# Modeling Plasmas with Strong Anisotropy, Neutral Fluid Effects, and Open Boundaries

Eric T. Meier

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2011

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

**Abstract**

Modeling Plasmas with Strong Anisotropy, Neutral Fluid Effects, and Open Boundaries

Eric T. Meier

Chair of the Supervisory Committee:
Professor Uri Shumlak
Aeronautics & Astronautics

Three computational plasma science topics are addressed in this research: the challenge of modeling strongly anisotropic thermal conduction, capturing neutral fluid effects in collisional plasmas, and modeling open boundaries in dissipative plasmas. The research efforts on these three topics contribute to a common objective: the improvement and extension of existing magnetohydrodynamic modeling capability. Modeling magnetically confined fusion-related plasmas is the focus of the research, but broader relevance is recognized and discussed. Code development is central to this work, and has been carried out within the flexible physics framework of the highly parallel HiFi implicit spectral element code.

In magnetic plasma confinement, heat conduction perpendicular to the magnetic field is extremely slow compared to conduction parallel to the field. The anisotropy in heat conduction can be many orders of magnitude, and the inaccuracy of low-order representations can allow parallel heat transport to "leak" into the perpendicular direction, resulting in numerical perpendicular transport. If the computational grid is aligned to the magnetic field, this numerical error can be eliminated, even for low-order representations. However, grid alignment is possible only in idealized problems. In realistic applications, magnetic topology is chaotic. A general approach for accurately modeling the extreme anisotropy of fusion plasmas is to use high-order representations which do not require grid alignment for sufficient resolution. This research provides a comprehensive assessment of spectral el-

ement representation of anisotropy, in terms of dependence of accuracy on grid alignment, polynomial degree, and grid cell size, and gives results for two- and three-dimensional cases.

Truncating large physical domains to concentrate computational resources is often necessary or desirable in simulating natural and man-made plasmas. A novel open boundary condition (BC) treatment for such domain truncation, lacuna-based open boundary conditions (LOBC), is presented. LOBC provide effective open BC for dissipative MHD and other hyperbolic and mixed hyperbolic-parabolic systems of partial differential equations. Based on manipulating Calderon-type near-boundary sources, LOBC damp hyperbolic effects in an exterior region attached to the simulation domain, and apply BC appropriate for the remaining parabolic effects (if present) at the exterior region boundary. LOBC and several alternative open BC are tested in gas dynamics and dissipative MHD problems, and their performance is compared. LOBC are found to give stable, low-reflection solutions even in the presence of strong parabolic behavior, while alternative open BC are either highly reflective or unstable.

Only a few specialized computational tools are available for capturing the effects of neutral particles in plasmas. The goal of this research has been to develop and apply a generalized, computationally tractable model based on first principles that serves as a first step toward more sophisticated models. This dissertation presents the derivation of a plasma-neutral fluid model from the Boltzmann equation, allowing for charge exchange, ionization, and recombination. Single-species, singly-ionized plasma and its parent neutral atoms are modeled. Mass, momentum, and energy exchange between the plasma and neutral species are tracked in a numerically stable, conservative implementation. The implementation has been applied to parallel-plate and coaxial plasma acceleration, ion spin-up in field-reversed configuration (FRC) plasmas with rotating magnetic field (RMF) current drive, and the interaction of FRC plasmas with neutral gas in the Electrodeless Lorentz Force (ELF) thruster. ELF simulations are compared with preliminary experimental results.

# Appendix A
# USING HIFI

This appendix presents some practical details for using HiFi [1, 2, 3]. The steps necessary to acquire 2D HiFi (SEL) are given in Section A.1. Running a simulation with pn.f, the physics module containing specifications for the plasma-neutral model presented in Chapter 5.3, is described in Section A.2. Finally, post-processing, and visualizing the simulation results is explained in Section A.3. This appendix is intended to serve as a rough guide for new users of the 2D HiFi code, although it may provide some useful insight for non-users.

