HIMAG

HyPerComp Incompressible MHD solver for Arbitrary Geometry

User's Guide

HIMAG 2019
HyPerComp, Inc.
2629 Townsgate Rd, Suite 105
Westlake Village, CA 91361
http://www.hypercomp.net
(T) 805-371-7500

Release 1.0
July 2019

Copyright Statement

Published in the U.S.A.

Trademarks-

CONTENTS

Preface to HIMAG 2019

The HIMAG 2019 User Guide is one of the three manuals that constitute the documentation for HIMAG (**H**yPerComp **I**ncompressible **M**HD solver for **A**rbitrary **G**eometry), the second one begin the HIMAG 2018 Developer's Guide and the last one is the HIMAG 2019 Technical Manual.

The HIMAG User's Guide tells you what you need to know to use HIMAG. This manual is divided into 7 Chapters along with two Appendices. Each section is described in detail. It covers everything from the installation of HIMAG to viewing the final result files.

The learning of any new applications is made more difficult than it ought to be because of the usage of the technical and application specific "buzz" words. Therefore, in order to aid the reader in learning HIMAG specific terms, this User Manual is provided.

The reader is advised to have a copy of this manual secured some place with easy access for everyday usage, in order to learn HIMAG.

The HIMAG team hopes that the reader would enjoy learning HIMAG 2019 as much as we enjoyed developing it.

Rupanshi C. Desai
Editor

Technical Support

If you encounter difficulties while using HIMAG, please first refer to the section(s) of the manual containing information on the commands you are trying to use or the type of problem you are trying to solve. The product documentation is available at the HyPerComp website. For support regarding HIMAG 2019 and other services related to HIMAG, please visit: ***www. hypercomp.net***

For further information, contact:

Rupanshi C. Desai (Editor, HIMAG 2019 User's Guide)
Phone: +1 (805) 371 7500 Ext. 106
Email: rchhabra@hypercomp.net

Peter Y. Huang (Editor, HIMAG 2019 Technical & Developer's Manual)
Phone: +1 (805) 371 7500 Ext. 104
Email: huang@hypercomp.net

For Graphical User Interface related information, contact:

Chris Rowell
Phone: +1 (805) 371 500 Ext. 103
Email: cmrowell@hypercomp.net

1 Introduction

1.1 Overview of HIMAG

The critical aspects of three dimensional magneto-hydrodynamics (MHD) flow fields in complex geometries (including conducting walls and multiple fluids in conduits,) are beyond popular engineering intuition and convenient analysis, thus requiring high fidelity numerical tools.

HIMAG (**H**yPerComp **I**ncompressible **M**HD solver for **A**rbitrary **G**eometry) was developed by HyPerComp, Inc., in collaboration with the Fusion Engineering group at the University of California at Los Angeles to model the flow of liquid metals in nuclear fusion reactor design. These flows are characterized by very high magnetic field strength, complex geometry and fluid-solid coupling via electric current and heat.

This unique code was developed to model multiple solid and liquid phase materials with arbitrary geometry. The inclusion of complex-geometry, electrically conducting walls and nozzles are essential since electric current closure paths are typically through these solid structures. In HIMAG, a second-order variable density projection method is used to simulate incompressible Navier–Stokes equations and the level set method is used to capture free surfaces. HIMAG is developed on unstructured grids, and can be run in parallel across multiple processors, and is thus able to efficiently solve large complex problems. HIMAG has already been validated for steady and unsteady single-fluid flow with/without MHD effects.

HIMAG is based on a second order accurate algorithm for both time and space. The code has been written for complex geometries and there is much flexibility in choosing a mesh: Hexahedral, Tetrahedral, Prismatic cells can be used. Electromagnetics and flow physics are solved in a coupled manner. Three different formulations for electromagnetics are now available: Φ, B and A-Φ.

HIMAG is a pioneer in the reliable computation of such flows among both commercial as well as research simulation software.

- HIMAG is a parallel, unstructured mesh based MHD solver.
- High accuracy at high Hartmann numbers is maintained even on non-orthogonal meshes
- HIMAG can model single-phase as well as two-phase (free surface) flows
- Graphical User Interfaces are provided for the full execution of HIMAG
- Extensive validation and benchmarking has been performed for canonical problems. Cases involving Ha > 1000 have never been demonstrated on non-rectangular meshes prior to HIMAG.

- An arbitrary set of conducting walls maybe specified.

HIMAG is well suited to studying closed channel flows encountered in blanket conditions.

1.2 Capabilities

Principal code features:

- Parallel iterative solver, based on well-developed base codes in CFD and CEM
- 2-D as well as 3-D flows can be simulated. A fully developed flow option is also present.
- Can use an arbitrary mesh structure, to resolve interfaces and complex geometry
- Facility to use Level Set methods for free surface capture
- Implicit methods to ease stiffness and time step constraints
- Choice of MHD models: ϕ formulation, B-formulation and A-ϕ formulation
- Three dimensional incompressible flow solver (2nd order accurate in space and time)
- Arbitrary mesh structure (hex/tetrahedral/prismatic cells)
- Well tested parallel code environment
- Electric potential as well as induced magnetic field formulations for MHD
- Point implicit scheme, solved in an iterative manner
- Multiple strategies to account for mesh skewness (no n-orthogonality)
- Ability to include multiple solid walls of different conductivity
- Modeling of axially periodic flows, fully developed flows and unsteady state flows
- Mass transfer, tritium transport and heat transfer
- Heat transfer, natural convection, temperature dependent properties can be modeled.

Besides the development of the solver, significant effort was expended in studying alternate MHD models (based on induced magnetic field and induced current) to overcome certain limitations of the Induction-less approach. These limitations tend to be chiefly numerical and become progressively more prominent at higher Hartmann numbers and magnetic field intensities. Much experience in the simulation of free surface flows on unstructured meshes has been gained. Parallel execution of the code for problems involving solid walls has been initiated and applied to several cases. High Hartmann number (up to 10,000) single phase MHD flows were studied. Very close to linear scaling of performance with respect to computer nodes, has been attained for large problems when HIMAG is run in the parallel mode.

HIMAG can complex geometries, such as curved surfaces, multiple solid obstacles, sharp corners, etc., by using arbitrary mesh topologies (tetrahedral, hexahedral

elements). It is able to economically resolve thin Hartmann layers and cracks in insulated walls using appropriate mesh-blocking strategies. So far, the code has shown great potential to overcome basic technological hurdles such as high Hartmann numbers and current leakage through extremely slender cracks in wall insulation.

The code is built on a robust iterative implicit solver, and can be run in parallel across a network of processors, and is primarily intended for use in PC based clusters running LINUX. With the low cost of building and operating these clusters, this is proving to be an inexpensive design strategy for researchers and engineers in this area. A network of 200 PC nodes at processor speeds of about 2 GHz across gigabit Ethernet is currently available for this research at UCLA. This will provide the ability to simulate very large scale problems such as encountered in real life, involving tens of millions of mesh points.

HIMAG is well suited to studying closed channel flows encountered in blanket conditions. Complex geometric features can be resolved by using the flexible mesh structure, while high Hartmann numbers require the parallelism for quick turnaround time.

1.3 Scope

This tutorial illustrates using HIMAG to setup and solve fluid flow and heat transfer problems. It provides step-by-step instructions on how to install and use HIMAG to create meaningful results. In this tutorial, you will learn to use HIMAG xyz to create grids, Ux2cgns to import grids from grid generating software like ICEM, HIMAG-Prep to set up the boundary conditions, and MHD2TEC for post processing. Other capabilities of parallel processing in HIMAG will also be discussed.

Most important aspects of this tutorial are as follows:

- Create a grid using HIMAG-XYZ or importing grid from a third-party grid generator
- Pre-Processing using HIMAG-Prep
- Running on big computer clusters using UX2PART and ColorMHD
- Running HIMAG
- Post-Processing results using MHD2TEC
- Viewing Results

This tutorial assumes that you have little to no experience with HIMAG-XYZ, Book, HIMAG-PREP, UX2PART, ColorMHD or MHD2TEC, and so each step will be explicitly described.

1.4 Document Structure

This document explains all the parts of HIMAG in a sequential manner.

Installation
Chapter 2 talks about the installation of HIMAG, the structure of the directories and the files required to run HIMAG.

Grid for HIMAG
Chapter 3 talks about creating the appropriate input file, the utility ***xyz*** used to create a grid, ***HIMAG-grid***, the GUI for ***xyz*** and steps thereafter for creating a grid for HIMAG. It also covers converting the grid to HIMAG's appropriate format which is done by the ***book*** command. This chapter will also include importing grids from other grid generating software like ICEM-CFD.

Pre-Processing
Chapter 4 covers gives the description on the utility called ***prep*** which is used to place boundary conditions on the mesh. Another utility which is involved in this chapter is ***mod_ugb*** which is used to place boundary conditions for magnetic vector potential.

Partitioning
Chapter 5 talks about ***Ux2Part*** which is used to distribute cells of a mesh to a given number of processors. It also covers the mesh partition for multiple processors which is done by ***ColorMHD***.

Running HIMAG
Chapter 6 gives a description of all the steps performed for the execution of HIMAG.

Post-Processing and Viewing the Result
Chapter 7 describes the process of post-processing through ***MHD2tec*** and eventually view the result in Tecplot. It will also talk about how to deal with all the other files generated by HIMAG as a part of the output.

2 Installation and Configuration

2.1 Startup Steps

In general for using HIMAG, the user needs to perform the following steps:

2.1.1 Get HIMAG Package

- Obtain the ***HIMAG2019.tar***
- Unpack the tar file and verify the directory structure
- Build HIMAG and create an executable
- Create a separate work directory and copy the executable in the work directory

2.1.2 Pre-Processing

- **Grid Generation**
  Either create a grid using **HIMAG-grid** or prepare grid input data file ***gridname*.xyz** without GUI or import a grid from a third party grid generation software, such as ICEM-CFD. If using ***xyz***, get a ***grid.ux*** file from ***book***.

- **Material Regions**
  Should material regions be involved, a ***mat.bin*** file is additionally created.

- **Boundary Conditions**
  Use ***prep*** to create individual patches from geometrical or other techniques and set the boundary conditions and material regions from **mat.bin**. Use ***mod_ugb*** to set the magnetic vector potential boundary conditions. Create a BC patch file using **HIMAG-Prep**.
  Save the BC file as ***gridname.ugb***.

- **Partitioning**
  Use ***ux2part*** and ***colormhd*** to partition the grid for multiple cores, ux2part results in a file ***gridname*.np.color**; while colormhd is to partition and results in a series of files named ***partition.mm.patch***, ***partition.mm.ux***, and ***partition.mm.ugb***.

2.1.3 Create the Input Files

- HIMAG ***hmg.input*** file will contain all the parameters a user can set

- An ***update.input*** file if any changes need to be made during the run
- ***User.dat*** file for Magnetic vector potential

2.1.4 Run HIMAG

- Use the HIMAG syntax along with the name of the input file and the grid name. Typically the run command is:
  - Single CPU: **himag -i hmg -g grid**
  - NP CPUs: **mpirun -np himag -i hmg**
  - CPU list from file: **mpirun -p4pg pgfile himag -i hmg**
    where *hmg.input* is the input file, *grid.ux* is the grid, and *pgfile* is a CPU list on which the code can be run when run in parallel, np is the number of CPUs to be used.

2.1.5 Post-Processing

- Understand the log files, save the restart files and remove other unnecessary files. The code writes diagnostic information to screen which can be redirected to an output file, and a confirmation file **op.cur** for all the parameters used either from input file or from default values if not in the input file.
- Solution output files are a serial of ***tec.mm.NSTEP.dat***, where mm = 0 ~ np, NSTEP is some time step.
- Use ***mhd2tec*** to create meaningful results from the tec.x.dat files.
- View in Tecplot or Paraview.

The files obtained in this process are:

Grid.xyz CheckXYZ.dat Univ.cemgrd	**Grid Generation Files**
Grid.ux Grid.ugb	**Grid Files**
Grid.np.color Partition.cpu.ux Partition.cpu.patch Partition.cpu.ugb	**Partition Files**

Hmg.input Updata.input User.dat intSol.dat intdata.dat	**Input Files**
Bc_ugb.dat Newugb.ugb	**Magnetic Potential BC Files**
Op.cur Logfile Debug.step.dat Bcfac.step.dat Intdata.dat Tec_var.dat Tec.cpu.step.dat	**Output Files**
Qrestart.cpu.unf Qrestart2.cpu.unf Qrest2d.unf	**Restart Files**
Grid.cpu.step.dat Grid.cpu.step.plt	**Result Data Files**
Patch[cpu].list Mat.bin/.lst/.dat	**Patch Files**

2.2 System Requirements

The following system components are required in order to install and successfully run HIMAG:

Required	Version
Operating System	Linux (Ubuntu, Fedora, Debian)
Fortran Compiler	gnu/gfortran 4.8 or later, ifort 12.0 or later
MPI Compiler	Openmpi or Mpich or Intel MPI
Qt	Qt3 only

To user HIMAG's full functionality, we do suggest the user to have some knowledge of these software components, particularly FORTRAN and to some extent C++.

Before proceeding with installing external and internal libraries, make sure that the version requirements are met. If any of the software or library requirements are not met, you or a system administrator should install the suggested (or later) version at a location accessible to you where you will be running HIMAG. Additional instructions on how to configure and use HIMAG are provided in the following section.

2.3 Installation of HIMAG 2019

First step towards understanding HIMAG as an application is to learn how to install this application. Installing HIMAG is easy, but the only thing which requires some work is to know which compiler your machine uses. If you already know that, then you can move on to step 2, but if you don't, start from step 1.

Step 1: To look for the compiler your machine uses,

```
mhd@machine$ dpkg --list | grep compiler
```

Step 2: Get HIMAG_2019 tar file and un-tar it. You will get six different directories, each containing an important part of HIMAG at the given location

```
mhd@machine$ cd ~/mhd/himag
```

2.4 Directory Structure

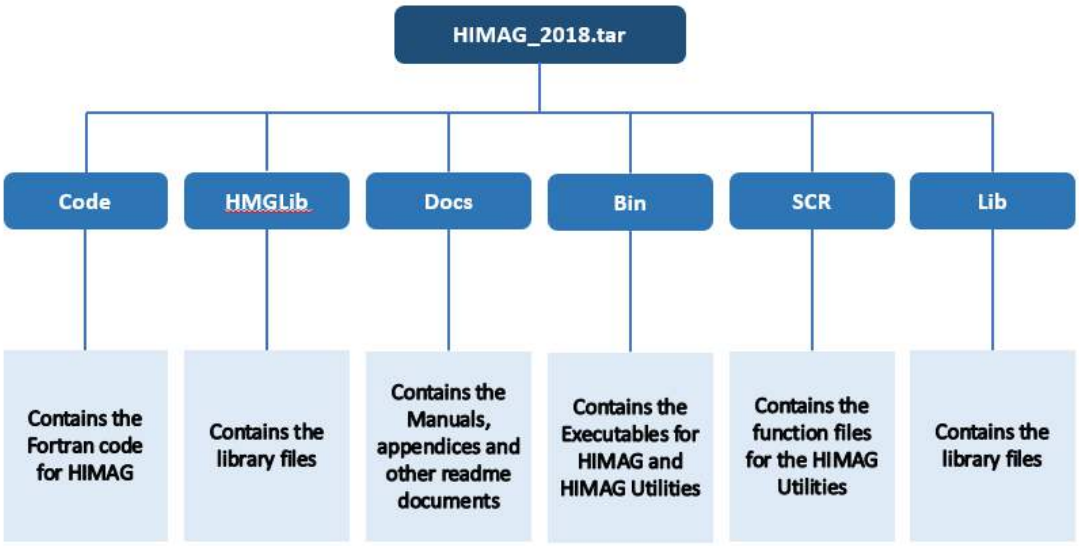


Figure 1: HIMAG Structure

CODE First directory is the Code, which as the name states, contains the FORTRAN code files of HIMAG. Any change made here will reflect throughout the code. Therefore, it is important not to make any changes here.

HMGLib HMGLib contains the library files for HIMAG and its utilities.

Docs This directory contains all the documents related to HIMAG needed to help the user run and execute HIMAG correctly and efficiently. This directory also contains the sample input files as well as the sample grid creation files to assist the user.

SCR This directory contains the function files for the HIMAG Utilities.

BIN BIN contains the executables created after compiling HIMAG and its utilities. More on this will be discussed in Chapter 6.

3 Grid Generation

3.1 Grid Requirements

The user should be aware of the following grid setup and construction requirements at the beginning of the problem setup. In general, no flow passage should be represented by less than 5 cells. Most cases will require many more cells to adequately model the flow. In regions of large gradients, as in the Hartmann layers or the mixing zones, the mesh should be fine enough to minimize the change in the flow variables from cell to cell.

For problems involving the flow of liquid metals in a magnetic field, the cross-sectional area of a duct (with infinitesimally thin walls) can be divided into 3 main regions. These regions are, the Hartmann layer (near walls perpendicular to the direction of the B-field), the side layer (parallel to the direction of the field) and the core. The thickness of the Hartmann layers is proportional to the reciprocal of the Hartmann number, whereas the thickness of the side layers is proportional to the reciprocal of the square root of the Hartmann number.

For high Hartmann number cases of interest to the fusion community, resolving the Hartmann layer can lead to a great disparity in the grid spacing in the core region, Hartmann layers, and the side layers.

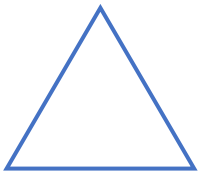
For using A-Φ formulation, the applied current or potential will show skin effect in the skin of the conductor. To witness this skin effect, the skin depth needs to be resolved adequately. A minimum of 5 cells in the skin depth is advisable.

Keeping these constraints in mind, good quality grids need to be generated by ensuring a smooth transition of grid spacing from one region to another. Sometimes, this may lead to a large mesh. Although accuracy increased with the larger meshes, the CPU and memory requirements to compute the solution and post-process the results also increases.

3.2 Types of Grids Supported

For 2D meshes, quadrilateral and triangular cells are acceptable and for 3D meshes, the types of cells supported in HIMAG are hexahedral, tetrahedral and prism/wedge.

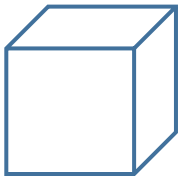
2D Cell Types



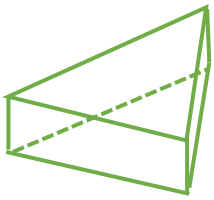
Triangle

Quadrilateral

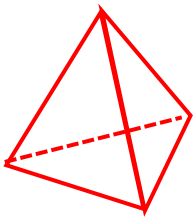
3D Cell Types



Hexahedron



Prism/ Wedge



Tetrahedron

The examples of the meshes using these cell types are shown later in this Chapter.

For creating the grids in HIMAG, the main criteria to follow are:

- Choosing the Appropriate Grid Type
- Grid Quality

Here, we have explained each criterion in detail to help the user get the exact grid supported by HIMAG.

3.2.1 Choosing the Appropriate Grid Type

As mentioned before, HIMAG can use 2D grid made using triangular or quadrilateral cells and tetrahedral, prism/wedge and hexahedral cells for 3D grid. The choice of grid will depend upon the application type but also on setup time that can be spent on

making the grid, the computational time that can be used to run HIMAG using that grid.

Sample grids acceptable in HIMAG and generated using some of the above-mentioned options are shown here. *Figure 2(a)* shows a prism grid, whereas *Figure 2(b)* and *1(c)* show a hexahedral grid for a half circular geometry and a polar grid, respectively.

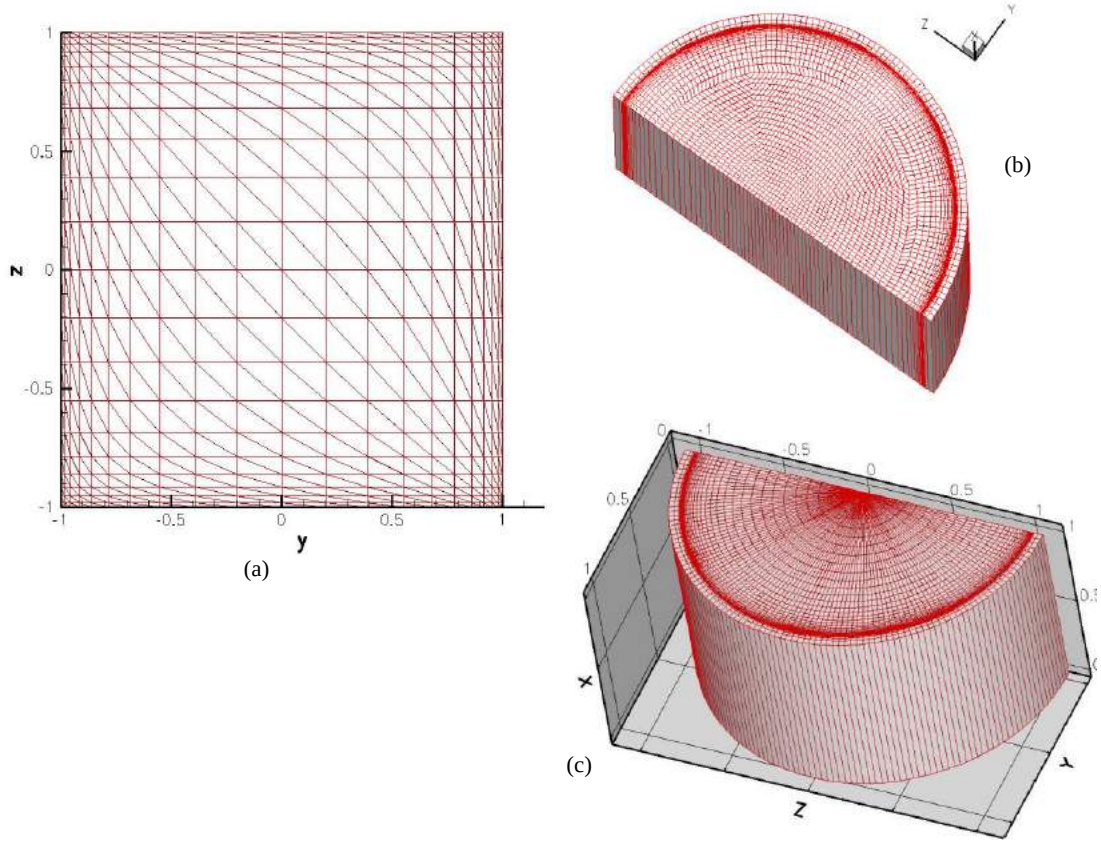


Figure 2: Sample meshes generated by HIMAG

3.2.2 Grid Quality

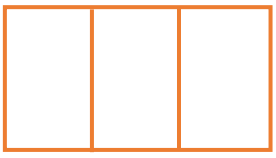
The quality of the mesh plays a significant role in the accuracy and stability of the numerical computation. Regardless of the type of mesh used in the domain, checking the quality of the mesh is essential.

Things which are to be kept in mind while making the grid –

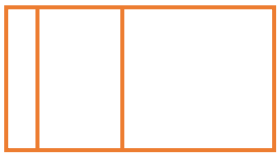
- Aspect ratio or grid stretching
- Orthogonal quality
- Skewness

The effect of resolution, smoothness and cell shape on the accuracy and stability of the solution process is dependent on the flow field being simulated. For example, skewed cells can be tolerated in benign flow regions, but can be very damaging in regions with strong gradients.

Aspect Ratio or Grid Stretching: The aspect ratio is the measure of stretching of a cell. It is calculated as the ratio of the maximum value to the minimum value of any of the following distances; the distance between the cell centroid and the face centroids; and the distances between the cell centroid and nodes. (ANSYS)

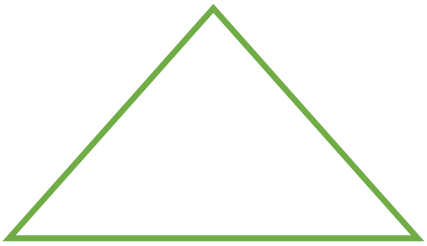


Uniform Grid

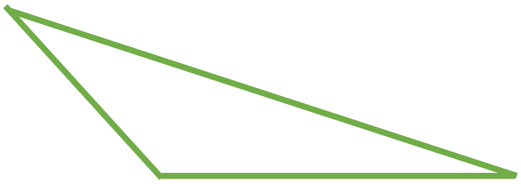


Stretched Grid

Skewness: Skewness is one of the primary quality measures for a grid. It determines how close to ideal shape (i.e. equilateral or equiangular) a cell is. Skewness is measured as the ratio of optimal cell size minus current cell size and optimal cell size.



Equilateral Triangle



Highly Skewed Triangle

Orthogonal Quality: The orthogonal quality for cells is computed using the face normal vector, the vector from the cell centroid to the centroid of each of the adjacent cells, and the vector from the cell centroid to each of the faces. (ANSYS)

The non-orthogonal feature is very useful in testing and benchmarking non-orthogonal corrections in the solver. For instance, one could generate an orthogonal, hexahedral mesh in a square duct and compute the flow field, electric potential and current under the fully developed flow assumptions (Shercliff solution or Hunt solution).

The user could then generate a non-orthogonal mesh and re-compute the solution with the non-orthogonal corrections. Figure 3 shows some non-orthogonal meshes generated by this feature. The user can define the degree of non-orthogonality by prescribing the two parameters (mgrd1 and mgrd2) which fixes the number of grid points orthogonal grid points in the each coordinate direction.

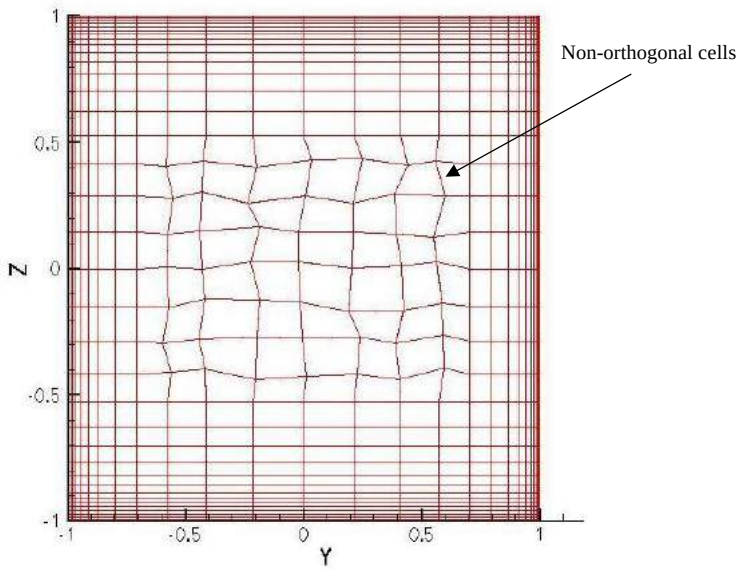


Figure 3: Non-orthogonal cells in a square domain

3.2.3 Diagnostics

In order to generate a good quality grid it is necessary to ensure a smooth transition in the grid spacing between adjacent segments along any coordinate direction. A trial and error method is necessary to select the stretching parameters in various segments to accomplish this goal. In order to simplify this process, the grid generation tool writes out a file called ***checkXYZ.dat***. This file has the location and grid-spacing for each grid point along each coordinate axis. The user can look at the grid spacing between various segments and make an intelligent guess as to the value of the stretching parameter to be used. This greatly facilitates the process of generating grids for geometrical configurations, requiring multiple segments.

3.3 Grid Generation Utility

With the intent of making HIMAG a user-friendly tool for researchers in the liquid-metal MHD community, HyPerComp Inc. has developed a grid generation capability with many useful features. Grid generation can be accomplished by a text-based input file or by the use of a GUI. In this section we will describe the salient features of the grid generation tool and also the GUI developed for problem set-up.

Grid generation for liquid-metal MHD: Square and circular ducts are among the most common configurations of interest to the fusion community. While the ultimate goal of HIMAG is to provide a detailed solution of liquid metal flows in complex manifolds, fully developed and/or developing flows in simple ducts can provide useful insight into the physics of such flows. Grid generation for flows in simple geometries is greatly simplified by the use of a tool developed by HyPerComp. This utility reads in a text input file prepared by the user (or generated by a GUI) and generates a single block to which the user can assign boundary conditions.

3.3.1 Text User Interface-Based Grid Generation

The process starts from a xyz file where the user is supposed to enter the grid dimensions and the type of grid required. The .xyz file is available for the users at in the **Code** directory of HIMAG. The user can copy this file from its location to their local folder. It is filled with dummy values and looks like:

```
nopt,   nout
  1        0
x-axis: Number of segments (nxsgm),   isymx,  xmin,   xmax
                                        1         0     0.0     100.0
#1:  nxcel,   segment-xmax
     100        100
     Stretch_option, stretching_factor
     0
y-axis: Number of segments (nysgm),   isymy,  ymin,   ymax
                                      2         1     -1.0     1.0
#1:  nycel,   segment-ymax
     5          -0.89
     Stretch_option, stretching_factor
     0
#2:  nycel,   segment-ymax
     60         0.89
     Stretch_option, stretching_factor
     3
     1.0
```

```
z-axis: Number of segments (nzsgm),   isymz,  zmin,   zmax
                                        2     1     -1.0       1.0

#1:  nzcel,   segment-zmax
     12          -0.75
     Stretch_option, stretching_factor
     1
     1.215
#2:  nzcel,   segment-zmax
     50          0.75
     Stretch_option, stretching_factor
     3
     1.0
```

All the variables in the xyz file are described in the help section which can be accessed by typing ***-h*** and is given below as well:

```
Options for the xyz data file:

  irot :    Rotate x-y-z or non-orthogonal grid
            [0]       -- no rotation
            [1]       -- yz → xy
            [2]       -- yz → zx (for prism use)
            [10]      -- xy non-ortho
            [11]      -- yz non-ortho
            [21]      -- rotate along x-axis
            [22]      -- rotate along y-axis
            [23]      -- rotate along z-axis

  mgrd1  :  number of ortho-nodes to keep (irot=10,11)
  mgrd2  :  number of ortho-nodes to keep (irot=10,11)

  amp    :  Amplifier
            0<amp<1 for non-ortho amplifier (=1: ortho)
            -180<amp<180 to rotate angle (irot=21,22,23)

  nopt   :  Grid Options
            [1]       -- Multi-segments hexahedral grid
            [2]       -- Multi-segments prism grid
            [3]       -- Tetrahedral grid
            [10]      -- Read from 1-D grid gridY.dat & gridZ.dat
            [20]      -- Read from 2-D (y-z) grid data file
            [21]      -- Read from 2-D (y-z) grid and modified
            [30]      -- Read from 3-D (x-y-z) grid data file
```

[31] -- Read from 3-D (x-y-z) grid + icel data file
[32] -- Import grid from CUBIT
[33] -- Import grid from TECPLOT
[34] -- Import grid from ANSYS CDB
[40] -- Hexahedral grid for a whole circular duct
[41] -- Hexahedral grid for a half circular duct
[50] -- Polar grid for a circular duct

nout : Output Grid Options
[0] -- Output normal grid (univ.cemgrid)
[1] -- Output 1-D data (gridY.dat & gridZ.dat)
[2] -- Output 2-D data (gridyz.dat)
[3] -- Output 3-D data (gridxyz.dat)
[4] -- Output grid to ASCII (mesh.dat)
[10] -- Output grid to tecplot (FNAME.tec.dat)

nxsgm: Number of Segments [Integer]

isym : [isymx, isymy, isymz]
[0] -- Non-Symmetric,
[1] -- Symmetry along the axis

x : [xmin, xmax] (real, real)
y : [ymin, ymax] (real, real)
z : [zmin, zmax] (real, real)

iopt : Stretch Option
[0] -- Uniform
[1] -- Robert-Left Stretch (fac): Left Smallest
[2] -- Robert-Right Stretch (fac): Right Smallest
[3] -- Robert-Both Sided Stretch (fac)
[4] -- Polynomial Stretch (dx1,dx2)
[5] -- Interior Stretch (fac,x0)

fac : Stretching Factor, stretch in cells
For iopt = [1,2,3] $1.0 < fac < 2.0$
fac = 1.0 -- Maximum Stretch
fac = 2.0 -- Uniform

After filling the *xyz file* with the dimensions and stretching options, save the *xyz file* by the appropriate name (for example: grid). The following steps are to be performed in order to get the correct grid in a suitable format:

Step 1: Type *xyz*

```
mhd@machine$ xyz
```

```
Enter filename [.xyz]: grid
```

Grid generation completed! will be displayed on the terminal screen.

Step 2: Check for the ***checkXYZ.dat*** file, *Figure 4*. It contains the size of each cell in the grid in x, y and z direction.

(a)

```
 grid       xf           dx           xc
    1  0.000000E+00  8.136532E-01  4.068266E-01
    2  8.136532E-01  7.822461E-01  1.204776E+00
    3  1.595899E+00  7.241007E-01  1.957950E+00
    4  2.320000E+00  6.471543E-01  2.643577E+00
    5  2.967154E+00  5.604857E-01  3.247397E+00
```

(b)

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Multi-segments HEXAHEDRAL grid (nopt= 1) with hunt3d_half.xyz
 min: dx= 4.72360E-01   dy= 2.06106E-03   dz= 1.73059E-02
 max: dx= 1.03228E+00   dy= 2.59095E-01   dz= 2.18595E-01
------------------------------------------------------------
 Domain range: x[    0.0000   25.0000]
               y[    0.0000    1.1000]
               z[   -1.1000    1.1000]
 Total number of nodes ( 33 x 19 x 33) =  20691
 Total number of cells ( 32 x 18 x 32) =  18432
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 4: CheckXYZ.dat file (a) - start of the file and (b) - end of the file

If there are 2 or more number of segments in the grid and if there is stretching in any part of the grid, then the change in the size of each cell should be almost constant throughout. In each direction, this can be checked by plotting *dx*, *dy* or *dz*. The transitions in the curve should be as smooth as possible (Figure 5).

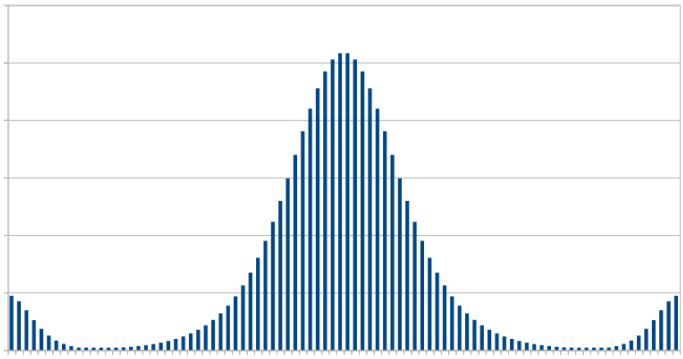


Figure 5: Plotting dx, dy or dz should look smooth

Step 3: Converting Grid to .ux format

In step 1, along with ***checkXYZ.dat***, another file is created by the name of ***univ.cemgrd*** which is then used in the process of converting the grid to .ux format.

```
mhd@machine$ Book grid.ux
```

3.3.2 Graphical User Interface-Based Grid Generation

To enhance the utility of our grid generation tool, we have also developed a GUI to facilitate the generation of the input file. The sample input files shown can be generated using this GUI. The GUI enables the user to define the number of segments, the number of grid points per segment and the stretching parameter. The GUI interface also writes out a text input file to enable future verification. It can be accessed by typing the name in the command box:

```
mhd@machine$ himag_grid
```

A screen like Figure 6 will appear indicating that it is ready for the user. Enter the project name and begin.

Figure 6: HIMAG Grid Creation Welcome window

3.3.2.1 Pre-defined Cases

These are the duct cases for which the grids are inbuilt and the user only needs to specify the dimensions. It gives the user different options, like if it's a rectangular duct or a circular one, if the walls are conducting or insulating. The pre-defined cases can also be used as an entry into the world of HIMAG grids.

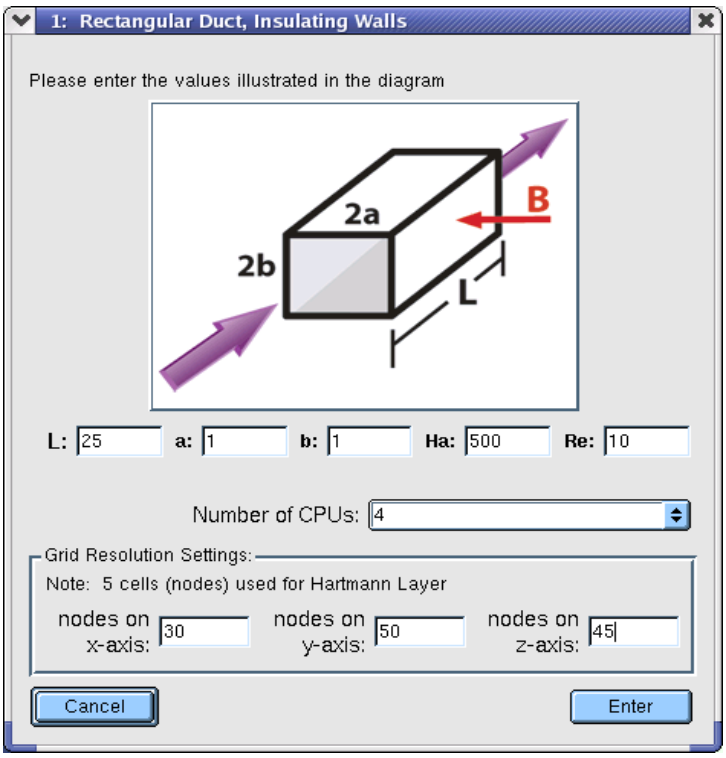


Figure 7: Example with rectangular duct and insulating walls

The user needs to enter the length of the duct, other dimensions, Hartmann number and Reynold's number for which the grid needs to be made. The number of CPUs will create the partitions for the grid for that many CPUs. The number of cells on each axis are also specified by the user. When all this is done, press Enter.

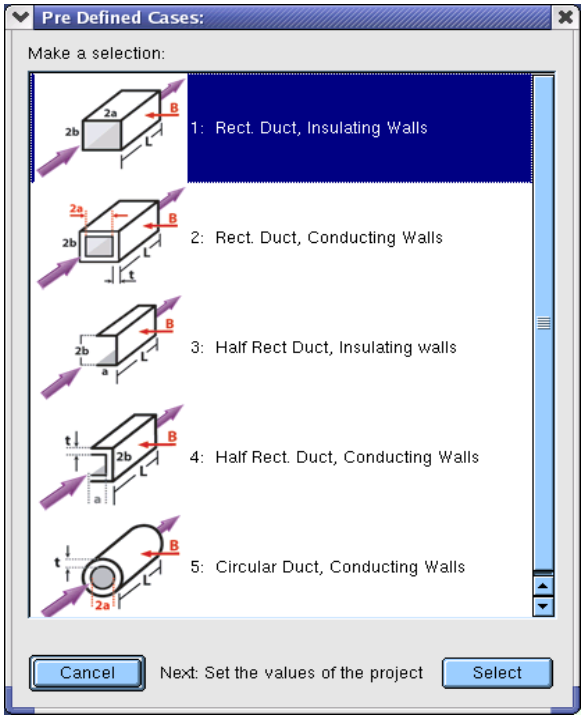


Figure 8: Menu for Pre-defined Cases

All the files created in this step are shown in Figure 9, test1.ux being the grid file, test1.ugb describes the boundary conditions setup, test1.xyz which gives you the access to the grid setup, test1.input being the input file for the case which will be discussed later in detail, (see Appendix I).

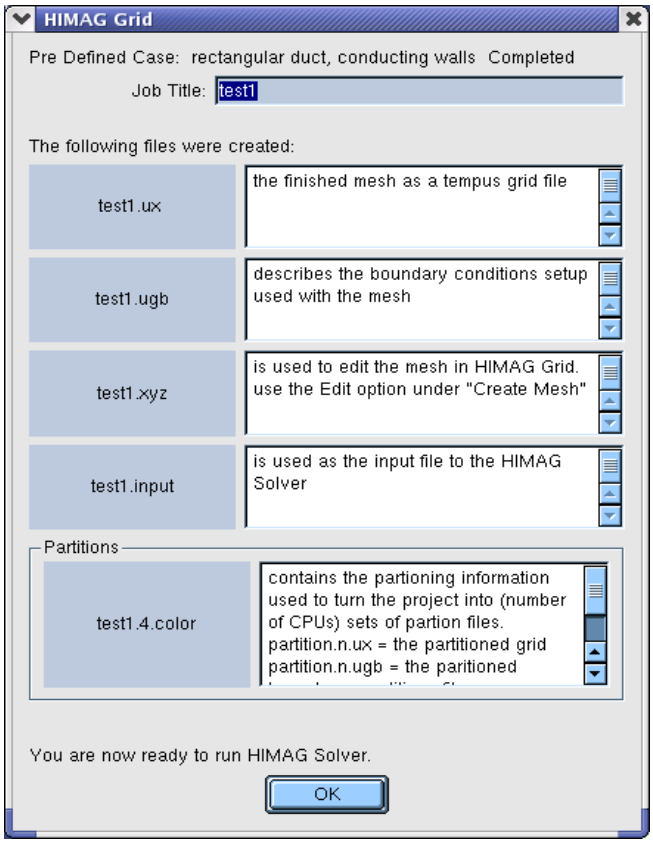


Figure 9: Dropbox showing all the files created along with their descriptions

3.3.2.2 Create Mesh

As shown in Figure 10, the next option is to create the mesh using the GUI. Create mesh option can also be used to edit an existing mesh. Again, start with specifying the project base name, for example, here we have given the name as ***test2*** and click on **Create Mesh**, shown in Figure 10.

Figure 10: HIMAG Grid Creation Welcome window

The generation of a hexahedral grid case is shown in these following diagrams, where the user creates piecewise linear segments and selects clustering parameters.

The user can create either hexahedral, prism or polar grid. Figure 11 window also shows the **Load Grid** tab where the user can load an existing grid to edit it.

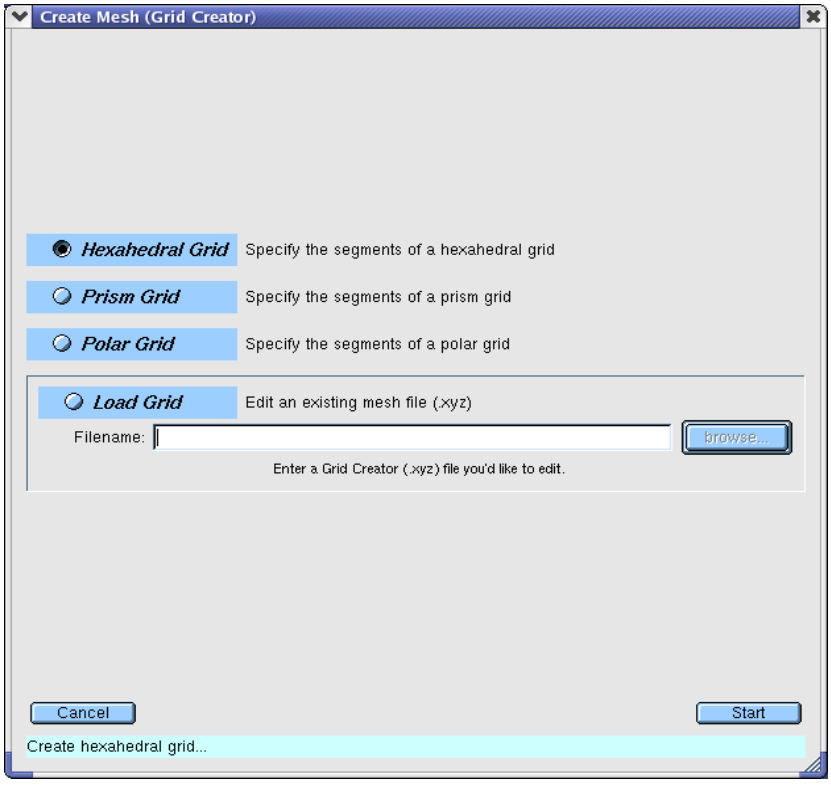


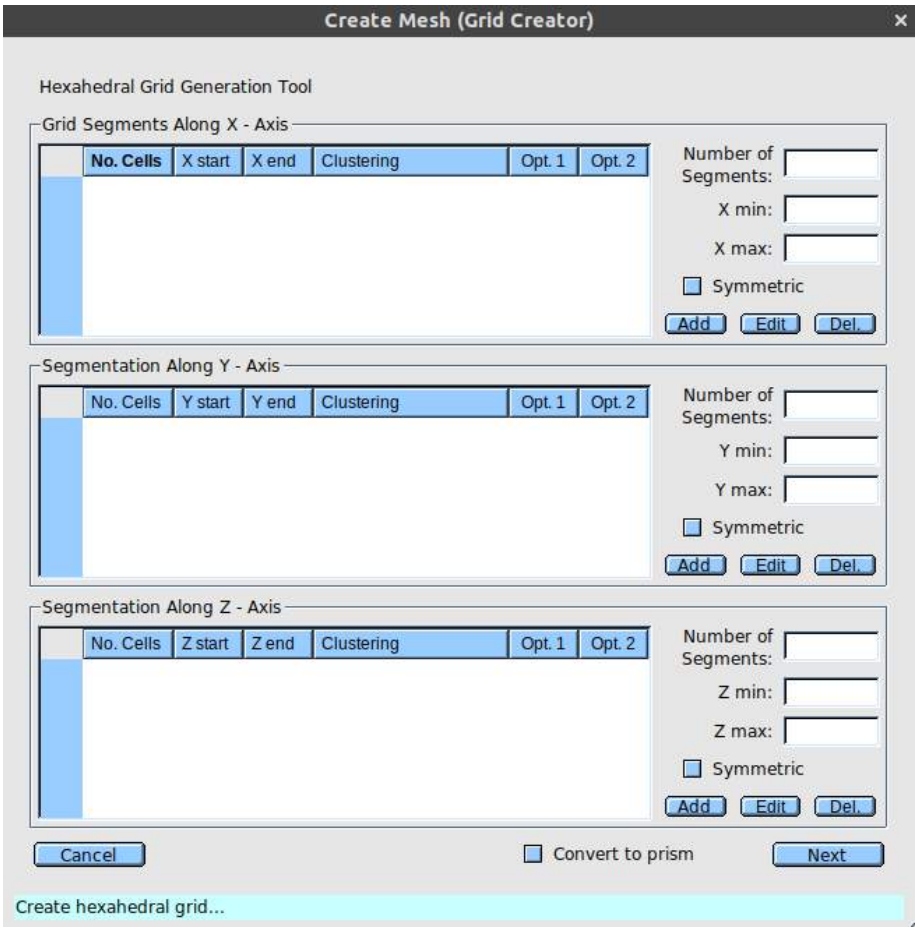
Figure 11: Create Mesh Window

Here, for the example, we go ahead with the hexahedral grid and click on **Start.**

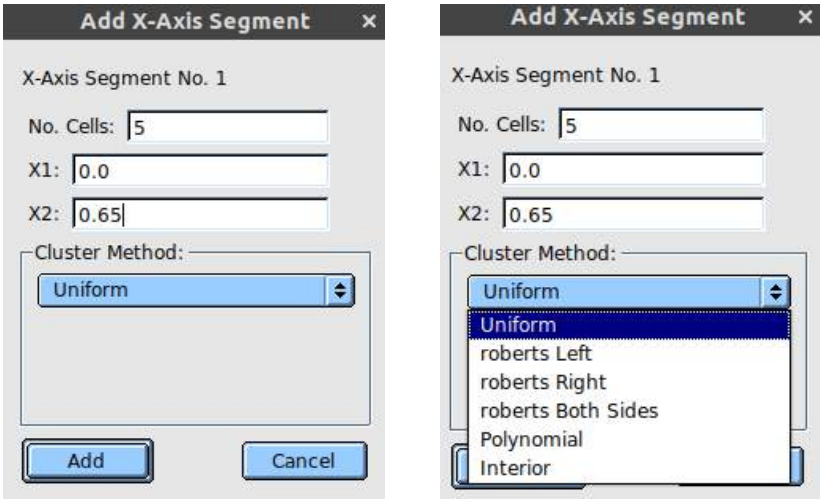
Figure 12(a) shows the window that will open after you click the **Start** button in Figure 11. This step is to add the number of segments, number of cells and clustering method for x-axis. Click on **Add** as shown in Figure 12*a*.

A pop-up will open as Figure 12*b* where the user can put the number of cells on x-axis in that segment. Choose the clustering method out of the six options shown in Figure 12*b* and click **Add**.

It will appear in the main window as a segment with all the details entered (Figure 13). Add as many segments as needed. We have added two segments here, one using Uniform Clustering and the other one using Polynomial Clustering. The final window will look like Figure 13 where all the information entered before is specified.



(a)



(b)

Figure 12: (a) Grid Segments along X-Axis, (b) Window showing Number of cells and Cluster Method

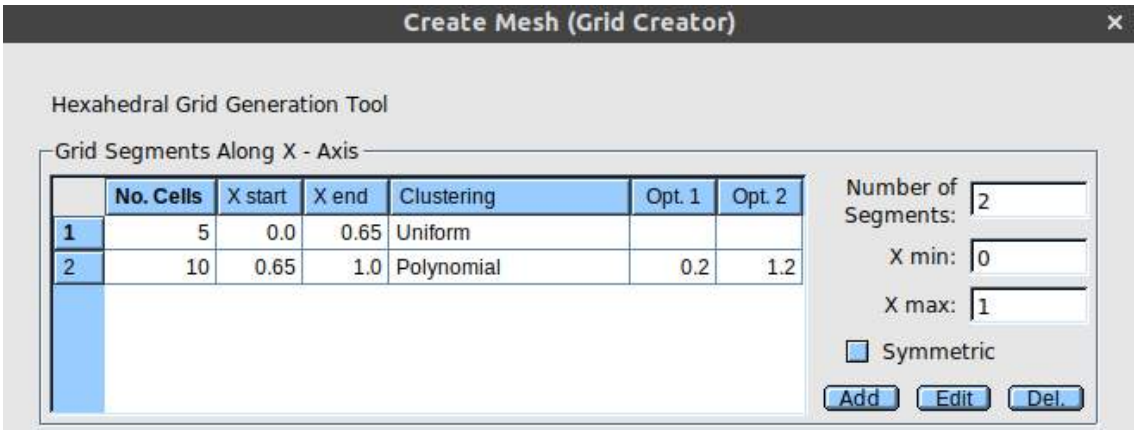


Figure 13: Create Mesh Window showing the segments in X axis

Similarly, add the segments and cells in Y-axis and Z-axis as needed as shown in Figure 14.

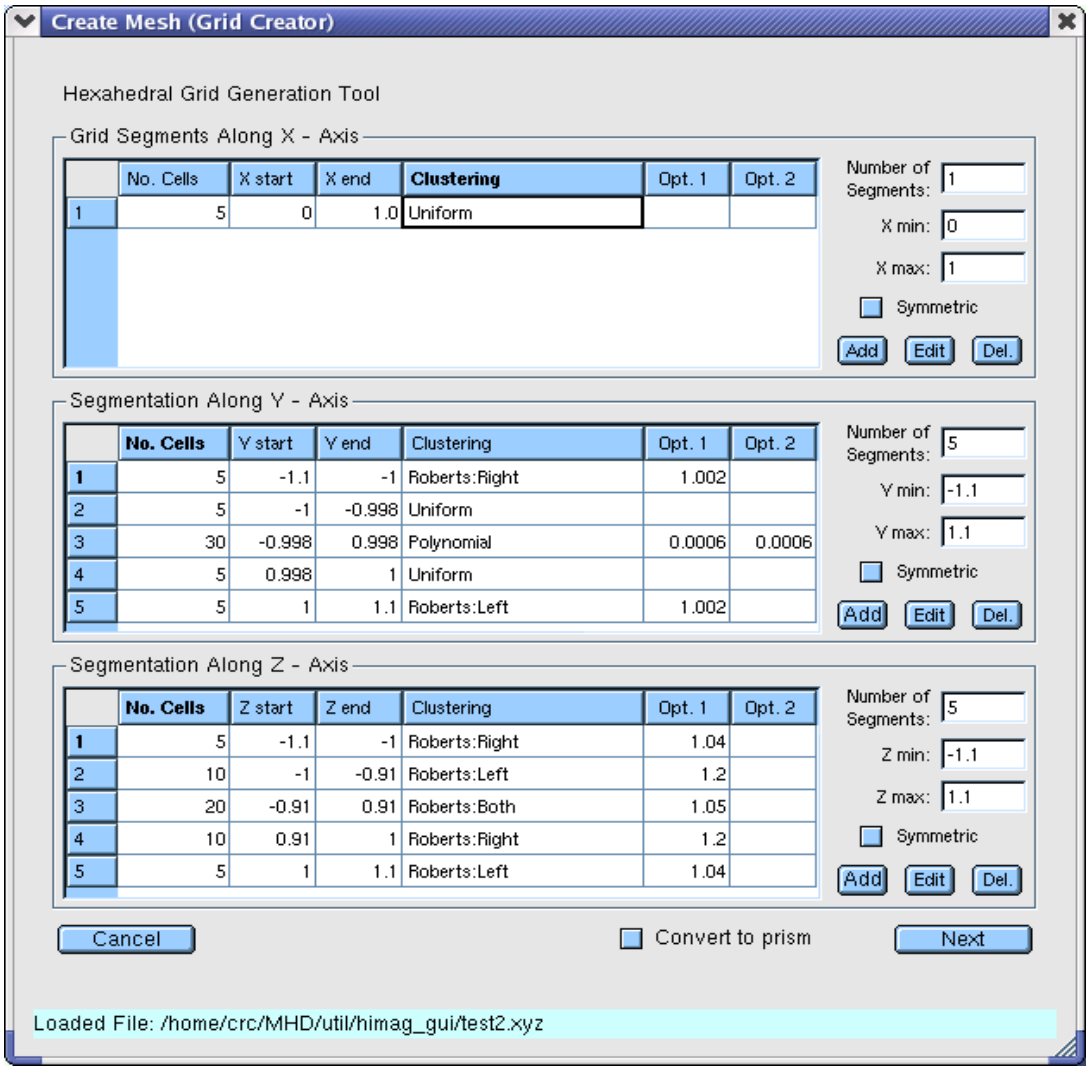


Figure 14: Final Mesh Window showing all the various segments added for the mesh

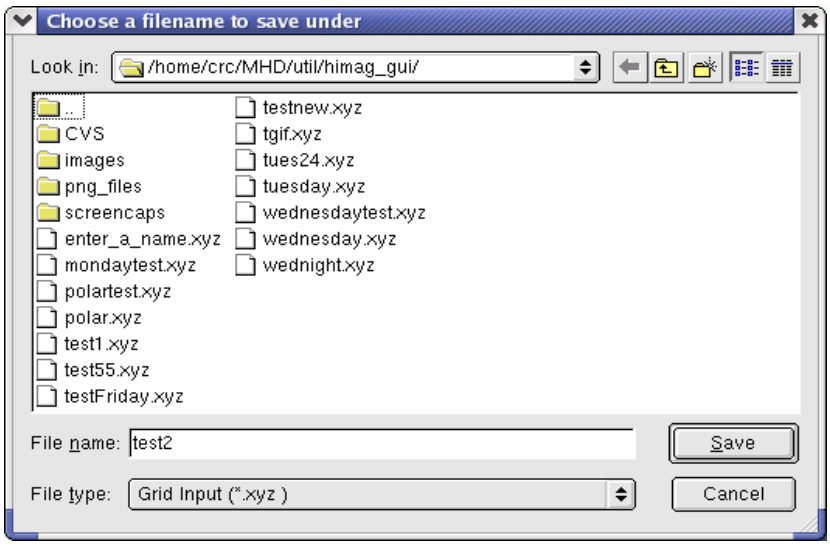


Figure 15: Save the Grid

Once the user has completed adding segments in Figure 14, click on **Next**. You will be asked to save the grid, Figure 15, give it a name, the format being ***.xyz*** and click on **Save**.

