The M3D-C^1 Users Guide

Last Update on 03/23/2022

Jin Chen, Nate Ferraro, Stephen Jardin @ PPPL
Brendan Lyons @GA
Seegyoung Seol, Usman Riaz, Mark Shephard @ RPI

□

Table of Contents

1. Downloading and Compiling

1.1 Accessing PPPL Machines

Once you obtain a PPPL Unix user account, please visit the website: http://researchcomputing.pppl.gov

There, you will find instructions for installing NX virtual Desktop.

In PPPL machine, first get an interactive session for a single processor on the "portal" computer. The M3DC1 code is located in the Github repository: PrincetonUniversity/M3DC1. Access to this repository requires a Github account and permission from nferraro@pppl.gov. Note that Section 1.5 describes how to use pre-compiled versions of M3D-C1 so you do not need github access.

1.2 GITHUB

Retrieve the current version of M3D-*C*[1] from the GIT repository. For the first time, to check out the sources do:

Initial access is with the "clone" command. This copies the source code from the master file into a working directory on your machine. You only do this once on each computer you work on.
module load git
git clone https://github.com/PrincetonUniversity/M3DC1

Subsequent GIT commands used to commit:

add/commit/push you ***add*** files to a list of files to update, ***commit*** the changes to your branch, and then ***push*** the changes to the master branch
git commit –m "message describing changes" (adding –a commits all changes)

diff lists the changes you made from the last commit, even if you haven't pushed your commits to github. To see how your files differ from what's on github, you can do:
git fetch origin master
git diff origin/master

status compares your branch with the "master branch"

pull updates your local branch to the current master branch
may need to origin master

stash takes uncommitted changes, saves them for later use, and reverts files in working directory
stash list ***stash drop*** ***stash apply*** ***stash pop (apply+drop)***

stash pop removes changes from your stash and reapplies them to working copy

stash apply keeps changes in stash, but reapplies them to working copy

reset –hard discards any changes to local branch since last *commit*

branch tells you what branch you are in

l***og*** (--oneline) (--after 2017-12-31) lists all the commits for the checked-out branch after that date

log –pretty=format:” %h - %an, %ar : %s”

checkout b8d17c0 switches to commit branch b8d17c0

1.2.1 Branches in git

To make a new branch called fp_phase2

```
====================================
> git checkout master                # switch to the master branch
> git pull                           # make sure the master branch is up-to-date
> git checkout -b fp_phase2          # The "-b" create a new branch
                                     #   this will be identical to master to start
> git push --set-upstream origin fp_phase2 # push this new branch to the remote repo so others can access it
```

Committing changes (e.g., newpar.f90)

```
====================================
> git pull                    # Always do this before you start committing
                              #   if you forget, you risk diverging your local branch from remote
> git add newpar.f90          # This stages the current changes in newpar.f90 for commit
                              #   you could then make more changes before committing,
                              #   but you'd have to add again to get those into the commit
> git commit -m "Changed newpar.f90" # Commit changes to your local branch
> git push                    # Push commits to the remote repo
                              #   --set-upstream only needs to be done the first time
```

Merge changes on master into fp_phase2 (e.g., some important bug fix)

```
====================================================================
> git checkout master    # switch to the master branch
> git pull           # get the latest commits on the master branch
> git checkout fp_phase2 # switch back to fp_phase2 (no need for -b now)
> git pull           # get latest commits on fp_phase2 to avoid conflicts
> git merge master      # merge any new commits into fp_phase2
                        #  - this makes a commit
                        #  - you may need to resolve conflicts
> git push           # push the merge commit to remote repo
```

Inverting fp_phase2 and master here would merge the development branch into master locally, then the push would send the merge to the remote master.

1.3 On-line Documentation

An extensive document that describes equations solved by the code and the input variables in C1input is in the GIT repo. Please refer to:
…./trunk/unstructured/doc/doc.pdf
Machine-specific instructions for portal, perseus, Edison, Cori are in: …/trunk/unstructured/README

Additional documentation is available on the site: w3.pppl.gov/~nferraro under the "M3D-C1" tab.

The latest copy of his document is available at: http://m3dc1.pppl.gov

1.4 Compilation

1.4.1 Load modules

PPPL RHEL6 machines (portal.pppl.gov)

[openmpi-1.8.4]

Load the following modules and copy sunfire.openmpi-1.8.4.mk to sunfire.pppl.gov.mk

openmpi/1.8.4	intel/2015.u1	gsl/1.16	szip/2.1	hdf5-parallel
hdf/4.2r1	scalapack	fftw		

See README/readme.portalr.openmp-1.8.4 for detailed instructions and an example job script

[openmpi-1.10.3]

Load the following modules and copy sunfire.openmpi-1.10.3.mk to sunfire.pppl.gov.mk

openmpi/1.10.3	intel/2015.u1	gsl	szip	hdf5-parallel/1.8.17

See README/readme.portal.openmp-1.10.3 for detailed instructions and an example job script.

[openmpi-4.0.1]

Load the following modules and copy sunfire.openmpi-4.0.1.mk to sunfire.pppl.gov.mk

openmpi/4.0.1	intel/2019.u3	gsl	szip	scalapack

See README/readme.portal.openmp-4.0.1 for detailed instructions and an example job script

[openmpi-4.0.3]

Load the following modules and copy sunfire.openmpi-4.0.3.mk to sunfire.pppl.gov.mk

openmpi/4.0.3	intel/2019.u3	hdf5-parallel/1.10.5
fftw/3.3.8	superlu/5.2.1	

See README/readme.portal.openmp-4.0.3 for detailed instructions and an example job script

PPPL RHEL7 machines (portalc7.pppl.gov)

Load the following modules and copy centos7.mk to sunfire.pppl.gov.mk

openmpi/4.0.3 intel/2019.u3 hdf5-parallel/1.10.5 fftw

See README/readme.centos7 for detailed instructions and an example job script.

cori.nersc.gov

See README/readme.cori and README/readme.corigpu for detailed instructions and an example job script.

perseus.princeton.edu

Follow the instructions in 5.1.1 in first setting up idl

Load the following modules:

intel/18.0/64/18.0.3.222 intel-mpi/intel/2018.3/64 gsl/2.4

hdf5/intel-17.0/intel-mpi/1.10.0 fftw/intel-16.0/intel-mpi/3.3.4

See README/readme.perseus for detailed instructions and an example job script.

For help regarding perseus, send email to: cses@princeton.edu

perseus-amd.princeton.edu

Load the following modules:

intel/18.0/64/18.0.3.222 intel-mpi/intel/2018.3/64 gsl

hdf5/intel-17.0/intel-mpi/1.10.0 fftw/intel-16.0/intel-mpi/3.3.4

See README/readme.perseusamd for detailed instructions and an example job script.

For help regarding perseus, send email to: cses@princeton.edu

stellar.princeton.edu

See README/readme.stellar for detailed instructions and an example job script.

For help regarding stellar, send email to: cses@princeton.edu

traverse.princeton.edu

Load the following modules:

pgi/19.9/64 openmpi/pgi-19.9/4.0.3rc1/64 hdf5/pgi-19.5/openmpi-4.0.2rc1/1.10.5

fftw/gcc/ openmpi-4.0.1/3.3.8 cudatoolkit/10.1

See README/readme.traverse for detailed instructions and an example job script.

For help regarding traverse, send email to: cses@princeton.edu

1.4.2 Make

The sources are located in the directory: …/trunk/unstructured".

By default, M3D-C[1] is linked with PETSc and release version of SCOREC libraries.

- For a 2D nonlinear version of the code: make OPT=1 MAX_PTS=25
- For a linear version of the code: make OPT=1 COM=1 MAX_PTS=25
- For a 3D nonlinear version of the code: make OPT=1 3D=1 MAX_PTS=60
- For the stellarator version of the code make OPT=1 3D=1 MAX_PTS=60 ST=1
- To compile all 4 versions do "make all"

To compile M3DC[1] with debug version of SCOREC libraries for lots of sanity checks and informative print statements, add "SCORECVER=debug" to the make command. Note that debug versions are available only on PPPL and NERSC Cori.

The executable files are located in a sub-directory that is named with an underscore followed by the host name and compile options. For a host name "xxxx", these commands will generate a folder and an executable file as the following, respectively.

- *_xxxx-opt-25/m3dc1_*2d
- *_xxxx-complex-opt-25/m3dc1_*2d_complex
- *_xxxx-3d-opt-60/m3dc1_*3d

A Tip for "MAX_PTS": All the M3D-C[1] control parameters are described in the file "C1input" and the file "C1input" should exist in the work folder where the simulation runs. The C1input parameters "int_pts_main", "int_pts_aux", and "int_pts_diag" must be the same or less than MAX_PTS.

1.5 Using pre-installed release versions

An alternative to compiling the code yourself, you can use a pre-installed release version of M3D-C1. The following instructions for using the modules are taken from the "Tutorial" document linked from https://w3.pppl.gov/~nferraro/m3dc1.html

On the PPPL cluster, load the following modules:

```
module use /p/m3dc1/modules
module load m3dc1/1.11
```

Release versions of m3D-C1 have also been installed on a number of other systems. The location of the M3D-C1 modules for each of these systems is:

PPPL Cluster: module use /p/m3dc1/modules
NERSC Cori: module use /project/projectdirs/mp288/C1/modules/cori
Phase 1: module load m3dc1/1.11-haswell
Phase 2: module load m3dc1/1.11-knl
Princeton stellar: module use /home/nferraro/modules
GA Iris: module use /fusion/projects/codes/m3dc1/modules

1.6 Regression Tests for Personal Version (example for cori_knl)

1. compile all versions from …/unstructured (OPT=1, OPT=1 COM=1, OPT=1 3D=1 MAX_PTS=60)
2. export M3DC1_MPIRUN=srun M3DC1_VERSION=local M3DC1_ARCH=cori_knl
   (other M3DC1_MPIRUN=mpiexec, other M3DC1_ARCH=stellar, centos7,m3dc1 ,cori)
3. from …/unstructured "make bin"
4. PATH=$PATH\: ……./unstructured/_$M3DC1_ARCH/bin
5. cd regtest
   ./clean cori_knl
   ./run cori_knl
   (wait until jobs finish)
   ./check cori_knl

NOTE: On some machines, such as cori, there is a limit as to the number of jobs that can be submitted to the debug queue. In this case, you need to wait until one job finishes, and submit the remaining job(s) manually by the command (for KPRAD_restart):
./run cori_knl KPRAD_restart

2. Mesh Management

The M3D-C[1] requires a geometric model and a mesh that are the representation of the analysis domain.

- PUMI is a parallel mesh infrastructure toolkit developed at SCOREC, RPI. For more information, visit http://www.scorec.rpi.edu/pumi
- Simmetrix provides a set of tools and libraries for engineering simulation including a state-of-art mesh generation. For more information, visit http://simmetrix.com.

[The model file extensions referred in this document]

- **.smd: Simmetrix-readable binary format model file**
  - **The model generated with Simmetrix is saved in this format.**
- **.dmg: PUMI-readable binary format model file**
  - **The model generated from PUMI mesh**
- **.txt: M3D-C[1]-readable ascii format model file**
  - **The model is generated from mesh generation tool (See Section 2.2)**
- **arbitrary filename: M3D-C[1]-readable ascii format model file**
  - **The first line of the file should contain five doubles (See Section 2.1)**

[The mesh file extensions referred in this document]

- **.sms: Simmetrix-readable binary format mesh file**
  - **The mesh generated with Simmetrix is saved in this format.**
  - **If a mesh is serial (1-part), the mesh file doesn't have a number before the extension**
  - **If a mesh is distributed (P-part, P>1), the mesh file has a number before the extension to represent the global part ID.**
- **.sms: ASCII format mesh file used in old PUMI stack**
  - **This format is not supported from January 2015**
  - **If a mesh is serial (1-part), the mesh file doesn't have a number before the extension**
  - **If a mesh is distributed (P-part, P>1), the mesh file has a number before the extension to represent the global part ID.**
- **.smb: PUMI-readable binary format mesh file**
  - **This format is used in the current M3D-C[1]**
  - **No matter if a mesh is serial (1-part) or distributed (P-part, P>1), the mesh file has a number before the extension to represent the global part ID.**
- **.vtu/pvtu: binary format mesh file for visualization with paraview. For more information, visit http://paraview.org.**

[Model/Mesh requirements for M3D-C[1]]:

- The model and mesh shall be generated as described in Section 2.1 and Section 2.2.
- The mesh file must be PUMI-readable .smb file. Note that a mesh file contains a "number" before the extension (.smb) to denote a global part ID.

- The model and mesh file must be present in the work directory
- The name of model and mesh file must be specified in "C1input" file in the work directory
    - mesh_model = model_file
    - mesh_filename = mesh_file.smb (NOTE: do not specify a number before the file extension)
- In a 2D run with *P* processes,
    - there should be *P* mesh files with part ID from *0* to *P-1*
- In a 3D run with *P*N* processes where 2D mesh is distributed to *P* parts,
    - there should be P mesh files with part ID from *0* to *P-1*
    - in "C1input" file, specify "*nplanes*" to *N* (e.g. nplanes=8), where "*nplanes*" describes how many 2D mesh copies to be loaded
    - the M3D-C^1 code should be compiled with "3D=1, MAX_PTS=60".
- The previous releases of M3D-C^1 supported the ASCII format .sms mesh files, which are not supported any more. Therefore, any existing ASCII format .sms mesh files shall be converted to binary format (.smb). See Section 2.3 for how to convert the mesh format.

The rest of this section is organized as follows: Section 2.1 describes a simple mesh generation tool without Simmetrix libraries. Section 2.2 describes a mesh generation tool with Simmetrix libraries. Section 2.3 describes how to convert old PUMI mesh file (.sms) to the current mesh format (.smb). Section 2.4 presents how to split a mesh into a bigger number of parts.

2.1 Simple Mesh Generation (not recommended)

This section describes simple model and mesh generation without Simmetrix libraries. Note that this is NO LONGER RECOMMENDED as the Simmetrix libraries produce superior meshes.

[Steps]

1. Create an ascii file of arbitrary name that contains space delimited five doubles to define vacuum wall: X_0 X_1 X_2 Z_0 Z_1 such that
   $X = X_0 + X_1 \cos(\text{theta} + X_2 * \sin(\text{theta}))$
   $Z = Z_0 + Z_1 \sin(\text{theta})$
2. run "create_smb" to generate .smb mesh file
   - argv[1]: the ascii file created in Step 1
   - argv[2]: relative mesh size, which is desired mesh edge length divided by the longest edge of the bounding box of the model.
   - the file "seed0.smb" should be present in the work directory
   - the output mesh is saved in PUMI (.smb) and Paraview (.pvtu)
3. In order to load the model and mesh, locate them in your work directory and modify C1input parameters
   - mesh_model = the ascii file created in Step 1
   - mesh_filename = PUMI-readable mesh file (.smb)
     - NOTE: do not specify the part ID in the mesh filename

[Location of ”create_smb” and ”seed0.smb”]

- **on portal.pppl.gov: /p/tsc/m3dc1/lib/SCORECLib/rhel6/**intel_ver- openmpi_ver**/petsc_ver/bin**
- **on portalc7.pppl.gov: /p/tsc/m3dc1/lib/SCORECLib/rhel7/**intel_ver-openmpi_ver**/petsc_ver/bin**
- **on cori.nersc.gov: /global/project/projectdirs/mp288/cori/scorec/knl|hsw-petsc_ver /bin**
- **on hydra. gate.rzg.mpg.de: /u/m3dc1/scorec/utilities/create_smb**

[Example] ./create_smb AnalyticModel 0.1
Output: AnalyticModel0.smb, AnalyticModel0.vtu and AnalyticModel.pvtu

Note: the file ”seed0.smb” is required to present in the work directory

See ”readme.create_smb” for detailed instructions and trouble shooting tips.

Please be noted that we recommend to use ”m3dc1_meshgen” for the mesh generation (Section 2.2) as it provides more advanced controls and features

2.2 Unstructured Mesh Generation with Simmetrix Libraries

This section describes a mesh generation program “m3dc1_meshgen” that runs with the Simmetrix libraries. It is currently available on the following location on PPPL portal:

- /p/tsc/m3dc1/lib/SCORECLib/rhel7/intel2019u3-openmpi4.0.3/petsc-3.13.5/bin

See ”readme.m3dc1_meshgen” for detailed instructions and trouble shooting tips.

[Steps]

1. Set environment variables for Simmetrix:

   module load intel/version openmpi/version
   module load simmodsuite/14.0-190402dev simmodeler/7.0-190402dev
   module load paraview (for mesh visualization with .vtk files)

2. Create an ascii file of arbitrary name that contains input parameters for model and mesh

- modelType: 0, 1, 2, 3, or 4
  - Type 0: a parameterized vacuum region defined by five doubles for analytic expression. For five doubles X_0 X_1 X_2 Z_0 Z_1, vacuum boundary is defined by
    $X = X_0 + X_1 \cos(\text{theta} + X_2 * \sin(\text{theta}))$
    $Z = Z_0 + Z_1 \sin(\text{theta})$
  - Type 1: a vacuum region defined by piece-wise linear points
  - Type 2: a vacuum region defined by piece-wise polynomials
  - Type 3: spline-fitted 3-region model (plasma, wall and vacuum)
  - Type 4: spline-fitted 3-region model (plasma, wall, and vacuum) with inner & outer boundary points to set resistive wall
- reorder: if 1, reorder PUMI mesh based on adjacency (default: 0) and generate vtk folders for mesh visualization. The mesh before and after reodering is saved in "original-mesh.vtk" and "reordered-mesh.vtk", respectively. Note that the element order of Simmetrix mesh is not affected.
- inFile: (modelType 0) not required
  (modelType 1 and 2) geometry file describing the vacuum
  (modelType 3 and 4) geometry file describing the inner plasma wall
- bdryFile: (modelType 0-3) not required
  (modelType 4) geometry file describing the outer plasma wall
- outFile: output file name to save model and mesh
- meshSize: relative mesh size for each region (default 0.05)
  - for modelType 3, set three doubles for plasma, resistive, vacuum, respectively
- useVacuumParams: for modelType 0 or 3, if 1, use parameterized vacuum wall (default 0)
- vacuumParams: five doubles to describe parameterized vacuum wall. Required if useVacuumParams=1.
- adjustVacuumParams: for modelType 0 or 3, if 1, multiply coordinates and parametric values of nodes on vacuum wall by vacuumFactor. Valid only if useVacuumParams=1 (default 0)
- vacuumFactor: for modelType 0 or 3, an optional double value used to multiply coordinates and parametric values of nodes on vacuum wall when adjustVacuumParams=1. Valid only if adjustVacuumParams=1 (default 2*PI)
- numVacuumPts: optional # interpolation points on parameterized vacuum wall. Valid only if useVacuumParams=1 (default 20)
- meshGradationRate: for modelType 3 or 4, optional mesh gradation rate (default: 0.3)
- resistive-width: for modelType 3, the width of resistive wall. If resistive-width=0, only plasma region is created (default 0.02)
- plasma-offsetX: for modelType 3, the offset in x direction to the left (default 0.0)
- plasma-offsetY: for modelType 3, the offset in y direction to the bottom (default 0.0)
- vacuum-width: for modelType 3 or 4, the width of vacuum region (default 2.5)
  vacuum-height: for modelType 3 or 4, the height of vacuum region (default 4.0)

Four input files are available for your reference:

- analytic-input (modelType 0)
- poly-input (modelType 2)
- circle-input (modelType 3)
- bdry-input (modelType 4)

3. Run "m3dc1_meshgen" – input file, inFile, and bdryFile (if applicable) should be in the work folder
   - argv[1]: the ascii file created in step 2
   - The output model is saved in three formats
     - M3D-C[1]-readable ".txt"
     - simmetrix-readable file ".smd" and
     - PUMI-readable ".dmg"

     For modelType 0-2, the model is saved in outFile.*

     For modelType 3 with resistive width *R*, vacuum-width *W* and vacuum-height *H*, the model is saved in outFile-*R*-*W*-*H*.*.

     For modelType 4 with vacuum-width *W* and vacuum-height *H*, the model is saved in outFile- *W*-*H*.*.

   - The output mesh is saved in three formats
     - Simmetrix-readable ".sms"
     - M3D-C[1]/PUMI readable ".smb"
     - Paraview

     For modelType 0-2 with # mesh faces *F*,
     - if *F* >1000, the mesh is saved in outFile-(*F*/1000).*
     - if *F*<1000, the mesh is saved in outFile-*F*.*

     For modelType 3 with # mesh faces *F*, resistive width *R*, vacuum-width *W*, vacuum-height *H*,
     - if *F* >1000, the mesh is saved in outFile- *R*-*W*-*H* -(*F*/1000).*
     - if *F*<1000, the mesh is saved in outFile- *R*-*W*-*H F*.*

     For modelType 4 with # mesh faces *F*, vacuum-width *W* and vacuum-height *H*,
     - if *F* >1000, the mesh is saved in outFile- *W*-*H* -(*F*/1000).*
     - if *F*<1000, the mesh is saved in outFile- *W*-*H F*.*

4. If the initial mesh is not good enough, run "simmodeler" to generate a mesh with more meshing controls.
   - Save mesh in .sms.
   - To convert Simmetrix mesh (.sms) into M3D-C[1] readable mesh (.smb), run "convert_sim_sms"
     - argv[1]: Simmetrix model file generated in Step 3 (.smd)
     - argv[2]: Simmetrix mesh file generated in Step 4 (.sms)

- argv[3]: output PUMI mesh file name (.smb)
- argv[4]: optional integer to turn ON/OFF adjacency-based mesh reordering in PUMI mesh file (default: 0/OFF).

- See Section 17 for detailed instructions

5. In order to load the model and mesh, locate them in your work directory and modify C1input parameters
   - mesh_model = M3D-C1 readable model file (.txt)
   - mesh_filename = PUMI-readable mesh file (.smb)
     - NOTE: do not specify the part ID in the mesh filename

2.3 Mesh File Conversion

As of approximately 1/27/2015, M3D-C[1] doesn't support old PUMI mesh files, which are ASCII formatted ".sms" files. Therefore, all the existing ascii-formatted ".sms" files need to be converted to binary-formatted ".smb" files. Please email to seols@rpi.edu if you have old PUMI mesh files to convert.

2.4 Mesh Re-Partitioning from P parts to N parts (P<N) aka Splitting

Given P-part input mesh, the program "split_smb" increases # parts to N (P<N).

- **Location: see $SCOREC_UTIL_DIR in host.mk**

Usage: mpirun –np N ./split_smb input-mesh.smb output-mesh.smb X

- input-mesh should be .smb
- output-mesh should be .smb
- N is the number of parts in the output mesh
- For a P-part input mesh, X must be N/P
- For both input and output mesh, do not specify a number before the file extension
- "split_smb" will insert a number in the output mesh file. The number represents a global part ID.
- Make sure that the output mesh doesn't have any empty part. Otherwise, the program crashes with the following error message:

```
APF warning: 1 empty parts

split_smb: /u/sseol/develop/core/mds/mds.c:614: check_ent: Assertion `e >= 0' failed.
```

[Examples]

Example 1: mpirun –np 6 ./split_smb struct-curveDomain.smb part.smb 6

- Input mesh: struct-curveDomain0.smb

- Output mesh: part0.smb, part1.smb, part2.smb, part3.smb, part4.smb, part5.smb

Example 2: mpirun –np 6 ./split_smb struct-curveDomain.smb part.smb 3

- Input mesh: struct-curveDomain0.smb, struct-curveDomain1.smb
- Output mesh: part0.smb, part1.smb, part2.smb, part3.smb, part4.smb, part5.smb

Example 3: mpirun -n 16 ./split_smb in.smb out.smb 4
input mesh parts is 16/8 = 4
This will split 4-part input mesh to 16 parts.

Example 4: mpirun -n 8 ./split_smb in.smb out.smb 2
input mesh parts is 8/2 = 4
This will split 4-part input mesh to 8 parts.

Example 5: mpirun -n 8 ./split_smb in.smb out.smb 8
input mesh parts is 8/8 = 1
This will split 1-part serial input mesh to 8 parts.

2.5 Mesh Re-Partitioning from P parts to N parts (P>N) aka Merging

Given P-part input mesh, the program "collapse" decreases # parts to N (P>N).

- **Location: see $SCOREC_UTIL_DIR in host.mk**

Usage: mpirun –np P ./collapse input-mesh.smb output-mesh.smb X

- input-mesh should be .smb
- output-mesh should be .smb
- P is the number of parts in the input mesh
- For a N-part output mesh, X must be P/N
- For both input and output mesh, do not specify a number before the file extension
- "collapse" will insert a number in the output mesh file. The number represents a global part ID.

[Examples]
Example 1: mpirun -n 4 ./collapse in.smb out.smb 4

- Input mesh: in0.smb, in1.smb, in2.smb, in3.smb
- Output mesh: out0.smb

Example 2: mpirun -n 16 ./collapse in.smb out.smb 4
output mesh parts is 16/4 = 4
This will change a 16-part input mesh into a 4-part mesh.

Example 3: mpirun -n 8 ./collapse in.smb out.smb 2
output mesh parts is 8/2 = 4
This will split 8-part input mesh to 4 parts.

Example 4: mpirun -n 8 ./collapse in.smb out.smb 8

output mesh parts is 8/8 = 1
This will change 8-part input mesh into 1-part serial mesh.

2.6 Mesh Adaptation

There are two ways to run mesh adaptation inside M3D-C1 .

1. by post processed magnetic flux field before time steps or
2. by error estimator at the end of every *N* time step (N>0)

See Section 16.2 for the details of mesh adaptation.

2.7 Element Order Visualization with Paraview

The element order information is provided with vtk files generated with “m3dc1_meshgen”. This section describes the steps to visualize element order using Paraview. For the topics not described herein, such as visualizing ghost elements, mesh parts, etc., please email to seols@rpi.edu or visit https://www.paraview.org/paraview-guide.

