III. IMPLEMENTATION OF THE GTNEUT 2D NEUTRALS TRANSPORT CODE FOR ROUTINE DIII-D ANALYSIS

(Z. W. Friis and W. M. Stacey, *Georgia Tech*; T. D. Rognlien, *Lawrence Livermore National Laboratory;* R. J. Groebner, *General Atomics*)

Abstract

The Georgia Tech Neutral Transport (GTNEUT) code[1,2] is being implemented to provide a tool for routine analysis of the effects of neutral atoms on edge phenomena in DIII-D. GTNEUT can use an arbitrarily complex two-dimensional grid to represent the plasma edge geometry[1]. The grid generation capability built into the UEDGE code[3], which utilizes equilibrium fitting data taken from experiment, is being adapted to produce geometric grids for the complex 2D geometries in the DIII-D plasma edge. The process for using experimental measurements supplemented by plasma edge calculations to provide the required background plasma parameters for the GTNEUT calculation will be systematized once the geometric grid generation is complete.

A. Introduction

In the past, most plasma physicists concentrated their research efforts on the exploration of core plasma, with little attention given to the edge region including the SOL, divertor and pedestal regions. This is now changing because many experiments have indicated that phenomena taking place in the plasma edge are very important for the overall performance characteristics of the confined plasma. Additionally, it has been shown that some of these edge phenomena are greatly influenced by neutral particles. It is for this reason we seek to better understand how neutral particles affect phenomena in the plasma edge.

In order to analyze the affects of neutral atoms on edge phenomena, an accurate but computationally efficient calculation of neutral particle transport in edge plasma is needed. For this purpose, the two-dimensional Georgia Tech Neutral Transport (GTNEUT) code[1] will be used. GTNEUT is a two-dimensional neutral particle transport code based on the Transmission and Escape Probabilities (TEP) method[2], which has been extensively benchmarked against both experiment and Monte Carlo calculations[4-7]. GTNEUT is computationally efficient compared with some of the standard Monte Carlo codes used today.

B. GTNEUT Geometric Input

While the GTNEUT code has many advantages over some of the Monte Carlo based codes, it does have one very big drawback. GTNEUT utilizes a coordinate-free geometry input file called "*toneut*". "Coordinate-free" means that the two dimensional mesh needed to represent a cross section of a Tokamak plasma only requires geometrical data from each cell such as lengths and angles, as well as relative positions of the sides of neighboring cells. The GTNEUT calculation consists of the calculation of the transmission of uncollided fluxes from an incident interface across a region through an exiting interface and the calculation of fluxes of collided particles exiting a region across a bounding interface as illustrated in the figure below. The interface balances on these various fluxes must be simultaneously solved to determine the fluxes, from which the densities within the region can be computed.

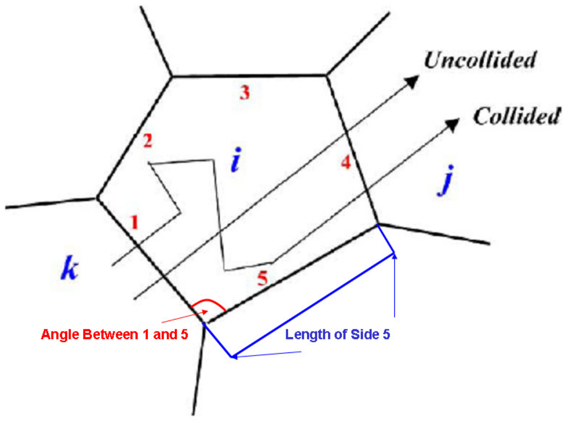


Figure 1: Schematic diagram showing region i and its adjacent regions and the partial currents at the interfaces. [1]

Actual (R,Z) coordinates are not needed and cannot be utilized by GTNEUT itself. Therefore, it can become quite tedious and error prone to manually input such information.

The GTNEUT package does include an automatic grid generator; however, this generator can only create very simple rectangular geometries which are often only used for test cases. For much more complicated geometries, such as a tokamak plasma edge, GTNEUT becomes dependent on the GRID generating capabilities of other codes. In fact, simply obtaining the GTNEUT geometric information for a tokamak plasma becomes a three step process. The first step is obtaining information about the plasma geometry from diagnostics, the second step is generating a mesh from the plasma geometry, and the third step is converting the mesh into a format that GTNEUT can utilize.

