Adam Griffith
Rice University

ORNL/FED
August 22, 2002

Introduction to RANT3D - The Three-dimensional Recessed Antenna Model

This document describes the background and use of the RANT3D antenna modeling code.

Authors: Adam Griffith[1], Mark Carter[2], Phil Ryan[2], Kate White[3], David Swain[2]
Keywords: RANT3D, antenna modeling, GLOSI, GuiRecess, fusion

[1] Rice University, Houston, Texas
[2] Oak Ridge National Laboratory, Fusion Energy Division
[3] University of Tennessee, Knoxville

TABLE OF CONTENTS

Introduction 3
The Recess Model 3-4
Versions of the Code 4-5
Inputs 5-8
    impede.plt 5-6
    recess_input 6-7
    current_element_input 7-8
    no_divj_source 8
GuiRecess 8-9
Running RANT3D 9-12
Output 12-13
    rant.info 12
    recess#x 12
    recess_geometry and recess_geometry.fld 13
    strap_impedance 13
Diagnostics 13-14
    rant_diag 13-14
Sample Input Files 14-16
    recess_input 15-16
    current_element_input 16
Figures 17-18
Acknowledgements 18-19
References 15

LIST OF FIGURES

Figure 1 - The NSTX RF Antenna 17
Figure 2 - The RANT3D Model of the NSTX Antenna 17
Figure 3 - NSTX Geometry Diagram 18

INTRODUCTION

The design of RF antennas for heating and current drive in tokamaks requires more than simple trial-and-error and guesswork methods. Such a sophisticated and complex piece of equipment must be carefully thought out before being built and tested. However, the physics behind such antennas is too complex for pencil and paper; computational models must be used. RANT3D is one of these models, and it was used extensively along with the GLOSI and PICES codes during the development of the high-harmonic fast wave (HHFW) RF antenna for the National Spherical Torus Experiment (NSTX). It has also been used to model antennas in DIII-D, TFTR, Tore Supra, and other tokamaks.

RANT3D solves Maxwell's equations for the geometry of the antenna, which is specified in Cartesian coordinates through a series of rectangular recesses and current straps. These recesses work as a series of connected rectangular waveguide sections. The code is used in conjunction with GLOSI, which creates a plasma impedance matrix for the edge of the tokamak plasma. The GLOSI/RANT3D combination results in accurate power coupling and RF field spectra, which are useful in analysis of antenna design and effectiveness.

Please direct RANT3D questions to Marc Carter (cartermd@ornl.gov), or Phil Ryan (ryanpm@ornl.gov).

THE RECESS MODEL

RANT3D creates a 3d model of an antenna by using recesses and current elements or straps to model the shape of the antenna. While the real antenna is curved in both the poloidal and toroidal directions, the RANT3D antenna model follows standard three-dimensional Cartesian coordinates. That is, all elements in the RANT3D model are rectangular boxes and planes; there are no curved elements. The periodic matching to the plasma assumes a uniform gap between the current strap and plasma, suitable for modeling antennas that closely conform to the plasma shape. For elements curving with respect to the backplane, image currents in the backplane can be approximated by using recesses with different depths behind sections of the current strap. More recesses can be used to better approximate the real antenna; however, this will result in longer computations.

Additionally, while the antenna's straps have thickness, and sometimes have rounded edges, RANT3D's model assumes that straps are two-dimensional rectangles with no thickness or rounding of the edges. By default, the current distribution across each strap is uniform; therefore, when the user considers the RANT3D code results for the mutual coupling between straps, he/she should remember that the self-inductance calculation (proportional to the imaginary part of the diagonal terms) may not be accurate. The strap loading and strap-to-strap mutual coupling should be reasonable accurate. The plasma contribution to the strap-to-strap coupling matrix can be isolated by performing a second RANT3D calculation with vacuum radiation boundary conditions, and then subtracting the result from the calculation performed with plasma.

The figures at the end of this document show the real NSTX antenna, a graphical representation of that antenna based on input to RANT3D, and a diagram of the geometry used by the NSTX antenna. As can be seen, the RANT3D model's current straps are flat, not curved, and the antenna array is also flat in the model. (Again, the boundary condition for matching to a periodic slab implicitly assumes a uniform gap between the plasma and the antenna structure.) Behind each strap a set of three recesses can be seen. This is actually formed from one large recess and two smaller ones. There is one recess that is the same height as the current strap. Then there are two recesses that are deeper than the first that are at the top and the bottom of the first recess. This creates a perfectly conducting boundary condition that is closer to the antenna in the middle, and farther away from the antenna at top and bottom. This approximates the curvature of the strap away from the back wall of its box, as can be seen in the picture of the real antenna. It can also be seen that the recesses model the septa between the current straps. There is an additional recess that is harder to distinguish in front of the current straps. Thus, the NSTX antenna is reasonably modeled with three recesses per strap and one for the entire array, or thirty-seven recesses. Determining exactly how to use recesses to model an antenna is tricky and may require testing several different arrangements before determining the correct arrangement to use.