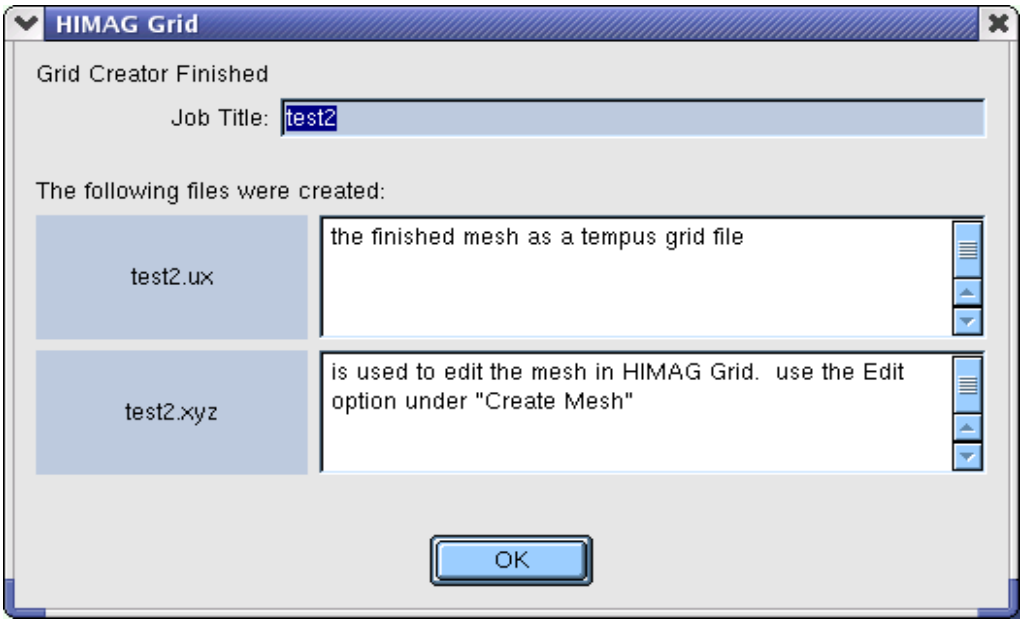


Figure 16: End of Grid Generation

The next window you see will indicate that the Grid Generation process is over and will mention the files created in this process. Those files will be ***grid.ux*** and ***grid.xyz***.

3.4 Examples of Grid Generation

An example of the use of this grid generation utility is for the study of a fully developed flow in a square duct is demonstrated. For this particular case, it is assumed that the magnetic field is applied along the Y-axis and the fully developed flow is in the X-axis. The half width for this duct is 1m and the center of the duct is at the origin.

The user specifies the extent of the domain and the number of regions (*nxsgm, nysgm, nzsgm*) for each of the coordinate directions (*X, Y* & *Z*). For each region, the user further specifies the number of grid points and the extent of the region. In the sample input file, the Y and Z coordinate axis have 3 regions (two for the near wall Hartmann/sidewalls and one for the core).

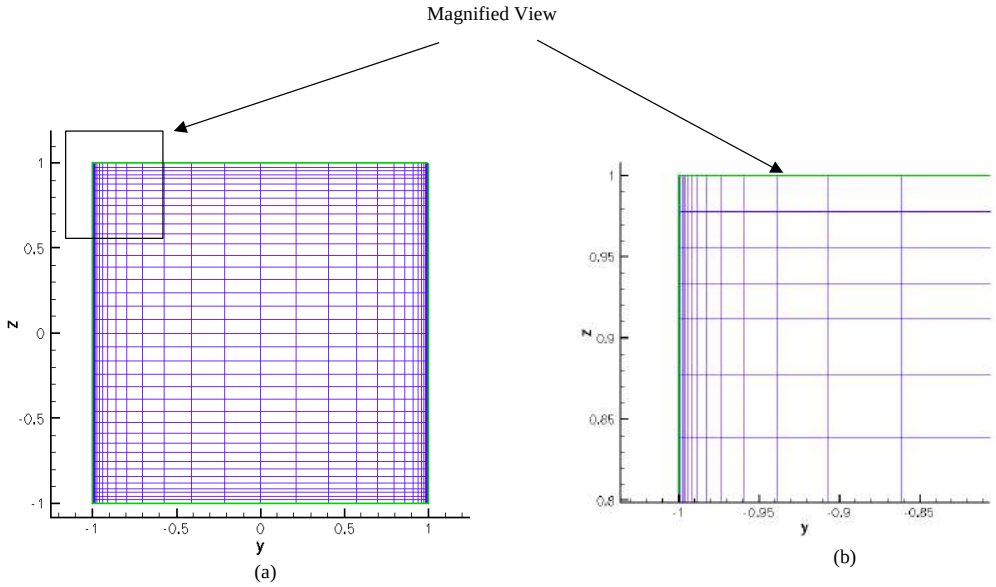


Figure 17: Grid layout (a) and magnified view of the Hartmann layer (b) for a square duct using the input file shown in Appendix 1

The appropriate stretching function with the stretching factor is used to ensure a smooth transition from the highly refined Hartmann/sidewall region to the core of the flow. In the core of the flow, gradients of velocity, electric potential and other variables are small, hence a coarse mesh is sufficient. Figure 17 shows the entire grid and an enlarged view of the grid spacing near the Hartmann and side walls.

Grid generation for Circular ducts: Thus far we have discussed the utility of the grid generation tool for generating hexahedral, orthogonal meshes for rectangular geometrical configurations. As mentioned earlier, HIMAG grid generation utility also

has the ability to generate grids for circular pipes.

Several types of grids with varying degrees of non-orthogonality can be generated. Figure 18 shows different kinds of grids that can be generated for circular pipes. Figure 18*(a)* shows a grid with an orthogonal core whereas regions near the outer periphery have highly non-orthogonal cells. Figure 18*(b)* shows another grid generated with cells which are not as non-orthogonal using our grid generation tool.

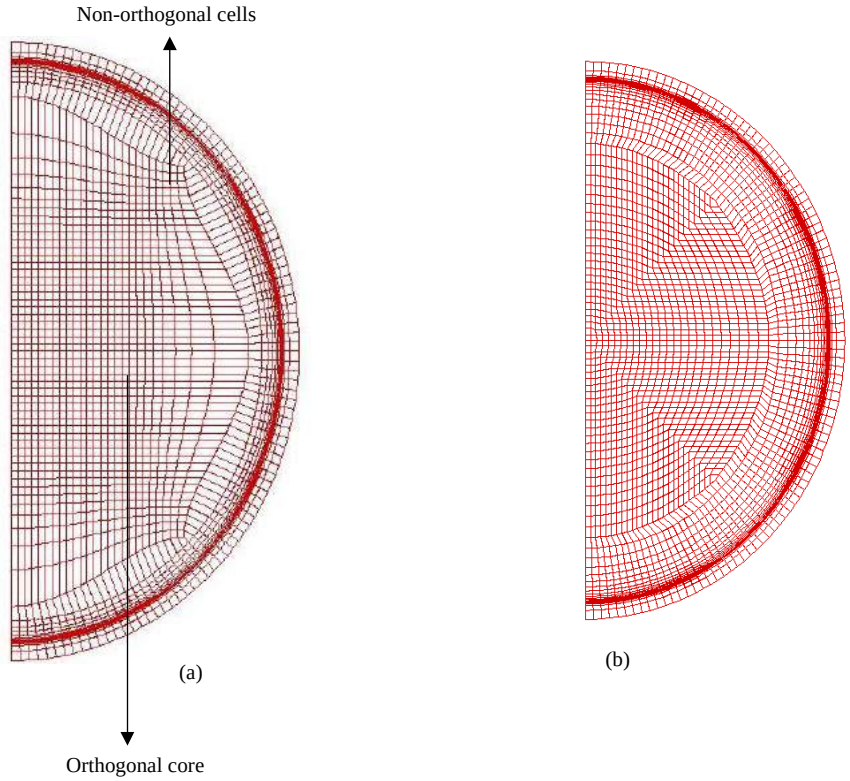


Figure 18: Grids generated for a circular duct with HIMAG-Grid

3.5 Importing Grid from ICEM

It's very much okay if the user does not want to familiarize themselves with HIMAG's grid generating software. HIMAG also incorporates a capability which converts the grid from ICEM in CGNS format to UX format that HIMAG uses, UX2CGNS.

```
mhd@machine$ ux2cgns grid.cgns
```

This command will make the conversion and provide the user with the ***grid_cgns.ux*** file, ready to use.

4 Pre-Processing

4.1 Introduction to HIMAG-Prep

This tutorial is intended to guide a novice HIMAG-prep user through a typical session. To begin this step, the user will need the following files: ***grid.ux***. During the tutorial, the user will open a .ux (geometry) file, create a patch on the geometry from a cutting plane, create a patch from free faces, assign boundary conditions to both patches, and save the project by writing a .ugb (unstructured grid MHD) file.

> ***NOTE:*** *The appearance of your dialog and windows will depend on the version and operation system you are using.*

From the command line run the program by typing "prep":

```
mhd@machine$ prep
```

A dialog will appear asking for a geometry file to load. Select "hunt3d.ux" and hit "open."

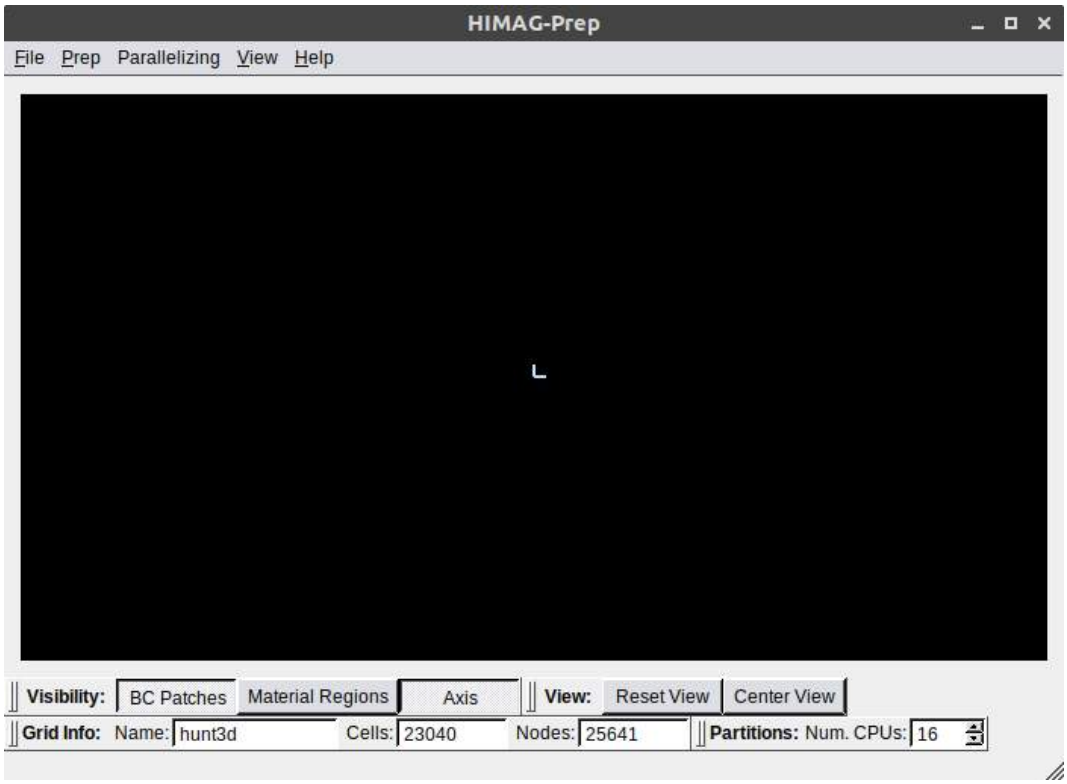


Figure 19: HIMAG-Prep Welcome Screen

Menu Bar:

- File : has all the known options, like Open, Save as and Exit.
- Prep : is the most important feature in HIMAG-Prep which is used to set the boundary conditions for any grid.

> ***USER TIP:*** *The name of the .ugb file will usually, but not always be the same as that of its corresponding geometry, or .ux, file. Normally, when a geometry file is loaded, HIMAG-prep will automatically look for a .ugb file (or a .ugm file, from old version, if .ugb file doesn't exist) with same name so that the properties of the geometry load at the same time. However, because the boundary conditions and materials properties of a particular geometry can be swapped out for other boundary conditions and material properties (created for that same geometry), multiple, uniquely named .ugb files may exist. This topic will be addressed again later in the tutorial.*

The Prep main window looks empty at first, Figure 19. The geometry is actually a rectangular duct, but only the white markers for the x, y, and z axis can be seen. While the geometry has loaded, no patches have been defined, so nothing is visible yet, except **Grid Info** (Name: hunt3d, Cells: 23040, Nodes: 25641). In the tutorial directory, you will notice that no *.ugb* file exists. No patch related information exists for this geometry yet. That will change as the tutorial is completed.

Let's talk a little about Figure 19, HIMAG-Prep welcome screen. At the top of the screen you will see the Main Menu. At the bottom of the screen is the secondary menu with toolbars for Visibility, View, Grid Info and Partitions.

Figure 20(a): HIMAG-Prep Toolbar 1, Visibility

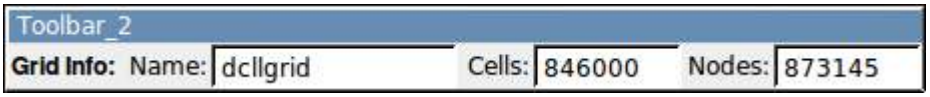


Figure 20(b): HIMAG-Prep Toolbar 2, Grid Info

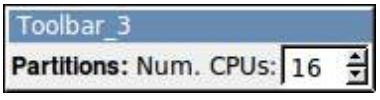


Figure 20(c): HIMAG-Prep Toolbar 3, Partitions

Figure 20(d): HIMAG-Prep Toolbar 4, View

These toolbars have toggles at each option for on and off condition. Under **Visibility**, there is **Axis** which can be made visible when required by turning on the toggle switch. Similarly, there are **BC Patches** and **Material Regions**, shown in Figure 20. Other toolbars are for **Grid Info** which contains the name of the grid, number of cells and number of nodes in the grid; **Partitions** which contains the information on number of CPUs for partition; and **View** which has toggle buttons for resetting and centering the grid.

Now let's focus on the main menu starting with **Prep**, Figure 21. It consists of an option to add **Boundary conditions** to the grid as well as to assign **Material Regions** on the grid.

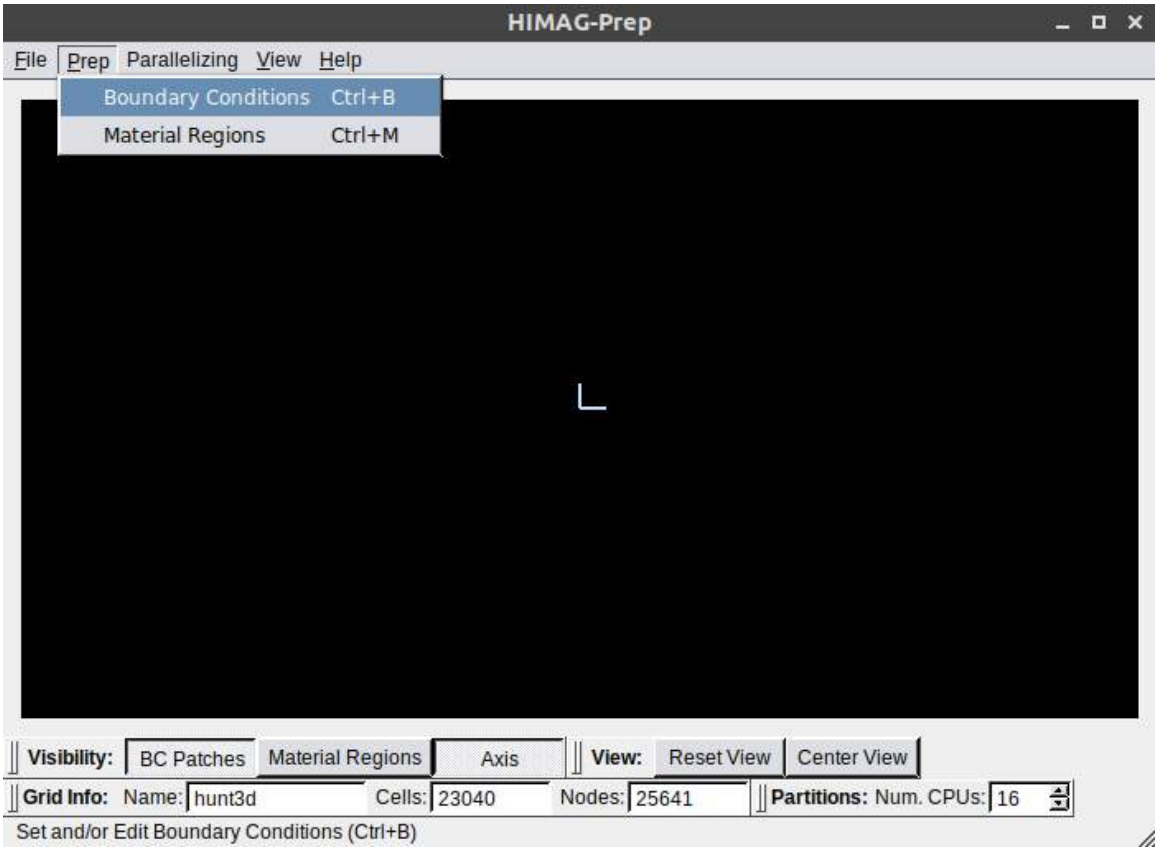


Figure 21: HIMAG-Prep Welcome Window, Prep Tab

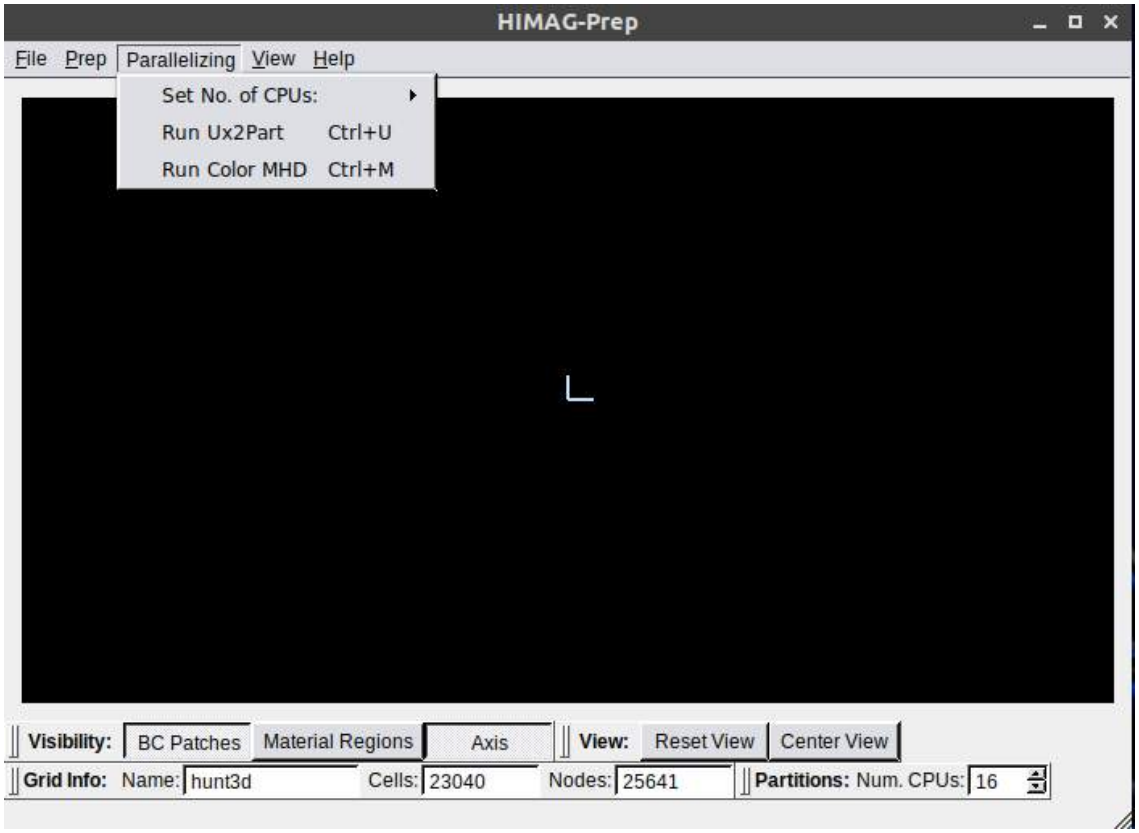


Figure 22: HIMAG-Prep Welcome Window, Parallelizing Tab

The next tab is **Parallelizing**. It consists of the three steps involved in partitioning the grid for running it on multiple cores, Figure 22. First, **Set No. of CPUs** is as the name states, to specify the number of CPUs for running the case. Second step is to **Run Ux2Part** which is to partition the grid in that many number of CPUs as specified in the first step. Third is to **Run ColorMHD** which is to create the .ux and .ugb files for the partitions.

The next tab is **View**. It consists of the three options, Grid Probe, Visibility and Orientation, as shown in Figure 24. These options are used to change the view of the grid, to turn it, rotate it

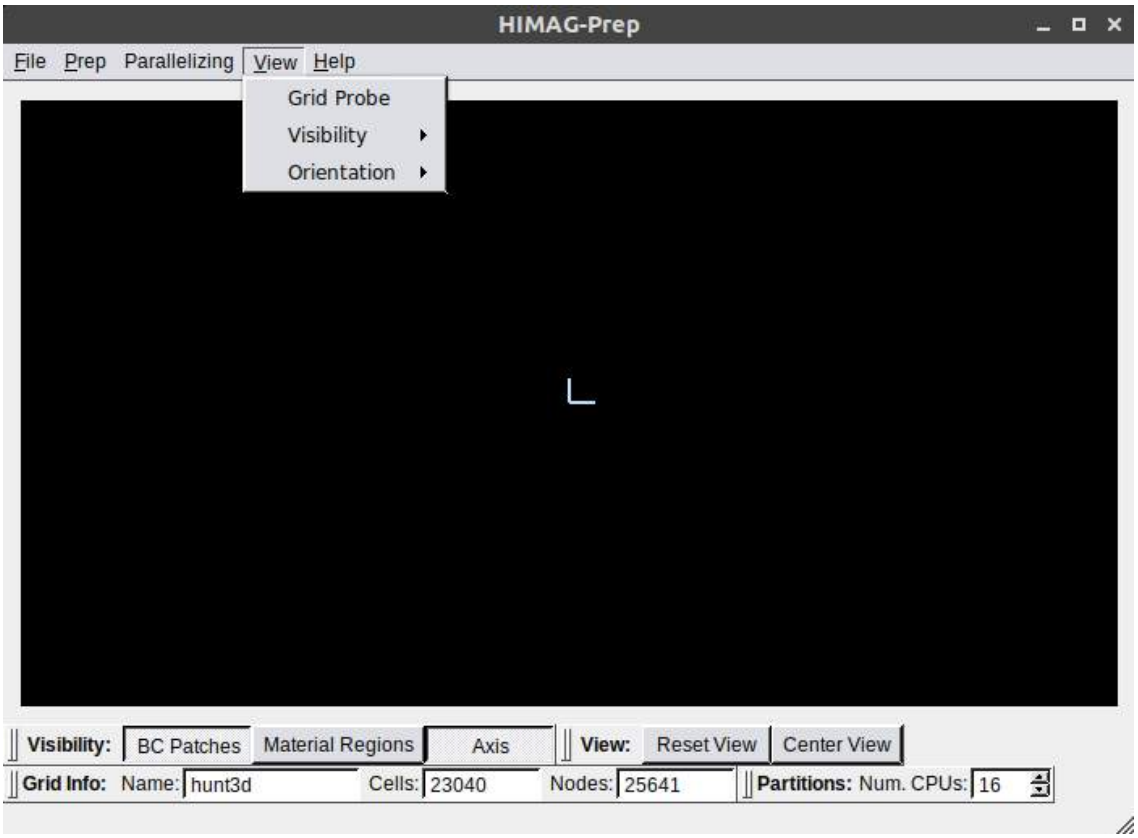


Figure 24: HIMAG-Prep Welcome Window, View Tab

Now, let's look into details of the **Prep** tab which is used for boundary conditions and material regions. The Boundary Conditions panel (Figure 23) has many functions. It allows the user to create and edit patches derived from the current geometry. Once patches are created, the panel describes the boundary conditions of all the patches in the geometry. Notice at the top of the panel a menu bar with

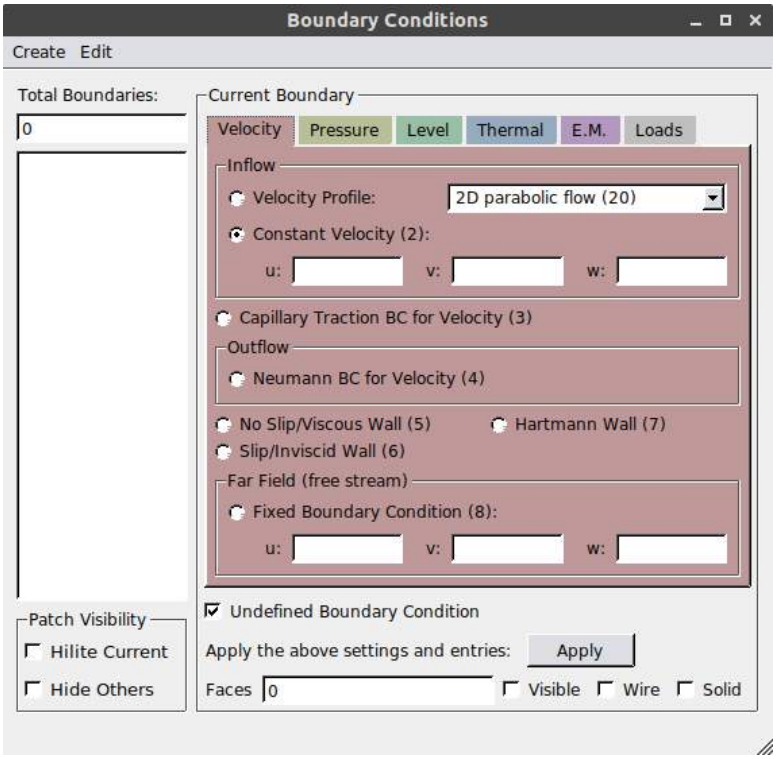


Figure 23: Boundary Condition Dialogue Box

Create and ***Edit*** as its options. The ***Create*** menu is where patches can be derived from the geometry either by a node range, a cutting plan, a min/max box, or from the remaining faces not included in any patch (free faces). The ***Edit*** menu is where patches can be duplicated and deleted.

Under Create, four different ways of creating a patch are given:

i. From the Node Range
ii. From the Cutting Plane
iii. From the Min/Max Box
iv. From the Free Surfaces

From the Node Range: In a grid, every node has an assigned number called the node number. Sometimes, a consecutive node numbers form a face which can be created in **Prep** using **Node Range**.

Let's say, the face of a channel duct has the node numbers ranging from 1 to 2500.

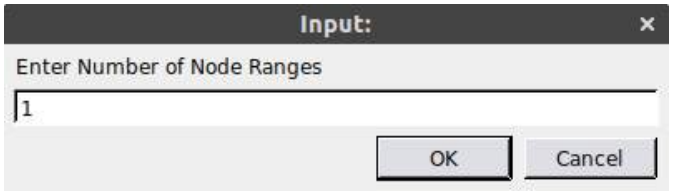


Figure 25: Number of Node Ranges

First, Prep will ask you to enter the number of Node Ranges that make up that face. So, we enter 1. Then the next pop up will ask you the beginning node number, so again we enter 1.

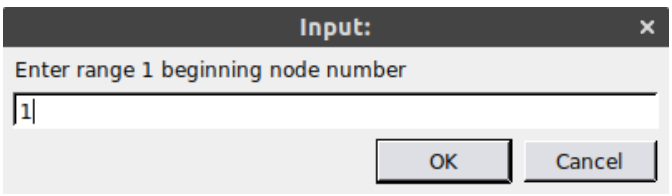


Figure 26(a): Beginning Node Number

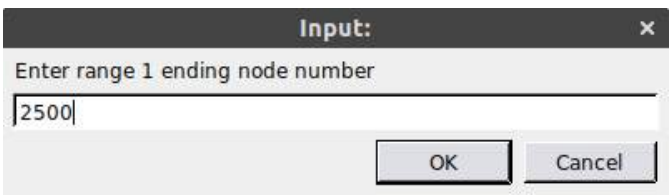


Figure 26(b): Ending Node Number

Then, the last pop up will ask you the ending node number, so we enter 2500, and a face is created. If the node numbers are entered correctly for the face, the face will show up in the ***Boundary Condition Panel*** as **1. 2:2:2:2:2:2** and will also update the column under *Total Boundaries*.

From Cutting Plane: It is used to create those faces of the grid which have one fixed coordinate. The dialogue box for cutting plane will mainly focus on the axis along which the cut needs to be made in order to get a face of the grid, for example, when you select the option ***From Cutting Plane***, a pop-up window will appear asking *Create patch along which cartesian plane?* : Enter "x", "y", or "z" according to your need. We select "x" for this tutorial.

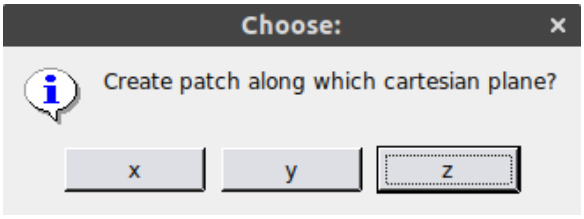


Figure 27: Patch along which plane

Another pop-up window will appear asking you for the tolerance, Figure 28. For now, we can let it remain the default value or 0.001.

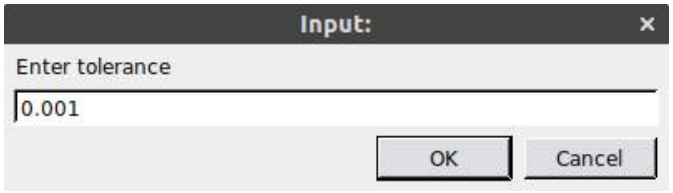


Figure 28: Enter the Tolerance Value

The next pop-up window will ask you for the plane value, Figure 29. Enter the coordinate value to place inlet/outlet on that particular cartesian plane that one entered as the answer for the first question, eg, 0 or 100.

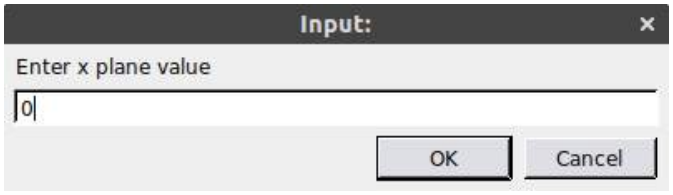


Figure 29: Value of plane in x-axis

A face of the grid present on the x = 0 plane will appear in the main window as shown in Figure.

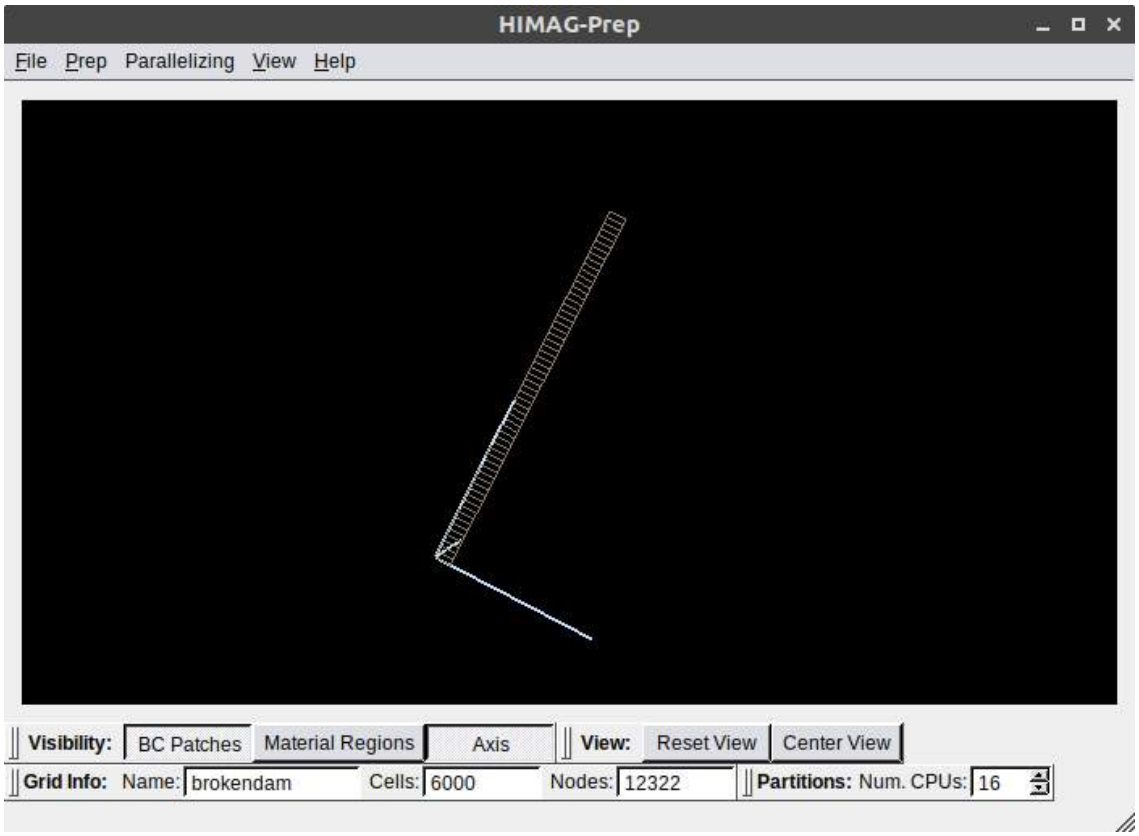


Figure 30: Patch created by cutting the plane in x-axis

From Min/Max Box: To create a separate patch in your grid. A surface which is a part of a wall but has different properties than the rest of the walls. The dialogue box for min/max box will open. Enter the surface dimensions:

Enter the minimum value of the face in x-axis

Figure 31(a): To create a box, enter minimum value in x-axis

Now enter the maximum value of the face in x-axis.

Figure 31(b): Enter maximum value of x-axis

Similarly, enter minimum and maximum value of y-axis for the face in the next 2 pop-up boxes.

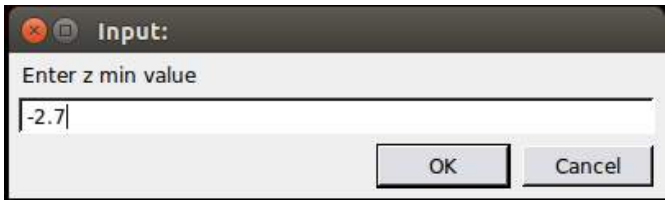
(a)

(b)

Figure 32: (a) Minimum value of box in y-axis, (b) maximum value of the box in y-axis

And finally, enter the min and max values for z axis.

(a)



(b)

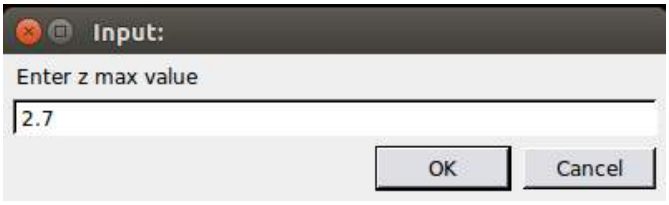


Figure 33: (a) Minimum value of box in z-axis, (b) maximum value of box in z-axis

From Free Surfaces: To create the remaining faces of the grid.

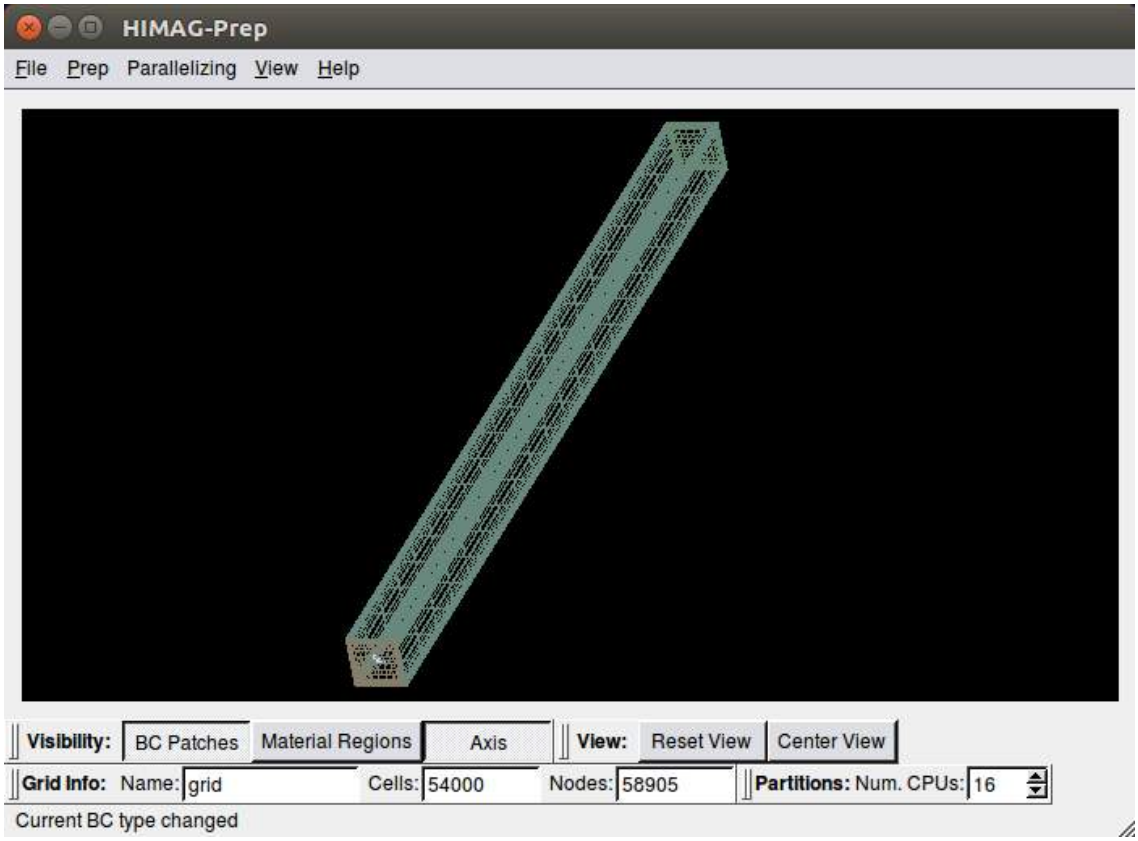


Figure 34: Final window with all the faces

4.1.1 Boundary Conditions

As we have mentioned before, that in the beginning each and every face of the grid will show up as 2:2:2:2:2:2. We will shortly mention what these numbers mean. For now, let's keep in mind that these are the default settings in Prep. Each digit corresponds to a BC in the multi-colored tabs of the Boundary Condition Panel: ***Velocity***, ***Pressure***, ***Level Set***, ***Thermal***, ***Electrical Potential*** and ***Loads***. Selecting a type from each of the six aspect tabs defines the BC of the faces. In this panel, we have radio buttons, which means only one choice per tab can be chosen.

In the ***Velocity*** tab (Figure 35) we can see we have options for *Inflow*, *Outflow*, *Capillary Traction*, *Walls* and *Far Field free stream*. For *Inflow*, we can either have a *Velocity Profile* or *Constant Velocity*. The *Velocity Profile* has a drop down list of options which includes a *2D* or a *3D parabolic flow*, *Fully developed flow* and *User defined* option as well. For Constant Velocity, the user needs to specify the velocity in x, y and z direction. These columns take real numbers.

You must have noticed that each of these options has a number in brackets like this **(3)**.

These numbers correspond to the 2:2:2:2:2:2 that you see in the left most column. If you select, let's say the Outflow option, Neumann BC for Velocity (4), and hit Apply, you will see that the 2:2:2:2:2:2 has now changed to 4:2:2:2:2:2. Similarly, if you want that face to become a wall, choose Slip/Inviscid Wall (6), and that configuration will now change to 6:2:2:2:2:2. As mentioned before, there are 6 digits in this configuration and each digit belongs to a BC in those 6 categories of Boundary Conditions.

> ***USER TIP:*** *You must make a selection from each of the six aspect tabs and then click the "Apply" button underneath the tabs to define or change a boundary condition. Leaving any one of the tabs unchanged means the definition of the patch's boundary condition remains the same as the default.*

Now, let's assign boundary conditions for our current grid as an example. Let's select the first face which will be our inflow and give it a Constant Velocity of u = 0.05, v = 0 and w = 0 and hit Apply.

Since, we selected *Constant Velocity* which has the number 2 next to it, the digits in the configuration of the face side will not change.

Now, let's move on to the Pressure tab, Figure 36. This one is easy. It has only two options, either you provide a value of pressure in the Constant Pressure option or you select Neumann Pressure.

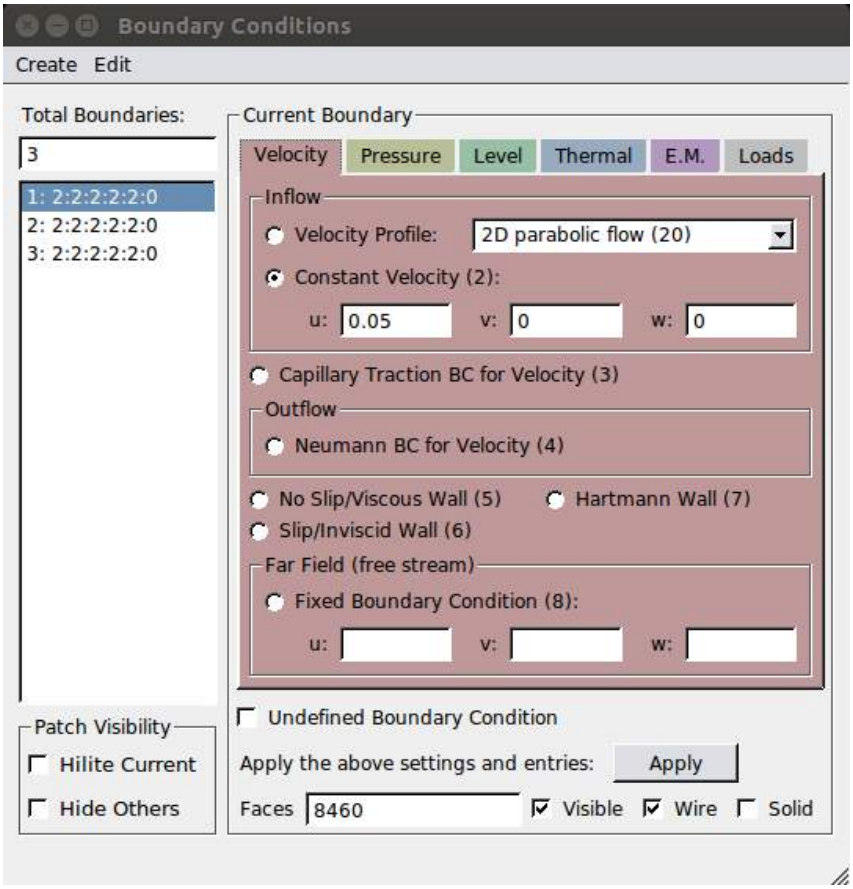


Figure 35: Velocity tab in Boundary Conditions

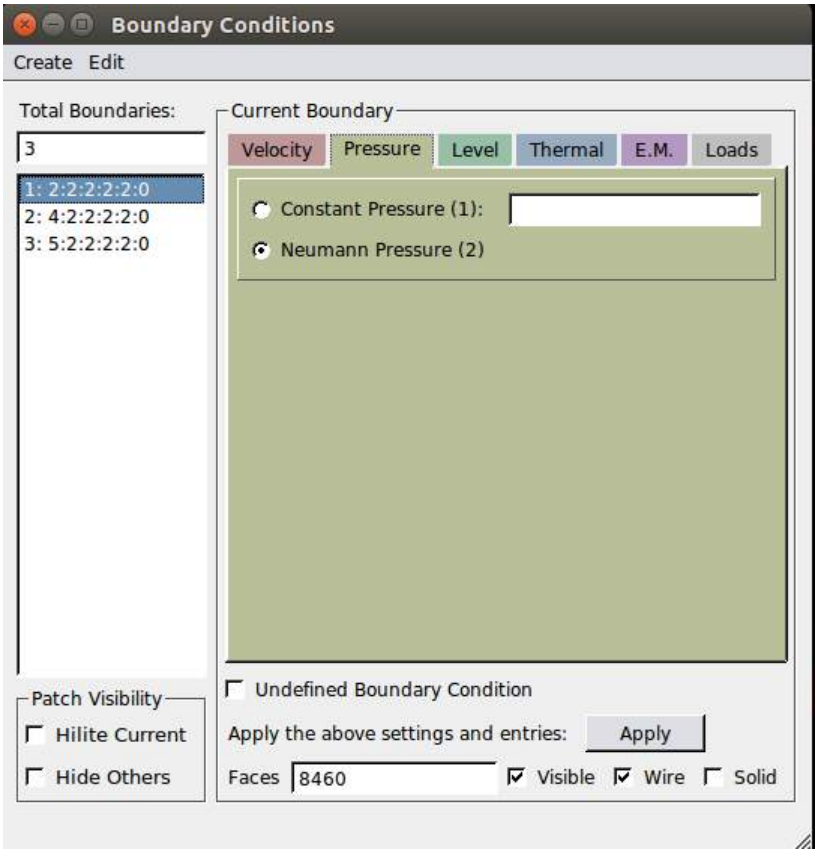


Figure 36: Pressure tab in Boundary Conditions

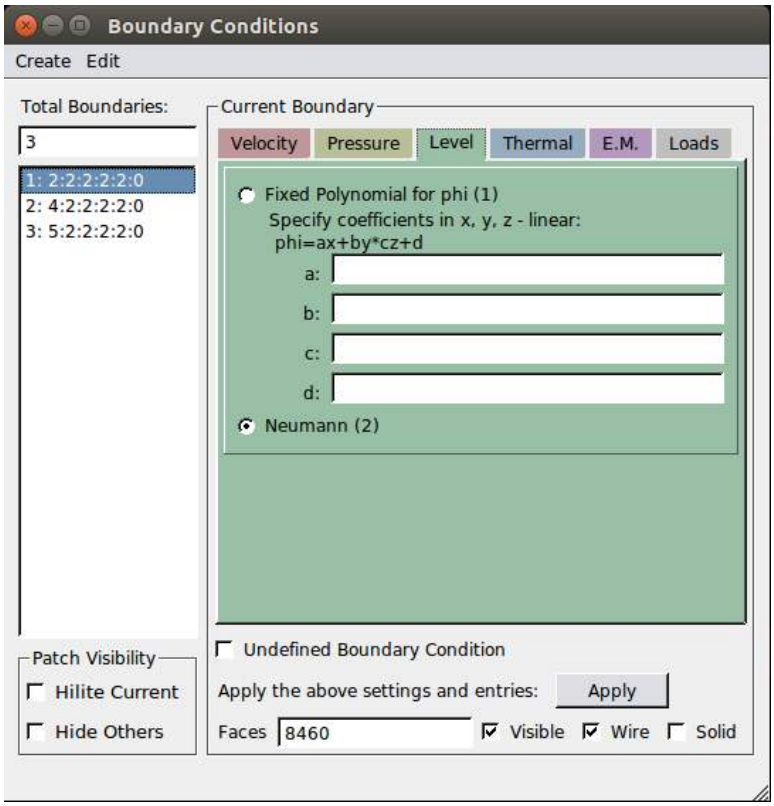


Figure 37: Level-Set tab in Boundary Conditions

Here we will choose Neumann Pressure and click on Apply.

The next tab is the Level tab which is short for Level-set, Figure 37. This tab also has two options, either we can choose Fixed Polynomial for phi or Neumann phi and we are going to choose option 2, Neumann.

The next tab is of Thermal, Figure 38. This tab has three options, either input the value of *Fixed Temperature (1)* where the value is specified in Kelvin, use *Adiabatic (2)* temperature or provide *Fixed Heat Flux (3)*. We will use the second option here.

The next tab is of E.M. (Figure 39) and has 5 options. First one is *Specified Phi (1)*, then $n \cdot J = 0$ *(2), Hartmann Layer Assumption (3), Neumann Phi (4)* and the last one being *Thin Conducting Wall (5)*. In this case we will go with $n \cdot J = 0$ and hit Apply.

The next tab is of Loads (Figure 40) which consists of 2 options, Don't include Force/Moment calculations (0) and Include Force/Moment Calcs (1). We will go with option 1 for now and hit Apply. With all six aspects defined, the panel will get updated, showing the x = 0 plane patch to be 2:2:2:2:2:0.

Next, the user will assign boundary conditions to the second face as 4:1:2:2:2:0 indicating Neumann BC for velocity and a constant pressure of 0. The last patch is the faces with solid walls for this example so just assign "No Slip/Viscous Wall" for Velocity and keep everything else unchanged. Notice that

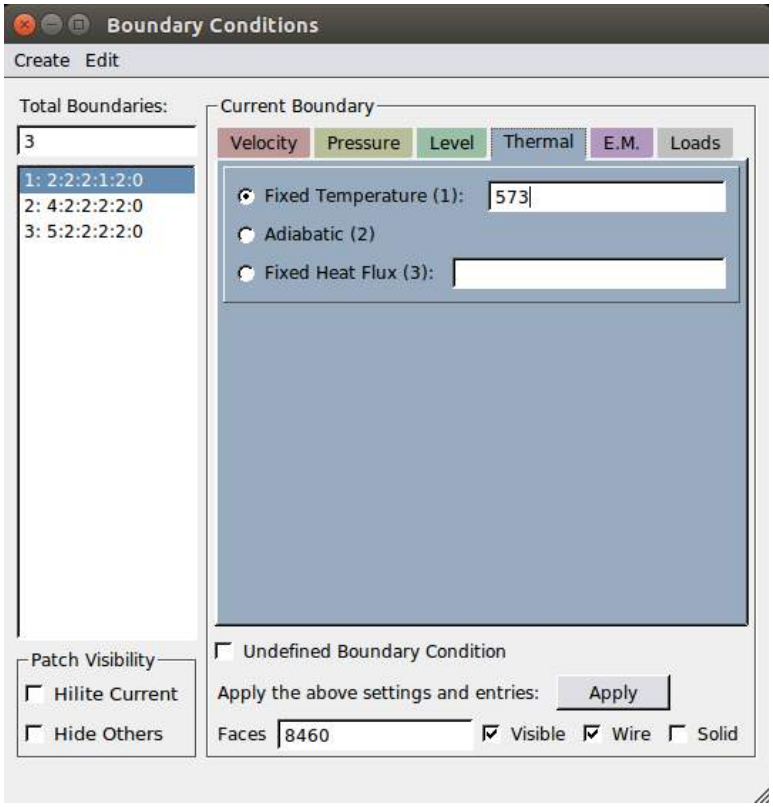


Figure 38: Thermal tab in Boundary Conditions

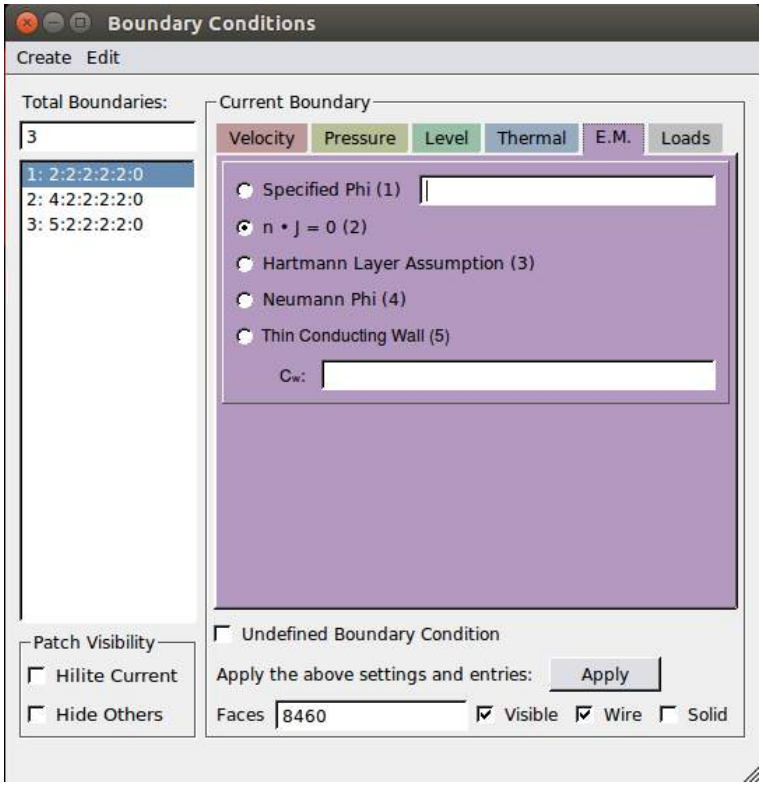


Figure 39: EM tab in Boundary Conditions

the “Total Boundaries” field indicates that there are now 3 patches. The panel should now look like Figure.