HiFi is written in Fortran 90/95, and users should be familiar with modern Fortran programming to use the code. An especially useful Fortran reference is the book by Redwine [121].

## *A.1 Acquiring HiFi*

### *A.1.1 User agreement*

The user agreement for HiFi is given in Figure A.1. The principal developers of HiFi are Dr. Vyacheslav (Slava) Lukin and Dr. Alan Glasser. By sending the signed agreement to Dr. Lukin at vlukin1(at)mailaps.org, a user name and password can be obtained for the online code repository. Note that this dissertation has focused on the 2D version of HiFi. 3D HiFi is also available from the same developers.

### *A.1.2 Code versions and repository structure*

HiFi is stored in a Subversion [122] repository. At the time of this writing, the current repository version is 378.

The 2D HiFi (SEL) repository includes the "trunk" code (SEL/trunk)and "branches" (SEL/branches). In the research presented in this dissertation, two different branches are used: SEL/branches/obc and SEL/branches/neutrals. Within the folder SEL/trunk, mul-

## HiFi (SEL) User Agreement Form

HiFi (SEL) is an open source code development project for solving systems of coupled non-linear PDEs on (semi-)structured logically hexahedral (rectangular) grids that abides by the following BSD-style license. The project has been supported, in part, by the U.S. Department of Energy. Before having access to the code, you must agree to the conditions of the license that serve as additional protections for the HiFi (SEL) code.

## Terms of Agreement

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3) Neither the name of the project nor the names of its developers may be used to endorse, promote, or publish products derived from this software without specific prior written permission by one of the principle developers.
4) Publications or figures made using results of the HiFi (SEL) code calculations will acknowledge the HiFi (SEL) code.
5) It is understood that the HiFi (SEL) code is still under development and thus may not contain all features that users may need/want for their problem of interest.
6) It is understood that the HiFi (SEL) project does not guarantee that support will always be available to users of the code. In addition, it is understood that extensive support from a HiFi (SEL) team member on a particular application generally implies that any publication derived from the application will include that team's member(s) as a co-author.

THIS SOFTWARE IS PROVIDED BY *HiFi team* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL *neither any HiFi team member, nor the University of Washington, nor the United States Government, nor any agency thereof, nor any of their employees* BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

If you agree to the above terms and would like to download and/or use the HiFi (SEL) code, please fill out this form and e-mail it to Vyacheslav Lukin at **vlukin1@mailaps.org**.

## HiFi (SEL) Agreement form

Name: _____

Organization: _____

Mailing Address: _____

Mailing Address: _____

Email Address: _____

Requested username for Web access: _____

Any comments or special requests:

**I agree to the above Terms of Agreement and certify that the information submitted in the form is true.**

NAME: _____

DATE: _____

Figure A.1: HiFi user agreement.

tiple versions of SEL may be present, e.g., "code_3.0.0", and "code_3.1". The latest version is "code_3.1", and it is this version that is the basis for the code used for this dissertation which is present in SEL/branches/open_bc/code_3.1 and SEL/branches/neutrals/code_3.1.

As discussed in Section 1.3, the core solver routines are separate from the "physics" module where PDEs to be solved are specified. As an example of how to use HiFi, the focus of this appendix is on the file pn.f. This file is present in the directory SEL/branches/neutrals/code_3.1. The core solver code is in the directory SEL/branches/neutrals/code_3.1/solver.

In SEL/trunk, the folder "post" contains the postprocessing code which will be discussed below.

In SEL/trunk, there is a file called README which has some details about the code organization and compiling the code. Further details will be available upon receiving access to the code.

## *A.2 Running a plasma-neutral simulation*

In this section, relevant details for running a simulation with the HiFi physics module pn.f are given, including: input deck (Section A.2.1); normalizations (Section A.2.2); grid (Section A.2.3); variables and equations (Section A.2.4); equilibrium (Section A.2.5); boundary conditions (Section A.2.6); and interpreting runtime output (Section A.2.7).

