

**A SEMI-AUTOMATED PROCEDURE FOR CREATING GEOMETRY AND BACKGROUND
PLASMA INPUT FILES FOR THE GTNEUT 2D NEUTRAL PARTICLE TRANSPORT CODE
USING THE UEDGE PLASMA EDGE CODE**

Z.W. Friis

**Fusion Research Center
Georgia Institute of Technology
July, 2009**

I. INTRODUCTION

The GTNEUT code¹⁻³ can provide a computationally economical and accurate calculation of neutral particle transport in the complex 2D geometry to the tokamak divertor, scrapeoff layer and edge regions inside of the separatrix. However, the GTNEUT code requires a geometric input file describing this geometry and a background plasma file describing the plasma density and temperature in the various geometric regions. This report describes semi-automated procedures: i) for using (and extending) the input preparation capability of the UEDGE⁴⁻⁵ code to generate a GTNEUT geometric grid input file from a DIII-D EFIT file and ii) for using the plasma density and temperature distributions calculated with the UEDGE code to generate the background plasma input file for GTNEUT.

II. BACKGROUND ON GTNEUT

Before reading any further, if you would like to use the GTNEUT code, you should familiarize yourself with the following journal article:

J. Mandrekas, "GTNEUT: A code for the calculation of neutral particle transport in plasmas based on the Transmission and Escape Probability method", *Computer Physics Communications* 161, 36 (2004).

The above journal article is the only manual for GTNEUT, and it is for an older version of GTNEUT freely available through the ntc repository hosted by PPPL.

In the current version of GTNEUT, there are a couple of new runtime options and a few inputs that are no longer needed. However, it would be advantageous to read the above journal. The manual explains in depth many of the features that will only be briefly touched upon.

III. DISCLAIMER ABOUT GTNEUT

Those wanting to use the Georgia Tech Neutrals code (GTNEUT) should first understand two very important issues that arise when attempting to create the GTNEUT geometry input file called "toneut".

First: Although simplified geometric input can be input manually to GTNEUT, the GTNEUT code does NOT possess intrinsic mesh generation capabilities for complex 2D geometries such as those found in a Tokamak plasma edge. Additionally, the GTNEUT methodology requires the use of a "region-free" grid structure. This means traditional grid points are not needed for GTNEUT and cannot be utilized in neutral particle transport calculations.

Second: GTNEUT does NOT possess the ability to generate its own plasma background parameters. This means, GTNEUT is dependent on the use of other codes or directly observable experimental values for the ion and electron temperatures and densities, recombination neutrals, ion fluxes to the walls, gas puffing rates, and pumping locations.

The following paper describes a procedure for generating geometric and background plasma input files for GTNEUT by utilizing the 2D fluid code UEDGE. While this generation of compatible GTNEUT and UEDGE geometric and plasma parameter files does NOT constitute a coupling of the two codes, it is a step in that direction.

UEDGE is a very powerful 2D fluid code that has the capability of generating very complex 2D meshes based on equilibrium fitting data files (EFIT). Additionally, UEDGE calculates a plasma background throughout the 2D mesh. We have created routines that can convert the 2D mesh generated by UEDGE into a format that can be directly imported into the geometry section of the GTNEUT input file called "toneut". Additionally, if a UEDGE solution has already been obtained, it is possible to extract the plasma background information into a format usable by GTNEUT as well.

The importation of experimental data into the GTNEUT geometric mesh to use as background plasma parameters still must be done manually.

IV. THE GTNEUT *TONEUT* FILE

As stated above, GTNEUT requires an input file called "toneut" to run.

In the toneut file there are essentially 4 sections: geometry, plasma background, sources for neutrals, and runtime options.

