7.1).

Syntax:

statistics
  starttime t_{start}
  endtime t_{end}
  plotwinsize *size*
  ...
end

Aliases: *NONE*

Parameter Description:

starttime t_{start} (float, default=0.0) Start time for collecting statistics
endtime t_{end} (float, default=1.0) End time for collecting statistics
plotwinsize *size* (float, default=0.1) Time window size for output of field statistics

Chapter 4

Cell-Centered Incompressible Navier-Stokes

The cell-centered incompressible Navier-Stokes solver uses finite volume discretization with a monotonicity-preserving advection algorithm and node-centered pressures to provide high-accuracy solutions for incompressible and low-Mach number flows. This chapter describes the keywords, that in conjunction with the general analysis keywords 3, may be used for calculating solutions to the incompressible flow Navier-Stokes equations. Additional information on the theoretical aspects of the incompressible flow solver may be found in the Hydra-TH Theory Manual[8].

4.1 cc_navierstokes

Input for this physics is contained in the **cc_navierstokes - end** block:

```
cc_navierstokes
   ...
   [analysis and incompressible Navier-Stokes specific keywords]
   ...
end
```

The supported analysis keywords supported are described in chapter 3. The remainder of this section describes the keywords specific to **cc_navierstokes** – **end**.

4.2 energy

This keyword activates the solution of the energy equation, and selects the form of the energy equation. If this keyword is used multiple times, the last occurrence defines the form of the energy equation.

Syntax:

energy *form*

Aliases: *NONE*

Parameter Description:

form (string, default=isothermal) Specifies the form of the energy equation.

`isothermal`	No energy equation is solved – the flow is isothermal
`temperature`	The energy equation is solved in terms of temperature
`enthalpy`	The energy equation is solved in terms of specific enthalpy

4.3 hydrostat

Prescribe the hydrostatic pressure. This may be used in conjunction with prescribed pressure boundary conditions, or by itself. When used by itself, the **hstat** keyword plays two roles. It makes the pressure-Poisson equation non-singular and it permits the pressure for the system to be uniquely determined. When the **hstat** keyword is used with prescribed pressure boundary conditions, then it only specifies the unique hydrostatic pressure level for the system. In either case, the pressure time-history and field output is adjusted to reflect the specified hydrostatic pressure level.

Syntax:

hydrostat
  nodeset *setId loadCurveId amplitude*
  ...
end

Note that multiple **nodeset** keywords may be specified within a single **hydrostat** block. However, only the last nodeset specified is used to the hydrostatic pressure.

Aliases: hstat

Parameter Description:

setId (integer, required) Specifies the node-set where the hydrostatic condition will be applied. Only a single node may be used in the nodeset.

loadcurveId (integer, required) Specify the load curve Id.

amplitude (float, required) Prescribed hydrostatic pressure level.

4.4 Initial Conditions

This section describes the keywords used to prescribe initial conditions.

4.4.1 initial

For many CFD problems, a simplified specification of initial conditions is sufficient. The **initial** – **end** block provides a simplified input mechanism for prescribing all initial conditions.

Syntax:

initial
  velx v_x
  vely v_y
  velz v_z
  tke k
  eps ε
  omega ω
  turbnu ν_T
  temperature T
  enthalpy h
end

Aliases: init

Parameter Description:

velx v_x (float, default=`0.0`) x-component of velocity.
vely v_y (float, default=`0.0`) y-component of velocity.
velz v_z (float, default=`0.0`) z-component of velocity.
tke k (float, default=`0.0`) Turbulent kinetic energy ($k-\epsilon$ and $k-\omega$ models).
eps ε (float, default=`0.0`) Turbulent dissipation rate ($k-\epsilon$ models).
omega ω (float, default=`0.0`) Inverse dissipation time scale (used for $k-\omega$ models).
turbnu k (float, default=`0.0`) Turbulent viscosity (Spalart-Allmaras and DES models).
temperature T (float, default=`0.0`) Temperature. Alias: **temp**.
enthalpy h (float, default=`0.0`) Enthalpy.

4.5 Body Forces

This section describes the specification of various body forces for the momentum equation.

4.5.1 body_force

Specifies a body force for the momentum equation. It may be time-varying, by specifying a load curve. In addition, the body force may be prescribed for a specific element set (element block) or all element sets in the mesh.

Syntax:

body_force
  set *setId*
  lcid *loadCurveId*
  fx *amplitude*
  fy *amplitude*
  fz *amplitude*
end

Aliases: bodyforce

Parameter Description:

set *setId* (integer, required) Specifies the element set on which the body force will be applied. The value -1 specifies all sets.
lcid *loadcurveId* (integer, optional) Specify the load curve Id. If not specified, then the force is assumed constant in time.
fx *amplitude* (float, default=`0.0`) Body force in the x-direction
fy *amplitude* (float, default=`0.0`) Body force in the y-direction
fz *amplitude* (float, default=`0.0`) Body force in the z-direction

4.5.2 boussinesqforce

Specifies a body force using the Boussinesq approximation to represent the buoyant forces induced by temperature. This body force is only active when the energy equation is solved in conjunction with the momentum equations. A load curve may be used to represent the effects of a time-dependent gravity field. The Boussinesq body force may be prescribed for a specific element set (element block) or all element sets in the mesh.

Syntax:

boussinesqforce
  set *setId*
  lcid *loadCurveId*
  gx *amplitude*
  gy *amplitude*
  gz *amplitude*
end

Aliases: bbodyforce

Parameter Description:

set *setId* (integer, required) Specifies the element set on which the force will be applied. The value -1 specifies all sets.
lcid *loadcurveId* (integer, optional) Specify the load curve Id. If not specified, then the force is assumed constant in time.
gx *amplitude* (float, default=`0.0`) Gravity force in the x-direction
gy *amplitude* (float, default=`0.0`) Gravity force in the y-direction
gz *amplitude* (float, default=`0.0`) Gravity force in the z-direction

4.6 Boundary Conditions

This section describes the specification of boundary conditions for the incompressible Navier-Stokes.

4.6.1 Scalar Dirichlet Boundary Conditions

All scalar Dirichlet boundary conditions are specified using the same form, described in this section. The scalar values that may be specified are given in Table 4.1.

Table 4.1: Scalar Dirichlet Boundary Condition Keywords

BC Keyword	Aliases	Description
enthalpybc	**ebc, enthalpy**	enthalpy, h
epsbc	**eps**	Turbulent dissipation, ε
distancebc	**dist, distance**	Distance function
pressurebc	**pbc, pressure**	hydrodynamic pressure, p
temperaturebc	**tbc, temperature**	temperature, T
turbnubc	**turbnu**	Turbulent viscosity, ν_T

Syntax:

BC
    sideset *setId loadCurveId amplitude*
    ...
end

Note that multiple **sideset** keywords may be specified within a single **BC** block. See Table 4.1 for valid values for keyword **BC**.

Parameter Description:

setId (integer, required) Specifies the side-set on which the boundary condition will be applied.
loadcurveId (integer, required) Specify the load curve Id.
amplitude (float, required) Scalar value of BC.

4.6.2 Velocity Dirichlet Boundary Conditions

Dirichlet velocity boundary conditions are specified in a component form using a sideset and load curve identifier.

Syntax:

velocitybc
    velx sideset *setId loadCurveId amplitude*
    vely sideset *setId loadCurveId amplitude*
    velz sideset *setId loadCurveId amplitude*
    ...
end

Note that multiple **velx sideset**, **vely sideset** and **velz sideset** keywords may be specified within a single **velocitybc** block.

Aliases: vel, velocity

Parameter Description:

setId (integer, required) Specifies the side-set on which the velocity boundary condition will be applied.
loadcurveId (integer, required) Specify the load curve Id.
amplitude (float, required) Scalar value of BC.

4.6.3 Symmetry Velocity Boundary Conditions

This boundary condition specifies a velocity symmetry condition in a coordinate direction on a side-set. The surface normal on the specified side-set is aligned with the coordinate direction.

Syntax:

symmetrybc
  velx sideset *setId loadCurveId amplitude*
  vely sideset *setId loadCurveId amplitude*
  velz sideset *setId loadCurveId amplitude*
  ...
end

Note that multiple **velx sideset**, **vely sideset** and **velz sideset** keywords may be specified within a single **symmetry** block.

Aliases: symmetry

Parameter Description:

setId (integer, required) Specifies the side-set on which the velocity boundary condition will be applied.
loadcurveId (integer, required) Specify the load curve Id.
amplitude (float, required) Scalar value of BC.

4.6.4 heatflux

Specifies a heat flux boundary condition.

Syntax:

heatflux
  sideset *setId loadCurveId amplitude*
  ...
end

Note that multiple **sideset** keywords may be specified within a single **heatflux** block.

Aliases: *NONE*

Parameter Description:

setId (integer, required) Specifies the side-set on which the boundary condition will be applied.
loadcurveId (integer, required) Specify the load curve Id.
amplitude (float, required) Surface-normal component of heat flux.

4.6.5 passiveoutflowbc

This **passiveoutflowbc** – **end** keywords provide a passive advective condition for use at outflow boundaries. This provides a mechanism to suppress artificial re-entrant flow conditions when the outflow boundary is not normal to the primary flow direction.

Syntax:

> **passiveoutflowbc**
>   **sideset** *setId*
>   ...
> **end**
>
> Note that multiple **sideset** keywords may be specified within a single **passiveoutflowbc** block.

Aliases: passiveoutflow

Parameter Description:

> *setId* (integer, required) Specifies the side-set on which the boundary condition will be applied.

4.6.6 pressureoutflowbc

This boundary condition is typically applied at outflow boundaries where the variation in pressure due to vortical flow structures is large. This boundary condition applies an extrapolated pressure as a traction force on the momentum equations to avoid large pressure jumps at an outflow boundary.

Syntax:

> **pressureoutflowbc**
>   **sideset** *setId*
>   ...
> **end**
>
> Note that multiple **sideset** keywords may be specified within a single **pressureoutflowbc** block.

Aliases: pressureoutflow

Parameter Description:

> *setId* (integer, required) Specifies the side-set on which the boundary condition will be applied.

4.7 Heat Sources

This section describes the specification of various body forces for the momentum equation.

4.7.1 heat_source

Specifies a volumetric heat source. It may be time-varying, by specifying a load curve. In addition, the heat source may be prescribed for a specific element set (element block) or all element sets in the mesh.

Syntax:

heat_source
  set *setId*
  lcid *loadCurveId*
  Q *amplitude*
end

Aliases: heatsource

Parameter Description:

set *setId* (integer, required) Specifies the element set where the heat force will be applied. The value -1 specifies all sets.

lcid *loadcurveId* (integer, optional) Specify the load curve Id. If not specified, then the force is assumed constant in time.

Q *amplitude* (float, default=`0.0`) volume tic heat source

4.8 Pressure, Momentum and Transport Solvers

This section describes the linear solvers that are available for solving the pressure-Poisson, momentum and auxiliary transport equations.

4.8.1 ppesolver

Define the attributes of the pressure-Poisson solver.

Syntax:

ppesolver
  type *method*
  smoother *AMGsmoother*
  cycle *AMGcycle*
  solver *AMGsolver*
  pre_smooth *AMGpreSmooth*
  post_smooth *AMGpostSmooth*
  levels *AMGlevels*
  itmax N_{iter}
  itchk N_{check}
  diagnostics *flag*
  convergence *flag*
  eps ϵ

zeropivot *pivot*
end

Aliases: ppesol

Parameter Description:

type *method* (string, default=`AMG`) Specifies the preconditioner – Krylov solver combination. Values can be one of the following:
- `AMG` Algebraic multigrid with the conjugate gradient method
- `SSORCG` Successive over-relaxation preconditioner with the conjugate gradient method
- `JPCG` Jacobi preconditioner with the conjugate gradient method

smoother *AMGsmoother* (string, default=`ICC`) Specifies the smoother for AMG solver. Values can be one of the following:
- `ICC` Incomplete Cholesky factorization with no-fill
- `ILU` Incomplete LU factorization with no-fill
- `SSOR` Successive over-relaxation
- `CHEBYCHEV` Chebychev polynomial smoother

cycle *AMGcycle* (string, default=`V`) Specifies the type of AMG cycle. Values for can be one of the following:
- `V` Use a V-cycle
- `W` Use a W-cycle

solver *AMGsolver* (string, default=`CG`) Specifies the underlying Krylov solver to be used with AMG. Values for can be one of the following:
- `CG` Conjugate gradient method
- `BCGS` Stabilized bi-conjugate gradient squared method
- `FGMRES` Flexible generalized minimum residual method

pre_smooth *AMGpreSmooth* (integer, default=`1`) Set the number of pre-smoothing sweeps for AMG.

pre_smooth *AMGpostSmooth* (integer, default=`1`) Set the number of post-smoothing sweeps for AMG.

levels *AMGlevels* (integer, default=`10`) Set the maximum number of AMG levels to use in the multigrid cycle.

itmax N_{itmax} (integer, default=`500`) Set the maximum number of iterations. In the case of AMG, this is the maximum number of V or W cycles.

itchk N_{itchk} (integer, default=`2`) Set the number of iterations to take before checking convergence criteria.

diagnostics *flag* (string, default=`off`) Enable/disable the diagnostic information from the linear solver. Values for may be one of the following:
- `off` Suppress diagnostic output
- `on` Activate diagnostic output

convergence *flag* (string, default=`off`) Enable/disable the convergence metrics for the linear solver. Values may be one of the following:
- `off` Suppress convergence output
- `on` Activate convergence output

eps ϵ (float, default=`1.0e-5`) Specify the convergence criteria for the linear solver.
pivot *pivot* (float, default=`1.0e-16`) Specify the value of a zero pivot for preconditioner.

