Grid Generation with GRID

S. Lisgo

v1.0 15/10/12
v1.1 26/03/13

Overview

GRID, the creatively named IDL-based grid generator for the OSM-EIRENE-DIVIMP (OEDGE) code package, is presently located in FUSE, which stands for Fusion Unified Simulation Environment (TM James Harrison) and is an attempt to organize a range of data analysis and simulation tools. The long-term goal is to provide a high quality graphical user interface for processing experimental data, executing codes, and visualizing the results.

Note: Everything is still at a very early stage, including this memo. Many of the individual tasks will eventually be well described and script-driven, but are not at the moment. Welcome to “the beginning”.

Figure 1: Use at own risk.

Motivation

Wish I had some. *Experiment to simulation in 15 minutes* is our goal, and GRID is a central part of this effort, i.e. semi-automated grid generation.

Installing FUSE

Contact David Elder (`david@starfire.utias.utoronto.ca`) and request access to the University of Toronto code repository. Then:

```
> cd
> svn co http://starfire.utias.utoronto.ca/svn/fuse/fuse/trunk fuse
```

Add the following to your shell setup files:

If using `bash`, edit the `.bashrc` or `.mybasrc` in your home directory:

```
export FUSEHOME=<location of the “fuse” directory created above>
```

```
alias cdf='cd $FUSEHOME'
alias cdfd='cd $FUSEHOME_DATA'
alias cdfo='cd $FUSEHOME/src/osm'
alias cdfe='cd $FUSEHOME/src/eirene07'
alias cdfi='cd $FUSEHOME/input'
alias cdfl='cd $FUSEHOME/idl'
alias cdfs='cd $FUSEHOME_DATA/mast/shots'
alias cdfr='cd $FUSEHOME_DATA/results'
alias cdfc='cd $FUSEHOME_DATA/cases'

PATH=$PATH:$FUSEHOME/scripts ; export $PATH

IDL_PATH=\+$IDL_DIR/lib:+~/fuse/idl:+~/fuse/idl/utils ; export IDL_PATH
```

For `csh` or `tcsh`, edit the `.cshrc / .mcshrc  / .tcshrc / .mytcshrc`:

```
setenv FUSEHOME <location of the “fuse” directory created above>

alias cdf  cd $FUSEHOME
alias cdfo cd $FUSEHOME/src/osm
alias cdfe cd $FUSEHOME/src/eirene07
alias cdfi cd $FUSEHOME/input
alias cdfl cd $FUSEHOME/idl
alias cdfs cd $FUSEHOME/shots
alias cdfr cd $FUSEHOME/results
alias cdfc cd $FUSEHOME/cases

setenv PATH $PATH”:$FUSEHOME/scripts”
```

The `.tcshrc` file need to define `$FUSEHOME` even if you are using `bash` since the FUSE scripts use `tcsh`.

It is also necessary to add `$FUSEHOME/idl` and `$FUSEHOME/idl/utils` to your IDL path.

Then:

```
> cd
> source <shell setup file>      i.e. <shell setup file> = .bashrc, .tcshrc, etc.
> rehash
```

Now run a setup script to create a few directories:

```
> cdf
> cd scripts
> chmod u+x *
> fuse –setup <machine>            <machine> = cmod, d3d, east, etc.
```

Using GRID

Compile the magnetic equilibrium translation programs in `$FUSEHOME/src/equtrn`:

```
> fuse -make equtrn
```

Create a “shot” directory, where `<shot>` is an identifier for a group of data files, i.e. the grid, radial profiles from probe data, etc. – it can be anything, but again, good to use something related to the shot number (and maybe the time slice, unless you want to put data for many time slices into a single directory):

```
> mkdir $FUSEHOME/shots/<machine>/<shot>
```

Acquire the EFIT g-file for the shot and time of interest and put it into `<shot>`, i.e. `$FUSEHOME/shots/<machine>/<shot>`.