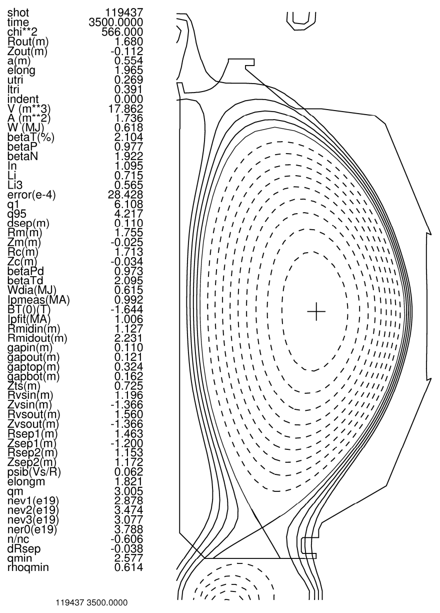
C. <u>EFIT</u>

Utilization of the DIII-D EFIT (Equilibrium Fitting) code is the first step in our process. The EFIT code was developed to translate measurements from plasma diagnostics into useful information like plasma geometry by solving the Grad-Shafranov equation. Such measurements are provided from diagnostics such as external magnetic probes, external poloidal flux loops, and the Motional Stark Effect (MSE)[8]. Running the EFIT code is a fairly simple procedure. One simply specifies the experiment number, the initial time slice to be studied, and the number of times to be studied as well as the time interval between each step. Additionally, one must specify which of the different SNAP versions are used.

Table 1:List of different SNAP versions, and their uses[9].

SNAP Version	Use
def	defaulted SNAP file, no edge gradients. For L-mode discharges, break-down error field analysis. Polynomial representation. No edge current.
j	finite edge gradients included in the current representation.
jt	Edge gradients constrained to vanish weakly. For H-mode discharges. Polynomial representation. Edge current is constrained to vanish weakly.
scrape	force-free scrape-off layer and vessel currents included in the fitting.
mses	For L- and H- mode discharges with MSE. Spline representation. Finite edge current allowed.
mse2_j1	MSE plus constrainted edge J.
mses_er	MSE with ER correction for shots later than 91000.

The code takes seconds to run and stores information about the plasma geometry in a number of files called "EQDSK" files. There are several types of EQDSK files, but the ones utilized most often are the AEQDSK and GEQDSK files. The AEQDSK file contains mostly scalar values as well as the global plasma parameters. The GEQDSK file holds most of the information about the flux surfaces and the R and Z positions[10]. Below is the output from a code designed to view the EQDSK files.

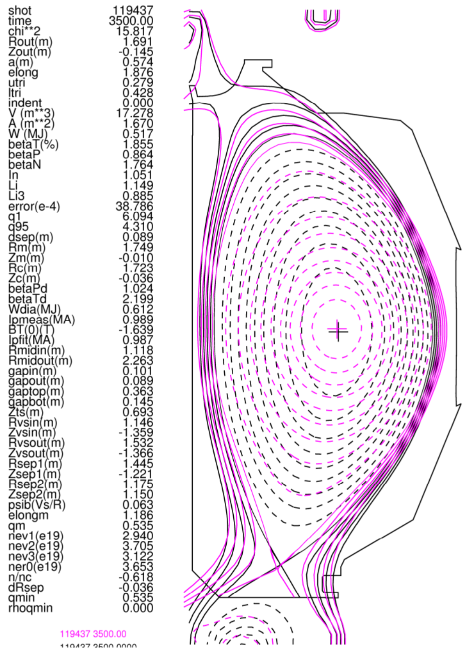


```
shot              119437
time           3500.0000
chi**2           566.000
Rout(m)            1.680
Zout(m)           -0.112
a(m)               0.554
elong              1.965
utri               0.269
ltri               0.391
indent             0.000
V (m**3)          17.862
A (m**2)           1.736
W (MJ)             0.618
betaT(%)           2.104
betaP              0.977
betaN              1.922
In                 1.095
Li                 0.715
Li3                0.565
error(e-4)        28.428
q1                 6.108
q95                4.217
dsep(m)            0.110
Rm(m)              1.755
Zm(m)             -0.025
Rc(m)              1.713
Zc(m)             -0.034
betaPd             0.973
betaTd             2.095
Wdia(MJ)           0.615
Ipmeas(MA)         0.992
BT(0)(T)          -1.644
Ipfit(MA)          1.006
Rmidin(m)          1.127
Rmidout(m)         2.231
gapin(m)           0.110
gapout(m)          0.121
gaptop(m)          0.324
gapbot(m)          0.162
Zts(m)             0.725
Rvsin(m)           1.196
Zvsin(m)          -1.366
Rvsout(m)          1.560
Zvsout(m)         -1.366
Rsep1(m)           1.463
Zsep1(m)          -1.200
Rsep2(m)           1.153
Zsep2(m)           1.172
psib(Vs/R)         0.062
elongm             1.821
qm                 3.005
nev1(e19)          2.878
nev2(e19)          3.474
nev3(e19)          3.077
ner0(e19)          3.788
n/nc              -0.606
dRsep             -0.038
qmin               2.577
rhoqmin            0.614

          119437 3500.0000
```