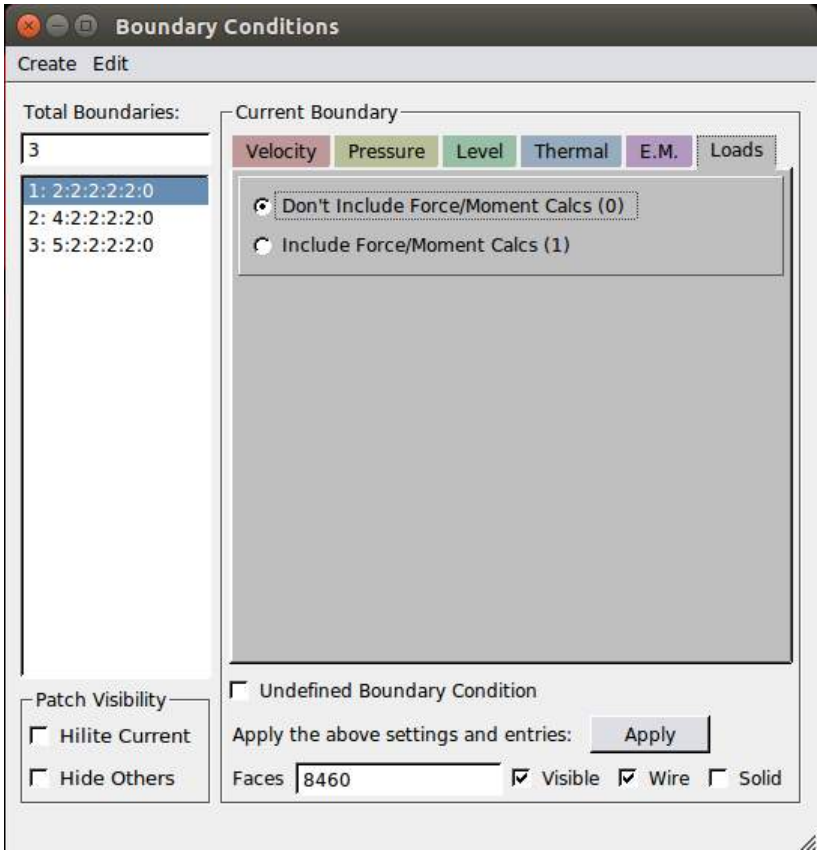


Figure 40: Loads tab in Boundary Conditions

You may check Patch Visibility “Hilite Current” and/or “Hide Others” to view your current selected patch clearly. “Hilite Current” will gray all other patches and keep the current with its color, while “Hide Others” will totally hide other patches except the current one.

After each boundary condition assignment a specific color is assigned to that patch in the main window. The color will have a hue, saturation, and value based upon the type of boundary condition assigned.

4.1.2 Manipulating the Geometry

The geometry can be scaled, rotated, and translated to aid in the viewing of the geometry as well as the boundary condition assignment of its patches.

1. SCALING / ZOOMING: On a three button mouse, using the RIGHT button to scale / zoom.

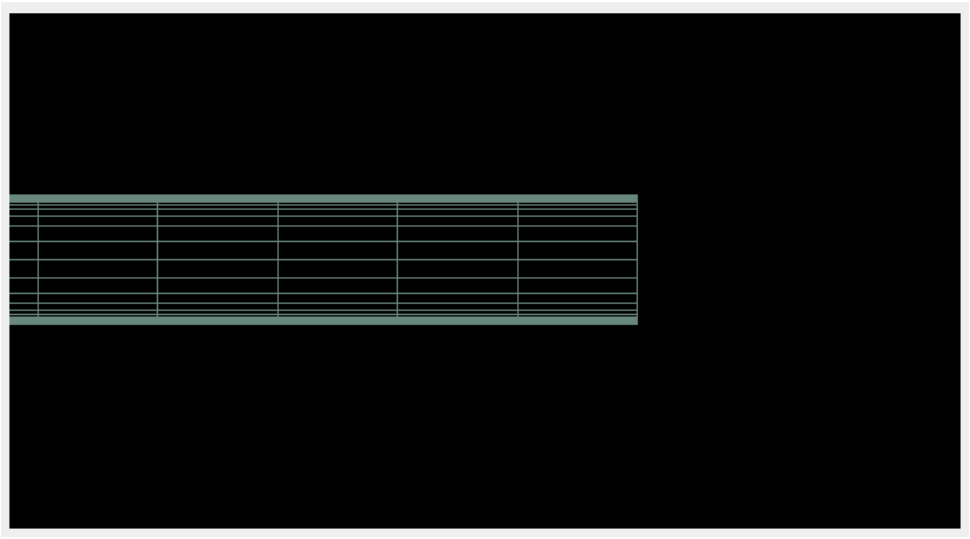


Figure 41: Zoom in Prep

2. TRANSLATION: On a three button mouse, use the MIDDLE button to translate in any direction.

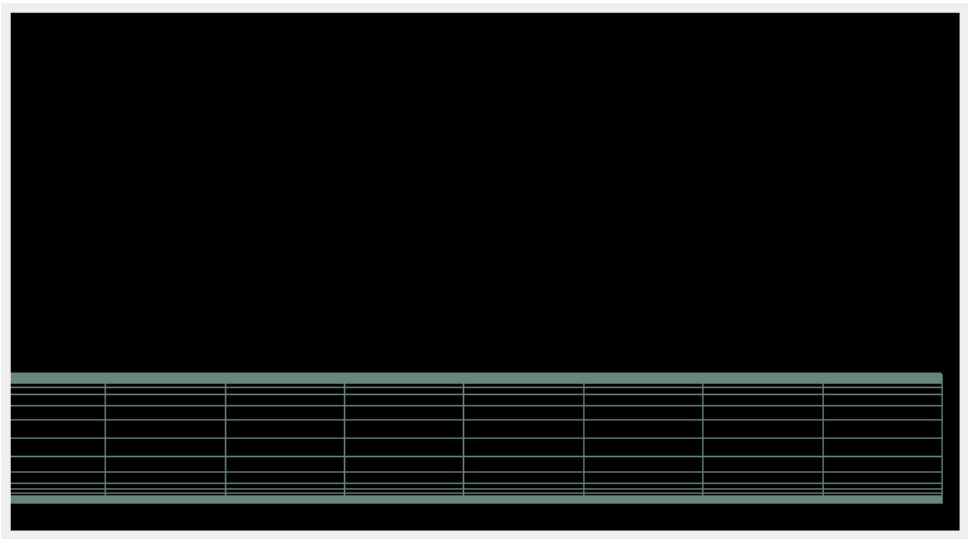


Figure 42: Translation in Prep

3. ROTATION: On a three button mouse, using the LEFT button to rotate.

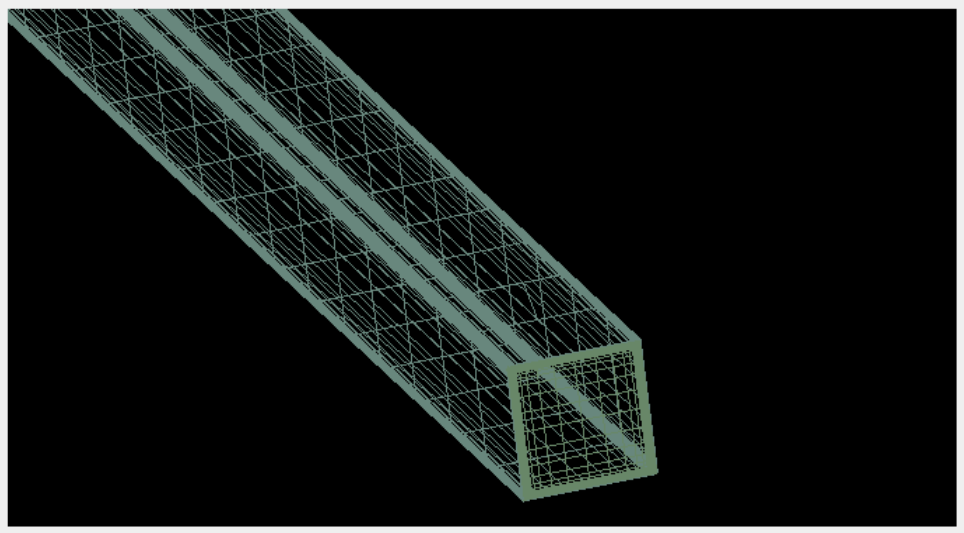


Figure 43: Rotation in Prep

FILE SAVING: Saving in HIMAG-prep writes the boundary condition, patch information, and materials information of the geometry to a file. ***Save .ugb File*** has the same root name as the current geometry but ends with the .ugb extension. This special file is searched for by HIMAG-prep whenever a geometry is loaded. By ***Save .ugb File as…***, the user can conveniently save the .ugb information to different name for another use.

OPEN: If there are multiple .ugb files for a particular geometry, the user can swap the .ugb file (thus changing the boundary condition information, etc.) once the geometry is loaded. This is done by selecting ***Open*** from the ***File*** menu. Doing so opens a file open dialog, which allows the user to select a different .ugb file from the one currently loaded.

EXIT: Select ***Exit*** from the ***File*** menu. This closes the HIMAG-prep program.

- Open the Boundary Condition panel and you will notice that the panel is set with the information from the session you just completed.
- Select “Open” from the “File” menu. You may now choose “hunt3d2.ugb” from the file open dialog. If you made any changes between hunt3d.ugb and hunt3d2.ugb, they would be reflected after loading.
- Select “Exit” from the top menu “File” to exit HIMAG-prep program. If you have already made some changes for the BC, it will prompt to ask if you want to save the changes before closing. Select “Yes” or “No” to close, or “Cancel” to make more changes.
- Select "Material Regions" under Prep. The Material Regions dialogue box will pop-up.

4.1.3 Boundary Conditions for A-Φ Formulation

PREP is fully accommodated with the boundary conditions related to Φ-formulation and B-formulation. For A-Φ formulation, the boundary conditions for magnetic vector potential are not present in PREP. Therefore an additional utility is made, ***mod_ugb***.

Modified UGB or mod_ugb is used after the Φ boundary conditions are already assigned in PREP. Once the grid.ugb file is ready from PREP, then another file bc_ugb.dat is created by the user. It consists the number of patches where magnetic vector potential boundary condition needs to be assigned, patch ID from PREP and corresponding boundary condition value. The bc_ugb.dat file from inside looks like:

```
0  5
1  10
2  0
3  40
4  40
5  0
```

Starting with the first row, the first value indicates … the second digit indicates the number of patches in the grid, so let's say we have 3 patches here. The second row starts with the data for patches. The first value indicates the patch ID from PREP shown in Figure 44.

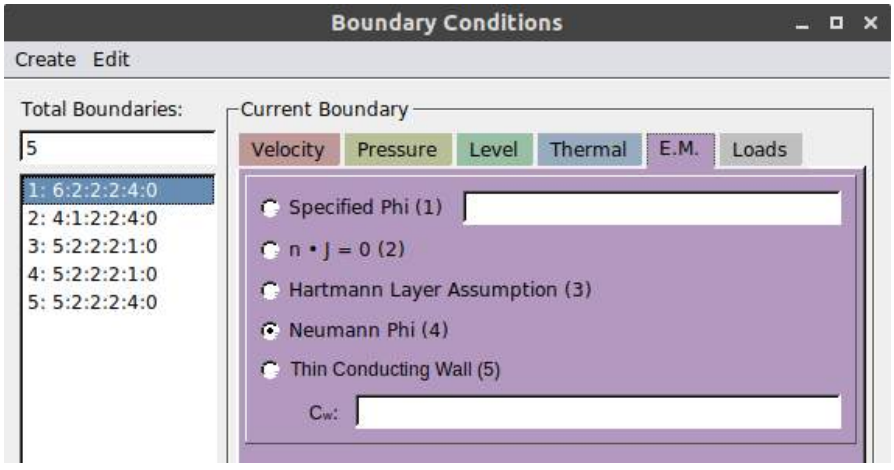


Figure 44: PREP Patch ID before A boundary condition

The second value is the number corresponding to the magnetic vector potential boundary conditions which are given as:

$$\mathbf{0:} \quad A = 0 \text{ (default)}$$

$$\mathbf{10:} \quad \frac{\partial A}{\partial n} = 0$$

$$\mathbf{40:} \quad A \times n = 0$$

The command to put magnetic vector potential boundary conditions or **A**-BC is:

```
mhd@machine$ mod_ugb [gridname]
```

The value used in bc_ugb.dat are then added to the EM tab's value as shown in Figure 45. For patch ID 1, we can see that in bc_ugb.dat we have 10 next to 1 in the second row, which means that we want $\frac{\partial A}{\partial n} = 0$ boundary condition on patch 1. As we can see in Figure 9, that after running mod_ugb, if we open the grid in PREP again, we see that the EM value is now 14 (4+10), indicating 4 for Neumann Φ for EM that we chose and 10 for $\frac{\partial A}{\partial n} = 0$. Similarly, for patch 2, we have 0 in bc_ugb.dat and therefore when we open the grid in PREP, we see (Figure 45) that the value for EM for patch 2 remains the same after running mod_ugb, since it's the default value and 0+2 = 2. For patch 3, we specified the value as 40 in bc_ugb and after running mod_ugb, in PREP, the value for EM parameter is now 41 (1+40). 1 for specified Φ value and 40 for $A \times n = 0$.

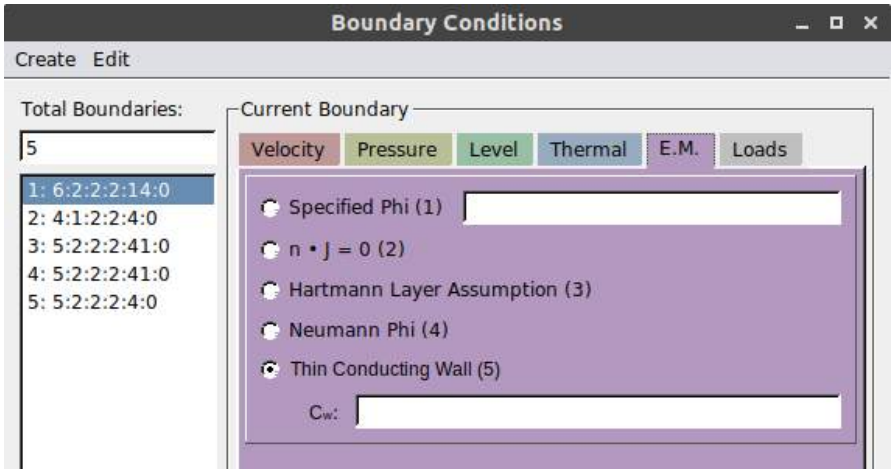


Figure 45: PREP after adding A boundary conditions

As you can see that since the fifth value in patch 3 and 4 are now modified but when patch 3 or 4 are selected, the old EM value that we entered is no more visible. Does it mean that the old potential value that we entered does not apply anymore? No. It still applies. This is just a visual bug in PREP at the moment and will be modified in the next edition. For now, the temporary fix is by using the following command to view the EM values that are associated to those patches.

```
mhd@machine$ mod_ugb [gridname] -ck
```

4.2 The Input Files

Even though HIMAG has default values set for all the keywords used in the code. But one important way to customize a case and give the user control over a case is by customizing the input files used by HIMAG.

There are ways that the code can itself be customized as per the requirement of the user, but for now there are three input files that a beginner can use to gain command over a case. The first one being .INPUT file or also goes by its default name hmg.input.

.INPUT file contains the basic input variables like the maximum number of steps, time-step, material properties, flow on/off, MHD on/off, levelset on/off, orthgonality correction on/off among other things.

A sample input file is given as:

```
nodes = 1,          #number of CPUs
iread = 0,          #To start from scratch and not from a restart
grid_scale = 1.0,   #grid scale will be multiplied to the dimensions
nmax  = 10000,      #Maximum number of run steps
iskip  = 100,       #Output (Tecfiles) files to be written after these many steps
nwrite  = 100,      #Restart files will be written after these many steps
dtime  = 1.0e-3,    # Time-step

visc1  = 1.0e-1,    #Viscosity of the fluid
rho1  = 1.0,        #Density of the fluid
sgmf1  = 1000,      #Electrical Conductivity of fluid
sgmw1 = 1000,       #Electrical Conductivity of solid wall1
sgmw2 = 1.0e-2,     #Electrical Conductivity of solid wall2
ubar  = 1.0,        #Initial Velocity
bval  = 1.0,        #Value of Magnetic Field
dpdx  = -82.27,     #Initial Pressure Gradient

iortho = 2,         #Orthogonal Correction Applied
igmax  = 10,        #Orthogonal Correction Iterations
nmomt  = 1,         #Momentum Equation
nppe  = 5,          #Pressure Poisson Equation
nmhd  = 5,          #MHD Poisson Equation
nheat  = 0,         #Heat Equation
ilevels= 0,         #Level Set
nskp  = 10,         #Skip steps to show Screen Outout
ipmax  = 100,       #Maximum Iterations for Pressure Poisson Equation
immax  = 100,       #Maximum Iterations for MHD Equation
epsmin = -10.0,     #Maximum Convergence Residual
```

A very basic explanation is given in the sample input file. A detailed explanation along with a whole list of keywords is given in the Appendix I "Quick Reference Guide to Input File".

The next input file of importance is updata.input. This file contains the same information as the hmg.input file but only if something needs to be updated in the middle of the run without manually interrupting the run. This file basically updates any parameter that needs to be changed and lets the rest of the parameters stay the same.

The keyword specific to this file is nstep, which indicates HIMAG that at this step some change needs to be made. As soon as the run reaches that step, the updata.input file is read and the changes written following the nstep are made. The only catch here is that this file is read at interval at which a restart file is written as mentioned in the sample file above next to the keyword ***nwrite***.

A sample updata.input file in relation to the sample hmg.input file shown looks like:

```
nstep = 2000,        #Step at which the change will be made
iskip  = 500,        #Output (Tecfiles) files to be written after these many steps
epsmin = -15.0,      #Maximum Convergence Residual
```

Since in the hmg.input ***nwrite*** = 100, this means that at every 100 run steps, the code will create a restart file and also look for the updata.input. We have mentioned ***nstep*** as 2000, so when the code reads the updata.input at step 2000, it will change the ***iskip*** value from 100 to 500 and ***epsmin*** value from -10 to -16.

The third input file is user.dat. This file is used for the sole purpose of MHD specifications. It has 10 empty places where various values can be specified depending upon the case. More description is given in the Appendix I "Quick Reference Guide to Input File". A sample of user.dat is shown here.

```
0           11      0       5000    1
80.012.7    0       0       0
```

4.3 Parallel Partitioning

For grid partitioning, HIMAG has two utilities to be used hand-in-hand, Ux2Part and MHD2Tec in the same order. ***Ux2Part*** is used to create a cell distribution (.color file) of a TEMPUS .ux grid file over a specified number of CPU's and the utility ***ColorMHD*** partitions a given mesh into nearly equal partitions, following the KMETIS algorithm.

4.3.1 Ux2Part

Ux2part is used to create a cell distribution (.color file) of a TEMPUS .ux grid file over a specified number of CPU's. The cell distribution is used by Ux2part to create grid files that are loaded on each CPU as well as information used for communication between the CPU's.

Ux2part requires all user specified information to be entered from the command line. The code automatically tries to open the TEMPUS .ux grid file and .ugp or .up unstructured grid parameters file by using the given grid base name. (The .up file is the older version of the .ugp file and the code attempts to read the .ugp file first.) If no grid parameters file is available, the graph is still generated but no PEC weighting of boundary faces is performed.

The user must specify the number CPU's to distribute the cells, cell's normal weight (>=1) and PEC weight (>=0). For each PEC face in a cell, the cell's normal weight is added by a PEC weight.

```
$ ux2part [grid_name] [N] [cell_wt]* [ipec_wt]* [icon_wt]* [iob_wt]* [intf_wt]*

where:
    grid_name       --  Base name of .ux and .ugp (or .up) files
    N               --  Number of CPU's to use (>=1)
    cell_wt         --  Normal cell weight (>=1)
    ipec_wt         --  PEC face cell weight (>=0)
    icon_wt         --  Contour face cell weight (>=0)
    iob_wt          --  Outer boundary face cell weight (>=0)
    intf_wt         --  Interior face cell weight (>=0)
   *indicates that the value is optional
```

4.3.2 ColorMHD

This utility uses the color file generated by UX2PART and creates partitioned grid and BC files for each CPU. The partitions will be numbered from 0 to N-1 where N is the number of CPUs. Following screenshots depict stages of execution, at the end of which a set of files named ***partition.mm.patch***, ***partition.mm.ux*** and ***partition.mm.ugb*** will be created where *mm* runs from 0 to N-1. Following screenshots represent the stages of execution.

4.3.3 Using Ux2Part and ColorMHD

There are three ways to use these utilities for partitioning the grid. One way is to use the HIMAG-Prep GUI and other way is to use the command line prompt and the third way is to use both together. We will be explaining all three ways here in detail.

4.3.3.1 From HIMAG-Prep GUI

From the top menu bar, select ***Parallelizing*** and choose number of CPUs under ***Set No. of CPUs***. You may select 2, 4, 8, 16, 24, 32, 64 or 128. Or you may just type your number at the lower right corner of the screen.

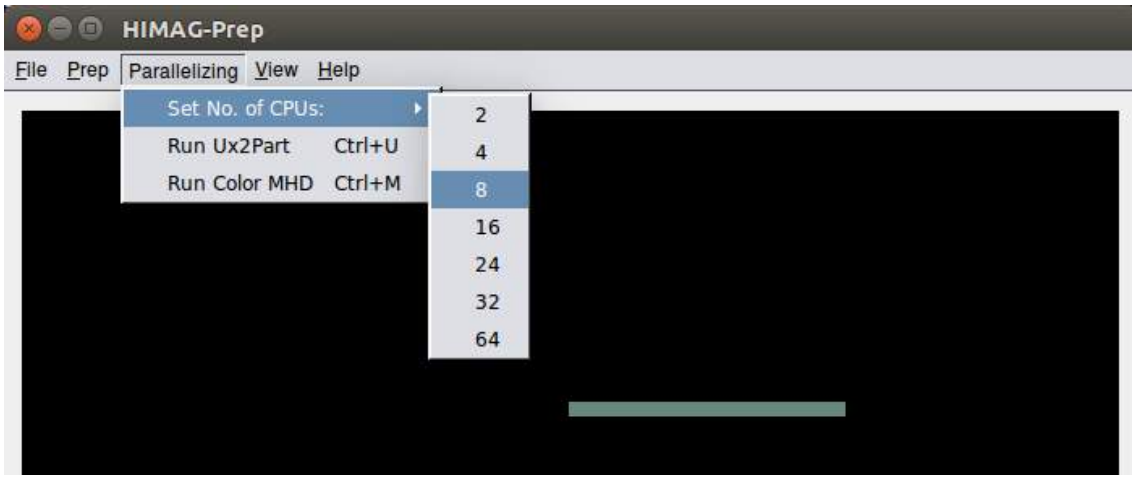


Figure 46: Partitioning using Prep GUI to set the number of CPUs

Click ***Run Ux2Part*** to run Ux2Part.

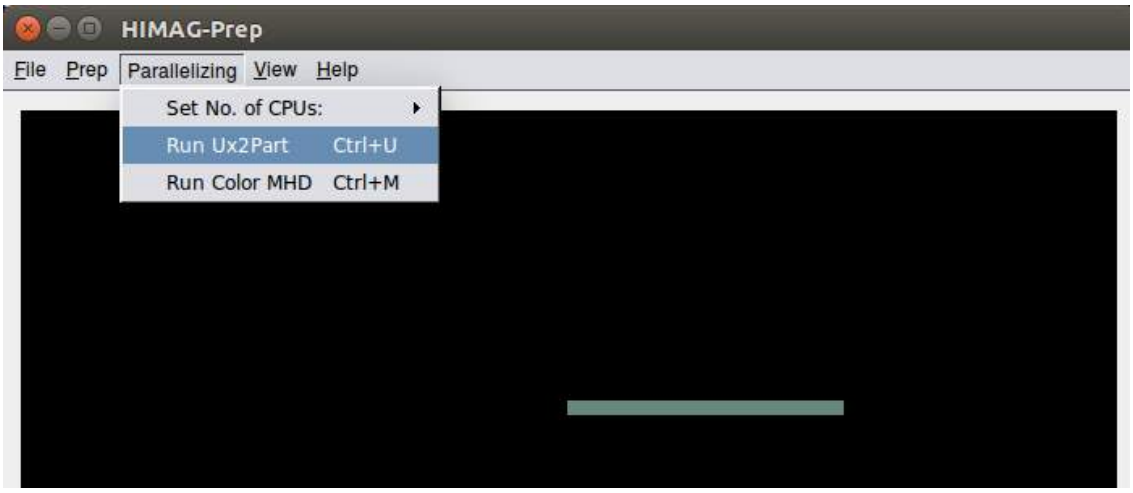


Figure 47: Running Ux2Part using Prep GUI

Next, click on ***Run Color MHD*** to run ***colormhd***.

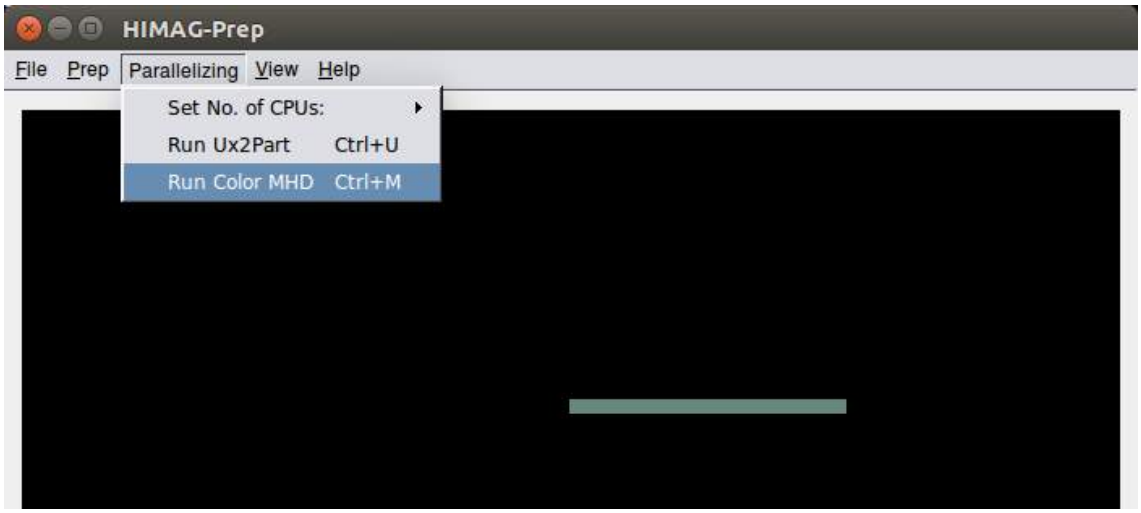


Figure 48: Running ColorMHD using Prep GUI

4.3.3.2 Command Line with GUI

Select the UX file

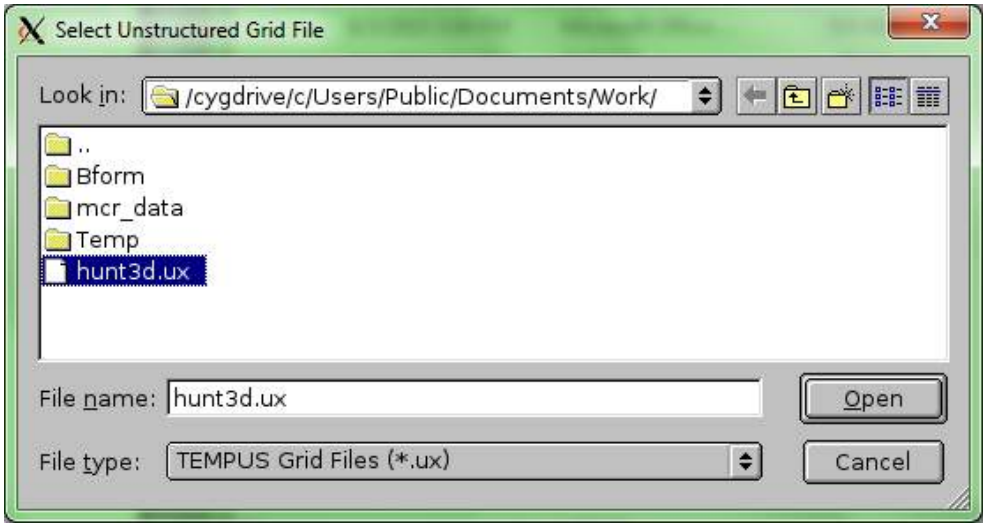


Figure 49: Select the .ux file

Enter the number of partitions to create

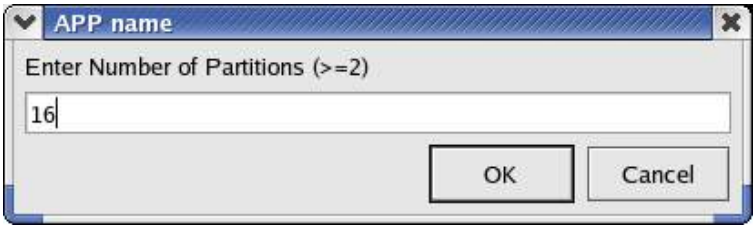


Figure 50: Enter the number of CPUs for Partitioning

Enter normal cell weight factor (≥1): Use 1 for HIMAG

Figure 51: Enter the normal cell weight

Enter PEC weight ((≥0): Use 0 for HIMAG

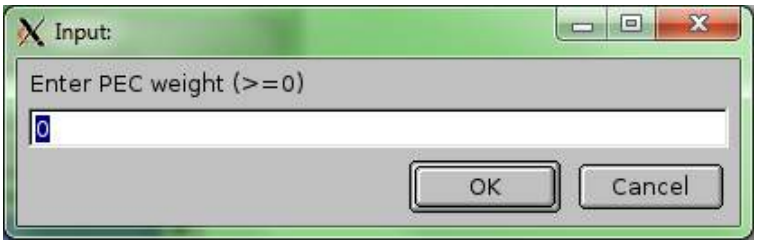


Figure 52: Enter the PEC Weight

Enter CONTOUR cell weight (≥0): Use 0 for HIMAG

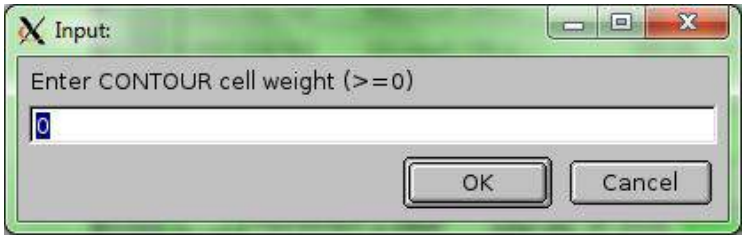


Figure 53: Enter the value for Contour Cell Weight

Enter OUTER cell weight (≥0): Use 0 for HIMAG

Figure 54: Enter the value for Outer Cell Weight

Enter INTERIOR cell weight (≥0): Use 0 for HIMAG

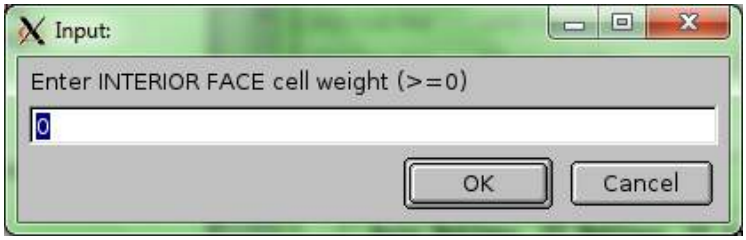


Figure 55: Enter the Interior Cell Weight

Select the *.ux* file

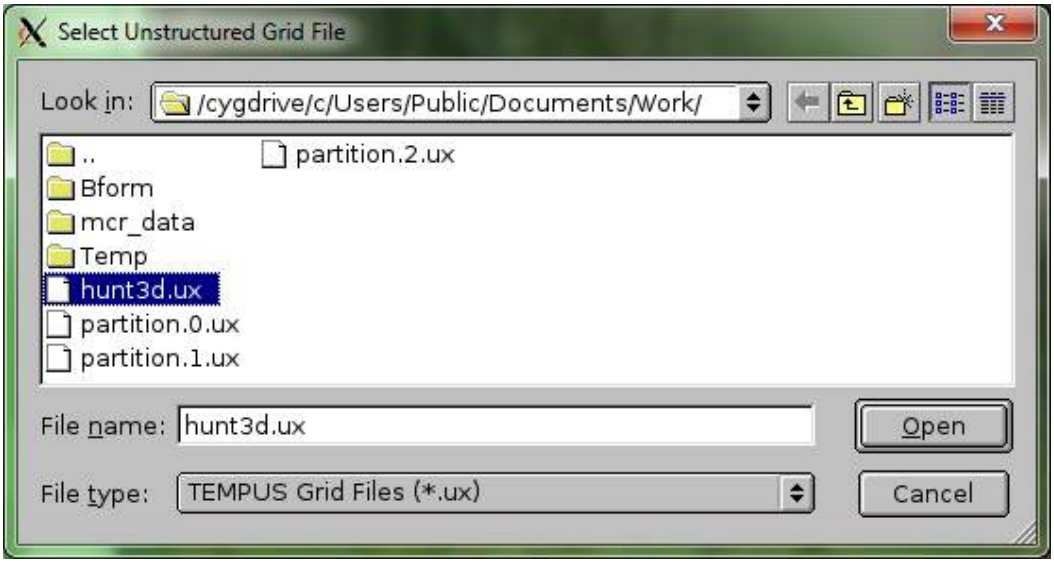


Figure 56: Select the grid.ux File

Select the *.color* file name

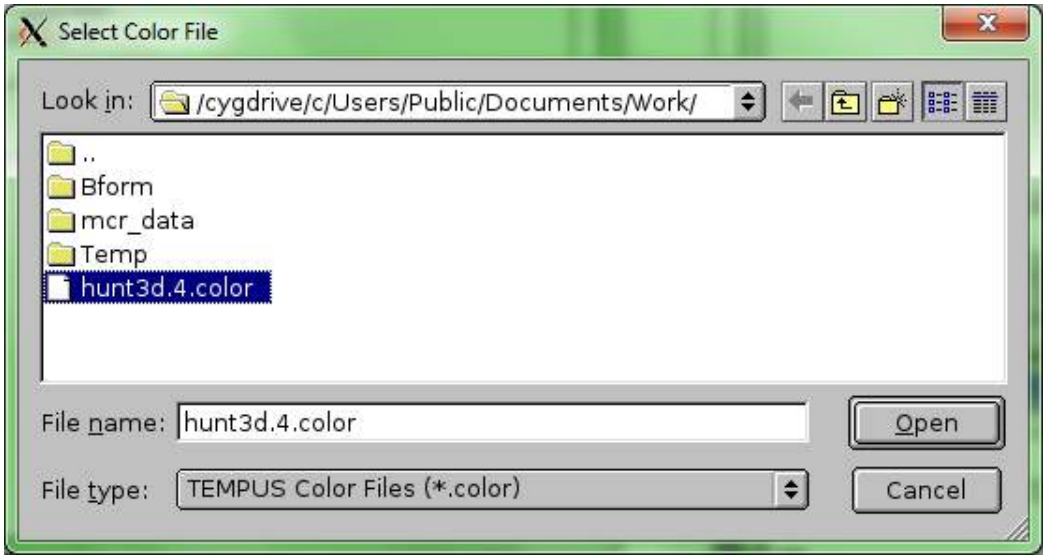


Figure 57: Select the grid.cpu.color file

4.3.3.3 Command Line Prompt

There are three ways to use these utilities for partitioning the grid. One way is to use the HIMAG

```
mhd@machine$  ux2part [gridname] [number of CPUs]
```

If no file names are given, you will be interactively prompted for them

```
mhd@machine$  colormhd [gridname].ux [gridname.nn.color]
```

4.3.4 Help Section

The command for Ux2Part help to show up is:

```
mhd@machine$  ux2part -h
```

This command will describe what the utility ColorMHD does and what the syntax is for it.

The command for ColorMHD help to show up is:

```
mhd@machine$  colormhd -h
```

This command will describe what the utility ColorMHD does and what the syntax is for it.

5 Running HIMAG

5.1 Creating HIMAG Executable File

To run HIMAG for any case, you will need the code's executable file. This assumes that you have already downloaded HIMAG (mentioned in Chapter 2) and also know the compiler present on your machine (also mentioned in Chapter 2). The steps to creating the executable are given below.

- Go to the HIMAG's library directory

```
mhd@machine$ cd ~/mhd/himag/hmglib
```

- Compile the libraries

```
mhd@machine$ make clean
mhd@machine$ make [compiler name]
```

- Go to HIMAG's source code directory

```
mhd@machine$ cd ~/mhd/himag/code
```

- Compile the source code

```
mhd@machine$ make clean
mhd@machine$ make [compiler name]
```

These steps will create the executable in the bin directory of the source code by the name ***himag_compiler***, for example himag_dod_intel_r8. This executable can be copied to the case folder to run.

5.2 Single Processor vs Multiple Processors

HIMAG can be run on a single core as well as multiple cores. The pre-processing step of Ux2Part and ColorMHD take place only when running on multiple cores. Once you are in the directory where the case setup is present, the command to run HIMAG on a single core is given below:

```
mhd@machine$ ./himag_executable –i [inputfile] -g [gridfile]
```

Where:

Himag_executable – the executable created in Chapter 5 [Creating HIMAG Executable File]

Inputfile – input file is the one which has the extension ***.input***, with all the material properties, time step created in Chapter 4 [Creating an Input File]

Gridfile – Grid file is the pair of files one of which contains the grid (extension ***.ux***) and other one contains the grid boundary conditions (extension ***.ugb***), created in Chapter 3 and 4.

```
mhd@machine$ mpirun -np [no. of cores] ./himag_executable -i [inputfile] -g [gridname]
```

Where:

No. of cores – is the number of cores/CPUs to be used for running the case

Himag_executable – the executable created in Chapter 5 [Creating HIMAG Executable File]

Inputfile – input file is the one which has the extension ***.input***, with all the material properties, time step created in Chapter 4 [Creating an Input File]

Gridfile – Grid file is the pair of files one of which contains the grid (extension ***.ux***) and other one contains the grid boundary conditions (extension ***.ugb***), created in Chapter 3 and 4.

5.3 Understanding the Screen Output

Once the case setup is complete, HIMAG is installed and we finally begin running, the terminal screen will print some data on the screen as well. Basically, the screen output is going to tell if the setup is done correctly.

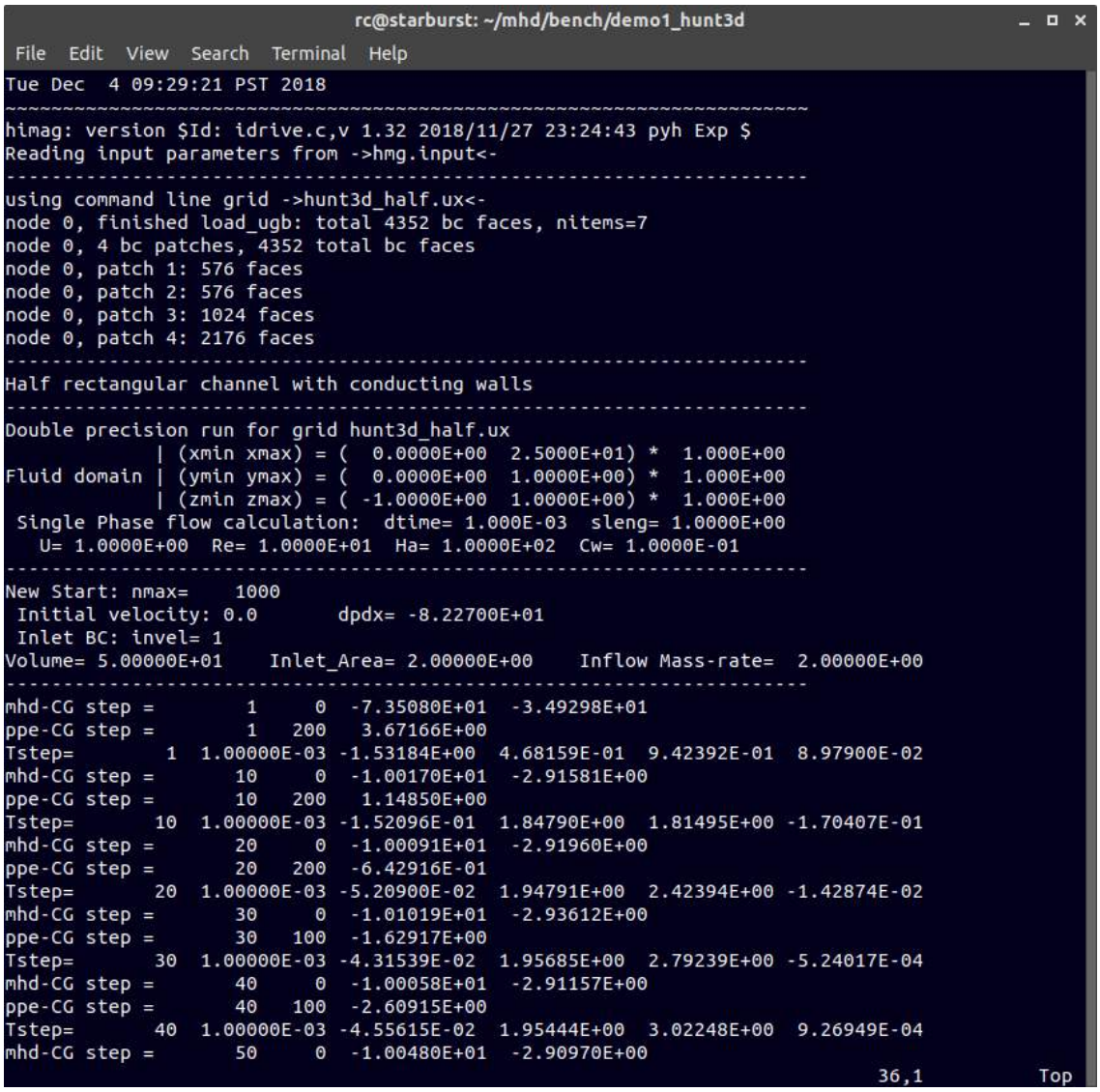


Figure 58 : Screen Output Example

For example, the screen output for one of the benchmark cases looks like **Figure 58 : Screen Output Example** But let's break it up so we can understand what each and every line tells us.

```
Tue Dec  4 09:29:21 PST 2018
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

The first line shows the Date and Timestamp.

```
himag: version $Id: idrive.c,v 1.32 2018/11/27 23:24:43 pyh Exp $
Reading input parameters from ->hmg.input<-
-----------------------------------------------------------------------
```

The next couple lines show the version number of HIMAG and indicating that ***hmg.input*** is present in the case directory and the code is now reading the values from it.

```
using command line grid ->hunt3d_half.ux<-
node 0, finished load_ugb: total 4352 bc faces, nitems=7
node 0, 4 bc patches, 4352 total bc faces
node 0, patch 1: 576 faces
node 0, patch 2: 576 faces
node 0, patch 3: 1024 faces
node 0, patch 4: 2176 faces
```

These lines show that the code has found the .ux file and as you can see, it shows only node = 0 and its patches which were created in PREP. This indicates that the case will run on a single core. If there were multiple cores, it will show node = 0 and its patches and then node = 1 and then its patches.

```
-----------------------------------------------------------------------
Half rectangular channel with conducting walls
-----------------------------------------------------------------------
```

This line tells the user about the walls used in the grid which are assigned using PREP boundary conditions.

```
Double precision run for grid hunt3d_half.ux
             | (xmin xmax) = (  0.0000E+00  2.5000E+01) *  1.000E+00
Fluid domain | (ymin ymax) = (  0.0000E+00  1.0000E+00) *  1.000E+00
             | (zmin zmax) = ( -1.0000E+00  1.0000E+00) *  1.000E+00
 Single Phase flow calculation:  dtime= 1.000E-03  sleng= 1.0000E+00
   U= 1.0000E+00  Re= 1.0000E+01  Ha= 1.0000E+02  Cw= 1.0000E-01
```

The next few lines show the Grid Dimension (Fluid Domain, xmin, xmax, ymin, ymin, zmax, zmin), Time-step (dtime), Reference length or default gridscale (sleng), initial velocity (U), Reynold's number (Re), Hartmann Number (Ha) and Length of the conducting wall (Cw).

```
-----------------------------------------------------------------------
New Start: nmax=    1000
 Initial velocity: 0.0       dpdx= -8.22700E+01
 Inlet BC: invel= 1
Volume= 5.00000E+01    Inlet_Area= 2.00000E+00    Inflow Mass-rate=  2.00000E+00
-----------------------------------------------------------------------
```

These lines indicate the data entered in the input file, like the maximum number of steps to run (nmax), initial velocity and pressure gradient (dpdx) and grid data as well, for example volume, inlet area and computed inflow mass flow rate.

```
mhd-CG step =        1     0  -7.35080E+01  -3.49298E+01
```

The next line of ***MHD-CG step*** will start showing the case progress by showing the details of solving MHD Poisson equation with first number being the step number (1 here), number of iterations to converge (0 here), residual (-7.35080E+01 here) and (-3.49298E+01 here).

```
ppe-CG step =        1   200   3.67166E+00
```

This line of ***PPE-CG step*** shows the details of solving Pressure Poisson equation with the first digit being the step number same as MHD Poisson equation (1 here), the number of iterations to reach convergence (200 here) and the residual (3.67166E+00 here).

```
Tstep=       1  1.00000E-03 -1.53184E+00  4.68159E-01  9.42392E-01  8.97900E-02
```

The next line with ***Tstep*** shows the details of Momentum equation with the first digit still being the step number (1 here), time step (1.00000E-03 here), equation residual (-1.53184E+00 here), mass flow rate (4.68159E-01 here), maximum velocity in the domain (9.42392E-01 here) and minimum velocity in the domain (8.97900E-02 here).

```
mhd-CG step =       10     0  -1.00170E+01  -2.91581E+00
ppe-CG step =       10   200   1.14850E+00
Tstep=      10  1.00000E-03 -1.52096E-01  1.84790E+00  1.81495E+00 -1.70407E-01
```

These next few lines follow the same trend as mentioned before with the next step number (step 10). Here, we have skipped the display of output between step 1 and step 10 by putting ***nskp = 10*** in the input file. It shows the MHD Poisson equation's convergence information first, then Pressure Poisson equation and last is the Momentum equation.

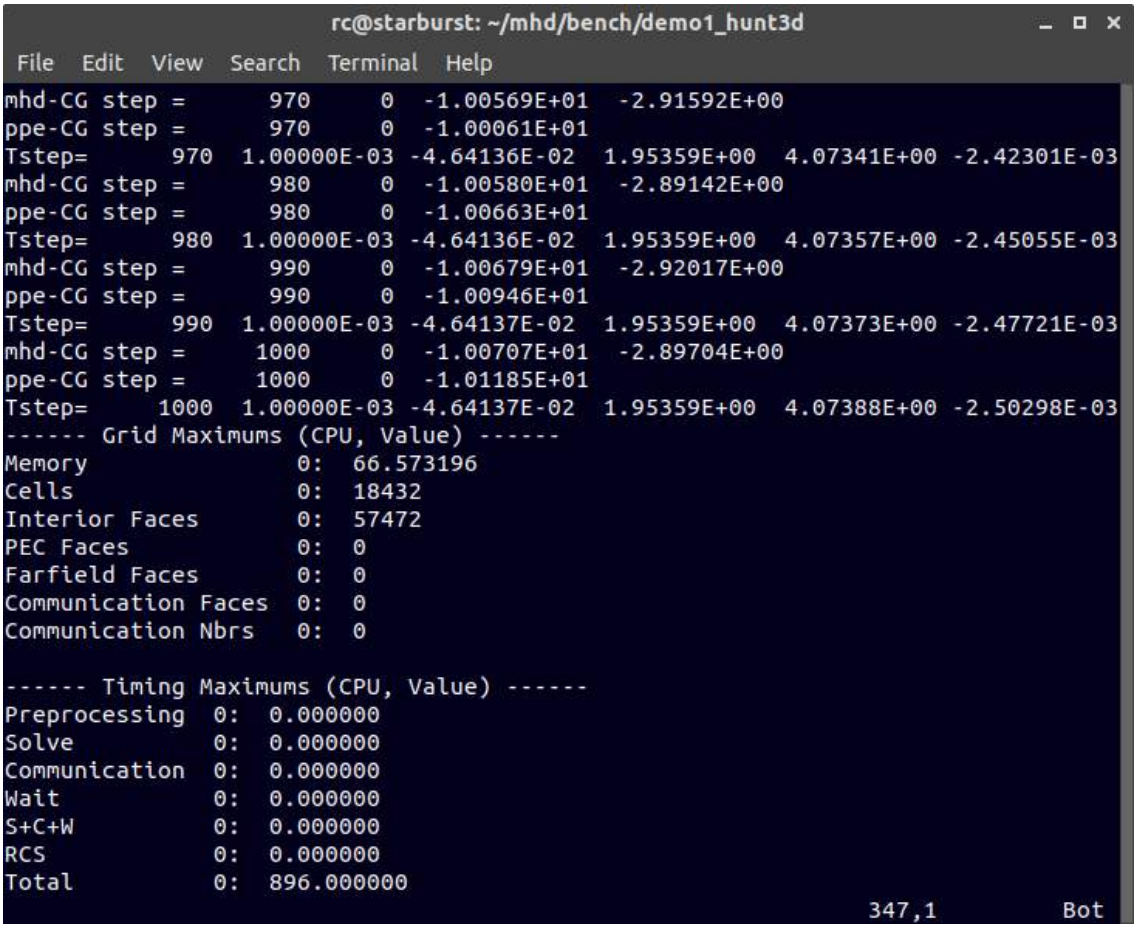


Figure 59: Screen Output

Figure 59 shows how a successful run looks like. Here in this example, we ran 1000 steps and when 1000 steps are completed, the code gives the grid description (Grid Maximums) and runtime details (Timings Maximums).

5.4 Warnings and Errors

Now, what happens when we don't have a successful run? What kind of errors and warnings will we see? And what do they mean? In this section we will talk about warnings and errors the user will see if the case setup is not correct.

1. If hmg.input is not present in the work directory, the screen output will point it out and will stop.

```
Thu Jan 31 16:27:07 PST 2019
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
himag: version $Id: idrive.c,v 1.32 2018/11/27 23:24:43 pyh Exp $
*** Cannot find input file ->hmg.input<-
    trying for hunt3d_half.input
*** can't find an input file, can't continue
```

2. If grid.ux file is not present in the work directory, or is given by a wrong name in the run command.

```
Thu Jan 31 16:26:33 PST 2019
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
himag: version $Id: idrive.c,v 1.32 2018/11/27 23:24:43 pyh Exp $
Reading input parameters from ->hmg.input<-
------------------------------------------------------------------------
using command line grid ->hunt3d_half.ux<-
error opening hunt3d_half.ux
```

3. If grid.ugb does not exist in the work directory, or is not given as the same name as the grid.ux file.

```
Thu Jan 31 16:31:18 PST 2019
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
himag: version $Id: idrive.c,v 1.32 2018/11/27 23:24:43 pyh Exp $
Reading input parameters from ->hmg.input<-
------------------------------------------------------------------------
using command line grid ->hunt3d_half.ux<-
partition 0: bc parameters file ->hunt3d_half.ugb<- not found
partition 0: bc parameters file ->hunt3d_half.ugm<- not found
```

4. If the number of cores is more than 1 in the input file but running with the single CPU command.

```
Thu Jan 31 16:27:52 PST 2019
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
himag: version $Id: idrive.c,v 1.32 2018/11/27 23:24:43 pyh Exp $
Reading input parameters from ->hmg.input<-
------------------------------------------------------------------------
should not be using MPI_Send
```

6 Post-Processing

6.1 Managing the Result Files

After HIMAG is done running, you will see hundreds and maybe thousands of files in your folder. They were all created during the process, but there are only few of them which are actually useful to us, the final result files, the restart files and the log files.

First we will talk about the final result files. Exactly, which files do we need? What do they look like? Why do we need only those? You can get the answer to all these questions in this section.

Let's talk about running the code on single processor first. After running HIMAG for 10000 iterations, you will find files like tec.001.1000.dat, tec.001.2000.dat … tec.001.9000.dat, tec.001.10000.dat. For multiple processors, the files will look like tec.001.1000.dat, tec.002.1000.dat, tec.003.1000.dat, tec.004.1000.dat … tec.001.10000.dat, tec.002.10000.dat, tec.003.10000.dat, tec.004.10000.dat where a file from each processor a particular step exists. These are the result files which will be post-processed to obtain meaningful data.

The next set of files are the restart files that go by the name ***qrestart.ooo.unf*** and ***qrestart2.ooo.unf***. Two restart files will be created on two different steps depending on what value of ***nwrite*** is specified in the input file ***hmg.input.*** You can use any of these restarts to restart a case from where it last ended, without having to run the whole case from the beginning.

The next set of files are the log files, one of which is ***op.cur***. This file contains all the values you enter in the input file but it also shows if any value was overwritten by the code itself or if you missed to specify any value which should've been specified. Another log file which is advised to be made is the one for the screen output. You should always save the screen output in a file because once the case has completed running, there is no way to look at the screen output from the beginning if there is no log file where the screen output was being written as it was being written on the screen. This log file is usually useful when running on a remote machine or a supercomputer where you cannot see the screen output. You can write a ***logfile*** using the following command.