4.8.2 momentumsolver

Define the attributes of the momentum solver.

Syntax:

momentumsolver
  type *method*
  restart $N_{restart}$
  itmax N_{itmax}
  itchk N_{check}
  diagnostics *flag*
  convergence *flag*
  eps ϵ
end

Aliases: momsol

Parameter Description:

type *method* (string, default=`FGMRES`) Specifies the preconditioner – Krylov solver combination. Values can be one of the following:
- `FGMRES` Flexible generalized minimum residual method
- `ILUFGMRES` ILU-preconditoned FGMRES
- `GMRES` Generalized minimum residual method
- `ILUGMRES` ILU-preconditioned GMRES

restart $N_{restart}$ (integer, default=`30`) Specifies the number of restart vectors used with GMRES/FGMRES.
itmax N_{itmax} (integer, default=`500`) Set the maximum number of iterations.
itchk N_{itchk} (integer, default=`2`) Set the number of iterations to take before checking convergence criteria.
diagnostics *flag* (string, default=`off`) Enable/disable the diagnostic information from the linear solver. Values for may be one of the following:
- `off` Suppress diagnostic output
- `on` Activate diagnostic output

convergence *flag* (string, default=`off`) Enable/disable the convergence metrics for the linear solver. Values may be one of the following:
- `off` Suppress convergence output
- `on` Activate convergence output

eps ϵ (float, default=`1.0e-5`) Specify the convergence criteria for the linear solver.

4.8.3 transportsolver

Define the attributes of the solver used for auxiliary transport equations. This includes the energy equation, and transport equations associated with turbulence models.

Syntax:

transportsolver
  type *method*
  restart $N_{restart}$
  itmax N_{itmax}
  itchk N_{check}
  diagnostics *flag*
  convergence *flag*
  eps ϵ
end

Aliases: trnsol

Parameter Description:

type *method* (string, default=`FGMRES`) Specifies the preconditioner – Krylov solver combination. Values can be one of the following:
- `FGMRES` Flexible generalized minimum residual method
- `ILUFGMRES` ILU-preconditoned FGMRES
- `GMRES` Generalized minimum residual method
- `ILUGMRES` ILU-preconditioned GMRES

restart $N_{restart}$ (integer, default=`30`) Specifies the number of restart vectors used with GMRES/FGMRES.

itmax N_{itmax} (integer, default=`500`) Set the maximum number of iterations.

itchk N_{itchk} (integer, default=`2`) Set the number of iterations to take before checking convergence criteria.

diagnostics *flag* (string, default=`off`) Enable/disable the diagnostic information from the linear solver. Values for may be one of the following:
- `off` Suppress diagnostic output
- `on` Activate diagnostic output

convergence *flag* (string, default=`off`) Enable/disable the convergence metrics for the linear solver. Values may be one of the following:
- `off` Suppress convergence output
- `on` Activate convergence output

eps ϵ (float, default=`1.0e-5`) Specify the convergence criteria for the linear solver.

4.9 time_integration

Define the attributes of the time-integration method used to solve the Navier-Stokes equations. The time-step control may be either fixed time-step or using a time-step based on a fixed CFL condition.

The fixed CFL time-step control uses the initial velocity field and with the initial CFL number CFL_0 to estimate the time-step. For all subsequent time-steps, CFL_{max} is used estimate the time-step. When the time-step can increase based on CFL_{max}, the growth is based on the time-step scale factor α. An upper bound on the time-step is set with Δt_{max}. This is shown schematically in Figure 4.1.

Syntax:

time_integration
  type *method*
  CFLinit CFL_0
  CFLmax CFL_{max}
  dtmax Δt_{max}
  dtscale α
  thetaa θ_A
  thetak θ_K
  thetaf θ_F
end

Aliases: timeint

Parameter Description:

type *method* (string, default=`fixed_cfl`) Specifies the time-step control method. Values can be one of the following:

- `fixed_cfl` Sets the time-step based on a fixed maximum CFL condition using CFL_{max}
- `fixed_dt` Uses a fixed time-step based on Δt_{max}

CFLinit CFL_0 (float, default=`1.0`) Specifies the initial CFL number to use at startup with the fixed CFL time-step control

CFLmax CFL_{max} (float, default=`2.0`) Set the maximum CFL number to use with the fixed CFL time-step control

dtmax Δt_{max} (float, default=`1.0`) Set the maximum time step that can be used with the fixed CFL time-step control

dtscale α (float, default=`1.025`) Factor used to increase the time-step with the fixed CFL time-step control

thetaa θ_A (float, default=`0.5`) Time-weight for the advective terms. By default, the time-weight uses a second-order centering in time. For explicit advection, $\theta_A = 0.0$, and for an implicit treatment, $\theta_A = 1.0$.

thetaK θ_K (float, default=`0.5`) Time-weight for the viscous/diffusive terms. By default, the time-weight uses a second-order centering in time. For an explicit treatment, $\theta_K = 0.0$, and for an implicit treatment, $\theta_K = 1.0$.

thetaF θ_F (float, default=`0.5`) Time-weight for source terms. By default, the time-weight uses a second-order centering in time. For an explicit treatment, $\theta_K = 0.0$, and for an implicit treatment, $\theta_K = 1.0$.

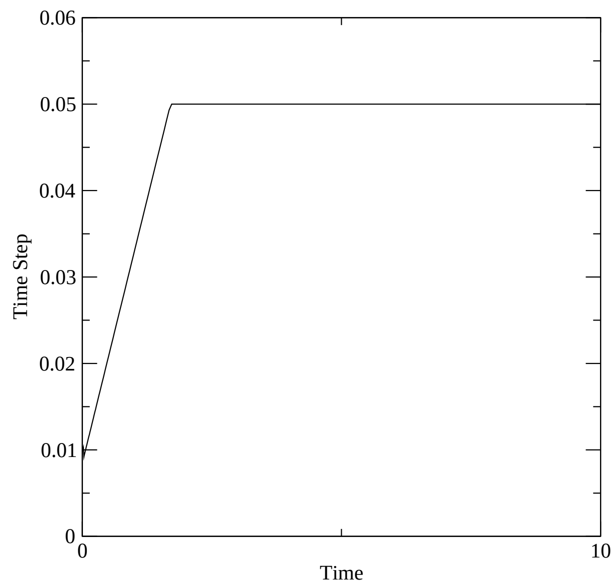


Figure 4.1: Fixed CFL time-step control where $\alpha = 1.025$ sets the slope of increasing time-step vs. time curve, and $\Delta t_{max} = 0.05$ sets the upper-bound for the time-step.

4.10 turbulence

This keyword activates the use of a turbulence model and selects the specific turbulence model.

Syntax:

turbulence *model*

Aliases: tmodel

Parameter Description:

form (string, default=no_turbmodel) Specifies the form of the energy equation.

`no_turbmodel`	No explicit turbulence model is used
`spalart_allmaras`	Activate the Spalart-Allmaras model
`spalart_allmaras_des`	Activate the detached-eddy model based on Spalart-Allmaras
`smagorinsky`	Use the Smagorinsky subgrid-scale model
`wale`	Use the wall-adapted large eddy subgrid-scale model
`rng_ke`	Activate the RNG k-ϵ model
`sst_kw`	Activate the SST k-ω model
`ksgs`	Use the k^{sgs} subgrid-scale model
`ldkm_ksgs`	Use the LDKM k^{sgs} subgrid-scale model

4.11 Output Variables

Hydra relies on output delegates to provide a rich suite of field and time-history output for visualization and post-simulation analysis.

At run-time, a complete list of registered output delegates is provided so the user may select the appropriate output for field, statistical and time-history data. Requested output variables that are not available for a specific analysis are identified with a warning message.

4.11.1 Instantaneous Field Output

Instantaneous field output may be requested at element, node or surface centering using the **plotvar** – **end** keyword block. The specific syntax used for output requests is presented in §3.5.4. Table 4.2 shows the complete list of variables and centering for instantaneous field output.

Variable Name	Centering	Meaning
`density`	Element, Node	Fluid density
`dist`	Element, Node	Normal distance from no-slip/no-penetration surfaces
`div`	Element	Velocity divergence ($\nabla \cdot \mathbf{v}$)
`enthalpy`	Element, Node	Fluid specific enthalpy
`enstrophy`	Element, Node	Enstrophy ($\omega \cdot \omega/2$)
`helicity`	Element, Node	Helicity ($\mathbf{v} \cdot \omega$)
`pressure`	Element, Node	Fluid pressure
`procid`	Element, Node	Processor Id (MPI Rank)
`temp`	Element, Node	Fluid temperature
`turbeps`	Element, Node	Turbulent kinetic energy dissipation rate, (ϵ)
`turbke`	Element, Node	Turbulent kinetic energy (k)
`turbnu`	Element, Node	Turbulent eddy viscosity
`u`	Node	Nodal displacement vector for ALE computations
`vel`	Element, Node	Fluid velocity vector
`vginv2`	Element, Node	Q-criteria, i.e., 2nd invariant of velocity gradient
`vorticity`	Element, Node	Vorticity, i.e., curl of velocity ($\nabla \times \mathbf{v}$)
`traction`	Face	Traction force vector
`straction`	Face	Shear traction force vector
`ntraction`	Face	Normal traction force vector
`wallshear`	Face	Wall shear force
`yplus`	Face	y^+ at a wall
`ystar`	Face	y^* at a wall
`heatflux`	Face	Heat flux vector at a wall
`nheatflux`	Face	Normal heat flux at a wall

Table 4.2: Instantaneous field output variables.

4.11.2 Statistics Output

Statistics field output may be requested at element, node or surface centering using the **plotstatvar** – **end** keyword block, whose syntax is the same as the **plotvar** – **end** block, described in §3.5.4. Statistics output requests should be used in conjunction with the **statistics** – **end** keyword block (see §3.7.2). Table 4.3 shows the list of variables and centering for statistics field output that are currently available. The statistics variable names adhere to the following conventions:

- Instantaneous variables are simply denoted by the variable name, e.g. density, see Table 4.2.
- Means, or more specifically Reynolds means, denoted by angled brackets: `< · >`. The mean (or mathematical expectation) is defined by

$$\langle \phi \rangle = \int \phi f(\phi) \mathrm{d}\phi, \tag{4.1}$$

  where $f(\phi)$ is the probability density function of the fluctuating variable, ϕ. Assuming the ergodic theorem holds, the expectation in Eq. 4.1 is numerically estimated by Δt-weighted time-averages over N time steps:

$$\langle \phi \rangle \approx \frac{\sum_{i=1}^{N} \phi^i \Delta t^i}{\sum_{i=1}^{N} \Delta t^i} \tag{4.2}$$

  where i is the time step.

- An apostrophe, $'$, denotes fluctuation about the Reynolds mean: $q' = q - \langle q \rangle$.
- The covariance of N variables, $p, q, \dots, r$, is generally denoted by `<p',q',...,r'>`, defined as

$$\langle p' q' \cdot \dots \cdot r' \rangle = \Big\langle \big(p - \langle p \rangle\big)\big(q - \langle q \rangle\big) \cdot \dots \cdot \big(r - \langle r \rangle\big) \Big\rangle. \tag{4.3}$$

  As an example, the variable name for the density-pressure covariance is denoted by `<density',pressure'>` and defined as the central moment

$$\langle \rho' p' \rangle = \Big\langle \big(\rho - \langle \rho \rangle\big)\big(p - \langle p \rangle\big) \Big\rangle. \tag{4.4}$$

- The variable names `tke` and `reynoldsstress` denote the turbulent kinetic energy and Reynolds stress tensor respectively, defined by the averages of the dot-, and tensor-products of the fluctuating velocity vector, `tke` $= \langle \mathbf{v}' \cdot \mathbf{v}' \rangle / 2$, and `reynoldsstress` $= \langle \mathbf{v}' \mathbf{v}' \rangle$, respectively.
- The "`rms-`" prefix denotes the square-root of the variance of the given fluctuating variable. For example, `rms-pressure` $= \langle p'^2 \rangle^{1/2}$.