Several subroutines in pn.f will be referred to by name. Each subroutine in pn.f has some comments about its function. Also, see the file the physics module template file, physics_templ.f, in SEL/trunk/code_3.1 for additional comments on individual physics module subroutines.

### *A.2.1 Input deck*

The following commented input deck is used to conduct the baseline ELF simulation described in Section 6.4. Note that all of the options are not listed and commented. Default values are used for those not listed. See the file README discussed in Section A.1.2 for an input deck with all options commented for `algorithm_input` and `universal_input`.

```
&algorithm_input
solve_type="condense"   ! use static condensation
step_type="theta"       ! theta method for time advance
theta=.5                ! theta=.5 --> Crank-Nicolson

adapt_dt=t              ! use adaptive time step
errtol=1.e-4            ! tolerance for nonlinear solve
ksp_restart=30
always_pc_reset=t

itmax=140               ! maximum number of newton iterations
itmax_incr=3            ! increase time step if fewer iterations
itmax_decr=7            ! decrease time step if more iterations
dt_incr=1.2             ! factor of time step increase
dt_decr=.6              ! factor of time step decrease

nodal=f                 ! use modal basis

quad_type="gl0"
grid_type="sel"
grid_inv_type="jacobi"

adapt_grid=f

monitor=t
fd_test=f
fd_bound_test=f
du_diagnose=f

outfile_type="hdf5"
parallel_write=t
parallel_read=f
/
```

```
&universal_input
dmout=10                ! write out solution every 10 steps
outdir="results/elf_BL" ! output directory

restart_flag=t          ! restart from earlier run
restart_dir="results/elf_BL_eq"
                        ! restart directory
restart_step=100        ! restart step
/

&pn_list
nx=72                   ! 72 cells in the axial direction
ny=8                    ! 8 cells in the radial direction
nbx=12                  ! split grid into 72/12=6 axial blocks;
                               ! the number of processors used must be divisible
                               ! by nbx.
np=8                    ! polynomial degree 8
nq=8                    ! 8 quadrature points (in a rectangular
                        ! grid, np=nq gives exact integration)
xperiodic=t             ! periodic in axial direction
yperiodic=f             ! not periodic in radial direction

dt=5.e-4                ! initial time step size
dtmax=5.e-3             ! maximum time step size
tmax=5.                 ! maximum time
nstep=10000             ! maximum number of time steps

init_type="trans_test"  ! run a "trans_type" simulation
cylinder=t              ! cylindrical coordinates

equilfile="frc_long.dat"! the equilibrium file name
```

```
xmin=-5.32              ! minimum axial position
lx=15.96                ! total axial extent
ly=1.                   ! total radial extent
                        ! in physical units (after multiplying by L0),
                        ! the domain extends axially from  -0.625 meters
                        ! to +1.625 meters.
                        ! the FRC is initially centered at x=0.

L0=.141                 ! length normalization
n0=7.e19                ! density normalization
b0=.012                 ! magnetic field normalization

atom="neon"             ! neon neutral gas and plasma

ddiff=2.e-3             ! density diffusion

eta_case="spitzer-chodura"
                        ! spitzer-chodura resistivity
etavac=2.               ! maximum resistivity

visc_case="braginskii"  ! braginskii (isotropic) plasma viscosity
mu_min=5.e-3            ! minimum viscosity
mu_sv=1.e-3             ! artificial viscosity

viscn_case="hard_sphere"! hard sphere neutral viscosity

kappa_case="braginskii" ! braginskii plasma thermal conduction
kappa_min=1.e-2         ! minimum plasma thermal conduction

kappan_case="hard_sphere"
                        ! hard sphere neutral thermal conduction

initv=3.7               ! initial axial speed
```

```
pmin=5.e-5              ! minimum initial pressure
rhomin=.005             ! minimum initial density

initrhon=10.            ! initial peak neutral density
initTn=.0025            ! initial neutral temperature

targ_type="gauss"       ! gaussian neutral gas profile

ion_fac=1.              ! ionization factor (1 --> on; 0 --> off)
recomb_fac=1.           ! recombination factor (1 --> on; 0 --> off)
cx_fac=1.               ! charge exchange factor (1 --> on; 0 --> off)
civ_fac=0.              ! CIV factor (>1 --> on; 0 --> off)

te_frac=.5              ! fraction of plasma pressure in electron species
/
```

### *A.2.2 Normalizations*

To avoid issues related to computer round-off error, working in normalized units is recommended. PDEs implemented in HiFi are typically normalized. For the plasma-neutral module, pn.f, normalizations are given in Table A.1.