```
mhd@machine$ ./himag_executable -i hmg -g gridname | tee Logfile
```

6.2 Introduction to MHD2Tec

MHD2Tec is a post-processing utility. This utility converts the raw output data created by each CPU into a consolidated and smoothly interpolated Tecplot data file. There are two ways to use this utility: one by the MHD2Tec GUI and the other one being the command line window. Following images depict the various steps involved, leading to the creation of files named tec.cpu.timestep.dat:

6.2.1 MHD2Tec GUI

MHD2Tec GUI can be called from the command line using the command:

```
mhd@machine$ mhd2tec
```

A Dialogue box will open asking to choose the grid.ux file for post-processing of the data files which go by the format tec.cpu.step.dat

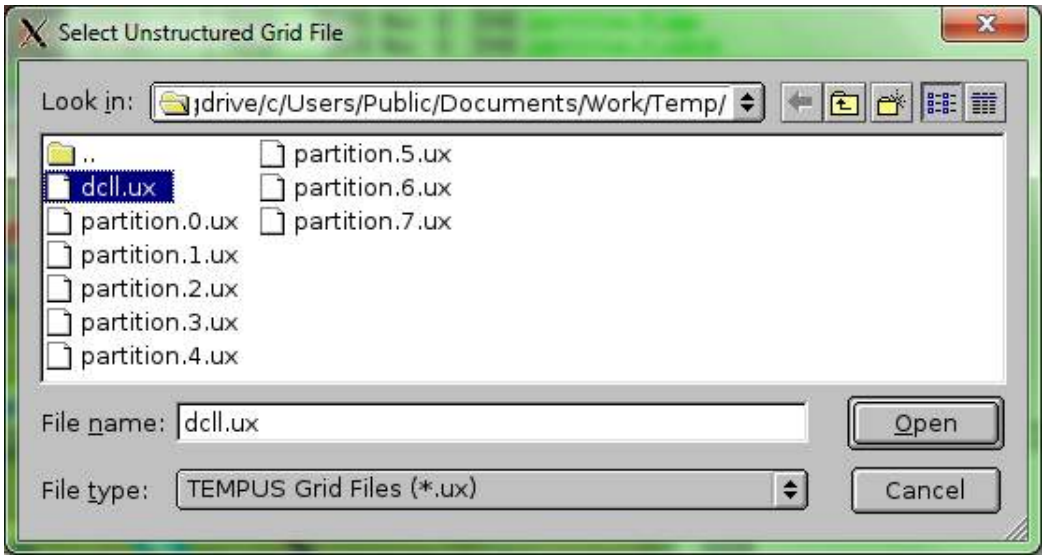


Figure 60: MHD2Tec GUI Dialogue Box

A dialogue box will pop-up (Figure 61) asking the user if the case was ran on multiple CPUs. Respond to this parallel/non-parallel question.

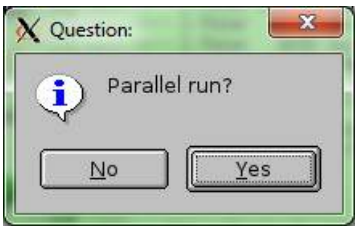


Figure 61: Dialogue Box to choose if the case was run on multiple cores

If the case was ran on multiple cores, select the color file (Figure 62)

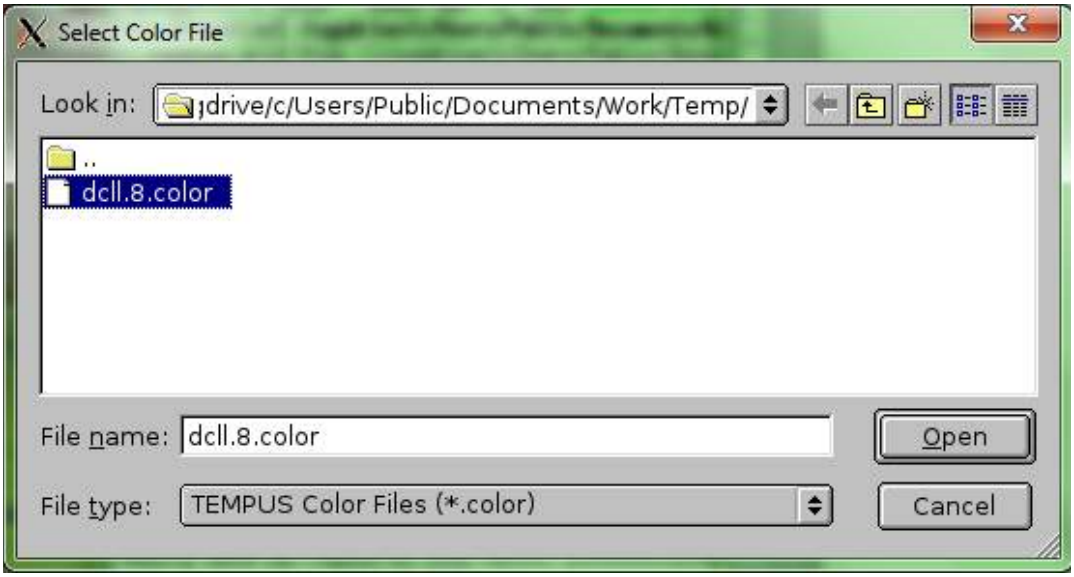


Figure 62: Choose the color file to incorporate partitioning into the process

Next pop-up box will ask if you wish to extract data at a material interface.

Figure 63: Answer the question to use the face list for any material interface

You need a .LIST file generated by PREP in order to use this feature. You will be prompted to select this file. Just in case the answer is yes in Figure 63, the next dialogue box will look like Figure 64.

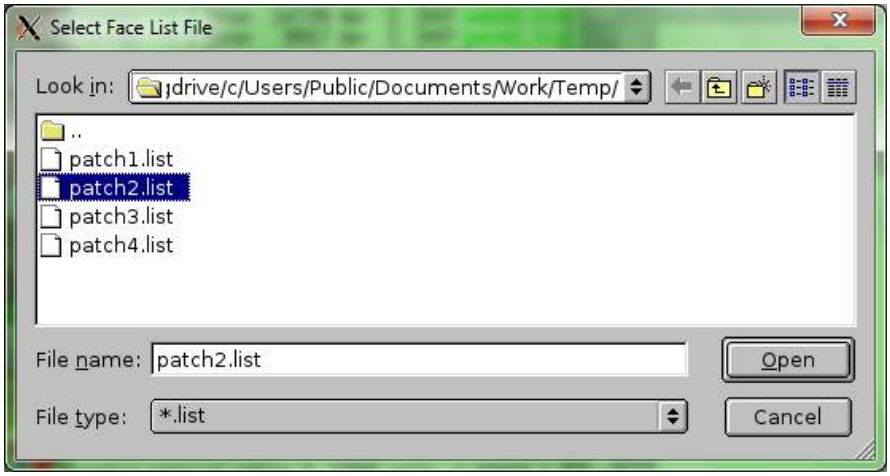


Figure 64: If the material interface we want is in Patch 2 then select Patch2.list

For most cases you respond "No" in Figure 63 to skip face list and go to next step. Next is to select the starting time-step for creating Tecplot files, followed by increments in time-step and the final time-step when to stop the data conversion process.

Figure 65: (a) Beginning Time-step; (b) Time-step increments; (c) Ending Time-step

6.2.2 Text User Interface for MHD2Tec

MHD2Tec can be called using the command line using command:

```
mhd@machine$ mhd2tec [gridname.ux] –p [gridname.np.color] –B step1
```

```
mhd@machine$ mhd2tec [gridname.ux] –p [gridname.np.color] –B [step1] –E [step2]
```

where,
GridName.ux is the name of your grid (.ux) file. If no GridName is given, the grid may be read in through the File menu.
np is the number of cores used in the grid partitioning
.color is created after running Ux2Part as shown in Chapter 4 (Partitioning)
step1, step2 are the start and skip step

ColorMHD has the following options available for the user for post-processing in various ways:

- -p ***colorfile*** activates parallel run. Colorfile (*.cpu#.color) is required
- -f ***facelist*** activates facelist usage. Facelist file required
- -E ***TimeStep_E*** timestep end *required if Timestep_end > Timestep_begin
- -S ***TimeStep_S*** timestep skip *required if Timestep_end > Timestep_begin

Irrespective of what method we use for post-processing, in the end we will end up with one compiled data file for each step in the format **grid.tec.step.dat** which can then be used to view the results using Tecplot, talked about in the next section.

Help for mhd2tec

```
mhd@machine$ mhd2tec -h
```

6.3 Viewing Results in Tecplot

This section will solely talk about the HIMAG aspect of Tecplot and in no way should be considered a tutorial to using Tecplot. The Tecplot version that is used over here is Tec360 2017.

For viewing the result files that we generated in the previous section of MHD2Tec, the command entered in the command window is:

```
mhd@machine$ tec360 [grid.tec.step.dat]
```

This will open the result file in Tecplot to view the data we generated using HIMAG. One example of this is given in figure.

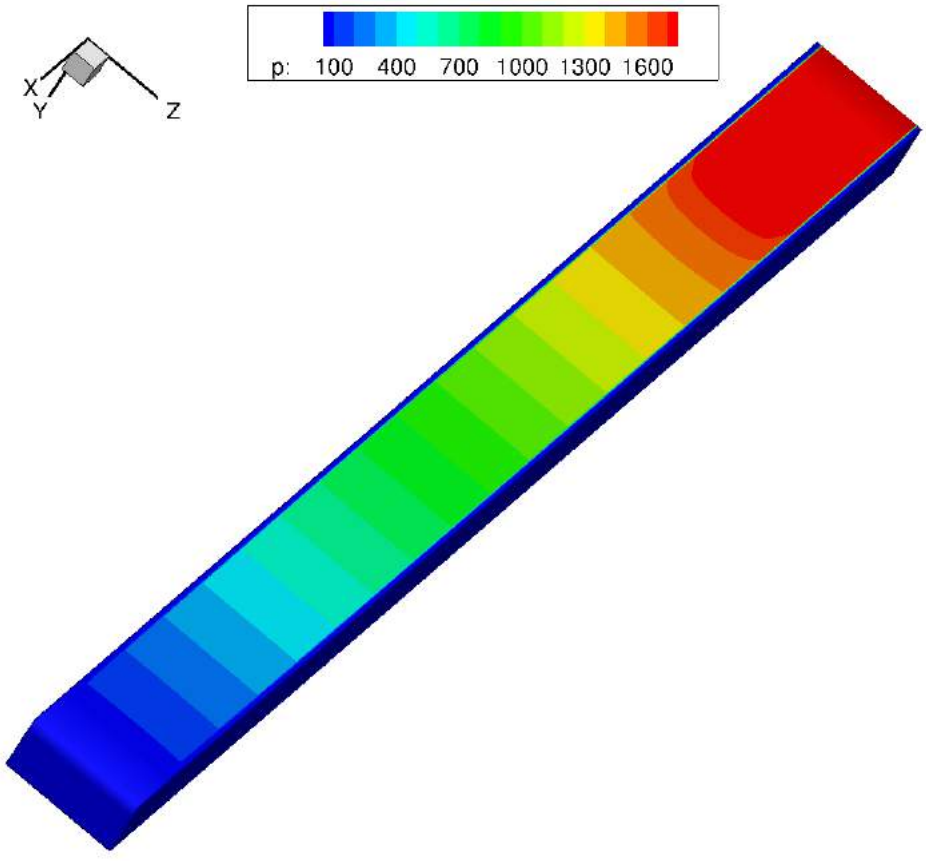


Figure 66: Example of Result File opened in Tecplot

This command might have excessive memory usage if the grid size is too big. A Tecplot tool, Preplot, provides a solution to this which is compatible with the result data file created by MHD2Tec. This tool can be found in the bin directory of the Tecplot installation. Using a terminal or command line window, enter the input ASCII file next to the name "preplot".

```
mhd@machine$ preplot [grid.tec.step.dat]
```

This method skips the "convert to binary" step normally performed during the loading of the ASCII files from the Tecplot 360 application. Now, with a copy of the file in binary format, time is saved on the initial load operation and the file size is a lot smaller. (Tecplot360)

```
mhd@machine$ tec360 [grid.tec.step.plt]
```

6.4 Viewing Results in ParaView

ParaView is an equally compatible data viewing software as Tecplot and can be accessed using the command window:

```
mhd@machine$ paraview
```

This command will open the ParaView interface for the user to explore.

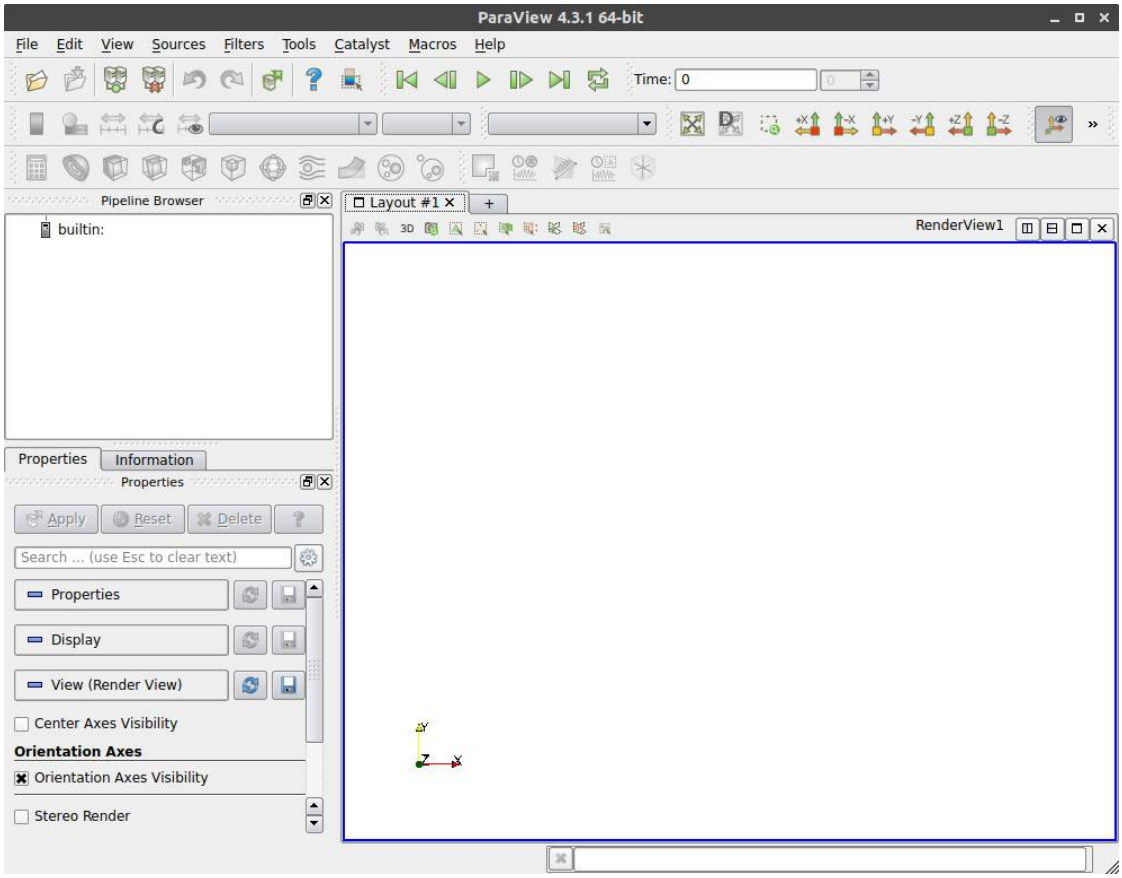


Figure 67: ParaView Blank Interface

For viewing the result files that we generated in the previous section of MHD2Tec, the command for the command window is:

```
mhd@machine$ paraview [grid.tec.step.dat]
```

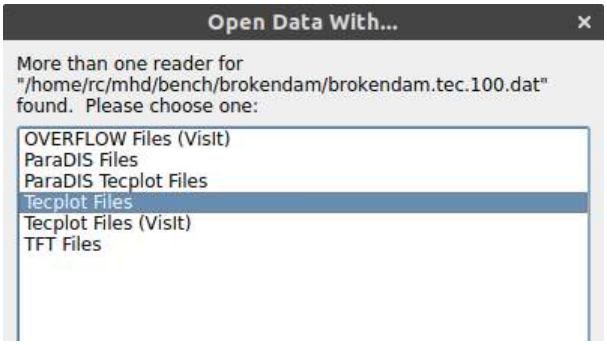


Figure 68: ParaView dialogue box to choose to open the data file in a particular format

Here, we choose Tecplot Files Option to open the *.dat* file format, which will load the file in ParaView for us to view as shown in Figure 69.

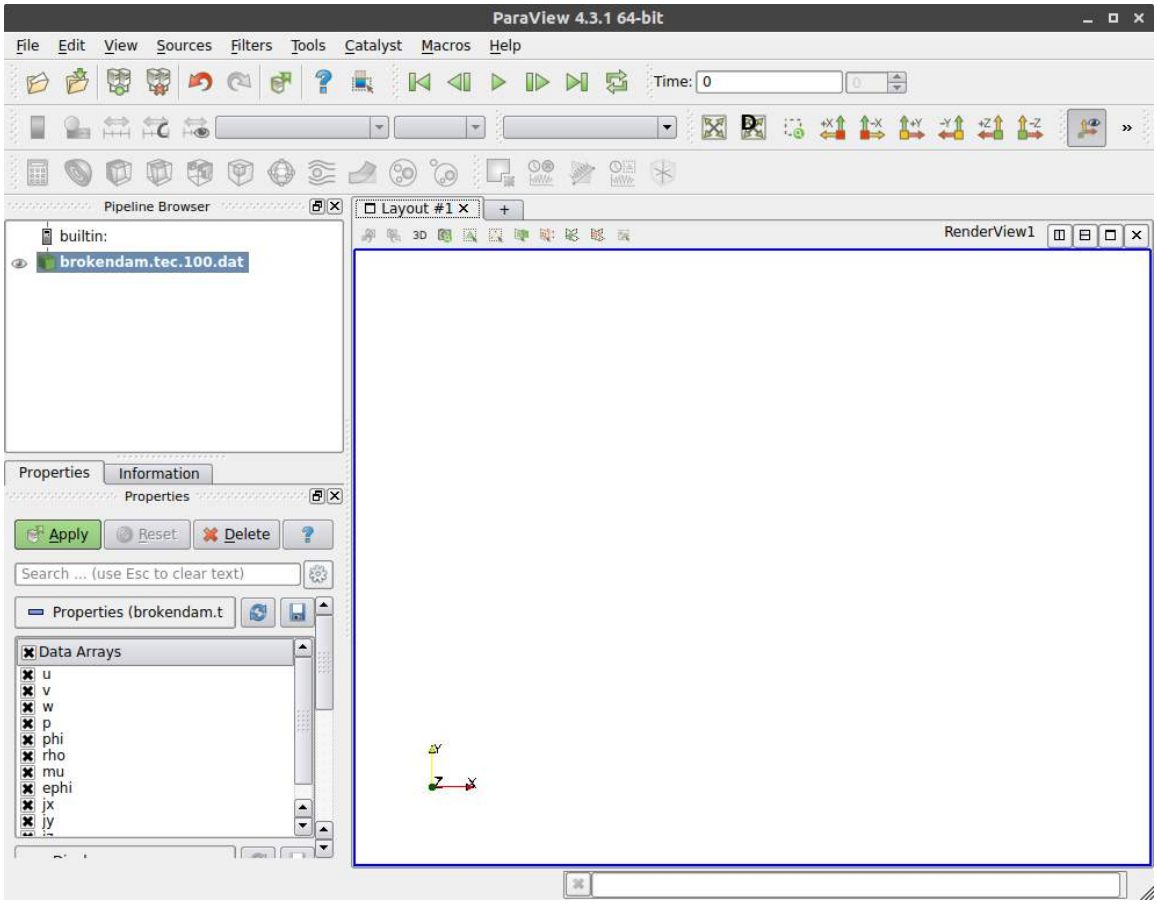


Figure 69: File loaded in ParaView

Click Apply to view the data, which here for *brokendam* case will look like Figure 70. This figure shows the Density (*rho*) contours of the materials used in the *brokendam* case.

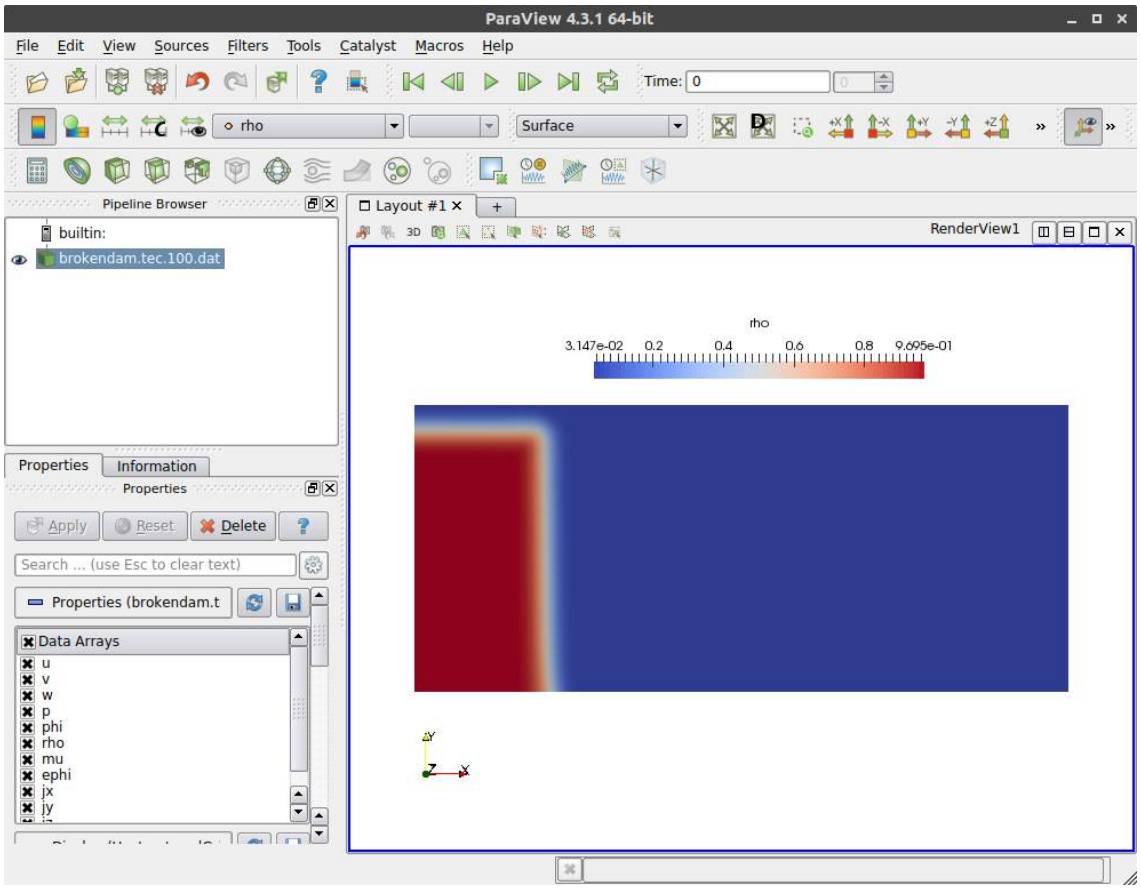


Figure 70: Viewing data file in Paraview

Other properties can be viewed using the dropdown box as shown in Figure 71.

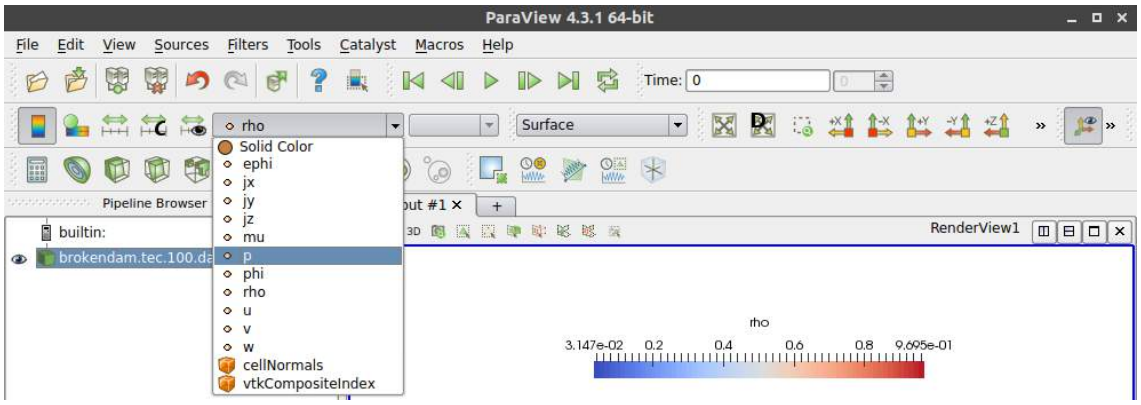


Figure 71: Dropdown box for viewing other properties

7 Tutorials

7.1 Tutorial 1: Basic Fluid Flow

In this tutorial, we will go step by step through the process of setting up a case, running HIMAG and post-processing results. This case is one of the benchmark cases present in the bench folder of HIMAG by the name *ductful*.

Firstly, we will start with the grid. Creation of *gridname.xyz* file is the first step, which for this case is called ***ductful.xyz***.

```
nopt,   nout
  1        0
x-axis: Number of segments (nxsgm),   isymx,  xmin,   xmax
                                        1        0       0.0     1.0
#1:  nxcel,     segment-Xmax
        1       1.0
     Stretch_option, stretching_factor
        0
y-axis: Number of segments (nysgm),   isymy,  ymin,   ymax
                                        2        1      -1.0     1.0
#1:  nycel,     segment-Ymax
        5       -0.9
     Stretch_option, stretching_factor
        0
#1:  nycel,     segment-Ymax
        30      0.9
     Stretch_option, stretching_factor
        3
        1.125
z-axis: Number of segments (nzsgm),   isymz,  zmin,   zmax
                                        2        1      -1.0     1.0
#1:  nycel,     segment-Ymax
        10      -0.75
     Stretch_option, stretching_factor
        0
#1:  nycel,     segment-Ymax
        20       0.75
     Stretch_option, stretching_factor
        3
        1.10
```

Once we have the ductful.xyz file ready, then we run XYZ to create a grid using the mentioned parameters.

```
mhd@machine$ xyz
Enter filename [.xyz]: ductful
```

The screen output for the *xyz* command is shown in Figure 72.

```
------------------------------------------------------------------------
  Enter filename[.xyz] or -h for help:
ductful
------------------------------------------------------------------------
  Generate grid file from ductful.xyz
  Read from ductful.xyz: nopt,nout=  1  0
========================================================================
Done for (smin, smax)=(   0.0000       1.0000    )
Done for (smin, smax)=(  -1.0000       1.0000    )
Done for (smin, smax)=(  -1.0000       1.0000    )
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Write grid log to checkXYZ.dat for nopt=  1
 Check array size: jgrd,kgrd,lgrd=           2          41          41
 Output grid to file: univ.cemgrd
 Done writing file: univ.cemgrd
 Grid generation completed!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 72: Screen Output for xyz – grid generation command

This will create a ***univ.cemgrd*** file and ***checkXYZ.dat*** file. We can open the checkXYZ.dat file to see if the spacing between cells is reasonable. The checkXYZ.dat file for this case looks like this: (the complete file is not shown, only the start and end is shown here)

(a)

```
------------------------------------------------
  grid       yf            dy           yc
    1 -1.000000E+00  2.000000E-02 -9.900000E-01
    2 -9.800000E-01  2.000000E-02 -9.700000E-01
    3 -9.600000E-01  2.000000E-02 -9.500000E-01
    4 -9.400000E-01  2.000000E-02 -9.300000E-01
    5 -9.200000E-01  2.000000E-02 -9.100000E-01
```

(b)

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Multi-segments HEXAHEDRAL grid (nopt= 1) with ductful.xyz
 min: dx= 1.00000E+00   dy= 2.00000E-02   dz= 2.50000E-02
 max: dx= 1.00000E+00   dy= 9.53378E-02   dz= 1.24625E-01
------------------------------------------------------------------------
 Domain range: x[    0.0000    1.0000]
               y[   -1.0000    1.0000]
               z[   -1.0000    1.0000]
 Total number of nodes (  2 x 41 x 41) =   3362
 Total number of cells (  1 x 40 x 40) =   1600
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 73: CheckXYZ.dat file (a) - start of the file and (b) - end of the file

Now that we have *univ.cemgrd* file and everything looks okay in *checkXYZ.dat*, we use *Book* command to create *.ux* file for the grid which will be used by HIMAG.

```
mhd@machine$ book ductful.ux
```

This command will give us ***ductful.ux*** file which can be opened in *HIMAG-Prep* to setup boundary conditions, which is our next step.

To open this file in *HIMAG-Prep*, let's type in the command for *Prep* GUI.

```
mhd@machine$ prep
```

Choose the grid file, i.e. *ductful.ux* in the *HIMAG-Prep* window.

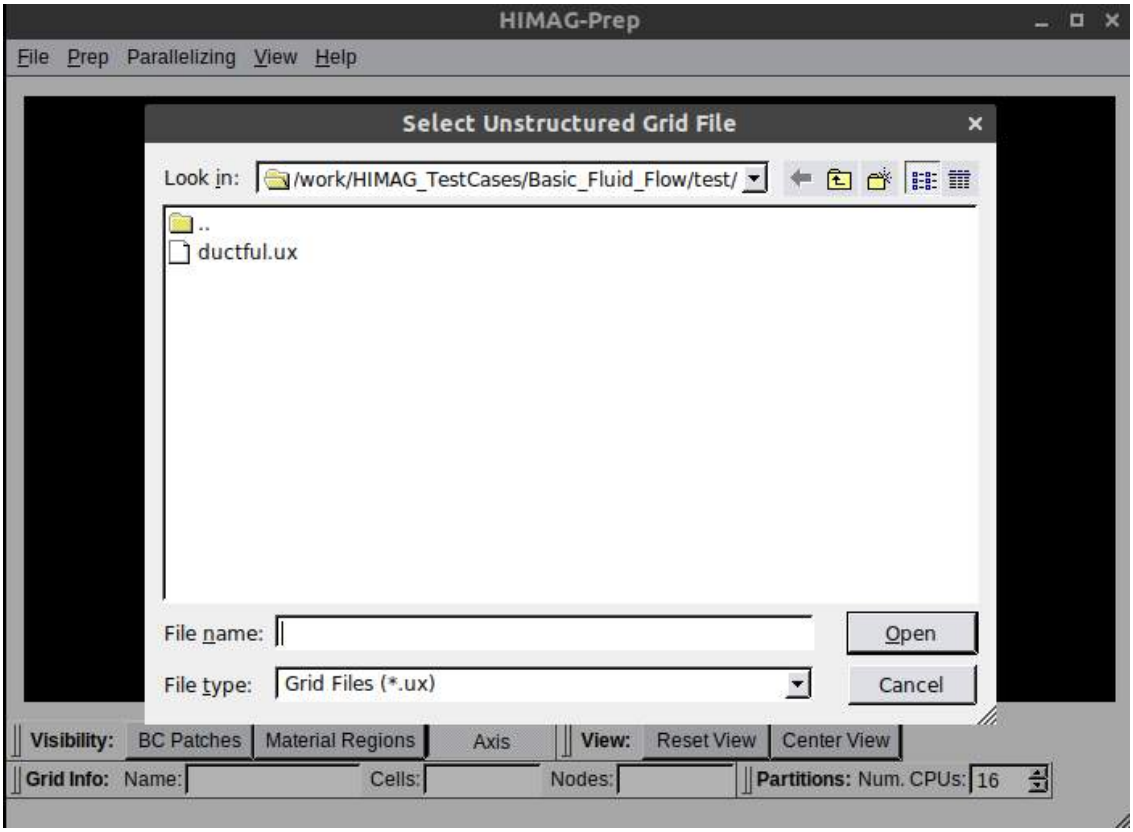


Figure 74: HIMAG-Prep grid selection

The next window will show the work window of *HIMAG-Prep* which will be empty with no grid but only grid data shown in the tabs below with the name, number of cells and nodes (Figure 75).

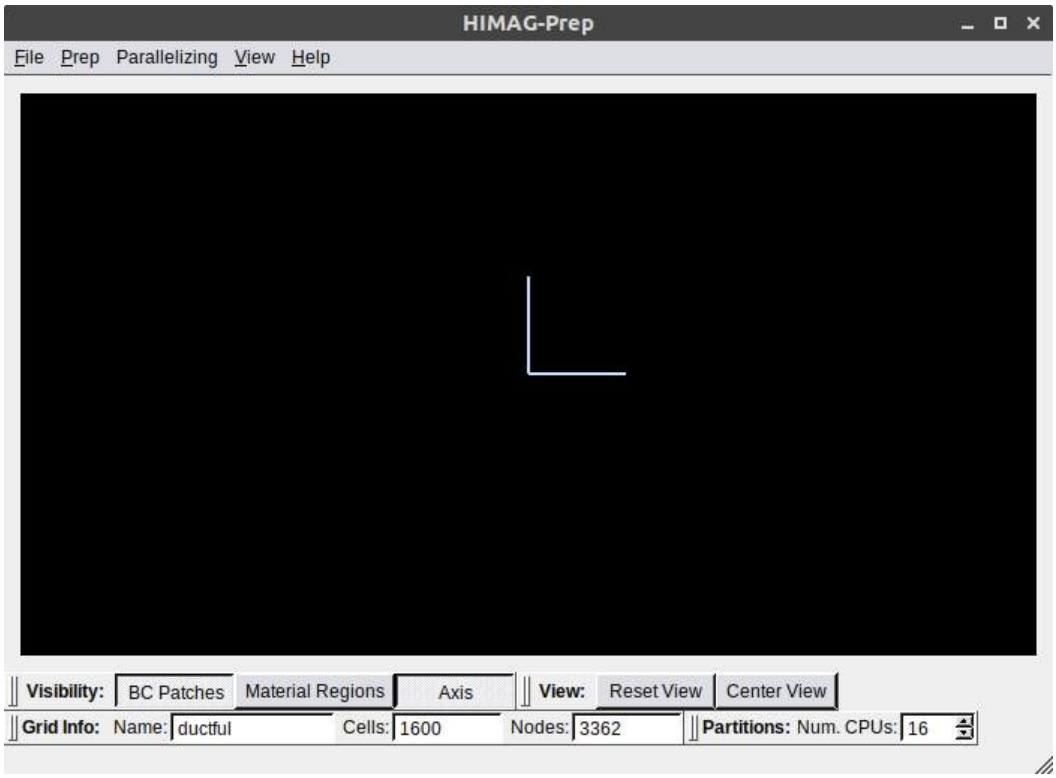


Figure 75: HIMAG-Prep work window

Select *Prep* on the top-left corner of the window and select *Boundary Conditions* under *Prep*. Another window will pop-up for *Boundary Conditions* as discussed in Chapter 4.

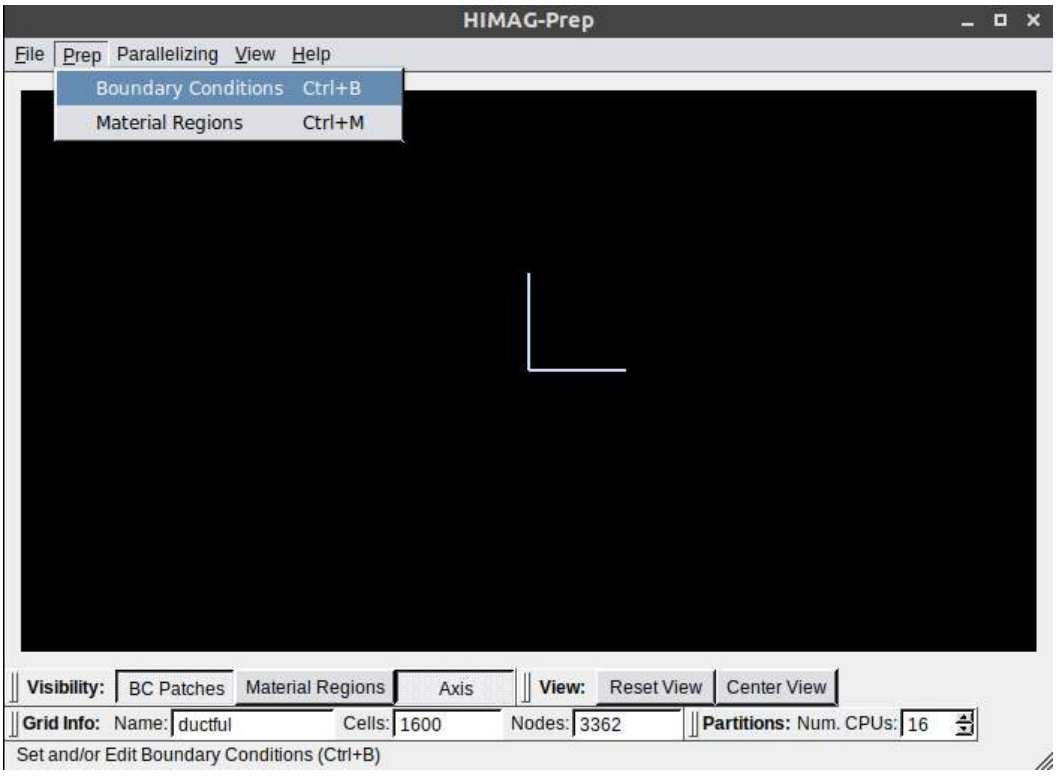


Figure 76: Prep work window showing the location of Boundary conditions tab.

Select *Create* and then *From Cutting Plane* in Figure 77.

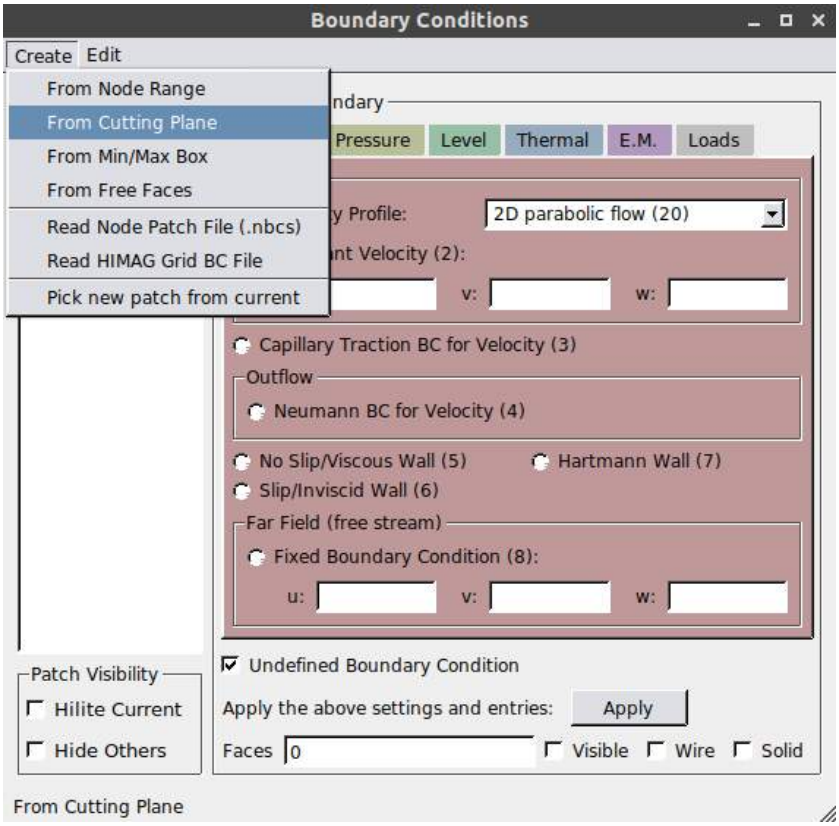


Figure 77: Creating a patch using Cutting Plane in HIMAG-Prep

A pop-up dialogue box (Figure 78) will open asking for the plane where the patch will be created. Select x.

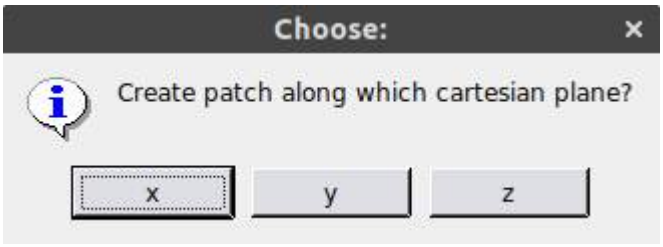


Figure 78: Pop-up dialogue box asking to select the plane

Next, you will be asked to enter tolerance. Let it stay the default value of 0.001.

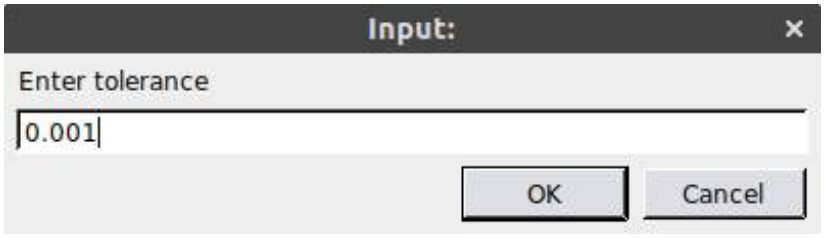


Figure 79: Dialogue box asking for tolerance

The next dialogue box will be to input the x plane value where the cut will be made for the patch to be made. Enter the value, 0.0.

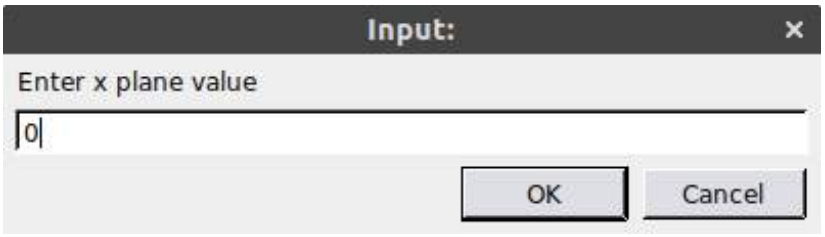


Figure 80: Dialogue box asking for plane value

The next thing you will notice is a patch made in the *HIMAG-Prep* work window.

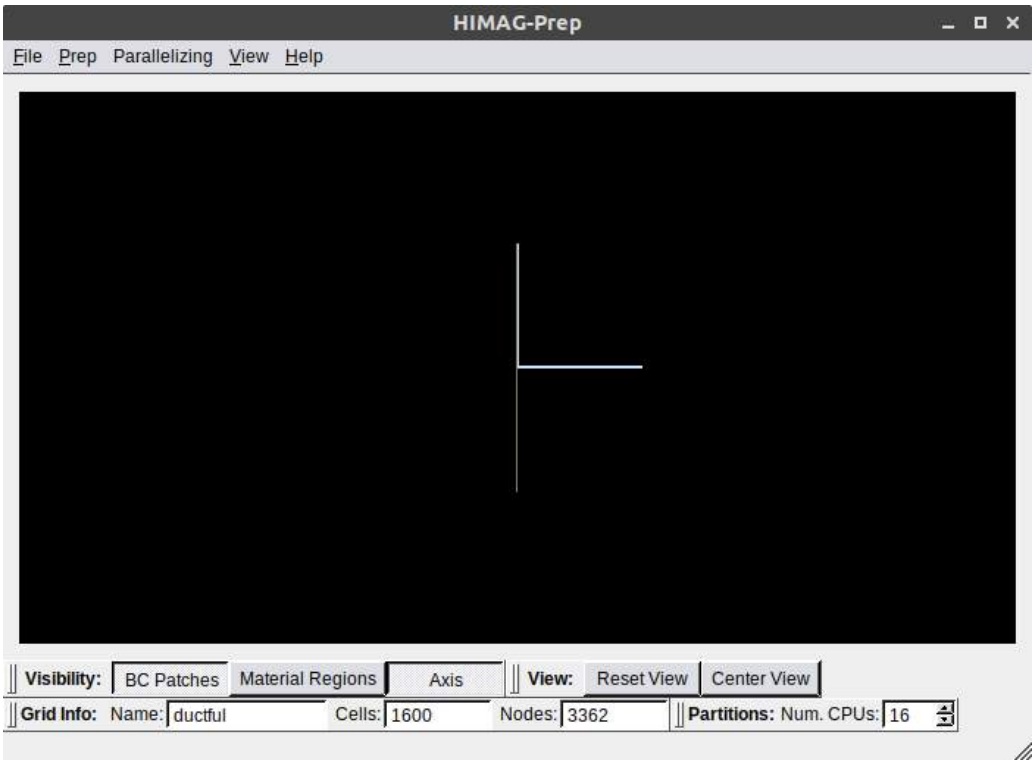


Figure 81: HIMAG-Prep work window after the first patch is created

A patch entry in the *Boundary Conditions* window which will go like 2:2:2:2:2:2.

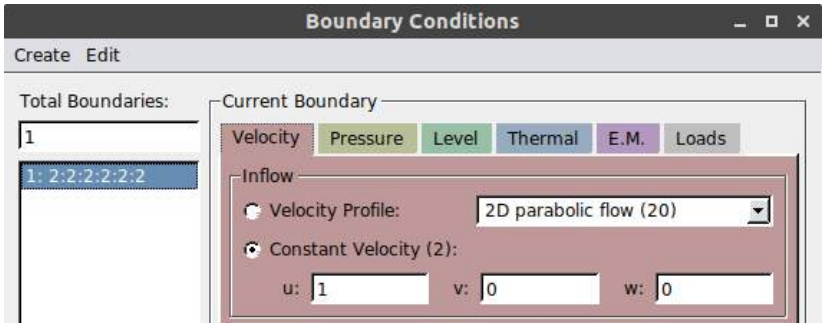


Figure 82: Boundary Conditions window after the first patch is created

Similarly, another patch is created at x = 1. Select *From Cutting Plane* under *Create* in the *Boundary Conditions* window. Again, choose x-axis for the plane for creating the patch. Let the tolerance remain as it is and enter the x-plane value to be 1.0 and click on OK (Figure 84) to see another patch form in the *HIMAG-Prep* work window (Figure 83).

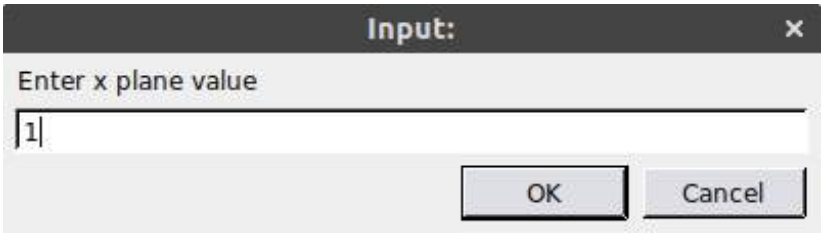


Figure 84: Dialogue box asking for plane value

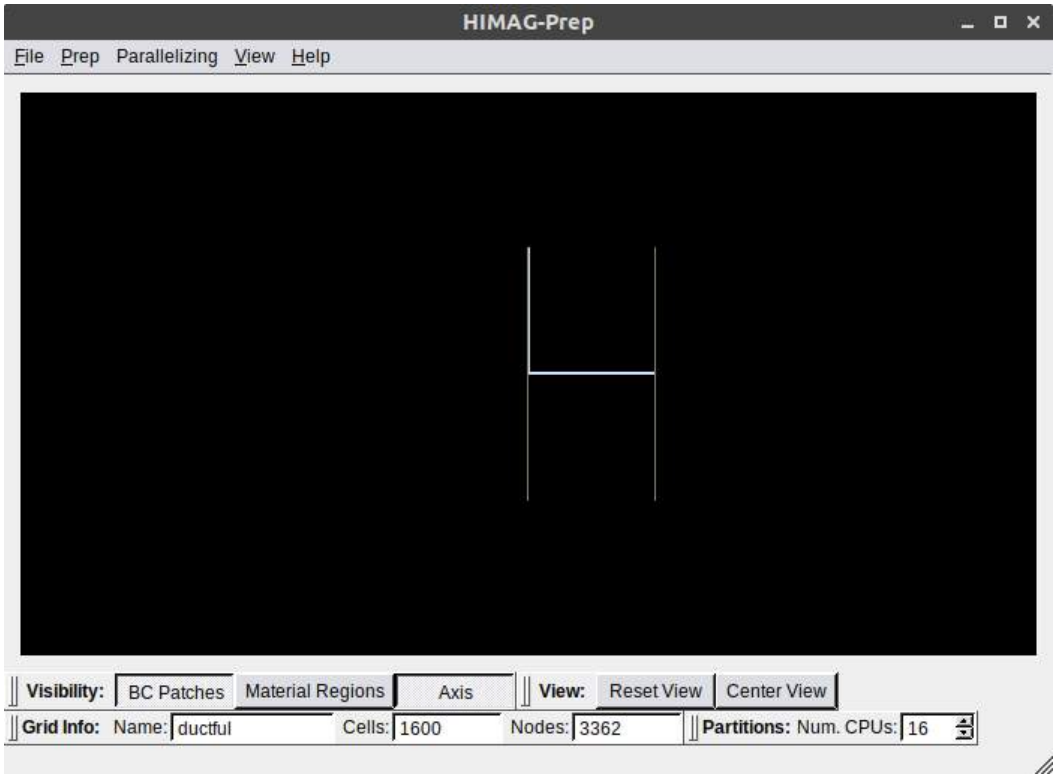


Figure 83: Prep window after second patch is created

The next step is to create the remaining faces which can be created by selecting *From Free Faces* under *Create* tab on *Boundary Conditions* window. This will form another patch with all the remaining faces and will complete the grid. The completed grid will look like Figure 86.

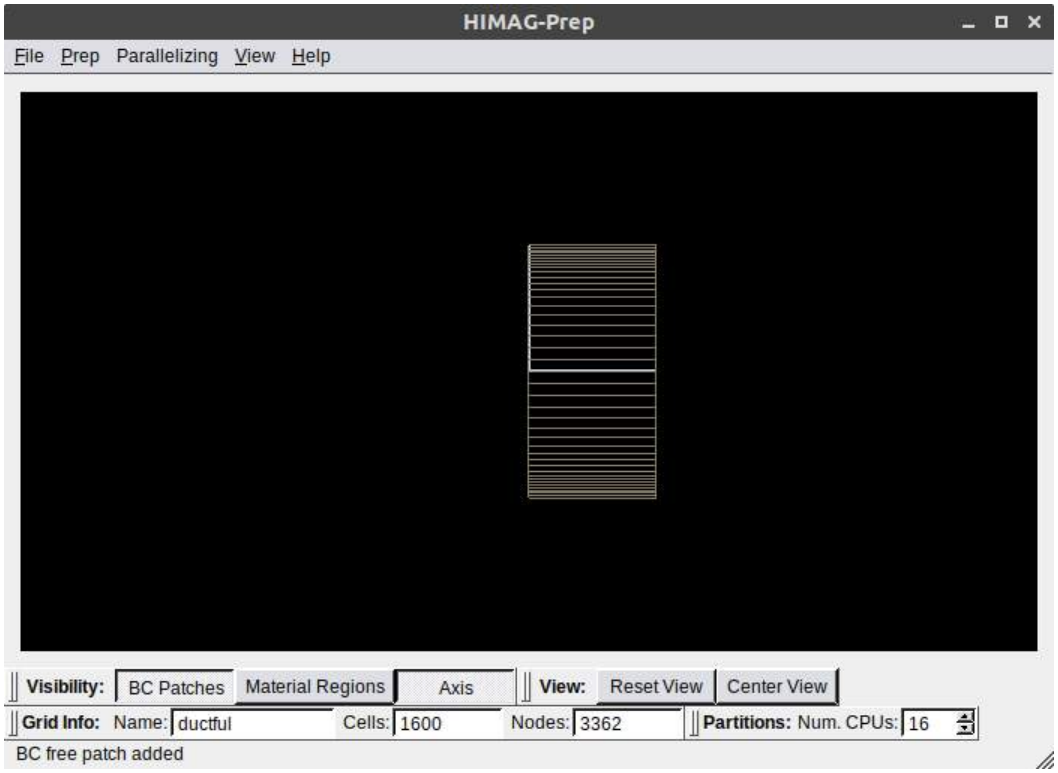


Figure 86: After all the patches are created

The next step is to assign boundary conditions. Let's start with patch 1. Under Velocity tab, select Outflow – Neumann BC for Velocity (4) and click Apply. We will notice that as soon as we hit Apply the loads tab selects Don't Include Force/Moment Calcs (0) on its own.

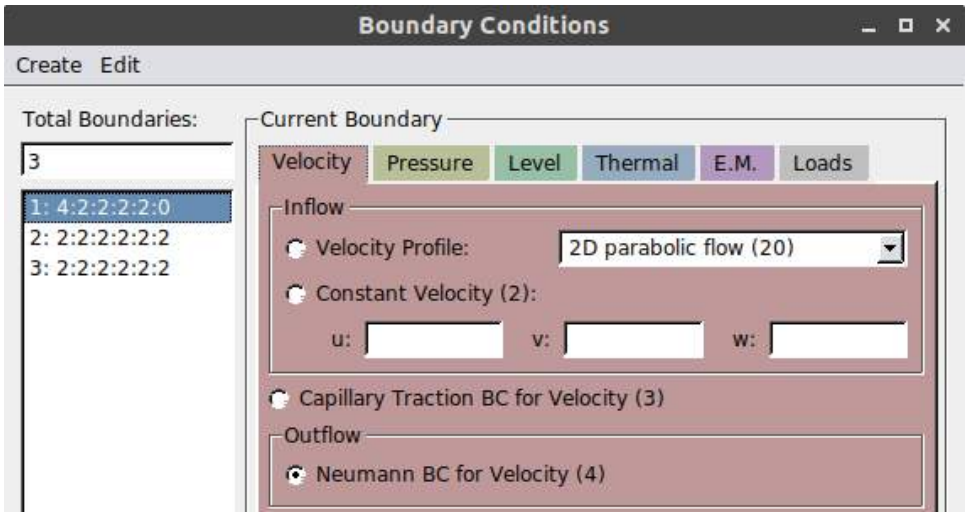


Figure 85: Velocity Boundary Conditions for Patch1

Next is our patch 2. We will again start with the velocity tab, where we will choose Neumann BC for Velocity (4) under Outflow (Figure 87).

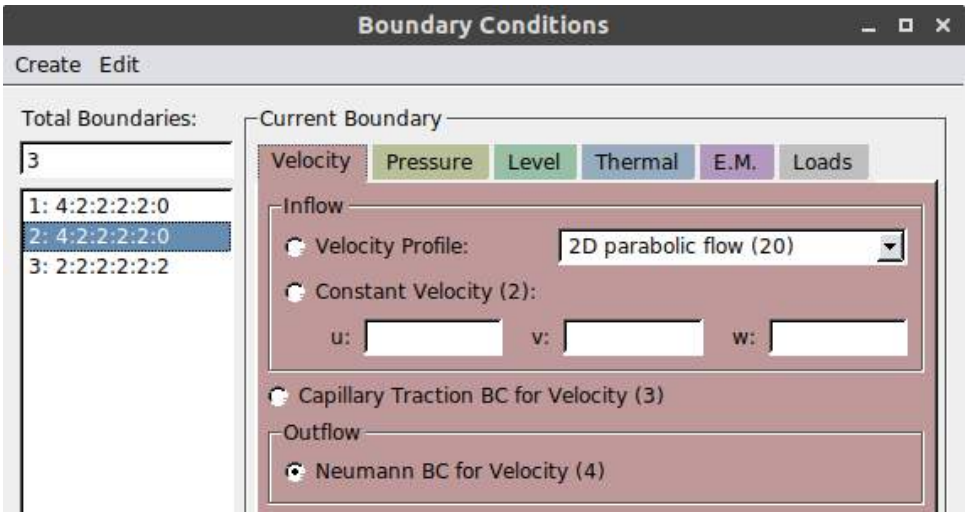


Figure 87: Velocity Boundary Condition for Patch2

Moving on to patch 3, which is the wall in this case. So, in velocity tab, we select No Slip/Viscous Wall (5) and hit Apply (Figure 88).