VERSIONS OF THE CODE

RANT3D was written using an object oriented coding style in FORTRAN 90. The primary version of the code was compiled on the IBM AIX platform, which is a UNIX

operating system. The code was recently ported to Linux, which makes it much more accessible to many users. There are both 32 bit and 64 bit versions of the code for AIX, which allows for varying degrees of processors and amounts of memory. The code can also be run in multi-processor mode under AIX, which allows RANT3D to be run on multi-processor machines and supercomputers. This can dramatically reduce the amount of time needed to run the code, especially for high-resolution cases.

Two changes were made in the code recently to decrease computation time. The first was the implementation of Kronecker product algebra for calculating strap impedance matrices. The second change was to analytically solve Fourier transforms and replace these methods in the code. Timings by Kate White show that the Kronecker product implementation reduced the computation time primarily for multiple strap antenna arrays, while the analytic Fourier solutions reduced the time more for smaller antennas. The combination of both improvements resulted in significant time decreases for parallel processing cases.

INPUTS

RANT3D uses four input files to properly set up and analyze an antenna. The first file is *impede.plt*. This file is created by the GLOSI ICRF heating code and describes a set of impedance properties for given plasma. If the *impede.plt* file is absent, the RANT3D code prompts the user for periodicity scale-lengths and two other possible boundary conditions: radiation to infinity in vacuum, or a perfectly electrically conducting (pec) boundary at a specified distance from the antenna. For more information on GLOSI, please see "Introduction to GLOSI" or contact Mark Carter (cartermd@ornl.gov). The second input file is *recess_input*, which describes the geometry and properties of the antenna recesses. The third input file is *current_element_input*, which describes the characteristics and geometry of the current straps. The fourth and final input is a file called *no_divj_source*.

IMPEDE.PLT

This file, as mentioned above, is produced by GLOSI. It is tokamak- and plasma-specific, but not antenna-specific. That is, any number of different antenna setups can be tested with the same *impede.plt* file. The only time when *impede.plt* must be recomputed is if the tokamak or plasma parameters are changed. Usually, the same *impede.plt* file is used for dozens

of RANT3D antenna calculations. Due to its size, it does not make sense to copy the file to several directories; therefore, a symbolic link to the file may be placed in the directory for a specific RANT3D calculation instead. An example of symbolic linking will be given below in the section on running RANT3D. Once the file has been copied or linked to, it need not be altered in any form to make it usable to RANT3D.

RECESS_INPUT

This file describes the recesses used in the antenna model by RANT3D. The *recess_input* file is one of a pair of files that defines the antenna geometry, antenna properties, and program parameters; the other is *current_element_input*. Please refer to "Introduction to GuiRecess" for assistance in setting up these files. The *recess_input* file has several inputs for each recess that must be properly set to ensure accurate RANT3D calculations. Each recess has its own section in the file, as can be seen in the sample shown later in this document. The file follows the standard RANT3D orientation of x-, y-, and z-coordinates representing radial, poloidal, and toroidal directions. An explanation of all of the inputs for each recess is given below. All position or distance inputs are measured in meters.

a, *b* - The first pair of inputs are labeled *a*, and *b*. These define the range or depth of the recess, which is in the x or radial direction. The first number, *a*, is the front position of the recess; it is the point closest to the plasma, with 0 being at the plasma surface. The second number, *b*, is the position of the rear of the recess.

ay, *bz* - The second pair of inputs are labeled *ay*, and *bz*. They define the poloidal height and toroidal width of the recess, respectively.

qcy, *qcz* - This third pair of inputs define the periodic boundary coordinates for the lower left corner of the recess. These essentially position the recess in space relative to all other recesses. For example, a recess with width .2 and height .8 centered at (0,0) would have (qcy, qcz) = (-.1,-.4). <u>For recesses, everything is relative to the lower left corner.</u>

yboundary - This input determines the boundary condition in the poloidal direction. Its value may either be *waveguid* or *periodic*. <u>The default value (that should not be changed) is *waveguid*.</u>

zboundary - This input determines the boundary condition in the toroidal direction. Its value may also either be *waveguid* or *periodic*. <u>The default value (that should not be changed) is *waveguid*.</u>

mmax, *nmax* - This pair of inputs sets the number of modes in the poloidal and toroidal directions respectively. A higher number results in more resolution, but also results in a longer amount of time needed for the computation. A faster computation may be carried out with a lower number of modes, but with the penalty of reduced resolution. The user must determine the relative importance of speed and accuracy in the RANT3D computation to determine where to set these numbers. The GuiRecess program will provide a rough guess as to how many modes should be used for each recess, but this number should not be taken as an absolute. In general, the modes of the recess closest to the plasma should be based on the dimensions of the smallest resolvable object. That is, if there are small objects such as gaps in the antenna that need higher resolution, then more mode numbers should be used for the front recess. Recesses positioned behind the current straps may have more relaxed resolutions without sacrificing accuracy.