Variable Name	Centering	Meaning
`<density>`	Element, Node, Face	Mean density
`<pressure>`	Element, Node, Face	Mean pressure
`<velocity>`	Element, Node, Face	Mean velocity vector
`<temperature>`	Element, Node, Face	Mean temperature
`<enstrophy>`	Element, Node	Mean enstrophy
`<heatflux>`	Face	Mean heat flux vector
`<helicity>`	Element, Node	Mean helicity
`<vorticity>`	Element, Node	Mean vorticity vector
`<pressure',pressure'>`	Element, Node, Face	Pressure variance
`<temp',temp'>`	Element, Node, Face	Temperature variance
`<density',pressure'>`	Element, Node, Face	Density-pressure covariance
`<pressure',velocity'>`	Element, Node	Pressure-velocity covariance
`rms-pressure`	Element, Node, Face	Root-Mean-Square pressure
`rms-temp`	Element, Node, Face	Root-Mean-Square temperature
`tke`	Element, Node	Turbulent kinetic energy
`reynoldsstress`	Element, Node	Reynolds stress tensor

Table 4.3: Statistics field output variables.

4.11.3 Time-History Output

Variable Name	Centering	Meaning
`density`	Element	Fluid density
`div`	Element	Velocity divergence ($\nabla \cdot \mathbf{v}$)
`enstrophy`	Element	Enstrophy, i.e., square of vorticity
`enthalpy`	Element	Fluid specific enthalpy
`helicity`	Element	Helicity, i.e., dot product of velocity and vorticity
`pressure`	Element	Fluid pressure
`temp`	Element	Fluid temperature
`turbeps`	Element	Turbulent dissipation rate (ϵ)
`turbke`	Element	Turbulent kinetic energy (k)
`turbnu`	Element	Turbulent eddy viscosity
`vel`	Element	Fluid velocity vector
`vorticity`	Element	Vorticity, i.e., curl of velocity ($\nabla \times \mathbf{v}$)
`avgpress`	Surface	Average pressure for a surface
`avgtemp`	Surface	Average temperature for a surface
`avgvel`	Surface	Average velocity vector for a surface
`force`	Surface	Integral force acting on a surface
`fvol`	Surface	Integrated volume of fluid that has crossed a surface
`heatflow`	Surface	Heat flow rate crossing a surface
`massflow`	Surface	Mass flow rate crossing a surface
`pressforce`	Surface	Integral pressure force acting on a surface
`surfarea`	Surface	Surface area
`viscforce`	Surface	Integral viscous shear force acting on a surface
`volumeflow`	Surface	Volume flow rate crossing a surface

Table 4.4: Time-history field output variables.

Chapter 5

Incompressible Navier-Stokes Example Problems

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

5.1 Poisueille Flow

Poiseuille flow in a channel is characterized by a balance between a pressure gradient and viscous shear forces, i.e.,

$$\frac{1}{Re}\frac{\partial^2 v_x}{\partial y^2} = \frac{\partial p}{\partial x} \tag{5.1}$$

where Re is the Reynolds number based on channel height H, and v_x, p, and x, y are the non-dimensional x-velocity, pressure and coordinates. The 2-D channel flow problem considered here consists of a 2-D duct with a 20:1 aspect ratio, $Re = 100$, and $\partial p/\partial x = -0.12$. This choice of Reynolds number and pressure gradient results in a steady parabolic velocity profile with $u_{max} = 1.5$ and $u_{avg} = 1.0$. Here, the non-dimensional equations were obtained using a length scale H, velocity scale U, and time scale U/H. Figure 5.1 shows the mesh with the side set Id's that are used to prescribe boundary conditions.

In order to represent the constant pressure gradient, an inflow pressure boundary condition of $p = 2.4$ is prescribed with $v_y = 0$ and $v_z = 0$. No-slip and no-penetration conditions $\mathbf{v} = (0, 0, 0)$ are prescribed along the top and bottom duct walls. The so-called "natural" velocity boundary conditions are applied at the outflow boundary. These conditions require no input by the user, and correspond to homogeneous Neumann conditions on the velocity in the boundary normal direction. The pressure at the outflow boundary is prescribed to be $p = 0$.

The control file is shown in Figure 5.2 for a Reynolds number of 100 based on the channel height. In this example, the time-accurate, second-order projection algorithm is used with time-weights for

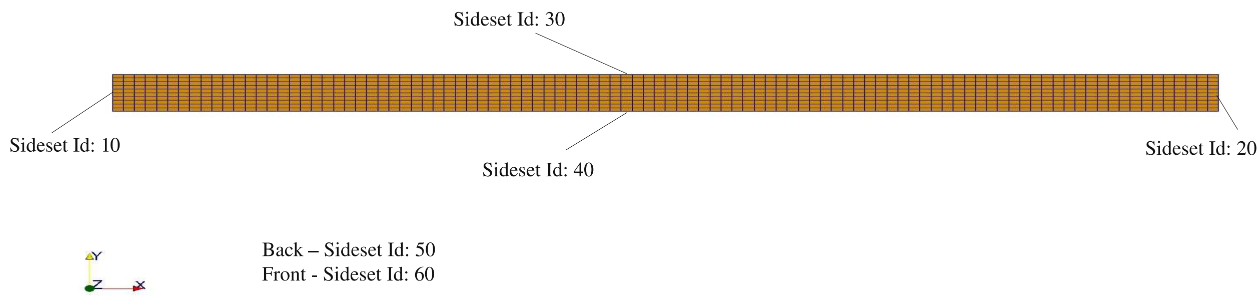


Figure 5.1: Mesh for two-dimensional Poiseuille flow.

a backward-Euler time integrator, i.e., `thetaa` = 1.0, `thetak` = 1.0, and `thetaf` = 1.0. Additional information on the time integration methods may be found in the Hydra-TH Theory Manual [8].

An initial CFL of 1.0 with a maximum CFL of 10.0 is prescribed with a maximum time step `dtmax` of 0.5, and an initial time step based on `deltat` = 0.001. The diffusional time scale for this problem is approximately 100 time units, and a total simulation time `term` = 150 time units is specified.

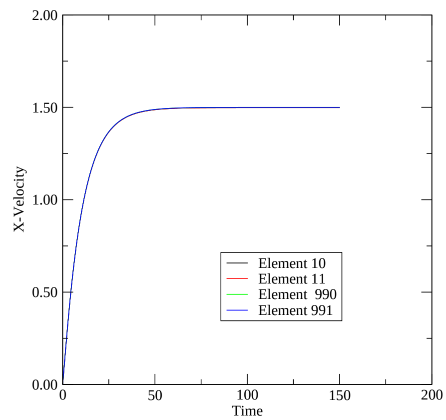
The duct calculation is carried out until a steady-state is achieved as indicated in Figure 5.3. Time-history points were placed at elements along the centerline of the duct at the inlet (elements 10, 11) and at the outlet (elements 990, 991). The time-history of the x-velocity is plotted in Figure 5.3 (a), and the global kinetic energy is shown in Figure 5.3 (b). The velocity and kinetic energy both reach asymptotic values by 100 time units indicating a steady-state has been reached. The time-step vs. time is shown in Figure 5.3 (c). The slope of the increasing time-step vs. time curve is defined by `dtscale` = 1.025 in the control file, and the maximum time-step is set by `dtmax` = 0.5. The x-velocity profile at the outlet is shown in Figure 5.3 (c), and is plotted with the exact solution for the velocity.

Example Problem Files:
Mesh File: duct.exo
Control File: duct.cntl

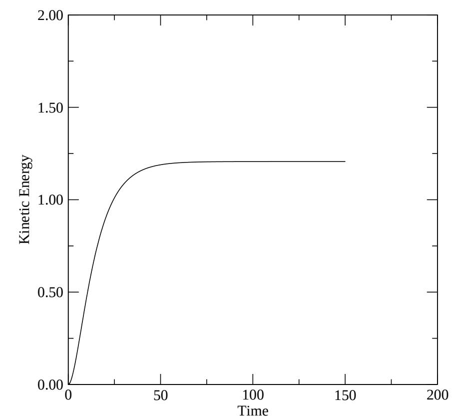
```
title
Re = 100 laminar Poiseuille flow

cc_navierstokes

  nsteps   1000
  deltat   0.001
  term     150.0

  time_integration
    type    fixed_cfl
    CFLinit 1.0
    CFLmax  10.0
    dtmax   0.5
    dtscale 1.025
    thetaa  1.0
    thetak  1.0
    thetaf  1.0
  end

  # Output options
  pltype exodusii
  filetype serial
  plti   50
  ttyi   10

  # Material model definition
  material
    id  1
    rho 1.0
    mu  1.0e-2
  end

  materialset
    id 10
    material 1
    block 1
  end

  plotvar
    elem vel
    elem volume
    elem density
    elem procid
    elem div
    elem enstrophy
    node vel
    node pressure
  end

  histvar
    elem 10  vel
    elem 11  vel
    elem 990 vel
    elem 991 vel
  end
```

```
  # Simple IC's
  initial
    velx 0.0
    vely 0.0
    velz 0.0
  end

  # Fixed pressures
  pressure
    sideset 10 -1 2.4
    sideset 20 -1 0.0
  end

  # Velocity BC's
  velocity
   # Inlet
   vely sideset 10 -1 0.0
   velz sideset 10 -1 0.0
   # Top
   velx sideset 30 -1 0.0
   vely sideset 30 -1 0.0
   velz sideset 30 -1 0.0
   # Bottom
   velx sideset 40 -1 0.0
   vely sideset 40 -1 0.0
   velz sideset 40 -1 0.0
   # Back - symmetry in z
   velz sideset 50 -1 0.0
   # Front - symmetry in z
   velz sideset 60 -1 0.0
  end

  ppesolver
    type  AMG
    itmax 250
    itchk 1
    coarse_size 100
    diagnostics off
    convergence off
    eps 1.0e-8
  end

  momentumsolver
    type  ILUFGMRES
    itmax 50
    itchk 2
    restart 20
    diagnostics off
    convergence off
    eps   1.0e-8
  end

end

exit
```

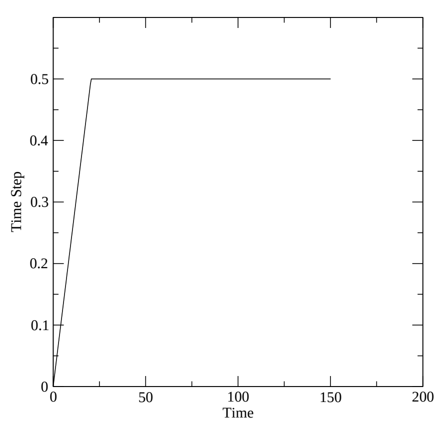
Figure 5.2: Control file for Poiseuille flow.



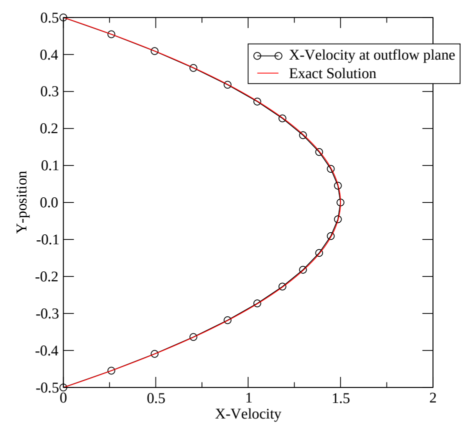
(a)



(b)

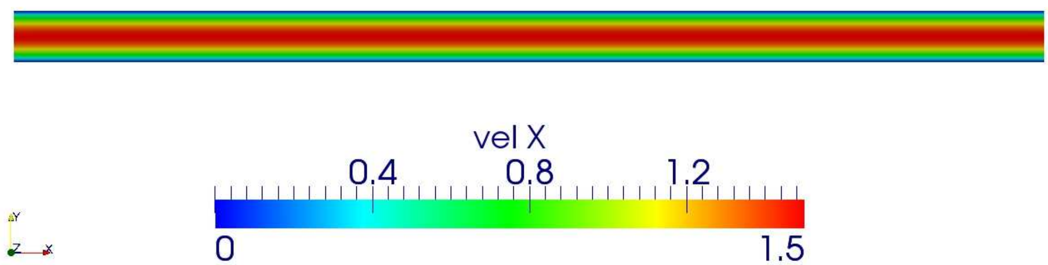


(c)

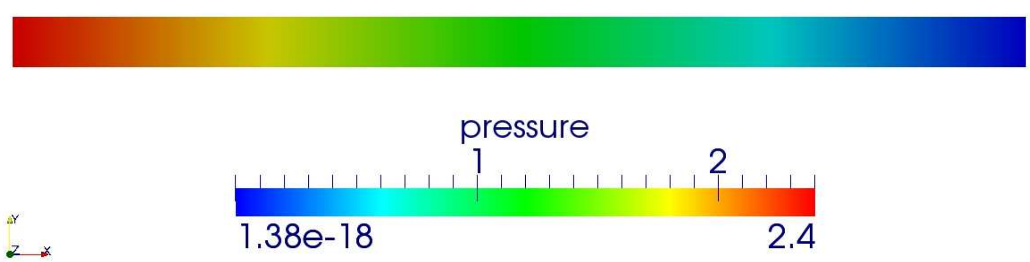


(d)

Figure 5.3: Time history plots for (a) x-velocity, (b) kinetic energy, (c) time-step, and (d) outlet x-velocity profile at t=150 time units.



(a)



(b)

Figure 5.4: Snapshot of (a) x-velocity and (c) pressure at $t = 150$ time units.