[Steps]

1. Launch paraview

module load paraview

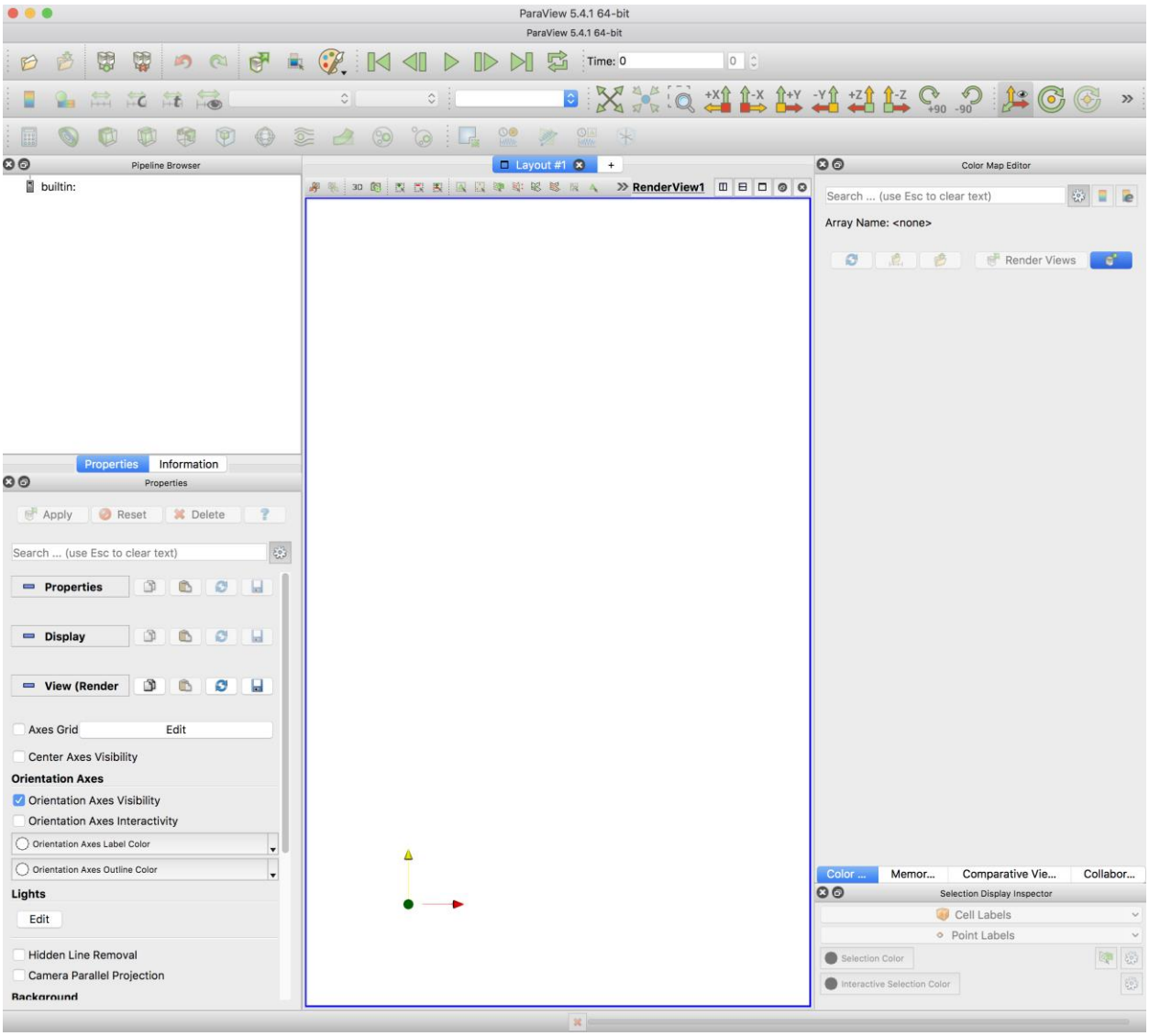


Figure 1 Initial Paraview Window with “White” background

2. To change the background color of Render View, go to the menu “Paraview Preferences” and select the “Color Palette” tab.

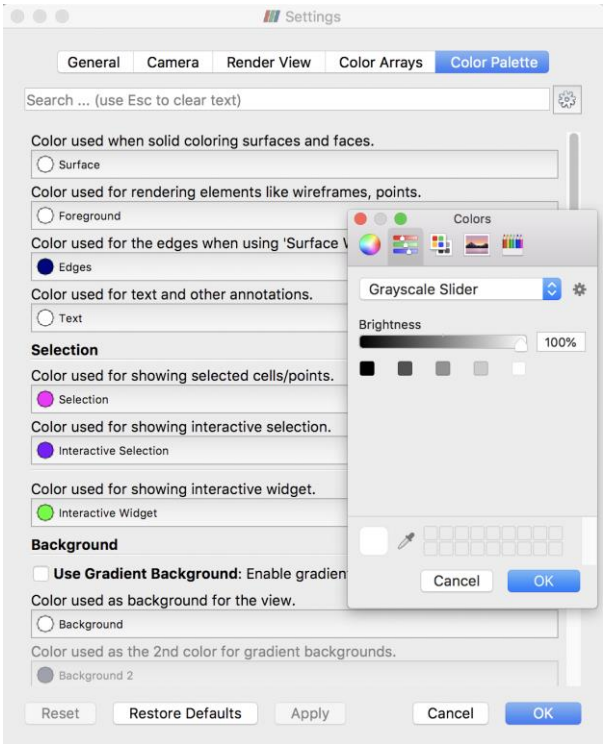


Figure 2 Paraview Settings

3. Using “Open” icon on the top left corner or the menu “File Open”, open a .pvtu file and then click the “eye” sign in “Pipeline Browser” on the left. You can open multiple files at the same time and select one to visualize. When a file is opened, all available attributes are listed in “Cell/Point Array Status” panel.

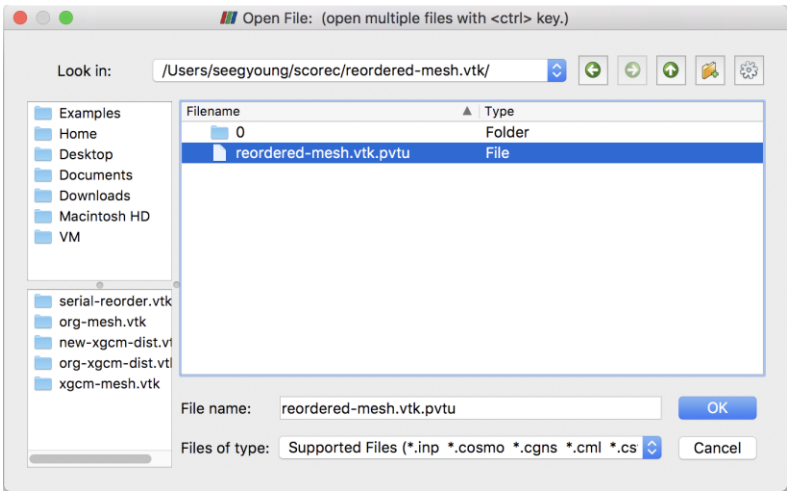


Figure 3 Select .pvtu file to open

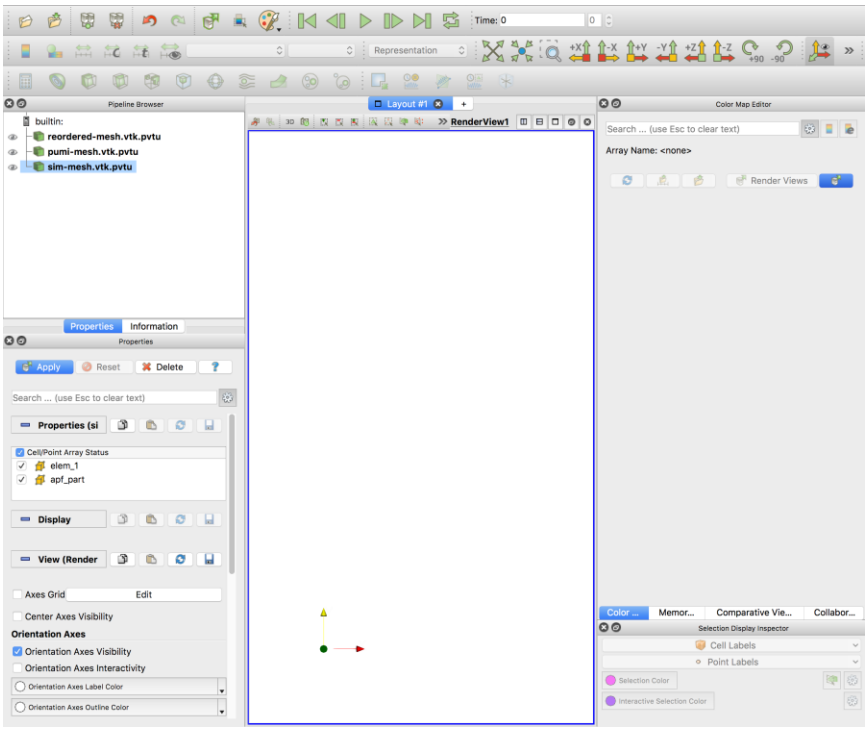


Figure 4 Click an "eye" from opened file list in the Pipeline Browser panel

4. Select “Surface with Edges” to visualize the mesh edges. Use a mouse to zoon in/out (dial in the middle) or move/rotate the mesh (left button).

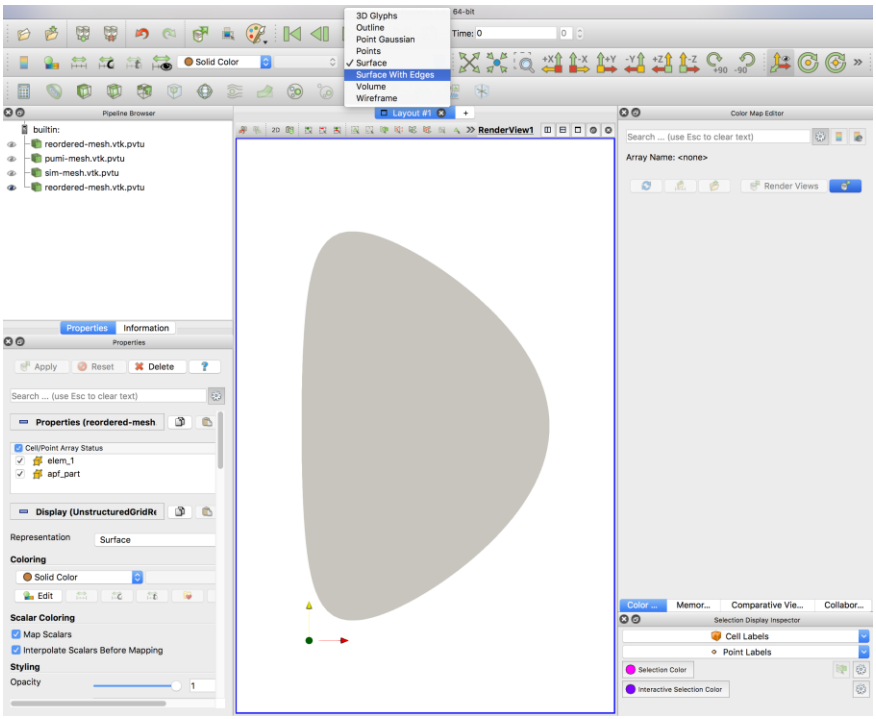


Figure 5 Select "Surface with Edges" for 2D Mesh and click “Apply” in the Properties panel

5. To visualize the element ID, select “elem_1” from the attribute list for Coloring.

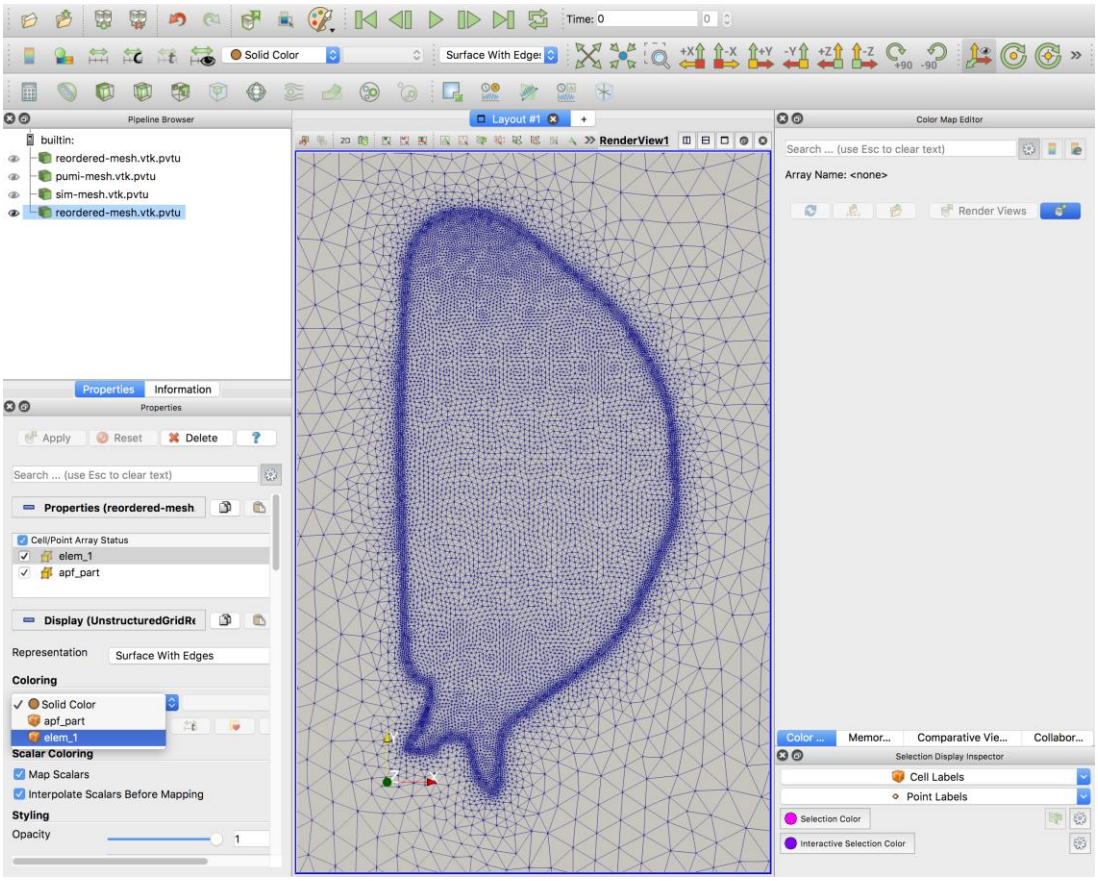


Figure 6 Select "elem_1" from Coloring Property

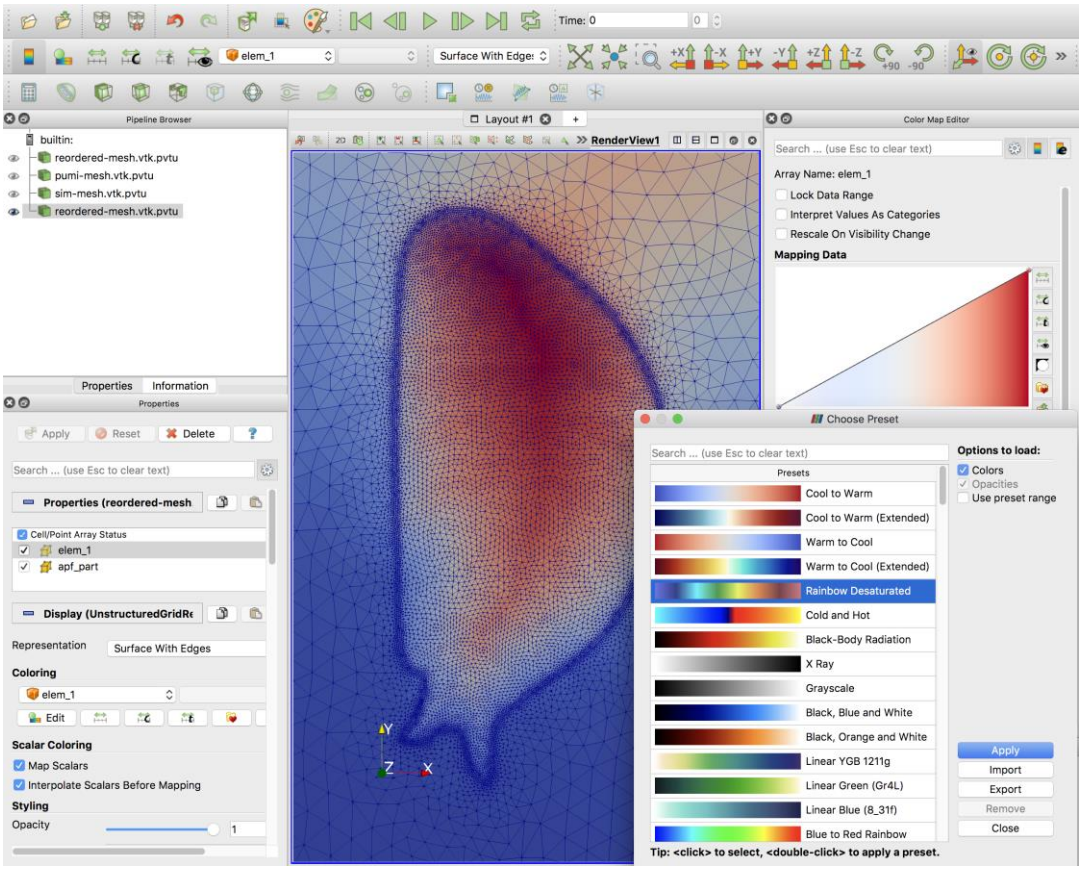


Figure 7 To change colors, click the "heart" button in Coloring panel and select from preset colors

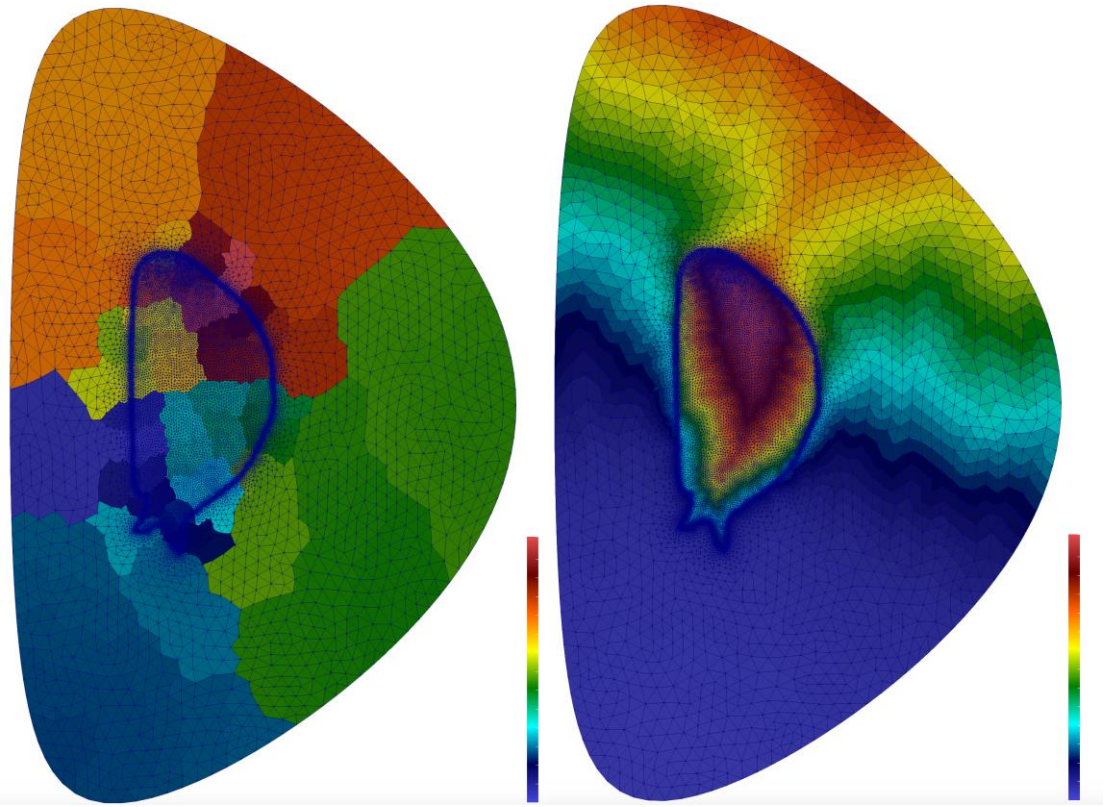


Figure 8 A 2D Mesh without (left) & with (right) adjacency-based ordering

3. Importing Equilibrium Files

3.1 Running from GEQDSK File Equilibrium

In addition to the files:

AnalyticModel
C1input
m3dc1
part0.smb part1.smb …..

You must have a geqdsk file called “geqdsk” in your directory. This is read with the input file option:

iread_eqdsk = 1
inumgs = 0

3.2 Running from JSOLVER Equilibrium File[1]

1. Compile the program “read_jsolver” that reads jsolver equilibria
   - Due to it requires the library “pspline”, compilation is available only on Edison or Portal
   - On Edison, load the module “pspline” and change “NTCCHOME” to “PSPLINE_DIR” in makefile
   - On Portal, load the module “ntcc”
   - Then run “make read_jsolver”

2. Run "read_jsolver" in the directory where the file “fixed” is located then the program will generate a file “POLAR”.

3. Run “convert_polar” to generate a model and mesh file from where the file “POLAR” is located
   - **on portal.pppl.gov: /p/tsc/m3dc1/lib/SCORECLib/rhel7/**intel_ver-openmpi_ver**/ simmodsuite_ver/bin**
   - **on cori.nersc.gov: NOT AVAILABLE**
   - **on hydra.gate.rzg.mpg.de: NOT AVAILABLE**

   Given the input file “POLAR”, the utility “convert_polar” generates the following files:
   - model.dmg: PUMI-readable model file
   - model.txt: M3DC1-readable model file
   - mesh0.smb: PUMI/M3DC1-readable mesh file
   - mesh.vtk: Paraview data files
   - norm_curv: ascii file containing nodes' normal/curvature information

4. Split the initial mesh to N parts and save into part.smb as described in Section 2.4:

```
mpiexec –np N ./split_smb model.dmg mesh.smb part.smb N
```

   :

5. Then, set the following C1input parameters:

```
iread_jsolver=1
ifixedb=1
nonrect=1
inumgs=0
mesh_filename = part.smb
mesh_model = model.txt
```

See ”readme.convert_polar” for detailed instructions and trouble shooting tips.

[1] The JSOLVER inequ file must have isym=0 (no up-down symmetry imposed)

3.3 Running from DSKGATO Equilibrium File

1. Compile the program “readgato” that reads dskgato equilibria
   - It requires the library “pspline”, compilation is available only on Edison or Portal
   - On Edison and portal, load the module “pspline” and change “NTCCHOME” to “PSPLINE_DIR” in makefile
   - Then run “make readgato”

2. Run "readgato" in the directory where the file “dskgato” is located then the program will generate files “POLAR”, “profiles-p”, and “profiles-g”. The profile files must be present where M3D-C[1] runs.

3. Run “convert_polar” to generate a model and mesh file from the file “POLAR”
   Location of ”convert_polar”:
   - **on portalc.pppl.gov: /p/tsc/m3dc1/lib/SCORECLib/rhel7/**intel_ver-openmpi_ver**/petsc-3.13.5/bin**

   Given the input file “POLAR”, the utility “convert_polar” generates the following files:
   - model.dmg: PUMI-readable model file
   - model.txt: M3DC1-readable model file
   - mesh0.smb: PUMI/M3DC1-readable mesh file
   - mesh.vtk: Paraview data files
   - norm_curv: ascii file containing nodes' normal/curvature information

4. Split the initial mesh to N parts and save into part.smb as described in Section 2.4:

```
mpiexec –np N ./split_smb model.dmg mesh.smb part.smb N
```

:

5. Then, set the following C1input parameters:

```
ifixedb=1
nonrect=1
inumgs=1
mesh_filename = part.smb
mesh_model = model.txt
```

See ”readme.convert_polar” for detailed instructions and trouble shooting tips.

4. Running Jobs

4.1 Running 2D or Linear

In 2D, the run can be either linear or nonlinear, depending on the C1input parameter "linear":

linear=0 (non-linear run: must compile with the option RL=1)
linear=1 (linear run: must compile with the option COM=1)

In both cases, set: nplanes=1. For the linear case, set the toroidal mode number with ntor=nn

In your batchfile for job submission, create a local directory and copy the following files over:
m3dc1_2d (non-linear) or m3dc1_2d_complex (non-linear)
C1input
AnalyticModel (or MultiEdgeAnalyticModel)
struct-dmg.sms
(and geqdsk if using option 3.1)

To run non-linear

"mpiexec –np 8 ./m3dc1_2d" (to run on 8 processors at PPPL).

To run linear

"mpirun –np 24 ./m3dc1_2d_complex -pc_factor_mat_solver_package mumps" (to run on 24 processors on Edison)

[NOTES]

- To add PETSc options: -ipetsc (to run petsc)
  -ipetscsuperlu (to run superlu via petsc)
- To add the option to use pdslin instead of SuperLU: –pdslin

4.2 3D Nonlinear

For the 3D nonlinear run, set linear=0 and set "nplanes" equal to the number of toroidal planes. The number of bjacobi blocks in the PETSc options file must also be equal to nplanes (see Section 7). The total number of processors to request must be the product of nplanes and M (the number of processors per plane).

Files required to be copied to the local directory are:

m3dc1
C1input,
partnn.smb (one for each poloidal plane partition)
options_bjacobi
m3dc1.xml (if using ADIOS)

geqdsk (if using option 3.1)

“mpiexec –np 16 ./ m3dc1 –ipetsc –options_file options_bjacobi” (to run 16 processes at PPPL)

See Section 7 for the format of the PETSc option file.

4.3 Batch Files

(submit from portal)

#!/bin/bash -vx

#SBATCH --partition=mque

#SBATCH --nodes=1

#SBATCH –ntasks-per-node=8

#SBATCH –mem-per-cpu=6000mb

#SBATCH –J myjob

#SBATCH -t 10:05:00

#SBATCH --mail-type=END

#SBATCH --mail-user=<username>@pppl.gov

Other partitions and maximum parameters

--partition=	--nodes=	--ntasks-per-node=	--mem-per-cpu
kruskal	36	32	2000mb
dawson	132	16	2000mb
ellis	10	4	4000mb
mque	13	32	6000mb

Initiate mpi in batch script:

#2D complex

mpiexec --bind-to none -np 8 m3dc1_2d_complex -pc_factor_mat_solver_package mumps > m3dc1_out

#2D real

mpiexec --bind-to none -np 8 m3dc1_2d

#3D real

mpiexec --bind-to none -np 32 ./m3dc1_3d -ipetsc -options_file options_bjacobi

NOTE: --bind-to-none enables subsequent jobs to run on separate processes

Submit job

sbatch jobscript

or: sbatch --dependency=afterok:123 jobscript

List all curent jobs for a user

squeue -u <username>

Delete a job

scancel <jobid>

Interactive Jobs

There is a “salloc” command on portal, that includes the different partitions

To submit an interactive job to slurm, you can first use the command "salloc": -- see example

salloc --ntasks=16 --mem=96000 --nodes=1 --partition=mque --time=48:00:00

And, from the head kruskal node allocated, run the command "srun":

srun /path/to/your_code

Changing the time limit of a running job

scontrol update Jobid=###### TimeLimit=3-08:00:00

Checking that the change worked

scontrol show job ###### | grep TimeLimit

4.4 Graphics Files

The graphics files are of two types. There is a single file called: C1.h5 that contains all the time-dependent scalar information. This must be saved and be present in the directory of a job so that it can be added to.

In addition to this file, each plot cycle will produce a file: time_nnn.h5, where nnn is the plot cycle number. The equilibrium is written into a file called equilibrium.h5. These must be stored in the same directory as the C1.h5 file.

4.5 Restart Files

4.5.1 Writing restart files

As of 6/23/2017, the parameter iwrite_restart is deprecated and hdf5 files are written in every time step. Therefore jobs can be restarted from the hdf5 "plot file", the same one that is used by the idl routines to make plots.

By default, the hdf5 files are written in single precision. If idouble_out is set to 1, hdf5 files are written in double precision.

4.5.2 Reading restart files for 2D real, 2D complex, or 3D real

As of 6/23/2017, you must restart from the C1.h5 files by setting "iread_hdf5=1" (default) in the C1input file. Restart with adios or Ascii files "C1restart*" is not supported any longer.

To start a normal simulation with the hdf5 files, set the C1input parameter "irestart" to 1.

However, the files C1.h5 and the final time_nnn.h5 file must be in the working directory. You may also restart from an intermediate time by setting irestart_slice=*nn* where *nn* is the *nn*th plot file. If this is not set, it will restart from the final plot file.