Figure 2: EFIT Data from shot 119437

There are several methods used run the EFIT code and some methods use different locations from where experimental data was obtained. Some methods use the diagnostic data on the MDSplus servers while others uses the data that has been reduced by experimentalist. For example, the diagram below shows two different EFITs for the same shot at the same time. The only difference is how the EFIT was generated. The EFIT equilibrium can be improved by adding extra measurements to the analysis. As the equilibrium is refined by the addition of data, the location of strike points and other divertor geometry may change in small but important ways, from the point of view of edge analysis. It is not clear if there is a "best" strategy for generating equilibria for edge analysis. Ascertaining which EFIT is correct requires collaboration with experimentalist.



shot	119437
time	3500.00
chi**2	15.817
Rout(m)	1.691
Zout(m)	-0.145
a(m)	0.574
elong	1.876
utri	0.279
ltri	0.428
indent	0.000
V (m**3)	17.278
A (m**2)	1.670
W (MJ)	0.517
betaT(%)	1.855
betaP	0.864
betaN	1.764
In	1.051
Li	1.149
Li3	0.885
error(e-4)	38.786
q1	6.094
q95	4.310
dsep(m)	0.089
Rm(m)	1.749
Zm(m)	-0.010
Rc(m)	1.723
Zc(m)	-0.036
betaPd	1.024
betaTd	2.199
Wdia(MJ)	0.612
Ipmeas(MA)	0.989
BT(0)(T)	-1.639
Ipfit(MA)	0.987
Rmidin(m)	1.118
Rmidout(m)	2.263
gapin(m)	0.101
gapout(m)	0.089
gaptop(m)	0.363
gapbot(m)	0.145
Zts(m)	0.693
Rvsin(m)	1.146
Zvsin(m)	-1.359
Rvsout(m)	1.532
Zvsout(m)	-1.366
Rsep1(m)	1.445
Zsep1(m)	-1.221
Rsep2(m)	1.175
Zsep2(m)	1.150
psib(Vs/R)	0.063
elongm	1.186
qm	0.535
nev1(e19)	2.940
nev2(e19)	3.705
nev3(e19)	3.122
ner0(e19)	3.653
n/nc	-0.618
dRsep	-0.036
qmin	0.535
rhoqmin	0.000

Figure 3 : Overlap of different EFITs for Shot 119437.

D. UEDGE Mesh Generation

Once we have generated the EFIT, the second step in the process is generating the mesh. Manually doing this can take several months. For example, the mesh depicted below was created by hand using a CAD program to calculate the lengths and angles of each cell.

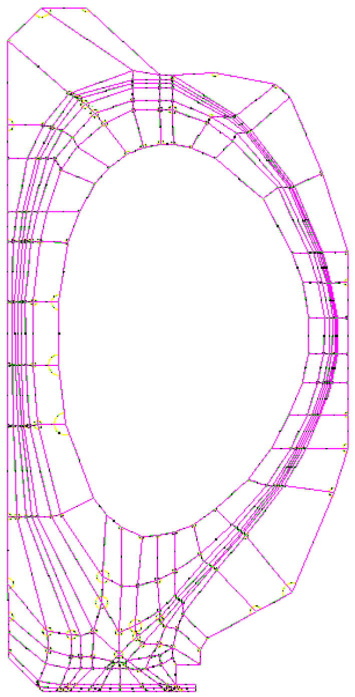


Figure 4: Manually Generated Mesh

While the manual method of grid generation may actually have some advantages such as being able to specify geometries at certain locations in more detail, the process is inherently cumbersome, prone to error, and very tedious. Instead of manually generating the mesh, we have opted to use the UEDGE code's mesh generating capabilities. UEDGE is a very powerful two-dimensional (2D) fluid transport code for collisional edge plasmas. UEDGE can perform a large number of calculations and even be coupled to several Monte Carlo based neutrals codes[3]. However, for now, we are primarily interested in UEDGE's grid generating capabilities.

UEDGE generates meshes using the EQDSK files specified in a previous section. Also, an input file is required to specify how coarse the mesh will be. Of most use to us are the inputs below.

Table 2: Inputs to specify UEDGE grid[3].