5.2 Lid-Driven Skew Cavities

This example example consists of a suite of five lid driven skewed cavity problems based on the work by Erturk and Dursun [13]. note that the results by Ghia, et al. [14] are also available for the specific 90^o lid driven cavity, but a direct comparison with this data is not included.

The geometrical configuration for the lid driven cavity is shown generically in Figure 5.5 with α defining the skew angle. On the bottom and side walls, no-slip/no-penetration boundary conditions were prescribed. Along the top "lid", a no-penetration boundary condition along with a unit lid velocity are prescribed. A single nodal pressure was prescribed in the bottom right-hand corner to set the hydrostatic pressure level.

The verification suite consists of five skewed cavities with $\alpha = 15, 30, 45, 60, 90^o$. Each skewed cavity uses three grids with 32×32, 128×128 and 256×256 elements. In order to simplify the prescription of boundary conditions, all of the meshes used a consistent sideset numbering relative to Figure 5.5 as shown in Table 5.1.

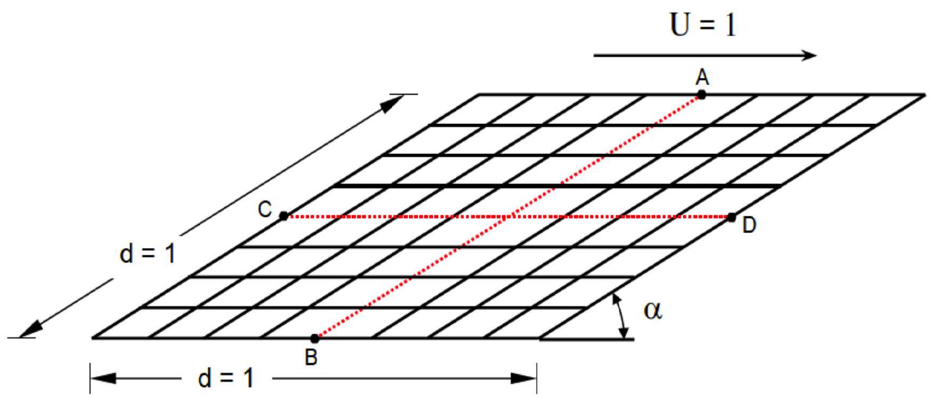


Figure 5.5: Skewed lid driven cavity geometry (reproduced from Erturk and Dursun[13] without permission).

Cavity Side	Side Set Id
Top (A)	4
Bottom (B)	5
Left (C)	1
Right (D)	2
Front/Back	3

Table 5.1: Side set Id's used for the lid-driven skew cavities.

For all computations, $CFL_{max} = 10$ and backward-Euler time integration is used since the goal is a steady-state solution. Time history plots of the global kinetic energy indicate that a steady-state solution is reached by ≈ 10 time units. Note that the diffusional time-scale varies with each skew angle slightly larger time-scales for the the larger skew angles. All problems for this example are run for 40 time units. The kinetic energy vs. time plots for the 128×128 grids are shown in Figure 5.7. Velocity data is extracted along the red center lines shown in Figure 5.5 for direct comparison with the reference data provided by Erturk and Dursun. The x-velocity profile is plotted against the vertical centerline, and the y-velocity profile is plotted against the horizontal centerline as shown in Figures 5.8 – 5.12.

All of the lid driven cavity problems achieve a steady-state (as verified by the global kinetic energy and velocity time-histories), and this provides a convenient way to assess the convergence behavior as the mesh is refined. All of the cavity meshes used uniform meshing, albeit with severely skewed elements for the 15^o cavity. Table 5.2 shows the asymptotic behavior of the kinetic energy as a function of the x-mesh size (h) which indicates $O(h^2)$ convergence in all velocity components for all of the skew angles.

Cavity Angle	Global Kinetic Energy Correlation
15^o	$0.00020907 - 0.006877\ h^2$
30^o	$0.00038731 - 0.010046\ h^2$
45^o	$0.00053854 - 0.014590\ h^2$
60^o	$0.00067314 - 0.019690\ h^2$
90^o	$0.00086136 - 0.029630\ h^2$

Table 5.2: Convergence behavior of the global kinetic energy vs. h for the lid-driven skewed cavities.

Example Problem Files:
Mesh File: ldcα_32x32.exo, ldcα_128x128.exo, ldcα_256x256.exo where $\alpha = 15,\ 30,\ 45,\ 60,\ 90^o$.
Control File: ldc_Re100.cntl

```
title
Re=100 lid-driven cavity

cc_navierstokes

  nsteps  1200
  deltat  0.01
  term    40.0

  time_integration
    type    fixed_cfl
    CFLinit 1.0
    CFLmax  10.0
    dtmax   0.05
    dtscale 1.025
    thetaa  1.0
    thetak  1.0
    thetaf  1.0
  end

  # Output options
  pltype exodusii
  filetype serial
  plti   20
  ttyi   20
  dump   0

  # Material model setup
  material
    id  1
    rho 1.0
    mu  1.0e-2
  end

  materialset
    id 10
    material 1
    block 1
  end

  plotvar
    elem density
    elem vel
    elem procid
    elem div
    node lambda
    node pressure
    node vel
    node vorticity
  end

  # Simple IC's
  initial
    velx 0.0
    vely 0.0
    velz 0.0
  end
```

```
  hydrostat
    nodeset 2 -1 0.0
  end

  velocity
    # Left wall
    velx sideset 1 -1 0.0
    vely sideset 1 -1 0.0
    velz sideset 1 -1 0.0
    # Right wall
    velx sideset 2 -1 0.0
    vely sideset 2 -1 0.0
    velz sideset 2 -1 0.0
    # Top wall (lid)
    velx sideset 4 -1 1.0
    vely sideset 4 -1 0.0
    velz sideset 4 -1 0.0
     # Bottom wall
    velx sideset 5 -1 0.0
    vely sideset 5 -1 0.0
    velz sideset 5 -1 0.0
     # Front/back
    velz sideset 3 -1 0.0
  end

  ppesolver
    type  AMG
    itmax 100
    itchk 1
    coarse_size 1000
    diagnostics off
    convergence off
    eps   1.0e-5
  end

  momentumsolver
    type  ILUFGMRES
    itmax 50
    itchk 2
    restart 45
    diagnostics off
    convergence off
    eps   1.0e-5
  end

end

exit
```

Figure 5.6: Lid driven cavity control file.

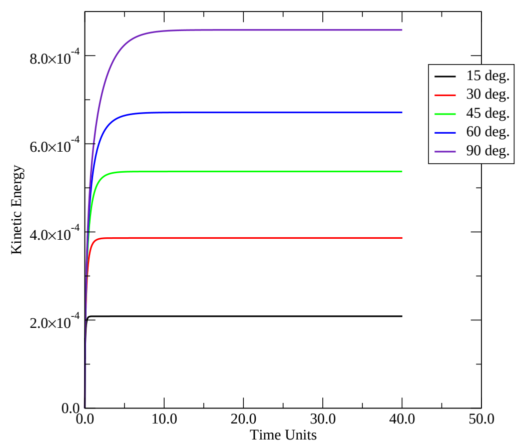
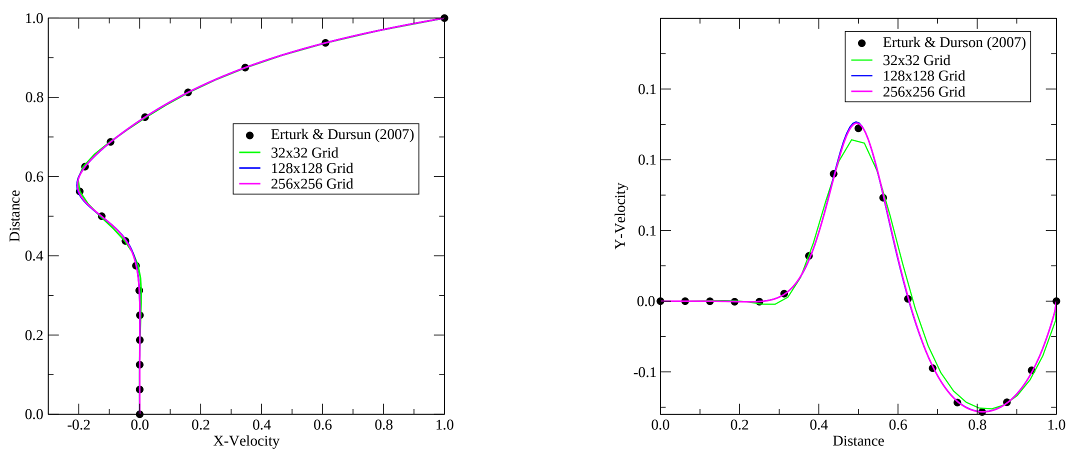
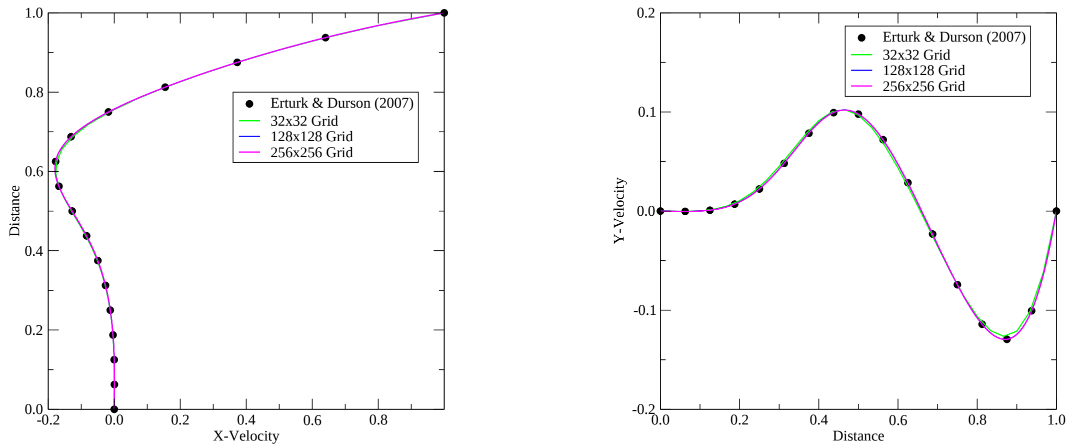


Figure 5.7: Kinetic energy vs. time for the 128×128 grids for $\alpha = 15, 30, 45, 60, 90^o$.



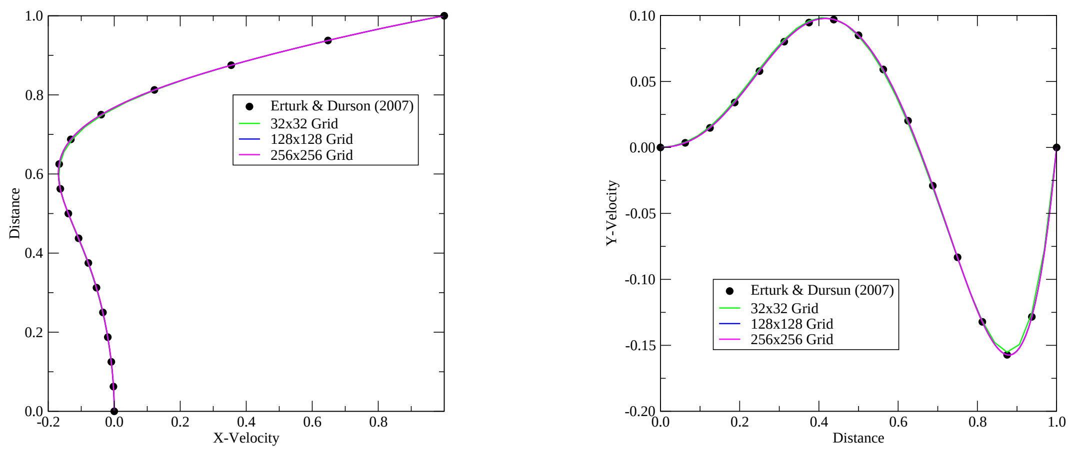
(a) (b)

Figure 5.8: 15^o lid-driven cavity: (a) x-velocity, (b) y-velocity.



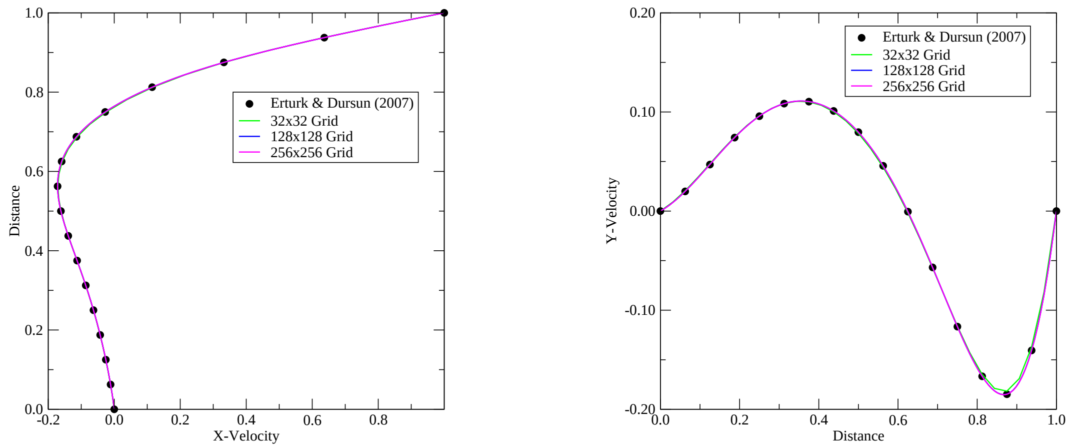
(a) (b)

Figure 5.9: 30^o lid-driven cavity: (a) x-velocity, (b) y-velocity.