V. GEOMETRY SECTION

The geometry section is arguably the most complicated section of the input file. It contains the geometric definition of the problem. The TEP method is unique in that it does not require a clearly defined coordinate system. Instead, cell information is made up of the lengths and angles of the cell interfaces. Additionally, neighboring cells must be tracked. The greatest challenge in creating an automated input procedure was devising a way of tracking cell lengths, angles, and cell neighbors. GTNEUT has a built-in routine that checks the geometry section of the input file for consistency.

The figure below is a very simplified illustration showing how we number the GTNEUT grid. The UEDGE mesh is essentially laid out in a poloidal and radial distribution.

The number of cells in the poloidal direction is referred to as NX. It is defined (starting at cell one and going in the clockwise direction) as $nxleg(1,1) + nxxpt + nxxpt + nxcore(1,1) + nxcore(1,2) + nxxpt + nxxpt + nxleg(1,2)$.

A better explanation of these terms are located in the UEDGE manual, but $nxleg(1,1)$ and $nxleg(1,2)$ are the number of cells in the poloidal direction for the inner and outer divertor strike points respectively.

Likewise, $nxcore(1,1)$ and $nxcore(1,2)$ are the number of poloidal cells for the regions above the X-Point. Resolution to the X-Point region may be added by increasing $nxxpt$. This creates $4 * nxxpt$ number of cells around the X-Point. Various options are available to determine how closely these extra cells are clustered around the X-Point.

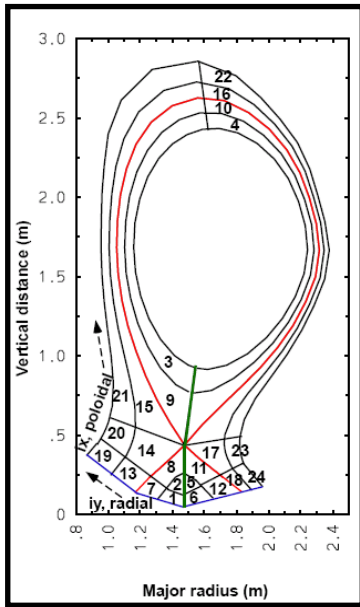


Fig 1: Simplified Geometry of UEDGE Mesh

The number of cells in the radial direction (NY) is the summation of $nycore(1)$ and $nysol(1)$. $nycore(1)$ is the number of poloidal layers within the separatrix. $nysol(1)$ is the number of poloidal layers outside the separatrix.

The conversion of numbering from the 2D array of UEDGE coordinates to the GTNEUT 1D array of cells is given by:

$$icn = ix + NX*(iy-1)$$

In total, there are $NX*NY$ cells in the UEDGE grid. This numbering scheme was suggested by Tom Rognlien.

The UEDGE grid, however, requires some modifications before it can be used in GTNEUT.

First, the grid should be fit to the divertor for all GTNEUT runs. Additionally, UEDGE imposes boundary conditions at the edge of the grid that are not extended to the wall. The UEDGE grid does not extend to the walls of the confinement vessel. The routines we have created extend the UEDGE mesh to the wall. Intrinsic to the UEDGE calculation, there is a dummy layer of cells along the last layer of cells in the SOL. We call this the HALO region.

When GTNEUT utilizes this layer of cells, there are effectively $(NY+1)*(NX)$ number of cells in the GTNEUT grid. The same formula

$$icn = ix + NX*(iy-1);$$

can be used to locate cells in term of icn . NY is now given by $NY = NY + 1$.

Lastly, the GTNEUT mesh also needs to fill in the gap in the private flux region with computational cells. These cells are slightly different. All of the other $NX*(NY+1)$ cells are 4 sided. The cells in the private flux region are 3 sided. The number of cells is determined by the sum of $nxleg(1,1)+nxxpt+nxxpt+nxleg(1,2)$.

In total there are:

$$NX*(NY+1)+nxleg(1,1)+nxxpt+nxxpt+nxleg(1,2)$$

cells in a GTNEUT calculation. Additionally, the number of plasma regions and the number of wall locations must be tracked. The number of plasma regions is simply

$$nxxpt+nxcore(1,1)+nxcore(1,2)+nxxpt$$

The number of wall locations is given by:

$$NX + 2* NY + 2$$

The additional 2 wall segments are from the 2 cells touching the wall in the private flux region.