4.5.3 Reading real restart files to initialize 2D complex calculation

- Run 2D linear=0
- Copy 2D C1.h5 and the final time_nnnn.h5 to the working directory
- Run 2D (complex) linear=1. In the initial restart, the time and cycle number will start from t=0 and N=0 for the complex run

4.5.4 Running 3D real simulation from 2D real restart files

To start a 3D simulation with 2D restart files, do the following:

- Run 2D
- Copy 2D C1.h5 and the final time_nnn.h5 to the working directory
- In 3D work folder, set the C1input parameter "irestart= 1"

Regardless of the time step when the restart files were written, the 3D simulation starts with time step 1.

4.5.5 Running 3D real simulation with the number of planes multiplied by *F*

To start a 3D simulation with 3D restart files with the number of planes multiplied by *F*, do the following:

- Run 3D with *P* processes
- In 3D work folder, set the C1input parameter "irestart_factor=*F*"
- Run 3D with *P* **F* processes on N*F planes

As of 3/29/2020, this feature is not supported.

4.6 Monitoring Jobs

You can monitor the progress of your running job in several ways:

A. C1ke file. Each time step, one line will be added to the ASCII C1ke file in the run directory that you can open with a text editor. The first 4 fields are:

cycle time kinetic_energy growth_rate

B. C1.h5 file: You can monitor a time dependent run by using the idl utility described below. Especially useful is the "plot_scalar, 'ke' " command and also "plot_scalar,'ke',/growth".

C. You can use a text editor to monitor the log file slurm-nnnn.out file (where nnnn is the job number assigned by SLURM)

4.7 Exporting Node/Vector/Matrix

(Developers only)

In order to write the node, vector, or matrix data into a file, add the following API's in M3D-C[1].

- To write nodes' global ID and XYZ coordinates, call
```
int m3dc1_node_write(const char* filename, int* s);
```
  - Each process *i* writes local node information into a file "filename-i".
  - Each line of the file consists of node's global ID and three double values representing XYZ coordinates.
  - Global node ID starts from *s*.
  - For a process *i* with *N nodes*, *filename-i* contains *N* lines.

- To write vector, call
```
int m3dc1_field_write(int* field_id, const char* filename, int* s);
```
  - Each process *i* writes local vector information into a file "filename-i".
  - Each line of the file consists of node's global ID and dof value.
  - Global node ID starts from *s*.
  - For a process *i* with *N* nodes and *D* dofs per node, *filename-i* contains *N*D* lines.

- To write PETSc's globally assembled matrix, call
```
int m3dc1_matrix_write(int* matrix_id, const char* filename, int* s);
```
  - Each process *i* writes local PETSc matrix information into a file "filename-i".
  - The first line of the file consists of #rows, #columns, and #non-zero values.
  - From the second line, each line consists of global row index, global column index and a double value representing the non-zero matrix value at the corresponding location.
  - Global row/column index start from *s*.
  - Only globally assembled matrix is written.

- To write Trilinos Epetra matrix, call
```
int m3dc1_epetra_write(int* matrix_id, const char* filename, int*
no_zero, int* s);
```
  - Before global assembly, each process *i* writes local Epetra matrix information into a file "filename-i".
  - After global assembly, each process *i* writes local Epetra matrix information into a file "assembled-filename-I".
  - The first line of the file consists of #rows, #columns, and #values on each process.
  - From the second line, each line consists of global row index, global column index and a double value representing the matrix value at the corresponding location.
  - If the input *no_zero* is equal to 0, all matrix values are written. Otherwise, only non-zero values are written.
  - Global row/column index start from *s*.

The following illustrates an example code snippet that writes node, solution vector and matrix inside the routine gradshafranov_solve (see gradshafranov.f90).

```
subroutine gradshafranov_solve
   …
  integer :: start_index, non_zero
  character(len=1)::node_filename, vector_filename, matrix_filename
   …
  ! solve equation
  call newsolve(gs_matrix,b1vecini_vec%vec,ier)
  ! set starting global ID to 1
  start_index=1
  ! write node coordinates
  call m3dc1_node_write(node_filename, start_index)
  ! write solution vector
  call m3dc1_field_write(b1vecini_vec%vec%id, vector_filename, start_index)
  ! write only non_zero matrix values
#ifdef M3DC1_TRILINOS    ! if Trilinos
  non_zero=1
  call m3dc1_epetra_write(gs_matrix%imatrix, matrix_filename,non_zero,
start_index)
#else                    ! if PETSc
  call m3dc1_matrix_write(gs_matrix%imatrix, matrix_filename, start_index)
#endif
   …
end subroutine gradshafranov_solve
```

For an example mesh on process 0 with Trilinos (122 nodes, 12 dofs per node, global ID starting from 1, the result is the following:

- the node file has 122 lines

```
3d/16p> wc node-0 (node file on process 0)
 122  488 3716 node-0
3d/16p> head node-0
1   4.200000  0.000000  0.000000
2   3.001330  1.300000  0.000000
3   3.800380  0.919239  0.000000
4   3.700000  0.000000  0.000000
5   3.100670  0.650000  0.000000
6   4.091910  0.497488  0.000000
7   3.406430  1.201040  0.000000
8   2.654090  1.201040  0.000000
9   3.450000  0.000000  0.000000
10  3.950000 0.000000 0.000000
```

- the vector file has 122*12=1464 lines

```
3d/16p> wc v-0 (vector file on process 0)
 1464  2928 24360 v-0
3d/16p> head v-0
1   0.000000E+00
1   -1.865168E-01
1   0.000000E+00
1   1.222315E-01
1   3.701650E-04
1   0.000000E+00
1   0.000000E+00
1   0.000000E+00
1   0.000000E+00
1   0.000000E+00
```

- If no_zero=0, the matrix file before assembly contains 1464*1464+1=2143297 lines and the first line of the file is "1464 1464 2143296".

```
$ wc m-zero-0 (matrix file on process 0 with no_zero=0)
 1464  1464 2143296 m-zero-0
```

- If no_zero=1 and #non-zero values in the local matrix is *N*, the matrix file before assembly contains *N*+1 lines and the first line of the file is "1464 1464 *N*".

```
3d/16p> wc m-0 (matrix file before assembly on process 0 with no_zero=1)
 11902  35706 252774 m-0
3d/16p> head m-0
1464 1464 11901
1    1    1.000000E+00
2    1    7.733400E-02
2    2    -4.999638E-03
2    4    4.599472E-05
2    5    -3.387551E-12
2    6    2.977057E-06
2    277  -3.546837E-02
2    278  -3.943023E-04
```

The number of owned nodes on process 0 is *61*. Therefore, the number of rows in the global matrix on process 0 is "61*12=732".

```
3d/16p> wc assembled-m-0 (matrix file after assembly on process 0 with
no_zero=1)
12982  38946 276502 assembled-m-0
3d/16p> head assembled-m-0
732  2808 12981
1    1    1.000000E+00
2    1    7.733400E-02
2    2    -4.999638E-03
```

```
2     3     1.715507E-10
2     4     4.599472E-05
2     5     -3.387551E-12
2     6     2.977057E-06
2     277   -3.546837E-02
2     278    -3.943023E-04
```

The example files are available in`/global/project/projectdirs/mp288/seol/tests/3d/16p (NERSC Edison)`

4.8 Archiving Data at PPPL

Data that is being used in current projects can be stored on the project disk /p/tsc or at /p/m3dc1.

4.9 NERSC: Hung NX

A particularly useful method of accessing NERSC is the "NX" program that can be downloaded from the NERSC web site. Occasionally, this fails or hangs. To fix it (for user u431): ssh nx.nersc.gov (or nerscnx.nersc.gov)

ps –ef|grep nx| grep u431
kill -9 process#

4.10 NERSC: Project Quotas

To check your disk quotas for individual and project, run:
myquota
prjquota mp288 (for repo mp288)

5. Viewing Results and Post-Processing

All of the graphics postprocessors read data from the C1.h5 file and the equilibrium.h5 and time_nnn.h5 files.

5.1 IDL

5.1.1 Compiling and running

To run idl, you should create a directory "idl" in your home directory. In that directory, create a single file named ".startup" that contains the following line:

```
device, retain=2, decomposed=0, true_color=24
```

You also need to include in your .login, .bashrc file (or .cshrc file) the line:
On portal:
export IDL_STARTUP=/u/username/idl/.startup (substitute your unix username for "username").

At NERSC:
export IDL_STARTUP=/global/homes/u/username/idl/.startup (again replace "username")

After you are in idl and the programs are compiled, you can then use the graphical interface by typing "c1view" or use the command line interface. There are over 15 (and still counting) basic idl commands, with required and optional arguments:

5.1.2 idl plot commands

For a more complete list, see: http://w3.pppl.gov/~nferraro/m3dc1.html (click on "IDL Reference)

(Note: to increase size of labels in subsequent plots use: !p.charsize=2)

plot_field,'psi',1,file='fname', points=200, {/lines,clevels=…}, {nlevels=…}, {/iso}, {/linear}, {/mesh}, {/lcfs}, {/xlim}, {/mks}, (here "1" is the time slice (can also use /last), also works for phi, v, chi, I, jphi, jy, jy_plasma, cs, p, den,eta,visc, pforce, pmach). You can also include 2 filenames, filename=["fname1","fname2"], and /diff to plot the difference in the files, or [1,2] and /diff to plot the difference between time slices 1 and 2 of the same file. (for a 3D run, include ",phi=ang" where ang is the angle in degrees. (you can also include cutz=0 to get a profile plot across the midplane. Adding outfile='p_vs_x' will create an ASCII file).

You can also specify an operation with op=#. Currently, the possible "op" values (for psi) are:

1) psi, 2) psi_R, 3) psi_Z, 4) psi_RR, 5)psi_ZZ, 7) psi_RR+psi_ZZ. Adding 10 to any of these will add one toroidal derivative. Adding 20 will add two toroidal derivatives. Adding ",q_con=1" will plot the q=1 contour.

For instance, to visualize *psi* field in time step 0, enter "plot_field,'psi',file='C1.h5',0,/iso"

Note 1: You can get a multicolor color map by specifying, range=[min,max] where min and max are of opposite sign.

Note 2: Some of the name conventions are as follows. In M3D-C[1], the velocity, magnetic, and current density fields are given as follows:

$$\mathbf{V} = R^2\nabla U \times \nabla\varphi + \omega R^2\nabla\varphi + R^{-2}\nabla_\perp\chi$$
$$\mathbf{A} = R^2\nabla\varphi\times\nabla f + \psi\nabla\varphi - F_0 \ln R\,\hat{z}$$
$$\mathbf{B} = \nabla\psi\times\nabla\varphi - \nabla_\perp f' + F\nabla\varphi$$
$$\mathbf{J} = \nabla F^* \times\nabla\varphi + \frac{1}{R^2}\nabla_\perp\psi' - \Delta^*\psi\nabla\varphi$$
$$F^* \equiv F_0 + R^2\nabla^2 f = F + f''$$

'phi' = U , 'v'= ω , 'vz'= $\hat{z}\bullet\mathbf{V}$,'chi'= χ , 'psi'= ψ , 'I'= F , 'f'= f , 'jphi'= $\Delta^*\psi$,' jy'= $-\Delta^*\psi / R$:

Note 3:

1. If there are coils in the grid, 'jy_plasma' will plot just the current due to the plasma current.

2. If you have a wall, adding /bound will draw the contours of the wall.

3. You can get a red-white-blue color scheme with' table=-1'. Adding '/csym' will ensure that zero is white.

plot_field_vs_phi,'te',file='filename',rrange=[,],cutz=,,,,slice=..,op=..

movie_field, 'te', file='filename',nn,range=[min,max],rrange=[rmin,rmax], ext='avi'

this will produce a movie in .avi format of the first nn frames. rrange can be used to control aspect ratio.

Other formats: flv gif matroska mjpeg mov mp4 swf wav webm

plot_scalar, 'ke', {xrange=[,]},{yrange=[,]}, {filename=["fname1","fname2"]}, {/growth}

Magnetic Energy “me”, kinetic energy “ke”

toroidal current ‘it’, plasma current “ip”, Loop Voltage “vl”,

wall current “iw” total current “itot”,

beta toroidal is ‘bt’, poloidal beta “bp”, normal beta “bn”,

total beta “beta”,

“psibound”, “psimin”, “psilim”,

timestep “dt” ”volume” “toroidal flux”,

“reconnected flux”, maximum electron temperature “temax”

thermal energy “p”, electron thermal energy “pe”,

particle number “n”, particles_in_plasma”n_p” electrons “ne”,

angular momentum, “vorticity”,

parallel viscous heating ”bwb2”, “flux”

Internal inductance ”li”, “li3”,

”xmag”, ”zmag”, ”runaways”,

Radiated power “radiation” Ohmic Heating Power “Pohm”

pellet rate “pelr”, pellet var “pelvar”

pellet radius “pelrad”, pellet R position “pelrpos”, pellet Z position “pelzpos”,

“wall_force_n0_x” Total n=0 JxB force on wall in R direction

“wall_force_n0_y” Total n=0 JxB force on wall in direction

“wall_force_n0_z” Total n=0 JxB force on wall in Z direction

“wall_force_n1_x” Total n=1 JxB force on wall in x=R cos(φ) direction

“wall_force_n1_y” Total n=1 JxB force on wall in y=R sin (φ) direction

m_iz Integral Z x Jφ

m_iz_co Integral Z x Jφ cosφ Noll Force

m_iz_sn Integral Z x Jφ sinφ “

Can also add color=[255,135]) Adding outfile='ke_vs_t.txt', etc. will create an ASCII file.

To get the real frequency for linear calculations, use the command:

plot_scalar, 'psi0', /power_spectrum, /ylog, /mks

The time axis will be radians/sec.

plot_energy, 'flux' {/ylog} (also works for 'energy' instead of 'flux')

plot_flux_average,'q',-1, /minor_radius, /xlim {,points=200, bins=100} (also 'p', 'beta')

(-1 indicates equilibrium, /minor radius as opposed to poloidal flux. Also can use /norm for normalized poloidal flux.)

You can use this command to write out a thermal conductivity profile that keeps the pressure fixed when read in with ikappafunc=11. As follows:

'kappa_implied',0,filename='C1.h5',points=400,bins=400,/norm,outfile='profile_kappa'

Then, copy this file profile_kappa to your run directory.

plot_pol_velocity,1, file='fname'{,points=50, maxvel=.01,/lcfs}

plot_timings, file='fname'

plot_mesh, file='fname'

plot_hmn,file=’fname’, yrange=[xxx,xxx], maxn=xxx, {/ylog} ,{/growth} ,{/ke},{/me}This will plot each of maxn Fourier harmonic {or their growth rate} of the kinetic energy {/ke} or magnetic energy {/me} as a function of timestep. (assumes ike_harmonics .ne. 0 and ibh_harmonics.ne.0 in C1input)

plot_kspits, file=’fname’. This will plot the number of PETSc iterations for matrices 5 (velocity), 17(pressure), and 6(magnetic field)

plot_perturbed_surface, 3., slice=1, fac=10., file='fname' . This will plot the perturbed q=3 surface with the normal displacement scaled by a factor of 10. What is actually being plotted is xi = -Te1 / |grad(Te0)| where Te1 and Te0 are the perturbed and equilibrium parts of the electron temperature.

plot_mag_probes, file=’fname’. This will plot a time series of the magnetic probe signals specified by imag_probs

/compensate_renorm Scales the time series to eliminate discontinuities introduced by renormalization in linear stability calculations

/deriv Plot derivative of signal (i.e. B)

/mks Plot values in SI units

/power_spectrum Plot the square of the fourier transform of the series

plot_flux_loops, file=’fname’. This will plot a time series of the flux loop signals specified by iflux_loops.

/compensate_renorm Scales the time series to eliminate discontinuities introduced by renormalization in linear stability calculations

/deriv Plot derivative of signal (i.e. voltage)

/mks Plot values in SI units

plot_at_boundary, ‘l’,file=’fname’,slice=nn,/ynozero. Also works for ‘jnorm’ instead of ‘l’. just adding fname and /iso shows where ‘length along boundary’ parameter actually falls on boundary. Note: sample points can be increased and result smoothed, eg: points=2000, smooth=10

plot_equation,’gradshafranov’,file=’fname’. Plots the different terms in the Grad-Shafranov equation and their sum

5.1.3 Summary table of plot field variables

psi	ψ - toroidal comp. of **A**		jphi	$\Delta^*\psi = -RJ_\varphi$
f	f - "small *f*" in **A** and **B**		jy	Toroidal current density J_φ
I	$F = R * B\_\text{toroidal}$		jy_plasma	J_φ in plasma region only
phi	Velocity variable U		vz	Z component of velocity
v	Angular velocity ω		vor	Vorticity Δ^*U
chi	Velocity variable χ		com	Compressibility $\nabla^2\chi$
p	Total pressure		torque_em	
pe	Electron pressure		torque_ntv	
den	Normalized density		bdotgradp	
te	Electron temperature		bdotgradt	$= n\mathbf{B}\bullet\nabla(p/n)$
ti	Ion temperature		sigma	S_n in density equation
eta	Resistivity η		force_phi	
visc	Viscosity ν		heat_source	S_e in energy equation
visc_c	Compressible viscosity		cd_source	$\dot{\psi} = \ldots + \eta(\Delta^*\psi - \text{cd\_source})$
visc_e	$-\mu_e R^2 \nabla F \bullet \nabla\psi / n\lvert\nabla\psi\rvert^4$		E_R	$E_R = \hat{R}\bullet\left[-\mathbf{V}\times\mathbf{B} + \eta\mathbf{J} + (1/ne)(\mathbf{J}\times\mathbf{B} - \nabla p_e)\right]$
kappa	Iso thermal con: κ		E_PHI	$E_\varphi = \hat{\varphi}\bullet\left[-\mathbf{V}\times\mathbf{B} + \eta\mathbf{J} + (1/ne)(\mathbf{J}\times\mathbf{B} - \nabla p_e)\right]$
vpar	$\mathbf{V}\bullet\mathbf{B} / \lvert B\rvert$		E_Z	$E_Z = \hat{Z}\bullet\left[-\mathbf{V}\times\mathbf{B} + \eta\mathbf{J} + (1/ne)(\mathbf{J}\times\mathbf{B} - \nabla p_e)\right]$
pmach	$\lvert V_\perp\rvert / C_S \times 1/(B_P / B)$		eta_J	$E_\varphi = \hat{\varphi}\bullet[\eta\mathbf{J}]$
vdotgradt[2]	$-n\mathbf{V}\bullet\nabla T - (\gamma-1)nT\nabla\bullet\mathbf{V}$		pforce	$a(1-\tilde{\psi})^N \times d^2 / [(\tilde{\psi} - \psi_c)^2 + d^2]$
eta_jsq[2]	$(\gamma-1)\left[\eta J^2 + S_e\right]$		deldotq_perp[2]	$\nabla\bullet\mathbf{q}_\perp = -\nabla\bullet\kappa_\perp\nabla T_e$
adv1[2]	U part of vdotgradt		deldotq_par[2]	$\nabla\bullet\mathbf{q}_\parallel = -\nabla\bullet\kappa_\parallel\mathbf{bb}\bullet\nabla T_e$
adv3[2]	χ part of vdotgradt		adv2[2]	ω part of vdotgradt
n_re	Number density of runaway electrons (for irunaway-1)		vn	$\mathbf{V}\bullet\nabla\psi / \lvert\nabla\psi\rvert$ "normal velocity"
Potential2[2]	Φ: Scalar Electrical Potential		f1eplot[2]	$-R^2\nabla\cdot\left[\eta\left(\mathbf{J}\times\nabla\varphi\right)\right]$
f3vplot[2]	$(\gamma-1)(T_i - T_e)n_e(QD)$		f2eplot[2]	$-T_e\nabla\bullet D\nabla n_e - T_e S_{ne}$
potential[1]	Electric potential Φ		f3eplot[2]	$-\nabla\cdot\mathbf{q}_\perp - \nabla\cdot\mathbf{q}_\parallel + \eta J^2 + S_e$
eta_jdb[1]	$\eta\mathbf{J}\bullet\mathbf{B}$		psidot[1]	$\hat{\varphi}\bullet\left[-\mathbf{V}\times\mathbf{B} + \eta\mathbf{J} + (1/ne)(\mathbf{J}\times\mathbf{B} - \nabla p_e) + \nabla\tilde{\Phi}\right]$
bdgp[1]	$-\mathbf{B}\bullet\nabla\Phi$		veldif[1]	$\hat{\varphi}\bullet\left[-\mathbf{V}\times\mathbf{B} + \nabla\tilde{\Phi}\right]$
jdbobs[2]	$\mathbf{J}\bullet\mathbf{B} / B^2$		vlbdgp[1]	$(V_L / 2\pi)\mathbf{B}\bullet\nabla\varphi$
rad_source[2]	Total radiated power		Jxb_z	Wall force in z-direction

[1]only for jadv=0, [2]must have itemp_plot=1

RHS of Te equation = eta_jsq + vdotgradt + deldotq_perp + f3vplot + deldotq_par + f2eplot

If (ikprad=1) we have the additional fields associated with impurity radiation:

kprad_sigma_e	Electron source		kprad_sigma_i	Ion source
kprad_rad	Line radiation		kprad brem	Bremsstrahlung radiation
kprad_ion	Total ionization		kprad_totden	Total density of impurity ions
kprad_reck	Recombination (kinetic)		kprad_recp	Recombination (potential)
kprad_n_00	Density of ionization st. 0		kprad_n_01	Density of state 1, etc.

5.1.4 Create postscript files from IDL

To have the IDL output come out on postscript, include the commands:

set_plot, 'ps'
device, /color

In the postscript file, you can change the line width:
10 setlinewidth → 20 setlinewidth
And the color: r g b setrgbcolor (note: 0 0 0 = black, 1 1 1 = white)

5.1.5 Create geqdsk file from IDL

write_geqdsk, points=400, file='C1.h5', slice=0, eqfile='geqdsk.out'
This will write two files geqdsk.out and jsfile (eqdskasci) that are processed as per the stability write-up in Section 6.

5.1.6 XRAY synthetic diagnostic

The xray signal is computed in the following way:
xray_signal(r0) = integral(d3r S(r0-r) * P(r) / |r0-r|^2)

P(r) is the bremsstrahlung power per volume: P(r) = n(r) * sqrt[Te(r)]
(no attempt at getting the overall units correct is made yet). S(r0-r) is the "shape" function of the detector: S(r0-r) = Exp[-a^2/(2.*sigma^2)]
where cos(a) = d.(r0-r)/|r0-r| and d is the unit vector in the direction of the chord of the detector.

The relevant C1input variables are:

xray_detector_enabled	(0 by default; 1 enables the signal calculation).
xray_r0, xray_phi0, xray_z0	(position of the xray detector)
xray_theta	(angle of the chord w.r.thorizontal, in degrees)
xray_sigma	(the variance of the shape function, in degrees)

The position of the detector should be outside the computation domain, otherwise there will be division-by-zero problems. For example, to set up a chord that extends vertically upwards from (R, phi, Z) = (1.6, 0, -2.), with a "width" of 1 degree:

```
xray_detector_enabled = 1
xray_r0 = 1.6
xray_phi0 = 0.
xray_z0 = -2.
xray_theta = 90.
xray_sigma = 1.
```

The xray_signal value is calculated at every timestep (note: you must have ike_only=0 for this to be calculated). The time series can be plotted using "plot_scalar, 'xray_signal'"

If you've set iwrite_aux_vars=1 (it is 1 by default), a field called "chord_mask" will also be output that has the value S(r0-r)/|r0-r|^2. Note that this field will usually be poorly resolved in the phi direction, since you would need ~360 planes to resolve a chord with a width of 1 degree. The actual diagnostic will not be so poorly resolved though, because its resolution is set by the integration quadrature which has ~5x better toroidal resolution than the finite elements.

5.2 Poincare Plots and q-Profiles

To run the field-line tracing code, copy the program “trace” from:

On the PPPL cluster:

/p/tsc/fio/fio-sunfire.r6/bin/trace

At NERSC:

/project/projectdirs/mp288/fio/cori/bin/trace

On stellar:

/home/nferraro/fusion-io/bin/trace

Below are two batch files; one computes a q profile, the other makes a poincare plot. Put these in the directory with your C1.h5 and equilibrium.h5 and timennnn.h5 files and the “trace” run file , and run

the appropriate one (make sure to make them executable with "chmod +x q.bat" and "chmod +x poincare.bat", if necessary. The explanation of each batch file follows:

(NOTE: precede ./trace with “mpiexec –np nproc” at PPPL , etc)

5.2.1 q-profiles

q.bat:

For a linear run or nonlinear run with eqsubtract=1:

```
./trace \
    -m3dc1 C1.h5 \   ! Load equilibrium time slice from C1.h5
    -m3dc1 C1.h5 1 1 \      ! Load time slice 1 from C1.h5 and multiply it by 1
    -dR0 0.1 \              ! First initial point for integration is at (R0 + 0.1, Z0)
    -dR 0.005 \             ! Each subsequent initial point is 0.005 m farther to the right
    -t 100 \                ! For each surface, do 100 toroidal transits
    -s 50 \                 ! for each toroidal transit, do 50 RK4 steps
    -p 200 \                ! Do 200 initial points
    -qout 1 \               ! Output q.out file
    -pout 0                 ! Do not output poincare plot
```

For a nonlinear run with eqsubtract=0:

```
./trace \
    -m3dc1 C1.h5 42 \       ! Load time slice 42 from C1.h5
    -dR0 0.1 \              ! First initial point for integration is at (R0 + 0.1, Z0)
    -dR 0.005 \             ! Each subsequent initial point is 0.005 m farther to the right
    -t 100 \                ! For each surface, do 100 toroidal transits
    -s 50 \                 ! for each toroidal transit, do 50 RK4 steps
    -p 200 \                ! Do 200 initial points
    -qout 1 \               ! Output q.out file
    -pout 0                 ! Do not output poincare plot
```

After running q.bat, you can see the results by running "gnuplot", and within gnuplot running

Plot 'q.out' with lines

Note: you can also add the option "-tavg n" and the code will do a toroidal average over n sampling points.

5.2.2 Poincare-plots

poincare.bat:

For a linear run or a nonlinear run with eqsubtract=1:

```
./trace \
    -m3dc1 C1.h5 \          ! Load equilibrium time slice from C1.h5
    -m3dc1 C1.h5 1 1 \      ! Load time slice 1 from C1.h5 and multiply it by 1
    -dR0 0.1 \              ! First initial point for integration is at (R0 + 0.1, Z0)
    -dR 0.005 \             ! Each subsequent initial point is 0.005 m farther to the right
    -t 100 \                ! For each surface, do  100 toroidal transits
    -s 50 \                 ! for each toroidal transit, do 50 RK4 steps
    -p 200 \                ! Do 200 initial points
    -qout  0 \              ! Do not output q.out file
    -pout  1                ! Do output Poincare plot
```

For a nonlinear run with eqsubtract=0:

```
./trace \
    -m3dc1 C1.h5 42 \       ! Load time slice 42 from C1.h5
    -dR0 0.1 \              ! First initial point for integration is at (R0 + 0.1, Z0)
    -dR 0.005 \             ! Each subsequent initial point is 0.005 m farther to the right
    -t 100 \                ! For each surface, do  100 toroidal transits
    -s 50 \                 ! for each toroidal transit, do 50 RK4 steps
    -p 200 \                ! Do 200 initial points
    -qout  0 \              ! Do not output q.out file
    -pout  1                ! Do output Poincare Plot
```

The script poincare.bat will produce files "out00, out01, ….outnn", one for each initial point that is successively outward. If one or more of those starting points is outside the plasma, the corresponding outnn file will be of zero length. You can check for this by running "ls -l out*" in the directory. If this is the case, you must edit the gplot file and remove mention of these outnn files that are of zero length.