CURRENT_ELEMENT_INPUT

This file describes the current elements used in the antenna model by RANT3D. As with *recess_input*, *current_element_input* has several parameters that must be set properly for each strap, with each strap having its own section of the file. Again, please refer to "Introduction to GuiRecess" for assistance in setting up this file. The file follows the standard RANT3D orientation of y-, and z-coordinates representing poloidal, and toroidal directions. An explanation of all of the inputs for each strap is given below. All position or distance inputs are measured in meters. Keep in mind that, according to the RANT3D model, the straps are infinitely thin 2d sheets.

feeder type, *traveling wave parameter* - The *feeder type* parameter can have values of 0, 1, and -1, which correspond to constant, standing, and stop input, respectively. The *traveling wave parameter* gives a half standing wave for the value of 0.0.

plasmad - This parameter is the distance from the plasma to the strap, essentially the radial position of the strap. The plasma is located at 0, and the units are in meters.

amp - This parameter specifies the amount of current on the strap in amperes.

tphaz - This parameter specifies the phase of the strap current <u>measured in radians</u>. For instance, if an antenna array with a phase shift of 90° between adjacent straps, this value should be set to 0 for the strap in the first position, $\pi/2$ for the second strap, π for the third strap, etc. Note that the <u>*tphaz* is input in terms of radians, not degrees</u>! The value of *tphaz* should also be

between -2π and $+2\pi$, or -360° and 360°. For example, if a value of *tphaz* should be 2π for a certain strap, replace the value with 0, which is an equivalent angle. A value of $5\pi/2$ would be replaced by $\pi/2$.

yc, *zc* - This pair of numbers represents the poloidal and toroidal center of the strap. The positioning of current strap is determined by the strap center, unlike recesses, which are referenced by the lower left corner.

theta - This number is the tilt of the strap in degrees. A 90° *theta* value corresponds to straps that are vertical.

width, *height* - This pair of numbers describes the full width and full height of the strap. Since straps are infinitely thin, there is no corresponding *depth* value.

phazi, *phazs* - The first of this pair is the ratio of the speed of light to the phase velocity along the antenna, v_p. A value of 0 for phazi corresponds to an infinite phase velocity or uniform current over the strap. Otherwise, it corresponds to $c/v_p = \beta c/\omega$. The second value is the location of ground. For instance, if a strap is centered at (0,0) with a height of .8 and is grounded at the bottom, *phazs* would have a value of -0.4. The strap current achieves a maximum at the ground point specified by *phazs*.

NO_DIVJ_SOURCE

This file contains no data. The file itself is a switch recognized by the RANT3D program, which can be used to simulate a Faraday screen. If a file *no_divj_source* is present, then the charge on the current strap caused by finite phase velocity is hidden from the plasma by effectively shunting the current to the back wall ($\nabla \cdot \bar{J} = \rho \equiv 0$). If the *no_divj_source* file is absent, the plasma sees the charge ($\nabla \cdot \bar{J} = \rho$) on the strap specified by the phase velocity given to the element in the *current_element_input* file, leading to capacitive coupling. The phase velocity with plasma but without a Faraday screen can be estimated in the present RANT3D code release by iteratively adjusting the phase velocity until the electric field along the current strap becomes small, but this iteration must be performed manually. For most calculations *no_divj_source* should be present.

GUIRECESS

Because the setup of the *recess_input* and *current_element_input* files is so time-consuming, tedious, and error-prone, a graphical user interface (GUI) has been written to assist in the preparation of these two files. GuiRecess is a program written using the Java 2 Standard Edition v. 1.3.1 language specification, which ensures its usability on a wide range of processors and operating systems. The program allows a user to draw recesses and current elements with a mouse and then specify the parameters of those recesses and elements. The program will then output the necessary *recess_input* and *current_element_input* files for use with RANT3D. This program eliminates much of the time and frustration of preparing the input files by hand and also limits errors. For information on GuiRecess, which also has other features, please see "Introduction to GuiRecess."