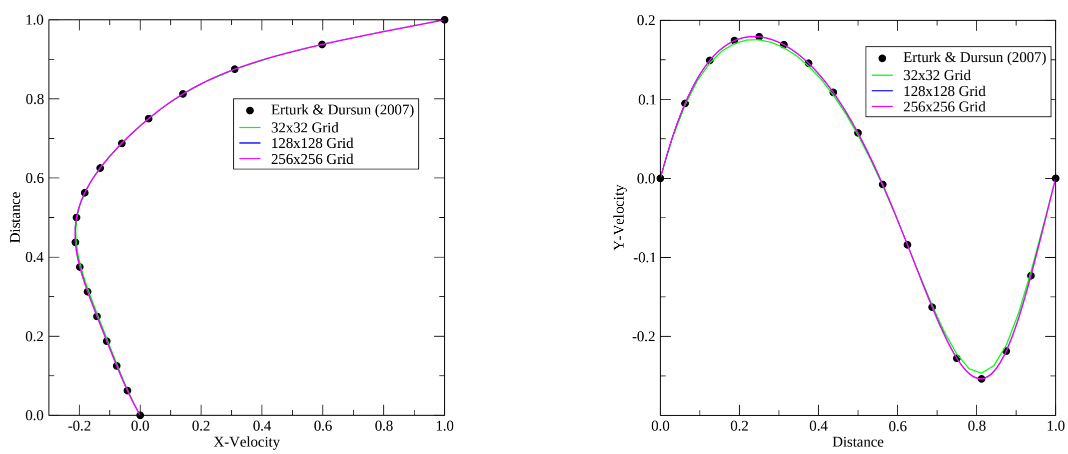
(a) (b)

Figure 5.10: 45^o lid-driven cavity: (a) x-velocity, (b) y-velocity.



(a) (b)

Figure 5.11: 60^o lid-driven cavity: (a) x-velocity, (b) y-velocity.



(a) (b)

Figure 5.12: 90^o lid-driven cavity: (a) x-velocity, (b) y-velocity.

5.3 Natural Convection in a Square Cavity

The thermal cavity benchmark introduced by De Vahl Davis [12, 11] is used here to demonstrate an application with buoyancy-driven flow, and the use of surface output to calculate the wall heat transfer. Figure 5.13 shows the computational domain, mesh, and sets used for the differentially heated cavity. In this example, a series of 5 meshes are provided for this example as shown in Table 5.3.

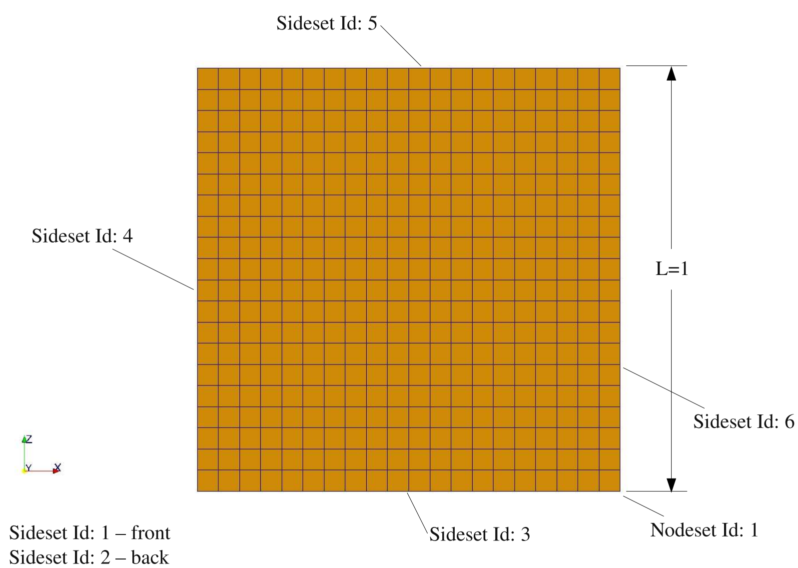


Figure 5.13: 20 x 20 thermal cavity mesh.

Mesh	Mesh Size	h
A	20×20	5.000E-2
B	40×40	2.500E-2
D	80×80	1.250E-2
D	160×160	6.250E-3
E	320×320	3.125E-3

Table 5.3: Meshes used for the De Vahl Davis benchmark problem

The non-dimensional governing equations for time-dependent thermal convection (in vector form) are the incompressible Navier-Stokes equations, conservation of mass, and the energy equation in terms of temperature:

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\nabla P + Pr \nabla^2 \mathbf{v} + Ra Pr \hat{k} \theta, \tag{5.2}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{5.3}$$

and

$$\frac{\partial \theta}{\partial t} + \mathbf{v} \cdot \nabla \theta = \nabla^2 \theta, \tag{5.4}$$

where $\mathbf{v}$, P and θ are the velocity, the deviation from hydrostatic pressure, and temperature respectively, and $\hat{k}$ the unit vector in the z-direction. These non-dimensional equations were obtained using the characteristic length L, velocity $V = \alpha/L$, time scale $\tau = L^2/\alpha$, and pressure $\tilde{P} = \rho V^2$ as described in De Vahl Davis [11]. Here, ρ is the mass density, g the gravitational acceleration, $\alpha = k/\rho C_p$ is the thermal diffusivity, and ν is the kinematic viscosity. The Prandtl number is $Pr = \nu/\alpha$ and fixed at $Pr = 0.71$. The Rayleigh number is

$$Ra = \frac{g\beta(T_h - T_c)L^3}{\nu\alpha}, \tag{5.5}$$

where $T_h - T_c$ is the temperature difference between the hot and cold walls, and β the coefficient of thermal expansion. The non-dimensional temperature is defined in terms of the wall temperature difference

$$\theta = \frac{T - T_c}{T_h - T_c}, \tag{5.6}$$

where T_h is the prescribed temperature of the hot wall, and T_c is the temperature of the cold wall.

The boundary conditions for this problem consist of no-slip and no-penetration walls with the top and bottom walls insulated. The left wall is held at hot temperature, and the right wall at the cold temperature corresponding to $\theta = 1$ along $x = 0$ and $\theta = 0$ along $x = 1$. The initial conditions are prescribed with $\mathbf{v} = \mathbf{0}$ and $T = (T_h + T_c)/2$ which corresponds to $\theta(\mathbf{x}, 0) = 1/2$. The control file for the $Ra = 10^4$ case using the 80×80 grid is shown in Figure 5.14. Here, the hydrostatic pressure is prescribed at a single node (using a nodeset) at the lower right-hand corner of the differentially-heated cavity.

A series of 4 sample calculations are presented below for $10^3 \leq Ra \leq 10^6$ using meshes B – E. In each calculation, the total duration of the calculation is 2.0 time units which corresponds to twice the diffusional time-scale for the differentially-heated cavity. This is sufficient for the flow to establish steady-state conditions. As illustrated in the control file, a backward-Euler time integrator is selected with `CFLmax` $= 40$. An initial time step `deltat` $= 1.0e - 3$ is used to permit a smooth startup during the early time period where heat conduction dominates the thermal-convective process.

Time-integration is carried out until an essentially steady-state condition results. This is easily monitored in terms of time-history data for the integrated wall heat transfer rate, velocity, temperature and kinetic energy. The use of the **histvar** – **end** block in the control file activates time-history date to monitor the the velocity and temperature at the elements at the mid-side of the vertical walls, and to output the integrated heat transfer rate on the heated wall. Figure 5.15(a) shows the variation in the Nusselt number along the vertical heated wall for the Rayleigh numbers considered here. Figure 5.15(b) shows the time-history of the kinetic energy. From this plot, it is clear that an asymptotic steady-state flow is achieved by approximately 1 time unit. This was confirmed by checking the velocity and temperature time-histories. Figure 5.15(c) and (d) show the x- and y-velocity profiles along the vertical and horizontal centerlines of the cavity respectively. Figure 5.16 shows the temperature distribution for the four Rayleigh numbers.

Pointwise comparison data is presented in Table 5.4 using the data obtained by Richardson extrapolation by De Vahl Davis [11]. Here, the minimum and maximum velocities are computed along the horizontal and vertical centerlines of the cavity.

```
title
Ra=1.0e+3, Pr=0.71 De Vahl Davis benchmark

cc_navierstokes

  nsteps  2000
  deltat  1.0e-3
  term    2.0

  time_integration
    type    fixed_cfl
    CFLinit 1.0
    CFLmax  40.0
    dtmax   0.25
    dtscale 1.025
    thetaa  1.0
    thetak  1.0
    thetaf  1.0
  end

  # Output options
  plti 100

  # Energy equation
  energy temperature

  # Material model setup
  material
    id  1
    rho 1.0
    Cp  1.0
    mu  0.71
    k11 1.0
    beta 7.1e+2
    Tref 0.5
  end

  materialset
    id 10
    material 1
    block 1
  end

  plotvar
    elem density
    elem vel
    elem procid
    elem div
    elem temp
    node lambda
    node pressure
    node vel
    node vorticity
    node temp
    side 4 heatflux
  end

  histvar
    elem 40  vel
    elem 40  temp
    elem 6359 vel
    elem 6359 temp
    side 4 heatflow
  end

  # Simple IC's
  initial
    velx 0.0
    vely 0.0
    velz 0.0
    temp 0.5
  end

  # Boussinesq body force
  boussinesqforce
    gx  0.0
    gy  0.0
    gz -1.0
  end

  velocity
    # Front wall
    vely sideset 1 -1 0.0
    # Back wall
    vely sideset 2 -1 0.0
    # Bottom wall
    velx sideset 3 -1 0.0
    vely sideset 3 -1 0.0
    velz sideset 3 -1 0.0
    # Left wall
    velx sideset 4 -1 0.0
    vely sideset 4 -1 0.0
    velz sideset 4 -1 0.0
    # Top wall
    velx sideset 5 -1 0.0
    vely sideset 5 -1 0.0
    velz sideset 5 -1 0.0
    # Right wall
    velx sideset 6 -1 0.0
    vely sideset 6 -1 0.0
    velz sideset 6 -1 0.0
  end

  temperature
    # Left Wall
    sideset 4 -1 1.0
    # Right Wall
    sideset 6 -1 0.0
  end

  hydrostat
    nodeset 1 -1 0.0
  end

  ppesolver
    type  AMG
    itmax 100
    itchk 1
    coarse_size 1000
    diagnostics off
    convergence off
    eps   1.0e-8
  end

  momentumsolver
    type  ILUFGMRES
    itmax 50
    itchk 2
    restart 45
    diagnostics off
    convergence off
    eps   1.0e-8
  end

end

exit
```

Figure 5.14: Control file for the $Ra = 10^3$ differentially heated cavity.

The mean Nusselt number is computed as

$$\overline{Nu} = \frac{1}{A} \int_{\Gamma} \nabla\theta \cdot \mathbf{n} \, d\Gamma \tag{5.7}$$

where A is the surface area. For this comparison, the `heatflow` time-history request results in the output of the integrated non-dimensional heat flow over the heated surface. For all computations, a z-dimension of $\Delta z = 0.0125$ was used with $L = 1$ resulting in an area $A = 0.0125$. In order to compute the mean Nusselt number, the `heatflow` output is scaled by $1/A$. The minimum and maximum Nusselt numbers pointwise correspond to the non-dimensional output requested with the `heatflux` plot variable output. Here, the minimum/maximum Nusselt numbers were extracted from the non-dimensional heat flux distribution along the heated wall. As can be seen, for the selection of meshes used here, the agreement with the De Vahl Davis benchmark data is quite good.

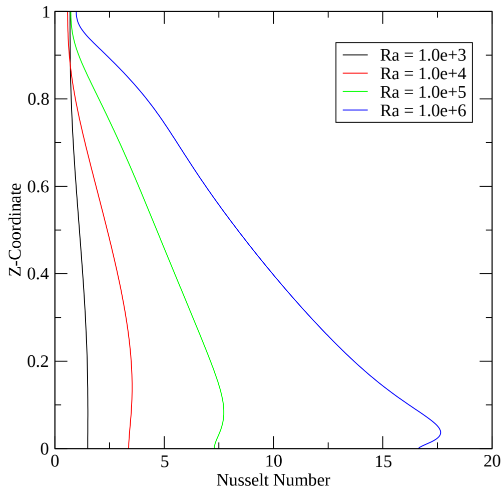
	Rayleigh Number (Ra)							
	10^3		10^4		10^5		10^6	
	Ref. [11]	80×80	Ref. [11]	160×160	Ref. [11]	320×320	Ref. [11]	320×320
$v_{x_{max}}$	3.659	3.648	16.178	16.179	34.73	34.79	64.63	64.91
$v_{z_{max}}$	3.697	3.692	19.617	19.574	68.59	68.62	219.36	220.33
$\overline{Nu}$	1.118	1.118	2.243	2.246	4.519	4.523	8.800	8.841
Nu_{min}	0.692	0.691	0.586	0.585	0.729	0.727	0.989	0.978
Nu_{max}	1.505	1.507	3.528	3.536	7.717	7.727	17.925	17.643

Table 5.4: Maximal velocities, mean and maximal Nusselt numbers compared with the extrapolated benchmark results obtained by De Vahl Davis [11].