You can see the results by running gnuplot and within gnuplot running:
load 'gplot'

The option "-phi0 0.0" will force all the lines to be launched from phi=0.
The option "–a nn" sets the toroidal angle of the puncture plane to be nn.

5.2.3 Hard-copy plot options for gnuplot

To produce a jpeg (or png) file, precede the “load ‘gplot’ ” command with:
set terminal jpeg (png)
set output ‘fname.jpeg’ (‘fname.png’)

5.3 Visit

Instructions for viewing results using VISIT are on:
http://w3.pppl.gov/~efeibush/visit/m3dc1/

6. Linear Stability Evaluation:

For stability reference, see http://w3.pppl.gov/lmhd

6.1 Transferring Equilibrium from geqdsk.out to Evaluate with PEST

(files should be available at pppl)

(1) geq2ps -pprime –c_ratio:xx geqdsk.out filename.cdf

   a. Where .02 < xx < .15 is the curvature ratio that it cuts the boundary at. This should be as small as possible (where jsolver still converges)

(2) ps2jso –npsi:129 filename.cdf (this produces an eqdska file that can be read by jsolver)

6.2 Linear Stability Evaluation

(1) On Portal, copy files from /p/swim/jchen/IPS/lmhd with:
cp –r /p/swim/jchen/IPS/lmhd/* .

(2) All input files (including eqdska) must be put into the directory “input”

(3) To run starting from jsolver to pest1: (eqdska)
python lmhd_driver.py opt:jso file:jsolver stability:pest1

(4) To run starting from map1: (eqb1)
Python lmhd_driver.py opt:jso file:map1 stability:pest1

(5) Other stability options: pest2, balloon, camino

7. PETSc Option File

When running M3D-C[1] in the 3D nonlinear mode, you need to include PETSc Options file as discussed in Section 4.2. There is a number "8" in the file below. It must be equal to the number of toroidal planes. It should be changed whenever you change the number of planes in the C1input file. The recommended options_bjacobi file is as follows:

-pc_type bjacobi
-pc_bjacobi_blocks 8 (for 8 toroidal planes…should be equal to nplanes in C1input)
-sub_pc_type lu
-sub_pc_factor_mat_solver_package superlu_dist (can exchange mumps for superlu_dist)
mat_superlu_dist_rowperm NOROWPERM (only needed for superlu_dist)
-mat_mumps_icntl_14 50 (only needed for mumps)
(50 means 50% of memory increase when needed. Users can make it 100 or more if encountering a runtime memory issue.)
-sub_ksp_type preonly
-ksp_type fgmres
-ksp_gmres_restart 220
-ksp_rtol 1.e-9
-ksp_max_it 10000
-on_error_abort

-hard_pc_type bjacobi
-hard_pc_bjacobi_blocks 8 (for 8 toroidal planes…should be equal to nplanes in C1input)
-hard_sub_pc_type lu
-hard_sub_pc_factor_mat_solver_type superlu_dist (can change mumps for superlu_dist)
-mat_superlu_dist_rowperm NOROWPERM (only needed for superlu_dist)
-mat_mumps_icntl_14 50 (only needed for mumps.)
(50 means 50% of memory increase when needed. Users can make it 100 or more if encountering a runtime memory issue.)
-hard_sub_ksp_type preonly
-hard_ksp_type lgmres
-hard_ksp_lgmres_argument 4
-hard_ksp_gmres_restart 220
-hard_ksp_rtol 1.e-9
-hard_ksp_max_it 10000

Optional additional optional arguments:
-ksp_converged_reason
-ksp_view
-help

-options_table
-options_left

-trdump
-malloc_log

8. Input Variables in C1input

8.1 Model Options

numvar	3	# of velocity variables 1: 2-Field; 2: 4-Field; 3: 6-Field
linear	0	1: use linearized equations
eqsubtract,	0	1: subtract equilibrium fields
extsubtract	0	1: subtract external fields
icsubtract	0	set to 1 if PF coils are in the domain These are defined in the files: “coil.dat” and “current.dat”
idens	0	1: include density equation
ipres	0	1: include total pressure and electron pressure equations If itemp=1, include both electron and ion temperature equations
ipressplit	0	1: separate pressure solves from field solves when isplitstep=1 (ipressplit must be 0 for isplitstep=0)
itemp	0	1: advance temperatures rather than pressures (if itemp=1 and ipres=1, advance both i and e temperatures)
gyro	0	1: include Braginskii gyroviscosity (note: needs db .ne.0 also)
igauge	0	0: loop voltage applied to boundary psi only
inertia	1	1: include V.Grad(V) terms
itwofluid	1	1: include two-fluid terms (electron form) 2: ion form (not recommended) 3: parallel pressure gradient in Ohm’s law only (not recommended)
ibootstrap	0	1: include bootstraph current
ibootstrap_model	0	1: J_BS = alpha F <p, psi> B
bootstrap_alpha	0	alpha parameter in bootstrap current model
imp_bf	0	1: Include implicit equation for f (recommended for 3D and 2D complex)
nosig	0	1: drop sigma terms from momentum equation
itor	0	1: use toroidal geometry
gravr	0.	
gravz	0.	
istatic	0	1: do not advance velocity fields 3: zero out “chi” velocity field only
iestatic	0	1: do not advance magnetic fields
chiiner	1.	factor to multiply the chi equation inertial terms
ieq_bdotgradt	1.	1: include equilibrium parallel T gradient
no_vdg_T	0	1: do not include V dot grad T in Temp equation (debug)
iwall_is_limiter	1	1: wall acts as limiter
kinetic	0	1: Use kinetic PIC for hot pressure 2: Incompressible CGL 3: Full CGL

iadiabat	0	1: Corrects several problems with itemp=1 option All new runs should have iadiabat=1 (added 6/2/2016)
irunaway	0	1: include runaway electron model
imp_temp	0	0: compute temperatures for isplitstep=0, itemp=0
iohmic_heating	1	1= include Ohmic heating terms in heating
irad_heating	1	1 = include radiation heat sink

8.2 Equilibrium

itaylor	0	switch for which of many test problems to initialize **for itor=1** (toroidal geometry) itaylor=0: tilting cylinder itaylor=1: calls Grad-Shafranov solver itaylor=2: magneto-rotational equilibrium itaylor=3: rotational instability itaylor=40: Fixed boundary stellarator itaylor=41: Free boundary stellarator **for itor=0** (slab geometry) itaylor=0: tilting cyclinder itaylor=1: Taylor reconnection itaylor=2: force free taylor state itaylor=3: GEM reconnection problem itaylor=4: wave propagation itaylor=5: gravitational instability equilibrium itaylor=6: Strauss equilibrium itaylor=7: circular_field_init itaylor=8,9: biharmonic itaylor=10,11,12,13 : analytic RWM test problem itaylor=14: 3D wave test itaylor=15: 3D diffusion test itaylor=16 : FRS cyclindrical equilibrium (see Section 19.1) itaylor=17: ftz_init itaylor=18: eigen_init itaylor=19: ASDEX profiles similar to Yu's itaylor=20: kstar profiles with multiple q=1 surfaces itaylor=21,22: fixed q(r) and p(r) profiles itaylor=23: startsev equilibrium with J = (2/R_0q_0)(1-r^2) itaylor=27: cylindrical test problem (see Section 18) itaylor=29: basicj profiles
iupstream	0	1: adds diffusion term to convection like upstream differencing
magus	5.e-2	magnitude of the upstream diffusion term
Iflip	0	1: flip handedness of coordinates
iflip_b	0	1: reverse equilibrium toroidal field
iflip_j	0	1: reverse equilibrium toroidal current
iflip_v	0	1: reverse equilibrium toroidal velocity
iflip_z	0	1: flip equilibrium across z=0 plane

icsym	0	symmetry of initial perturbation 0: no symmetry, 1: even in U 2: odd in U For icsym=3, maxn=1: $U = \text{eps} \times r^2 e^{-r^2/(\ln)} \sin(\theta - \varphi)$
bzero	1.	vacuum toroidal field is bzero at R=rzero
bx0	0.	initial field in x-direction for some test problems
vzero	0.	initial toroidal velocity for some test problems
phizero	0.	initial poloidal velocity stream function for some test problems
v0_cyl	0.	Central toroidal velocity for cylindrical test problem
v1_cyl	0.	VZ = v0_cyl + v1_cyl*psi**beta 0 < psi < 1 for cyl. test problem
idevice	0	define coils for a particular device -1: reads coil.dat file 0: generic dipole configuration 1: CDX-U 2: NSTX 3: ITER 4: DIII
iwave	0	defines which wave to initialize in wave propagation test
eps	0.01	magnitude of initial random perturbations
maxn	200	maximum Fourier component for random initial perturbation
verzero	0	magnitude of initial vertical velocity
irmp	0	1: apply nonaxisym. fields throughout plasma reads rmp_coil.dat for (R, Z) of window pane coils reads rmp_current.dat for (+-) currents in kA and phases in degrees toroidal mode number of current specified by ntor 2: apply nonaxisym. fields only at boundaries
rmp_atten	0	additional exponential decay of RMP field from r=1 for irmp=2
iread_ext_field	0	1: read external field
beta	0.	parameter used in some model equilibrium initializaitons
ln,	0	length scale parameter used in some model equilibrium
elongation	1	elongation used in Solovev equilibrium
isample_ext_field	1	factor to down-sample external field data toroidally
isample_ext_field_pol	1	factor to down-sample external field data poloidally
scale_ext_field	1	factor to scale external field
shift_ext_field	0	toroidal shift (in deg) of external fields
ibasicj_solvep	0	0: uniform p, solve for F; 1: uniform F, solve for p
basicj_nu	1	exponent in basicj equilibrium
basicj_j0	1	On-axis current density in basicj equilibrium
basicj_voff	1	Radial extent of flat toroidal rotation in basicj equilibrium
basicj_vdelt	1	Width of velocity drop-off, as fraction of ln, in basicj equilibrium
basicj_dexp	1	parameter for basicj equilibrium
basicj_dvac	1	parameter for basicj equilibrium
basicj_q0	0	parameter for basicj equilibrium
basicj_qa	0	parameter for basicj equilibrium
pf_shift	0	(array) horizontal shift of PF coil
pf_shift_angle	0	(array) direction of PF shift in degrees
pf_tilt	0	(array) Angle of PF from vertical indegrees

pf_tilt_angle	0	(array) Axis of rotation for PF tilt in degrees
tf_shift	0	horizontal shift of TF coil
tf_shift_angle	0	direction of TF shift in degrees
tf_tilt	0	angle of TF from vertical in degrees
tf_tilt_angle	0	axis of rotation for TF tilt in degrees

8.3 Grad-Shafranov Solver

For the definition of Grad-Shafranov solver, see Section 11.

inumgs	0	1: use numerical def. of p and g from profile-p and profile-g files
igs	80	max number of Grad-Shafranov iterations
eta_gs	1000.	factor for smoothing nonaxisymmetries in psi in 3D GS solve
igs_pp_ffp_rescale	0	1: rescale p' and FF' to match p and F
nv1equ	0	1: use numvar=1 equilibrium for numvar>1
tcuro	1.	scaled initial plasma toroidal current ($\mu_0 I_p$ in GS)
xmag	1.	R-coordinate of initial current centroid (R_0 in GS)
zmag	0.	Z-coordinate of initial current centroid
xmag0	0.	non-zero: specify target magnetic axis x-position for feedback
zmag0	0.	non-zero: specify target magnetic axis z-position for feedback
xlim	0.	R-coordinate of limiter #1
zlim	0.	Z-coordinate of limiter #1
xlim2	0.	R-coordinate of limiter #2
zlim2	0.	Z-coordinate of limiter #2
rzero	1.	nominal major radius of device for itor=1 (rzero*bzero = g_0 in GS)
libetap,	1.2	approximate initial value of $\ell_i / 2 + \beta_P$ for free-boundary equ.
p0,	0.01	initial central pressure (if reading p from file, still used in density def)
pi0	0.005	initial central ion pressure. electron pressure is pe = p0 – pi0
p1	0	analytic pressure function parameter (p_1 in GS)
p2	0	analytic pressure function parameter (p_2 in GS)
pedge	-1.	pressure outside separatrix (ignore if < 0) (p_e in GS). Also, BC on pressure
tedge	-1.	temperature outside separatrix (ignore if < 0). Only used in GS solve. Boundary value of electron temp is twall = pedge*pefac/den_edge
expn	0.	density profile $n = \mathrm{den0} \times \left(p / p_0 \right)^{\mathrm{expn}} + \mathrm{den\_edge}$ for idens.ne.0 and idenfunc=0
q0	1.	central safety factor for analytic function(q_0 in GS)
djdpsi	0.	parameter in analytic equilibrium function(J_ψ in GS)
th_gs	0.8	implicitness of GS Picard iterations
tol_gs	1.e-8	convergence criteria for GS iteration
psiscale,	1.	depricated
pscale	1.	factor multiplying pressure profile
bscale	1.	factor multiplying toroidal field profile
bpscale	1.	Factor multiplying F' (keeping F0 constant)
vscale	1.	Factor multiplying toroidal rotation profile

iread_bscale	0	1: read profile_bscale for factor to scale F
iread_pscale	0	1: read profile_pscale for factor to scale p and p'
batemanscale	1.	Bateman scale the TF, keeping current profile fixed
irot	0	1: include toroidal rotation in equilibrium calculation (see Section 12)
iscale_rot_by_p	1	see below and Section 12
alpha0	0.	α_0 in analytic rotation profile
alpha1	0.	α_1 in analytic rotation profile
alpha2	0.	α_2 in analytic rotation profile
alpha3	0.	α_3 in analytic rotation profile

For iread_omega=0, the function $\alpha(\psi)$, is parameterized by:

$$\tilde{\alpha} = \alpha_0 + \alpha_1 s + \alpha_2 s^2 + \alpha_3 s^3$$

For iscale_rot_by _p = 0: $\alpha = \tilde{\alpha} \times n(\psi) / p(\psi)$,

For iscale_rot_by_p = 1: $\alpha = \tilde{\alpha}$

For iscale_rot_by_p = 2: $\alpha = \left[\alpha_0 + \alpha_1 e^{-\left[(\psi-\alpha_2)/\alpha_3\right]^2} \right] \times n(\psi) / p(\psi)$

In all cases, the angular velocity is then determined by:

$$\omega = \left[\frac{2\alpha}{R_0^2} \frac{p(\psi)}{n(\psi)} \right]^{1/2}$$

idenfunc 0 select density function. Here $\tilde{\psi} = (\psi - \psi_{\min}) / (\psi_1 - \psi_{\min})$

0: $n = \text{den0} \times (p / p_0)^{\text{expn}} + \text{denedge}$

1: $n = \text{den0} \times \frac{1}{2} \times \left[1 + \tanh\left(\frac{(\tilde{\psi} - (\text{psibound+denoff} \times (\text{psibound-psimin})))}{(\text{dendelt} \times (\text{psibound-psimin}))} \right) \right]$

2: $n = \text{den0} + \frac{1}{2}(\text{den\_edge-den0}) \times \left[1 + \tanh\left((\tilde{\psi} - \text{denoff}) / \text{dendelt} \right) \right]$

3: if $\tilde{\psi} < \text{denoff}$ and $(\psi_l - \psi_0)\left[d\psi/dx(x - x_{\text{MA}}) + d\psi/dz(z - z_{\text{MA}}) \right] > 0$

Then $n = \text{den0}$

Else $n = \text{den\_edge}$

den_edge	0.	edge density. If zero, set to den0*(pedge/p0)**expn
den0	1.	central density
dendelt	0.1	width of transition region for idenfunc=1,2
denoff	1.	offset for idenfunc= 1,2,3
divertors	0	number of divertors (for use in equilibrium field calculation)
xdiv	0.	R position of divertor coils
zdiv	0	Z position of divertor coil
divcur	0.1	normalized current in divertor coil
xnull	0.	guess for R-coordinate of active x-point

znull,	0	guess for Z-coordinate of active x-point
mod_null_rs	0	if 1, you can reset xnull and znull from C1input at restart
xnull0	0	Target R-coordinate of x-point for feedback
znull0	0	Target Z-coordinate of x-point for feedback
xnull2	0.	guess for R-coordinate of inactive x-point
znull2,	0	guess for Z-coordinate of inactive x-point
mod_null_rs2	1	if 1, you can reset xnull2 and znull2 from C1input at restart
gs_pf_psi_width	0	width of psi smoothing into private flux region
gs_vertical_feedback	0	proportional FB of each coil to (zmag-zmag0) (array)
gs_vertical_feedback_i	0	integral FB of each coil to (zmag-zmag0) (array)
gs_vertical_feedback_x	0	proportional FB of each coil to (znull-znull0) (array)
gs_vertical_feedback_x_i	0	integral FB of each coil to (znull-znull0) (array
gs_radial_feedback	0	proportional FB of each coil to (xmag-xmag0) (array)
gs_radial_feedback_i	0	integral FB of each coil to (xmag-xmag0) (array)
gs_radial_feedback_x	0	proportional FB of each coil to (xnull-xnull0) (array)
gs_radial_feedback_x_i	0	integral FB of each coil to (xnull-xnull0) (array
igs_extend_p	0	extend p past pis=1 using ne and Te profiles
igs_feedfac	1	proportionality factor for external field feedback
igs_forcefree_lcfs	-1	ensure that GS solution is force-free at LCFS
igs_start_xpoint_search	0	number of GS iterations before searching for x-point
sigma0	0	width of Gaussian for initial current distribution for GS iteration
igs_extend_diamag	1	1=extend diamagnetic rotation past psi=1

8.4 Transport Coefficients

ivisfunc 0 select viscosity function:

0: visc = amu

1: visc = amu + $\frac{1}{2}$amu_edge$\times\left\{1+\tanh\left[\frac{\psi-\left(\psi_l+\text{amuoff}\times\left(\psi_l-\psi_0\right)\right)}{\text{amudelt}\times\left(\psi_l-\psi_0\right)}\right]\right\}$

2: visc = amu + $\frac{1}{2}$amu_edge$\times\left\{1+\tanh\left[\frac{\tilde{\psi}-\text{amuoff}}{\text{amudelt}}\right]\right\}$ $\tilde{\psi}\equiv\frac{\psi-\psi_0}{\psi_b-\psi_0}$

if (amuoff2.ne.0 .and. amudelt2.ne.0)

visc = amu + $\frac{1}{4}$amu_edge$\times\left\{2+\tanh\left[\frac{\tilde{\psi}-\text{amuoff}}{\text{amudelt}}\right]+\tanh\left[\frac{\tilde{\psi}-\text{amuoff2}}{\text{amudelt2}}\right]\right\}$

3: visc=amu or amu_edge depending on criteria in define_fields

amuoff 0.

amudelt 0.

amuoff2 0.

amudelt2 0

amu 0. isotropic viscosity

amuc 0. compressional viscosity

amue 0. bootstrap viscosity coefficient

amupar 0. parallel viscosity

amu_edge 0.

iresfunc 0 select resistivity function

0: eta = etar + eta0 x $(n_e/p_e)^{3/2}$

1: eta = etar + $\frac{1}{2}$eta0$\times\left\{1+\tanh\left[\frac{\psi-\left(\psi_l+\text{etaoff}\times\left(\psi_l-\psi_0\right)\right)}{\text{etadelt}\times\left(\psi_l-\psi_0\right)}\right]\right\}$

2: eta = etar + $\frac{1}{2}$eta0$\times\left\{1+\tanh\left[\frac{\tilde{\psi}-\text{etaoff}}{\text{etadelt}}\right]\right\}$; $\tilde{\psi}\equiv\frac{\psi-\psi_0}{\psi_b-\psi_0}$

The following two options are applied in a way that they should not have negative values…even if the idl plots indicate otherwise

3: $\tilde{\psi}=\left(\psi-\psi_0\right)/\left(\psi_l-\psi_0\right)$

eta = etar for $\tilde{\psi}$ < etaoff, otherwise eta0

4: eta = Spitzer resistivity with offset.

Define: $T_{wall}\equiv$ pedge*pefac/den_edge

$$\eta\sim\begin{cases}\left(T_e-T_e^{off}\right)^{-3/2} & T_e>T_{wall}-T_e^{off}\\ \left(T_{wall}-T_e^{off}\right)^{-3/2} & T_e\le T_{wall}-T_e^{off}\end{cases}$$

Can be increased by inputing eta_fac > 1.

5: simple neoclassical model:

		eta = eta0 x $(n_e/p_e)^{3/2}/(1 - 1.46\ (r/R)^{1/2})$
eta_te_offset	0.	T_e^{off} for iresfunc=4
ikprad_te_offset	0	if 1, T_e^{off} also applied in kprad and ablation routines
eta_fac	1	for iresfunc=4, eta gets multiplied by eta_fac
etaoff	0.	see description of iresfunc
etadelt	0.	see description of iresfunc
etar	0.	see description of iresfunc
eta0	0.	see description of iresfunc
eta_mod	0	1: remove d/dphi terms in resistivity
eta_max	0	maximum resistivity in plasma region (defaults to etavac)
ikappafunc	0	select electron thermal conductivity function
	0:	kappa = kappat + kappa0 x $(n^3/p)^{1/2}$
	1:	$\text{kappa} = \text{kappa0} \times \frac{1}{2}\left[1 + \tanh\left(\frac{\psi - \left(\psi_l + \text{kappaoff} \times \left(\psi_l - \psi_0\right)\right)}{\text{kappadelt} \times \left(\psi_l - \psi_0\right)}\right)\right]$
	2:	$\text{kappa} = \begin{cases} \text{kappa0} \times \frac{1}{2}\left[1 + \tanh\left(\frac{\tilde{\psi} - \text{kappaoff}}{\text{kappadelt}}\right)\right] & \text{for } \tilde{\psi} < 1 \\ \text{kappa0} \times \frac{1}{2}\left[1 + \tanh\left(\frac{2 - \tilde{\psi} - \text{kappaoff}}{\text{kappadelt}}\right)\right] & \text{for } \tilde{\psi} \geq 1 \end{cases}$
	3:	kappa= kappat + kappa0 x 1 /(p n)$^{1/2}$
	4:	kappa = kappat + kappa0*(1 + kappadelt*$\lvert\nabla$Te$\rvert^2$)
	5:	kappa = kappat + kappa0/Te …. Limited by kappa_max
	10,11:	read from profile_kappa file in m^2/s (10) or normalized units (11)
	12:	option to go with itaylor=27
kappai_fac	1	ion thermal conduction is kappai_fac*kappa
ikapscale	0	1: kappar gets scaled by kappa
ikappar_ni	0	1: include 1/n terms in parallel heat flux
kappaoff	0.	see ikappafunc
kappadelt	0.	see ikappafunc
kappat	0.	isotropic thermal conductivity
kappa0	0.	see ikappafunc
ikapparfunc	0:	Parallel thermal conductivity (PTC) = kappar
		PTC = $\text{kappar} \times \left[\left(T_e / T\right)^{5/2} + 1\right]^{-1}$
kappar	0.	parallel thermal conductivity
tcrit	0.	Te for ikapparfunc = 1
kappari_fac	1.	Ion parallel thermal conductivity is kappari_fac x electron value
kappax	0.	coefficient of B x Grad(T) temperature diffusion
kappah	0.	if kappah .ne. 0 $\text{kappa} = \text{kappah} \times \tanh^2\left[\left(\tilde{\psi} - 1.\right)/.2\right]$
kappaf	1.	Factor to multiply kappa when grad(p) < gradp_crit
kappag	0.	Thermal diffusion proportional to pressure gradient
gradp_crit	0.	Critical pressure gradient for kappaf, kappag model
k_fac	1.	Factor by which TF is multiplied in denominator of kappa_par
temin_qd	0.	Min temperature used in equipartition for ipres=1

idenmfunc		0	Selects form of particle diffusion
	0	denm79 = denm	
	1	denm79 = denm + denmt/Te	
	10	read from file profile_denm in m^2/sec	
	11	read from file profile_denm in normalized units	
denm		0	
denmt		0	multiplier of 1/Te for idenmfunc = 1
denmmin		0.	minimum value of denm
denmmax		1.e6	maximum value of denm

8.5 Hyper Diffusivity

imp_hyper	0	$\lambda_H \nabla^2 \mathbf{J}$ explicit for ψ, implicit for F
	1	$\lambda_H \nabla^2 \mathbf{J}$ implicit for ψ, implicit for F
	2	$(\mathbf{B}/B^2)\nabla\bullet\lambda_H \nabla\sigma$ implicit for ψ, implicit for F ($\sigma \equiv \mathbf{J}\bullet\mathbf{B}/B^2$)
deex	1.	scale length used in the hyper coefficients (see ihypdx)
hyper	0.	hyper coefficient for psi equation
hyperc	0.	hyper coefficient for poloidal velocity
hyperi	0	hyper coefficient for toroidal field
hyperp	0.	hyper coefficient for pressure
hyperv	0.	hyper coefficient for toroidal flow
ihypdx	2	hyper terms multiplied by deex**ihypdx
ihypeta	1	1: magnetic field hyper coefficients multiplied by eta 2: magnetic field hyper coefficients multiplied by p (for imp_hyper=1)
ihypamu	1	1: velocity hyper coefficients are multiplied by amu
ihypkappa	1	1: pressure hyper coefficients are multiplied by kappa

8.6 Normalizations

b0_norm	1.e4	normalization magnetic field (in G)
n0_norm	1.e14	normalization density (in e-/cm3)
l0_norm	100.	normalization length (in cm)

8.7 Boundary Conditions

isurface	1	include surface terms in Galerkin method
icurv	2	if > 0, include curvature from mesh
nonrect	0	1: non-rectangular boundary
ifixedb	0	1: force psi=0 on boundary
com_bc	0	1: forces del^2(chi) = 0 on boundary
vor_bc	0	1: forces del^*(phi) = 0 on boundary
iconst_p	1	1: hold pressure constant on boundary
iconst_n	1	1: hold density constant on boundary
iconst_t	1	1: hold temperature constant on boundary
iconst_bn	1	1: hold normal field constant on boundary
iconst_bz	0	1: hold toroidal field constant on boundary Linear runs should normally have iconst_bz=0 Nonlinear runs normally have iconst_bz=1 (See Section 20)
inograd_p	0	1: no normal pressure gradient on bounday
inograd_t	0	1: no normal temperature gradient on boundary
inograd_n	0	1: no normal density gradient on boundary

inonormalflow	1	1: no-normal-flow boundary condition
inoslip_pol	1	1: no-slip boundary condition on poloidal velocity
	2:	no-slip only on the sum of the two poloidal velocity componnets
inoslip_tor	1	1: no-slip boundary condition on toroidal velocity
inostress_tor	0	1: no stress (toroidal flow) on boundary
inocurrent_pol	0	1: no poloidal current on boundary
inocurrent_tor	0	1: no toroidal current on boundary
inocurrent_norm	0	1: no normal current on boundary
ifbound	-1	boundary condition on f
		1: Dirichlet, 2: Neumann (See Section 20)
iconstflux	0	1: conserve toroidal flux in nonlinear calculation
iper	0	1: periodic boundary condition in R direction
jper	0	1: periodic boundary condition in Z direction

8.8 Time Step

ntimemax	20	total number of time steps
integrator	0	0: Crank-Nicholson, 1: BDF2
isplitstep	1	0: unsplit time step; 1: split time step
iteratephi	0	1: iterate field solve
imp_mod	1	type of split step.
		0: standard, 1: caramana
		For nonlinear runs, normally set imp_mod=0 (more stable)
		For linear runs with isplitstep=1 set imp_mod=1 (more accurate)
idiff	0	1: solve for difference between n and n+1 in B,p
idifv	0	1: solve for difference between n and n+1 for V
		For idiff=idifv=1, should increase ksp_rtol from 10^{-9} to 10^{-8}
irecalc_eta	0	1: recalculate transport coefficients after density solve
iconst_eta	0	1: don't evolve resistivity
itime_independent	0	1: exclude d/dt terms
thimp	0.5	implicitness of timestep (.5<thimp<1)"
thimpsm	1.	implicitness parameter for smoothers
harned_mikic	0.	coefficient of Harned-Mikic 2F stabilization term
isources	0	1: include "source" terms in velocity advance
nskip	1	number of times steps per matrix recalculation
pskip	1	number of times the preconditioner is reused
iskippc	1	number of times preconditioner is reused
dt	0.1	initial size of time step. Can only change on restart if dtkecrit=0
ddt	0	
frequency	0	frequency in time-independent calculations

! variable_timestep parameters: Timestep is constant unless dtkecrit.ne.0

dtmin	4.0	minimum timestep for variable timestep calculation
dtmax	40.	maximum timestep for variable timestep calculation
dtkecrit	0.	lower timestep if ekin is above this (0.01 typical)
dtfrac	.10	max fractional change of timestep in 1 cycle

max_repeat	3	max # time step is repeated for ksp_max iterations exceeded
ksp_max	10000	max number of ksp iterations before repeating time step
ksp_min	1200	increase dt if ksp < ksp_min
ksp_warn	1600	decrease dt if ksp > ksp_warn

8.9 Mesh

nplanes	1	number of toroidal planes for 3D nonlinear
xzero	0.	R-coordinate of lower left corner for rectangular mesh
zzero	0.	Z-coordinate of lower left corner for rectangular mesh
tiltangled	0.	angle a rectangular mesh is tilted
mesh_model		model file name from which the mesh is generated
mesh_filename		mesh file name
ipartitioned	0	obsolete

NOTE:

- The mesh file in the input directory must have a digit representing part ID between filename and .smb even when it's a serial mesh. For instance, for 2D serial mesh, the mesh file in the input directory should NOT be struct-curveDomain.smb, BUT be struct-curveDomain0.smb. However, in C1input, specify filename.smb (no part ID) for mesh_filename.
- For more details on model and mesh file, see Section 2.