RUNNING RANT3D

Once the input files have been properly created, running RANT3D is fairly easy. To run a case, place all of the input files in a directory and then make that directory the active directory by using the **cd** command. The RANT3D executable can then be run by typing its name if the location of the executable is in the user's PATH. If the executable is not in the user's PATH, it can be executed by writing out the full path of the executable relative to the active directory. It is highly recommended that the output be directed to a file, rather than the screen by adding **> filename** after the RANT3D execution command. Persons who are more familiar with UNIX may also use the **nohup** and **time** commands to catch the output, and may wish to run the program in the background using the **&** command. RANT3D may take a long time to finish its calculations, depending on the size of the antenna being analyzed. Operation on non-standard UNIX and Linux platforms may be substantially different. An example of the setup and execution of RANT3D is given below. Items in ***bold italic ten-point Courier New*** indicate commands that are written by the user. Items in `regular Courier New` are displayed by the computer. This script was produced in a UNIX environment.

```
Script command is started on Tue Aug 20 09:54:28 EDT 2002.
$ cd rant
$ ls
SHOT_107899            current_element_input  description.txt
no_divj_source         recess_input
$ mkdir demonstration
$ cp current _element _input demonstration/
```

```
$ cp recess _input demonstration/
$ cp no _divj _source demonstration/
$ cp description.txt demonstration/
$ cd demonstration
$ ls
current_element_input  description.txt        no_divj_source
recess_input
$ ln -s ~/glosi/SHOT _107899/FULL _MODES/impede.plt impede.plt
$ ls
current_element_input  description.txt        impede.plt
no_divj_source         recess_input
$ nohup time rantv4 &
[1]     25152
$ Sending nohup output to nohup.out.
$ ls
current_element_input  impede.plt             nohup.out
recess_input           description.txt        no_divj_source
rant.info
$
[1] + Terminated               nohup time rantv4 &
$ head nohup.out
@(#)RUN TIME IS:>Tue Aug 20 09:57:45 EDT 2002
@(#)RUN HOST IS:>carter
@(#)RUN DIRECTORY IS:>/home/arj/rant/demonstration
@(#) >@(#)See rant.info file for code details >
 &GLOSI_RANT3D
 FRF=30000000.0000000000, KY1=-73.8636363636363740,
KY2=73.8636363636363740, NKY=131, KZ1=-63.2791242169208346,
KZ2=63.2791242169208346, NKZ=201, RMAJ=1.58030000000000004,
RMIN=0.880000000000000004, BZ0=4.00000000000000000,
0.000000000000000000E+00, 0.000000000000000000E+00,
0.000000000000000000E+00, 0.000000000000000000E+00,
0.000000000000000000E+00, 0.000000000000000000E+00,
0.000000000000000000E+00, 0.000000000000000000E+00,
0.000000000000000000E+00, IBZPROF=20, BY0=0.000000000000000000E+00,
IBYPROF=20
 /
 mdim= 65 , ndim= 100
 12  actual current elements read
  number of attempts to read recess information =  38
$

Script command is complete on Tue Aug 20 09:59:17 EDT 2002.
```

An explanation of the commands used is given below.

1) **ls -** This command lists the contents of the directory. In this case it shows that the directory contains the *current_element_input*, *no_divj_source*, and *recess_input* files. There is also a *description.txt* file that the user wrote explaining what conditions were used for the calculation. This file is not necessary to run the program, but something of this sort is very helpful for organization purposes.

2) **mkdir demonstration -** This command (**mkdir**) creates a directory (**demonstration**) that RANT3D will be run in.