Input Name	Purpose
nxleg(1,1)	Number of Poloidal mesh pts from inner plate to x-point
nxcore(1,1)	Number of Poloidal . mesh pts from x-point to top on inside
nxcore(1,2)	Number of Poloidal mesh pts from top to x-point on outside
nxleg(1,2)	Number of Poloidal mesh pts from x-point to outer plate
nysol(1)	Number of Radial mesh pts in SOL
nycore(1)	Number of Radial mesh pts in core

By default, the mesh generator produces orthogonal meshes; however, it can be very useful to generate non-orthogonal meshes. This is especially true if one wants to fit the mesh to the divertor. By altering an input option called “ismmon” and specifying the divertor plate locations in the UEDGE input file, the grid generator will extend the mesh to the divertor producing a nice fit. An example of this can be seen below[3].

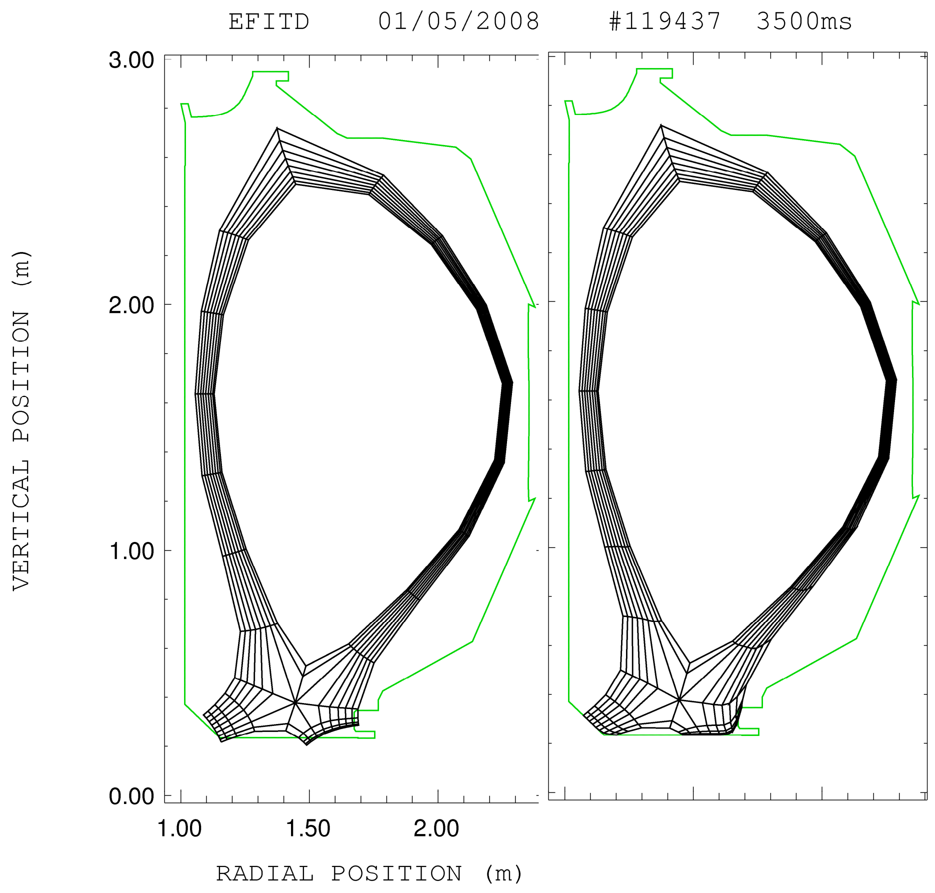


Figure 5: Comparison of Orthogonal and Non-Orthogonal Meshes

Additionally, a full UEDGE run is not required in order to generate the meshes. Typing the following at the UEDGE command prompt will generate a mesh file called *gridue*[3].

```
call flxrun
call grdrun
```

The mesh produced by the UEDGE grid generator can be very useful for GTNEUT calculations; however, as seen in the examples above, the grid does not extend to the walls of the confinement vessel. In between the SOL and Wall (which we call the Gap region), it is necessary to extend the UEDGE grid to the wall for the GTNEUT grid.

E. Adaptation of UEDGE Mesh for GTNEUT Input

For simplicity, the most efficient way to do this is simply extending the last layer of cells in the SOL perpendicularly to the wall. Below is an example of what we have done.