Imatassemble	0	1: use petsc matrix parallel assembly instead of scorec
imulti_region	0	1: Mesh has multiple regions that includes resistive wall and vacuum. Wall resistivity is “eta_wall” (poloidal resistivity is eta_wallRZ. Vacuum resistivity is “eta_vac”
toroidal_pack_angle	0	toroidal angle of maximum mesh packing
toroidal_pack_factor	1	ratio of longest to shortest toroidal element

8.10 Solver

NOTE: These are over written by petsc options file (see Section 7)

solver_type	0	for PETSc only. 0: direct solver, 1: iterative solver for Trilinos, iterative solver is used
solver_tol	1.e-9	solver tolerance
num_iter	100	maximum number of iterations

8.11 Mesh Adaptation (Will be deprecated soon)

iadapt	0	0: no adaptation 1: adapt mesh from the magnetic flux field in the equilibrium (A)

		2: adapt mesh from the estimated error in the solution field (B) 3: run both (A) and (B)
adapt_control	1	
adapt_hmin	0.001	
adapt_hmin_rel	0.5	
adapt_hmax	0.1	
adapt_hmax_rel	2	
adapt_ke	0	
adapt_pack_factor	0.02	
adapt_psin_vacuum	0	treat the entire vacuum region as having that value of psin
adapt_psin_wall	0	treat the entire wall as having that value of psin
adapt_smooth	2./3.	
adapt_target_error	0.0001	
iadapt_max_node	10000	
iadapt_ntime	0	
iadapt_order_p	3	
iadapt_pack_rationals	0	number of mode-rational surfaces to pack mesh around
iadapt_removeEquiv	0	
iadapt_useH1	0	
iadapt_writesmb	1	if 1, write the adapted mesh in "ts*N*-adapted.smb", *N*=time step
iadapt_writevtk	0	if 1, write the initial and adapted mesh in vtk format The initial mesh before adaptation is written in folder "ts0-initial" The adapted mesh is written in folder "ts*N* -adapted", *N*=time step
adapt_coil_delta	0.	Width parameter for packing mesh around coil locations must include files adapt_coil.dat and adapt_current.dat
adapt_pellet_length	0.	Length of pellet path to pack mesh along
adapt_pellet_delta	0.	Width parameter for packing mesh along pellet path
adapt_factor	1	
adapt_qs(array)	0	Safety vactor values (up to 32) about which to adapt. Literally, the normalized flux used for adaptation gets hacked so that it's close to unity near the specified safety factors

See Section 16.2 for the details of mesh adaptation.

8.12 Numerical Options

jadv	1	1: use Del*(psi) (ie, current) eqn. instead of psi eqn.
ivform	1	V = R^J Grad(U)XGrad(phi) + R^K V Grad(phi) + R^L Grad(chi) 0: J=0, K=0, L=0; 1: J=2, K=2, L=-2
int_pts_main	25	must be consistent with MAX_PTS at compile time
int_pts_aux	25	MAX_PTS must be GE int_pts_tor*int_pts_nnn for 3D
int_pts_diag	25	
int_pts_tor	5	
max_ke	1.	value of ke at which linear runs are rescaled
equilibrate	0	1: scale trial functions so L2 norms=1

regular	0.	regularization constant in chi equation
iset_pe_floor	0	1: do not let pe drop below pe_floor
pe_floor	0	minimum allowed value for pe when iset_pe_floor=1
iprecompute_metric	0	1: precompute full metric tensor

8.13 Input

iread_eqdsk	0	1: read geqdsk equilibrium file (see Section 3.1) 2: read psi from geqdsk, but uses analytic profiles for p and F 3: read profiles from geqdsk, but not the eqdsk psi
iread_dskbal	0	1: read dskbal equilibrium
iread_jsolver	0	1: read equilibrium file "fixed" from jsolver (see Section 3.2)
iread_omega	0	nonzero: reads in rotation profile
iread_omega_ExB	0	read ExB rotation (same options as ireead_omega)
iread_omega_e	0	read electron rotation (same options as iread_omega)
iread_ne	0	nonzero: read in electron density profile
iread_te	0	nonzero: read in temperature profile
iread_p	0	1: read pressure profile from profile_p
iread_neo	0	1: read velocity profiles from NEO output
ineo_subtract_diamag	0	1: subtract diamag term from input vel when reading neo velocity
iread_heatsource	0	1: read heat source profile (psi normalized) scaled by ghs_rate
iread_particlesource	0	1: read particle source profile (psi normalized) scaled with pellet_rate
iread_f	0	if 1, read R BT from file

8.14 Output

iprint	0	1,2: additional debug output
ntimepr	5	number of time steps per field output
ntimers	0	number of time steps per restart output (defaults to ntimepr)
iglobalout	0	1: write global restart files
iglobalin	0	1: read global restart files
iwrite_adios	0	1: write adios restart files (no longer supported)
ifout	-1	1: output f field
idouble_out	0	1: use double-precision floating in output hdf5 files
itemp_plot	0	1: output vdotgradt, deldotq_perp, deldotq_par, eta_jsq
ibdgp	0	option for plotting partial terms for bdgp plot (1) $[\psi,\Phi]$, (2) (f',Φ), (3) $-R^{-2}F\Phi'$
iveldif	0	option for plotting partial terms for veldif plot

(1) $|\psi, U| + R(U, f')$ (4) $R^{-1}\Phi' + R^{-1}FU'$

(2) $R^{-3}(\psi, \chi) + R^{-2}|\chi, f'|$ (5) $|\psi, U| + R(U, f') - R^{-1}FU'$

(3) $R^{-1}\Phi'$ (6) $R^{-2}|\chi, f'|$

icalc_scalars	1	1: calculate scalar diagnostics
ike_only	0	1: only calculate ke scalar diagnostic
ike_harmonics	0	number of toroidal harmonics of kinetic energy to be calculated for diagnostics
ibh_harmonics	0	number of toroidal harmonics of magnetic energy to be calculated for diagnostic
irestart	0	0: start from time step 0 1: normal restart 3: in 2D complex run, read a 2D real restart file
irestart_factor	1	if >1, multiply the number of planes in 3D run
iread_hdf5	1	restart runs from hdf5 file (now default) Also, best to have idouble_out=1
irestart_slice	-1	If set to a value (n), and iread_hdf5=1, will restart from a time-step other than the last one
iread_adios	0	1: restart using adios (no longer supported)
itimer	0	1: output internal timer data
iwrite_transport_coeffs	1	output transport coefficient fields
iwrite_aux_vars	1	output auxiliary variable fields

8.15 Diagnostics

xray_detector_enabled	0	1: enable xray detector
xray_r0	0.	R coordinate of xray detector
xray_phi0	0	phi coordinate of xray detector
xray_z0	0	Z coordinate of xray detector
xray_theta	0	angle of xray detector chord (degrees)
xray_sigma	1.	spread of xray detector chord (degrees)

imag_probes: number of mag probes

mag_probe_x(i):	0	R-coordinate of mag probe i
mag_probe_phi(i):	0	phi-coordinate of mag probe i
mag_probe_z(i):	0	Z-coordinate of mag probe i
mag_probe_nx(i):	0	R-component of normal vector of mag probe i
mag_probe_nphi(i)	0	phi-component of normal vector of mag probe i
mag_probe_nz(i)	0	Z-component of normal vector of flux loop i

These values can be plotted using the plot_mag_probe s IDL routine. (see Section 5.1.2)

iflux_loops:	0	number of flux loops
flux_loop_x(i):	0	R-coordinate of flux loop i
flux_loop_z(i):	0	Z-coordinate of flux loop i

These values can be plotted using the plot_flux_loops IDL routine. (see Section 5.1.2)

ifixed_temax	0	if nonzero, temax evaluated at (xmag0,0,zmag0)

8.16 Sources/Sinks

! beam source

Ibeam	0	
		1: include neutral beam particle, energy, and momentum source
		$S = \frac{nb_n}{4r\pi^2 nb_{dr}^2}\exp-\left[\left(r-nb_r\right)^2+\left(z-nb_z\right)^2\right]/2nb_{dr}^2$
		2: include only particle and energy source (no torque)
		3: include only energy source (no torque or particle source)
		4: include only momentum and energy (no particle source)
		5: include only momentum (no energy or particle source)
beam_x	0	R-coordinate of beam center (m)s
beam_z	0	Z-coordinate of beam center (m)
beam_v	1.e4	beam voltage (in volts)
beam_rate	0.	ions/second deposited by beam
beam_dr	0.1	dispersion of beam deposition
beam_dv	100.	dispersion of beam voltage (in volts)
beam_fracpar	1	cos of beam angle relative to parallel (for momentum source)

$$nb_n = \text{beam\_rate}\frac{t_0}{n_0 l_0^3} \qquad nb_r = \text{beam\_x}\frac{100}{l_0} \qquad nb_z = \text{beam\_z}\frac{100}{l_0}$$

$$nb_{dr} = \text{beam\_dr}\frac{100}{l_0} \qquad nb_v = \left[\frac{2(\text{beam\_zeff})\times e_c \times(\text{beam\_v})10^8}{cM_i}\right]^{1/2}/V_0$$

$$e_c = 4.8032\times10^{-10} \quad c = 2.9979\times10^{+10} \quad M_i = (\text{ion\_mass})\times1.6726\times10^{-24}$$

$$l_0 = 100 \quad b_0 = 10^4 \quad n_0 = 10^{14} \quad V_0 = b_0/(4\pi M_i n_0)^{1/2} \quad t_0 = l_0/V_0$$

$$\dot{n} = \cdots + S(r,z)$$

$$R^2\rho\dot{\omega} = \mu\Delta^*\left(R^2\omega\right)\cdots + R\,S(r,z)\left[nb_v - R\omega\right]$$

$$\dot{p} = \cdots + \tfrac{1}{2}S(r,z)\left[\left(nb_v^2 + nb_{dv}^2\right) - 2\,nb_v\,R\omega + R\omega^2\right]$$

vloop	0.	initial loop voltage. NOTE: to change vloop at restart time must have control_type=-1
tcur	0.	target (scaled) plasma current for current control: $\mu_0 I_P$
tcuri	0	if tcuri .ne. tcurf, the target current is a function of time
tcurf	0	tcur = tcuri + (tcurf-tcuri)×.5×(1+tanh((t−tcur_t0)/tcur_tw))
tcur_t0	0	
tcur_tw		

control_type	0	current control type:
	-1:	no current control. Constant vloop applied
	0:	old current control algorithm (not recommended)
	1:	standard PID control with the following control parameters
control_p	0.	proportional control coefficient
control_i	0.	integral control coefficient
control_d	0.	derivative control coefficient

! density source

ipellet

0 density source if non-zero (3D part equals 1 for 2D simulations)
Double-digit values have volume integrals normalized to 1
Make negative for initial perturbation only

With $G_{2D} = \frac{1}{2\pi R V_p^2} \exp\left[-\frac{(R-R_p)^2+(Z-Z_p)^2}{2V_p^2}\right]$

1: $S = G_{2D} \times \frac{R}{\sqrt{2\pi}V_t} \exp\left[-\frac{RR_p\left(1-\cos(\varphi-\varphi_p)\right)}{V_t^2}\right]$

2: $S = \text{den0} \times \left(\max(p, p_{edge})/p_0\right)^{\exp n}$ 2D and 3D

3: Gaussian source proportional to pressure

$S = p \times G_{2D} \times \frac{R}{\sqrt{2\pi}V_p} \exp\left[-\frac{RR_p\left(1-\cos(\varphi-\varphi_p)\right)}{V_p^2}\right]$

4. Same distribution as ipellet=1 in 3D

$S = \sqrt{2\pi} R V_p \times G_{2D} \times \frac{1}{2\pi V_p V_t} \exp\left[-\frac{RR_p\left(1-\cos(\varphi-\varphi_p)\right)}{V_t^2}\right]$

11: Same as #1 but numerically normalized

12: Spherical, Cartesian Gaussian; numerically normalized

2D: $S = RG_{2D}$

3D: $S = \exp\left[-\frac{(R\cos\varphi - R_p\cos\varphi_p)^2+(R\sin\varphi - R_p\sin\varphi_p)^2+(Z-Z_p)^2}{2V_p^2}\right]$

13: Axisymmetric, toroidal Gaussian; numerically normalized

2D & 3D: $S = G_{2D}$

14: Toroidal distribution is a blend of a von Mises and Cauchy distribution

$$S = G_{2D} \times \frac{R}{\sqrt{2\pi}V_t} \left\{ (1-f_c) \times \exp\left[-\frac{RR_p\left(1-\cos(\varphi-\varphi_p)\right)}{V_t^2}\right] + f_c \times \frac{\cosh\left(\frac{V_t}{\sqrt{RR_p}}\right) - \cos(\varphi_p)}{\cosh\left(\frac{V_t}{\sqrt{RR_p}}\right) - \cos(\varphi-\varphi_p)} \right\}$$

Here: f_c = cauchy_fraction (default: 0.)

15. Toroidal von-Mises distribution with angular half-width

$S = G_{2D} \times \exp\left[\cos(\varphi-\varphi_p)/V_p^2\right]$

Pellet_var_tor radians for ipellet=15, distance otherwise

ipellet_z	0	Atomic number of pellet (0 for main-ion species)
ipellet_abl	0	Turn on pellet ablation Recommended: double-digit ipellet for particle conservation 1: include ablation model [Parks NF94] calibrated on DIII-D (Li) 2. Include new ablation model [Parks,2015] for small pellets (Li) 3. Parks model developed 6/20/2017 (for Neon)
temin_abl	0	Minimum temperature at which ablation turns on
iread_pellet	0	0: Single pellet defined by scalar parameters below 1: Read pellet.dat, with one row per pellet 13 space-delimited columns are: pellet_r pellet_phi pellet_z pellet_rate pellet_var … pellet_var_tor pellet_velr pellet_velphi pellet_velz … r_p cloud_pel pellet_mix cauchy_fraction
pellet_r	0.	initial radial position of pellet (R_p)
pellet_phi	0	initial toroidal position of the pellet (φ_P)
pellet_z	0.	initial vertical position of pellet (Z_p)
pellet_rate	0.	Density source is pellet_rate × S as defined by ipellet; If ipellet_abl.ne.0, ablation routines define pellet_rate
pellet_var	1.	poloidal spatial dispersion of pellet source (V_p)
pellet_var_tor	0	toroidal spatial dispersion of pellet source (V_t) If zero, pellet_var_tor = pellet_var
pellet_velr	0	Initial radial velocity of the pellet[2]
pellet_velphi	0	Initial toroidal velocity of pellet[2]
pellet_velz	0	Initial vertical velocity of the pellet[2]
r_p	1.e-3	Initial pellet radius
cloud_pel	1.	Parameter used to change the width of the density source if ablating. In this case, pellet_var = cloud_pel * r_p
pellet_mix	0	Molar fraction of diatomic main-ion molecules in pellet (e.g., D_2)
irestart_pellet	0	will read all pellat attributes from the *h5 file at restart
	1	will read the following from C1input at restart pellet_rate pellet_rate_D2 pellet_var_tor pellet_var cloud_pel pellet_mix cauchy_fraction Other pellet parameters from the *.h5 file
abl_fac	1.0	factor to multiply ablation rate from predefined formulae

[2] Note that pellets are ballistic; velocities are in internal (normalized units). They're converted to Cartesian coordinates and then held constant

n _control_type	-1	no density control type -1: density control. 0: old density control algorithm 1: standard PID control with the following control parameters
n_target	1.	target density
n_control_p	0.	proportional feedback constant
n_control_i	0.	integral feedback constant
n_control_d	0.	derivative feedback constant

igaussian_heat_source	0	1: include Gaussian heat source
ghs_x	0.	R coordinate of Gaussian heat source
ghs_z	0.	Z coordinate of Gaussian heat source
ghs_rate	0.	amplitude of Gaussian heat source
ghs_var	1.	variance of Gaussian heat source
ghs_phi	0	phi coordinate of Gaussian heat source
ghs_var_tor	0	toroidal variance of Gaussian heat source
ionization	0	
ionization_rate	0.	
ionization_temp	0.01	
ionization_depth	0.01	

! current drive source $\dot{\psi} = \ldots + \eta(\Delta^* \psi - J_{cd})$ $J_{cd} = J_0 \exp[-(R-R_0)^2 - (Z-Z_0)^2]/W_{CD}^2 - \Delta_{CD}$

icd_source	0	1: include current drive
J_0cd	0	magnitude of Gaussian: J_0
R_0cd	0	R-coordinate of maximum: R_0
Z_0cd	0	Z-coordinate of maximum: Z_0
W_cd	0	width of Gaussian: W_{CD}
delta_cd	0	shift of Gaussian: Δ_{CD}

isink	0	density sink
sink1_x	0.	
sink1_z	0.	
sink1_rate	0.	
sink1_var	1.	
sink2_x	0.	
sink2_z	0.	
sink2_rate	0.	
sink2_var	1	

idenfloor	0	1: density in vacuum pegged to den_edge
alphadenfloor	0	multiplier of (den_edge – den) must be .lt. 1/DT

! Poloidal Momentum source

ipforce	0	1: include poloidal momentum source

$$\mathbf{F} = f\left(\tilde{\psi}\right)\nabla\psi\times\nabla\varphi \qquad \tilde{\psi}\equiv\left(\psi-\psi_0\right)/\left(\psi_b-\psi_0\right)$$

$$f\left(\tilde{\psi}\right) = a\left(1-\tilde{\psi}\right)^N\left[\frac{\delta^2}{\left(\tilde{\psi}-\psi_c\right)^2+\delta^2}\right]$$

dforce	0.	δ
xforce	0.	ψ_c
nforce	0	N
aforce	0	a

2: include Luca Guzzatto form of momentum source

$$S = a\hat{\psi}^2 n\cos\left(\frac{\theta}{2}\right)\frac{\nabla\psi\times\nabla\varphi}{\left|\nabla\psi\times\nabla\varphi\right|} \qquad \psi_{\min}<\psi<\psi_{\max}$$

$$\hat{\psi} = 1-\frac{\psi-\psi_{\min}}{\psi_{\max}-\psi_{\min}}$$

iheat_sink	0	if 1, special heat sink for itaylor=27
coolrate	0	S = coolrate*(pedge – p) for iheat_sink=1
iarc_source	0	1: density source due to halo current
arc_source_alpha	0	parameter for iarc_source
arc_source_eta	.01	parameter for iarc_source

8.17 Resistive Wall

eta_vac	1	resistivity of vacuum region
eta_wall	0.001	resistivity of conducting wall regions
eta_wallRZ	0.001	poloidal resistivity of wall region (if different from eta_wall)
iwall_breaks	0	number of wall breaks
eta_break	1	resistivity of wall break (array)
wall_break_phimax	0	max phi coordinate for break (array)
wall_break_phimin	0	min phi coordinate for break (array)
wall_break_xmax	0	max x coordinate for break (array)
wall_break_xmin	0	min x coordinate for break (array)
wall_break_zmax	0	max z coordinate for break (array)
wall_break_zmin	0	min z coordinate for break (array)
iwall_regions	0	number of resistive wall regions
wall_region_eta()	1.e-3	resistivity of each wall region
wall_region_etaRZ()	1.e-3	poloidal resistivity (if different from wall_region_eta)
wall_region_filename()	-	file name with wall contour points
eta_rekc	0	resistivity of runaway electron killer coil (REKC)
ntor_rekc	0	toroidal mode number of REKC

mpol_rekc	0	poloidal mode number of REKC
phi_rekc	0	toroidal angle of fixed point of REKC
theta_rekc	0	poloidal angle of fixed point of REKC
rzero_rekc	0	R0 for computing theta of REKC
zzero_rekc	0	Z0 for computing theta of REKC
isym_rekc	0	if non-zero, coil is double helix with (+,-) mpol_rekc

8.18 Miscellaneous

gam	5/3	ratio of specific heats
db	0	ion skin depth (overrides db_fac)
db_fac	0	factor multiplying physical value of ion skin depth
mass_ratio	0	ratio of ion to electron mass
lambdae	0	lambdae
z_ion	1.	Z-effective
ion_mass	1.	ion mass in units of m_p
lambda_coulomb	17.	Coulomb logarithm
thermal_force_coeff	0	coefficient of thermal force
ntor	0	toroidal mode number for 3D (complex) linear
mpol	0	poloidal mode number for certain test problem initialization

8.19 Deprecated

ipartitioned	0	
igs_method	-1	now deprecated.
		Formerly defined as follows:
		1: use node-based method (fastest, least accurate)
		2: use element-based method and calculate p from input p profile
		3: use element-based method and calculate p from input p’ profile
ibform	-1	
delta_wall	1.	wall thickness

8.20 Trilinos Options

drop_tolerance	0	ILU drop tolerance
graph_fill	0	graph fill level
ilu_fill_level	1	ILU fill level
ilu_omega	1	relaxation parameter for rILU
krylov_solver	gmres	Krylov solver
poly_ord	1	polynomial order for certain preconditioners
preconditioner	dom_decomp	preconditioner
sub_dom_solver	ilu	subdomain solver in preconditioner
subdomain_overlap	1	subdomain overlap

8.21 Simple Radiation Model

iprad	0	1: call Prad radiation module with one impurity species $P_{loss} = n_e n_D L_D(T_e) + n_e n_Z L_Z(T_e)$ Cooling rate of deuterium is $L_D = 5.35\times10^{-37} T_e^{1/2}[\mathrm{keV}]\mathrm{W}\cdot\mathrm{m}^3$ $L_Z(T_e)$ taken from Post, et al, *Atomic data and nuclear data tables*, **20** pp. 397-439,(1977)
prad_fz	1	density of impurity species, as fraction of n_e: $n_Z = \mathrm{prad\_fz}\times n_e$
prad_z	1	Z of impurity species (Z=6(C), 18(Argon), 26(Fe) are available
iread_prad	0	1: Read impurity density from profile_nz (units of $10^{20}/\mathrm{m}^3$)

8.22 KPRAD Radiation Model

ikprad	0	1: KPRAD module with one impurity species
kprad_z	1	Z of impurity species in KPRAD module..Presently available: 2 Helium 4 Beru;;oi, 6 Carbpm 10 Neon 18 Argon
kprad_fz	0	Density of neutrals as fraction of ne
kprad_nz	0	Density of neutral impurities
kprad_nemin	1e-12	Minimum (normalized) electron density for KPRAD evolution
kprad_temin	2e-7	Minimum (normalized) electron temp. for KPRAD evolution
ikprad_max_dt	0	Set max time step for KPRAD ionization 0: MHD time step dt 1: **RECOMMENDED**: dt/(kprad_z + 1) (ensures evolution through all charge states)
ikprad_evolve_internal	0	Update local temperature during KPRAD subcycling 0: Te fixed before subcycling

		1: **RECOMMENDED** Local ne and Te used for KPRAD ionization/radiation updated during subcycling each KPRAD time step due to density and energy changes
ikprad_evolve_neutrals	0	Determine how KPRAD neutrals evolve spatially 0: Neither advect nor diffuse 1: **RECOMMENDED** Advect and diffuse like other charge states 2: Diffuse but do not advect
Ikprad_min_option	1	Determines how KPRAD behaves below minimum density /temperature (ikprad_nemin/kprad_temin) 1: No radiation/ionization/recombination (based on ne/te before sybcycling) 2**: RECOMMENDED** Recombination but no radiation/ionization (based on ne/Te during subcycling) 3: No radiation/ionization/recombination (based on ne/Te during subcycling) NOTE: 1 & 3 behave the same if ikprad_evolve_internal=0)
iread_lp_source	0	1: Read impurity source from Lagrangian Particle code cloud.txt (UNDER DEVELOPMENT)

8.23 Stellarator geometry

type_ext_field	0	1: reads either VMEC or MGRID file
file_ext_field		(string) FIELDLINES/MGRID file. Must start with 'fieldlines' or 'mgrid'
iread_vmec	0	1: read VMEC file to determine geometry. Must be =1 for stellarator
vmec_filename		(string) VMEC output .nc file
bloat_factor	0	Free boundary only: Scale factor to bloat computational boundary using input geometry.
bloat_distance	0	Free boundary only: Distance to expand computational boundary from input geometry
igeometry	0	1: Must be set to use stellarator version
nperiods	1	Number of field periods (stellarator geometry). Note nplanes should be equal to at least 2*(# toroidal modes per field period)*nperiods
ifull_torus	0	0: Solve on one field period 1: Solve on full torus
nzer_factor	-1	(integer) Scale factor for resolution of Zernike polynomial (used for interpolation of VMEC) -1: n_zer=2*mpol (fixed boundary) and 1*mpol (free boundary) Otherwise: n_zer=nzer_factor*mpol where mpol is poloidal resolution in VMEC.
nzer_manual	-1	(int) Resolution of Zernike polynomial (mainly for testing).

Note: Overridden by nzer_factor=-1 if less than the corresponding value.