3) **`cp (filename) (directory)`** - The next four commands follow this format. The command (**`cp`**) copies the specified file (**`filename`**) into the directory (**`directory`**). In this case, the input files are copied into the *demonstration* directory.
4) **`cd demonstration`** - This command (**`cd`**) changes the active directory to the *demonstration* directory (where all of the inputs have been copied).
5) **`ls`** - This command now lists the contents of the *demonstration* directory, since that is now the active directory. This shows that the input files have indeed been properly copied.
6) **`ln -s ~/glosi/SHOT_107899/FULL_MODES/impede.plt impede.plt`** - This long command is broken into four parts. The first part (**`ln`**) is the command to make a link. The second part (**`-s`**), called a flag, makes the link a symbolic link. The third part (**`~/glosi/SHOT_107899/FULL_MODES/impede.plt`**) is the location of the *impede.plt* input file relative to the active directory. The final part (**`impede.plt`**) is the name of the symbolic link file to be created. In this case the name of the symbolic link is the same as the name of the linked file.
7) **`ls`** - Again listing the directory contents, it can be seen that the *impede.plt* symbolic link appears the same as any other file.
8) **`Nohup time rantv4 &`** - This command is actually four separate-but-linked commands. The first (**`nohup`**) is a command that outputs the results of the following commands to the *nohup.out* file and allows the user to logout without killing the process. The second command (**`time`**) keeps track of how long the following command has been running, both in user time and real time. The command **`rantv4`** executes the RANT3D program. The name of this command is the name of the RANT3D executable file, which may be called *rant*, *rant64*, *rantv4*, or perhaps yet another name. If the RANT3D executable is not in your PATH, then this command will need to be specified with the relative location of the executable file. Although not displayed, the *rantv4* executable used in this script is located at *~/bin/rantv4*. Finally, the **`&`** command tells the operating system that the previous commands are to be run in the background, allowing the user to continue work inside the same terminal. At this point the RANT3D program is running. The amount of time required for the computation to finish will depend on the user's setup.
9) **`ls`** - This command shows that immediately after executing RANT3D, the *rant.info* file has been produced, along with the *nohup.out* file from the **`nohup`** command.

10) **`head nohup.out`** – After the program has been terminated, the user uses the **`head`** command to display the first few lines of the *nohup.out* file. This command, along with **`more`**, **`cat`**, and **`tail`**, is a useful command for viewing text files without leaving the command prompt.

Obviously, some things will change from the above depending on what the user wants to do and where files are located; however, this should provide a general example of how to use RANT3D. Users familiar with UNIX and Linux commands will most likely already know everything shown above. Those unfamiliar with the commands may wish to consult a guide on working in UNIX or may use the **`man`** command to display information (a manual) on commands. For example, typing **`man nohup`** provides a guide on the **`nohup`** command. There is also a command, **`man man`**, which displays information on the **`man`** command.

OUTPUT

While RANT3D only uses four input files, it can produce numerous output files. A brief description of each type of output is given below.

RANT.INFO

The *rant.info* file gives a description of the version of RANT3D being run. It also describes the latest updates and modifications. However, in the current version of the code, this file is obsolete. The file is produced, but the information in it may not be correct.

RECESS#X

RANT3D produces a *recess#x* file for each recess in *recess_input*, where x is the position of the recess in *recess_input*. There is also a *recess#0* file that is for the entire antenna array, somewhat corresponding to a recess containing the plasma. Each file contains information on one recess that can be used to analyze the properties of that part of the antenna. However, it is usually the *recess#0* file that is used for diagnostics, since it contains information about the entire antenna. Some of these files can get large due to the amount of information processed. For the NSTX RANT3D cases being run at the time this document was written, the *recess#0* file was about 6 MB, *recess#1* was about 1 MB, and all of the other files were about 15 or 26 Kb. Additionally, these files are encoded and require another program for analysis.

RECESS_GEOMETRY AND RECESS_GEOMETRY.FLD

The *recess_geometry* and *recess_geometry.fld* files are for use with the AVS diagnostics program. If one has knowledge of this program, and the proper version, these files may be useful. Otherwise, these files are not used.

STRAP_IMPEDANCE

The final output of RANT3D is the *strap_impedance* file. This file contains an n x n impedance matrix for an array of n straps, with both real and imaginary terms, which can be used in diagnostics. The file is in text format and can be read using any text editor. This file is especially useful for strap loading calculations. The coupling matrix is ordered according to the order of the current elements specified in the *current_element_input* file. The equation for impedance is show below:

$$V' = \overline{Z}\frac{dI'}{dt}$$

The calculations required to interpret and use the *strap_impedance* file/matrix are **very** complicated and should only be done with care and if the user knows what he/she is doing.

DIAGNOSTICS

An accompanying program, *rant_diag*, is used to perform power balance diagnostics on any recess. Additional information is produced concerning the launched power spectrum and the electric field components in the recess when used on the *recess#0* output file.

RANT_DIAG

This program performs diagnostics on any *recess#x* file, although *recess#0* is the file normally used. To use it, simply execute the program by typing **`rant_diag`** at the prompt while in the directory containing *recess#0*. The program will request the recess file name, at which point the user types in `recess#0` or another *recess#x* file if desired. The *rant_diag* program can only be used if it is run in the directory containing the *recess#x* file to be used. The program will begin computing various meaningful results from the *recess#0* file. It will then ask for the field line angle in radians, after which it will continue with computations. At

then end it will ask if a pices dump is requested. This option will produce a file to be used in the PICES code. Don't worry about keeping track of the output displayed on the screen; it is recorded into a file.