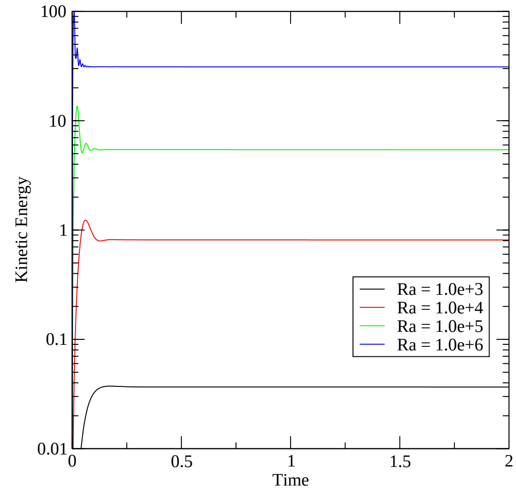
Example Problem Files:

Mesh File: dd_20x20.exo, dd_40x40.exo, dd_80x80.exo, dd_160x160.exo, dd_320x320.exo

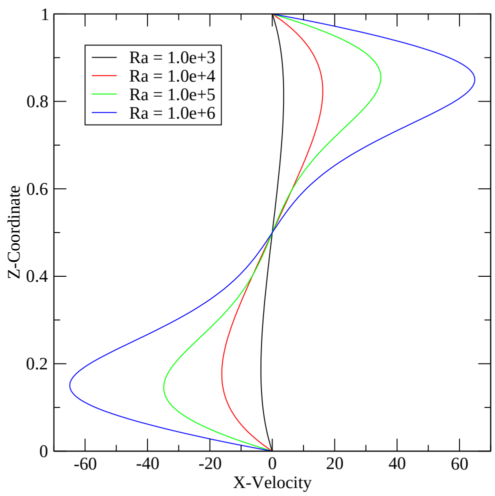
Control File: dd_20x20_Ra1e3.cntl, dd_40x40_Ra1e3.cntl, dd_80x80_Ra1e3.cntl, dd_160x160_Ra1e4.cntl, dd_320x320_Ra1e5.cntl, dd_320x320_Ra1e6.cntl



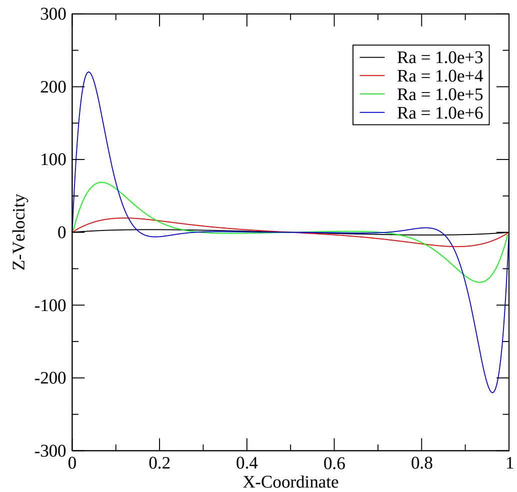
(a)



(b)



(c)



(d)

Figure 5.15: (a) Nusselt number profile along vertical heated wall, (b) global kinetic energy time history, (c) x velocity along vertical centerline, and (d) z-velocity along the horizontal centerline for $Ra = 10^3, 10^4, 10^5, 10^6$.

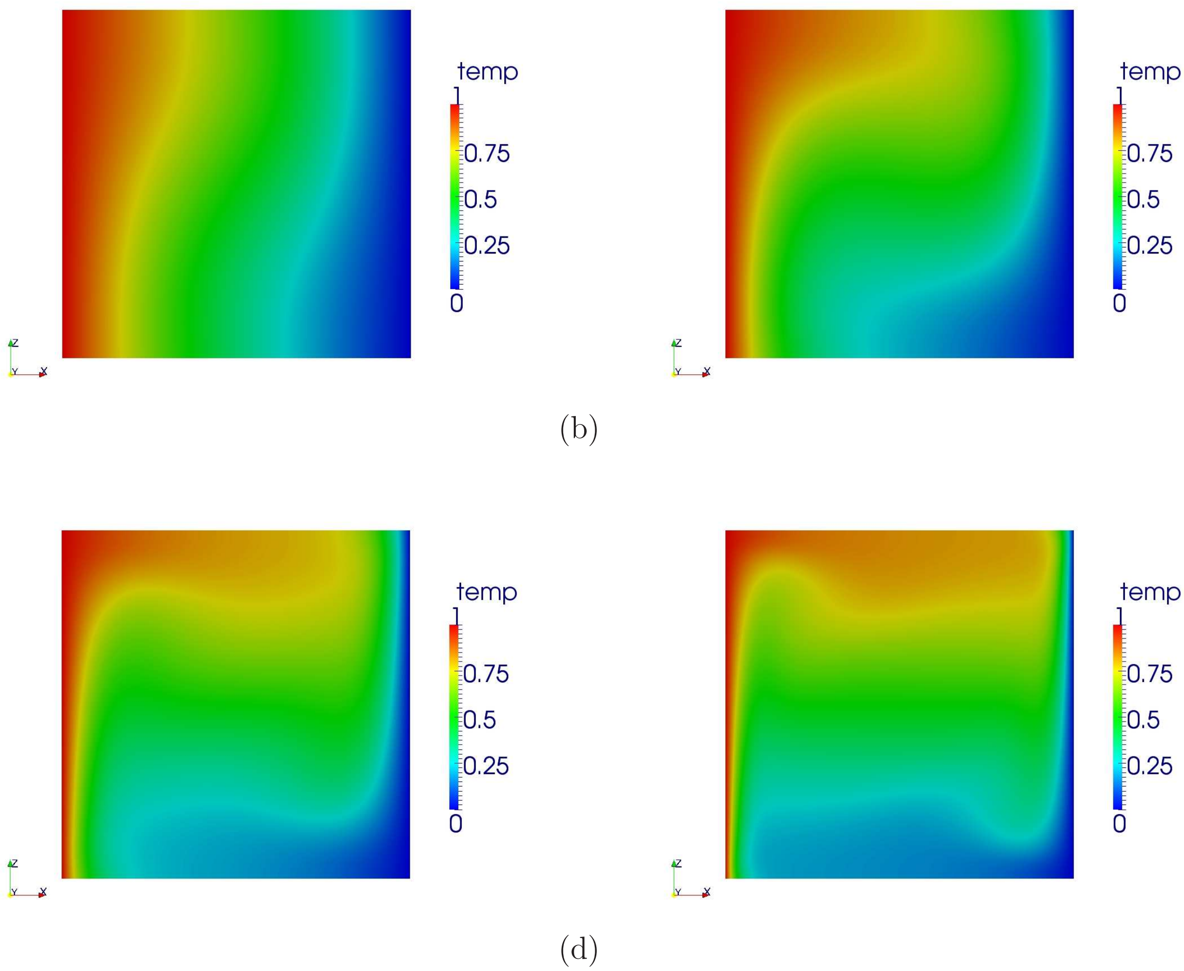


Figure 5.16: Temperature distribution at $t = 2$ time units for (a) $Ra = 10^3$ using mesh B, (b) $Ra = 10^4$ using mesh C, (c) $Ra = 10^5$ using mesh D, and (d) $Ra = 10^6$ using mesh E.

5.4 Ahmed's Body

The Ahmed body example is an exterior flow problem based on the experimental investigation of the flow structures and drag characteristics for ground vehicles performed by Ahmed, et al. [1]. In this experiment, a bluff body was used with a slant-back emulating the design of car bodies of the era, e.g., the Volkswagen rabbit.

In a time-averaged sense, the flow around a Ahmed's body exhibits massive separation and complex three-dimensional large-scale features that propagate in the downstream flow direction. For this example, we consider the so-called "high-drag" 30^o slant-back configuration reported by Ahmed, et al. [1]. The computational domain used for this problem is shown in Figure 5.17. The overall dimensions of the Ahmed body are: length of 1.044 m, width of 0.389 m, and height of $0.288m$, with projected area of 0.112 m^2 in the primary flow direction. For this example, the support legs for the body were omitted for simplicity, and because the reported drag coefficients exclude the drag on the supports.

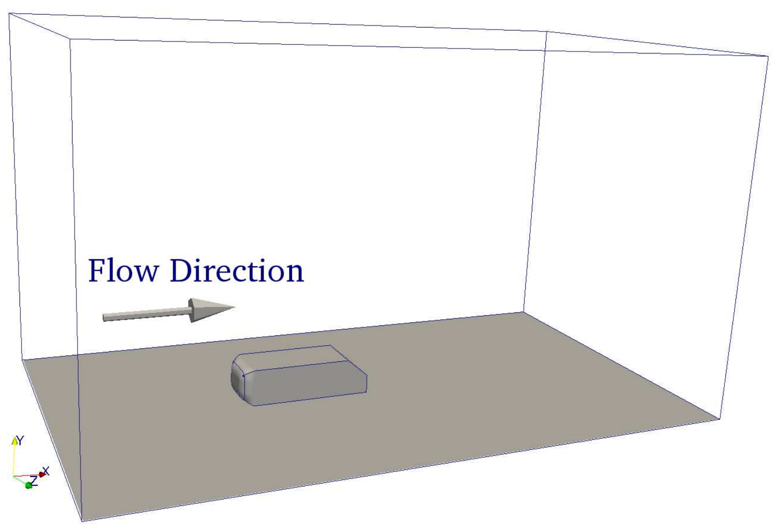


Figure 5.17: Flow past Ahmed's body with a 30^o slant-back at $Re = 4.29 \times 10^6$.

The Reynold number for this problem is $Re = 4.29 \times 10^6$. based on the free-stream velocity $V^\infty = 60m/s$, the characteristic body length ($L = 1.044\ m$), density and molecular viscosity for air at standard temperature and pressure, i.e., $\rho = 1.2951\ kg/m^3$, $\mu = 1.7355 \times 10^{-5}\ kg/m/s$. Due to the relatively high Reynolds number, the Spalart-Allmaras model is used for this calculation with a backward-Euler marching strategy intended to find a steady-state flow solution.

The computational domain is shown in Figure 5.18, along with the definition of the necessary sets for the prescription of boundary conditions. No-slip/no-penetration boundary conditions are applied on the "ground" (sideset Id 5), and Ahmed's body (sideset Id 7). Tow-tank conditions are prescribed at the front/back and upper flow domain boundaries. Inlet conditions prescribe a constant x-velocity, i.e., $\mathbf{v} = (60, 0, 0)$, and homogeneous Neumann conditions for the Spalart-Allmaras transport equation are used at the outflow boundary.

For the Spalart-Allmaras model, both the normal distance and the Spalart-Allmaras variable ($\tilde{\nu}$) are prescribed to be zero along the ground and on the Ahmed body. This is a relatively coarse mesh resulting in an average $y^+ \approx 80$ on the ground and $y^+ \approx 55$ on the body surface. Thus, the boundary layer on the Ahmed body is somewhat under resolved. In addition to the distance and $\tilde{\nu}$ boundary conditions, initial conditions for $\tilde{\nu} \approx 5\nu$ are prescribed, where ν is the kinematic molecular viscosity.