The numbering scheme for the plasma and wall cells can appear complicated.

However, it is fairly straight forward. The plasma numbering scheme starts from the total number of cells in GTNEUT. So, if there are

$$NX*(NY+1)+nxleg(1,1)+nxxpt+nxxpt+nxleg(1,2)$$

cells in the problem, and the first plasma region is:

$$NX*(NY+1)+nxleg(1,1)+nxxpt+nxxpt+nxleg(1,2) +1$$

The first plasma location is located adjacent to the first internal cell bordering the plasma region. The numbering likewise goes in the clockwise fashion.

The wall regions start at:

$$NX*(NY+1)+nxleg(1,1)+nxxpt+nxxpt+nxleg(1,2) + nxxpt+nxcore(1,1)+nxcore(1,2)+nxxpt$$

The numbering starts at the wall location immediately adjacent to internal cell 1. In many of the GTNEUT input options, the wall locations are labeled from 1 to # of wall segments. Be aware, in this numbering convention, wall segment 1 is located adjacent to cell 1. The wall segments increase linearly from 1 to # of wall locations in a clockwise fashion until it reaches the wall segment bordering the first private flux region cell.

It should be noted, we are unaware of what problems this redefining of the NY+1 may cause in the evaluation of an actual UEDGE run. All of our calculations have been run in post. It is conceivable this may modify the UEDGE calculation if the two codes are truly coupled. The script is very simple to modify. It should be trivial to create copies of the UEDGE grid locations used in our UEDGE to GTNEUT mesh converter, and only extend the copies of the UEDGE NY+1 layer of cells to the wall.

VI. PROCEDURE TO CREATE GEOMETRY SECTION

The first step in creating the GTNEUT geometry section is obtaining the EFIT files you would like to use to create a grid from. The steps to do this are quite simple:

1st: Run IDL.

2nd: From the IDL command prompt, generate the “a” and “g” EQDSK files by issuing the commands:

```
writea,shotnumber,timeslice,runid  
writeg,shotnumber,timeslice,runid
```

Next, export the EQDSK files into a directory of your choosing. You will need a UEDGE template to generate the mesh. Templates can be found in my directory:

/u4/friis/uedge/runs/UEEDGE_DEMO

Several templates have the extension, “.template”. You will also need several of the files in this folder in order to simply run UEDGE. As previously stated, all UEDGE runs need to be run with the divertor fitting option. When running the mesh generation routines, keep in mind the geometry of the divertor shelf, and insure the correction divertor configuration is used.

Several options are available to create a large variety of UEDGE meshes. Some of these options may cause the make_toneut script to break down. Only a limited number of test cases were carried out.

After the template file has been set up and the EFIT files have been generated, UEDGE may be run.

It should be noted that the versions of UEDGE and GTNEUT utilized are all located on the linux cluster at GA. Zeus was primarily used for most of the calculations; however, since Zeus’ demise, Delphi has been used with success.

The version of UEDGE used for all of our run is located in:

/d2/uedge/Ver_5.0b_linux/dev/lrx-2.3-i32/bin/xuedge

The version of GTNEUT used is located in:

/u4/friis/GTNEUT/

The routine used to create the convert the UEDGE mesh into a GTNEUT mesh can be found in:

/u4/friis/uedge/GTNEUT_DEMO

and it is called:

make_toneut

Before the script `make_toneut` can be run, the UEDGE mesh must be created. This can be accomplished by running UEDGE using the executable provided above and then typing:

read "template file name"

This routine sets up the UEDGE file. Next, the UEDGE grid must be created. Instead of actually running UEDGE, this is accomplished by typing:

**call flxrun
call grdrun**

This simply makes the UEDGE grid. One can view the mesh by typing

win on

followed by

read plotmesh

If the newly created mesh has been generated successfully, the `make_toneut` routine may be called by typing

read make_toneut

By running this, you have just extended the mesh to the wall and generated several outputs.