By multiplying the normalized quantity by the normalization, the value in physical units is found. For example, to determine the density in SI units, the normalized density ($\tilde{\rho}$) is multiplied by the density normalization constant: $\rho = \tilde{\rho}\rho_0$. As shown in Table A.1, $\rho_0 = n_0 m_i$. A spreadsheet is recommended to facilitate conversions and, effectively, comprehension of code input/output.

As an example of normalization, consider the plasma continuity equation,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v} - D_\rho \nabla \rho) = m_i(\Gamma_i^{ion} - \Gamma_n^{rec}),$$

which is Eqn. (5.70) of Section 5.3. The normalized equation is

Table A.1: Normalizations for plasma-neutral simulation. The quantities used as the normalization basis are density, magnetic field, and length ($n_0, B_0$, and $L_0$) in SI units. Normalizations are given in SI units except for temperature which is given in electron volts. The quantities $q_e$, $\mu_0$, and $k_B$ are the elementary charge, permeability of free space, and Boltzmann constant, respectively: $q_e = 1.601 \times 10^{-19}$ C, $\mu_0 = 4\pi \times 10^{-7}$ T m/A, and $k_B = 1.381 \times 10^{-23}$ J/K.

| normalization | definition | units |
|---|---|---|
| $\rho_0$ | $n_0 m_i$ | kg / m$^3$ |
| $p_0$ | $B_0^2/\mu_0$ | Pa |
| $T_0$ | $B_0^2/(\mu_0 q_e n_0)$ | eV |
| $A_0$ | $L_0 B_0$ | T m |
| $v_0$ | $B_0/\sqrt{\mu_0 n_0 m_i}$ | m/s |
| $j_0$ | $B_0/(L_0 \mu_0)$ | A/m$^2$ |
| $t_0$ | $L_0\sqrt{\mu_0 n_0 m_i}/B_0$ | s |
| $D_{\rho,0}$ | $L_0^2/t_0$ | m$^2$/s |
| $\eta_0$ | $\mu_0 L_0^2/t_0$ | $\Omega$ m |
| $\xi_0$ | $\rho_0 L_0^2/t_0$ | Pa s |
| $\kappa_0$ | $n_0 k_B L_0^2/t_0$ | W/(m K) |

$$\frac{\partial\tilde{\rho}}{\partial\tilde{t}} + \tilde{\nabla}\cdot(\tilde{\rho}\tilde{\mathbf{v}} - \tilde{D}_\rho\tilde{\nabla}\tilde{\rho}) = \tilde{\Gamma}_i^{ion} - \tilde{\Gamma}_n^{rec}, \tag{A.1}$$

where the normalized quantities are accented with tildes. The atomic physics source rates are normalized such that, for example, $\tilde{\Gamma}_i^{ion}\Gamma_0 = \Gamma_i^{ion}m_i$, where $\Gamma_0 = \rho_0/t_0$. Source rates have units of kg/(m$^3$s). In pn.f, the variables `recomb_norm`, `ion_norm`, and `cx_norm` are defined such that the normalized quantities are computed as simple functions of normalized variables. For ionization,