While *rant_diag* is working, it produces numerous output files. The majority of these files are spectra of one form or another. The program produces about forty **.sds*, or Scientific Data Sheet, files that describe various physical properties of the antenna. There are also several **.dat* and **.dx* files produced. These files can be used in the Data Exchange program. Two **.pos* files are also produced. The PICES dump is in a file named *pices_wall*, which can provide an electric field boundary condition that is mapped onto the flux coordinate system used in the PICES code. The file *poynt.out* contains the information displayed on-screen when *rant_diag* is run. Part of this information is the Poynting (total power) flux passing through the recess boundary. In the case of *recess#0*, this corresponds to the amount of power that reaches the plasma. Finally there are two output files, *fort.10* and *fort.9*, that give the power spectrum for the different Fourier modes analyzed. A brief listing of the **.sds*, **.dat*, **.dx*, and **.pos* files is given below. *#x* refers to the *recess#x* file used by *rant_diag*. At the present time, many of the following files are in the process of being reviewed in an attempt to update the code and output for use with more recent viewer software.

0ecess#x*.sds, 1ecess#x*.sds

- * = mdbx, mdby, mdbz
- * = mdeb, mdep, mdex, mdez
- * = phbx, phby, phbx
- * = pheb, phep, phex, phey, phez
- * = px, py, pz

recess#x*.sds

- * = mdey, pxmn, pymn, pzmn

imag*.dat, imag*.dx, real*.dat, real*.dx

- * = bz0, bz1
- * = ey0, ey1

realbz0.pos, realey0.pos

SAMPLE INPUT FILES

Samples of the *recess_input* and *current_element_input* files are displayed below. They have been taken from a calculation performed on the NSTX antenna. What is shown is actually only a small portion of the total input files. Because of the size and complexity of the NSTX antenna, the full input files are quite long.

RECESS_INPUT

The first four recesses for the NSTX antenna are shown. These correspond to the front recess for the whole antenna and the three recesses for the first strap. There are thirty-three more recesses described in the full *recess_input* file.

```
# opening closest to plasma
  enter range for this recess, closest x value first (a,b)
 0.006 0.055
  enter height (poloidal) and width (toroidal) of cavity, (ay,bz)
 0.94  2.72
  enter periodic recess coordinates for lower left corner (qcy,qcz)
 -0.47  -0.0925
  enter boundary condition type for poloidal direction (yboundary)
  enter either waveguide or periodic
 waveguid
  enter boundary condition type for toroidal direction (zboundary)
  enter either waveguide or periodic
 waveguid
  enter number of modes in y' and z' directions (mmax,nmax)
 20 52

 # SEPTUM box 1
  enter range for this recess, closest x value first (a,b)
 0.055 0.0810
  enter height (poloidal) and width (toroidal) of cavity, (ay,bz)
 0.94  0.175
  enter periodic recess coordinates for lower left corner (qcy,qcz)
 -0.47  0.1075
  enter boundary condition type for poloidal direction (yboundary)
  enter either waveguide or periodic
 waveguid
  enter boundary condition type for toroidal direction (zboundary)
  enter either waveguide or periodic
 waveguid
  enter number of modes in y' and z' directions (mmax,nmax)
 12 8

 # top recess to model curvature away from backplane
  enter range for this recess, closest x value first (a,b)
 0.0810 0.140
  enter height (poloidal) and width (toroidal) of cavity, (ay,bz)
 0.304  0.175
  enter periodic recess coordinates for lower left corner (qcy,qcz)
 0.166  0.1075
  enter boundary condition type for poloidal direction (yboundary)
  enter either waveguide or periodic
 waveguid
  enter boundary condition type for toroidal direction (zboundary)
  enter either waveguide or periodic
 waveguid
  enter number of modes in y' and z' directions (mmax,nmax)
 6 8
```

```
# bottom recess to model curvature away from backplane
 enter range for this recess, closest x value first (a,b)
0.0810 0.140
 enter height (poloidal) and width (toroidal) of cavity, (ay,bz)
0.304  0.175
 enter periodic recess coordinates for lower left corner (qcy,qcz)
-0.47  0.1075
 enter boundary condition type for poloidal direction (yboundary)
 enter either waveguide or periodic
waveguid
 enter boundary condition type for toroidal direction (zboundary)
 enter either waveguide or periodic
waveguid
 enter number of modes in y' and z' directions (mmax,nmax)
6 8
```

