GEOM: Discrete Geometric Mapping for Toroidal Devises

H. McGuinness[†], E. Strumberger[‡]

[†]Rensselaer Polytechnic Institute, Nuclear Engineering Program
Troy, New York, U.S.A.

[‡]Max-Planck-Institut für Plasmaphysik, IPP-Euroatom Association
85748 Garching, Germany

IPP Report 5/109 October 2004

Contents

Abstract

The GEOM code creates a discrete, gridded geometric map of a toroidal device. The user defines the positions of the First Wall, Last Closed Magnetic Surface, divertor and baffle plates, and how many partitions the device space should be divide into. GEOM outputs a three dimensional array of the device geometry. The output of GEOM serves as an input to a modified version of the GOURDON code, which with this output, significantly decreases run time for a GOURDON job.

1 Introduction

The GEOM code (GEOmetric Mapping), developed at the Max Planck Institute for Plasma Physics, was created in order to decrease the run time of a version of the well known GOURDON code.[1] The output of GEOM serves as an input to a modified and extended version of the original GOURDON which traces field lines and guiding centers.[8][1] GEOM was first utilized to study divertor issues of the Wendelstein 7-X stellarator and has since been modified for studying divertor issues of NCSX.

The GEOM code works in the following manner. The computational region in which GOURDON will trace field lines is partitioned into a grid. Each grid point is then given a numerical tag which relates to the geometry of the device being studied. The tags represent different sections of the geometry. Different sections are, for example, the area between the First Wall (FW) and outside the Last Closed Magnetic Surface (LCMS), the area enclosed by a divertor and baffle plates, the boundary of the LCMS, etc.

This tagged geometric map of the computational region of interest to GOURDON serves as an input to GOURDON. Every time GOURDON traces one step further GOURDON queries the GEOM map to see if the field line is close to the boundary of a Plasma Facing Component (PFC). If it is close to one of these surfaces then an extensive numerical procedure is cared out to determine if indeed the field line in that one integration step crossed the boundary, and if so, where exactly it crossed. Using the GEOM map in the GOURDON code saves GOURDON from making this extensive calculation every time it steps through a tracing, thereby significantly speeding up tracing time.

GEOM fits into a series of codes which begin with the calculation of the vacuum field from the coil geometry and currents with the VACFIELD code and ends with the tracing of field lines and guiding centers with GOURDON. Figure 1 shows input and output dependencies of codes close to GEOM in chain. Assuming nested flux surfaces, fixed or free-boundary three-dimensional equilibria are computed with the NEMEC code for given pressure and rotational transform profiles or pressure and toroidal current profiles.[2] [3] GEOM takes as input the Fourier representation of the LCMS and the FW and the geometry of the other PFCs to create the tagged geometric map of the region of interest. GOURDON then uses the field calculated by MFBE and the geometric map to trace field lines inside and outside the LCMS.[5] [6]

GEOM's role in the larger system code is to decrease the time GOURDON spends tracing. Through faster tracing the deposition patterns of field lines and particles on PFCs such as divertor and baffle plates can be determined more quickly, allowing for a larger number of structural and magnetic field geometries to be considered while designing a particular machine.

[1] The code was modified by Dr. Strumberger of IPP. In GOURDON there is the option for any particular run of the code to trace field lines or guiding centers. Therefore, in this document, it will be understood that "tracing field lines" means the same as tracing field lines and guiding centers.

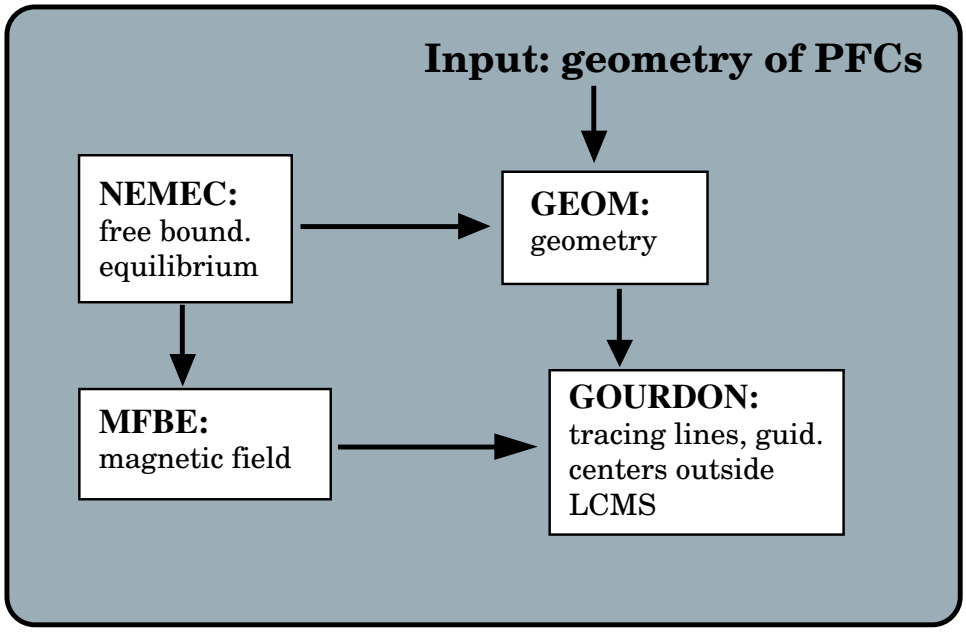

Figure 1: Part of the code system related to GEOM and their interdependencies.