$$\begin{aligned}
\tilde{\Gamma}_i^{ion} &= \Gamma_i^{ion}m_i/\Gamma_0 \\
&= A\times 10^{-6}\frac{1+P*(\phi_{ion}/T_e)^{1/2}}{X+\phi_{ion}/T_e}\left(\frac{\phi_{ion}}{T_e}\right)^K e^{-\phi_{ion}/T_e} n n_n \frac{m_i}{\Gamma_0} \\
&= A\times 10^{-6}\frac{1+P*(\phi_{ion}/T_e)^{1/2}}{X+\phi_{ion}/T_e}\left(\frac{\phi_{ion}}{T_e}\right)^K e^{-\phi_{ion}/T_e} \tilde{\rho}\tilde{\rho}_n n_0 t_0 \\
&= \texttt{ion_norm}\frac{1+P*(\phi_{ion}/T_e)^{1/2}}{X+\phi_{ion}/T_e}\left(\frac{\phi_{ion}}{T_e}\right)^K e^{-\phi_{ion}/T_e} \tilde{\rho}\tilde{\rho}_n.
\end{aligned} \tag{A.2}$$

The formula used for $\Gamma_i^{ion}$ involves the constants $A$, $P$, $X$, and $K$, as discussed in Appendix D. Notice that the quantity `ion_norm` $= A\times 10^{-6}n_0t_0$ is an accumulation of constants so that $\tilde{\Gamma}_i^{ion}$ is a simple function of the dimensionless quantity $\phi_{ion}/T_e$, and the normalized variables $\rho$ and $\rho_n$. Constants are similarly accumulated for viscosities, thermal conductivities, and resistivity.

### *A.2.3 Grid*

In the subroutine physics_grid, the input arguments are `ksi` and `etag`, the logical coordinates of the quadrature points within a given cell. The output arguments are `x` and `y`, the (normalized) physical coordinates for the axial and radial directions, respectively, within the cell. The arrays of each of these arguments have two indices; the first is the axial index and the second is the radial index. For example, in a square cell, `x(1,1)`=`x(1,2)`=`x(1,3)`, etc., and `y(1,1)`=`y(2,1)`=`y(3,1)`, etc.

Simulations in cylindrical or cartesian coordinates can be run with the pn.f module. The comments in this appendix address simulations in cylindrical coordinates, where the $r - z$ plane is discretized. In pn.f, "x" corresponds to the axial direction, and "y" to the radial direction.

The physics_grid subroutine is called cell-by-cell, as are all of the other pn.f subroutines discussed in this appendix. So, for example, locally computing the maximum value of x will not necessarily give the maximum value of the global logical space (which is one).

### *A.2.4 Variables and equations*

In all 2D HiFi (SEL) physics modules, the subroutine "physics_rhs" is where flux and source terms are specified for the interior equations. The module pn.f has 10 variables. As indicated in the comments there, the variables are

1. Plasma density, $\rho$
2. Negative phi-direction magnetic vector potential, $-A_\phi$
3. Plasma pressure, $p$
4. Plasma axial momentum, $\rho v_z$
5. Plasma radial momentum, $\rho v_r$
6. Out-of-plane current density, $j_\phi$
7. Neutral density, $\rho_n$
8. Neutral axial momentum, $\rho_n v_{z,n}$
9. Neutral radial momentum, $\rho_n v_{r,n}$
10. Neutral pressure, $p_n$

In this list and in pn.f, normalized variables are implied and the tilde accents are dropped.

The variable $j_\phi$ is an auxiliary variable in the sense that it is not a primary variable evolved in the equations presented in Section 5.3. Fluxes and sources can be functions of only the variables and their first spatial derivatives. To compute the source term $\eta j_\phi^2$ in the pressure evolution equation (see Section 5.3), the presence $j_\phi$ as an auxiliary variable is necessary.

The first index of the variables is the variable number as given in the list above. The second and third indices are the axial and radial positions of the quadrature points, corresponding to the first and second indices in the grid variables (see Section A.2.3).

Jacobians (i.e., derivatives with respect to the variables) of the fluxes and sources are required, and are computed in the subroutine physics_rhs_drdu.