9.0 Relation between itor=1 and itor=0

The vector fields for itor=1 and itor=0 are defined as follows:

itor=1	itor=0
$\mathbf{B} = \nabla\psi \times \nabla\varphi - \nabla_\perp f' + F\nabla\varphi$	$\mathbf{B} = \nabla\psi \times \hat{y} - \nabla_\perp f' + F\hat{y}$
$\mathbf{V} = R^2\nabla U \times \nabla\varphi + \omega R^2\nabla\varphi + R^{-2}\nabla_\perp\chi$	$\mathbf{V} = \nabla U \times \hat{y} + \omega\hat{y} + \nabla_\perp\chi$

Note that this implies that when comparing a itor=1 run with major radius R_0 with a itor=0 run with rzero:

1. When applying a loop voltage to a configuration with a given resistivity, the voltage is applied in such a way that the total plasma current and wall current 'IP,IW' and current density 'jy' should be comparable for itor=0,1 if the cross section is the same and rzero=R_0
2. The IDL quantity "jphi"(itor=1) should be compared with jphi*rzero (itor=0)
3. The velocity variables U, ω, χ (itor=1) should be compared with U/rzero, ω/rzero, $\chi \times (\text{rzero})^2$

10. Dimensionless Scaling

Herein we consider the scale factors that make the internal equations dimensionless. Consider first the momentum equation:

$$nM_i\frac{d\mathbf{v}}{dt}+\nabla p=\mathbf{J}\times\mathbf{B}+\nu\nabla^2\mathbf{v}$$

$$n_0\tilde{n}M_i\frac{v_0}{t_0}\frac{d\tilde{\mathbf{v}}}{d\tilde{t}}+\frac{p_0}{\ell_0}\tilde{\nabla}\tilde{p}=J_0B_0\tilde{\mathbf{J}}\times\tilde{\mathbf{B}}+\frac{\nu_0v_0}{\ell_0^2}\tilde{\nu}\tilde{\nabla}^2\tilde{\mathbf{v}}$$

$$\frac{\ell_0^2}{t_0^2}\frac{\mu_0n_0M_i}{B_0^2}\tilde{n}\frac{d\tilde{\mathbf{v}}}{d\tilde{t}}+\frac{\mu_0p_0}{B_0^2}\tilde{\nabla}\tilde{p}=\tilde{\mathbf{J}}\times\tilde{\mathbf{B}}+\frac{\nu_0v_0\mu_0}{B_0^2\ell_0}\tilde{\nu}\tilde{\nabla}^2\tilde{\mathbf{v}}$$

Next, define:

$$t_0=\left[\frac{\mu_0n_0M_i}{B_0^2}\right]^{1/2}\ell_0,\qquad p_0=\frac{B_0^2}{\mu_0},\qquad J_0=\frac{B_0}{\mu_0\ell_0},\qquad v_0=\frac{\ell_0}{t_0},\qquad \nu_0=\frac{B_0^2t_0}{\mu_0}=\frac{n_0M_i}{t_0}\ell_0^2$$

Then, the momentum equation becomes simply: $\tilde{n}\frac{d\tilde{\mathbf{v}}}{d\tilde{t}}+\tilde{\nabla}\tilde{p}=\tilde{\mathbf{J}}\times\tilde{\mathbf{B}}+\tilde{\nu}\tilde{\nabla}^2\tilde{\mathbf{v}}$

The magnetic field evolution equation (electron form) is:

$$\frac{\partial\mathbf{B}}{\partial t}=\nabla\times\left[\mathbf{v}\times\mathbf{B}-\eta\mathbf{J}-\frac{1}{ne}[\mathbf{J}\times\mathbf{B}-\nabla p_e]-\frac{m_e}{ne^2}\dot{\mathbf{J}}\right],\qquad \frac{\partial\tilde{\mathbf{B}}}{\partial\tilde{t}}=\tilde{\nabla}\times\left[\tilde{\mathbf{v}}\times\tilde{\mathbf{B}}-\frac{\eta_0t_0}{\mu_0\ell_0^2}\tilde{\eta}\tilde{\mathbf{J}}-d_i[\tilde{\mathbf{J}}\times\tilde{\mathbf{B}}-\nabla\tilde{p}_e]-d_e^2\dot{\tilde{\mathbf{J}}}\right]$$

$$\eta_0=\frac{\mu_0\ell_0^2}{t_0}=B_0\ell_0\left[\frac{\mu_0}{n_0M_i}\right]^{1/2}\qquad d_i=\frac{1}{\ell_0}\left[\frac{M_i}{\mu_0n_0e^2}\right]^{1/2}=\frac{c}{\ell_0\omega_{pi}}\qquad d_e^2=\frac{c^2}{\ell_0^2\omega_{pe}^2}$$

The ion form of the magnetic evolution equation is:

$$\frac{\partial\mathbf{B}}{\partial t}=\nabla\times\left[\mathbf{v}\times\mathbf{B}-\eta\mathbf{J}-\frac{1}{ne}[nM_i\left(\frac{\partial\mathbf{v}}{\partial t}+\mathbf{v}\bullet\nabla\mathbf{v}\right)+\nabla p_i]\right],\qquad \frac{\partial\tilde{\mathbf{B}}}{\partial\tilde{t}}=\tilde{\nabla}\times\left[\begin{array}{l}\tilde{\mathbf{v}}\times\tilde{\mathbf{B}}-\frac{\eta_0t_0}{\mu_0\ell_0^2}\tilde{\eta}\tilde{\mathbf{J}}-\underbrace{\frac{1}{\ell_0}\left[\frac{M_i}{\mu_0n_0e^2}\right]^{1/2}}_{d_i}\left(\frac{\partial\mathbf{v}}{\partial t}+\mathbf{v}\bullet\nabla\mathbf{v}\right)\\ \underbrace{+\frac{1}{\ell_0}\left[\frac{M_i}{\mu_0n_0e^2}\right]^{1/2}}_{d_i}\nabla p_i\end{array}\right]$$

$$\eta_0=\frac{\mu_0\ell_0^2}{t_0}=B_0\ell_0\left[\frac{\mu_0}{n_0M_i}\right]^{1/2}\qquad d_i=\frac{1}{\ell_0}\left[\frac{M_i}{\mu_0n_0e^2}\right]^{1/2}=\frac{c}{\ell_0\omega_{pi}}$$

Next, consider the energy equation:

$$n\frac{dT_e}{dt} = \cdots + (\gamma - 1)\left[nQ_\Delta (T_i - T_e) + \eta J^2 + \nabla \bullet \kappa \nabla T_e \right], \qquad Q_\Delta = \frac{3e^2 n}{M_i}\eta$$

$$\frac{n_0 T_0}{t_0}\tilde{n}\frac{d\tilde{T}_e}{d\tilde{t}} = \cdots + (\gamma - 1)\left[n_0 T_0 Q_{\Delta 0}\tilde{n}\tilde{Q}_\Delta (\tilde{T}_i - \tilde{T}_e) + \eta_0 J_0^2 \tilde{\eta}\tilde{J}^2 + \frac{\kappa_0 T_0}{\ell_0^2}\tilde{\nabla} \bullet \tilde{\kappa}\tilde{\nabla}\tilde{T} \right]$$

$$\tilde{n}\frac{d\tilde{T}_e}{d\tilde{t}} = \cdots + (\gamma - 1)\left[Q_{\Delta 0} t_0\, \tilde{n}\tilde{Q}_\Delta (\tilde{T}_i - \tilde{T}_e) + \frac{t_0}{n_0 T_0}\eta_0 J_0^2 \tilde{\eta}\tilde{J}^2 + \frac{t_0}{n_0 \ell_0^2}\kappa_0 \tilde{\nabla} \bullet \tilde{\kappa}\tilde{\nabla}\tilde{T} \right]$$

$$T_0 = \frac{t_0}{n_0}\eta_0 J_0^2 = \frac{t_0}{n_0}\frac{\mu_0 \ell_0^2}{t_0}\left(\frac{B_0}{\mu_0 \ell_0} \right)^2 = \frac{B_0^2}{n_0 \mu_0} = \frac{p_0}{n_0} \qquad \kappa_0 = \frac{n_0 \ell_0^2}{t_0} = \ell_0 B_0 \left[\frac{n_0}{\mu_0 M_i} \right]^{1/2}$$

$$t_0 Q_{\Delta 0} = E = t_0 \left(\frac{3e^2 n_0}{M_i}\eta_0 \right) = \mu_0 \ell_0^2 \frac{3e^2 n_0}{M_i}, \qquad Q_{\Delta 0} Q_\Delta = \frac{3e^2 n}{M_i}\eta = \left(\frac{3e^2 n_0}{M_i}\eta_0 \right)\tilde{n}\tilde{\eta}$$

Temperature and resistivity:

$$T_e(keV) = \frac{\tilde{p}_e}{\tilde{n}} x 49.66 \left[\frac{B_0^2(T)}{n_0(20)} \right] \qquad \eta_{neo} \sim 10^{-7} \times \left[T(keV) \right]^{-3/2} = 2.8 \times 10^{-10} \times \left[\frac{\tilde{p}_e}{\tilde{n}} \right]^{-3/2} \left[\frac{B_0^2(T)}{n_0(20)} \right]^{-3/2}$$

Ion skin depth: $d_i = \frac{1}{\ell_0}\left[\frac{M_i}{\mu_0 n_0 e^2} \right]^{1/2} = .0227 \frac{1}{\ell_0 [m]} \left[n_0[20] \right]^{-1/2}$

Thermal conductivity: $\kappa_0 = 2.18 \times 10^{26} \ell_0 [m] B_0 [T] n_0^{1/2} [20]$

If we input the magnetic field in Tesla and all the lengths in meters, then $B_0 = \ell_0 = 1$, and the only quantity we need to scale with is the density n_0 (in units of 10^{20}).

11. Grad-Shafranov Solver

The poloidal flux function is initialized to solve the plasma equilibrium equation:

$$J_\varphi(R,\psi) = \frac{\partial}{\partial R}\frac{1}{R}\frac{\partial \psi}{\partial R} + \frac{\partial}{\partial Z}\frac{1}{R}\frac{\partial \psi}{\partial Z} = -\left(Rp'(\psi) + \frac{1}{R} gg'(\psi) \right) \tag{12.1}$$

Here $p(\psi)$ is the plasma pressure and $g(\psi)$ is the toroidal field function so that $g(\psi)/R$ is the toroidal field strength, and prime denotes a derivative with respect to ψ, the solution. When importing an equilibrium solution from a file, the functions $p(\psi)$ and $g(\psi)$ are read from the file. For a stand-alone calculation, the following functional forms are used. Let $s \equiv (\psi - \psi_0)/\Delta\psi$ be the normalized poloidal flux, where $\Delta \equiv \psi_L - \psi_0$ (the difference between ψ at the limiter, or separatrix, and the magnetic axis). The pressure is defined as the following polynomial in s:

$$p(s) = p_0 \begin{bmatrix} 1 + p_1 s + p_2 s^2 - (20 + 10p_1 + 4p_2)s^3 + (45 + 20p_1 + 6p_2)s^4 \\ -(36 + 15p_1 + 4p_2)s^5 + (10 + 4p_1 + p_2)s^6 \end{bmatrix} + p_e$$

The toroidal field function is of the following form:

$$\frac{1}{2} g^2(s) = \frac{1}{2} g_0^2 + \gamma_2 G_2(s) + \gamma_3 G_3(s) + \gamma_4 G_4(s)$$

where

$$\begin{aligned} G_2(s) &= s - 10s^3 + 20s^4 - 15s^5 + 4s^6 \\ G_3(s) &= s^2 - 4s^3 + 6s^4 - 4s^5 + s^6 \\ G_4(s) &= 1 - 20s^3 + 45s^4 - 36s^5 + 10s^6 \end{aligned}$$

The pressure is thus specified by the three input variables: p_0, p_1, p_2. The three constants in the toroidal field function definition are used to prescribe the total plasma current I_p, the safety factor on axis q_0, and the measure of the slope of the current density near the axis J_ψ. The constant g_0 is the value of the toroidal field function due to the external fields. It is seen that near the magnetic axis, $(s = 0)$ the pressure and toroidal field functions have the form:

$$\begin{aligned} p' &= p_0 [p_1 + 2p_2 s]/\Delta\psi \\ G_2'(s) &= 1/\Delta\psi + \cdots \\ G_3'(s) &= 2s/\Delta\psi + \cdots \\ G_4(s) &= 0 \end{aligned}$$

Terms of order s^2 and higher have been dropped. We write the current density near the axis as

$$J_\varphi\left(R_0,\psi\right)=J_0+\frac{8g_0J_\psi}{R_0^2}s\,,$$

and note that the central current density and the central safety factor are related by: $J_0=2g_0/R_0^2q_0$. Evaluating Eq. (12.1) near the magnetic axis then gives the following expressions for γ_2 and γ_3.

$$\gamma_2=-\left(R_0^2p_0p_1+2g_0\Delta\psi/R_0q_0\right)$$
$$\gamma_3=-\left(4g_0J_\psi\Delta\psi/R_0+R_0^2p_0p_2\right)$$

Here, R_0 is the major radius of the magnetic axis and the integrals over the plasma area are given by:

$$I_1=\int dA\,Rp'\qquad I_k=\int dA\,\frac{1}{R}G_k'$$

The expression for γ_4 constrains the total plasma current to be equal to the input variable I_p.

$$\gamma_4=-\left(-I_P+\gamma_2I_2+\gamma_3I_3+I_1\right)/I_4$$

12. Grad-Shafranov Solver with Toroidal Flow

When toroidal flow is included, the pressure is no longer solely a function of ψ but is given by:

$$p(R,\psi) = p(\psi)\exp\left[\alpha(\psi)\left(\frac{R^2 - R_0^2}{R_0^2}\right)\right]$$

The function $\alpha(\psi)$, proportional to the square of the angular velocity is parameterized by:

$$\tilde{\alpha} = \alpha_0 + \alpha_1 s + \alpha_2 s^2 + \alpha_3 s^3$$

For iscale_rot_by_p = 0: $\alpha = \tilde{\alpha} \times n(\psi) / p(\psi)$,

For iscale_rot_by_p = 1: $\alpha = \tilde{\alpha}$

For iscale_rot_by_p = 2: $\alpha = \left[\alpha_0 + \alpha_1 e^{-\left[(\psi-\alpha_2)/\alpha_3\right]^2}\right] \times n(\psi) / p(\psi)$

In all cases, the angular velocity is then determined by:

$$\omega = \left[\frac{2\alpha}{R_0^2}\frac{p(\psi)}{n(\psi)}\right]^{1/2}$$

13. Accessing TRANSP Data

To search for information on which TRANSP runs were performed for a particular shot on the PPPL unix system (NSTX shot 124379), you can type:

“ls /p/transpgrid/inf_new/NSTX/*/124379*.INF > lsout”

Then, individual files (such as …./124379A32TR.INF) can be examined with a text editor.

Data files can be obtained from: /p/transparch/result/NSTX/*/124379*.CDF

rplot can then be run with the following input options:

rplot
T
Y
NSTX.07
N
124379A30
Q

14. Suggested Boundary Conditions

As of 5/17/2015 we recommend:

inoslip_pol = 0 (for time-independent runs)
inoslip_pol=1 (for time-dependent runs)
inoslip_tor = 1
inonormalflow = 1
inocurrent_norm = 0
inocurrent_tor = 0
inocurrent_pol = 1
iconst_bz = 0 (may need to be =1 for 2F runs if I is perturbed at boundary
iconst_bn = 1
iconst_p = 1
iconst_n = 1

15. Magnetic Boundary Conditions

Definitions:

$$\mathbf{A} = R^2\nabla\varphi\times\nabla f + \psi\nabla\varphi - F_0\ln R\hat{Z}$$
$$\mathbf{B} = \nabla\psi\times\nabla\varphi - \nabla_\perp f' + F\nabla\varphi$$
$$= \nabla\psi\times\nabla\varphi - \nabla f' + F^*\nabla\varphi$$
$$F \equiv F_0 + R^2\nabla\bullet\nabla_\perp f$$

Time advance:

$$\frac{\partial\mathbf{B}}{\partial t} = -\nabla\times\mathbf{E} \qquad (1)$$
$$\nabla\varphi\bullet\frac{\partial\mathbf{B}}{\partial t} = -\nabla\bullet(\mathbf{E}\times\nabla\varphi)$$
$$\frac{1}{R^2}\dot{F} = \nabla\bullet(\nabla\varphi\times\mathbf{E})$$

Or, substituting from the definition of F

$$\frac{1}{R^2}\dot{F}_0 + \nabla\bullet\left[\nabla_\perp\dot{f} - \nabla\varphi\times\mathbf{E}\right] = 0 \qquad (2)$$

Integrate Eq. (2) over the boundary contour and apply Gauss's Theorem letting $dS = RdRdZ$:

$$\iint \frac{1}{R^2}\dot{F}_0 dS + \iint \nabla \bullet \left[\nabla_\perp \dot{f} - \nabla\varphi \times \mathbf{E}\right] dS = 0$$

$$\iint \frac{1}{R^2}\dot{F}_0 dS + \int \hat{n} \bullet \left[\nabla_\perp \dot{f} - \nabla\varphi \times \mathbf{E}\right] R d\ell = 0$$

$$\dot{F}_0 \iint \frac{1}{R^2} dS + \int \frac{\partial \dot{f}}{\partial n} R d\ell = \int \hat{t} \cdot \mathbf{E}\, d\ell \tag{3}$$

Here we assume $\left(\hat{n}, \hat{\varphi}, \hat{t}\right)$ form a right-handed system that represents the normal, toroidal, and tangential directions. Another way to obtain the same result:

$$\frac{\partial \mathbf{A}}{\partial t} = -\mathbf{E} - \nabla\Phi$$

$$\nabla\varphi \times \left[R^2 \nabla\varphi \times \nabla_\perp \dot{f} + \dot{\psi}\nabla\varphi - F_0 \ln R \hat{Z} = -\mathbf{E} - \nabla\Phi\right]$$

$$\nabla_\perp \dot{f} + \dot{F}_0 \frac{1}{R} \ln R \hat{R} = \nabla\varphi \times \mathbf{E} + \nabla\varphi \times \nabla\Phi \tag{4}$$

Multiply Eq. (4) by $R d\ell \hat{n} \cdot$ and integrate around the boundary contour, noting that Φ is single valued, and using the identity:

$$\int \ln R \hat{R} \cdot \hat{n}\, d\ell = \iint \frac{1}{R^2} dS = \iint \frac{1}{R} dR\, dZ \quad , \tag{5}$$

we also obtain Eq. (3).

Now, from integrating the toroidal component of Eq. (1) over an area, we have:

$$\iint \dot{F} \frac{1}{R^2} dS = \int \hat{t} \cdot \mathbf{E}\, d\ell \tag{6}$$

Equating the LHS of Eq. (3) and (6) gives a solvability condition:

$$\dot{F}_0 \iint \frac{1}{R^2} dS + \int \frac{\partial \dot{f}}{\partial n} R d\ell = \iint \dot{F} \frac{1}{R^2} dS \tag{7}$$

Assuming that the solvability constraint is satisfied at time t=0, we can integrate Eq. (7) in time. Then, a consistent set of boundary conditions is:

$$\frac{\partial f}{\partial n} = 0 \tag{8a}$$

$$F_0 = \frac{\iint F R^{-1} dR dZ}{\iint R^{-1} dR dZ} \tag{8b}$$

Eq. (8a) is enforced by setting ifbound = 2. Equation (8b) needs to be implemented.

Suggested options:

1. Set iconst_bz=1 as this just keeps $F = const.$ on the boundary, so that the boundary toroidal field remains equal to the vacuum value, which is set by the current in the TF coils.

and

2. Either: (a) set ifbound=1 to keep $f = 0$ on the boundary, and $\partial f / \partial n$ should adjust to Eq. (7).

Or (b) set ifbound=2, to keep $\partial f / \partial n = 0$ (8a) but adjust F_0 each time step according to (8b)

16. Mesh Generation and Adaptation

(Contributed by Fan Zhang 9/2/2015)

This section describes how to generate a mesh with an enclosed vacuum vessel domain and how to perform mesh adaptation in M3D-C[1]. All the examples shown in this document are made in PPPL *portal* except for converting the initial Simmetrix mesh (.sms) into .vtk and .smb files used for Paraview and PUMI, respectively.

Section 16.1 presents how to generate a model and mesh files used in M3D-C[1]. Section 16.1 describes how to run mesh adaptation in M3D-C[1].

16.1 Mesh Generation (deprecated)

Load following modules on portal:

Intel/2015.u1 openmpi/1.8.4 paraview

16.1.1 Example 1: NSTX-1

- Location: */p/tsc/m3dc1/lib/develop.petsc3.Fan/MeshDemo/NSTX-1*
- To run:

```
cp /p/tsc/m3dc1/lib/SCORECLib/rhel6/openmpi-1.8.4/utilities/create_mesh/create_smd your_folder
cd your_folder
./create_smd
```

- Input:
  - The file *"input"* with the meshing control parameters
    - *modelType*: 0 for interpolate analytic model, 1 for piece-wise, 2 for three region model, 3 for piece-wise polynomial
    - *modelName*
    - *pointFile*: a file to describe the geometry
    - *meshSizes*: mesh size for the plasma (used by all types), wall (used by type 2) and vacuum (used by type 2) areas
    - *meshGradationRate*
    - *numInterPts*: parameter for type 0

        - *thickness*: wall thickness, only for type 2
        - *height/width*, *offsetX, offsetY*: for vacuum vessel in type 2
    - *NSTX*: the ordered set of points on the wall boundary.
        - In the file *"input",* parameter *"pointFile"* is set to "NSTX".
- Output
    - *NSTX0.02.smd*: Simmetrix model file.
    - *NSTX0.02.txt*: geometric model file used by M3D-C[1]
    - *NSTX0.02.sms*: mesh in PUMI .sms format
- Converting .sms mesh to .vtk and visualizing in portalr5

```
cp /p/tsc/m3dc1/lib/SCORECLib/rhel5/utilities/convert_sms/convert_sms your_folder
cd your_folder
./convert_sms NSTX0.02.sms mesh.vtk
module load paraview
paraview&
```

- Converting .sms mesh to .smb mesh partitioned into N parts in portalr5

```
mpirun -np N /p/tsc/m3dc1/lib/SCORECLib/rhel5/utilities/split_sms/split_sms NSTX0.02.sms
mesh.smb
```

To use the geometry and the mesh by M3D-C[1], specify model/mesh file in C1input

- *Set the parameter "mesh_model" to "NSTX0.02.txt"*
- *Set the parameter "mesh_filename" to "mesh.smb"*

Alternatively, an initial mesh can be generated through simmodeler. Launch simmodeler in portal. The model file, *NSTX0.02.smd,* can be opened through simmodeler. The mesh generated is saved as Simmetrix .sms file and can be converted to .smb file by the following steps.

- Converting a serial Simmetrix mesh (.sms) to PUMI mesh (.smb) in portal

```
cp /p/tsc/m3dc1/lib/SCORECLib/rhel6/openmpi-1.8.4/utilities/create_mesh/convert_sim_sms
your_folder
cd your_folder
./convert_sim_sms NSTX0.02.smd NSTX0.02.sms in-mesh.smb
```

- Splitting a serial PUMI mesh (.smb) to N parts in portal

```
cp /p/tsc/m3dc1/lib/SCORECLib/rhel6/openmpi-1.8.4/bin/split_smb/make_model your_folder
cp /p/tsc/m3dc1/lib/SCORECLib/rhel6/openmpi-1.8.4/binsplit_smb/split_smb your_folder
cd your_folder
mpirun –np 1 ./make_model in-mesh.smb model.dmg
mpirun –np N ./split_smb model.dmg in-mesh.smb mesh.smb N
```

16.1.2 Example 1: NSTX-2

Change "*meshSizes"* in *input* from "0.05 0.1 0.1" to "0.05 0.1 0.2" and see coarser mesh in the vacuum area.

16.1.3 Example 3: NSTX-3

Change parameters "*width*", "*height*", and "*offsetX*", to adjust the size and position of the vacuum area.

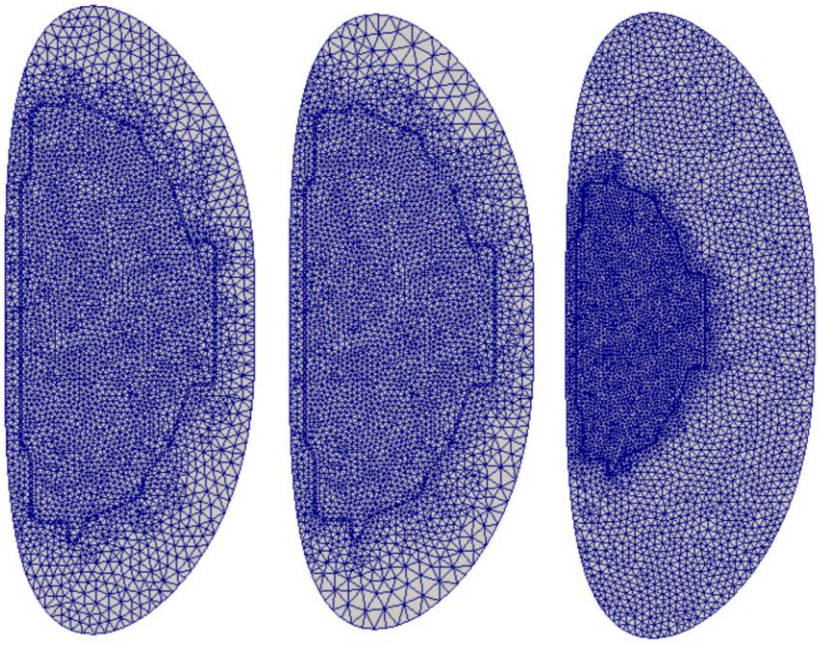


Figure 1 Mesh under NSTX-1, NSTX-2, NSTX-3

16.1.4 Example 4: NSTX-4

Due to improper parameters in *input*, the geometry will be invalid. Therefore, you will see the error message.

"Error: Code: 1103 String: Unable to mesh face 3"

Visualize the model with simmodeler and change the parameters to shift the vacuum curve.

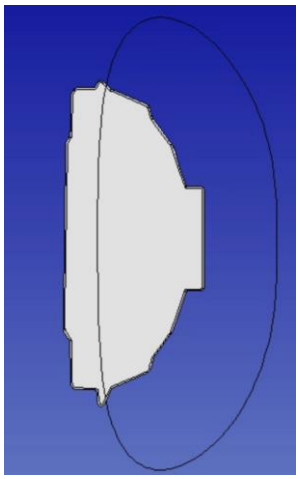


Figure 2 Invalid geometry

16.2 Anisotropic Mesh Adaptation (Will be deprecated)

The mesh is adapted to match the mesh size field defined either by post-processed magnetic flux field in the equilibrium or the estimated error in solution fields [1].

16.2.1 Adaptation by magnetic flux field

The mesh size field is defined as:

The normalized poloidal flux is defined as: $\tilde{\psi} \equiv \dfrac{\psi - \psi_0}{\psi_l - \psi_0}$, $\psi_0 =$ value at magnetic axis, $\psi_l =$ value at plasma boundary

inside plasma: $\tilde{\psi} < a_1$ exterior to plasma: $\tilde{\psi} > a_1$

$$\tilde{h}_1 = a_{4P}\left[1 - e^{-\left|\frac{\tilde{\psi}}{a_1}-1\right|^{a_2}}\right] + a_7 \qquad \tilde{h}_1 = a_{4V}\left[1 - e^{-\left|\frac{\tilde{\psi}}{a_1}-1\right|^{a_3}}\right] + a_7$$

$$\tilde{h}_2 = a_{5P}\left[1 - e^{-\left|\frac{\tilde{\psi}}{a_1}-1\right|^{a_2}}\right] + a_6 \qquad \tilde{h}_2 = a_{5V}\left[1 - e^{-\left|\frac{\tilde{\psi}}{a_1}-1\right|^{a_3}}\right] + a_6$$

$$h_i^{-1} = \tilde{h}_i^{-1} + \frac{1}{l_{ci}}\left[\frac{1}{1+\left(\dfrac{\tilde{\psi}-\psi_c}{W_c}\right)^2}\right]; \qquad i = 1,2$$

Note that h_1 is the length normal to the surfaces and h_2 the length tangential.