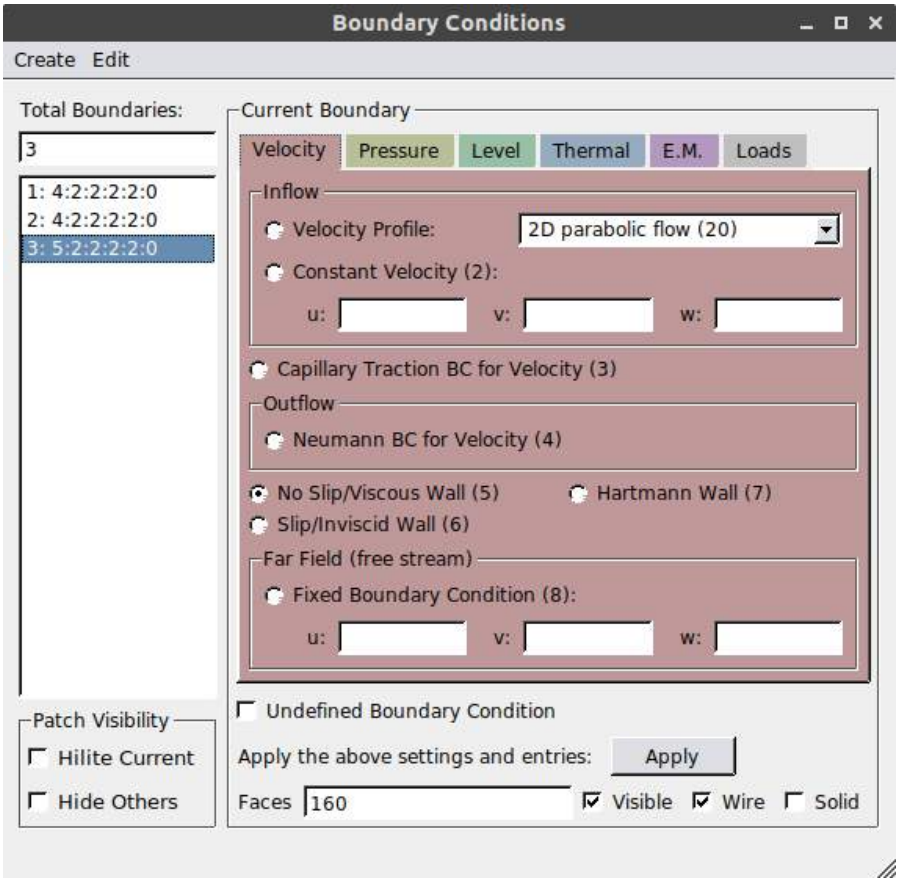


Figure 88: Velocity Boundary Condition for Patch3

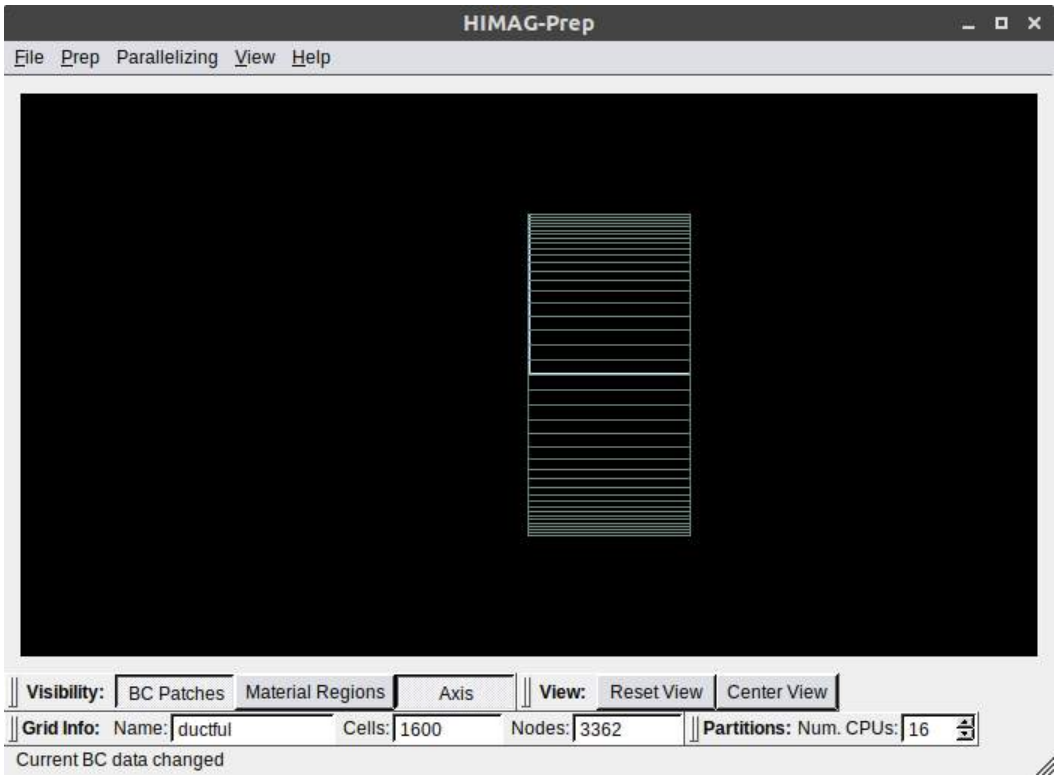


Figure 89: Grid after setting the boundary conditions in Prep

Before closing the window, click on *File* in the *Prep* work window located on the top-left corner of the window. Select *Save .ugb File* in the drop down menu. It will save the .ugb file with the same name as the .ux file, i.e. ***ductful.ugb***.

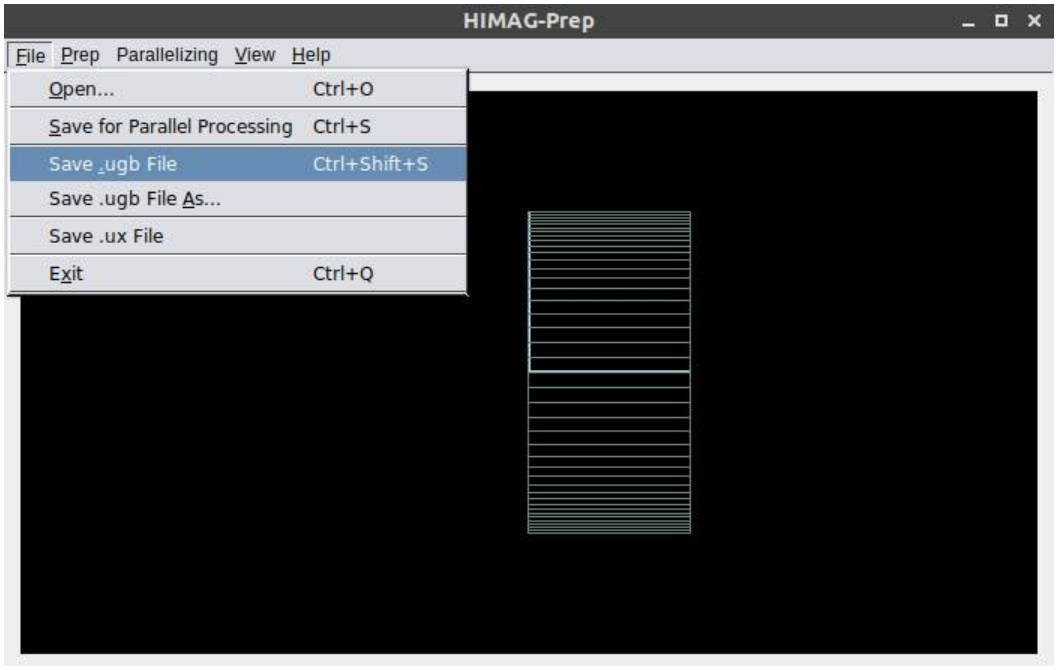


Figure 90: Save to create .ugb file

Now, we have the ugb file as well. The next step is to create an input file, ***hmg.input***. This file will look like:

```
nodes    = 1,          #number of CPUs
iread    = 0,          #0 for reading no restart file
grid_scale = 1.0,      #grid scale will be multiplied to the dimensions

nmax     = 2000,       #Maximum number of run steps
iskip    = 2000,       #Output (Tecfiles) files to be written after these many steps
nwrite   = 2000,       #Restart files will be written after these many steps
dtime    = 1.0e-2,     # Time-step

#Material Properties
visc1    = 1.0e-1,     #Viscosity of the fluid
rho1     = 1.0,        #Density of the fluid

#Numerical Parameters
ubar     = 1.0,        #Initial Velocity
dpdx     = -82.27,     #Initial Pressure Gradient
iortho   = 1,          #Orthogonal Correction Applied
nvel     = 1,

#Solver Options
nmomt    = 1,          #Momentum Equation
nppe     = 0,          #Pressure Poisson Equation
nmhd     = 0,          #MHD Poisson Equation
nheat    = 0,          #Heat Equation
ilevels  = 0,          #Level Set
ichan    = 0,
epsmin   = -16.0,      #Maximum Convergence Residual
```

Once the input file is ready, we are ready to run HIMAG for this case. Assuming that HIMAG is already built on the machine and the executable file is available for use and added to the path. The command that is used to run HIMAG is:

```
mhd@machine$ himag -i hmg -g ductful | tee logfile
```

This will start running HIMAG for the case demo1_hunt3d and the screen output will look like Figure 92 and when the case finishes, the screen output will look like Figure 91. This means that the output files have been generated and are ready to be post-processed.

```
HIMAG MPI run with 1 node(s) started - node 0 on starburst
himag: version $Id: idrive.c,v 1.34 2019/02/15 00:02:56 pyh Exp $
Reading input parameters from ->hmg.input<-
*** Warning: sleng=0.000000 invalid, will be re-defined automatically
-----------------------------------------------------------------
using command line grid ->ductful.ux<-
node 0, finished load_ugb: total 3360 bc faces, nitems=7
node 0, 3 bc patches, 3360 total bc faces
node 0, patch 1: 1600 faces
node 0, patch 2: 1600 faces
node 0, patch 3: 160 faces
-----------------------------------------------------------------
Duct without wall domain
-----------------------------------------------------------------
Double precision run for grid ductful.ux
             | (xmin xmax) = (  0.0000E+00  1.0000E+00) *  1.000E+00
Fluid domain | (ymin ymax) = ( -1.0000E+00  1.0000E+00) *  1.000E+00
             | (zmin zmax) = ( -1.0000E+00  1.0000E+00) *  1.000E+00
 Single Phase flow calculation:  dtime= 1.000E-02  sleng= 1.0000E+00
   U= 1.0000E+00  Re= 1.0000E+01  Ha= 0.0000E+00  Cw= 0.0000E+00
-----------------------------------------------------------------
New Start: nmax=    2000
 Initial uniform velocity: uvel=  1.000          dpdx= -7.11354E-01
Volume= 4.00000E+00    Inlet_Area= 4.00000E+00    Inflow Mass-rate=  4.00000E+00
-----------------------------------------------------------------
getdu res =          1     2   1.09790E+00   1.54595E+02
Tstep=        1  1.00000E-02  4.59133E-16  3.80199E+00  1.00694E+00 -3.28012E-01
```

Figure 92: Screen Output on starting the case with HIMAG

```
getdu res =       2000     2  -1.20505E+01   1.51809E-11
Tstep=     2000  1.00000E-02 -1.96316E-15  4.01195E+00  2.09102E+00  3.09947E-03
------ Grid Maximums (CPU, Value) ------
Memory               0:  7.168372
Cells                0:  1600
Interior Faces       0:  6480
PEC Faces            0:  0
Farfield Faces       0:  0
Communication Faces  0:  0
Communication Nbrs   0:  0

------ Timing Maximums (CPU, Value) ------
Preprocessing  0:  0.000000
Solve          0:  0.000000
Communication  0:  0.000000
Wait           0:  0.000000
S+C+W          0:  0.000000
RCS            0:  0.000000
Total          0:  4.003906
```

Figure 91: Screen Output at the End of the simulation

Next step is to post-process the output files (tecfiles). The ***tec.cpu.step.dat*** files will be post-processed for the last step, step 2,000.

```
mhd@machine$ mhd2tec ductful.ux -B 2000
```

To view the file in tecplot, the command used is:

```
mhd@machine$ tec360 ductful.tec.2000.dat
```

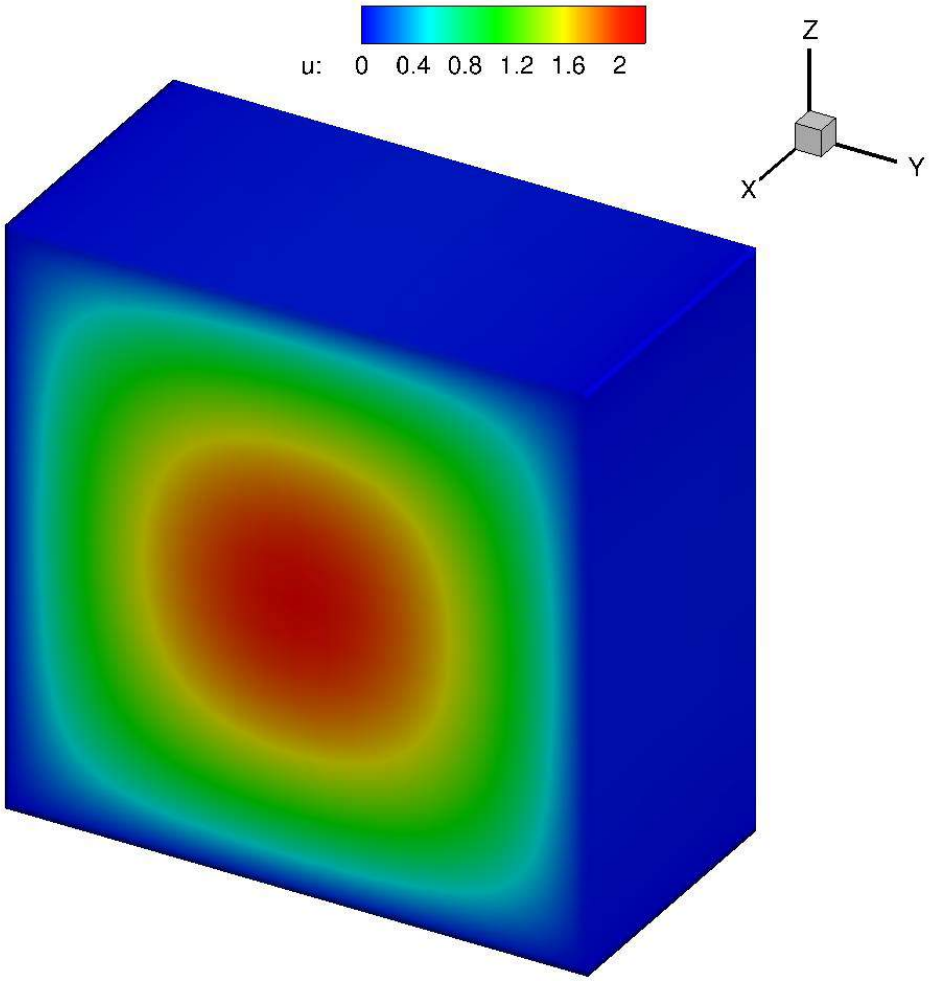


Figure 93: Duct case result in Tecplot

7.2 Tutorial 2: Flow with MHD

This tutorial will introduce you to running a case with flow along with MHD. We will go step by step through the process of setting up a case, running HIMAG and post-processing results. This case is one of the benchmark cases present in the bench folder of HIMAG by the name *demo1_hunt3d*.

Firstly, we will start with the grid. Creation of *gridname.xyz* file is the first step, which for this case is called ***hunt3d_half.xyz***.

```
nopt,   nout
  1        0
x-axis: Number of segments (nxsgm),   isymx,  xmin,   xmax
                                      2      0     0.0     25.0
#1:  nxcel,     segment-Xmax
           6     4.0
      Stretch_option, stretching_factor
           2
           1.45
#2:  nxcel,     segment-Xmax
           26   25.0
      Stretch_option, stretching_factor
           1
           1.36
y-axis: Number of segments (nysgm),   isymy,  ymin,   ymax
                                      3     0     0.0         1.1
#1:  nycel,     segment-Ymax
           10   0.99
      Stretch_option, stretching_factor
           2
           1.01
#2:  nycel,     segment-Ymax
           4     1.0
      Stretch_option, stretching_factor
           2
           1.70
#3:  nycel,     segment-Ymax
           4     1.1
      Stretch_option, stretching_factor
           1
           1.05
z-axis: Number of segments (nzsgm),     isymz,    zmin,      zmax
                                        3         1         -1.1        1.1
```

```
#1: nzcel,     segment-Zmax
          4    -1.0
    Stretch_option, stretching_factor
          2
          1.70
#2: nzcel,     segment-Zmax
          6    -0.8
    Stretch_option, stretching_factor
          1
          1.20
#3: nzcel,     segment-Zmax
          12   0.8
    Stretch_option, stretching_factor
          3
          1.10
```

Once we have the hunt3d_half.xyz file ready, then we run ***xyz*** to create a grid using the mentioned parameters.

```
mhd@machine$ xyz
Enter filename [.xyz]: hunt3d_half
```

This will create a ***univ.cemgrd*** file and ***checkXYZ.dat*** file. We can open the checkXYZ.dat file to see if the spacing between cells is reasonable. The checkXYZ.dat file for this case looks like this: (the complete file is not shown, only the start and end is shown here)

(a)

```
 grid       xf           dx           xc
    1  0.000000E+00  8.136532E-01  4.068266E-01
    2  8.136532E-01  7.822461E-01  1.204776E+00
    3  1.595899E+00  7.241007E-01  1.957950E+00
    4  2.320000E+00  6.471543E-01  2.643577E+00
    5  2.967154E+00  5.604857E-01  3.247397E+00
```

(b)

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Multi-segments HEXAHEDRAL grid (nopt= 1) with hunt3d_half.xyz
 min: dx= 4.72360E-01   dy= 2.06106E-03   dz= 1.73059E-02
 max: dx= 1.03228E+00   dy= 2.59095E-01   dz= 2.18595E-01
-------------------------------------------------------------
 Domain range: x[    0.0000   25.0000]
               y[    0.0000    1.1000]
               z[   -1.1000    1.1000]
 Total number of nodes ( 33 x 19 x 33) =  20691
 Total number of cells ( 32 x 18 x 32) =  18432
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 94: CheckXYZ.dat file (a) - start of the file and (b) - end of the file

Now that we have univ.cemgrd file and everything looks okay in checkXYZ.dat, we use Book command to create .ux file for the grid which will be used by HIMAG.

```
mhd@machine$ book hunt3d_half.ux
```

This command will give us ***hun3d_half.ux*** file which can be opened in *HIMAG-Prep* to setup boundary conditions, which is our next step.

To open this file in *HIMAG-Prep*, let's type in the command for PREP GUI.

```
mhd@machine$ prep
```

Choose the grid file, i.e. *hun3d_half.ux* in the ***HIMAG_Prep*** window.

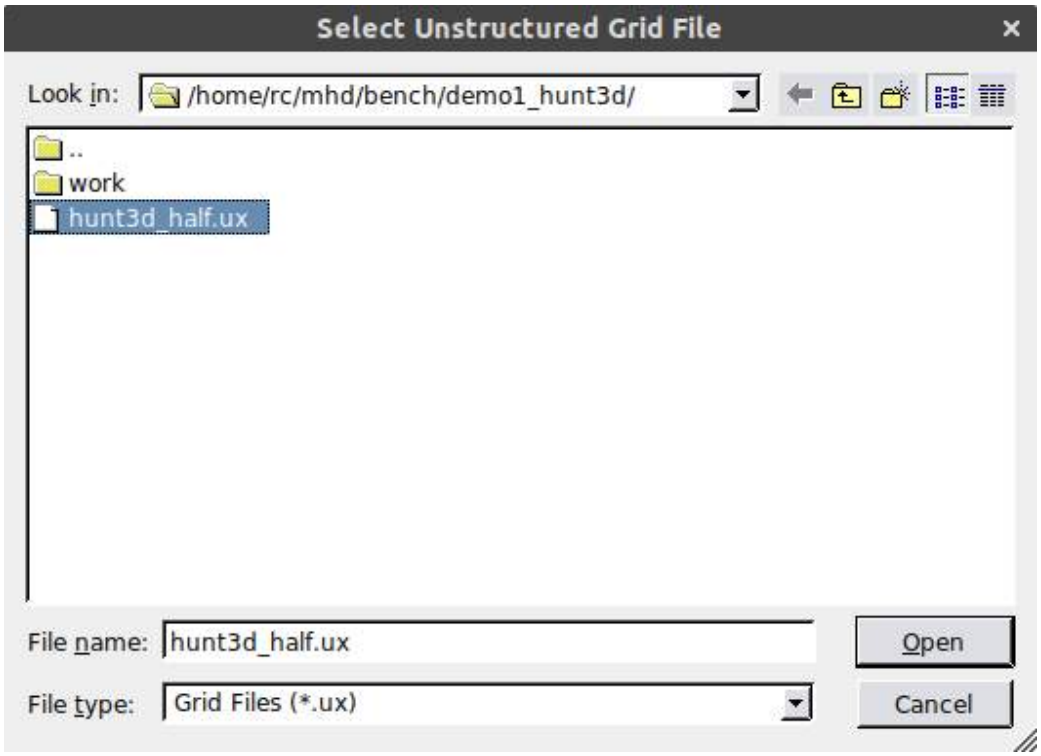


Figure 95: HIMAG-Prep grid selection

The next window will show the work window of ***HIMAG-Prep*** which will be empty with no grid but only grid data shown in the tabs below with the name, number of cells and nodes (Figure 96).

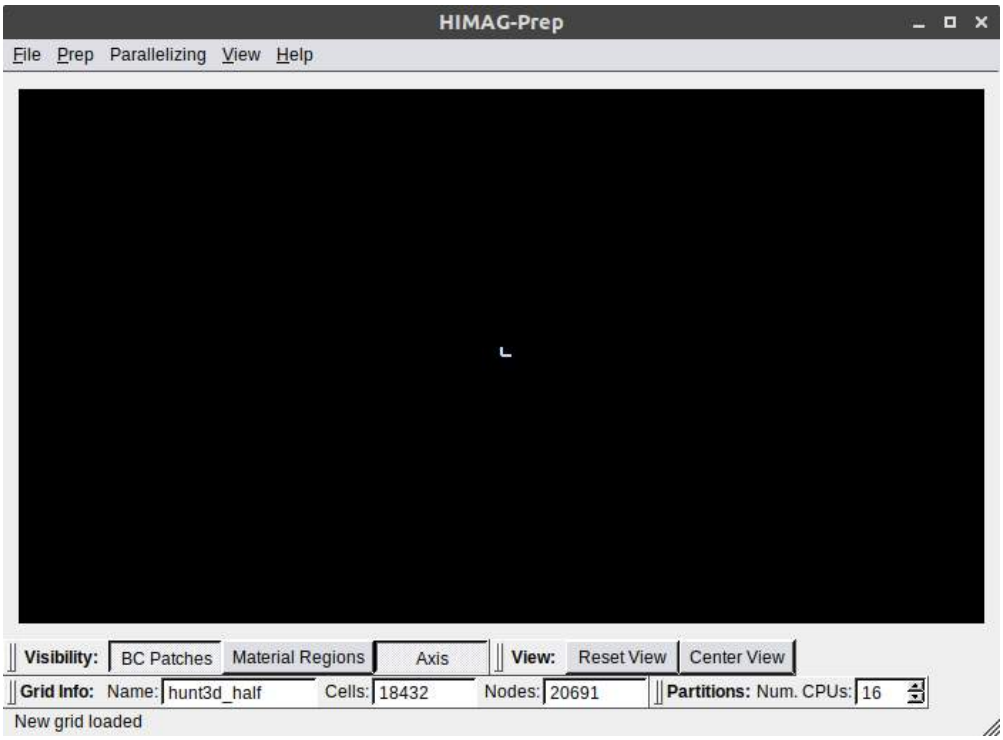


Figure 96: HIMAG-Prep work window

Select *Prep* on the top-left corner of the window and select *Boundary Conditions* under *Prep*. Another window will pop-up for *Boundary Conditions* as discussed in Chapter 4.

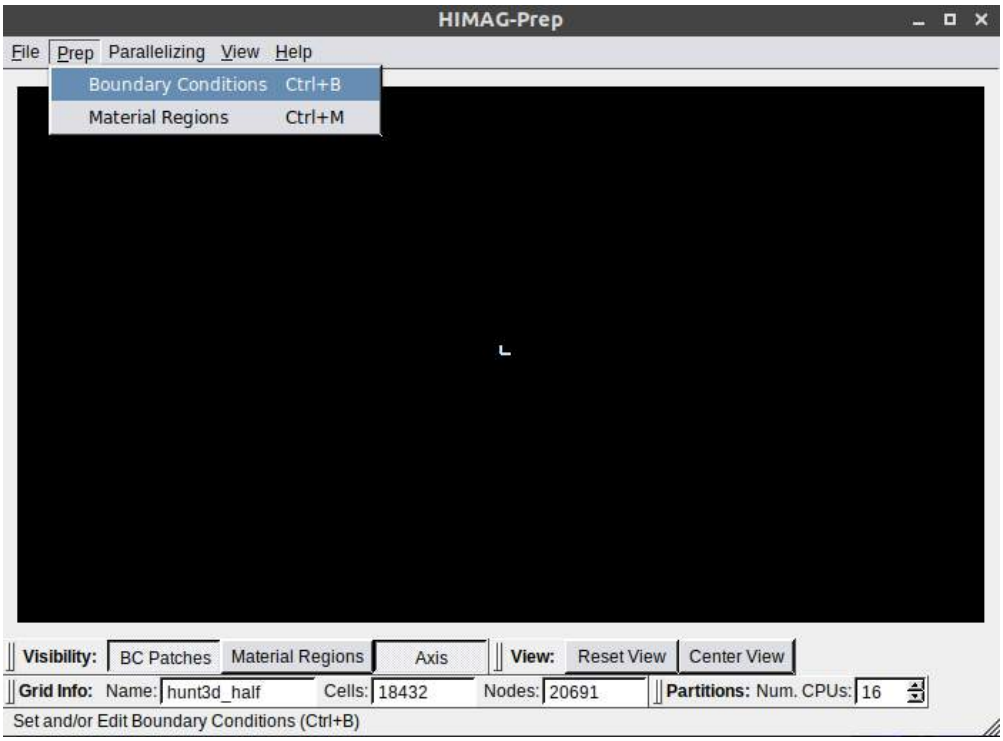


Figure 97: Prep work window showing the location of Boundary conditions tab.

Select *Create* and then *From Cutting Plane* in Figure 97.

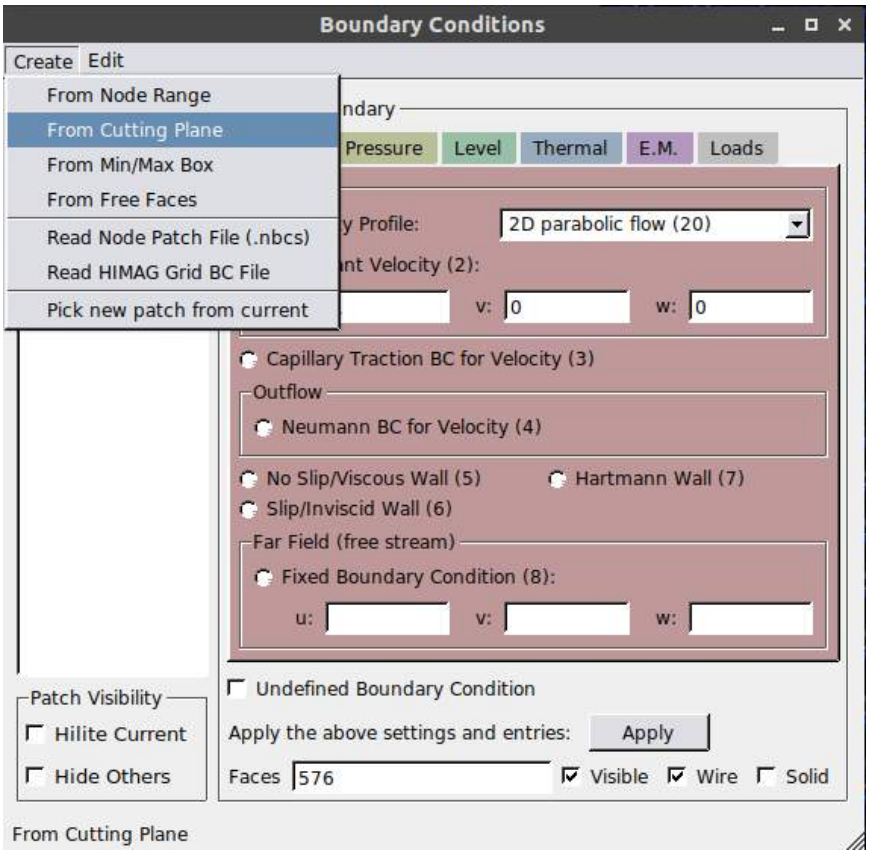


Figure 98: Creating a patch using Cutting Plane in PREP

A pop-up dialogue box (Figure 99) will open asking for the plane where the patch will be created. Select x.

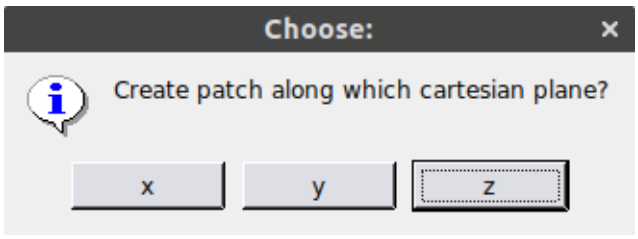


Figure 99: Pop-up dialogue box asking to select the plane

Next, you will be asked to enter tolerance. Let it stay the default value of 0.001.

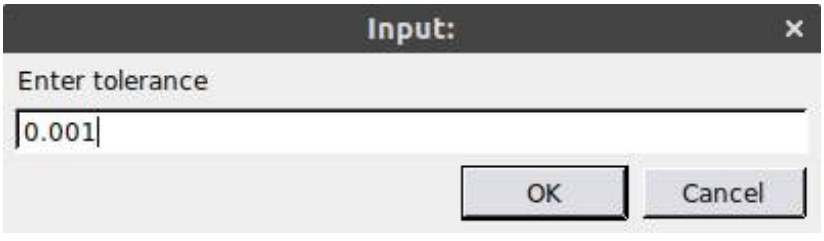


Figure 100: Dialogue box asking for tolerance

The next dialogue box will be to input the x plane value where the cut will be made for the patch to be made. Enter the value, 0.0.

Figure 101: Dialogue box asking for plane value

The next thing you will notice is a patch made in the *HIMAG-Prep* work window and a patch entry in the *Boundary Conditions* window which will go like 2:2:2:2:2:2.

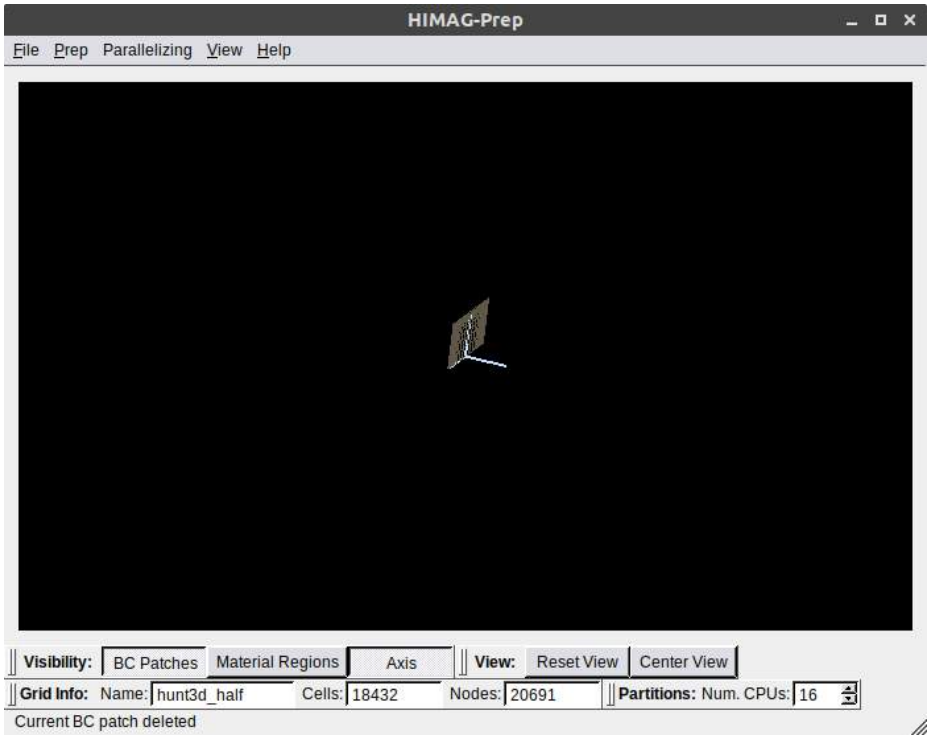


Figure 102: HIMAG-Prep work window after the first patch is created

Similarly, another patch is created at x=25. Select *From Cutting Plane* under *Create* in the *Boundary Conditions* window. Again, choose x-axis for the plane for creating the patch. Let the tolerance remain as it is and enter the x plane value to be 25.0 and click on OK to see another patch form in the *HIMAG-Prep* work window.

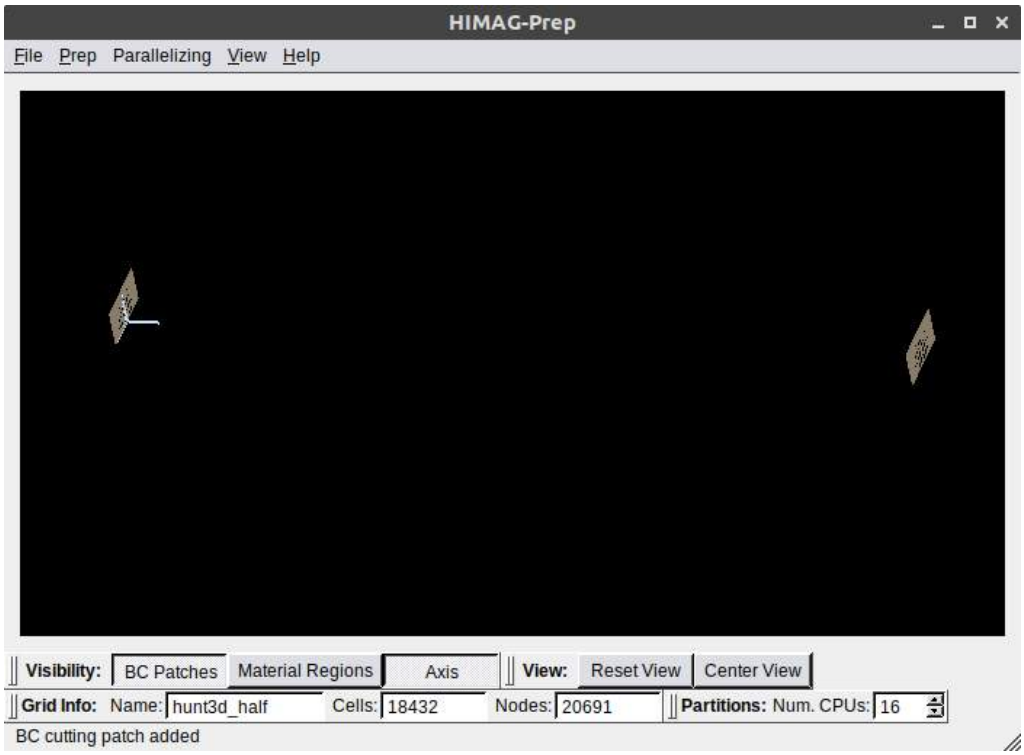


Figure 103: Prep window after second patch is created

The third patch is in the y plane. Therefore, again select From Cutting Plane under Create on the Boundary Conditions window, choose the patch plane as y-axis and let the tolerance remain the same. The value for the axis will be 0.0 and then click OK.

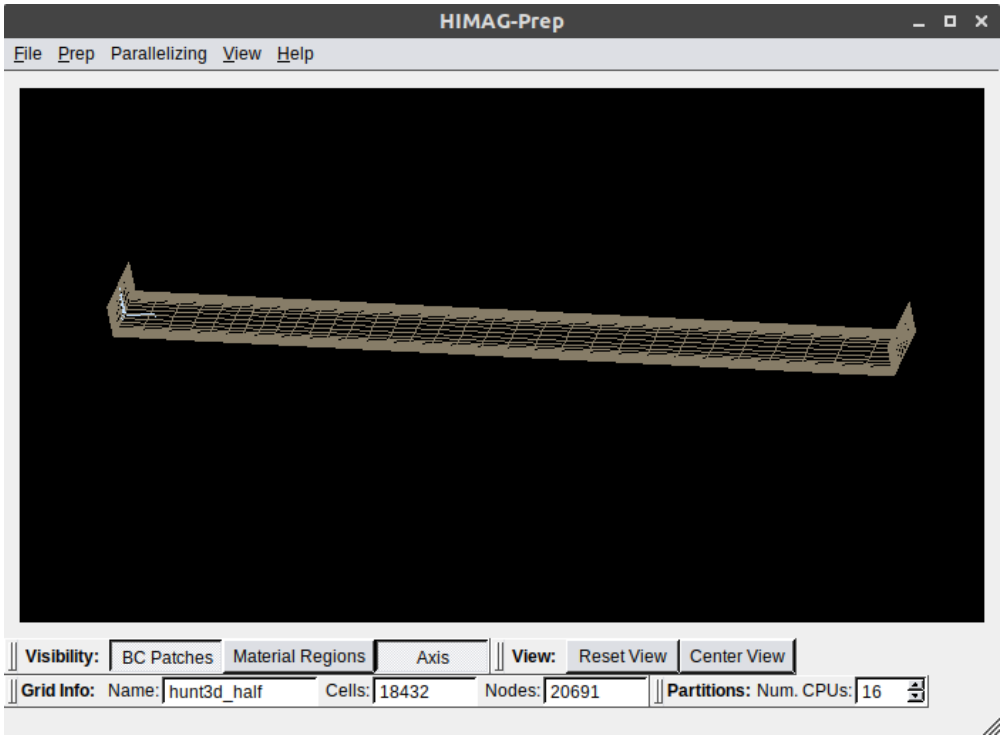


Figure 104: Prep window after the third patch is created

Now, we have 3 patches in the Boundary Conditions window and we can see them all under *Total Boundaries* in the Left most side of the window. We can tick on the *Hilite Current* and *Hide Others* conditions under ***Patch Visibility*** present on the bottom-left corner of the Boundary Conditions window, to inspect each patch.

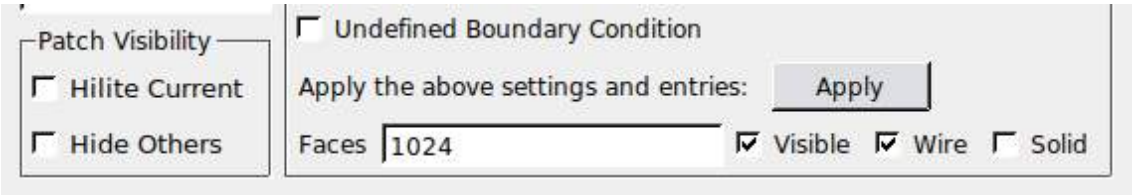


Figure 105: Hilite Current and Hide Others in the Boundary Condition Dialogue Box

The next step is to create the remaining faces which can be created by selecting *From Free Faces* under *Create* tab on *Boundary Conditions* window. This will form another patch with all the remaining faces and will complete the grid. The completed grid will look like Figure 106.

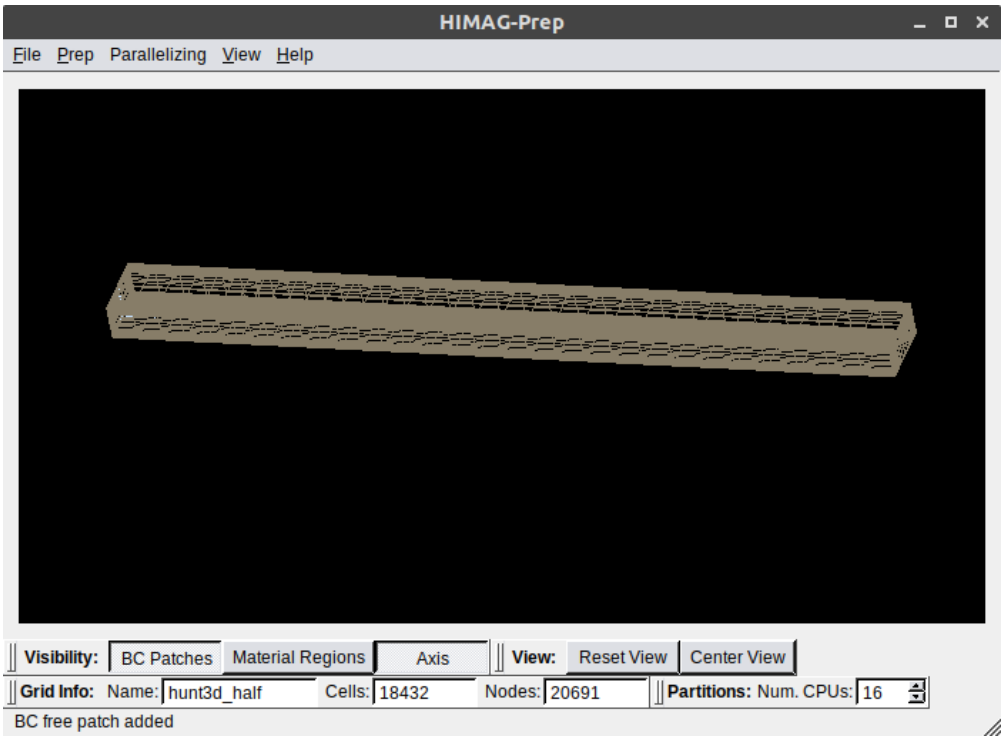


Figure 106: After all the patches are created

The next step is to assign boundary conditions. Let's start with patch 1. Under Velocity tab, select constant velocity (2) and assign u = 1, v=0 and w=0 and click Apply. We will notice that as soon as we hit Apply the loads tab selects Don't Include Force/Moment Calcs (0) on its own.

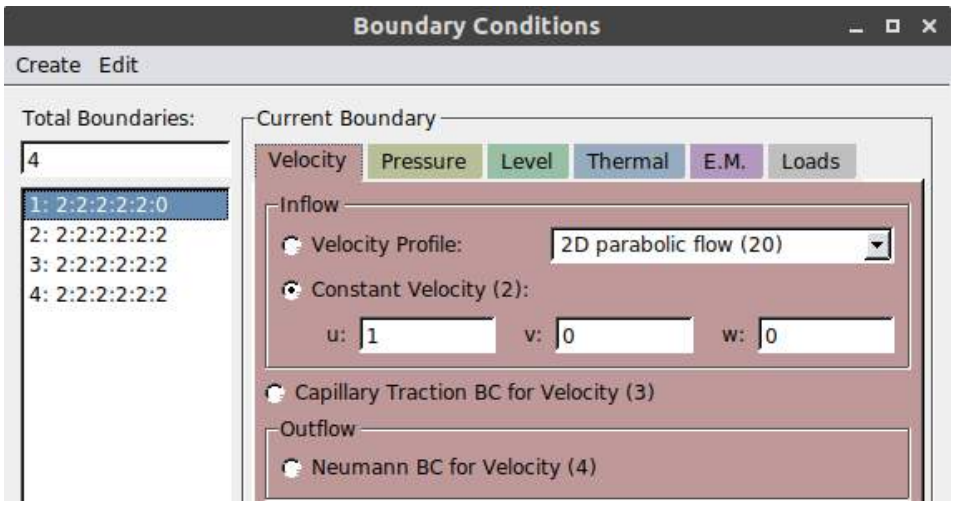


Figure 107: Velocity Boundary Conditions for Patch1

Next select EM tab and select Specified Phi (1) and enter the value 0 next to it and click Apply as shown in Figure 108.

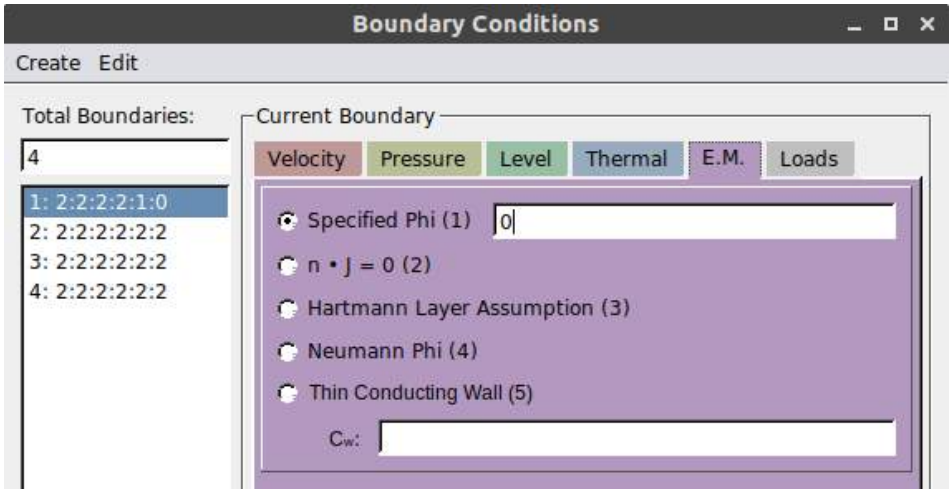


Figure 108: EM Boundary Condition for Patch1

Next is our patch 2 which is our outflow. We will again start with the velocity tab, where we will choose Neumann BC for Velocity (4) under Outflow (Figure 109).

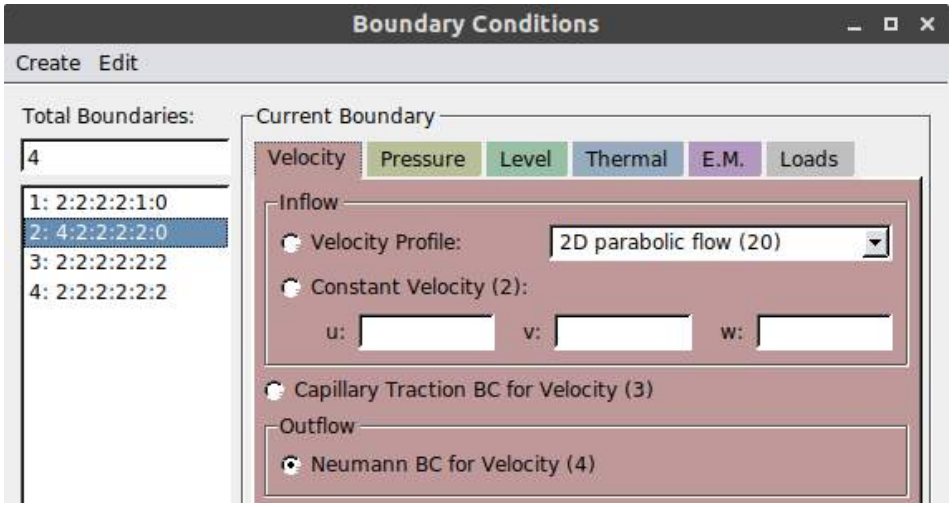


Figure 109: Velocity Boundary Condition for Patch2

If a patch is assigned to be outflow, then Pressure also needs to be adjusted to o Constant Pressure (1) in the Pressure tab and hit Apply (Figure 110).

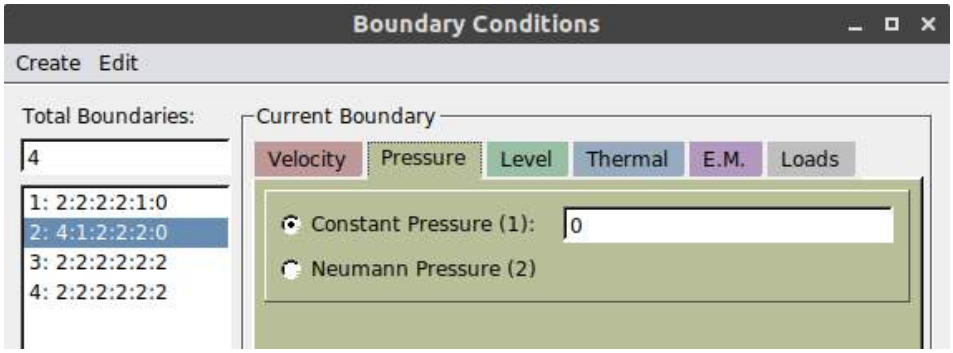


Figure 110: Pressure Boundary Condition for Patch2

Moving on to patch 3, which is the wall in this case. So, in velocity tab, we select Slip/Inviscid Wall (6) and hit Apply (Figure 111).

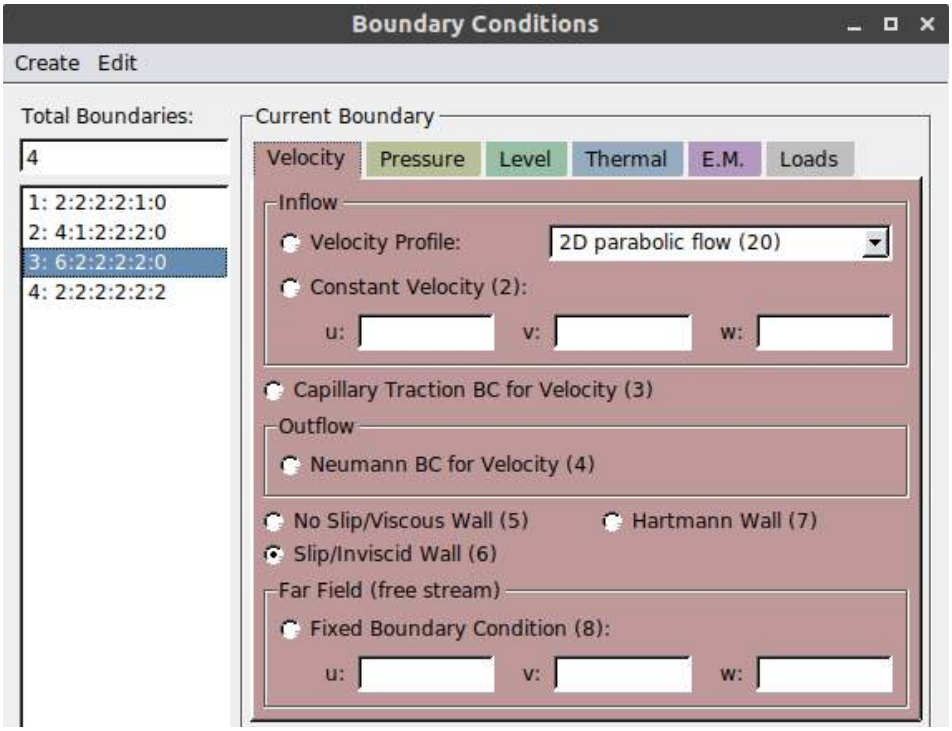


Figure 111: Velocity Boundary Condition for Patch3

Lastly, for patch 4, we select No Slip/Viscous Wall (Figure 112) in the Velocity tab and click on Apply. That's all. We are done with the boundary conditions.

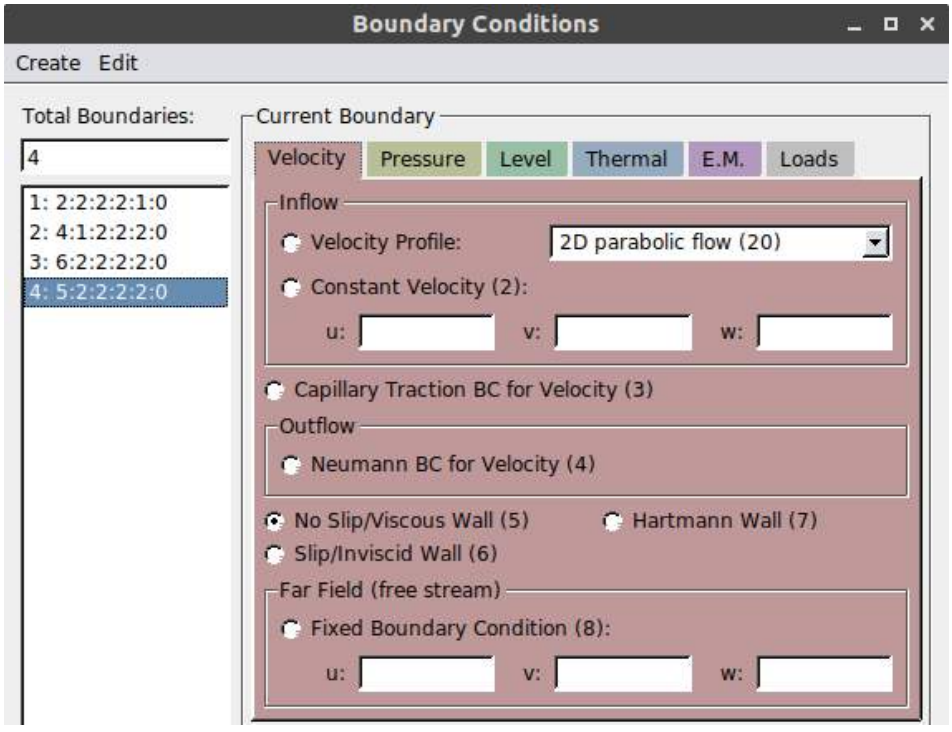


Figure 112: Velocity Boundary Condition for Patch4

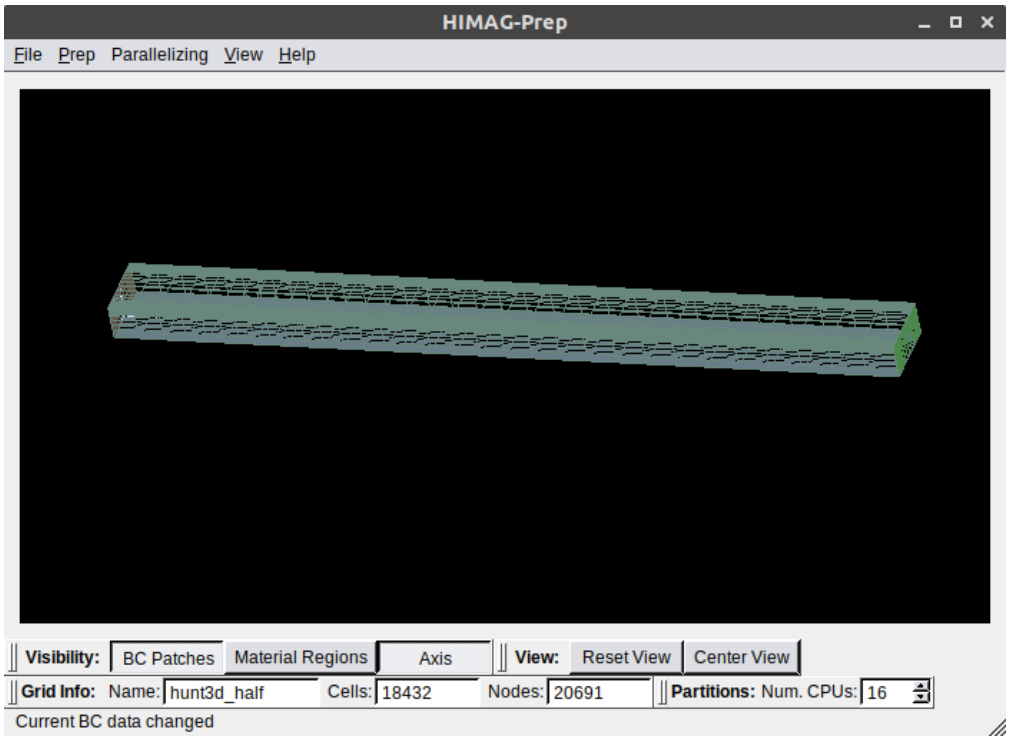


Figure 113: Grid after setting the boundary conditions in PREP

Before closing the window, click on *File* in the *Prep* work window located on the top-left corner of the window. Select *Save .ugb File* in the drop down menu. It will save the .ugb file with the same name as the .ux file, i.e. ***hunt3_half.ugb***.