CURRENT_ELEMENT_INPUT

The first three current straps from the NSTX antenna are shown below. There are nine additional straps not shown from the *current_element_input* file. These straps display a negative 30° (or -0.5236 radian) phasing. This input file was obviously not used for a *strap_impedance* calculation because phazi is set to 1.8181.

```
 enter feeder type (0 constant, 1 standing, -1 stop input)
   and traveling wave parameter (0.0 gives half standing wave)
0 0.00
 enter distance from plasma to strap in meters, plasmad
0.0194
 enter amp,tphaz,yc,zc,theta (theta in degrees)
1.00  0.0  -0.02  0.195 90.0
 enter width,height,phazi,phazs
0.085 0.90 1.8181 -0.47

 enter feeder type (0 constant, 1 standing, -1 stop input)
   and traveling wave parameter (0.0 gives half standing wave)
0 0.00
 enter distance from plasma to strap in meters, plasmad
0.0194
 enter amp,tphaz,yc,zc,theta (theta in degrees)
1.00  -0.5236  -0.02  0.390 90.0
 enter width,height,phazi,phazs
0.085 0.90 1.8181 -0.47

 enter feeder type (0 constant, 1 standing, -1 stop input)
   and traveling wave parameter (0.0 gives half standing wave)
0 0.00
 enter distance from plasma to strap in meters, plasmad
0.0194
 enter amp,tphaz,yc,zc,theta (theta in degrees)
1.00  -1.0472  -0.02  0.585 90.0
 enter width,height,phazi,phazs
0.085 0.90 1.8181 -0.47
```