The mesh adaptation by the post-processed magnetic flux field requires a file "*sizefieldParam*" in your work directory to specify 13 size field parameters in the following order (single line, space delimited)

a_1 a_2 a_3 a_{4p} a_{4v} a_{5p} a_{5v} a_6 a_7 l_{c1} l_{c2} W_c ψ_c

- The mesh adaptation is performed before time steps if the C1input parameter "iadapt" is 1 or 3
- After adaptation, the initial equilibrium is re-calculated on adapted mesh so the analysis can continue
- Parallel anisotropic mesh size smoothing is not supported. (Will be available later)
- a_6/a_7 affects the aspect ratio of the element in the adapted mesh at the flux surface with the normalized psi value equaling to a_1.

16.2.2 Adaptation by error estimator

The mesh adaptation by error estimator is performed at the end of time step if one of the following conditions is met and the C1input parameter "iadapt" is set to 2 or 3

(1) iadapt_ntime > 0 and mod (current time step, iadapt_ntime) = 0
(2) linear=0 and iadapt_ntime = 0
(3) linear=1, adapt_ke > 0 and kinetic energy > adapt_ke

For the detailed discussions on mesh adaptation by error estimator, see Chapter 3 of Dr. Fan Zhang's dissertation, which is available in https://www.scorec.rpi.edu/~seol/m3dc1/fanzhang-thesis-chap3.pdf

16.2.3 Control parameters in C1input

- ***iadapt***
  - 0: no adaptation
  - *1: adapt mesh from the magnetic flux field in the equilibrium*
  - *2: adapt mesh from the estimated error in the solution field*
  - *3: adapt mesh from the magnetic flux field and the estimated error*
- ***adapt_hmin, adapt_hmax***
  - maximum and minimum sizes of the mesh elements in the adapted mesh.
- ***adapt_hmin_rel, adapt_hmax_rel***
  - bounds of a mesh element that can be changed from its original size in the adapted mesh (rel=relative).
- ***adapt_target_error***
  - target discretization error on the adapted mesh.
- ***iadapt_order_p***
  - target mesh size of a mesh element is proportional to the original mesh size as $(\tau/h_{org})^{-p-1}$ [2], where τ is the estimated error contributed by the mesh element. The value is no larger than 3 in H^2 space for M3D-C^1 [3].

- ***iadapt_max_node***
  - maximum node number in the adapted mesh. If the estimated mesh node number from *adapt_target_error* exceeds *iadapt_max_node,* the target mesh size in the adapted mesh is scaled such that the mesh node number is below *iadapt_max_node.*
- ***adapt_ke***
  - *For linear, if adapt_ke* >0 *and kinetic energy is greater than adapt_ke, run mesh adaptation by error estimator*
- ***iadapt_ntime***
- ***adapt_control:***
  - 0: *adapt_target_error* is global (integral over the domain) [2]
  - 1: *adapt_target_error* is local (integral over the element)
- ***iadapt_useH1:***
  - set value to 1 if fluid viscosity and electrical resistivity << 1
- ***iadapt_removeEquiv:***
  - set value to 1 to remove the terms containing the equilibrium solution in the estimated error
- ***iadapt_writesmb***
  - if 1, write the adapted mesh in "ts*N*-adapted.smb", N: time step when the mesh adaptation is performed (default 1)
- ***iadapt_writevtk***
  - if 1, write the initial and adapted mesh in VTK format (default: 0)
  - the initial mesh is written in the folder "ts*0*-initial"
  - the adapted mesh is written in the folder "ts*N*-adapted", N: time step when the mesh adaptation is performed

16.2.3 Examples

The examples presented in this document are available in M3D-C[1] repository/DATA/adapt as well as

/p/tsc/m3dc1/lib/develop.petsc3.Fan/Aug26/DATA/adapt/

[1] adapt/anisotropic (4 processes)

Plasma equilibrium is obtained by Grad-Shafranov solver in M3D-C[1].

[C1input]
iadapt=1
iadapt_writevtk=1

[sizefieldParam]
0.9 2 1 .05 .5 .05 .5 .1 .01 5. 5. 0.3 0.148

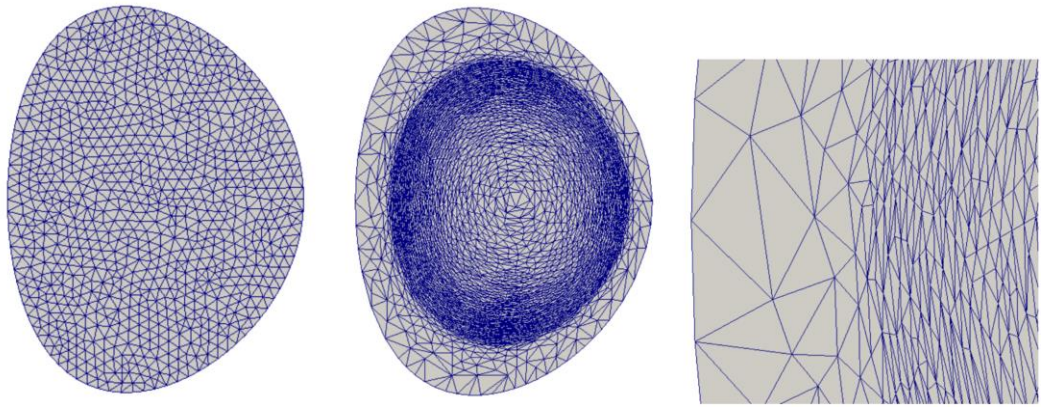


Figure 3 Initial meshes, adapted mesh and its close-up under folder adapt/anisotropic

[2] adapt/anisotropic2 (4 processes)

[C1input]

iadapt=1

iadapt_writevtk=1

[sizefieldParam]

0.8 2 1 .05 .5 .05 .5 .1 .02 5. 5. 0.3 0.148

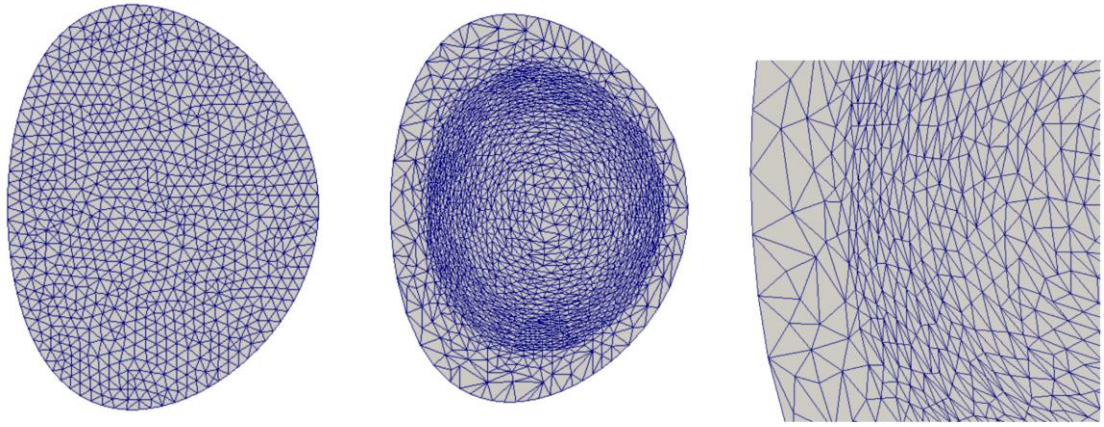


Figure 4 Initial meshes, adapted mesh and its close-up under folder adapt/anisotropic2

[3] adapt/tilt

This example presents the mesh adaptation with the error estimation using the tilt mode. The solution is transferred to the new mesh in the non-linear simulation.

[C1input]
mesh_filename = tilt.smb
mesh_model = tilt.txt
iadapt = 2
iadapt_ntime = 4
adapt_target_error = 0.02
adapt_control = 0
iadapt_max_node = 600
iadapt_writevtk = 1
iadapt_order_p = 1.5
adapt_hmin = 0.03
adapt_hmax = 0.4
ntimemax=200
To visualize the result, launch IDL and enter the following

```
plot_field, 'jphi',
 file='/p/tsc/m3dc1/lib/develop.petsc3.Fan/Aug26/DATA/adapt/tilt/C1.h5',
00, /mesh
```

To visualize *jphi* field on the adapted mesh in different time slice, change 00 to 30, 60, 90, or 200.

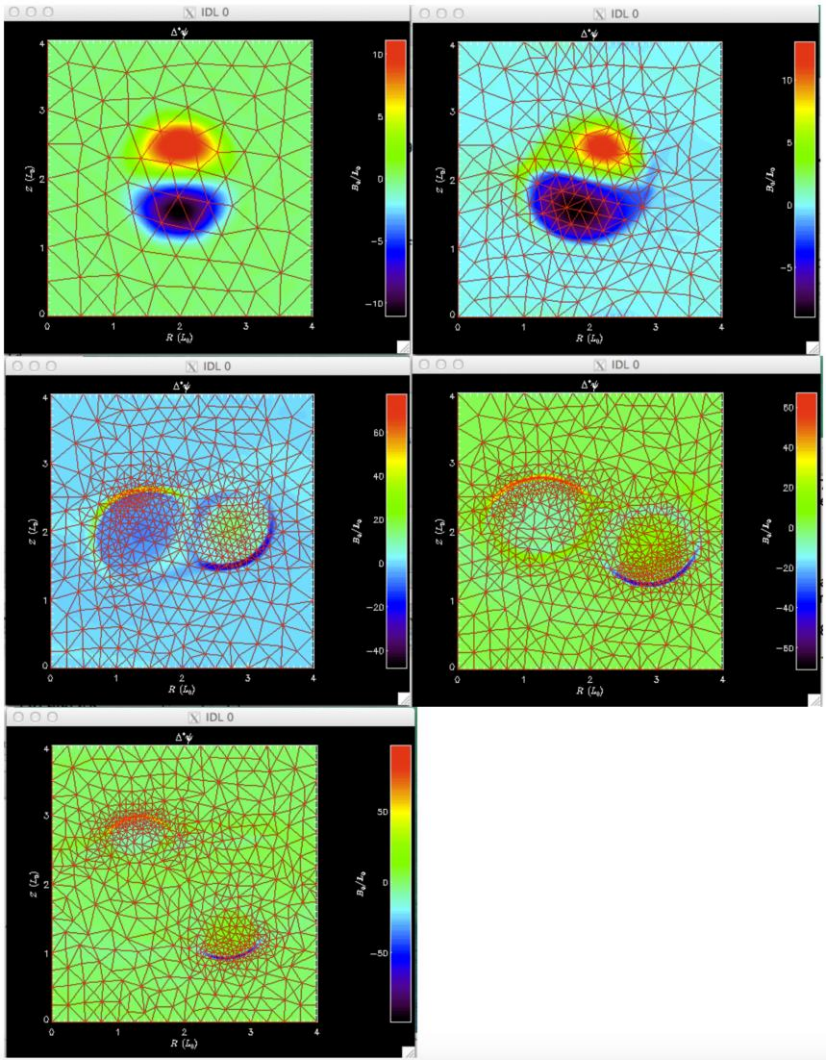


Figure 5 *jphi* field on the adapted mesh in time step 00, 30, 60, 90 and 200

[4] adapt/ELM (8 processes)

The mesh is adapted from the estimated error in the eigen mode.

[C1input]
mesh_filename = Analytic.smb
mesh_model = AnalyticModel
iadapt = 2
iadapt_max_node = 15000
iadapt_useH1=1
iadapt_removeEquiv =1
adapt_target_error = 0.005

adapt_hmin = 0.005
adapt_hmax = 0.1
adapt_ke = 5e-2
iadapt_order_p = 2
iadapt_writevtk = 1

In order to visualize the growth rate on the adapted mesh, launch IDL and enter the following

```
plot_scalar, 'ke',
file='/p/tsc/m3dc1/lib/develop.petsc3.Fan/Aug26/DATA/adapt/ELM/C1.h5',
/growth, yrange=[0.,0.15]
```

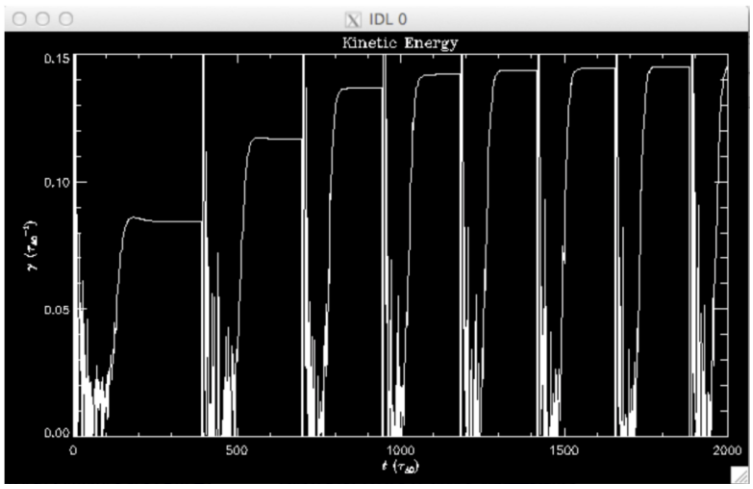


Figure 6 Growth rate of kinetic energy on the adapted mesh (the mesh is adapted 7 times)

To start a new simulation with the adapted mesh,

```
cd adapt
mkdir ELM2
cd ELM2
cp  ../ELM/adapt943*smb .
./change_name.sh adapt943 adapt 8
Copy the input files from adapt/ELM
Modify C1input to set mesh_filename to "adapt.smb"
Run the simulation
```

Enter the following command in IDL to view the growth rate.

```
plot_scalar, 'ke', file='/p/tsc/m3dc1/lib/develop.petsc3.Fan/Aug26/DATA/adapt/ELM2/bk/C1.h5',
/growth, yrange=[0.145,0.15]
```

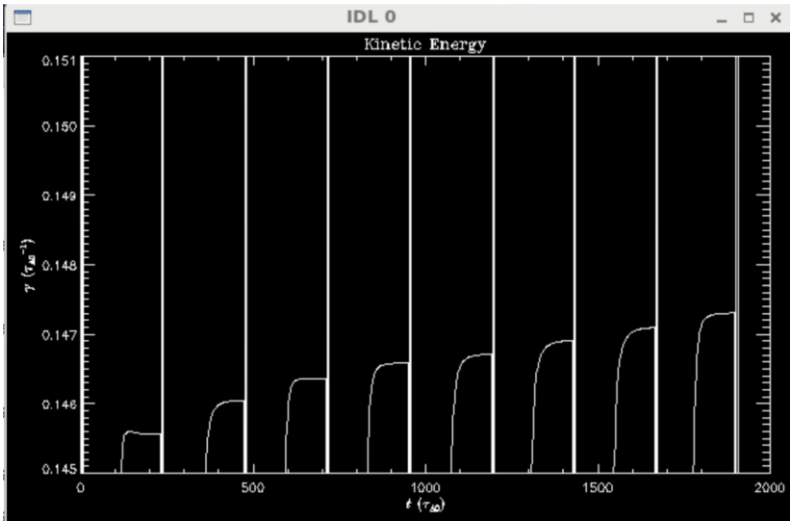


Figure 7 Growth rate of kinetic energy starting from the previously adapted mesh

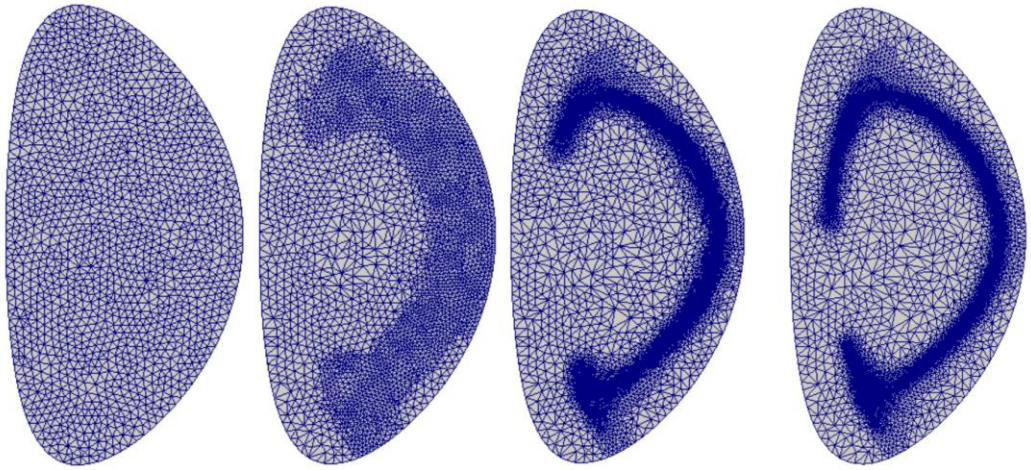


Figure 8 Adapted meshes for ELM after 0, 1, 6, 13 times adaptation

(Visualized with Paraview, note that the current IDL will only show mesh at time 0 for linear runs)

[5] adapt/doubleTearing (8 processes)

The mesh is adapted from the estimated error in the eigen mode.

[C1input]
mesh_filename = Analytic.smb
mesh_model = AnalyticModel
iadapt = 2

iadapt_order_p = 2
adapt_target_error = 1e-16
adapt_hmin = 0.01
adapt_hmax = 0.1
adapt_hmin_rel = 0.3
adapt_hmax_rel = 3.
iadapt_max_node = 8000
iadapt_useH1=1
iadapt_removeEquiv =1
iadapt_writevtk = 1
iadapt_ntime = 300

References

M. Ainsworth and J. T. Oden, “A posteriori error estimation in finite element analysis,” Comput. Methods Appl. Mechanics Eng., vol. 142, no. 1, pp. 1–88, Mar. 1997

E. Onate and G. Bugeda, “A study of mesh optimality criteria in adaptive finite element analysis,” Eng. Comput., vol. 10, no. 4, pp. 307–321, Dec. 1993.

T. J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Mineola, NY, USA: Dover Publications, 2012.

17. Mesh Generation with Simmodeler GUI

(Contributed by D. Pfefferle 4/27/16)

More info: http://redmine.scorec.rpi.edu/projects/pafs/wiki/Mesh_Generation_for_PPPL

1. Login and load environment:
   module load simmodeler

2. Create ascii_file as in Step 2 of Section 2.2. For modelType 3 (3 region model), inFile is a file with a list of points describing the resistive wall. The input files (nstx-input and nstx_conduct_vessel_spline.dat) used herein can be found under the directory /p/m3dc1/dpfeffer/NSTX-VDE/Shot132859/MeshGen/.

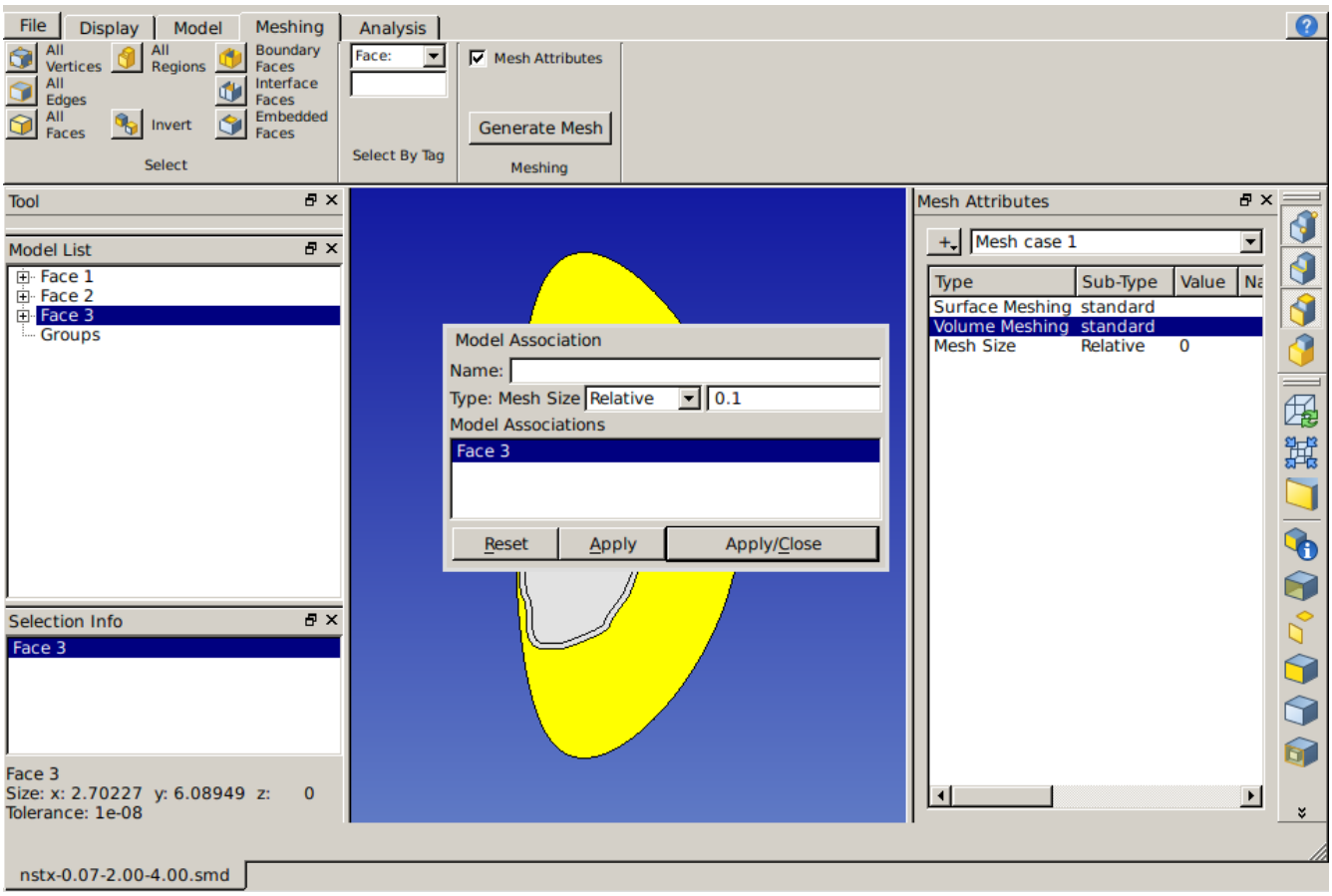
3. Run
   /p/tsc/m3dc1/lib/SCORECLib/rhel6/openmpi-1.8.4/bin/m3dc1_meshgen nstx-input

   It will complain that the license is not working but it will correctly generate the .smd and a .txt file.

4. Run Simmodeler

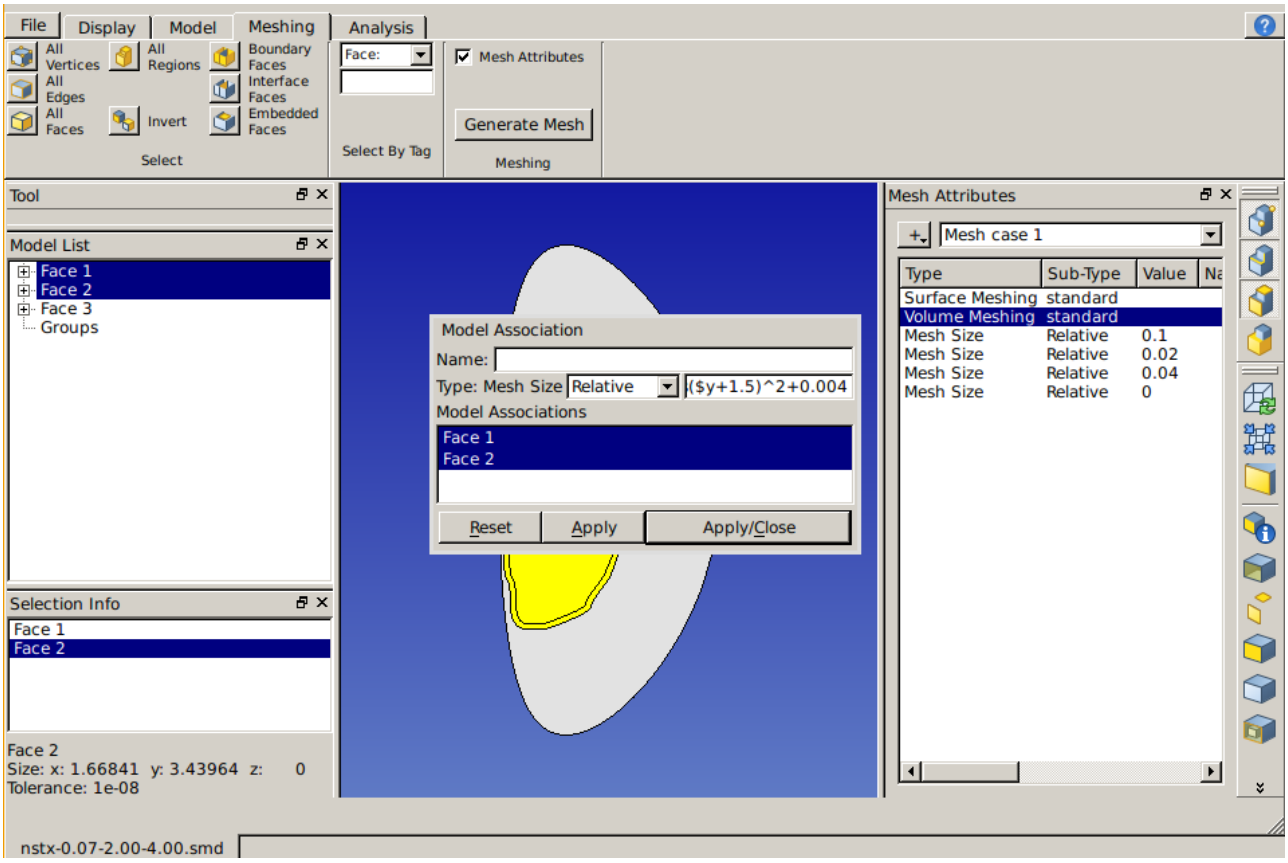
simmodeler

5. File->Open Model->nstx-xxx-xxx-xxx.smd (where the xxx are numbers related to characteristic widths)
6. In the upper panel, in the views section click on “Front” to view the model, then go to Meshing tab
7. Select outer region, click “+” in “Mesh Attributes” and select “Mesh Size”->”relative”. Enter a value (typically 0.1)



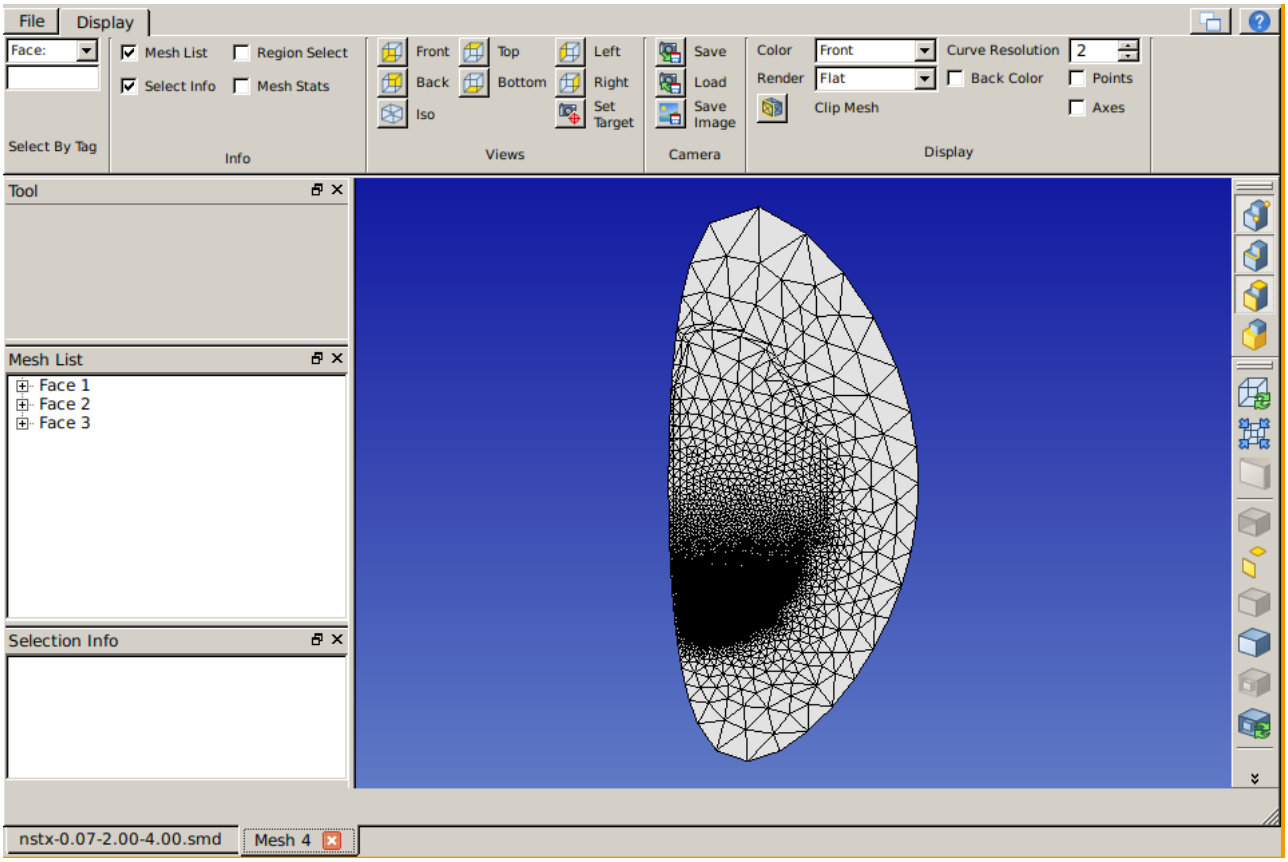
8. Select wall region, click “+” in “Mesh Attributes” and select “Mesh Size”->”relative”. Enter a value (typically 0.02)
9. Select inner region, click “+” in “Mesh Attributes” and select “Mesh Size”->”relative”. Enter a value (typically 0.04).