The mass matrix for the interior equations is set in the subroutine physics_mass. The default values of the mass matrix are set to one for the diagonal entries (i.e., `mass(1,1,:,:)`, `mass(2,2,:,:)`, etc.). For most of the variables in pn.f, the mass matrix diagonal entries are changed to `r_fac`, which is a factor equal to the radial distance at each quadrature point. This factor is used to cancel the $1/r$ that occurs in the divergence in cylindrical coordinates.

### *A.2.5 Equilibrium*

In the subroutine physics_init, the solution is initialized. Input arguments are `x` and `y`, which are arrays of axial and radial quadrature point physical coordinates, respectively. The output argument is `u`, which has three indices: the first refers to variable number, the second to the axial position, and the third to the radial position.

The case "trans_test" first calls pn_equil, which reads in an FRC equilibrium from the file specified in sel.in with the variable `equilfile`. Next, the initial axial speed is set, and the neutral density profile is initialized as described in Section 6.4.

### *A.2.6 Boundary conditions*

The subroutine physics_boundary sets the boundary condition (BC) types for each equation. The subroutine has arguments `left`, `right`, `top`, and `bottom`, which are `edge_type` variables corresponding, respectively, to the boundaries at the minimum axial position, and maximum axial position, radial wall, and the cylindrical axis.

Within the `edge_type` derived type, there is a character variable `bc_type`, and a logical variable `static`. The variables `bc_type` and `static` are arrays with entries for each variable (10 variables in the case of pn.f). As described in Section 1.3.3, BC are either flux BC or explicit local BC. For the "trans_test" case, two flux BC types are used: "normflux" and "zeroflux". As the names imply, "normflux" allows the normal flux at the boundary to be specified and "zeroflux" sets the normal flux to zero. The explicit local BC "robin" is also used, requiring the solution to satisfy a specified boundary equation. See the boundary conditions discussion in Section 6.4.1. The variable static is set to true for all variables except for the second variable, $-A_\phi$, at the radial wall. Setting static to false indicates that a time-dependent term may be used in the robin BC. See discussion of physics_edge_mass below. Time-variation of $-A_\phi$ represents a voltage applied at the wall as discussed in Section 6.4.1.

In the subroutine physics_edge_rhs, the output argument `c` is set either to the flux for "normflux" BCs, or to the right-hand side for "robin" BCs, where the left-hand side is zero if the variable `static` is true, and is a time varying term if `static` is false. The Jacobian of these boundary equations (i.e., the derivative of `c` with respect to each variables) is set in the subroutine physics_edge_drdu.

The subroutine physics_edge_mass sets the mass matrix for the time-varying terms of the "robin" boundary equations for which the variable `static` is false. As mentioned above, for the variable $-A_\phi$, the mass matrix entry is set to one.

### *A.2.7 Interpreting runtime output*

At runtime, an output file called sel.out is generated. An excerpt of SEL.out for the ELF baseline run for which the input deck is shown in Section A.2.7 follows.

```
------------------------------------------------------------------------
 physics = pn, step_type = theta, nproc =     48, nbx =   12
 xper = T, yper = F, nx =   72, ny =    8, np = 8, nq = 8
 solve_type = condense

  iout   m  it  jac  ksp     t        dt     wclock   griderr   condno

    0  100   0   0     0 5.000E-02 5.00E-04 4.25E+00 6.73E-03 0.000E+00
    1  110  67   6    67 5.300E-02 3.00E-04 1.03E+02 6.28E-03 1.000E+00
    2  120  38   4    38 5.663E-02 5.18E-04 6.75E+01 1.13E-02 1.000E+00
    3  130  40   5    40 6.308E-02 1.07E-03 8.21E+01 3.43E-02 1.000E+00
    4  140  44   3    44 7.830E-02 1.55E-03 5.39E+01 6.08E-02 1.000E+00
    5  150  53   3    53 9.595E-02 1.86E-03 5.57E+01 3.60E-02 1.000E+00
    6  160  53   2    53 1.145E-01 1.86E-03 4.11E+01 4.48E-02 1.000E+00
    7  170  55   2    55 1.331E-01 1.86E-03 4.17E+01 7.22E-02 1.000E+00
    8  180  55   2    55 1.517E-01 1.86E-03 4.15E+01 5.93E-02 1.000E+00
    9  190  56   3    56 1.702E-01 1.86E-03 5.85E+01 3.15E-02 1.000E+00
   10  200  60   2    60 1.888E-01 1.86E-03 4.24E+01 4.29E-02 1.000E+00
```