Now, we have the ugb file as well. The next step is to create an input file, ***hmg.input***. This file will look like:

```
nodes = 1,          #number of CPUs
iread = 0,          #0 for reading no restart file
grid_scale = 1.0,   #grid scale will be multiplied to the dimensions

nmax   = 1000,      #Maximum number of run steps
iskip  = 100,       #Output (Tecfiles) files to be written after these many steps
nwrite = 100,       #Restart files will be written after these many steps
dtime  = 1.0e-3,    # Time-step

visc1  = 1.0e-1,    #Viscosity of the fluid
rho1   = 1.0,       #Density of the fluid
sgmf1  = 1000,      #Electrical Conductivity of fluid
sgmw1 = 1000,       #Electrical Conductivity of solid wall1
sgmw2 = 1.0e-2,     #Electrical Conductivity of solid wall2
alpha  = 1.0,
omega  = 1.00,
```

```
sleng = 1.0,
ubar  = 1.0,          #Initial Velocity
bval  = 1.0,          #Value of Magnetic Field
bx0   = 4.0,
twal  = 0.1,          #Wall Thickness
dpdx  = -82.27,       #Initial Pressure Gradient
iortho = 2,           #Orthogonal Correction Applied
igmax  = 10,          #Orthogonal Correction Iterations
nvel   = 0,

nmomt  = 1,           #Momentum Equation
nppe   = 5,           #Pressure Poisson Equation
nmhd   = 5,           #MHD Poisson Equation
nheat  = 0,           #Heat Equation
ilevels= 0,           #Level Set

ngrad  = 2,
ichan  = 2,
nskp   = 10,          #Skip steps to show Screen Outout
ipmax  = 100,         #Maximum Iterations for Pressure Poisson Equation
immax  = 100,         #Maximum Iterations for MHD Equation
epsmin = -10.0,       #Maximum Convergence Residual
```

Once the input file is ready, we are ready to run HIMAG for this case. Assuming that HIMAG is already built on the machine and the executable file is available for use and added to the path. The command that is used to run HIMAG is:

```
mhd@machine$ himag -i hmg -g hunt3d_half | tee logfile
```

This will start running HIMAG for the case demo1_hunt3d and the screen output will look like *Figure 115* and when the case finishes, the screen output will look like *Figure 114*. This means that the output files have been generated and are ready to be post-processed.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
himag: version $Id: idrive.c,v 1.32 2018/11/27 23:24:43 pyh Exp $
Reading input parameters from ->hmg.input<-
---------------------------------------------------------------------
using command line grid ->hunt3d_half.ux<-
node 0, finished load_ugb: total 4352 bc faces, nitems=8
node 0, 4 bc patches, 4352 total bc faces
node 0, patch 1: 576 faces
node 0, patch 2: 576 faces
node 0, patch 3: 1024 faces
node 0, patch 4: 2176 faces
---------------------------------------------------------------------
Half rectangular channel with conducting walls
---------------------------------------------------------------------
Double precision run for grid hunt3d_half.ux
             | (xmin xmax) = (  0.0000E+00  2.5000E+01) *  1.000E+00
Fluid domain | (ymin ymax) = (  0.0000E+00  1.0000E+00) *  1.000E+00
             | (zmin zmax) = ( -1.0000E+00  1.0000E+00) *  1.000E+00
 Single Phase flow calculation:  dtime= 1.000E-03  sleng= 1.0000E+00
   U= 1.0000E+00  Re= 1.0000E+01  Ha= 1.0000E+02  Cw= 1.0000E-01
---------------------------------------------------------------------
New Start: nmax=    1000
 Initial velocity: 0.0       dpdx= -8.22700E+01
 Inlet BC: invel= 1
Volume= 5.00000E+01    Inlet_Area= 2.00000E+00    Inflow Mass-rate=  2.00000E+00
---------------------------------------------------------------------
mhd-CG step =        1     0  -7.35080E+01  -3.49298E+01
ppe-CG step =        1   200   3.67166E+00
Tstep=       1  1.00000E-03 -1.53184E+00  4.68159E-01  9.42392E-01  8.97900E-02
```

Figure 115: Screen Output on starting the case with HIMAG

```
mhd-CG step =     1000     0  -1.00336E+01  -2.86961E+00
ppe-CG step =     1000     0  -1.01173E+01
Tstep=     1000  1.00000E-03 -4.64169E-02  1.95358E+00  4.07385E+00 -2.49223E-03
------ Grid Maximums (CPU, Value) ------
Memory               0:  66.837900
Cells                0:  18432
Interior Faces       0:  57472
PEC Faces            0:  0
Farfield Faces       0:  0
Communication Faces  0:  0
Communication Nbrs   0:  0

------ Timing Maximums (CPU, Value) ------
Preprocessing  0:  0.000000
Solve          0:  0.000000
Communication  0:  0.000000
Wait           0:  0.000000
S+C+W          0:  0.000000
RCS            0:  0.000000
Total          0:  1024.000000
```

Figure 114: Screen Output at the End of the simulation

Next step is to post-process the output files (tecfiles). The ***tec.cpu.step.dat*** files will be post-processed for the last step, step 1,000.

```
mhd@machine$ mhd2tec hunt3d_half.ux -B 1000
```

This step will give us a file in format, gridname.tec.step.dat. For this case, it will be ***hunt3d_half.tec.1000.dat***. Next step is to convert .dat to .plt using Preplot, a tecplot utility, since we will be viewing the result in tecplot.

```
mhd@machine$ preplot hunt3d_half.tec.1000.dat
```

Running this command will give us .plt file in the format *gridname.tec.step.plt*. In this case it will be ***hunt3d_half.tec.1000.plt***.

Finally, we will view the result file in tecplot. To open the file in tecplot, the command is given as:

```
mhd@machine$ tec360 hunt3d_half.tec.1000.plt
```

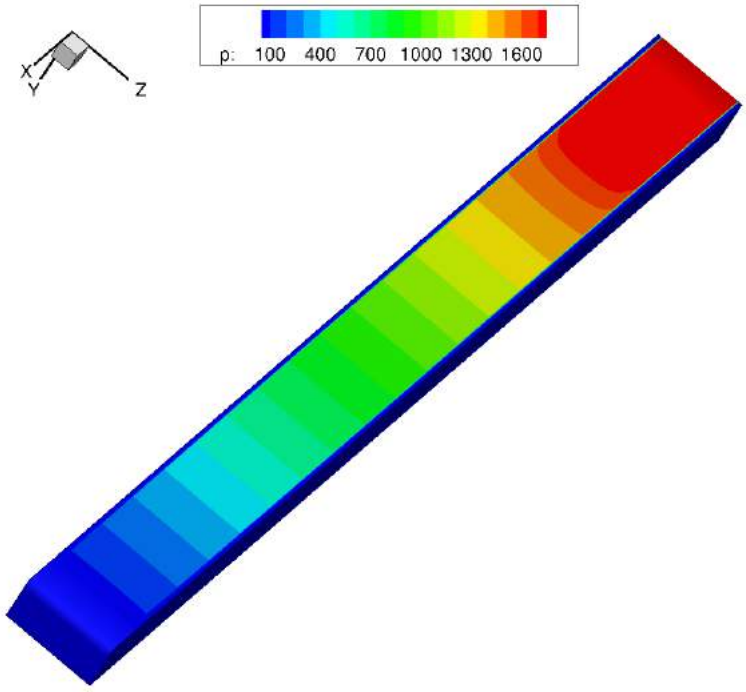


Figure 116: Hunt3D case result in Tecplot

7.3 Tutorial 3: Flow with Heat

In this tutorial, we will go step by step through the process of setting up a case of flow driven by heat for Ha = 100. This case is one of the benchmark cases of natural convection present in the bench folder of HIMAG by the name *nat_conv*.

Firstly, we will start with the grid. Creation of *gridname.xyz* file is the first step, which for this case is called ***natc_ha100.xyz***.

```
nopt,   nout
  1        0
x-axis: Number of segments (nxsgm),   isymx,  xmin,   xmax
                                       2      1       -20.0     20.0
#1:  nxcel,     segment-Xmax
        12       -16.0
      Stretch_option, stretching_factor
        1
        1.2
#2:  nxcel,     segment-Xmax
        12       16.0
      Stretch_option, stretching_factor
        3
        1.035
y-axis: Number of segments (nysgm),   isymy,  ymin,   ymax
                                       2      1        -20.0       20.0
#1:  nycel,     segment-Ymax
        5        -19.8
      Stretch_option, stretching_factor
        0
#2:  nycel,     segment-Ymax
        24       19.8
      Stretch_option, stretching_factor
        3
        1.004
z-axis: Number of segments (nzsgm),    isymz,    zmin,     zmax
                                       1      0         0.0         200.0
#1:  nzcel,     segment-Zmax
        100     200.0
      Stretch_option, stretching_factor
         0
```

Once we have the ***natc_ha100.xyz*** file ready, then we run ***xyz*** to create a grid using the mentioned parameters.

```
mhd@machine$ xyz
Enter filename [.xyz]: natc_ha100
```

This will create a ***univ.cemgrd*** file and ***checkXYZ.dat*** file. Now that we have univ.cemgrd file, we use *book* command to create ***.ux*** file for the grid which will be used by HIMAG.

```
mhd@machine$ book natc_ha100.ux
```

This command will give us ***natc_ha100.ux*** file which can be opened in *HIMAG-Prep* to setup boundary conditions, which is our next step.

To open this file in *HIMAG-Prep*, let's type in the command for PREP GUI.

```
mhd@machine$ prep
```

Choose the grid file, i.e. *natc_ha100.ux* in the ***HIMAG_Prep*** window.

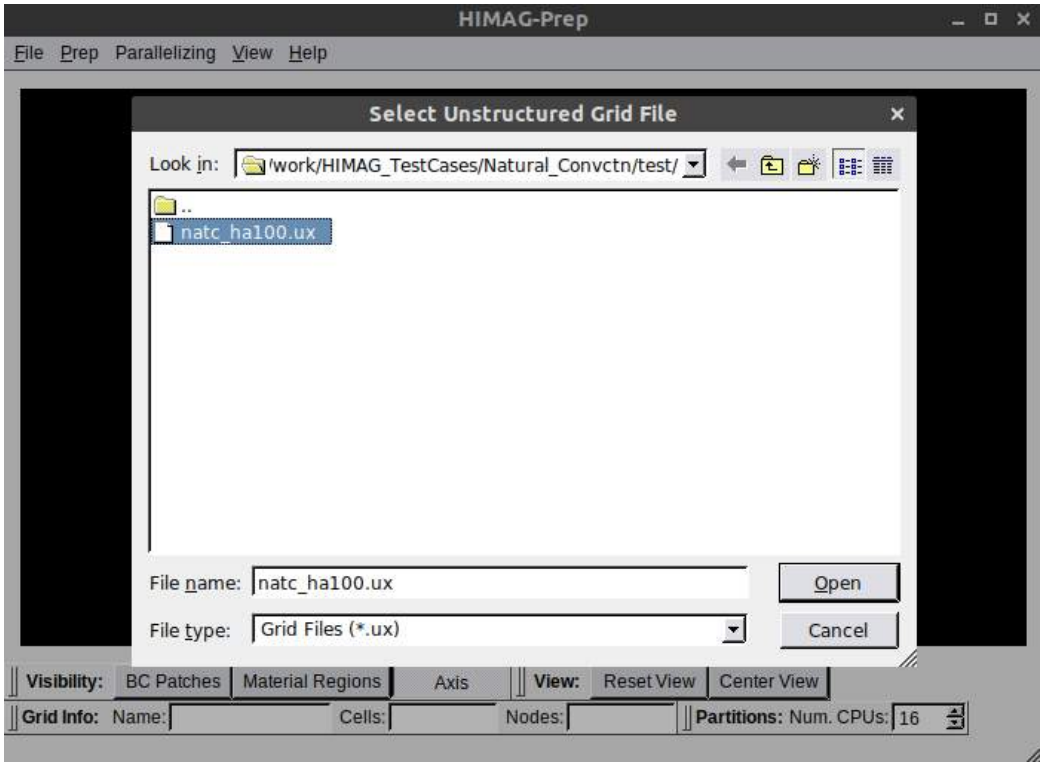


Figure 117: HIMAG-Prep grid selection

The next window will show the work window of ***HIMAG-Prep*** which will be empty with no grid but only grid data shown in the tabs below with the name, number of cells and nodes (Figure 118).

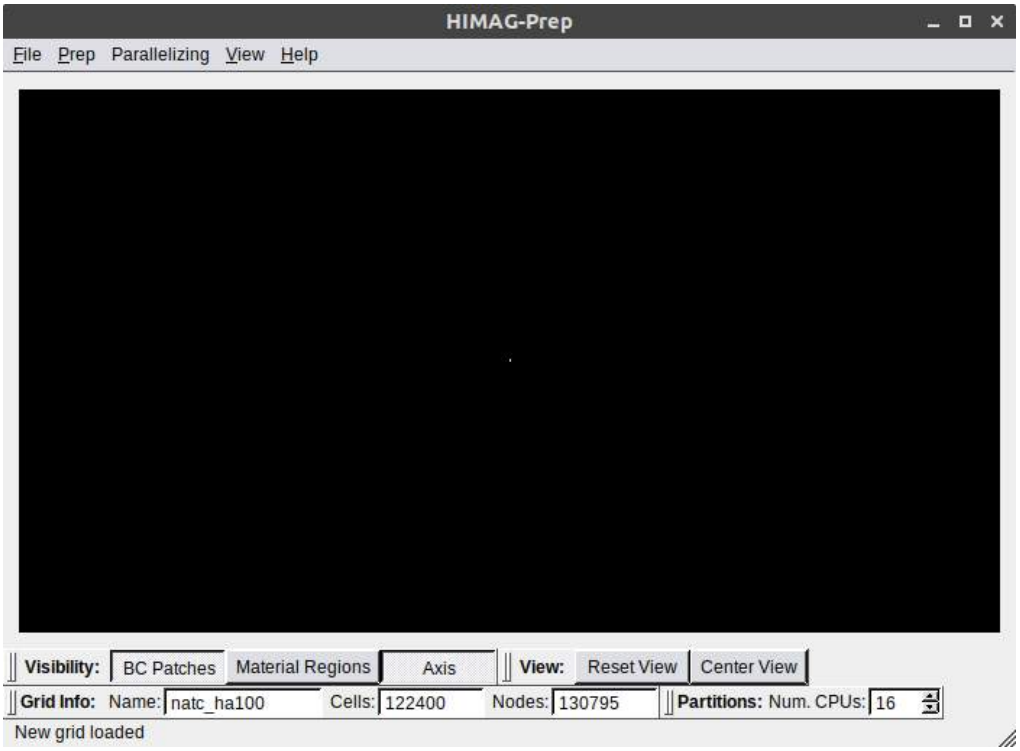


Figure 118: HIMAG-Prep work window

Select *Prep* on the top-left corner of the window and select *Boundary Conditions* under *Prep*. Another window will pop-up for *Boundary Conditions* as discussed in Chapter 4.

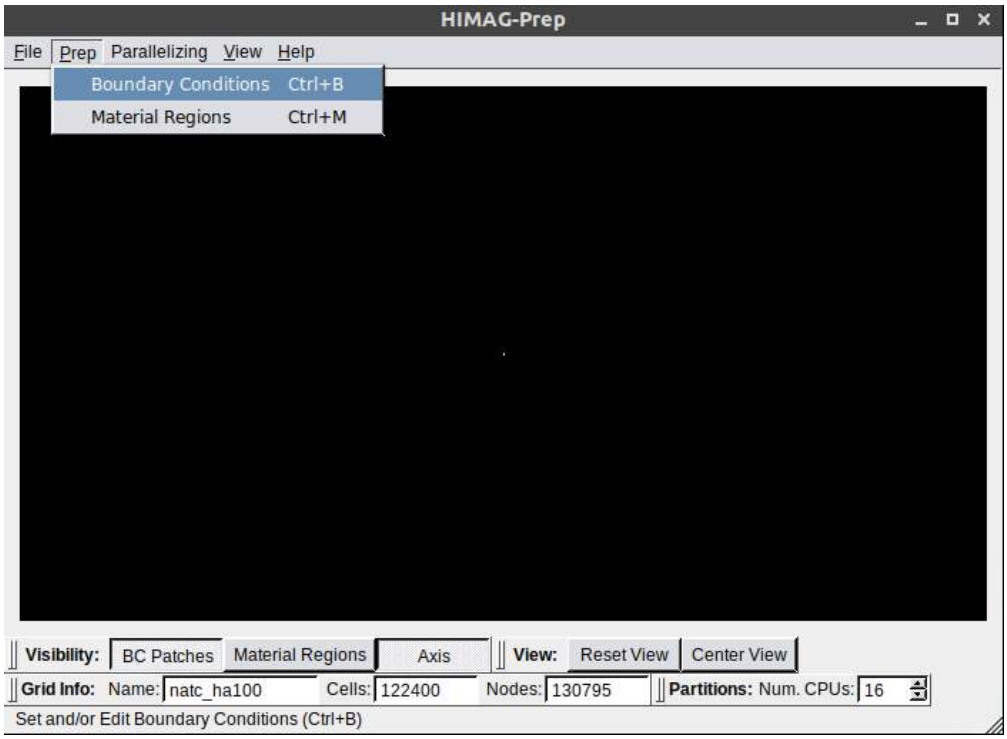


Figure 119: Prep work window showing the location of Boundary conditions tab.

The ***Boundary Conditions*** dialogue box will open (Figure 120). Select ***Create*** in the *Main Menu* and select ***From Cutting Plane***.

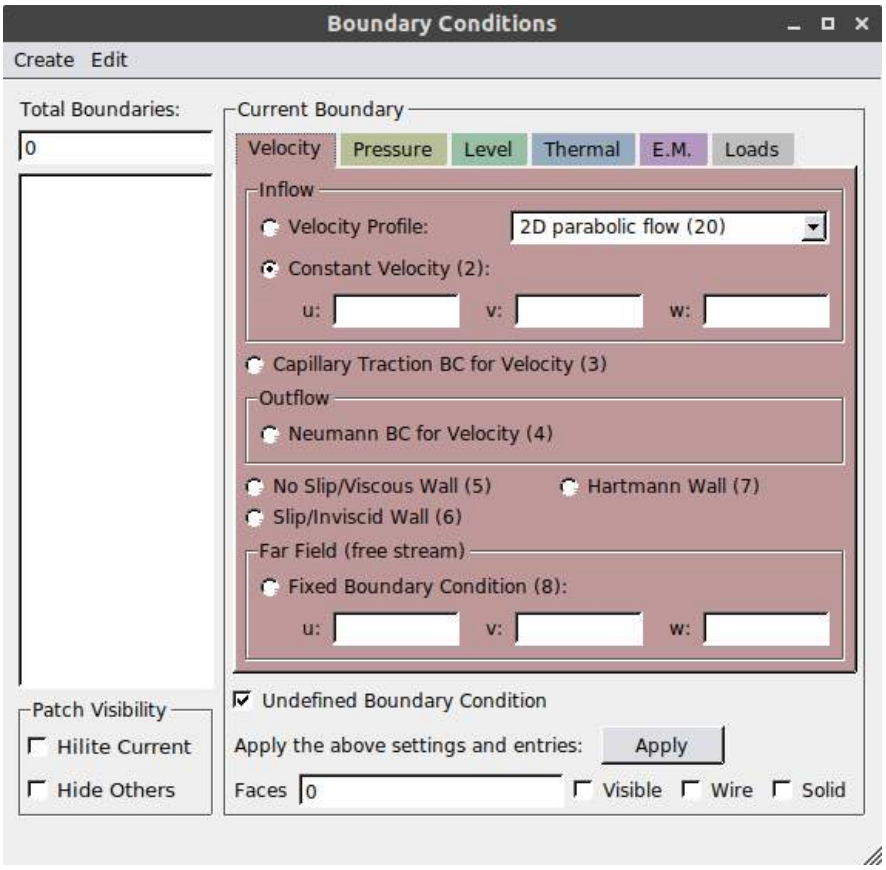


Figure 120: Boundary Conditions dialogue box

A pop-up dialogue box (Figure 121) will open asking for the plane where the patch will be created. Select z.

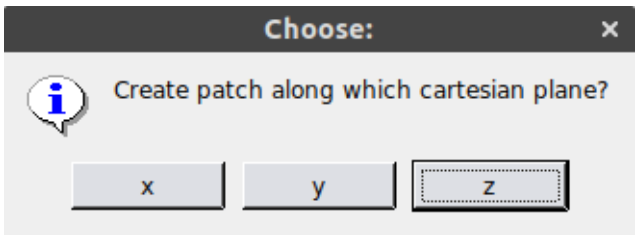


Figure 121: Pop-up dialogue box asking to select the plane

Next, you will be asked to enter tolerance (Figure 122). Let it stay the default value of 0.001.

Figure 122: Dialogue box asking for tolerance

The next dialogue box will be to input the z-plane value where the cut will be made for the patch to be made (Figure 123). Enter the value, 0.0.

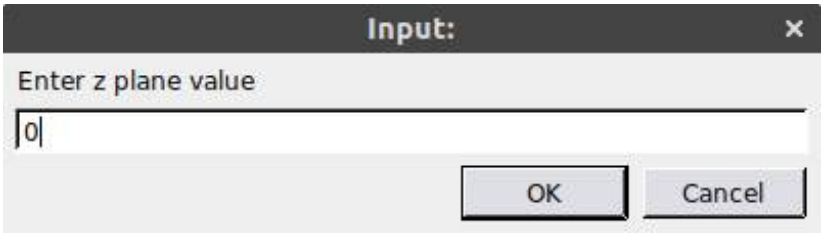


Figure 123: Dialogue box asking for plane value

The next thing you will notice is a patch made in the *HIMAG-Prep* work window (Figure 124).

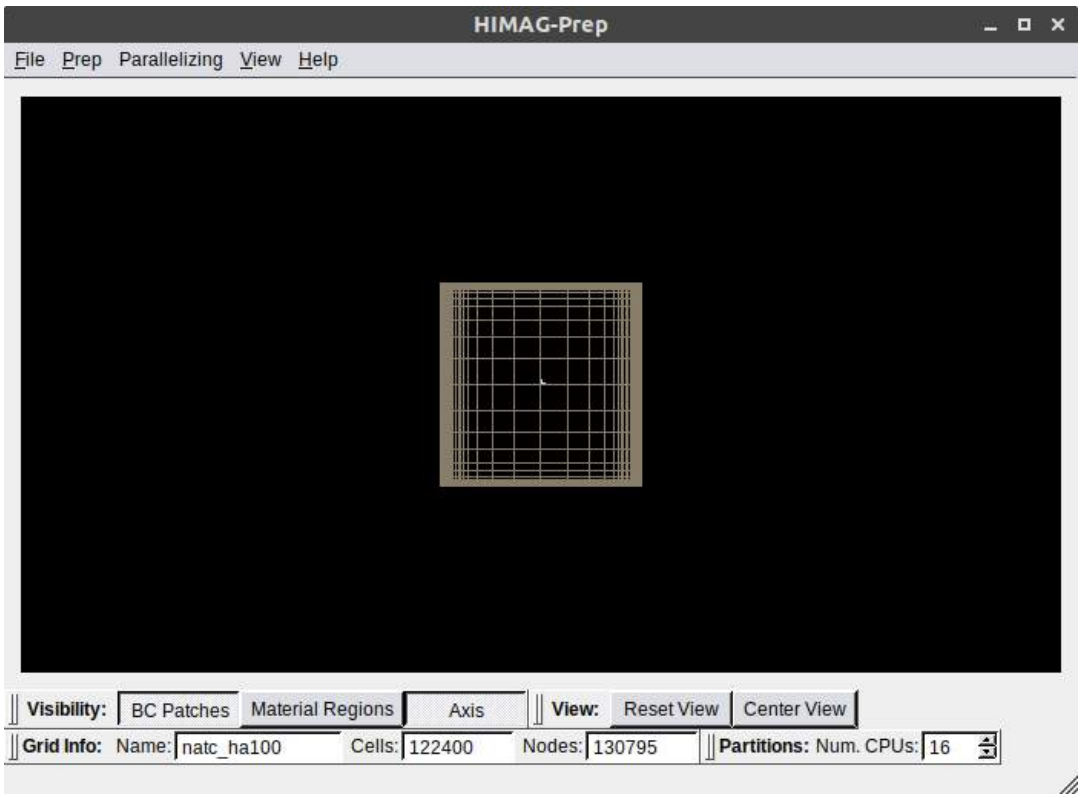


Figure 124: HIMAG-Prep work window after the first patch is created

Similarly, another patch is created at z = 200. Select *From Cutting Plane* under *Create* in the *Boundary Conditions* window. Again, choose z-axis for the plane for creating the patch. Let the tolerance remain as it is and enter the z-plane value to be 200.

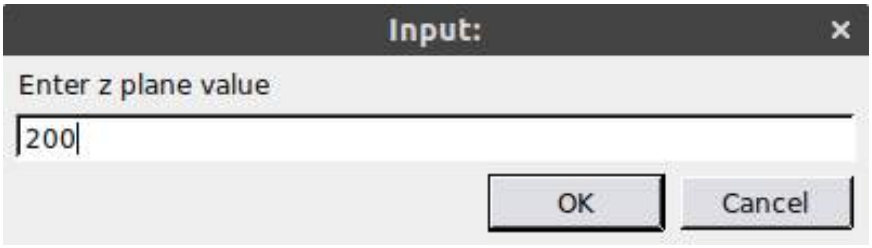


Figure 125: Dialogue box asking for plane value

Click on OK to see another patch form in the *HIMAG-Prep* work window.

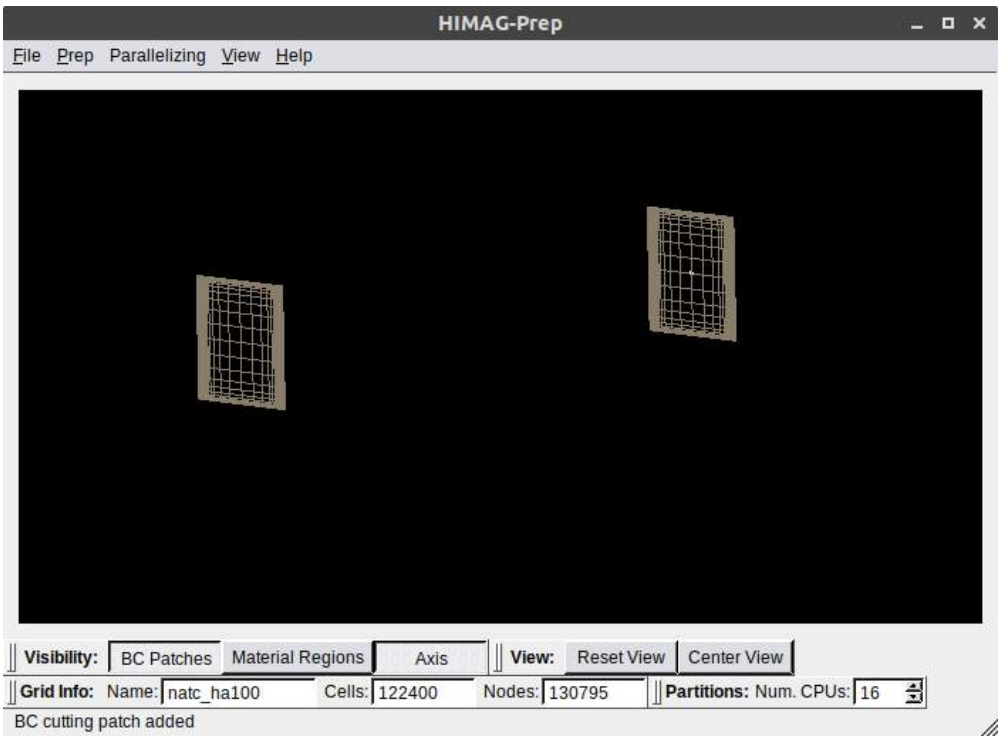


Figure 126: Prep window after second patch is created

The third patch is in the x-plane. Therefore, again select From Cutting Plane under Create on the Boundary Conditions window, choose the patch plane as x-axis.

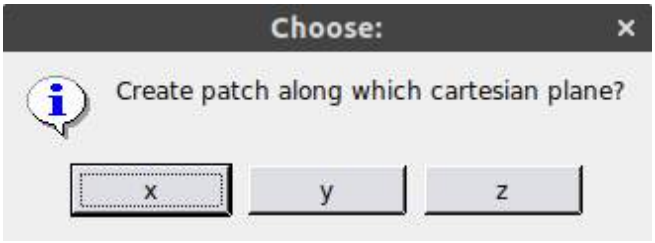


Figure 127: Dialogue box asking to select the plane

The tolerance remains the same. The value for the axis will be 0.0 and then click OK.

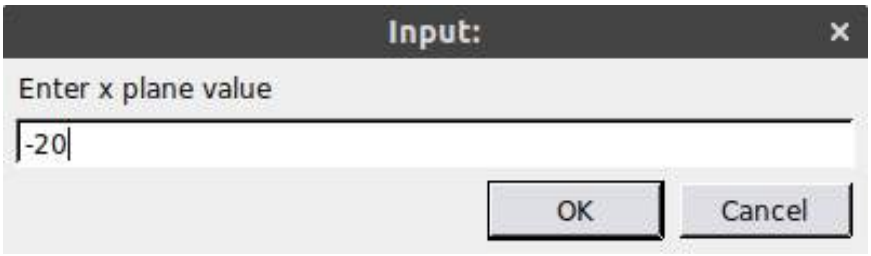


Figure 128: Dialogue box asking for plane value

Another patch will be created in the Prep work window.

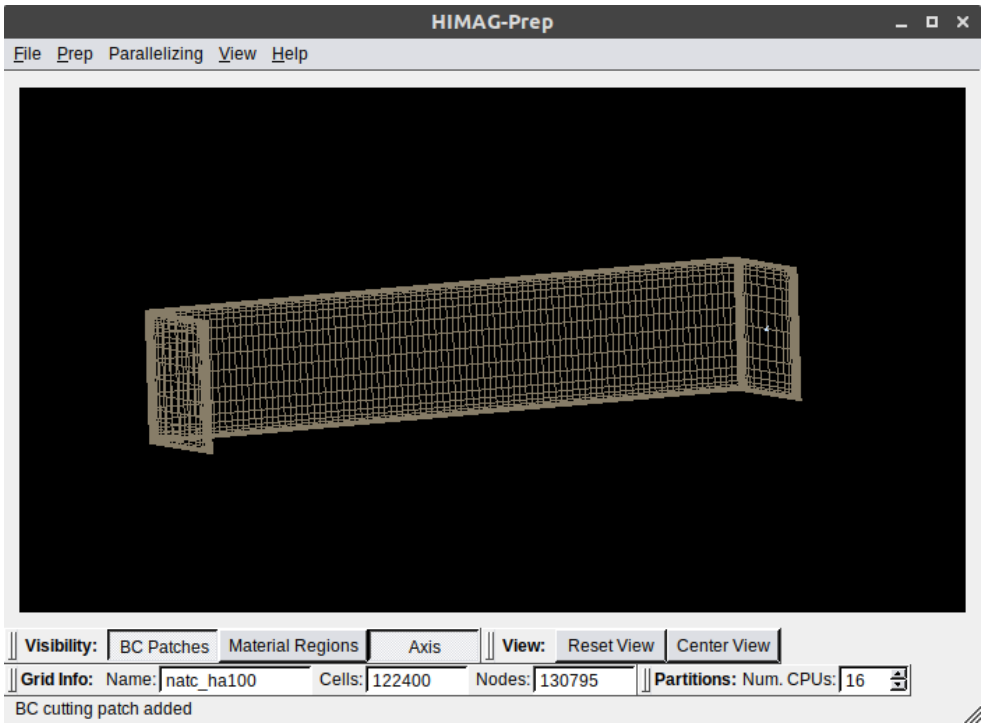


Figure 129: Prep window after the third patch is created

The next patch is also in the x-plane. Therefore, again select From Cutting Plane under Create on the Boundary Conditions window, choose the patch plane as x-axis.

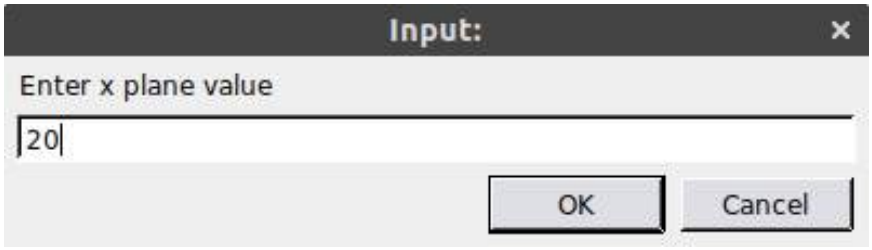


Figure 130: Dialogue box asking for plane value

Another patch will be created in the Prep work window.

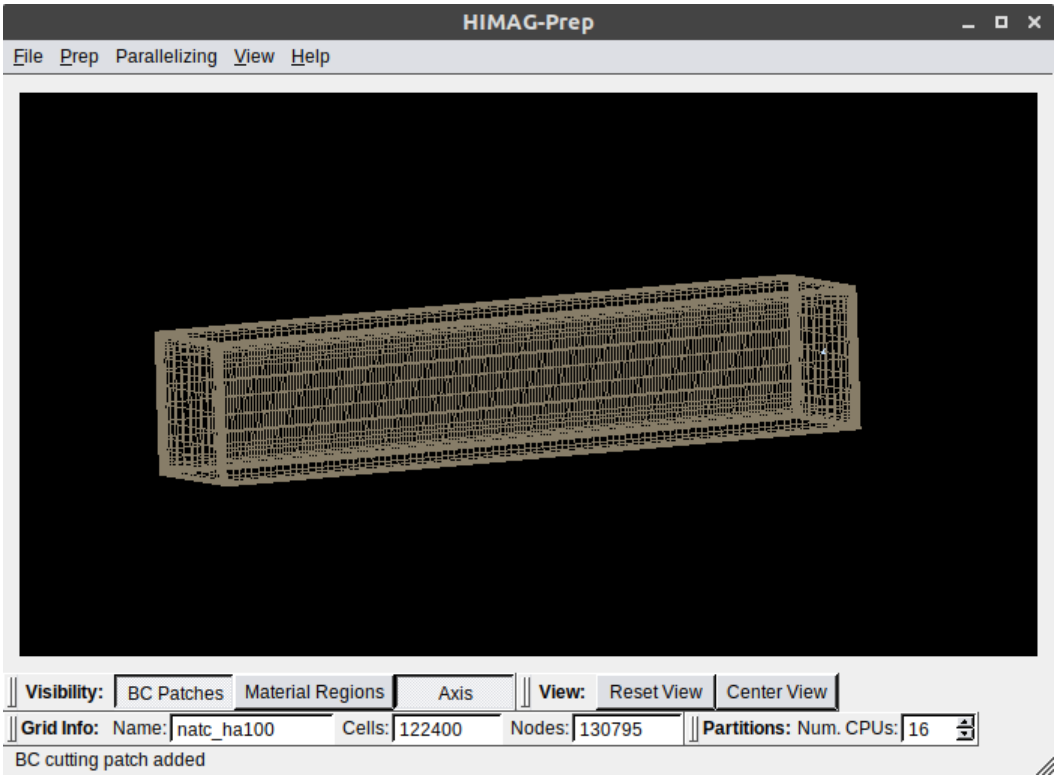


Figure 132: Prep work window after creating four patches

Now, we create the remaining faces by selecting *From Free Faces* under *Create* tab on *Boundary Conditions* window. The completed grid will look like Figure 131.

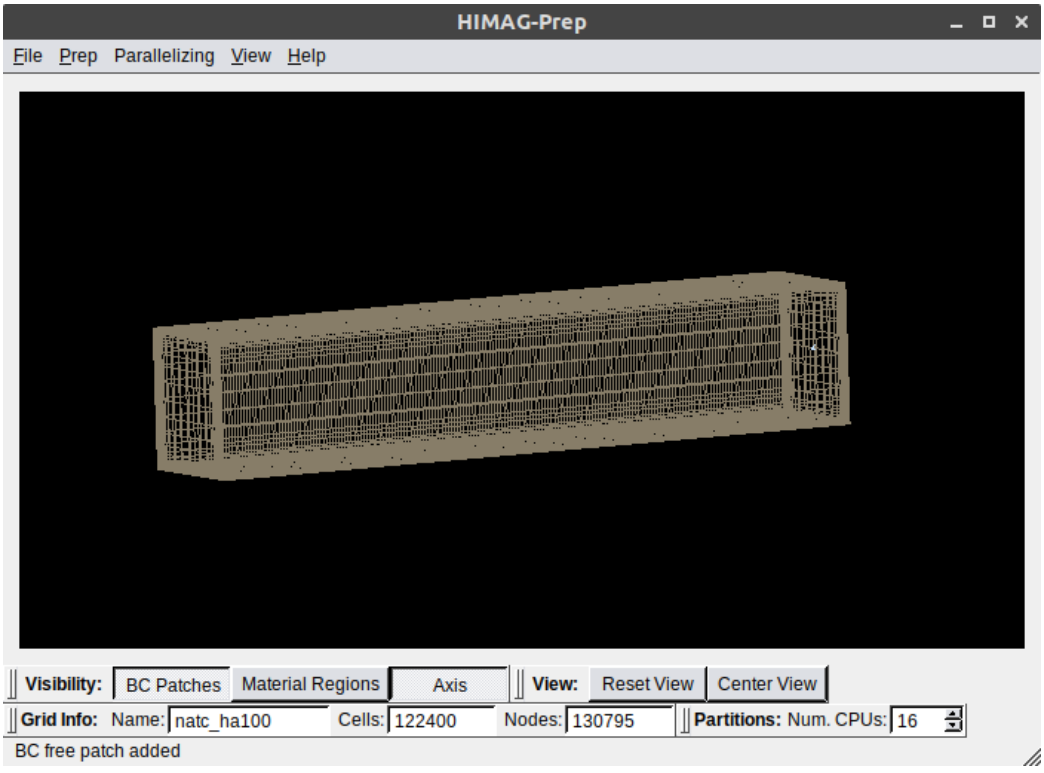


Figure 131: After all the patches are created

The next step is to assign boundary conditions. Let's start with patch 1. Under *Velocity* tab, select *No Slip/Viscous Wall (5)* and click *Apply* (Figure 133).

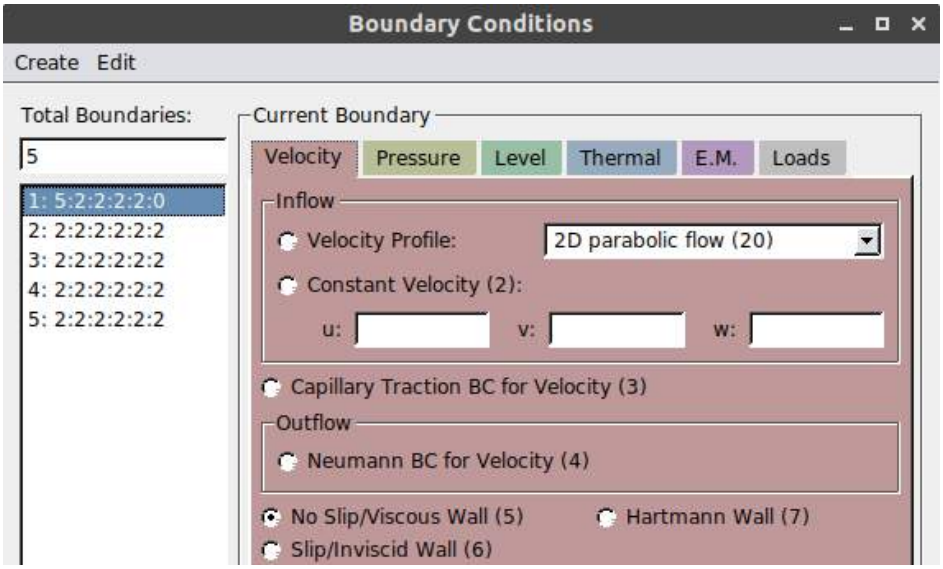


Figure 133: Velocity Boundary Conditions for Patch1

For patch 2 also, we select *No Slip/Viscous Wall (5)* under *Velocity tab* and click *Apply* (Figure 134).

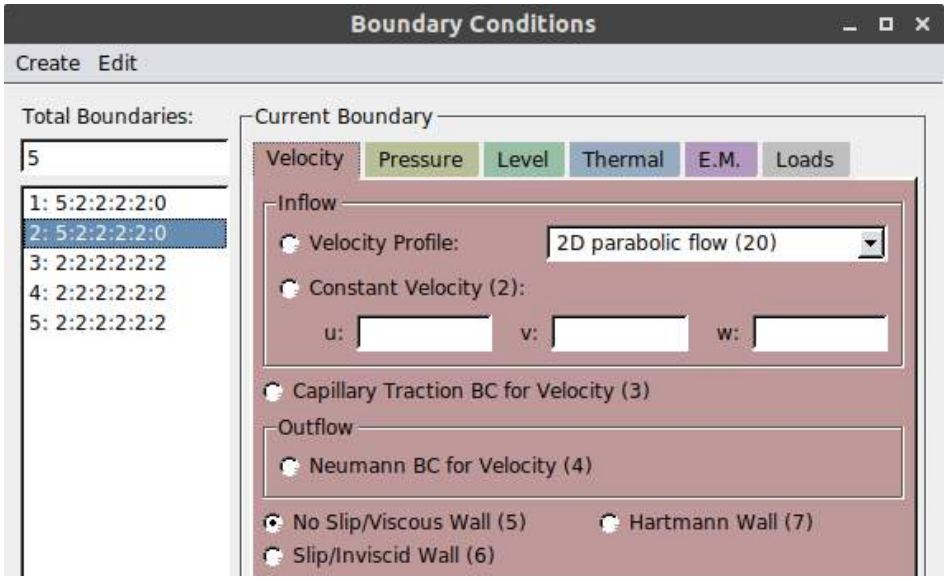


Figure 134: Velocity Boundary Condition for Patch2

Similarly for patch 3, we select *No Slip/Viscous Wall (5)* under *Velocity tab* and click *Apply* (Figure 135).

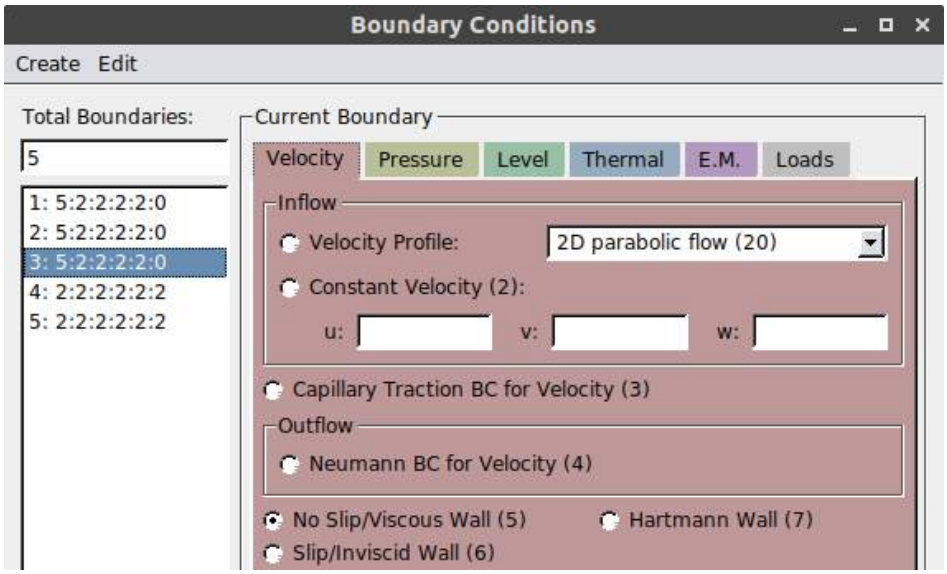


Figure 135: Velocity Boundary Condition for Patch3

Here, we will also assign a temperature value to the wall. So, click on Thermal tab, select Fixed Temperature and input a value of 5 (Figure 136).

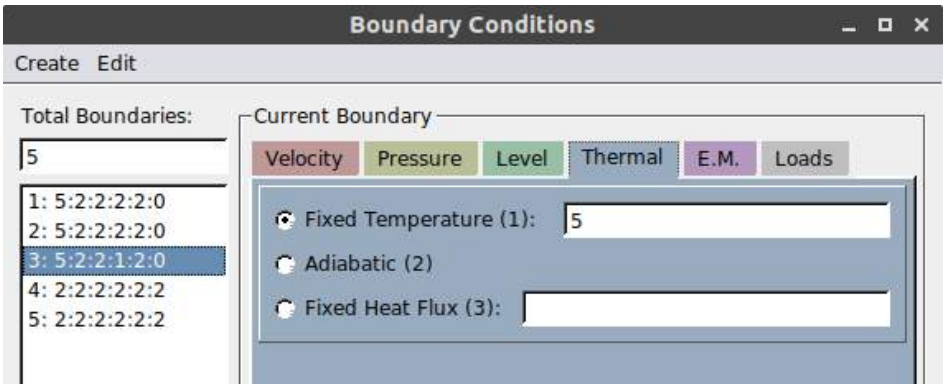


Figure 136: Thermal Boundary Condition for Patch3

The next patch is patch 4, and for Velocity Boundary Condition we select No Slip/Viscous Wall option (Figure 138) and for Thermal Boundary Condition, we select Fixed Temperature (1) and input a value of -5 (Figure 137).

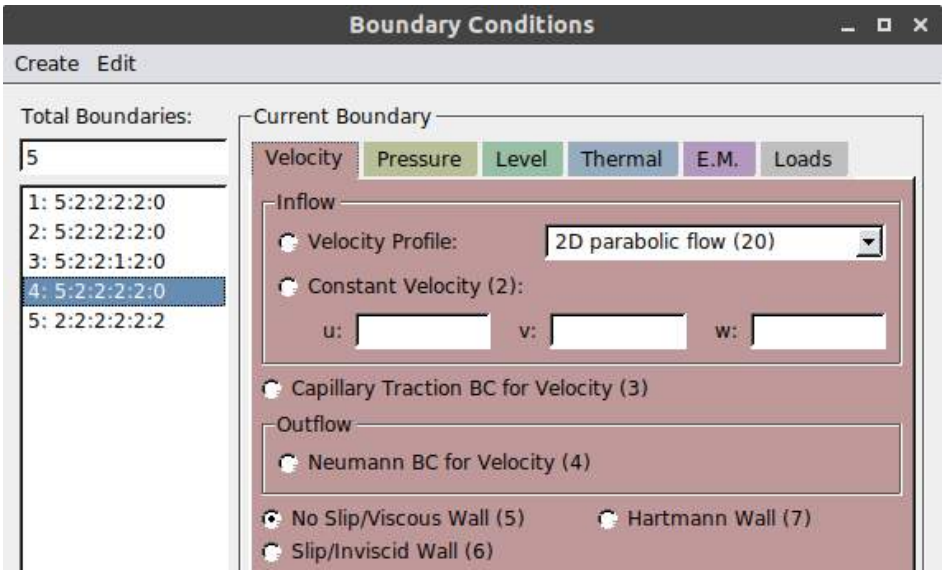


Figure 138: Velocity Boundary Condition for Patch4

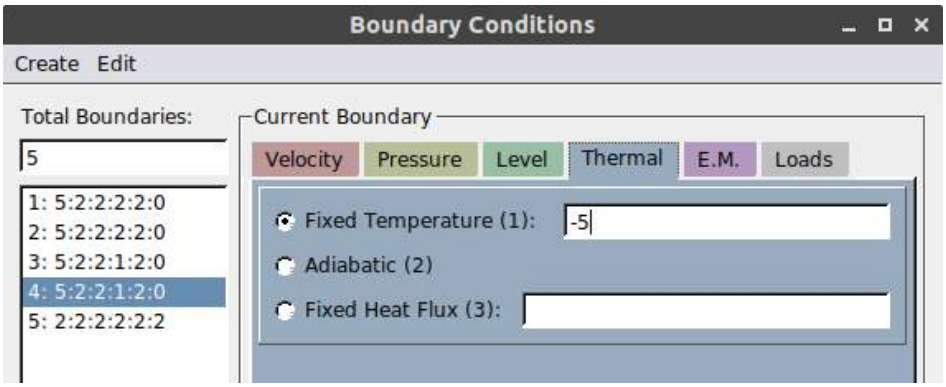


Figure 137: Thermal Boundary Condition for Patch4

Lastly, for patch 5, we select No Slip/Viscous Wall (Figure 139) in the Velocity tab.

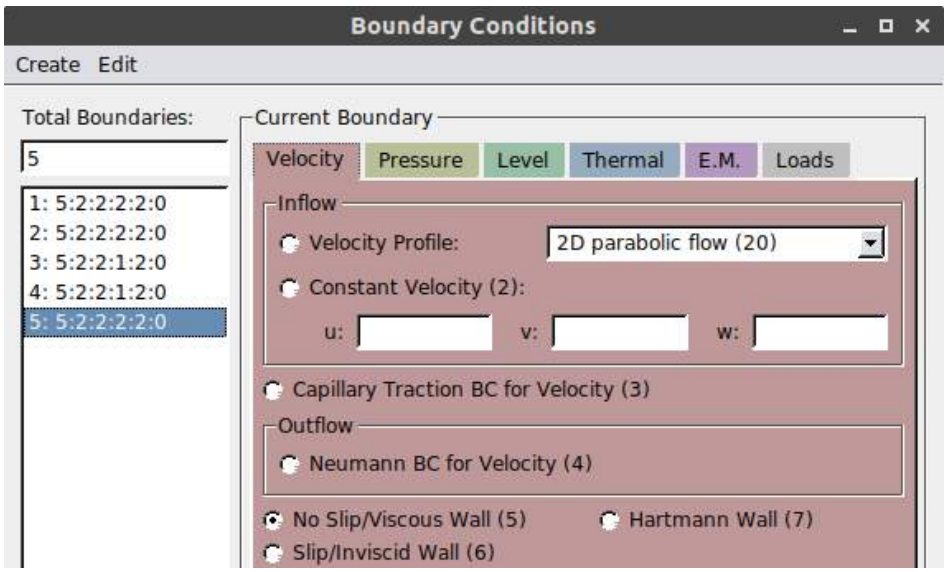


Figure 139: Velocity Boundary Condition for Patch5

Before closing the window, click on *File* in the *Prep* work window located on the top-left corner of the window. Select *Save .ugb File* in the drop down menu (Figure 140). It will save the .ugb file with the same name as the .ux file, i.e. ***natc_ha100.ugb***.

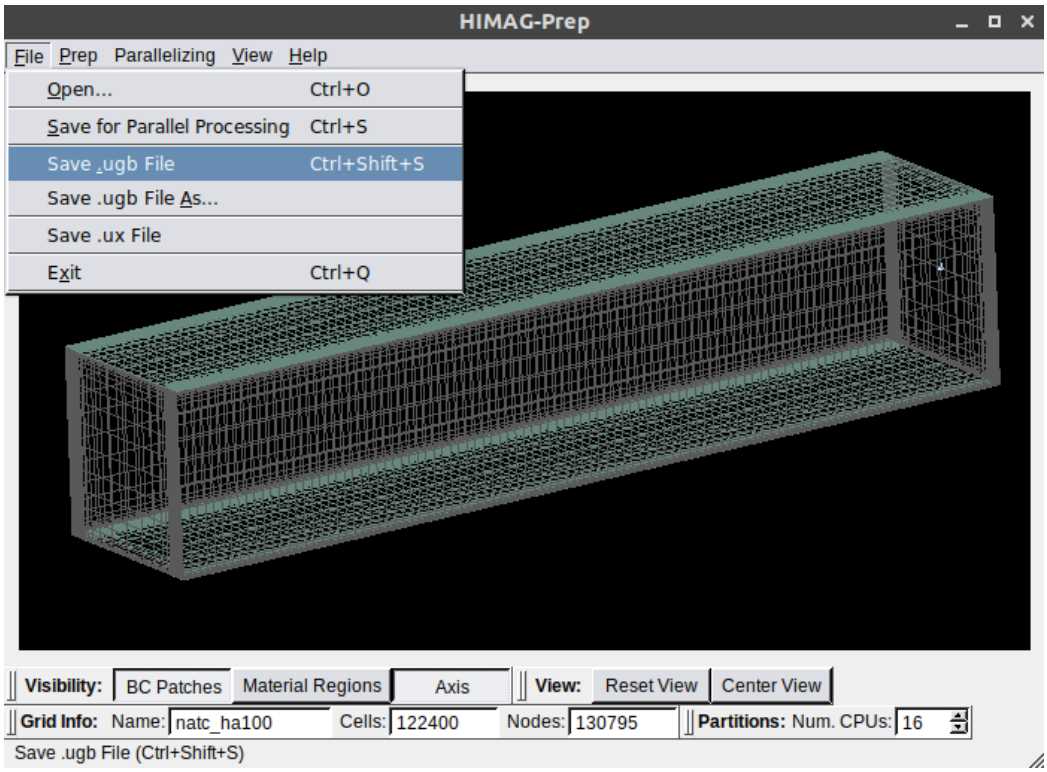


Figure 140: Grid after setting the boundary conditions in PREP

This grid has 122,400 cells which if ran on single cpu will take a long time. So, here we will partition the grid for 8 cpus using ***ux2part***–

```
mhd@machine$ ux2part natc_ha100 8
```

This will give us natc_ha100.8.color file, which will then be used for ***colormhd*** -

```
mhd@machine$ colormhd natc_ha100.ux natc_ha100.8.color
```

This command will provide us with partition files for all 8 cpus in the format: *partition.0.patch*, *partition.0.ugb*, *partition.0.ux*, *partition.1.patch*, *partition.1.ugb*, *partition.1.ux* and so on until *partition.7.patch*, *partition.7.ugb*, *partition.7.ux*.

Now, we have the ugb file as well. The next step is to create an input file, ***hmg.input***.