To view the new GTNEUT mesh, you may type

read plotgtneutmesh

This routine is located in my post directory within the UEDGE directory. It will create a very color depiction of the new GTNEUT mesh.

The output files created from `make_toneut` are primarily the geometry files.

There are 5 output files. Table 1 breaks down the output files from `make_toneut`.

The `toneut-cells`, `toneut-cores`, and `toneut-walls` contain all of the pertinent geometric data.

`toneut-flatflux` create a uniform temperature and density distribution throughout the computational grid. This isn't really useful, except for making sure the geometry is set up properly.

Lastly, `toneut-rwall_gex` contains the run options.

Table 1: make_toneut output files.

<code>toneut-cells</code>	contains all iType(0) cells and needed information.
<code>toneut-cores</code>	contains all iType(1) regions and tracks neighboring cells.
<code>toneut-walls</code>	contains all iType(2) regions and tracks neighboring cells.
<code>toneut-flatflux</code>	gives all of iType(0) and iType(1) cells a uniform temperature and density.
<code>toneut-rwall_gex</code>	generates the end of the toneut file with specified options.

To test the geometry, use the unix `cat` command to make the `toneut` file:

cat toneut-cells toneut-cores toneut-walls toneut-flatflux toneut-rwall_gex > toneut

Run GTNEUT using the executable provided previously.

If there is a problem with the geometry, it will be instantly obvious. If GTNEUT starts running though, simply cancel the run the using. There is no point in running GTNEUT with the flatflux approximation. It may not converge to a solution.

If GTNEUT ran under the flatflux run, this means the geometry section should be correct.

Now the plasma background is needed.

VII. THE PLASMA BACKGROUND SECTION

As previously stated, a plasma background is needed for GTNEUT. It is up to the reader of this procedure to generate their own plasma background. Any number of codes can be used, but keep in mind the numbering scheme for the cells generated by converting the UEDGE mesh to a GTNEUT grid.

Fortuitously, if UEDGE has been run a plasma background has been generated already. A plasma background exist for the cells numbering 1 through $NX \cdot NY$ (in this case NY is the original NY , not $NY+1$).

For cells in the Halo and Private Flux Regions, the user must specify a plasma background. For the analysis this procedure was developed for, a uniform plasma background was used for each of these regions.

Additionally, the user must also specify the plasma temperatures and densities for the albedo boundary condition at the plasma core.

A routine to perform all of these tasks has been created, and is located in the same UEDGE_DEMO folder. To call the routine, just type:

read gtneut_temp_dense

at the UEDGE command prompt. The script is very simple and easily modifiable to suit the needs of the user.

The script creates a fill called:

GTNEUT-TEMPS-DENSE

This file contains all of the plasma background properties.

Additionally, sources of neutrals are needed for GTNEUT.

VIII. THE SOURCE SECTION

There are 3 sources of neutrals GTNEUT can use (show in the table below) and from recombination sources, gas puffing sources, and the recycling source.

S_ext	Volumetric Source (due to recombination)
g_ex	gas puffing
g_ion	ions recycling off the wall

Luckily, the recombination source is based off the plasma background and is calculated by UEDGE. The recombination source is different from the other two sources in the sense that there can be a source at every cell in the mesh. The other sources are wall sources.

To extract the volumetric source, use the command:

gtneut_volume_source

which creates the file “recombination.dat”.

The recombination source from this file does not include the Halo and private flux region recombination sources. Those have to be calculated by the user.

Unfortunately, new routines have NOT been created to extract the recycling source from UEDGE or the gas puffing source.