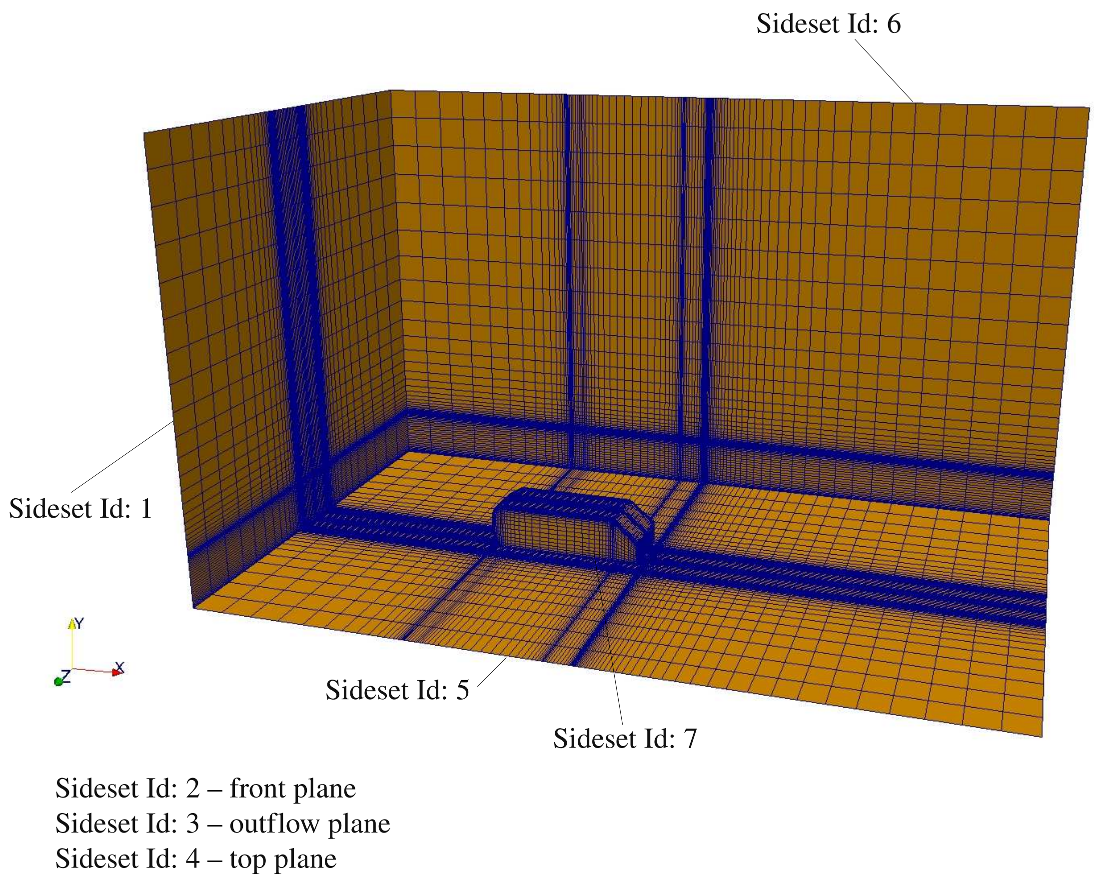


Figure 5.18: Mesh for Ahmed's body.

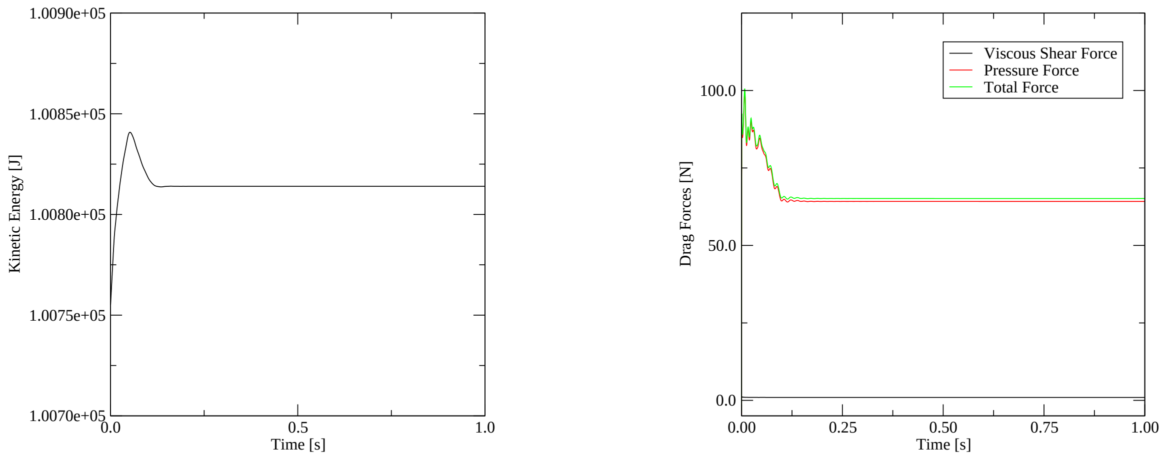
The control file for this example problem is shown in Figure 5.19. For this calculation, we choose to execute Hydra in parallel and use distributed plot files are selected using `filetype distributed` resulting in a single sub-domain plot file being written for each processor used in the computation. In order to monitor the $y+$, surface field output is requested for sidesets 5 and 7 corresponding to the ground and body surface. In the **histvar** – **end** block, the total force, viscous shear and pressure forces integrated over Ahmed's body are requested as well. Note too the additional boundary condition specifications in the **distance** – **end** and **turbnu** – **end** keyword blocks in the control file.

For this computation, we monitor the time-history of the global kinetic energy and the forces on the surface of Ahmed's body. As shown in Figure 5.20, the kinetic energy and forces achieve essentially steady-state conditional by $t = 0.4\ s$. Using the asymptotic values of the viscous shear,

```
title
Ahmed Body - SA Model

cc_navierstokes

  nsteps 2000
  deltat 1.0e-4
  term   1.0

  time_integration
    type    fixed_cfl
    CFLinit 1.0
    CFLmax  20.0
    dtmax   0.2
    dtscale 1.025
    thetaa  1.0
    thetak  1.0
    thetaf  1.0
  end

  # Output options
  pltype exodusii
  filetype distributed
  plti   100
  ttyi   10
  dump   2000

  # Turbulence model
  tmodel spalart_allmaras

  # Material model setup
  material
    id  1
    rho 1.2951
    mu  1.7355e-5
  end

  materialset
    id 10
    material 1
    block 5
  end

  # Set definitions
  # Sideset 1 - Inlet
  # Sideset 2 - Front
  # Sideset 3 - Outflow
  # Sideset 4 - Top
  # Sideset 5 - Ground
  # Sideset 6 - Back
  # Sideset 7 - Ahmed body

  plotvar
    elem vel
    elem turbnu
    node vel
    node pressure
    node dist
    node vorticity
    node helicity
    side 5 yplus
    side 7 yplus
  end

  histvar
    side 7 force
    side 7 viscforce
    side 7 pressforce
  endvar
```

```
  # Simple IC's
  initial
    velx 60.0
    vely 0.0
    velz 0.0
    turbnu 7.0e-5
  end

  pressure
    sideset 3 -1 0.0
  end

  distance
    sideset 5 -1 0.0
    sideset 7 -1 0.0
  end

  turbnu
    sideset 5 -1 0.0
    sideset 7 -1 0.0
  end

  velocity
    velx sideset 1 -1 60.0
    vely sideset 1 -1  0.0
    velz sideset 1 -1  0.0
    velx sideset 2 -1 60.0
    velz sideset 2 -1  0.0
    velx sideset 4 -1 60.0
    vely sideset 4 -1  0.0
    velx sideset 5 -1  0.0
    vely sideset 5 -1  0.0
    velz sideset 5 -1  0.0
    velx sideset 6 -1 60.0
    velz sideset 6 -1  0.0
    velx sideset 7 -1  0.0
    vely sideset 7 -1  0.0
    velz sideset 7 -1  0.0
  end

  ppesolver
    type  AMG
    itmax 400
    itchk 1
    solver cg
    smoother ICC
    coarse_size 500
    eps   1.0e-5
  end

  momentumsolver
    type  ILUFGMRES
    itmax 50
    itchk 2
    restart 20
    eps   1.0e-5
  end

  transportsolver
    type  ILUFGMRES
    itmax 50
    itchk 2
    restart 20
    eps   1.0e-5
  end

end

exit
```

Figure 5.19: Control file for the Ahmed body.



(a) (b)

Figure 5.20: Time-history plots showing (a) global kinetic energy, and (b) the viscous shear, pressure and total force on Ahmed's body.

pressure and total forces, the drag coefficient is computed using the definition in Ahmed, et al. [1].

$$C_w = \frac{F}{\frac{1}{2}\rho V\infty^2 A_p} \tag{5.8}$$

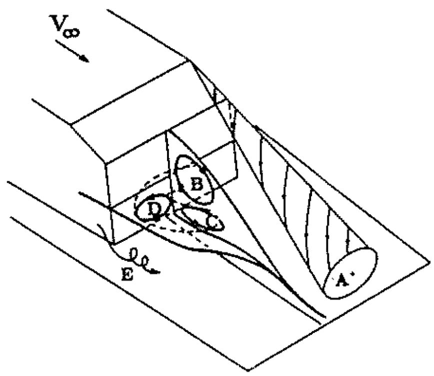
where $A_p = 0.1120\ m^2$ is the projected area of the body in the streamwise direction.

Using this definition for the drag coefficient, the calculated results are compared with the experimental results reported by Ahmed, et al. [1] (Table 1, and Table 2). Here, the effect of the "tare drag" were removed from the experimental results. The results for the experimental and computational drag coefficients are shown in Table 5.5. Based on the experimental results, it's clear that the total drag is dominated by the pressure or so-called form drag. The computed pressure drag coefficient is within 1% of the experimental results reinforcing this observation. However, we note that the shear drag coefficient is quite low for the computation which is a consequence of the relatively large $y+$ on the Ahmed body and the relatively under resolved boundary layer. Typically $3 \leq y^+ \leq 5$ is required for the Spalart-Allmaras model, however, a coarse mesh with $y^+ \approx 55$ on the Ahmed body was intentionally used for the purposes of presenting the relatively quick-running example problem. The consequence is that the total drag is only within 14.5% of the experimental result.

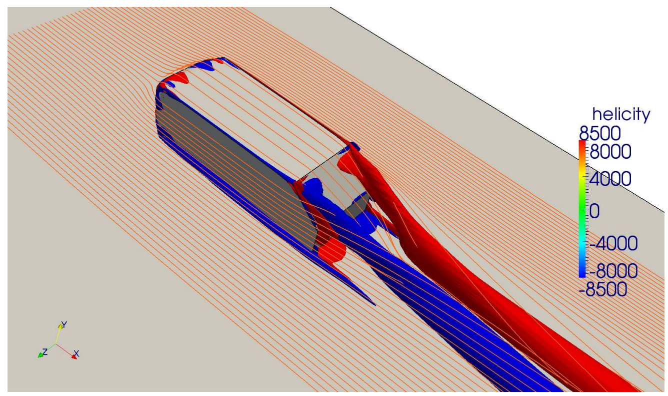
	Total Drag	Pressure Drag	Viscous Drag
Ahmed, et al. [1]	0.378	0.321	0.047
Spallart-Allmaras Model	0.323	0.318	0.005

Table 5.5: Experimental drag coefficients reported by Ahmed, et al. [1], and current results using the Spalart-Allmaras model.

Figure 5.21(a) shows a schematic of the horseshoe vortex system in the wake of the Ahmed body as suggested by Ahmed, et al. [1]. Here, a longitudinal vortex structure emanates from the juncture of the slant-back and horizontal surface of the body surface, with significant counter-rotating circulation immediately behind the body. In Figure 5.21(b), a very similar structure is observed in the streamlines and isosurfaces of helicity. Helicity is a good indicator of coherent longitudinal vortical structures and suggests a correlation between the flow direction and the primary rotation of the vortices. Figure 5.21(b) shows the pair of counter-rotating helical structures generated at the juncture between horizontal and slant-back surfaces. The particle traces indicate the swirl generated in the wake of Ahmed's body consistent with experimental observations.



(a)



(b)

Figure 5.21: Three-dimensional wake pattern behind Ahmed's body with slanted rear surface: (a) reproduced from Ahmed, et al. [1], snapshot showing helicity ($\mathbf{v}\cdot\omega$) for the Spalart-Allmaras model.

Example Problem Files:
Mesh File: ahmed.exo
Control File: ahmed_sa.cntl

Appendix A

ASCII Mesh Format

The ASCII mesh format is intended to provide a simplified, machine independent mesh specification that permits finite element codes to interface with a number of public-domain and commercial mesh generation tools. For this reason, the ASCII mesh format has been intentionally simplified to permit easy adaptation of existing neutral file formats produced by commercial mesh generators.

The ASCII mesh file contains an 80-character title line, the spatial coordinates of the nodes, element connectivity, and all node and side sets. The mesh title line is limited to 80 characters, must be the first line in the mesh file. Lines in the mesh file may be commented out by using a #, ∗, $ followed by a blank. Because the data in the mesh file is usually generated by an automatic mesh generator, the data in this file is only partially format-free in style. However, all input in the mesh file is case insensitive.