It is necessary to convert the `g-file` to an `equ` equilibrium file, which is the format used by the DG/CARRE grid generation tool used to make grids and configure runs for SOLPS. GRID has adopted this format as well. To convert the file run:

```
> fuse –equ <machine> <shot> <g-file> <equ>
```

Note that `<.equ file>` must not include the .equ file extension.

```
> fuse -equ cmod 1100303017 g1100303017.01380 cmod_1100303017_01380
```

The `.equ` files will appear in `<shot>`. Several files are generated that have different spatial resolutions, i.e. `<equ>.equ`, `<equ>.x2.equ`, `<equ>.x4.equ`, etc., where the larger the `x` number the higher the spatial resolution of the `equ` file. Using a file with higher spatial resolution when running GRID helps to produce smoother magnetic flux contours – and using a file where the resolution is too low will sometimes cause GRID to crash. Conversely, running with higher spatial resolution will slow exedcution, so I suggest that the `x2` file be used when previewing the settings for the radial grid boundaries (see below), and then use `x8` when building the grid, and maybe `x16` for a final version or if problems are encountered. To check the number of data points in a `equ` file:

```
> more <equ>.<resolution>.equ

    jm   :=  no. of grid points in radial direction;
    km   :=  no. of grid points in vertical direction;
    r    :=  radial   co-ordinates of grid points  [m];
    z    :=  vertical co-ordinates of grid points  [m];
    psi  :=  flux per radiant at grid points     [Wb/rad];
    psib :=  psi at plasma boundary              [Wb/rad];
    btf  :=  toroidal magnetic field                  [t];
    rtf  :=  major radius at which btf is specified   [m];

    jm    =          129
    km    =          129
```

Compile GRID:

```
> fuse –make grid
```

Run the grid generator with the `–preview` option to see where the outer radial boundaries of the grid will be:

```
> fuse –grid –preview <machine> <shot> <equ> <grid>

> fuse -grid –preview cmod 1100303017 cmod_1100303017_01380.x8.equ test
```

You cannot change the boundaries at the moment, but this will change “soon”, i.e. there will be an input file for GRID where you can set the options for grid generation.

Running with the default settings for now, generate the grid:

```
> fuse –grid <machine> <shot> <equ> <grid>

> fuse -grid cmod 1100303017 cmod_1100303017_01380.x8.equ test
```

You will see the IDL prompt if the grid generation fails. If not, the grid will be stored in the `<shot>` directory, .i.e. `$FUSEHOME/shots/<machine>/<shot>.`

Testing the Grid

OSM and EIRENE need to be compiled:

```
> fuse –make osm
> fuse –make eirene07
> fuse –make triangle
```

Edit `$FUSEHOME/input/grid_test.osm` so that the new grid is loaded:

```
…
'{GRID FILE} … '  '<grid>'  $ OPT%F_GRID_FILE
…
…'<wall>'  $ LIST OF LINE SEGMENTS IN A FILE
…
```

You can find `<wall>`, which is a continuous set of line segments that defines the wall of the divertor and main chamber, in `$FUSEHOME/idl/grid_input.pro`, i.e. it's `vessel_wall4.dat` for the C-Mod example case and is located in `$FUSEHOME/shots/cmod/default`.

Now run OSM-EIRENE and the post-processing code CORTEX (IDL based plotting) to test the grid, where `$FUSEHOME/input/test_grid.osm` and `$FUSEHOME/input/test_grid.ctx` are the full file names for `<osm>` and `<cortex>`, respectively.

```
> fuse –run <machine> <shot> <osm> <cortex>

> fuse -run cmod 1100303017 grid_test grid_test

.
```

To check that the grid didn't generate errors, assuming the case ran without crashing:

```
> cdfr
> grep –i trash <osm>.eirprn
```

where `.eirprn` is the EIRENE log file. You should see that `PTRASH` and `ETRASH` are zero:

```
TRASH: TEST PARTICLES KILLED DUE TO ERRORS
PTRASH=      0.0000E+00
ETRASH: ENERGY ABSORBED DUE TO ERRORS
ETRASH=      0.0000E+00
```

If there are errors, send me the input file and grid files (the grid, the .equ file, and the wall segment file) and I'll have a look.

If OSM-EIRENE runs without problems then a postscript file will have been generated by CORTEX. The name of the file is sent to the screen when the plots are finished (`Plot stored in file: […]`), but it is just `$FUSEHOME/results/<osm>.idl.ps`. Note that CORTEX is also in active development, i.e. is pre-alpha, and so not very pretty.

To run CORTEX by itself:

```
 > fuse –cortex <osm> <cortex>
```