   Comment: here, one can already generate the mesh by clicking on “Generate Mesh” and verify if the mesh sizes are suitable
10. Select both inner and wall regions (holding shift key), click “+” in “Mesh Attributes” and select “Mesh Size”->”relative”. Enter a function, e.g. 0.01*abs($y+1.5)^2+0.004 to specify an anisotropic mesh density on top of previous settings



Comment: there are many available parameters for fine-tuning the mesh density. For example the Mesh urvature Refinement with parameter packs more elements near the edges of the resistive wall.

11. "Generate Mesh" and "Show Mesh" to view result in new windows



12. If the result is satisfactory, File->Save Mesh, give it a meaningful name with the extension .sms. Close Simmodeler. The original .smd has been automatically saved by the program with your mesh modifications.
13. Copy the .txt, .smd and .sms files to the simulation directory and run the following splitting routine to obtain .smb parts
    /p/tsc/C1/m3dc1-sunfire.r6-1.5/bin/part_mesh.sh nstx-xxx-xxx-xxx.smd mygeneratedmesh.sms X
    where X is the number of .smb parts you need.
14. Modify the C1input file accordingly
    mesh_filename = 'part.smb'

    mesh_model = 'nstx-xxx-xxx-xxx.txt'

17.1 No more Simmetrix licenses available:

NOTE: If you get a message that there are no more Simmetrix licenses available, to see who is using:
cd /usr/pppl/Simmetrix
./rlmutil rlmstat -a

18. ITAYLOR=27, IKAPPAFUNC=12, IRESFUNC=0

bz_qp	bzero*rzero		rzero_qp	rzero	R
r0_qp	alpha0		p0_qp	p0	p_0
q0_qp	q0	q_0	pedge_qp	pedge	p_{edge}
q2_qp	alpha1	q_a	kappa_qp	kappa0	κ_0
q4_qp	alpha2	a	kappae_qp	alpha3	κ_V

$$\psi = a_1 r^2 + a_2 r^4 + a_3 r^6$$
$$= r^2\left[\tfrac{1}{4}J_0 + (r/a)^2\left(-\tfrac{5}{16}J_0 + \tfrac{1}{3}J_2\right) + (r/a)^4\left(\tfrac{1}{9}J_0 - \tfrac{4}{27}J_2\right)\right]$$
$$J = \frac{1}{r}\frac{d}{dr}r\frac{d}{dr}\psi = J_0 + (r/a)^2\left(-5J_0 + \tfrac{16}{3}J_2\right) + (r/a)^4\left(4J_0 - \tfrac{16}{3}J_2\right)$$

$$\psi(a) = a^2\left(\tfrac{7}{144}J_0 + \tfrac{5}{27}J_2\right) \qquad \left.\frac{d\psi}{dr}\right|_{r=a} = a\left(-\tfrac{1}{12}J_0 + \tfrac{4}{9}J_2\right)$$

$$\psi_V = \psi(a) + a\left.\frac{d\psi}{dr}\right|_{r=a}\ln\left(\frac{r}{a}\right) = a^2\left(\tfrac{7}{144}J_0 + \tfrac{5}{27}J_2\right) + a^2\left(-\tfrac{1}{12}J_0 + \tfrac{4}{9}J_2\right)\ln\left(\frac{r}{a}\right)$$

$$q(r) = \frac{B_T}{R}\frac{r}{d\psi/dr} \qquad J_0 = \frac{2B_T}{q_0 R} \qquad J_2 = \frac{3}{8}\frac{B_T}{R}\left(\frac{6}{q_a} + \frac{1}{q_0}\right)$$

$$J = \frac{2B_T}{q_0 R}\left[1 + (r/a)^2\left(-4 + 6\frac{q_0}{q_a}\right) + (r/a)^4\left(3 - 6\frac{q_0}{q_a}\right)\right] \qquad I = 2\pi\int_0^a Jrdr = 2\pi\frac{B_T}{q_a R}$$

$$\frac{1}{r}\frac{d\psi}{dr} = \frac{B_T}{R}\left[\frac{1}{q_0} + (r/a)^2\left(-\frac{2}{q_0} + \frac{3}{q_a}\right) + (r/a)^4\left(\frac{1}{q_0} - \frac{2}{q_a}\right)\right]$$

$$q(r) = \begin{cases} q_0\left[1 + (r/a)^2\left(-2 + 3\dfrac{q_0}{q_a}\right) + (r/a)^4\left(1 - 2\dfrac{q_0}{q_a}\right)\right]^{-1} & r < a \\ q_a (r/a)^2 & r \ge a \end{cases}$$

Monotonic for $3q_0 > q_a > \frac{3}{2}q_0$. Define $A = (1/a)^2\left(-2 + 3\dfrac{q_0}{q_a}\right)$, $B = (1/a)^4\left(1 - 2\dfrac{q_0}{q_a}\right)$, $y = r^2$

$$q(y) = \begin{cases} q_0\left[1 + Ay + By^2\right]^{-1} & y < a^2 \\ q_a (1/a)^2 y & y \ge a^2 \end{cases}$$

$$\frac{dq}{dy} = \begin{cases} -q_0\left[1 + Ay + By^2\right]^{-2}(A + 2By) & y < a^2 \\ q_a (1/a)^2 & y \ge a^2 \end{cases}$$

Pressure:
$$p = p_0\left[1 + (r/a)^2\left(-4 + 6\frac{q_0}{q_a}\right) + (r/a)^4\left(3 - 6\frac{q_0}{q_a}\right)\right]^{2/3} + p_{edge}$$

Define $A=(1/a)^2\left(-4+6\frac{q_0}{q_a}\right),\qquad B=(1/a)^4\left(3-6\frac{q_0}{q_a}\right),\qquad y=r^2$

$$p=\begin{cases} p_0\left[1+Ay+By^2\right]^{2/3}+p_{edge} & y<a^2 \\ p_{edge} & y\ge a^2 \end{cases}$$

$$\frac{dp}{dy}=\begin{cases} p_0\frac{2}{3}\left[1+Ay+By^2\right]^{-1/3}(A+2By) & y<a^2 \\ 0 & y\ge a^2 \end{cases}$$

$\eta J \sim p^{-3/2}J \sim \text{const.}\qquad \frac{1}{r}\frac{d}{dr}r\kappa\frac{dp}{dr}=-CJ$

$$\kappa\frac{dp}{dr}=-C\frac{1}{r}\int_0^r r\left[1+(r/a)^2\left(-4+6\frac{q_0}{q_a}\right)+(r/a)^4\left(3-6\frac{q_0}{q_a}\right)\right]dr$$

$$=-Cr\frac{1}{2}\left[1+(r/a)^2\left(-2+3\frac{q_0}{q_a}\right)+(r/a)^4\left(1-2\frac{q_0}{q_a}\right)\right]$$

$$\kappa=\kappa_0\frac{\left[1-2(r/a)^2\left(1-\frac{3}{2}\frac{q_0}{q_a}\right)+(r/a)^4\left(1-2\frac{q_0}{q_a}\right)\right]\left[1-4(r/a)^2\left(1-\frac{3}{2}\frac{q_0}{q_a}\right)+3(r/a)^4\left(1-2\frac{q_0}{q_a}\right)\right]^{1/3}}{\left[\left(1-\frac{3}{2}\frac{q_0}{q_a}\right)-\frac{3}{2}(r/a)^2\left(1-2\frac{q_0}{q_a}\right)\right]}$$

$$A=\left(1-\frac{3}{2}\frac{q_0}{q_a}\right),\quad B=\left(1-2\frac{q_0}{q_a}\right)$$

$$\kappa=\begin{cases} \kappa_0\left[1-2(r/a)^2A+(r/a)^4B\right]\left[1-4(r/a)^2A+3(r/a)^4B\right]^{1/3}\left[A-\frac{3}{2}(r/a)^2B\right]^{-1} & y<a^2 \\ \kappa_V & y\ge a^2 \end{cases}$$

19. Example and Test Programs

19.1 Regression Tests

Start an interactive session with 8 processors.
Go to the directory …./unstructured/regtest and run “make N=8” (if necessary, first run “make clean”)
The tests will take 5-10 min to run
If each test succeeds, “Success!” will be printed
If not, the tests will stop with an error.
Details of each test case are in the README file in that directory.

19.2 Test Programs

The following is a series of test problems that a new user can use as a guide to running M3D-C[1].
Directories are subdirectories of PPPL directory: /p/tsc/m3dnl/Test

Test1: CMOD equilibrium with q0=0.6 (COM=1, 4-5 min)
This is a complete "step-by-step" guide to perform a first run with the M3D-C1 code for new users. It assumes that the code has already been downloaded from the GIT repo (as explained in Sec. 1.2) and compiled (Sec. 1.4.2)

1. Copy the files into your own directory and submit the batch file batchr8 via the command: "sbatch batchr8". This will generate the mesh, split it into 8 partitions, and solve the Grad Shafranov equation on the mesh. Note that you can change the shape of the mesh boundary with the parameters "vacuumParams" and can change the mesh resolution with "meshsize". These are all parameters in the file analytic-input

2. Verify from the output file (slurm-####.out) that the GS equilibrium iteration has converged. (search for Converged GS error). You can also see the error in previous iterations (this error is set with the variable tol_gs in the C1input file.

3. View the equilibrium fields with idl postprocessor: psi, I, jy, p, pe, den,eta,visc, kappa. (Add cutz=0 to view midplane profiles.) IDL must be configured as explained in Sec. 5.5.5. Go to the IDL directory to plot. Example (Sec. 5.1.2) "plot_field, 'psi',file='path to your Test 1 files/C1.h5', 0

4. View the surfaces and q-profile with the Poincare postprocessor (Section 5.2).

5. Optional: repeat with different mesh size to verify results are converged

Test2: CMOD linear stability (COM=1, ~ 15 min)
1. This is the same as Test 1, except it runs for 100 time steps instead of 0. Compare C1input files from Test2 and Test1 to verify.
2. Submit job by executing "sbatch sbatchc8.
3. View mode growth rate with idl postprocessor: plot_scalar,'ke',/growth,yrange=[0,.01]. Adjust yrange min and max to obtain growth rate to 3 digits.
4. View the surfaces with the Poincare postprocessor (subdirectory Plots). Note that the multiplier may need to be adjusted.
5. View the eigenfunction with the idl postprocessor: plot_field,'jphi',1,/linear (also, 'phi', etc)
6. Test2a: Repeat this with numvar=1 (reduced MHD) and compare the difference in the growth rate and eigenfunction for full MHD and reduced MHD
7. Test2b: Repeat this with numvar=3; ipres=1, gyro=1, db=0.1, itwofluid=1 and compare the difference in the growth rate and eigenfunctions for two-fluid MHD, full MHD, and reduced MHD.
9. Optional: repeat with finer mesh and smaller resistivity to determine scaling of growth rate with resistivity, etc.

Test3: FRS Tearing mode using adaptive mesh (COM=1, ~ 2 hours)
This is a high-resolution test case of a m=2, n=1 tearing mode in a cylindrical tokamak with equilibrium profiles taken from Furth, Rutherford, Selberg 1973 (FRS). It is a tokamak of aspect ratio 10 taken as a cylinder in the large aspect ratio limit. The cross section is a circle of radius 1. The plasma extends from r=0 to r=1 and there is a perfectly conducting wall at r=1. Itaylor=16 corresponds to the peaked equilibrium profile from FRS for the current, and a parabolic pressure profile p=p0*(1-r**2).
The mesh is adapted to be finer at the rational surface by doing the following:
1. A uniform mesh is created by running "sbatch batch" in directory MESH. It is split into 16 parts which are stored in partnn.smb.
2. M3DC1 is then run with iadapt=1 by running "sbatch batchr16" in directory ADAPT. The parameters of the mesh adaptation are in sizefieldParam. This creates the adapted mesh files adaptednn.smb
3. Switch to the directory "Run" and run with the adapted mesh for 500 time steps with dt=10 by executing "sbatch batchc16.
4. View the growth rate with the idl postprocessor: plot_scalar,'ke',/growth,yrange=[ymin,ymax]
5. View the eigenfunction with the idl postprocessor: plot_field,'jphi',2,/linear (also 'vr')

Test4: Double tearing Mode

Execute the batch script via "sbatch batchc8". This will generate a mesh, partition it into 8 partitions, solve the Grad-Shafranov equation for an equilibrium with 2 q=2 surfaces, and calculate the linear eigenfunctions and growth rate of a double tearing mode in this equilibrium.

Test5: 2D long time evolution in CMOD (RL=1, ~ 3 hrs) (**Not presently available**)
1. Prepare and submit a batch file with the REAL 2D code version.
2. Note that the C1input file has control systems specified to keep the plasma current and total number of particles constant in time.
3. Verify with the idl postprocessor that the toroidal plasma current "it" and the number of particles "particles" remain approximately constant in time. The loop voltage "vl" varies to keep "it" constant.

Test6: NSTX linear stability with strong rotation starting from geqdsk file (COM=1, ~ 3 hrs) **(Not presently available)**
1. Using the AnalyticModel file in this directory, generate a mesh with mesh size .04
2. First calculate an equilibrium (ntimemax=0, irestart=0) and verify that it is diverted by (1) finding "Plasma is diverted" after the last GS iteration in the output file, and (2) in the idl postprocessor: "plot_field,'jphi',0,/lcfs " You should see the last closed flux surface and x-point in red.
3. Can you calculate the ratio of the maximum toroidal velocity to the Alfven velocity? To the sound velocity?
4. In the same directory, change the C1input variables (ntimemax=100, irestart=1) and resubmit. Calculate the growth rate to 3 digits with the idl routine: "plot_scalar,'ke',/growth,yrange=[min,max]" by adjusting the min and max.
5. Test4a: Repeat this with irot = 0 to verify it is stable without rotation.

Test7: 3D nonlinear run (3D=1, 6 hours) **(Not presently available)**
This is a nonlinear 3D run using an initial equilibrium similar to Test2a, Note this is NUMVAR=1 (reduced MHD) with 8 planes. You must generate 2 partn.sms files by running PTNMESH with M=2 using the struct-curveDomain.sms file from Test1. (see Section 2.4)
1. First set (irestart=0, ntimemax=0) and submit (~ 20 min). Verify that the equilibrium has converged.
2. Next, set (irestart=1, ntimemax=100) and submit (~ 6 hrs). This nonlinear run should develop the instability seen in Test2a. You can plot the energy and the growth rates of the different toroidal harmonics with "plot_kehmn". Add /ylog or /growth to get a logarithmic scale or the growth rate. Add yrange=[min,max] to adjust the scale. Compare the growth rate of the n=1 harmonic with that obtained for the growth of the kinetic energy in the linear run Test2a.
3. Run the Poincare plotter in the subdirectory: Plots to view the island
4. If islands are not yet visible, restart calculation to timestep 250 and run Poincare plotter again.
5. Repeat this run at NERSC with more planes, smaller grid size, and lower resistivity (eta). Also, repeat with NUMVAR=3 and compare growth rate with that obtained in Test2.
6. At NERSC: change the parameter "eqsubtract" from 1 to 0 so that the equilibrium is not being subtracted off. Add density evolution (idens=1) and controllers for the current and density as in the 2D run inTest3.
7. At NERSC, include 2-fluid effects by setting: itwofluid=1, gyro=1, db=.04, harned_mikic=.05)

Test8: n=1 plasma response to DIII-D RMP coils (COM=1 7 min) **(Not presently available)**
An example M3D-C^1 case for n=1 response in M3D-C^1

This is a low-resolution case, with 1 kA in the I-coils in even parity configuration. The (R, Z) locations of the I-coils are defined in rmp_coil.dat, and the currents in each coil are defined in rmp_current.dat. Aside from that and C1input, the other input files are:

geqdsk: the EFIT "g" file
profile_omega: the ExB toroidal rotation frequency in krad/s, versus psi_norm
profile_te: the Te profile in keV, versus psi_norm
profile_ne: the ne profile in 10^20/m^3, versus psi_norm

This is a time-independent calculation, so there is only one time step. There will be two times output: time0, which contains the vacuum fields, and time 1, which contains the full time-independent plasma response.

In C1input, you can specify whether the calculation is time-independent or not with "itime_independent". If you want, you can try running a time-dependent case to steady-state as well. Assuming that there's no instability, the time-independent solution should be the same as the time-dependent solution when it has reached steady-state. Of course, the time-independent calculation is much more efficient.

20. Instructions for using qsolver to initialize a toroidal m3dc1 run

1. Copy the executable "go" and the input files "inequ" and "eqxz" from the PPPL directory /p/tsc/m3dnl/Qsolver/Testcase. You can run it with the command "./go" and it will produce the two output files "profiles-g" and "profiles-p" and the restart file "eqb1".

This corresponds to a fixed boundary equilibrium with outermost flux surface given by:
X(theta) = xzero + aguess*cos(theta + $\sin^{-1}$(dguess)*sin(theta))
Z(theta) = aguess*eps*sin(theta)

The pressure and q profiles are given by:
P = p0*(1.0 – psinorm**beta)**alpha
q = q0 for psinorm < qalph
= q0 + qdp0*(psinorm – qalph)**qpof

Note that all these input quantities need to be in a special place in the inequ file. The first 2 digits of each input line indicates what it expects to find on that line. The 7 fields on each line start in column number 11,21,31, etc. If an input variable is in the wrong field it will be misread.

2. Changes must be made incrementally. After each run, copy eqb1 to eqxz. The p0 in the original file was 0.020. Let's say you want to increase it to 0.025. Change p0 in the input file and run: ./go. When it converges, part of the printout (29th line from the bottom) will give the value of p(0). In this case: p(0)=.024953. Note that this differs from the input value p0 = .025. To get them to agree, you must rerun with a different value of aisw2. (type 04 field 6). P(0) will be a linear function of aisw2 so you should only need to rerun 3 times at most. Each time you rerun, copy eqb1 to eqxz. I found that by increasing aisw2 from 7.85 to 7.8575 the output value of p(0) = 2.5000E-02, correct to 5 digits.

3. Now copy files profiles-g and profiles-p to a directory where you are running m3dc1. In the m3dc1 C1input file you must set itor=1 and inumgs=1

4. For the m3d-c1 run, a mesh file must first be prepared using the method described in Sec. 2.2, model type 0. Note that this parameterization of the boundary must be exactly the same as that used in step 1 above. (Note that due to differing convention X_2 must be set to $\sin^{-1}$(dguess))

5. A sample m3dc1 run using the files generated from qsolver can be found in /p/tsc/m3dnl/Qsolver/TestStab

21. Running with stellarator geometry

Presently, it is possible to do 3D runs in stellarator geometry. (The presence of toroidal mode coupling in stellarator geometry means a generalization of the linear capability is less straightforward.). Running the code in stellarator geometry is very similar a standard 3D run (see 4.2) although additional parameters must be specified, primarily to describe the geometry.

For both fixed- and free-boundary runs, the following options must be set in C1input:

igeometry = 1
type_ext_field = 1
iread_vmec = 1

Fixed boundary:

1. For a fixed boundary run, the geometry must be specified by providing a VMEC wout*.nc file in the working directory. In C1input, set vmec filename to the name of the VMEC wout file. This will determine the geometry and the initial fields.

2. In C1input, ensure nperiods is set equal to the number of field periods. Set ifull_torus = 0 (one field period) or 1 (full torus).

3. Run the code with the executable m3dc1_3d_st (see 1.4.2).

Free-boundary:
1. For a free-boundary run, the following flags must be set in C1input (along with what is required for a fixed-boundary run):

iread_ext_field = 1
type_ext_field = 1

2. Like a fixed-boundary run, the geometry is specified by a VMEC wout*.nc (vmec_filename). The initial fields are given by either a FIELDLINES (vacuum or finite-beta) or MGRID (vacuum) file. In C1input, set file_ext_field to the name of the file. Note that it must start with either 'fieldines' or 'mgrid'.

3. The domain for free-boundary calculations is obtained by 'bloating' the geometry given by the VMEC file and can be set with either bloat_factor or bloat_distance. Note that bloat_distance overrides bloat_factor.

4. Run the code with the executable m3dc1_3d_st (see 1.4.2).

22. Future Work

22.1 Demonstrate that JADV=0 and JADV=1 give the same (converged) results for the GEM problem. Extend this to use the ion form of the 2F equations. Can NUMVAR=1 be run with JADV=0?

22.2 Restore iper=1 options for slab geometry with multiple processors

22.3 Explore use of GPUs to do integrations

22.4 Implement better, more scalable preconditioners.

Can the pressure matrix use multigrid in phi?

Can we reorder the velocity variables to have 3 SuperLU solves per plane

22.5 Routine nonlinear 2F capabilities

22.6 Implement better transport models

22.7 Symmetric matrices?

22.8 Full neoclassical model

22.9 Add the ability of the idl postprocessor to plot the difference of 1 time slice at two toroidal angles.

22.10 allow itemp=1 with ipressplit=0

23. References

Journal articles that describe the mathematical formulation, algorithms, and verification studies related to M3D-C^1 are the following:

Title: **Multi-region approach to free-boundary three-dimensional tokamak equilibria and resistive wall instabilities**
Author(s): Ferraro, N.M., Jardin, SC, Lao, LL, Shephard, MS, Zhang, F
Source: Physics of Plasmas, **23**:056114 (2016)

Title: **Mesh generation for confined fusion plasma simulations**
Authors: F. Zhang, R. Hager, S.H.Ku, C.S. Chang, S. Jardin, N. Ferraro, S. Seol, E. Yoon, M. Shephard,
Source: Engineering with Computers, **32**:285-293(2016)

Title: **Self-Organized Stationary States of Tokamaks**
Author(s): Jardin, S. C., Ferraro, N.M., Krebs, I.
Source: Physical Review Letters, **115,** 215001 (2015)

Title: **Role of plasma response in displacements of the tokamak edge due to applied non-axisymmetric fields**
Author(s): Ferraro N. M.; Evans, T.E.; Lao, L. L.; et al.
Source: Nuclear Fusion **53** 073042 (2013)

Title: **Multiple timescale calculations of sawteeth and other global macroscopic dynamics of tokamak plasmas**
Author(s): Jardin, S.C; Ferraro N. M.; Breslau, J; Chen J
Source: Computational Science & Discovery **5** 014002 (2012)

Title: **Calculations of two-fluid linear response to non-axisymmetric fields in tokamaks**
Author(s): Ferraro N. M.
Source: PHYSICS OF PLASMAS Volume: **19** 056105 (2012)

Title: **Fluid Modeling of Fusion Plasmas with M3D-C^1**
Author(s): Ferraro N. M.; Jardin S.; Shephard M.; Bauer A.; Breslau J.; Chen J.; Delalondre F.; Luo X.; Zhang F.
Source: Proc. SciDAC 2011, Denver, CO July 10-14, 2011, http://press.mcs.anl.gov/scidac2011

Title: **Ideal and resistive edge stability calculations with M3D-C^1**
Author(s): Ferraro N. M.; Jardin, S, C.; Snyder P. B.
Source: PHYSICS OF PLASMAS **17 102508** OCT 2010

Title: **Calculations of two-fluid magnetohydrodynamic axisymmetric steady-states**
Author(s): Ferraro N. M.; Jardin, S.C.
Source: J. OF COMPUTATIONAL PHYSICS **228** 7742-7770 **(**2009)

Title: **Some properties of the M3D-C^1 form of the three-dimensional MHD equations**
Author(s): Breslau J.; Ferraro N.; Jardin, S.
Source: PHYSICS OF PLASMAS **16** 092503 (2009)

Title: **A high-order implicit finite element method for integrating the two-fluid MHD equations in 2D**
Author(s): Jardin S. C.; Breslau J.; Ferraro N.
Source: J. OF COMPUTATIONAL PHYSICS **226** 2146-2174 (2007)

Title: **Finite element implementation of Braginskii's gyroviscous stress with application to the gravitational instability**
Author(s): Ferraro N. M.; Jardin S. C.
Source: PHYSICS OF PLASMAS **13** 092101 **(2006)**

Title: **Implicit solution of the four-field extended-MHD equations using high-order high-continuity finite elements**
Author(s): Jardin S.; Breslau JA
Source: PHYSICS OF PLASMAS **12** 056101 (2005)

Title: **A triangular finite element with first-derivative continuity applied to fusion MHD applications**
Author(s): Jardin, S.
Source: J. OF COMPUTATIONAL PHYSICS **200** 133-152 **(**2004)