FIGURES

Figure 1 - The NSTX RF Antenna

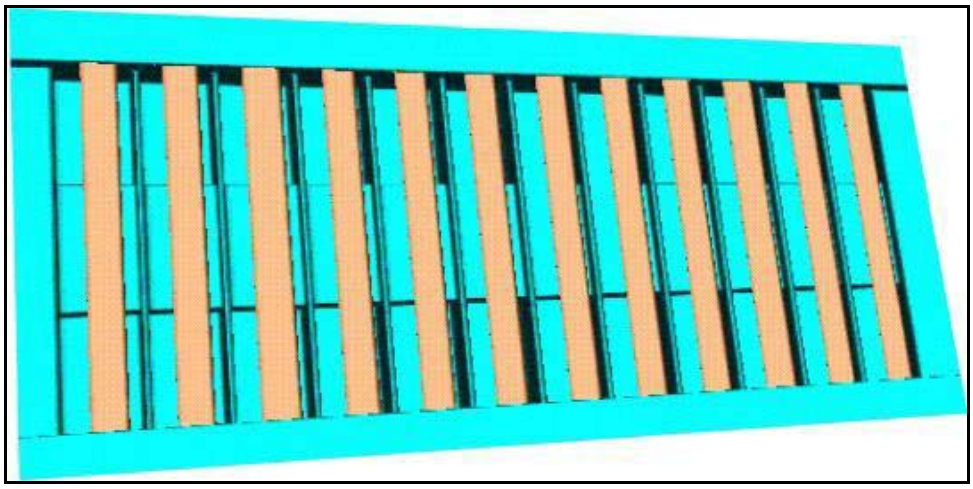

Figure 2 - The RANT3D Model of the NSTX Antenna

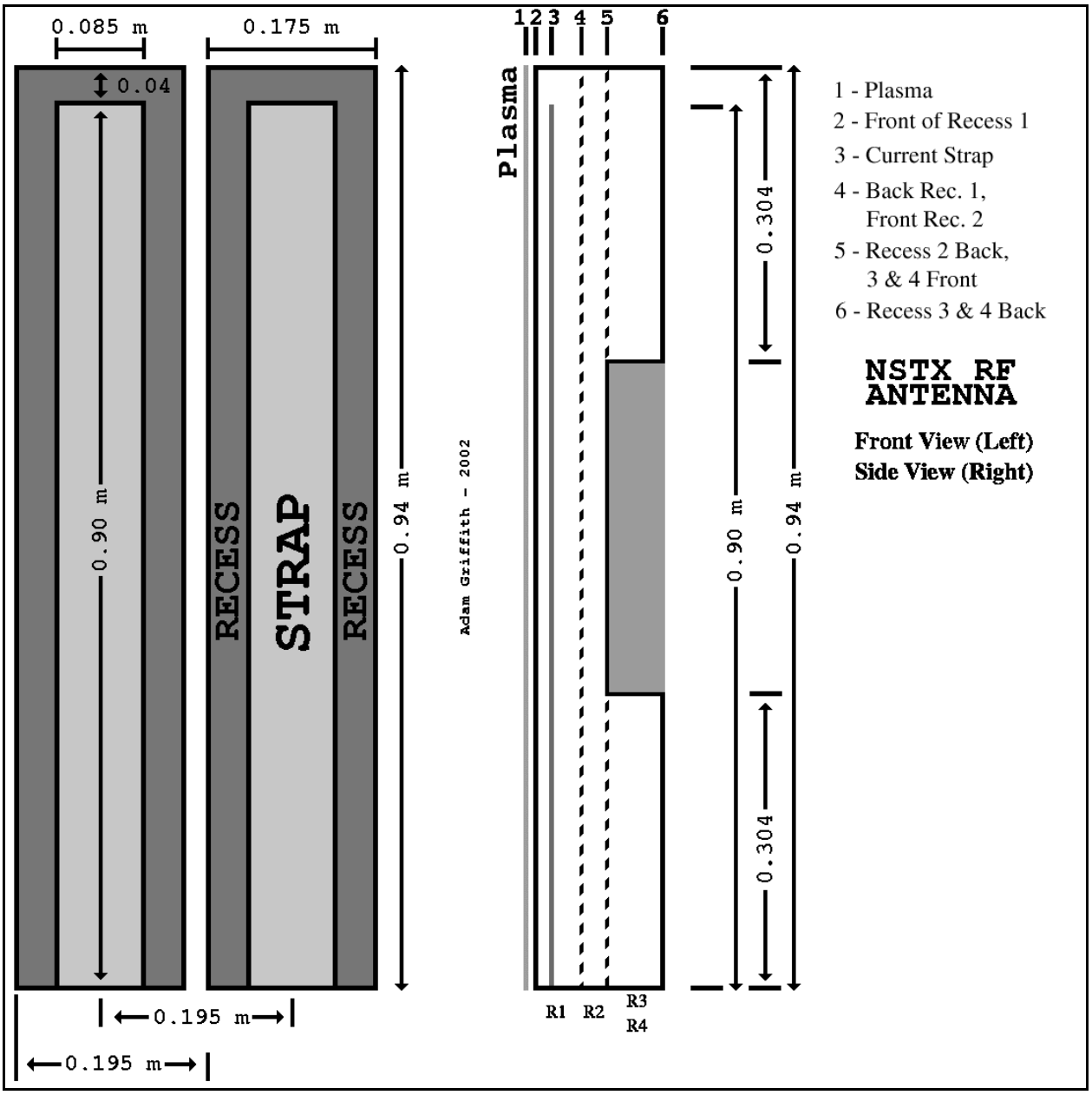

Figure 3 - NSTX Geometry Diagram

ACKNOWLEDGEMENTS

This document was written by Adam Griffith (argriffi@rice.edu *) under the direction of Mark Carter (cartermd@ornl.gov) and Phil Ryan (ryanpm@ornl.gov). Kate White (kate@utk.edu) and David Swain (swaindw@ornl.gov) provided additional input. Adam Griffith is an undergraduate student at Rice University, Houston, Texas, who participated in

the Higher Education Research Experiences at Oak Ridge National Laboratory 2002 Summer session. Phil Ryan, Mark Carter, and David Swain all work at the Fusion Energy Division of the Oak Ridge National Laboratory. Kate White is an undergraduate student at the University of Tennessee, Knoxville, who participated in the DOE Energy Research Undergraduate Laboratory Fellowship 2002 Summer session. Both Adam and Kate were assigned to the Fusion Energy Division for their terms.

* This email will be changed to argriffi@alumni.rice.edu shortly after Adam's graduation in May, 2003. If argriffi@rice.edu does not work, try argriffi@alumni.rice.edu.

REFERENCES

Carter, M.D., *et al.*, 1995. "Three-dimensional Physics of ICRF Launchers For Fusion Devices," *Nuclear Fusion*, vol. 36, no. 2, 1996, pp. 209-23.

Carter, M.D., D.B. Batchelor and D. C. Stallings 1993. *Three Dimensional Effects for Radio Frequency Antenna Modeling*, ORNL/TM-12464.

Carter, M.D., *et al.*, 1998. *Radio Frequency Current Drive Considerations for Small Aspect Ratio Tori*, ORNL/TM-13576.

Carter, M.D., P.M. Ryan and D.W. Swain 1998. "Expectations for the High Harmonic Fast Wave System for NSTX," *Proc. Annu. Meet. Am. Nucl. Soc./ 13th Top. Conf.*, Nashville, TN, June 7-11, 1998.

Griffith, A.R., "Introduction to GLOSI - The Global and Local One-dimensional Solution for ICRF Heating," Aug. 2002 (unpublished).

Griffith, A.R., "Introduction to GuiRecess v 1.0 - The RANT3D Input Java Graphical User Interface," Aug. 2002 (unpublished).

White, K.F., "Improving the Usability of Physics Modeling Codes," Presented at ERULF Poster Session, Aug. 2002.