```
 nodes    = 8,              #number of CPUs
 iread    = 0,              #0 for reading no restart file
 grid_scale = 1.0e-3,       #grid scale will be multiplied to the dimensions
 nmax     = 100000,         #Maximum number of run steps
 iskip    = 1000,           #Output (Tecfiles) files to be written after these many steps
 nwrite   = 1000,           #Restart files will be written after these many steps
 dtime    = 5.0e-4,         # Time-step

#Material Properties
 visc1    = 1.548e-3,       #Viscosity of the fluid
 rho1     = 1.3579e+4,      #Density of the fluid
 sgmf1    = 1.02e+6,        #Electrical Conductivity of fluid
 hk1      = 8.67,
 cp1      = 1.4e+2,
 gras     = 3.0e+7,
 tref     = 293.0,
 deltT    = 10.0,

#Physical Parameters
 sleng    = 40.0,
 ubar     = 0.01,           #Initial Velocity
 bval     = 0.097392475,    #Value of Magnetic Field
 bx0      = 0.0,
 twal     = 0.0,            #Wall Thickness
 dpdx     = -0.0,           #Initial Pressure Gradient
 iuvw     = 3,
 ibf      = 2,
 nbf      = -1,
 ichan    = 0,

#Solver Options
 nmomt    = 1,              #Momentum Equation
 nppe     = 5,              #Pressure Poisson Equation
 nmhd     = 5,              #MHD Poisson Equation
 nheat    = 1,              #Heat Equation
 ilevels  = 0,              #Level Set
 ibuoy    = 1,
 igravt   = 0,

#Numerical Options
 nvel     = 0,
 iortho   = 1,
 ngrad    = 7,
 nskp     = 10,             #Skip steps to show Screen Output
 ipmax    = 25,             #Maximum Iterations for Pressure Poisson Equation
 immax    = 25,             #Maximum Iterations for MHD Equation
 ihmax    = 40,
 nbcg     = 0,
 epsmin   = -16.0,          #Maximum Convergence Residual
```

Once the input file is ready, we are ready to run HIMAG for this case. Assuming that HIMAG is already built on the machine and the executable file is available for use and added to the path. The command that is used to run HIMAG on multiple cpus is:

```
mhd@machine$ himag -i hmg | tee logfile
```

This will start running HIMAG for the natural convection case and once the case ends, we will post-process the output files. The **tec.cpu.step.dat** *files will be post-processed for the last step, step 100,000.*

```
mhd@machine$ mhd2tec natc_ha100.ux -p natc_ha100.8.color -B 100000
```

This step will give us a file in format, gridname.tec.step.dat. For this case, it will be ***natc_ha100.tec.100000.dat***. Next step is to convert .dat to .plt using Preplot, a tecplot utility, since we will be viewing the result in tecplot.

```
mhd@machine$ preplot natc_ha100.tec.100000.dat
```

Running this command will give us .plt file in the format gridname.tec.step.plt. In this case it will be ***natc_ha100.tec.100000.plt***. For a grid this size, it's better to convert the grid from.dat to .plt using preplot as a .dat file will require a lot of machine memory. Finally, we will view the result file in tecplot.

```
mhd@machine$ tec360 natc_ha100.tec.100000.plt
```

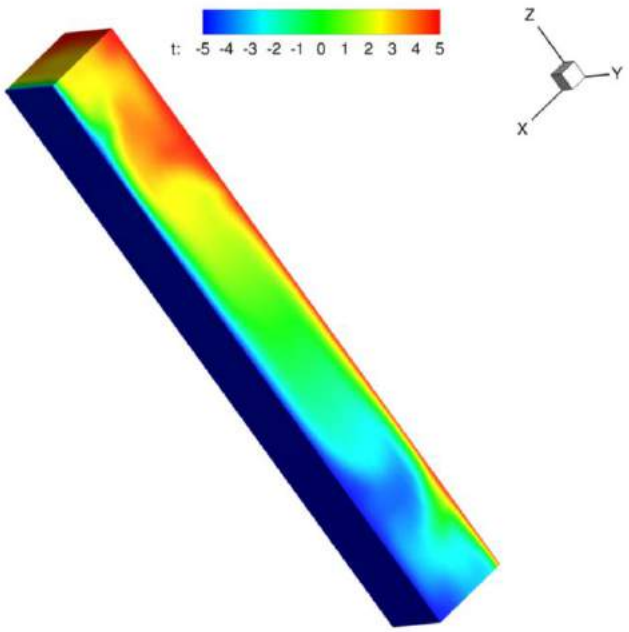


Figure 141: Natural Convection case result in Tecplot

7.4 Tutorial 4: Fluid Flow with Surface Tension

In this tutorial, we will go step by step through the process of setting up a case, running HIMAG and post-processing results. This case is one of the benchmark cases present in the bench folder of HIMAG by the name *brokendam*.

Firstly, we will start with the grid. Creation of gridname.xyz file is the first step, which for this case is called ***brokendam.xyz***.

```
nopt,   nout
  1        0
x-axis: Number of segments (nxsgm),   isymx,  xmin,   xmax
                                      1        0      0.0     5.0
#1:  nxcel,     segment-Xmax
        100     5.0
      Stretch_option, stretching_factor
        0
y-axis: Number of segments (nysgm),   isymy,  ymin,   ymax
                                      1        o       0.0        2.2
#1:  nycel,     segment-Ymax
        60      2.2
      Stretch_option, stretching_factor
        0
z-axis: Number of segments (nzsgm),    isymz,    zmin,      zmax
                                      1        1        0.0        0.1
#1:  nycel,     segment-Ymax
        1       0.1
      Stretch_option, stretching_factor
        0
```

Once we have the brokendam.xyz file ready, then we run XYZ to create a grid using the mentioned parameters.

```
mhd@machine$ xyz
Enter filename [.xyz]: brokendam
```

This will create a ***univ.cemgrd*** file and ***checkXYZ.dat*** file. We can open the checkXYZ.dat file to see if the spacing between cells is reasonable. The checkXYZ.dat file for this case looks like this: (the complete file is not shown, only the start and end is shown here)

(a)

```
 grid       xf            dx            xc
    1  0.000000E+00  5.000000E-02  2.500000E-02
    2  5.000000E-02  5.000000E-02  7.500000E-02
    3  1.000000E-01  5.000000E-02  1.250000E-01
    4  1.500000E-01  5.000000E-02  1.750000E-01
    5  2.000000E-01  5.000000E-02  2.250000E-01
```

(b)

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Multi-segments HEXAHEDRAL grid (nopt= 1) with brokendam.xyz
 min: dx= 5.00000E-02   dy= 3.66667E-02   dz= 1.00000E-01
 max: dx= 5.00000E-02   dy= 3.66667E-02   dz= 1.00000E-01
------------------------------------------------------------------------
 Domain range: x[    0.0000    5.0000]
               y[    0.0000    2.2000]
               z[    0.0000    0.1000]
 Total number of nodes (101 x 61 x  2) =  12322
 Total number of cells (100 x 60 x  1) =   6000
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 142: CheckXYZ.dat file (a) - start of the file and (b) - end of the file

Now that we have univ.cemgrd file and everything looks okay in checkXYZ.dat, we use Book command to create .ux file for the grid which will be used by HIMAG.

```
mhd@machine$ book brokendam.ux
```

This command will give us ***brokendam.ux*** file which can be opened in *HIMAG-Prep* to setup boundary conditions, which is our next step.

To open this file in *HIMAG-Prep*, let's type in the command for *Prep* GUI.

```
mhd@machine$ prep
```

Choose the grid file, i.e. brokendam.ux in the *HIMAG-Prep* window.

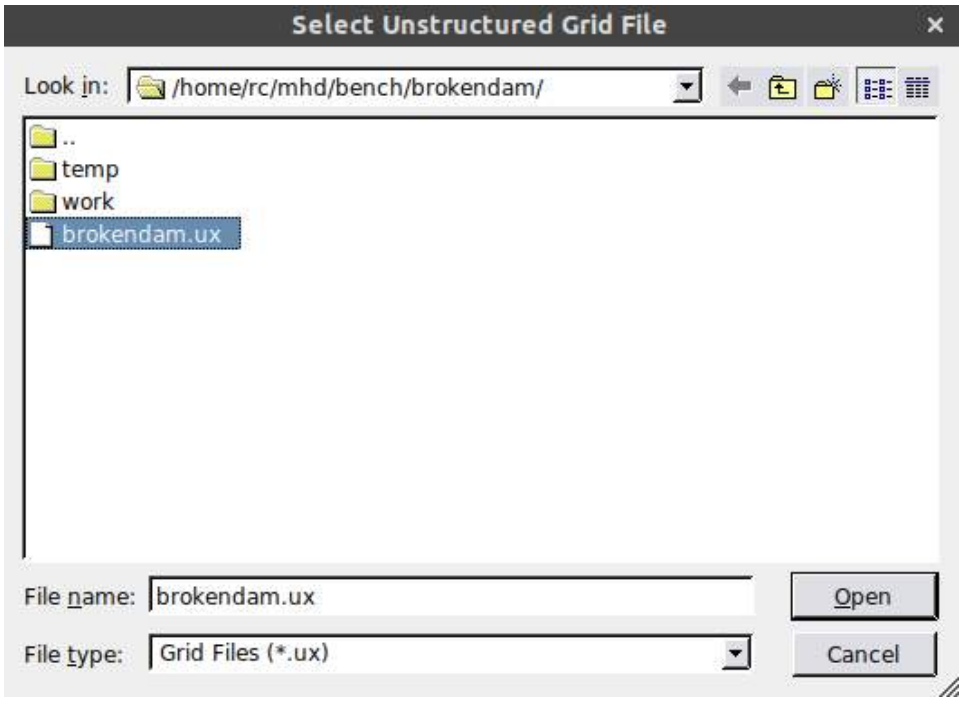


Figure 143: HIMAG-Prep grid selection

The next window will show the work window of *HIMAG-Prep* which will be empty with no grid but only grid data shown in the tabs below with the name, number of cells and nodes (Figure 154).

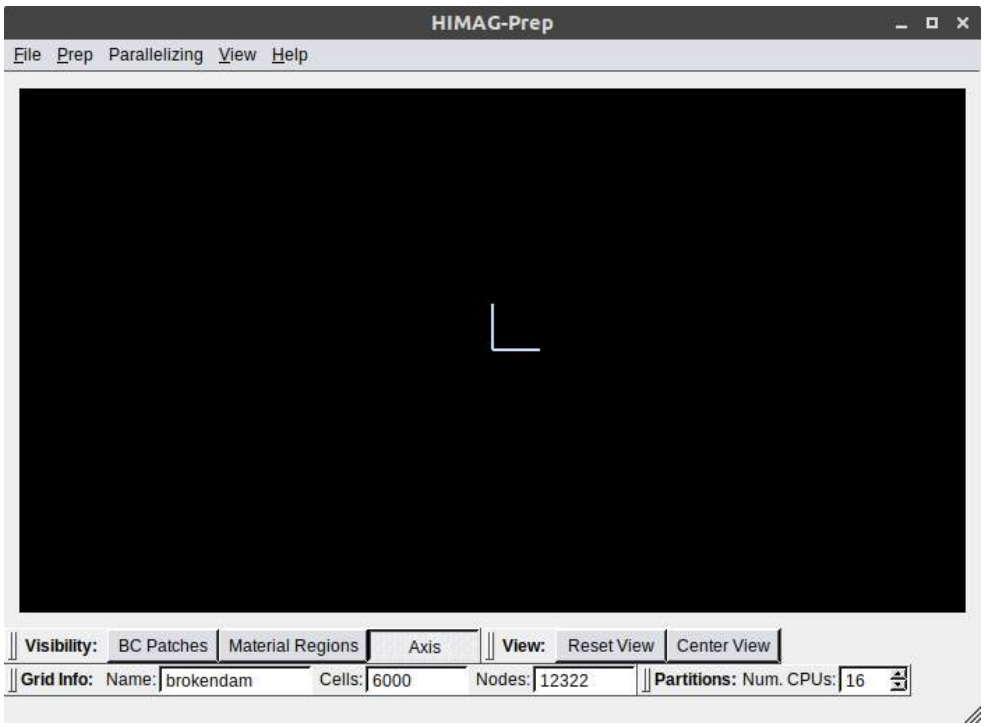


Figure 144: HIMAG-Prep work window

Select *Prep* on the top-left corner of the window and select *Boundary Conditions* under *Prep*. Another window will pop-up for *Boundary Conditions* as discussed in Chapter 4.

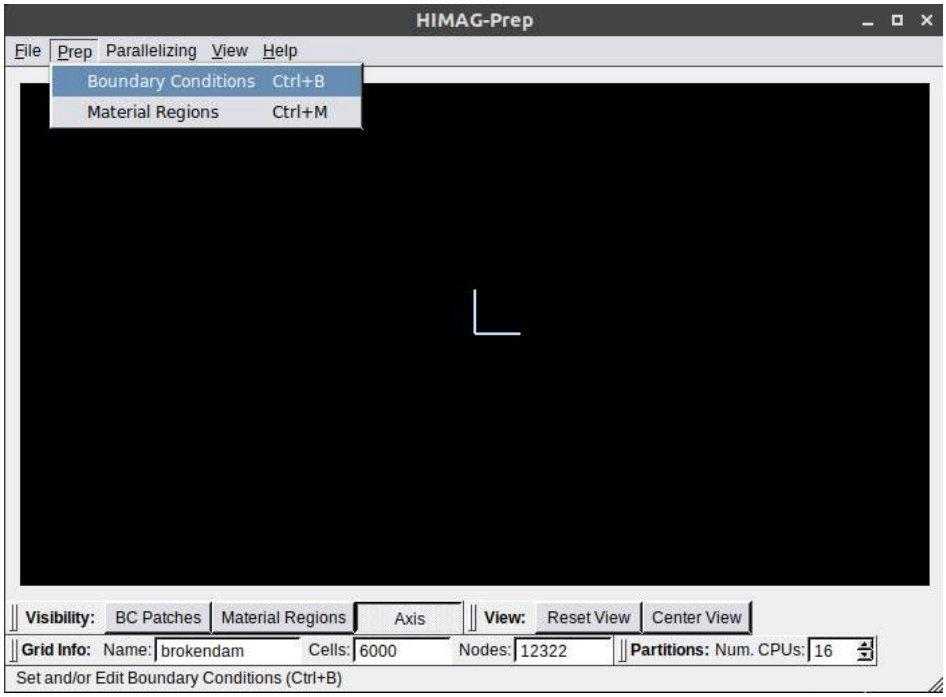


Figure 145: Prep work window showing the location of Boundary conditions tab.

Select *Create* and then *From Free Faces* in Figure 156.

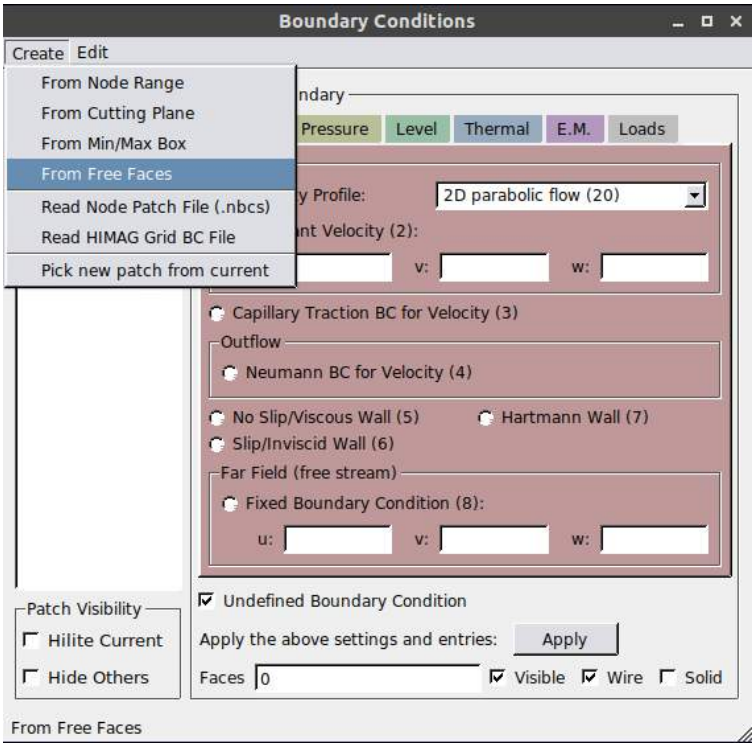


Figure 146: Creating a patch using Cutting Plane in HIMAG-Prep

This will form another patch with all the remaining faces and will complete the grid. The completed grid will look like .

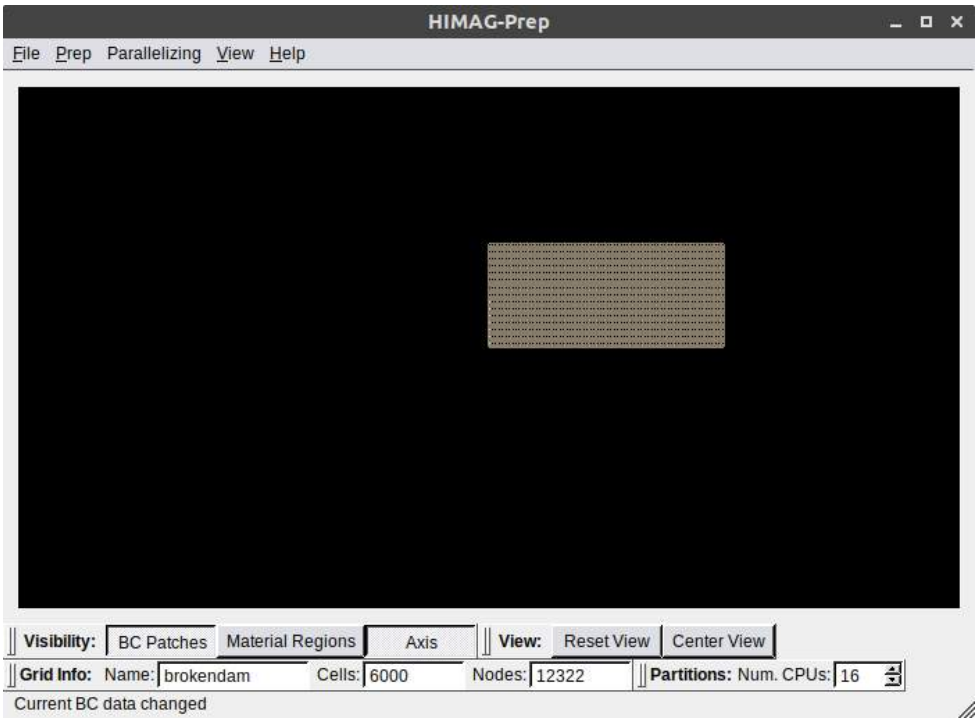


Figure 147: After all the patches are created

The next step is to assign boundary conditions. There is only one patch and for this one we will only apply boundary condition for *Velocity*. Under *Velocity* tab, select *Slip/Inviscid Wall (6)* and click *Apply*. We will notice that as soon as we hit *Apply* the *Loads* tab selects *Don't Include Force/Moment Calcs (0)* on its own.

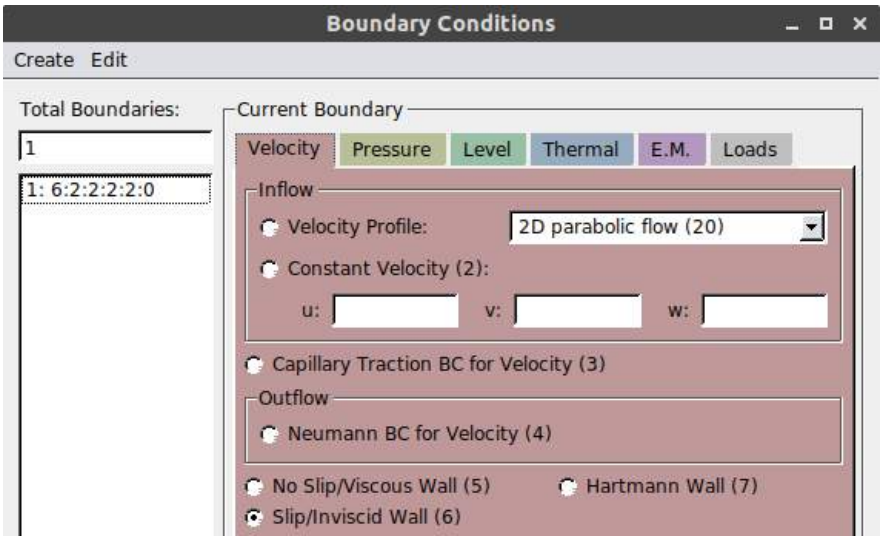


Figure 148: Velocity Boundary Conditions for Patch1

Before closing the window, click on *File* in the *Prep* work window located on the top-left corner of the window. Select *Save .ugb File* in the drop down menu. It will save the .ugb file with the same name as the .ux file, i.e. ***brokendam.ugb***.

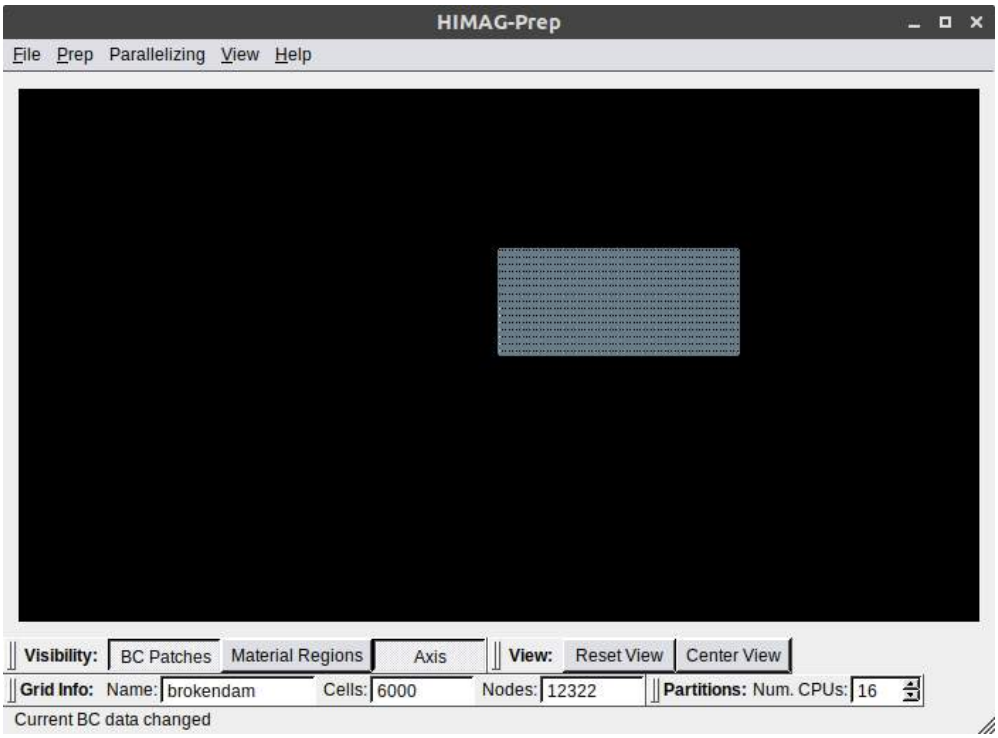


Figure 149: Grid after setting the boundary conditions in HIMAG-Prep

Now, we have the ugb file as well. The next step is to create an input file, ***hmg.input***. This file will look like:

```
 nodes   = 1,          #number of CPUs
iread    = 0,          #0 for reading no restart file
grid_scale = 1.0,      #grid scale will be multiplied to the dimensions

nmax     = 1500,       #Maximum number of run steps
iskip    = 100,        #Output (Tecfiles) files to be written after these many steps
nwrite   = 100,        #Restart files will be written after these many steps
dtime    = 2.0e-3,         # Time-step

visc1    = 1.0e-3,         #Viscosity of the fluid
rho1     = 1.0,        #Density of the fluid
ssig     = 1.0e-2,         #Surface Tension

omega    = 1.00,
sleng    = 2.2,
ubar     = 0.0,        #Initial Velocity
dpdx     = 0.0,
iortho   = 1,          #Orthogonal Correction Applied
```

```
lambda  = 1,
nvel    = 0,

nmomt   = 1,        #Momentum Equation
nppe    = 5,        #Pressure Poisson Equation
nmhd    = 0,        #MHD Poisson Equation
nheat   = 0,        #Heat Equation
ilevels = 11,           #Level Set

ichan   = 0,
nskp    = 1,        #Skip steps to show Screen Outout
ipmax   = 20,       #Maximum Iterations for Pressure Poisson Equation
irmax   = 10,       #Maximum Iterations for Level Set
epsmin  = -16.0,    #Maximum Convergence Residual

igravt  = 1,        #Gravity ON/OFF
gx      = 0,
gy      = -1,
gz      = 0,
```

Once the input file is ready, we are ready to run HIMAG for this case. Assuming that HIMAG is already built on the machine and the executable file is available for use and added to the path. The command that is used to run HIMAG is:

```
mhd@machine$ himag -i hmg -g brokendam | tee logfile
```

This will start running HIMAG for the case *brokendam* and the screen output will look like **Error! Reference source not found.** and when the case finishes, the screen output will look like **Error! Reference source not found.**. This means that the output files have been generated and are ready to be post-processed.

```
Run himag0 with (hmg.input brokendam.ux)
HIMAG MPI run with 2 node(s) started - node 1 on salus
Wed May 25 13:53:26 PDT 2016
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
HIMAG MPI run with 2 node(s) started - node 0 on salus
himag: version $Id: idrive.c,v 1.23 2015/11/05 00:53:02 pyh Exp $
Reading input parameters from ->hmg.input<-
*** Warning: nodes=2 over-ridden at command line
-----------------------------------------------------------------------
# of BC data per patch: nitems= 7
node 0, finished load_ugb: total 6162 bc faces
node 1, finished load_ugb: total 6158 bc faces
-----------------------------------------------------------------------
Channel without wall domain
-----------------------------------------------------------------------
Double precision run for grid brokendam.ux
             | (xmin xmax) = (  0.0000E+00  5.0000E+00) * 1.0000E+00
Fluid domain | (ymin ymax) = (  0.0000E+00  2.2000E+00) * 1.0000E+00
             | (zmin zmax) = (  0.0000E+00  1.0000E-01) * 1.0000E+00
 dpdx=  0.00000E+00
 Two Phase flow calculation                        sleng= 2.2000E+00
  U= 0.0000E+00   Re1=0.000E+00   Ha1=0.000E+00   Cw=0.0000E+00
                  Re2=0.000E+00   Ha2=0.000E+00
-----------------------------------------------------------------------
New Start: nmax=    1500
 Inlet BC: invel= 0
 Initial velocity: 0.0
Volume= 1.10000E+00    Inlet_Area= 0.00000E+00    Inflow Mass-rate=  0.00000E+00
-----------------------------------------------------------------------
ppe-CG step =        1   200  -6.97218E+00
level set residual =   0.00000E+00   0.00000E+00
reinitialization res at it= 101  -3.30583E+00  -3.35321E+00
Tstep=        1  2.00000E-03  0.00000E+00  0.00000E+00  2.37073E-03 -3.26711E-03
```

Figure 151: Screen Output on starting the case with HIMAG

```
ppe-CG step =     1500    20  -6.99026E+00
level set residual =  -9.77787E-01  -1.64612E+00
reinitialization res at it=  11  -2.73810E+00  -3.13716E+00
Tstep=     1500  2.00000E-03  0.00000E+00  0.00000E+00  2.28790E+00 -1.52697E+00
------ Grid Maximums (CPU, Value) ------
Memory               0:  11.509528
Cells                0:  3000
Interior Faces       0:  12115
PEC Faces            0:  0
Farfield Faces       0:  0
Communication Faces  0:  68
Communication Nbrs   0:  1

------ Timing Maximums (CPU, Value) ------
Preprocessing  0:  0.109375
Solve          0:  -1.000000
Communication  1:  2.257812
Wait           1:  1.085938
S+C+W          0:  0.000000
RCS            0:  0.000000
Total          0:  43.593750
```

Figure 150: Screen Output at the End of the simulation

Next step is to post-process the output files (tecfiles). The ***tec.cpu.step.dat*** files will be post-processed for the last step, step 1,500.

```
mhd@machine$ mhd2tec brokendam.ux -B 1500
```

This step will give us a file in format, gridname.tec.step.dat. For this case, it will be ***brokendam.tec.1500.dat***. Next step is to convert .dat to .plt using Preplot, a tecplot utility, since we will be viewing the result in tecplot.

```
mhd@machine$ preplot brokendam.tec.1500.dat
```

Running this command will give us .plt file in the format gridname.tec.step.plt. In this case it will be ***brokendam.tec.1500.plt***. Now, we will view the result file in tecplot. To open the file in tecplot, the command is given as:

```
mhd@machine$ tec360 brokendam.tec.1500.plt
```

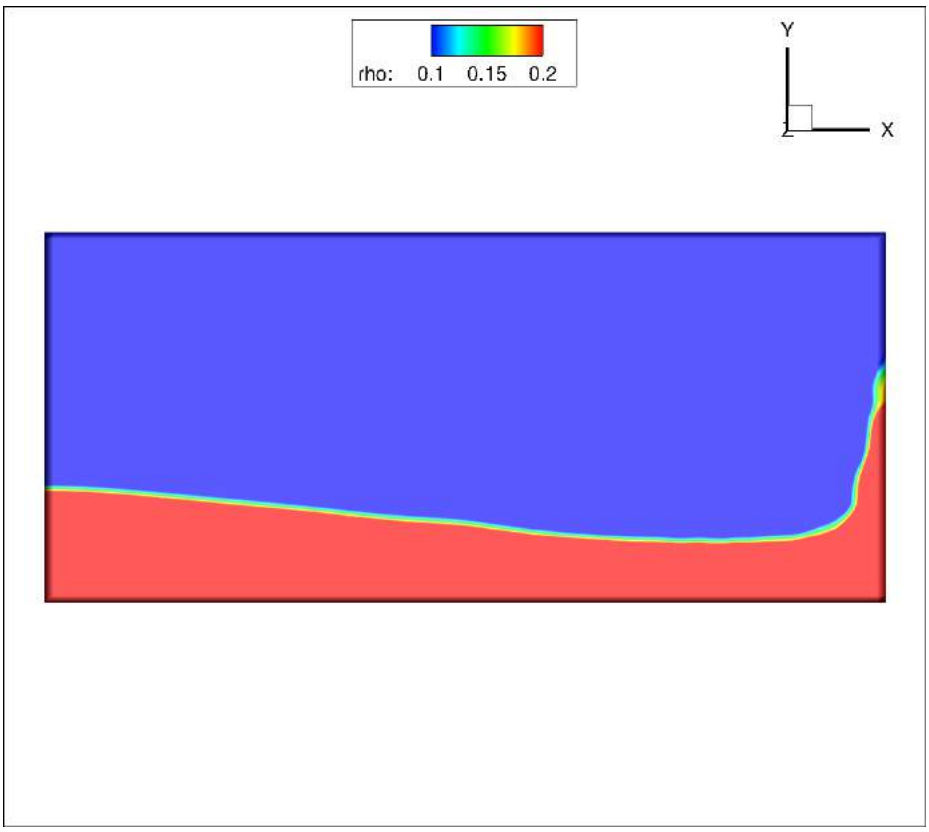


Figure 152: Brokendam case result in Tecplot

7.5 Tutorial 5: Using A-Φ Formulation

In this tutorial, we will use A-Φ formulation for an applied pulse of current. This case is one of the benchmark cases present in the bench folder of HIMAG by the name wire3D.

Firstly, we will start with the grid. Creation of *gridname.xyz* file is the first step, which for this case is called ***wire3d.xyz***.

```
nopt,   nout
  50       0
x-axis: Number of segments (nxsgm),   isymx,  xmin,   xmax
                                      1          0        0.0     120.0
#1:  nxcel,      segment-Xmax
      120        120.0
      Stretch_option, stretching_factor
      0
r-axis: Number of segments (nrsgm),   isymr,  rmin,   rmax
                                      2          0        0.0      1.1
#1:  nycel,      segment-Ymax
      25         11.3
      Stretch_option, stretching_factor
      2
      1.4
#2:  nycel,      segment-Ymax
      4          1.0
      Stretch_option, stretching_factor
      0
Theta: ncel, angle1, angle2
            9      0.0      90.0
```

Once we have the ***wire3d.xyz*** file ready, then we run ***xyz*** to create a grid using the mentioned parameters.

```
mhd@machine$ xyz
Enter filename [.xyz]: wire3d
```

This will create a ***univ.cemgrd*** file and ***checkXYZ.dat*** file. Now that we have univ.cemgrd file, we use *book* command to create .ux file for the grid which will be used by HIMAG.

```
mhd@machine$ book wire3d.ux
```

This command will give us ***wire3d.ux*** file which can be opened in *HIMAG-Prep* to setup boundary conditions, which is our next step.

To open this file in *HIMAG-Prep*, let's type in the command for PREP GUI.