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

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

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

A.1 Mesh Title Line

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

A.2 Header Block

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

ASCII Mesh File Header Block

Keyword	Variable	Meaning
Nnp	Nnp	Number of nodes, Nnp.
Nel	Nel	Number of elements, Nel.
Nel_tri3	$Nelt3$	Specify the number of 3-node tri elements, $Nelt3$.
Nel_poly2d	$Nelp2d$	Specify the number of 2-D polygonal elements, $Nelp2d$.
Nel_quad4	$Nelq4$	Specify the number of 4-node quad elements, $Nelq4$.
Nel_hexh8	$Nelh8$	Specify the number of 8-node hex elements, $Nelh8$.
Nel_poly3d	$Nelp3d$	Specify the number of 3-D polyhedral elements, $Nelp3d$.
Nel_pyr5	$Nelp5$	Specify the number of 5-node pyramid elements, $Nelp5$.
Nel_tet4	$Nelt4$	Specify the number of 4-node tet elements, $Nelt4$.
Nel_wedge6	$Nelw6$	Specify the number of 6-node wedge elements, $Nelw6$.
Ndim	$Ndim$	Specify the number of dimensions, $Ndim$.
Nmat	$Nmat$	Specify the number of materials, $Nmat$.
Nnd_sets	Nnd_sets	Specify the number of node sets, Nnd_sets.
Nsd_sets	Nsd_sets	Specify the number of side sets, Nsd_sets.
end		Terminate the header block

```
The 80-character title line comes first in the mesh file.
#
# This is the header block
#
Nnp       1852
Nel       1760
Nel_quad4 1760
Ndim         2
Nmat         1
Nnd_sets     3
Nsd_sets     3
end
#
```

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

A.3 Element Connectivity

The ordinal node numbers and block Id's associated with Nel elements are required in this section of the mesh file. For 1-D calculations ($Ndim = 1$), the last 6 node number are ignored if they are present in the connectivity. For 2-D calculations ($Ndim = 2$), the last 4 node numbers are ignored if they are present in the connectivity. The table below shows the format specifications for the element connectivity. The standard fixed topology elements may be specified with the format shown below.

Fixed Topology Element Connectivity

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

The 2-D polygonal (poly2d) and 3-D polyhedral elements require a slightly different format. The poly2d elements are specified using the following format. Here, a total of $Nnpf$ ordinal node Id's are required for each poly2d element.

2-D Polygonal Element Connectivity

Columns	Format	Description
1–8	I8	ordinal element Id
9–13	I5	element block Id
14–21	I8	number of nodes connected to this element ($Nnpf$)
22–29	I8	ordinal node #1
30–37	I8	ordinal node #2
38–45	I8	ordinal node #3
46–53	I8	ordinal node #4
...		...
	I8	ordinal node #$Nnpf$

The 3-D polygonal (poly3d) element is defined in terms of a list of unique polygonal faces in the mesh, and a face connectivity for each element. The polygonal faces are defined using the same input format that the poly2d elements use as shown below. The number of faces to be read precedes the definition of the polygon faces as shown in Figure A.2. The face connectivity is immediately followed by connectivity that relates the ordinal face Id's to ordinal element Id's.

```
#
# 3-D Polygonal face definition
Nfaces 3
#
# Ordinal Block
# FaceId   Id    Nnpf      n1      n2      n3     n4     ...
       1   10       3     120     500     271
       1   10       4     590       3      21    522
       ...
# Ordinal Block
# Element  Id    Nfpe      f1      f2      f3     f4     ...
       1   10       3     120     500     271
       ...
```

Figure A.2: Example 3-D polyhedral element definition.

Polygonal Face Connectivity

Columns	**Format**	**Description**
1–8	I8	ordinal face Id
9–13	I5	number of nodes connected to this face ($Nnpf$)
14–21	I8	ordinal node #1
22–29	I8	ordinal node #2
30–37	I8	ordinal node #3
38–45	I8	ordinal node #4
...		...
	I8	ordinal node #$Nnpf$

3-D Polyhedral Element Connectivity

Columns	**Format**	**Description**
1–8	I8	ordinal element Id
9–13	I5	number of faces connected to this face ($Nfpe$)
14–21	I8	ordinal face #1
22–29	I8	ordinal face #2
30–37	I8	ordinal face #3
38–45	I8	ordinal face #4
...		...
	I8	ordinal face #$Nfpe$

A.4 Nodal Coordinates

Nnp nodal coordinates are required in this section of the mesh file. For 1-D analyses ($Ndim = 1$), the y and z-coordinates are ignored. For 2-D analyses ($Ndim = 2$), the z-coordinate is ignored in the mesh file. The format specifications for the nodal coordinates are shown in the table below.

Nodal Coordinates

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

A.5 Node Sets

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

Node Sets - Part 1

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

Node Sets - Part 2

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

Node Sets - Part 3

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

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

Figure A.3: Example node set section of the ASCII mesh file.

A.6 Side Sets

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

Side Sets - Part 1

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

Side Sets - Part 2

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

Side Sets - Part 3

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

The canonical local node numbering scheme is shown with the side numbering in Figure A.4. The following tables show the local node ordering corresponding to each side number for the 2-D quadrilateral and 3-D hexahedral element. A sample side set section of the mesh file is shown in Figure A.5. Note that for each side set segment, the segment lists consist of the element number and the associated side number based upon the canonical local node ordering.

Side Numbers - 2-D Quadrilateral

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

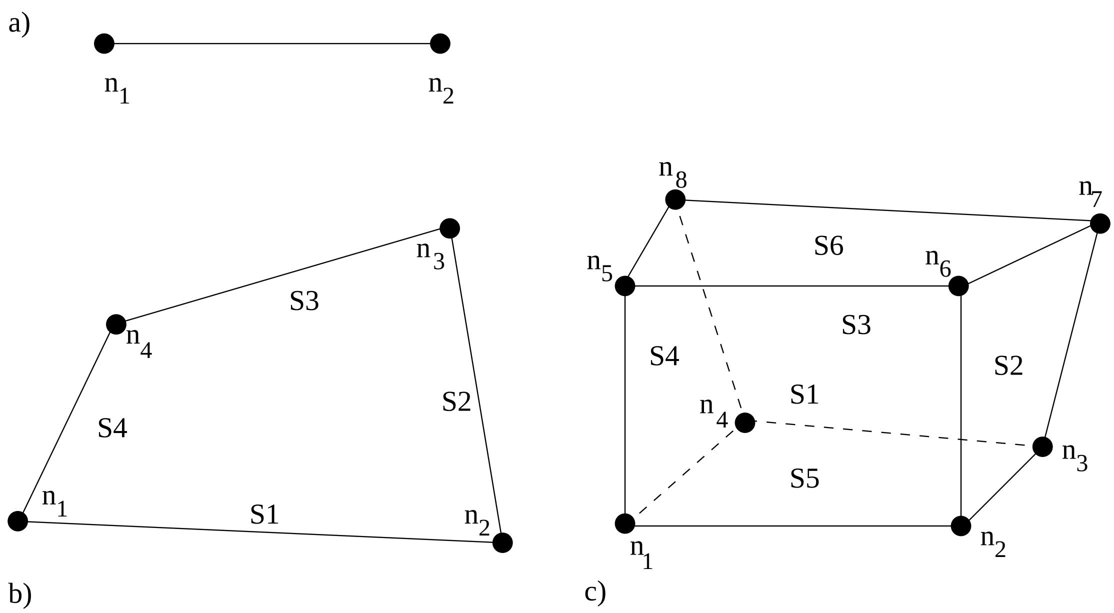


Figure A.4: Canonical node and side numbering for the a) 1-D linear element, b) 2-D quadrilateral element, and c) the 3-D hexahedral element.

Side Numbers - 3-D Hexahedral Element

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

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

Figure A.5: Example side set section of the ASCII mesh file.

A.7 Sample ASCII Mesh File

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

```
Sample ASCII mesh file
#
# $ * denote comments
#
Nnp       1681
Nel       1600
Nel_quad4 1600
Ndim         2
Nmat         1
Nnd_sets     1
Nsd_sets     6
end
#
# ===== Element Connectivity =====
#
            1       1       3     161     160
            1       3       4     162     161
     ...
            1    1681      81      42      83
#
# ===== Nodal Coordinates =====
#
       1      5.0000000000000e-01-5.0000000000000e-01
       2      5.0000000000000e-01 5.0000000000000e-01
     ...
    1681     -4.7499999403954e-01 4.7499999403954e-01
#
# 1 Node-sets
         1
# Node Set  Number of Nodes
        10        41
#
# Node Set Number : 10
# No. of Nodes    : 41
#
         1       122
         2       123
       ...
        41         1
```

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

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

Bibliography

[1] S. R. Ahmed, G. Ramm, and G. Faltin. Some salient features of the time-averaged ground vehicle wake. In *SAE Technical Paper Series, International Congress & Exposition*, number 840300, Detroit, Michigan, March 1984.

[2] H. T. Ahn, M. Shashkov, and M. A. Christon. The moment-of-fluid method in action. *Communications in Numerical Methods in Engineering*, 25(10), 2008.

[3] John B. Bell, Philip Colella, and Harland M. Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85:257–283, 1989.

[4] J. Blasco, R. Codina, and A. Huerta. A fractional-step method for the incompressible navier-stokes equations related to a predictor-multicorrector algorithm. *International Journal for Numerical Methods in Fluids*, 28:1391–1419, 1998.

[5] M. A. Christon and R. S. Patil. A finite element projection method for low-Mach number reacting flows. In K. J. Bathe, editor, *Third MIT Conference on Computational Fluid and Solid Mechanics*, pages 617–622, New York, June 2005. Elsevier.

[6] Mark A. Christon. Domain-based parallelism and the projection algorithm for transient, viscous incompressible flow. *in preparation for Computer Methods in Applied Mechanics and Engineering*, 1998.

[7] Mark A. Christon. The new incompressible flow capabilities in LS-DYNA. In *6th International LS-DYNA Users Conference 2000*, Dearborn, Michigan, April 2000. ETA.

[8] Mark A. Christon. Hydra-TH Theory Manual. Technical Report LA-UR 11-05387, Los Alamos National Laboratory, September 2011.

[9] Mark A. Christon and Daniel E. Carroll. An unstructured-grid, parallel, projection solver for computing low-speed flows. In S. N. Atluri and P. E. O'Donoghue, editors, *International Conference on Computational Engineering Science*, pages 845–850, Atlanta, Georgia, October 1998. Tech Science Press.

[10] Sharen J. Cummins and Murray Rudman. An sph projection method. *Journal of Computational Mechanics*, 152:584–607, 1999.

[11] G. de Vahl Davis. Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.

[12] G. de Vahl Davis and I. P. Jones. Natural convection in a square cavity: a comparison exercise. *International Journal for Numerical Methods in Fluids*, 3:227–248, 1983.

[13] Ercan Erturk and Bahityar Dursun. Numerical solutions of 2-d steady incompressible flow in a driven skewed cavity. *ZAMM - Journal of Applied Mathematics and Mechanics*, 87:377–392, 2007.

[14] U. Ghia, N. Ghia, and C. T. Shin. High-*re* solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1983.

[15] James Glimm, John W. Grove, X. L. Li, and D. C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM Journal for Scientific Computations*, 21:2240–2256, 2000.

[16] J.-L. Guermond and L. Quartapelle. A projection FEM for variable density incompressible flows. *Journal of Computational Physics*, 165:167–188, 2000.

[17] Mindy Lai, John B. Bell, and Phillip Colella. A projection method for combustion in the zero mach number limit. In *Eleventh AIAA Computational Fluid Dynamics Conference*, pages 776–783. AIAA, 1993.

[18] Mindy Fruchtman Lai. *A Projection Method for Reacting Flow in the Zero Mach Number Limit*. PhD thesis, University of California at Berkeley, 1993.

[19] Habib Najm, Peter S. Wyckoff, and Omar M. Knio. A semi-implicit numerical scheme for reacting flow. *Journal of Computational Physics*, (143):381–402, 1998.

[20] Richard B. Pember, Ann S. Almgren, William Y. Crutchfield, Louis H. Howell, John B. Bell, Phillip Colella, and Vincent E. Beckner. An embedded boundary method for the modeling of unsteady combustion in an industrial gas-fired furnace. Technical Report UCRL-JC-122177, Lawrence Livermore National Laboratory, October 1995.

[21] Elbridge G. Puckett, Ann S. Almgren, John B. Bell, Daniel L. Marcus, and William J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130:269–282, 1997.

[22] S. P. Schofield, M. A. Christon, V. Dyadechko, R. V. Garimella, R. B. Lowrie, and B. K. Swartz. Multi-material incompressible flow simulation using the moment-of-fluid method. *International Journal for Numerical Methods in Fluids*, 63:931–952, 2010. (Los Alamos National Laboratory LA-UR-09-00733).

[23] Mark Sussman, Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. An adaptive level set approach for two-phase flows. *Journal of Computational Physics*, 148:81–124, 1999.

[24] The Truchas Team. Truchas physics and algorithms. Technical Report LA-UR-03-0166, Los Alamos National Laboratory, 2003.

[25] Shuangzhang Tu and Shahrouz Aliabadi. Development of a hybrid finite volume/element solver for incompressible flows. *International Journal for Numerical Methods in Fluids*, 20:177–203, 2007.

[26] Shuangzhang Tu, Shahrouz Aliabadi, Reena Patel, and Marvin Watts. An implementation of the spalart-allmaras des model in an implicit unstructured hybrid finite volume/element solver for incompressible turbulent flow. *International Journal for Numerical Methods in Fluids*, 30:1051–1062, 2009.

[27] Matthew W. Williams, Doug Kothe, Deniece Korzekwa, and Phil Tubesing. Numerical methods for tracking interfaces with surface tension in 3-D mold filling processes. In *Proceedings of FEDSM '02*, Montreal, Canada, July 14-18 2002. 2002 ASME Fluid Engineering Division Summer Meeting, ASME.

Index