The header of sel.out reflects some of the input deck options (see Section A.2.1). Also, the number of processors used for the run is shown as *nproc*. In this case, 48 processors are used. Each row of data is written when the code writes output files containing the solution data in terms of basis function amplitudes in each cell. The column labeled *iout* indicates the output number; $m$ indicates the step number; $it$, the number of nonlinear iterations required; $jac$, the number of Jacobians required during the nonlinear solve; $ksp$, the number of Krylov space solver iterations required (in this case, a direct LU solver is used and $ksp{=}it$); $t$, the time; $dt$, the time step size; *wclock*, the wall clock time required since the last data output; *griderr*, the grid error;[1] and *condno*, the condition number for the iterative linear solver (it is one in this case because a direct LU solver is used).

[1]Grid error is a measure of how well-converged the spectral representation is. *griderr* reports the worst grid error for all variables in all cells. More specifically, *griderr* is a comparison of the amplitude of the highest-order basis function to the overall magnitude of the variable.

### *A.3 Post-processing and visualizing HiFi output*

Post-processing is done using the code called "post" in the repository folder SEL/trunk/post. Post-processing is done serially. When the post code is executed, it retrieves the post.in input deck. The following post.in file can be used to produce output appropriate for viewing the ELF baseline run with the VisIt [123] visualization tool. (See also the file post.in in SEL/trunk/code_3.1.)

```
&post_input

indir="results/elf_BL"  ! directory containing output files
postout="visit"         ! directory that output should be written to
out_type="hdf5"         ! output type; hdf5 output can be read by VisIt

job_type="." ! setting job_type to "." gives the default output
                        ! specialized post-processing is possible but generally
                        ! requires customized code.

nxw0=10                 ! the number of interpolary points
nyw0=10

drawgrid=f
polar_crd=f

mfile=1000
stride=10

contour=t
/

&post_input
xt_flag=f               ! this flag can be set to true to generate 1D plots
                                ! in the "x" direction at various times at fixed
                                ! "y" position which are viewable with the xdraw
```

```
                                    ! software (consult HiFi developers for more information)
yt_flag=f                ! see comment for xt_flag
/
```

After post-processing using the above input deck, .hdf5 files will be present in the directory "visit" along with .xmf files which contain the information necessary for VisIt to access the data in the .hdf5 files.[2] By opening the the family of .xmf files with VisIt, a variety options will be available, including contour plots, pseudocolor plots, etc. The variables are numbered as in the list given in Section A.2.4 — for example, the first variable, density, is read into VisIt as "U01". The second variable is represented by U02, etc.

[2]HiFi developers currently recommend VisIt version 2.2.0.

# VITA

Eric Meier was born in Shreveport, Louisiana, and spent much of his childhood in Nacogdoches, Texas. At age 13, he moved with his family to Salt Lake City, Utah. In high school, he enjoyed mountain recreation in Utah, played many sports including wrestling and tennis, and developed a strong interest in physics and mathematics. He earned a BS in mechanical engineering at the University of Utah in 2000. His first job after graduating was with an aerospace company presently called Aerojet in Redmond, Washington. After two years at Aerojet, he and a fellow engineer formed Space Transport Corporation (STC) with the primary goal of capturing the Ansari X Prize. Though STC fell short of this goal, and was ultimately dissolved, the experience instilled in Eric a passion for scientific research. In 2005, he joined the Department of Aeronautics and Astronautics at the University of Washington, where he earned a MS and PhD specializing in computational plasma science.