```
mhd@machine$ prep
```

Choose the grid file, i.e. *wire3d.ux* in the ***HIMAG_Prep*** window.

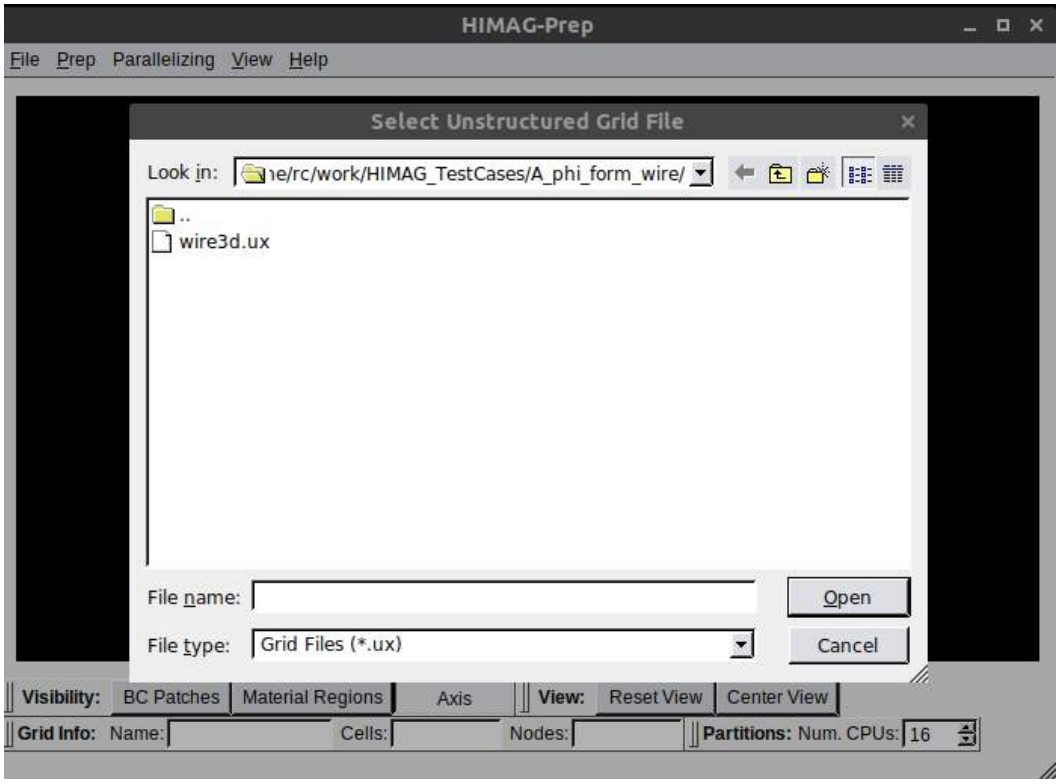


Figure 153: HIMAG-Prep grid selection

The next window will show the work window of ***HIMAG-Prep*** which will be empty with no grid but only grid data shown in the tabs below with the name, number of cells and nodes (Figure 154).

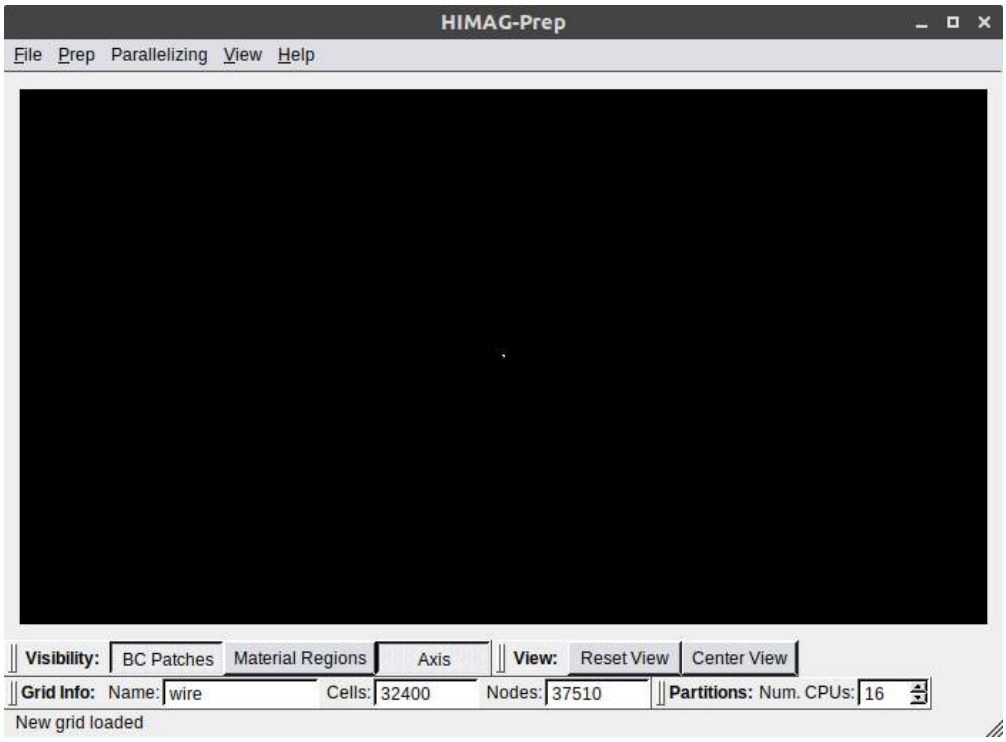


Figure 154: HIMAG-Prep work window

Select *Prep* on the top-left corner of the window and select *Boundary Conditions*. Another window will pop-up for *Boundary Conditions* as discussed in Chapter 4.

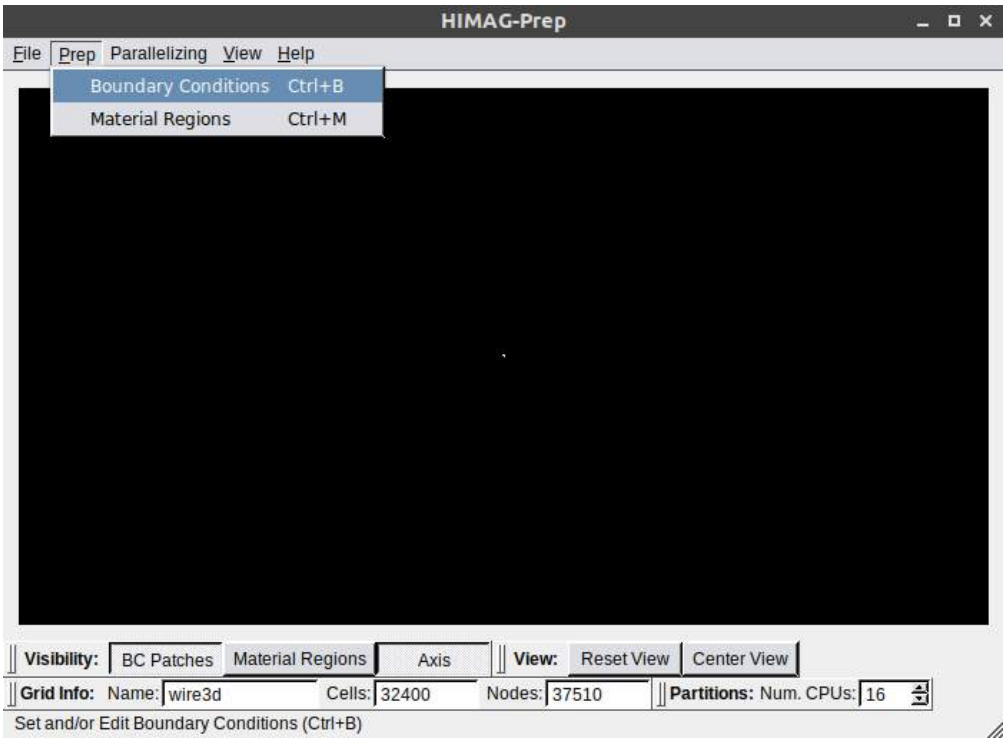


Figure 155: Prep work window showing the location of Boundary conditions tab.

In the *Boundary Conditions* dialogue box, select *Create* and then *From Cutting Plane* in Figure 156.

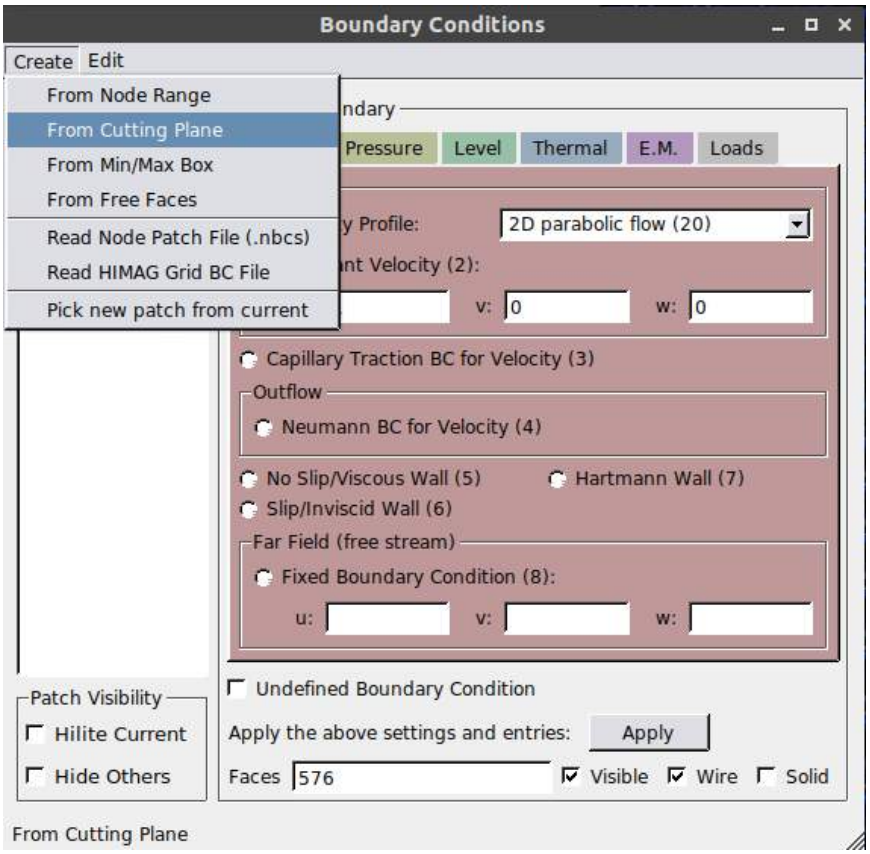


Figure 156: Creating a patch using Cutting Plane in PREP

A pop-up dialogue box (Figure 172) will open asking for the plane where the patch will be created. Let's first select x.

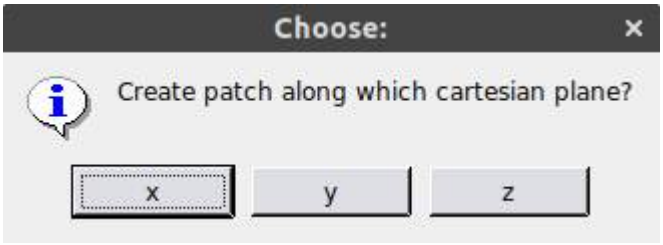


Figure 157: Pop-up dialogue box asking to select the plane

Next, you will be asked to enter tolerance. Let it stay the default value of 0.001.

Figure 158: Dialogue box asking for tolerance

The next dialogue box will be to input the x-plane value where the cut will be made for the patch to be made. Enter the value, 0.0.

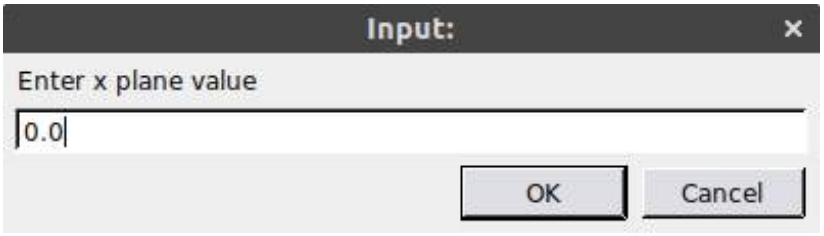


Figure 159: Dialogue box asking for plane value

The next thing you will notice is a patch made in the *HIMAG-Prep* work window and a patch entry in the *Boundary Conditions* window which will show as 2:2:2:2:2:2.

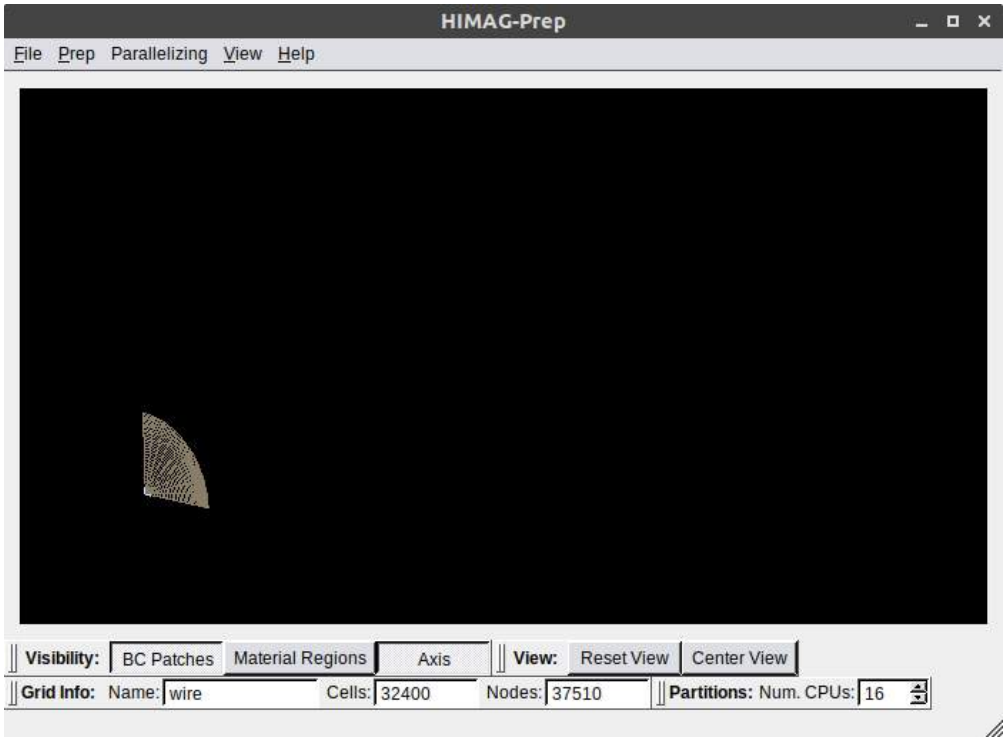


Figure 160: HIMAG-Prep work window after the first patch is created

Similarly, select *From Cutting Plane* under *Create* again and choose x-axis for the plane for creating the patch. Let the tolerance remain as it is and enter the x-plane value to be 120.0 (Figure 161).

Figure 161: Dialogue box asking for plane value

Click on OK to see another patch form in the *HIMAG-Prep* work window (Figure 162).

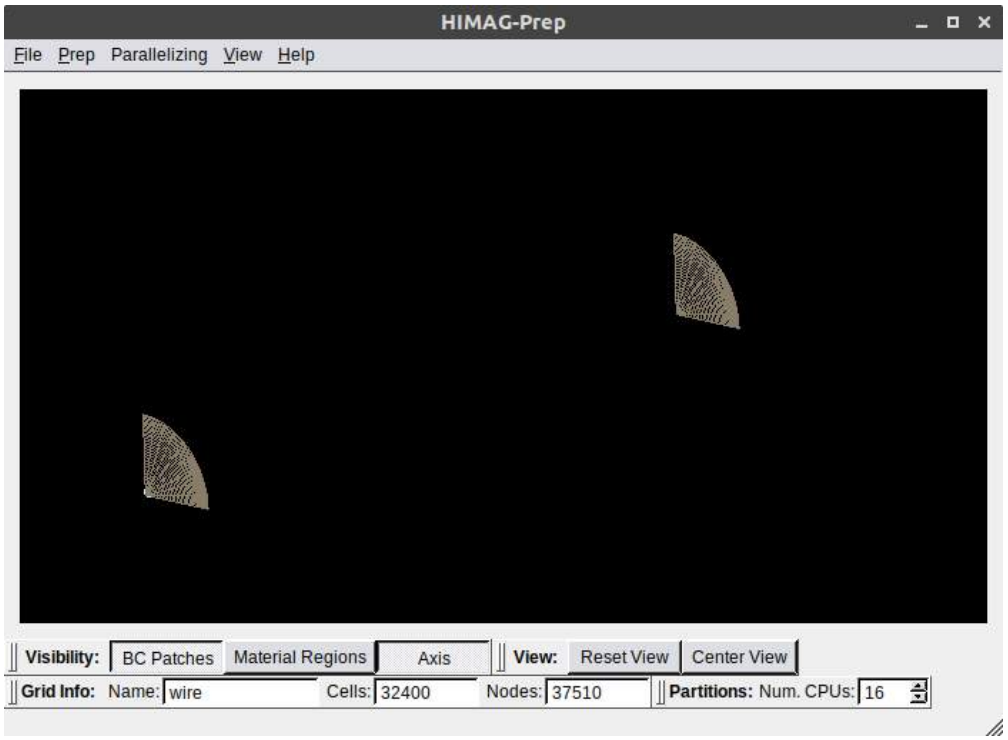


Figure 162: Prep window after second patch is created

Again, select *From Cutting Plane* under *Create* in the *Boundary Conditions* window. This time choose y-axis for the plane for creating the patch.

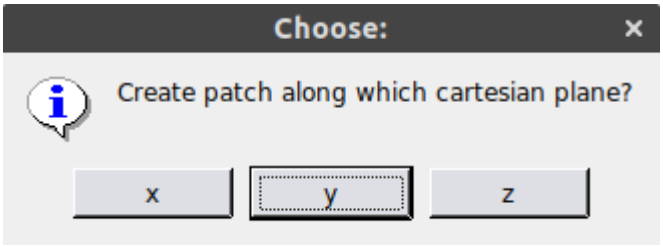


Figure 163: Pop-up dialogue box asking to select the plane

Click OK for tolerance and enter the y-plane value to be 0.0 (Figure 164).

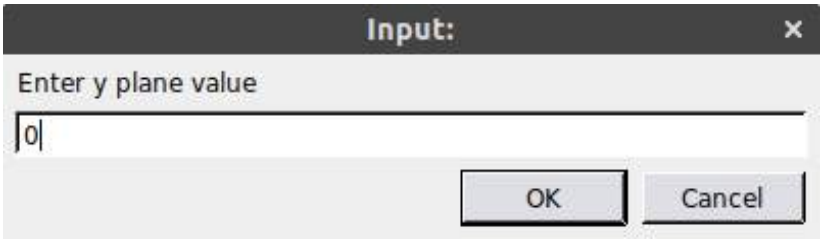


Figure 164: Dialogue box asking for plane value

This will create another patch which can be seen in the *HIMAG-Prep* work window (Figure 175).

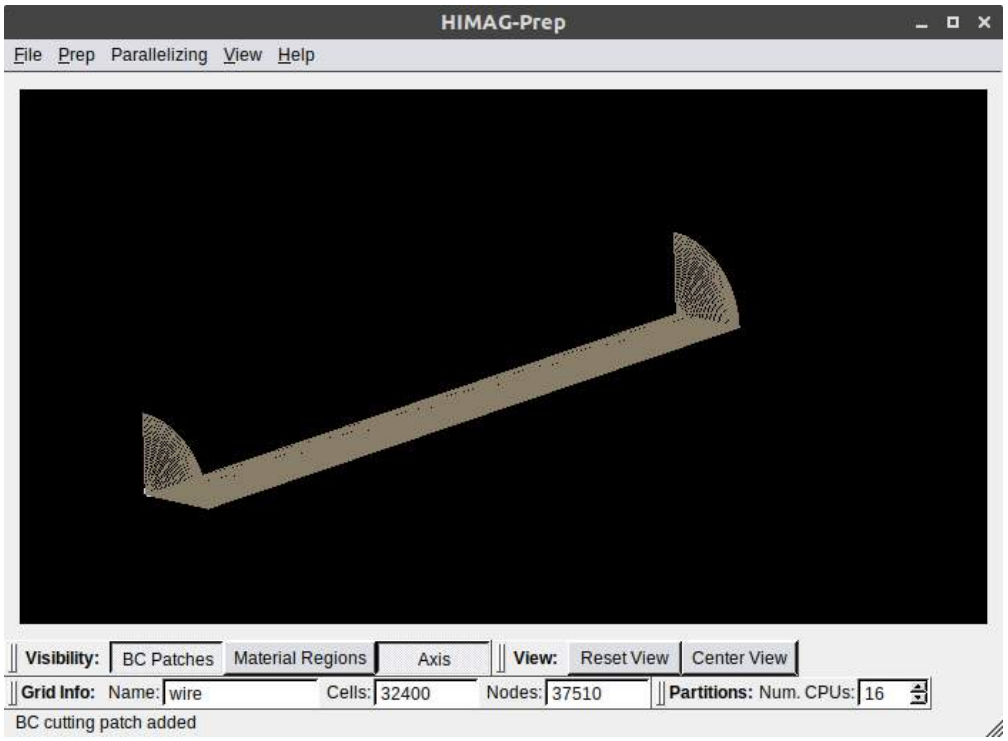


Figure 165: Prep window after third patch is created

Again, select *From Cutting Plane* under *Create* in the *Boundary Conditions* window. This time choose z-axis for the plane for creating the patch.

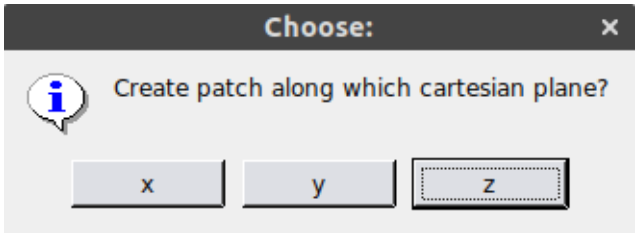


Figure 166: Pop-up dialogue box asking to select the plane

Let the tolerance remain as it is and enter the z-plane value to be 0.0 (Figure 177).

Figure 167: Dialogue box asking for plane value

Click on OK to see another patch form in the *HIMAG-Prep* work window (Figure 178).

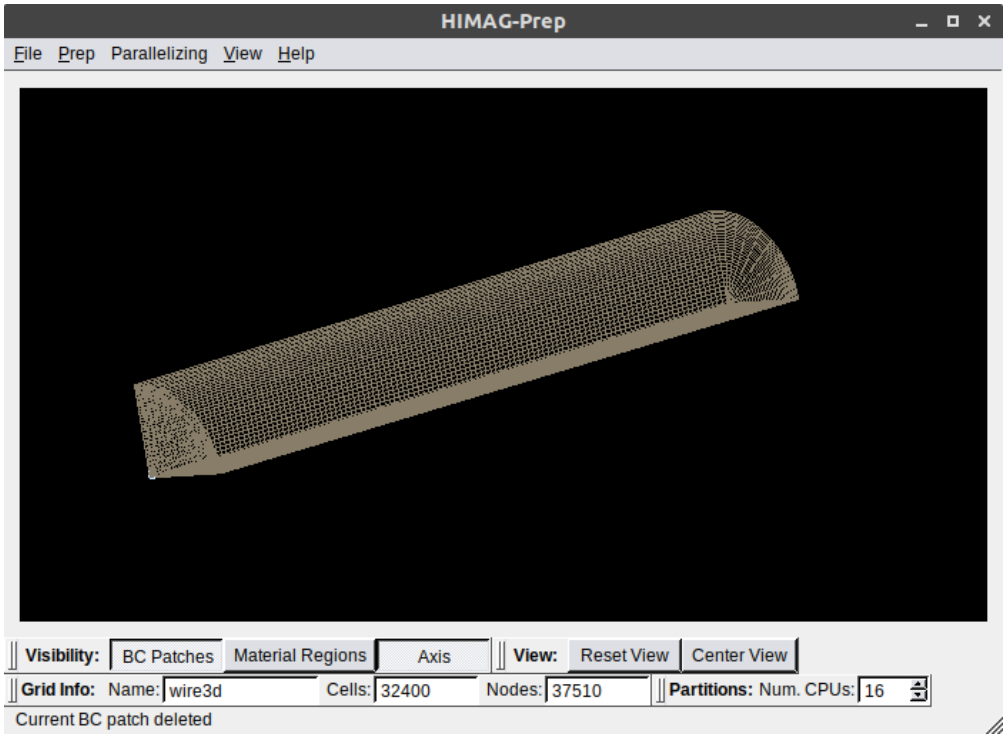


Figure 168: Prep window after fourth patch is created

The next step is to create the remaining faces which can be created by selecting *From Free Faces* under *Create* tab on *Boundary Conditions* window.

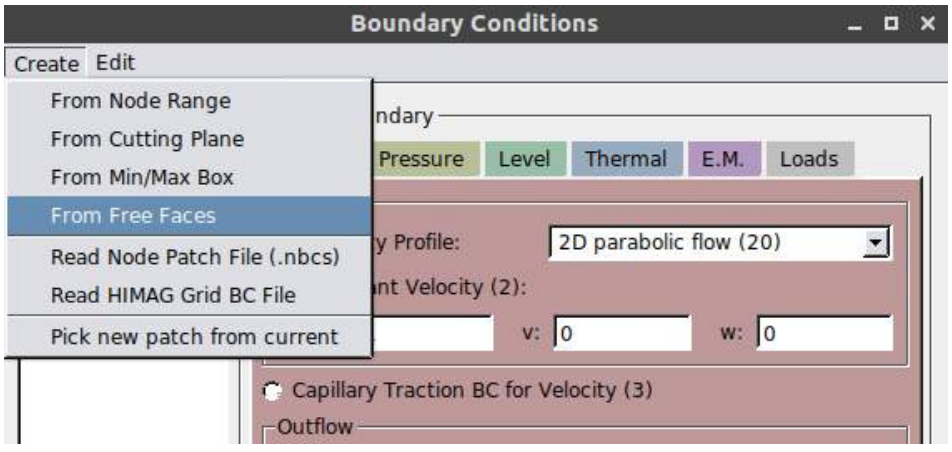


Figure 169: From Free Faces for the Grid

This will complete the grid and it will look like Figure 170.

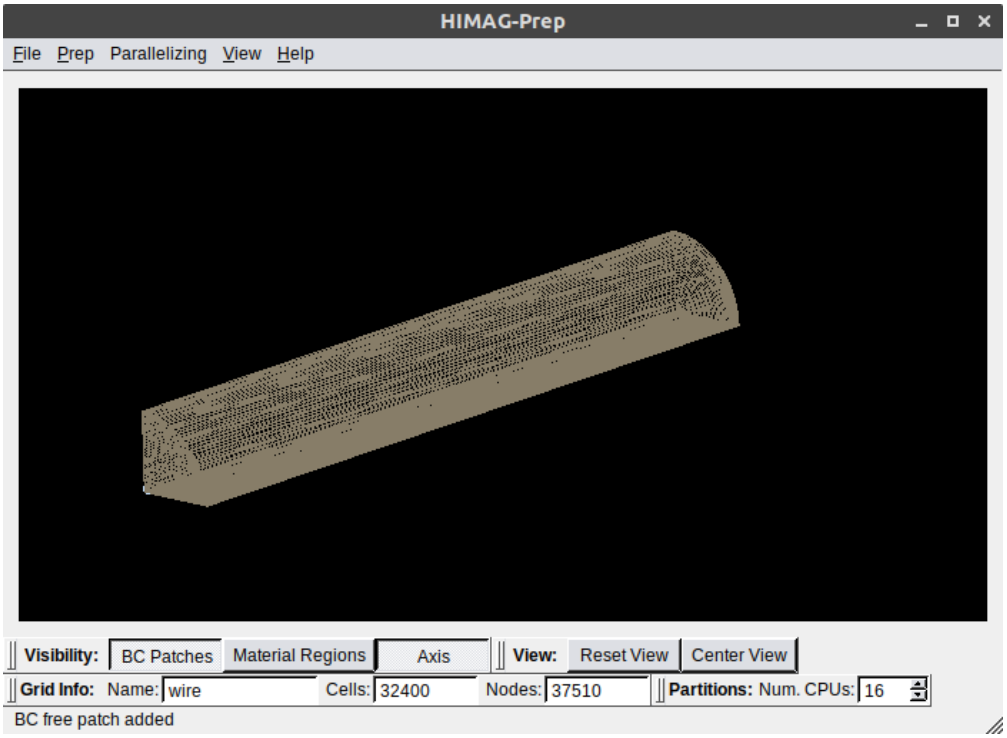


Figure 170: After all the patches are created

We can combine the patches created at y = 0 and z = 0, i.e. patch 3 and patch 4. Select Patch 3 in the Boundary Conditions dialogue box and open the dropdown box for Edit in the Main Menu. Select Combine Patches as shown in Figure 171.

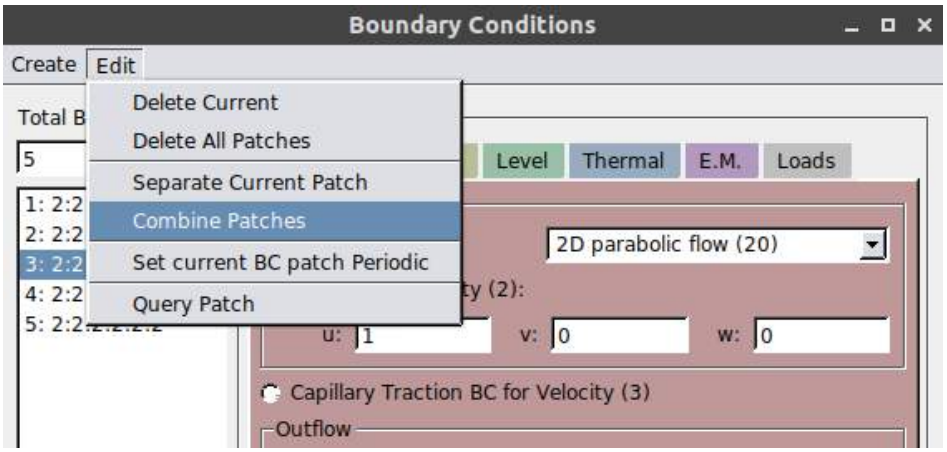


Figure 171: Combine 2 patches

It will open a dialogue box asking the patch number to which the selected patch will be combined with (Figure 172).

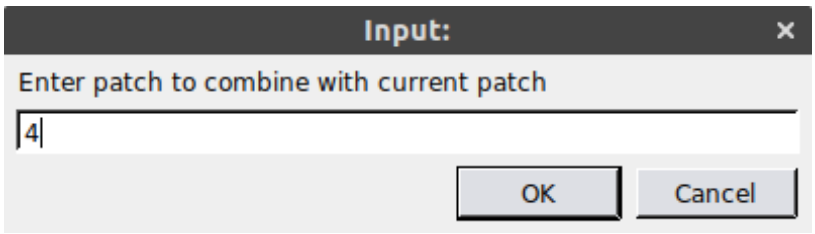


Figure 172: Pop-up dialogue box asking to select the plane

We can see in the Boundary Conditions dialogue box that now we have only 4 patches.

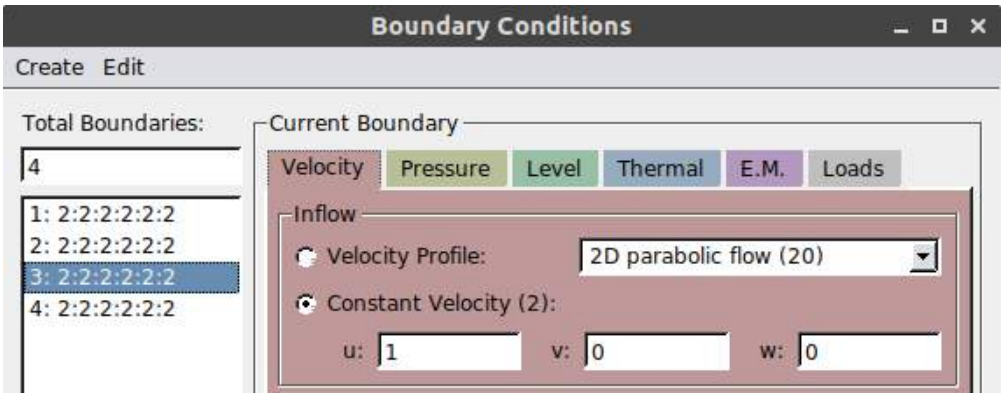


Figure 173: Combined Patches

If you select patch 3, you'll notice in the HIMAG-Prep work window that the patches for y = 0 and z = 0 are highlighted.

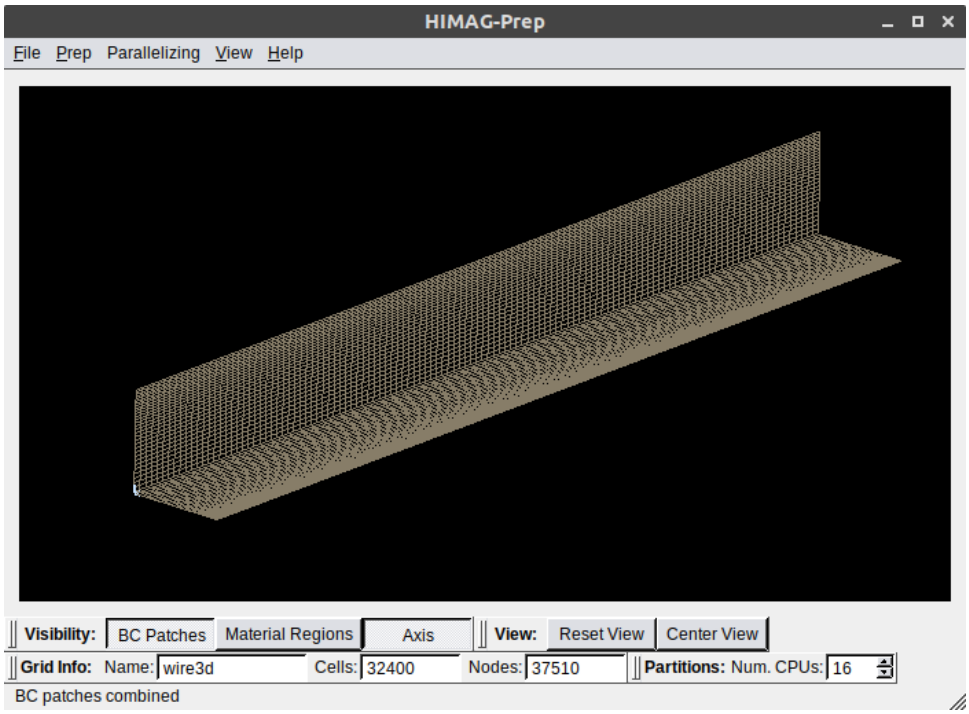


Figure 174: Highlighted patch3

The next step is to assign boundary conditions. Let's start with patch 1. Under *Velocity* tab, select *No Slip/Viscous Wall (5)* and click *Apply*.

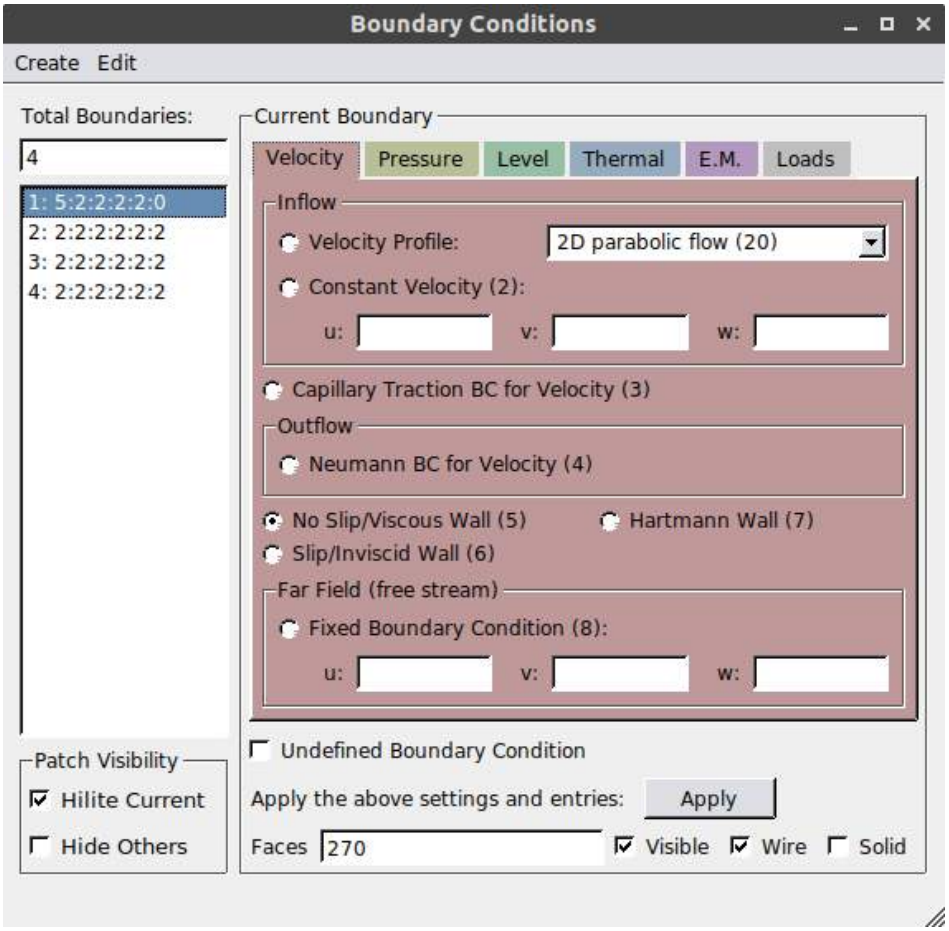


Figure 175: Velocity Boundary Conditions for Patch1

Next select *EM* tab and select *Specified Phi (1)* and enter the value 40 next to it and click *Apply* as shown in Figure 176.

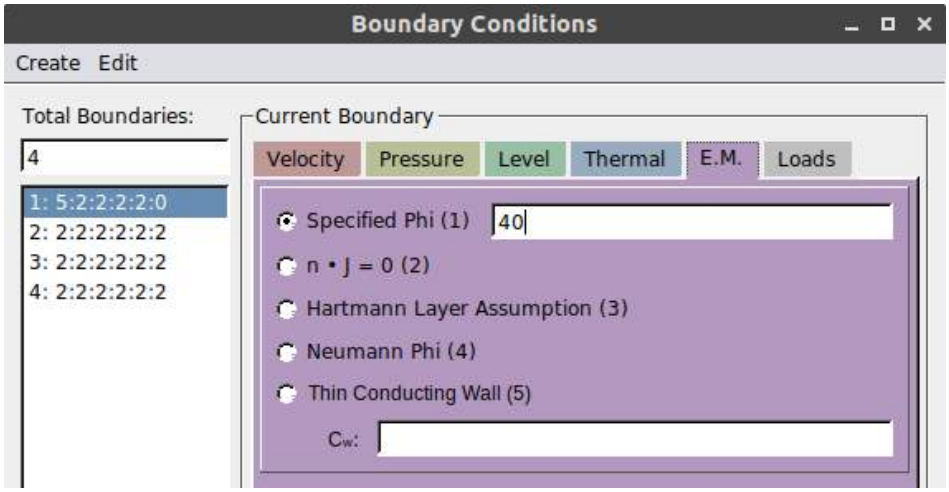


Figure 176: EM Boundary Condition for Patch1

Next is patch 2. Again in *Velocity* tab, choose *No Slip/Viscous Wall (5)* as shown in Figure 177.

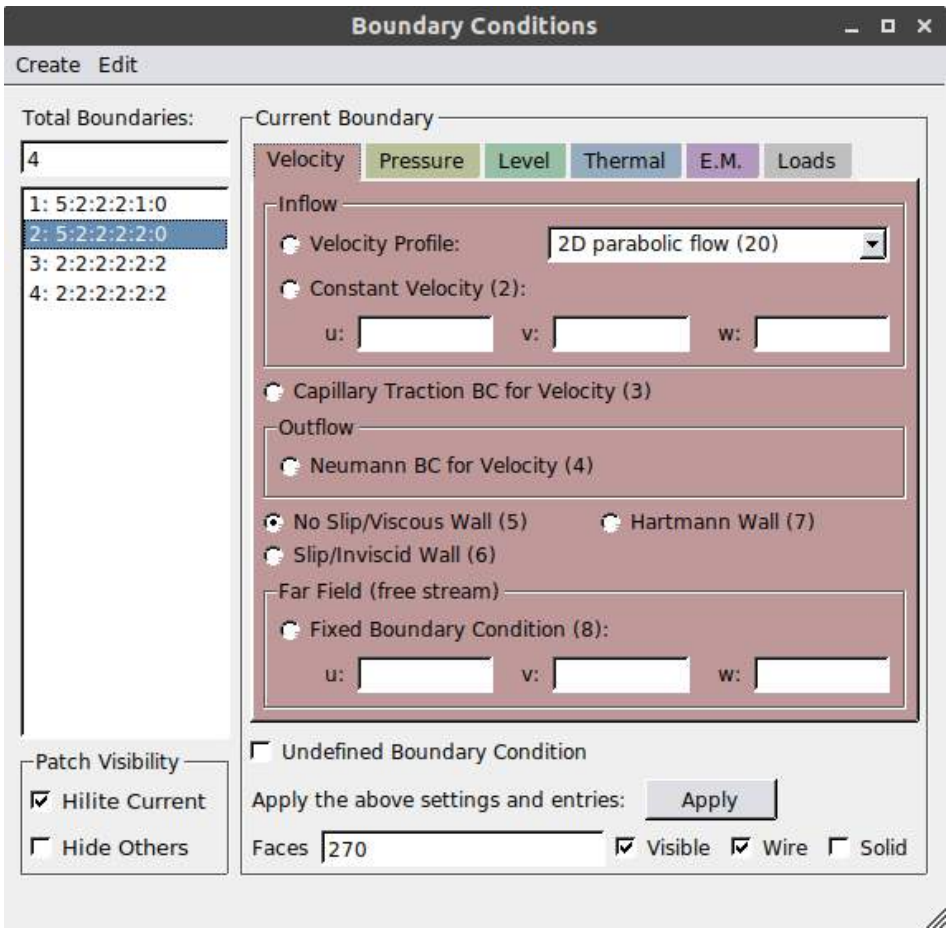


Figure 177: Velocity Boundary Condition for Patch2

For patch 2, in EM tab, select *Specified Phi (1)* and put the value as 0.0 as shown in Figure 178.

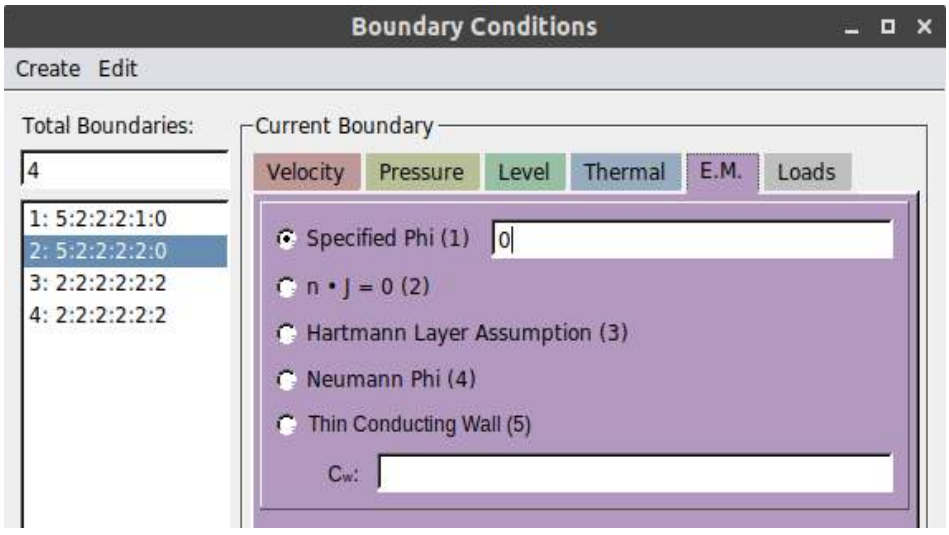


Figure 178: EM Boundary Condition for Patch2

Moving on to patch 3. In *Velocity* tab, select *No Slip/Viscous Wall (5)* and hit *Apply*.

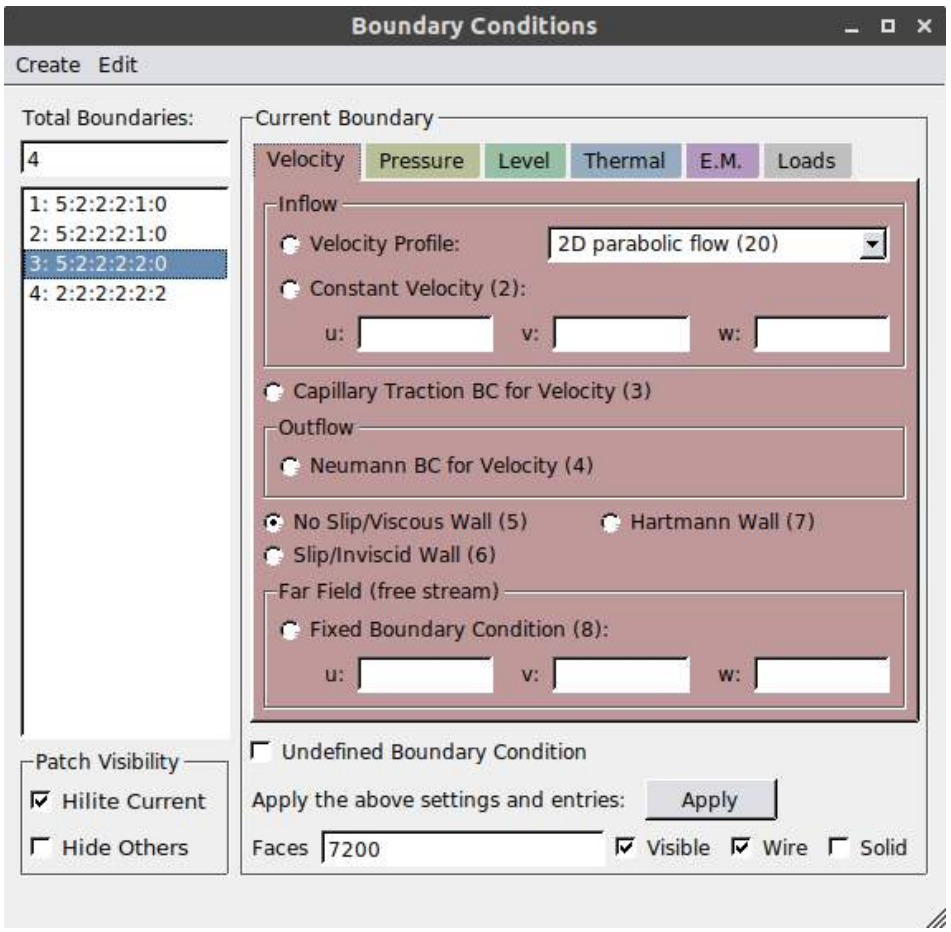


Figure 179: Velocity Boundary Condition for Patch3

For patch 3, in *EM* tab, select *Neumann Phi (4)* and click *Apply* as shown in Figure 180.

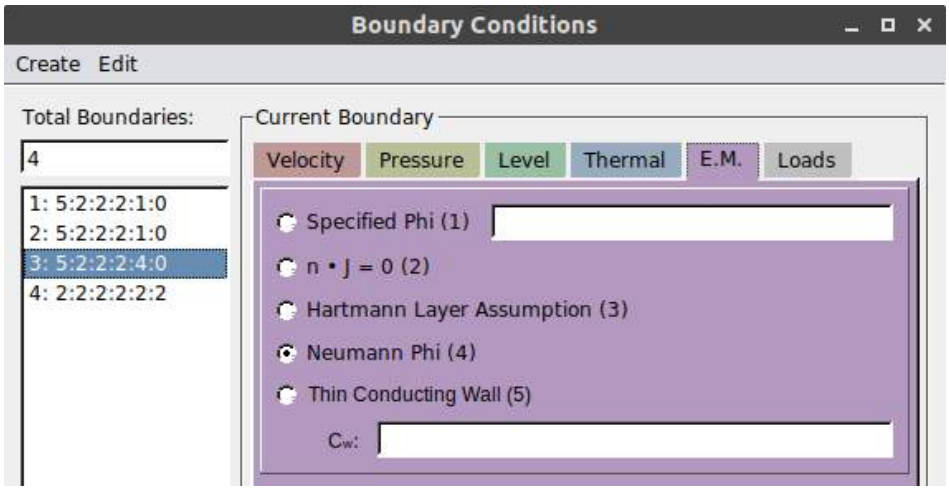


Figure 180: EM Boundary Condition for Patch3

Lastly for patch 4 also, we select *No Slip/Viscous Wall (5)* in the *Velocity* tab.

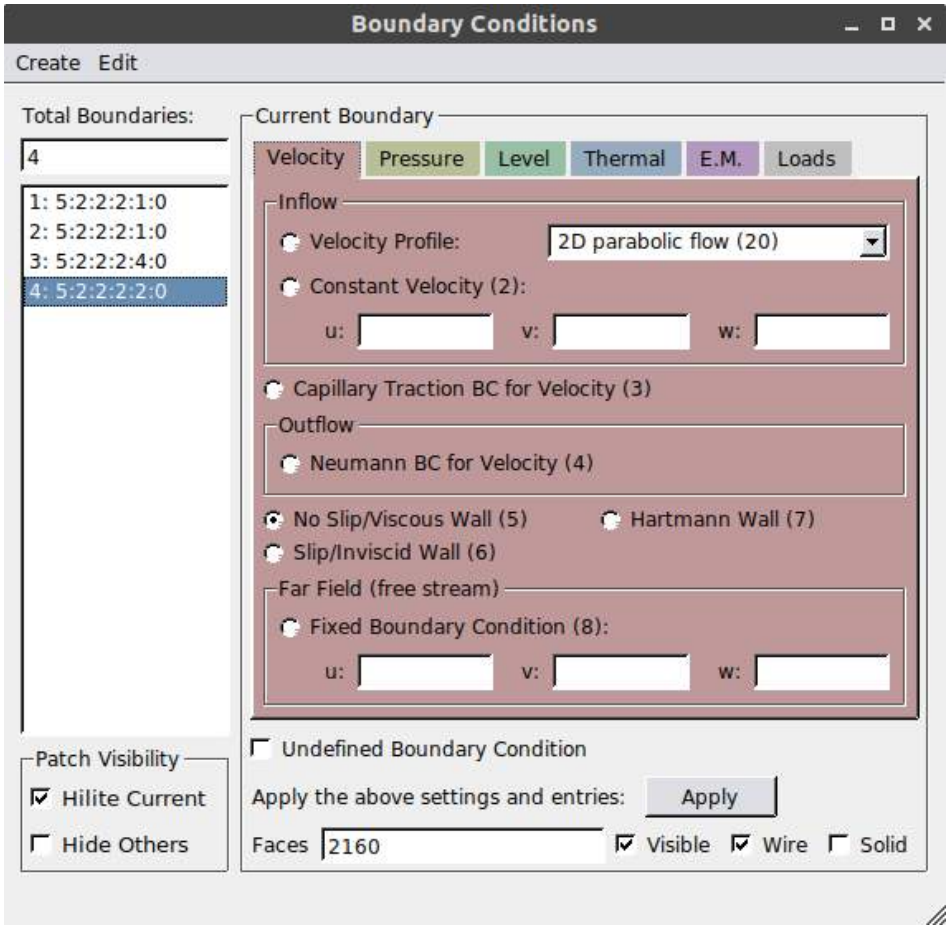


Figure 181: Velocity Boundary Condition for Patch4

For *EM* boundary condition, select *Neumann Phi (4)* for patch4 and click on *Apply* as shown in Figure 182.

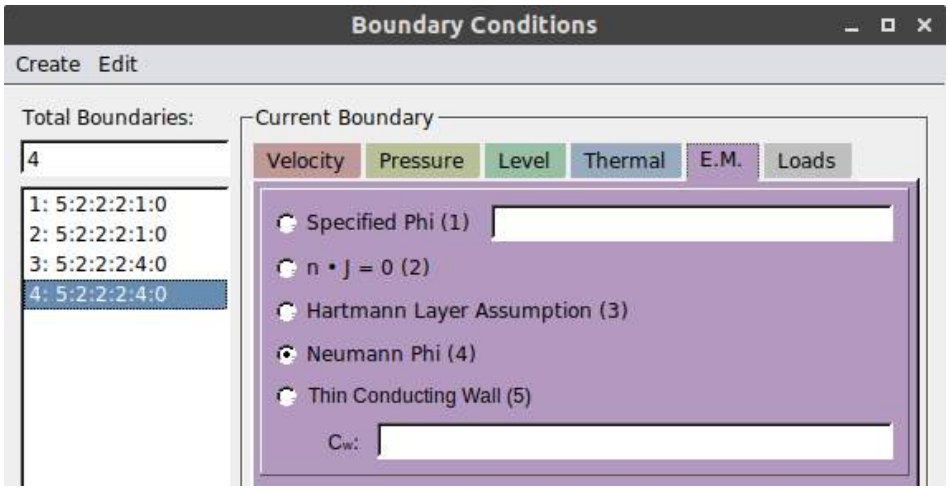


Figure 182: EM Boundary Condition for Patch4

Before closing the window, click on *File* in the *Prep* work window located on the top-left corner of the window. Select *Save .ugb File* in the dropdown menu. It will save the .ugb file with the same name as the .ux file, i.e. ***wire3d.ugb***.

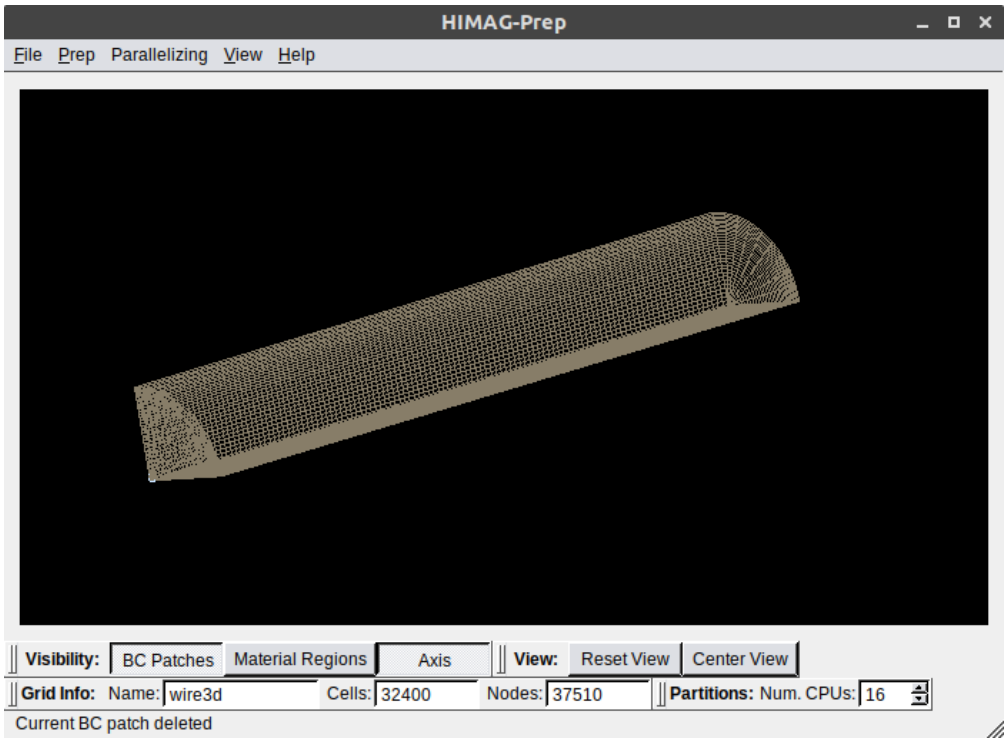


Figure 183: Grid after setting the boundary conditions in PREP

Once this is done, now we have to set boundary conditions for magnetic-vector potential or **A**. Open an empty bc_ugb.dat file for A-boundary conditions.

```
mhd@machine$ vi bc_ugb.dat
```

Now, enter values the following values. Details for these boundary conditions are given in 4.1.3.

```
0       3
1       10
2       10
3       10
```

Save this file and run the modification command.

```
mhd@machine$ mod_ugb wire3d.ugb
```

The modified grid file can be viewed in HIMAG-Prep.

```
mhd@machine$ prep
```

This will open the new grid in the HIMAG-Prep work window.

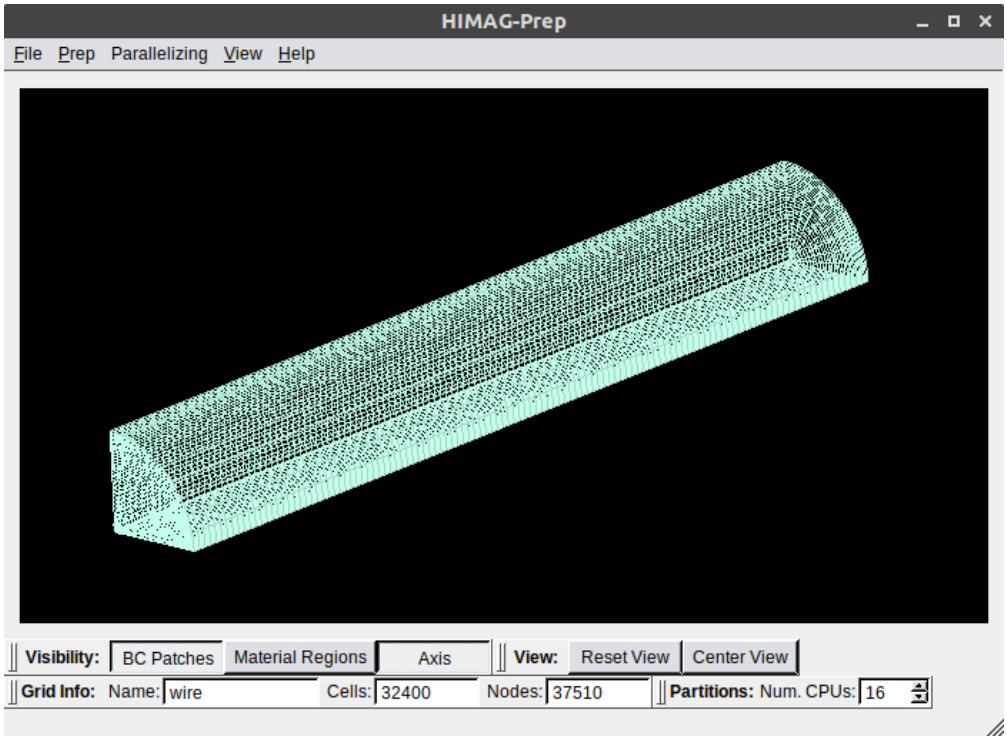


Figure 184: Grid after setting the A-boundary conditions

The modified boundary conditions can be seen in the Boundary Conditions dialogue box as shown in Figure 185.

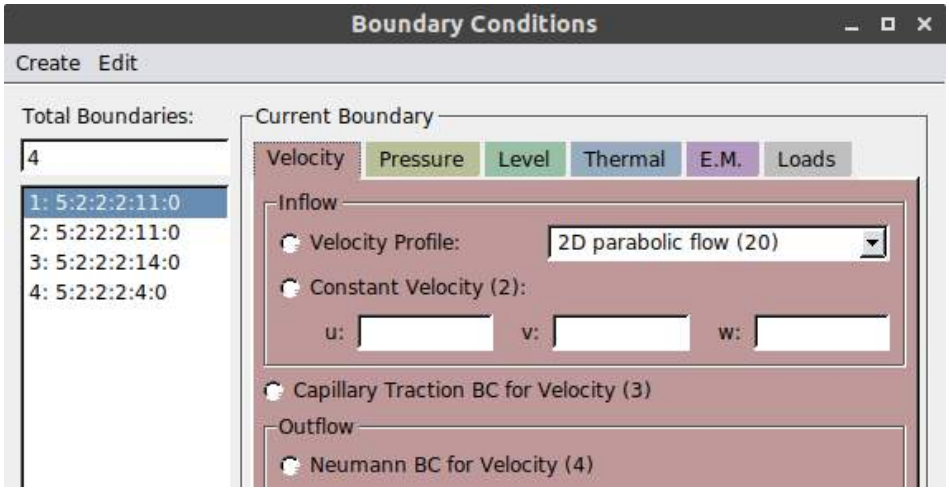


Figure 185: Modified boundary conditions

Now, we have the modified grid file ready. The next step is to create an input file, ***hmg.input***. This file will look like:

```
nodes    = 1,          #number of CPUs
iread    = 0,          #0 for reading no restart file
grid_scale = 1.0e-3,   #Grid scale will be multiplied to the dimensions

nmax     = 6000,       #Maximum number of run steps
iskip    = 100,        #Output (Tecfiles) files to be written after these many steps
nwrite   = 1000,       #Restart files will be written after these many steps
dtime    = 1.0e-9,     # Time-step
ovrelx   = 0.5,

#Material Properties
rhow1    = 7800,       #Density of the fluid
sgmw1    = 1e+6,       #Electrical Conductivity of solid wall

#Physical Parameters
sleng    = 12.0,
ubar     = 0.0,        #Initial Velocity
twal     = 0.0,        #Wall Thickness
dpdx     = 0.0,        #Initial Pressure Gradient
ichan    = 100,
iuvw     = 1,

#Magnetic Field
bval     = 0.0,        #Value of Magnetic Field
ibf      = 3,
nbf      = 104,

#Solver Options
iortho   = 1,          #Orthogonal Correction Applied
nmomt    = 0,          #Momentum Equation
nppe     = 0,          #Pressure Poisson Equation
nmhd     = 5,          #MHD Poisson Equation
nheat    = 0,          #Heat Equation
ilevels  = 0,          #Level-set Equation

#Numerical Options
ngrad    = 7,
nvel     = 0,
nskp     = 1,          #Skip steps to show Screen Outout
ipmax    = 50,         #Maximum Iterations for Pressure Poisson Equation
immax    = 3000,       #Maximum Iterations for MHD Equation
nsbmx    = 1000,
ischme   = 17,
epsmin   = -16.0,      #Maximum Convergence Residual
omega    = 1.0,
ncase    = 100,
```

Once the input file is ready, we are ready to run HIMAG for this case. The current we are applying for this case is in the form of a sinusoidal wave with a frequency of 1e+6 (Figure 186), so we will have to create a user.dat file with those specific values.

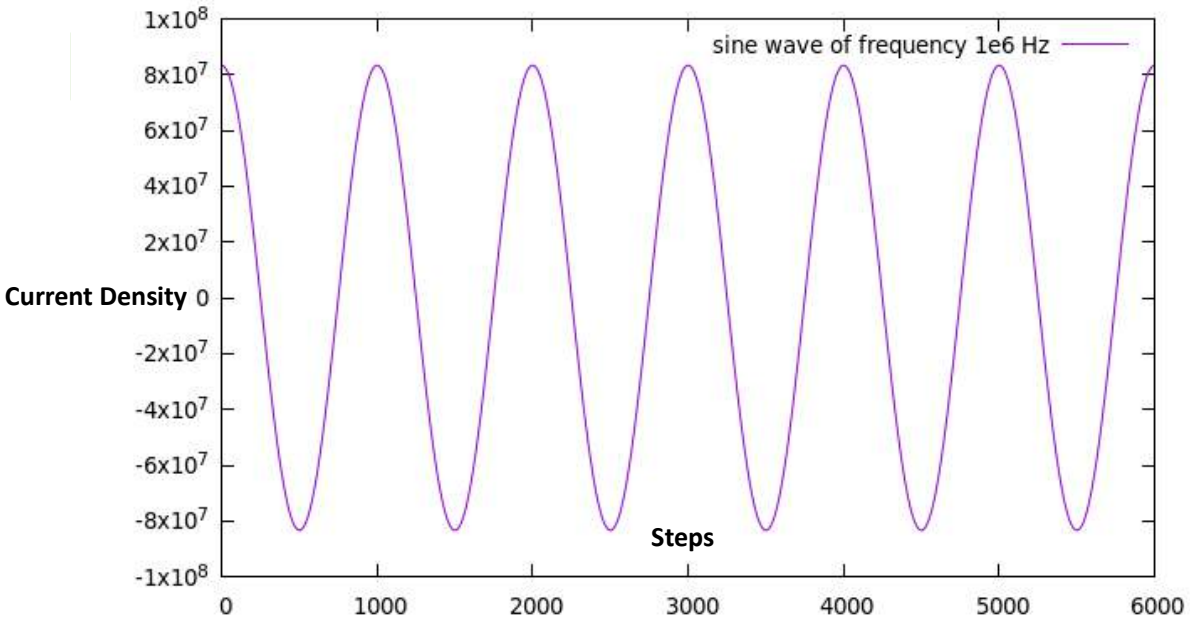


Figure 186: Sine wave of current applied to the wire

Open an empty user.dat file:

In this file, we have to enter the values for electric potential, wire dimensions and frequency of the sinusoidal wave to be applied.

```
0       10      0       0       0
10.0    12.7    0.      1e6     0.
```

Once we have the user.dat file and assuming that HIMAG is already built for this run on the machine and the executable file is available for use and added to the path. The command that is used to run HIMAG is:

```
mhd@machine$ himag -i hmg -g wire3d | tee logfile
```

This will start running HIMAG for the wire3d case and the screen output will look like Figure 187.

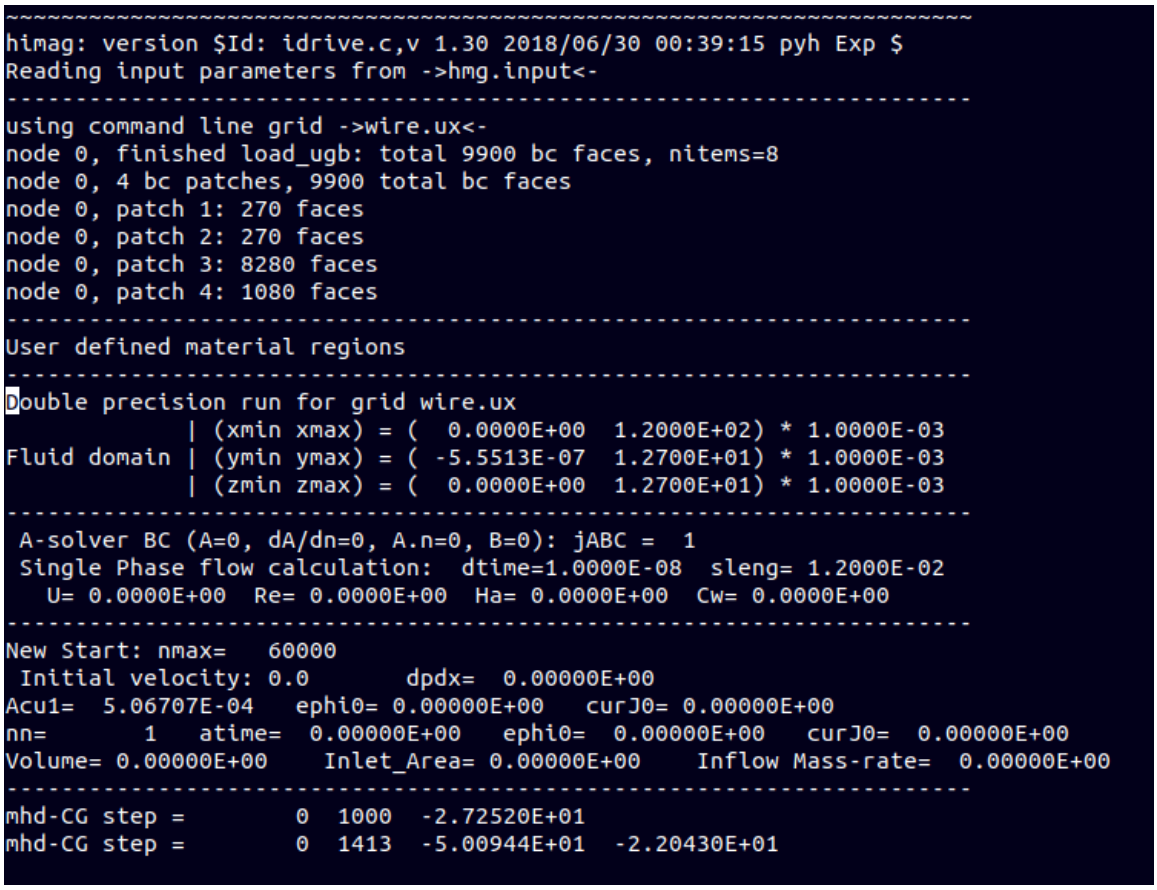


Figure 187: Screen Output on starting the case with HIMAG

Once the case is finished, we can post-process the output files *(tecfiles)*. The ***tec.cpu.step.dat*** files will be post-processed for every 1,000 steps.

```
mhd@machine$ mhd2tec wire3d.ux -B 2000
```

This step will give us a file in format, gridname.tec.step.dat. For this case, it will be ***wired3d.tec.1000.dat***. Next step is to convert .dat to .plt using Preplot, a Tecplot utility, since we will be viewing the result in Tecplot.

```
mhd@machine$ preplot wire3d.tec.2000.dat
```

Running this command will give us .plt file in the format gridname.tec.step.plt. In this case it will be ***wire3d.tec.2000.plt***. Then we will view the result file in Tecplot. To open the file in Tecplot, the command is given as:

```
mhd@machine$ tec360 wire3d.tec.2000.plt
```

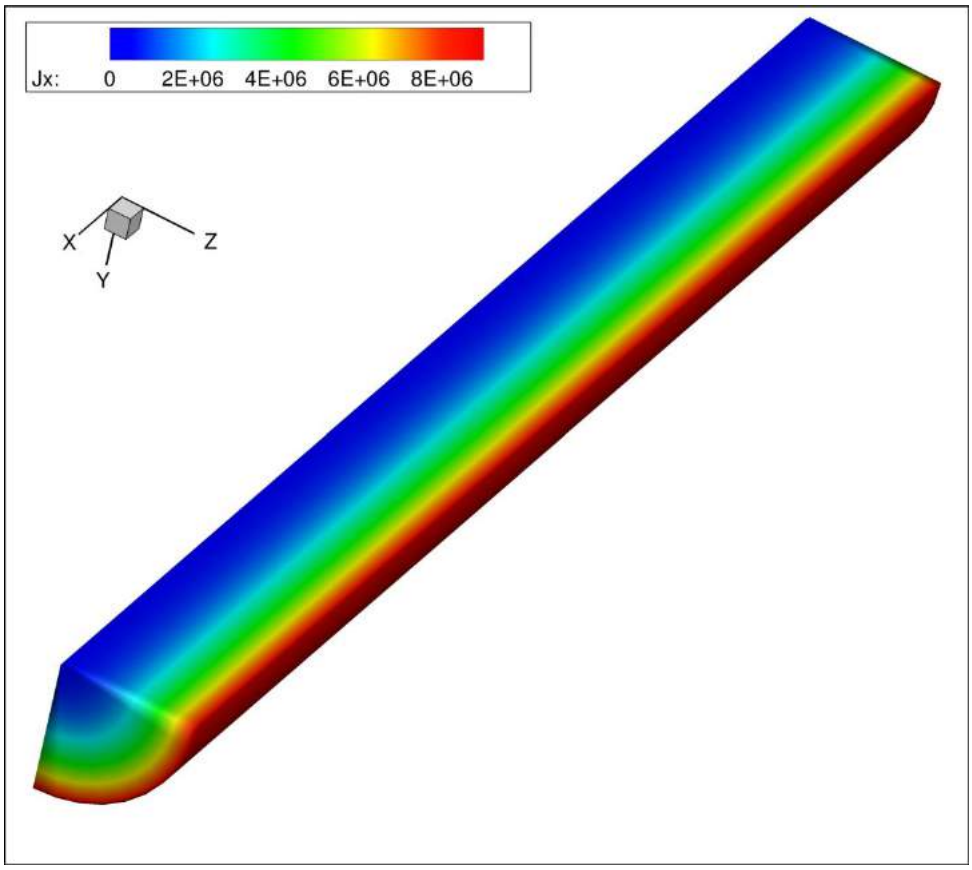


Figure 189: Wire3D case result in Tecplot

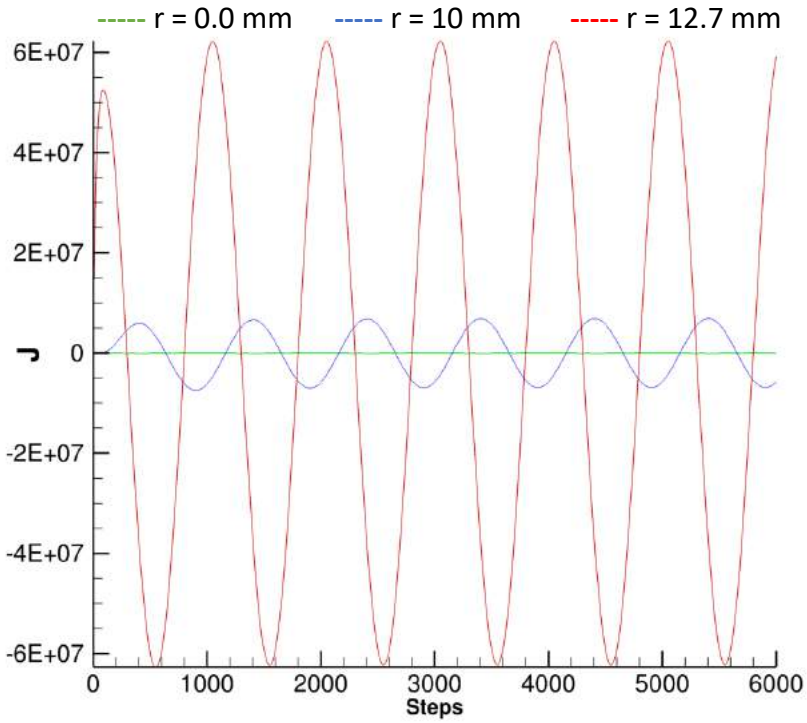


Figure 188: Observed current in the wire at r = 0mm, r = 10mm and r = 12.7mm

Index

- List of Figures
- Index of Commands
- Index of Data Files
- Nomenclature

List of Figures

Tutorial 5 – Using A-Φ Formulation

Figure 188: Observed current in the wire at r = 0mm, r = 10mm and r = 12.7mm

Index of Commands

Compiler check
dpkg --list | grep compiler

HIMAG directory location
cd ~/mhd/himag

Text Based Grid Generation Utility - XYZ
xyz

Convert xyz to .ux format
Book grid.ux

GUI Based Grid Generation Utility – HIMAG_GRID
himag_grid

Convert CGNS to UX format
ux2cgns grid.cgns

HIMAG Boundary Conditions Utility - PREP
prep

Magnetic Vector Potential Boundary Condition Utility – MOD_UGB
mod_ugb [gridname]

Partition Utility – Ux2Part
ux2part [gridname] [number of CPUs]
colormhd [gridname].ux [gridname.nn.color]

HIMAG Code Directory
cd ~/mhd/himag/code

HIMAG Library Files Directory
cd ~/mhd/himag/hmglib

HIMAG Compiling Command
make [compiler name]

Running HIMAG
./himag_executable –i [inputfile] -g [gridfile]

```
mhd@machine$ mpirun -np [no. of cores] ./himag_executable -i [inputfile] -g
             [gridname]
```

Post-Processing Result Files Utility – MHD2Tec

```
mhd2tec [gridname.ux] -p [gridname.np.color] -B step1
```

Viewing in Tecplot

```
tec360 [grid.cpu.step.dat]
preplot [grid.cpu.step.dat]
tec360 [grid.cpu.step.plt]
```

Index of Data Files

Nomenclature

HIMAG is dimensional and uses SI units for all its quantities.

a	Area (m^2)
B	Magnetic Field Strength (T, kg/Amp-s^2)
c_p	Specific Heat Capacity (S/m)
g	gravitational acceleration (m/s^2), standard value = 9.80665 m/s^2
Gr	Grashof number (dimensionless)
h	Heat transfer coefficient (W/m^2)
Ha	Hartmann number (dimensionless)
I	Current (Amp)
J	Current Density (Amp/m^2)
k	Thermal Conductivity (W/m-K)
L,l	Length scale (m)
m	Mass flow rate (kg/s)
p	Pressure (Pa, kg/m-s^2)
q	Heat flux (W/m^2)
r	Radius (m)
Re	Reynolds number (dimensionless)
T	Temperature (K)
t	Time (s)
u, v, w	Velocity (m/s)
V	Volume (m^3)
β	Coefficient of thermal expansion (K^{-1})
Δ	Change in variable, final – initial (for example, Δp, Δt)
ρ	Density (kg/m^3)
σ	Electrical Conductivity (S/m)
σ_s	Surface Tension (kg/m)
μ	Dynamic Viscosity (kg/m-s)

Bibliography

[1] ANSYS Fluent User's Guide, Software Release Version 15.0, 2013.

[2] FIELDVIEW Reference Manual, Software Release Version 10, Intelligent Light, 2004.

[3] ANSYS ICEM CFD User Manual, Release 14.5, 2012.

[4] NASA FUN3D Online Guide, Release 13.4, 2018.

[5] Tecplot 360 EX User's Manual, Release 2, 2018.

[6] MSC Nastran 2014 Release Guide, Version 2014.1, 2014.

[7] HIMAG Internal Report on High Hartmann Number Cases, 2014.

[8] *"A comprehensive high performance predictive tool for fusion liquid material hydromagnetics"*, Technical Report, DOE-SBIR (Phase II), 2017.