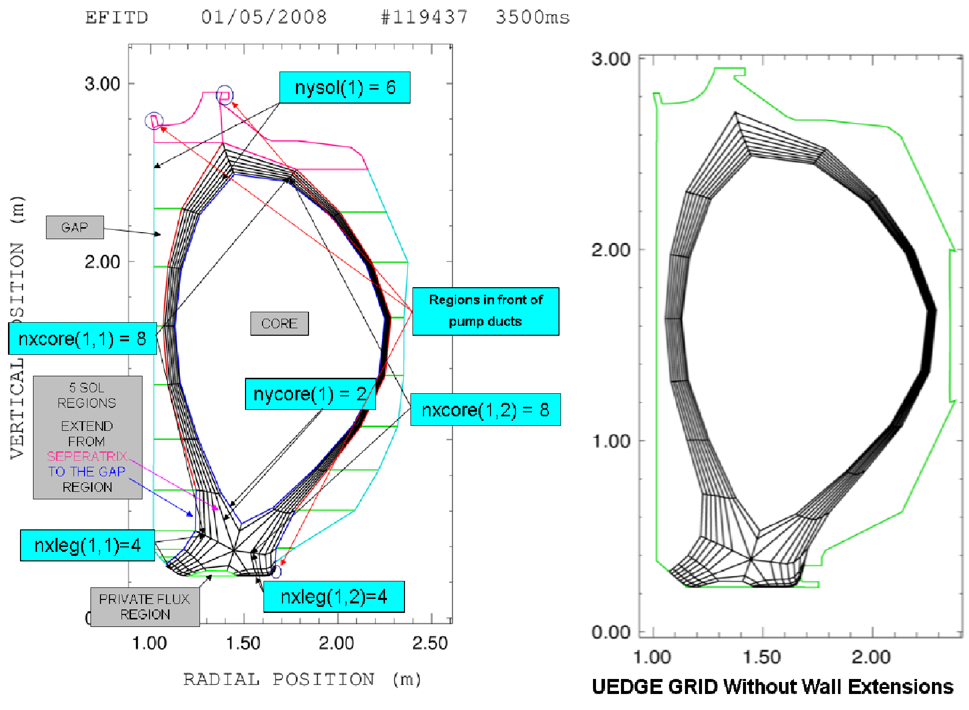


Figure 6: Comparison of UEDGE grid extended to wall.

By, examining what we have done here, we can also see how the UEDGE input file works. Notice there are six SOL regions on the right hand figure (SOL regions are the regions that lie outside of the seperatrix. This was accomplished by setting nysol(1) equal to 6. The left hand figure only has five, plus the Gap region. We have simply redefined the last region of cells. Most of the cells in our grid have four sides. The two main exceptions are the private flux region show in lime green on the left side. It is defined as a function of the UEDGE input file and *gridue* file. Also, the cells shown in magenta at the very top of the vessel may have more than 4 sides. Lastly, the outermost corners of the divertor regions may contain only 3 sides depending on their location with respect to the wall. As of now, this is not a purely automatic grid generating system. One must first plot the grid to make sure errors have not arisen before proceeding.

F. Summary and Conclusions and Present Work

The present version of the UEDGE to GTNEUT grid adaptor is much more efficient and accurate than the manual method. It has been successfully used on discharges 119437 and 119436. It still needs to be tested on other discharges using different EFIT versions. Also, the present version of the adaptor only works for single lower null discharges. Modifications to work on a single upper null or double null discharge should not be too difficult.

Presently, routines are being added to the adaptor to actually write the GTNEUT input file. This currently exists for the plasma regions between the core and the gap regions. However, proper tracking of the cells is of the utmost importance. We are breaking the grid down into regions as illustrated by various colors in the diagram on the left in Fig. 5 to facilitate this tracking more easily. Additionally, the cells are being numbered in a way that will make assigning temperatures and densities from diagnostics much easier.

References

1. J. Mandrekas, Comp. Physics Commun., 161, 36 (2004).
2. W. M. Stacey and J. Mandrekas, Nucl. Fusion, 34, 1385 (1994).
3. T. D. Rognlien, et al., *User Manual of UEDGE Edge-Plasma Transport Code,* LLNL report (2007)
4. W. M. Stacey, J. Mandrekas and R. Rubilar, Fusion Sci. Technol., 40, 66 (2001).
5. R. Rubiliar, W. M. Stacey, J. Mandrekas, Nucl. Fusion, 41, 1003 (2001).
6. J. Mandrekas, R. J. Colchin, W. M. Stacey, et al., Nucl. Fusion, 43, 314 (2003).
7. D-K. Zhang, J. Mandrekas and W. M. Stacey, Phys. Plasmas, 13, 062509 (2006).
8. http://fusion.gat.com/theory/Efitdef
9. http://fusion.gat.com/theory/Efitsnap
10. http://fusion.gat.com/theory/Efitoutputs