It should be fairly straight forward to define the gas puffing source; however, for the discharges analyzed using this procedure, gas puffing was turned off. In theory, one just has to determine the number of particles being puffed into the system, and assign them a specific wall location. This procedure is described in Mandrekas’ manual.

The ion recycling source for our analysis came directly from experiment and had to be manually imported into the input file.

It shall be left up to the user to determine the best way to ascertain the recycling source for their problem.

IX. RUNTIME OPTIONS

Run-time options should be specified once the other sections are complete. The original GTNEUT manual contains descriptions of most of the options. The only new option in the current version of GTNEUT is the option to run GTNEUT with the ANE approximation.

To do this, the number of ANE iterations must be greater than 1. The author of the ANE approximation indicates that typically 3 are sufficient:

neitr = 3

It is important to note, if use the ANE approximation the following option must be used.

i_e0=3

This initializes the neutral energies to 1.5 Ti, which is needed for the ANE approximation. More information on the science behind this method can be found in Dingkang Zhang’s PhD thesis:

http://etd.gatech.edu/theses/available/etd-04132005-165515/unrestricted/zhang_dingkang_200505_phd.pdf

X. EXECUTING GTNEUT

Once you’ve obtained the background and the neutrals sources, once again use the cat command to create the toneut file:

```

cat toneut-cells toneut-cores toneut-walls
GTNEUT-TEMPS-DENSE recombination.dat
halo_recom.dat g_ions.dat toneut-rwall_gex >
toneut

```

For the example in the UEDGE_DEMO folder, we have included the experimentally observed recycling measurements: **g_ions.dat**.

Now that you have the toneut file you can run GTNEUT using the executable found in:

/u4/friis/GTNEUT

Running GTNEUT will produce the same outputs described in Mandrekas' paper. Additionally, a file called "nmat.m" is created after each run. This is a matlab file containing output from GTNEUT. If matlab is utilized for the analysis, this file could be very important. The information in nmat.m was specifically designed for a particular analysis; however, by looking at the output.f file in the GTNEUT folder, one can easily tell what the different outputs are.

Lastly, a file called "loadnd" has been created.

This file can be called from the UEDGE command prompt using:

read loadnd

This file essentially creates a number of variables within the UEDGE run from the output of GTNEUT. It is left up to the reader to determine the best way to do this, but it is primarily used for plotting purposes.

XI. GRAPHICAL INTERFACING

Several scripts that use the data from loadnd have been created. They are located in the file:

/u4/friis/uedge/post

To run any of these files, just ensure that you've typed "win on", and the type "read script name".

The script names are giving in the table below.

Script Name	Plots the Quantity
plotpoly_nd	Neutral Particle Density
plotpoly_ir	Ionization Rates
plotpoly_id	Ionization Densities
plotpoly_cx	Charge Exchange Fractions
plotpoly_cxr	Charge Exchange Rates
plotpoly_ne	Electron Density
plotpoly_ni	Ion Density
plotpoly_te	Electron Temperature
plotpoly_ti	Ion Temperature
plotpoly_mpf	Mean Free Path

To plot the correct temperatures and densities, you have to make sure you have updated the quantities in the halo region. Unfortunately, these routines will not plot data in the private flux region; however, very little is going in the GTNEUT's private flux region so it is not that important.

XII. CAUTIONARY REMARKS

The procedure described here works fairly well provided UEDGE has been run; however, this procedure only works for Lower Single Null cases. One should always plot the meshes to insure the make_toneut script has created a decent mesh. The script has only been tested for a set number of cases. The adjustment of geometric parameters for the part of make_toneut that defines the very top of the mesh near the upper baffle or the first and last cells of the HALO region may be required to create a working GTNEUT mesh. The same is true for the GTNEUT private flux as well.

XIII. ACKNOWLEDGMENTS

The author of this procedure would like to express his deepest gratitude to Dr. Richard Groebner of General Atomics and Dr. Tom Rognlien of LLNL. This procedure could not have been developed without their guided support during the early stages its creation.