2 Numerical Details

2.1 Description of the Grid

The grid area must be large enough to at least cover the region which GOURDON will trace over, and must be fine enough so that field lines which actually are not close to a surface do not trigger the computationally expensive procedure of calculating if a surface has been crossed. By convention the grid is in cylindrical coordinates, centered around the coordinate (R_0, Z_0), and has radial width of $2\Delta R$ and vertical height of $2\Delta Z$. Neither the central coordinate nor the width or height of the box vary from toroidal cross section, and it must be made sure that the region of interested is covered with this constraint.

The grid is broken up into N_R radial, N_Z vertical and N_ϕ toroidal sections. Consequently the three dimensional array "index", which stores the tag for each point as a function of position, has dimensions index(N_R, N_Z, N_ϕ). "Side" and "top" profiles of the grid are shown in figure 2.

2.2 Tag Determination

2.2.1 Line Parameterization

Once the grid has been created each point in the grid must receive a numerical tag which represents the point's relation to the geometry of the device and plasma surface. Tags are meant to distinguish the points which make up different sections of the geometry, such as the region outside the LCMS or inside a region closed in by divertor and/or baffle plates,

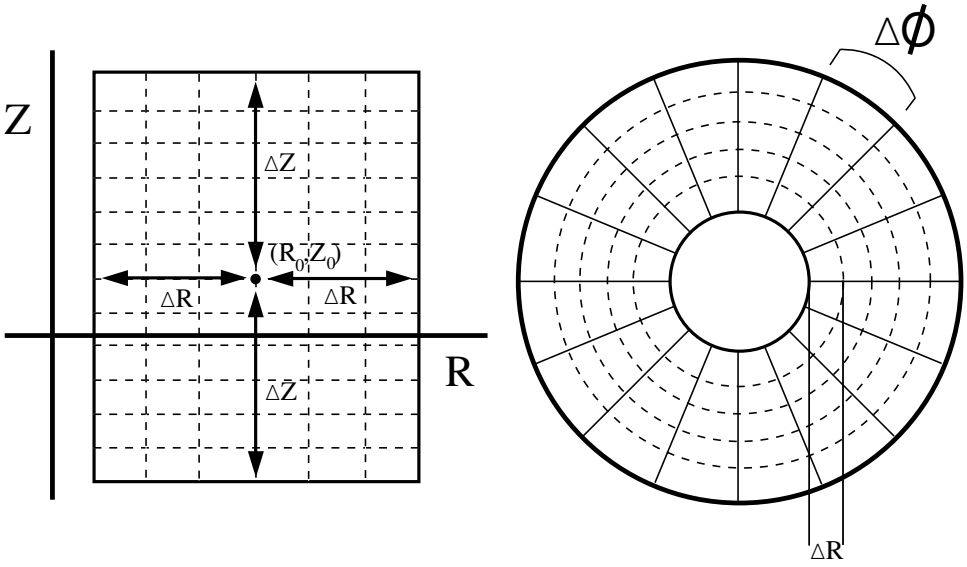

Figure 2: On the left is a side view of the computational box showing the center coordinates (R_0, Z_0) and width and height (ΔR and ΔZ). On the right is a top view, showing ΔR and $\Delta\phi$. The definition of $\Delta\phi$, which is measured in degrees, is simply $360/N_\phi$ for a single period device.

or lie close to particular surfaces such as the FW, LCMS or the divertor region surfaces. For points inside regions this task is accomplished through the numerical evaluation of the complex contour integral

$$\int_\Omega \frac{1}{z - z_0} dz \tag{1}$$

around the boundary Ω of the region *for each cross sectional step* in the ϕ direction. In this manner tags are given on the basis of the cross sectional geometry for each cross section. But before the integrations can be carried out for each region the boundary line of the region on which to integrate must be defined.

It is important to note that although the tags are given to grid points these grid points represent the cells created by the grid structure. A given grid cell is represented in a toroidal plane by the lower, left grid point which makes up the grid cell. The tag of the grid cell is given by this lower, left grid point. See figure 3 for an illustration of this point.

The strategy in defining the line of integration is to parameterize the surface line into a certain number (given as input to GEOM) of (R, Z) points, which will be called N_θ here for convenience. Two one-dimensional arrays, "rrc" and "zzc", which have size N_θ, store the R and Z coordinate data at each consecutive index for each parameterized point of the line. It is important to note that these parameterizations and arrays are independent of the grid points; the defining points of a line need not have the same coordinates as a

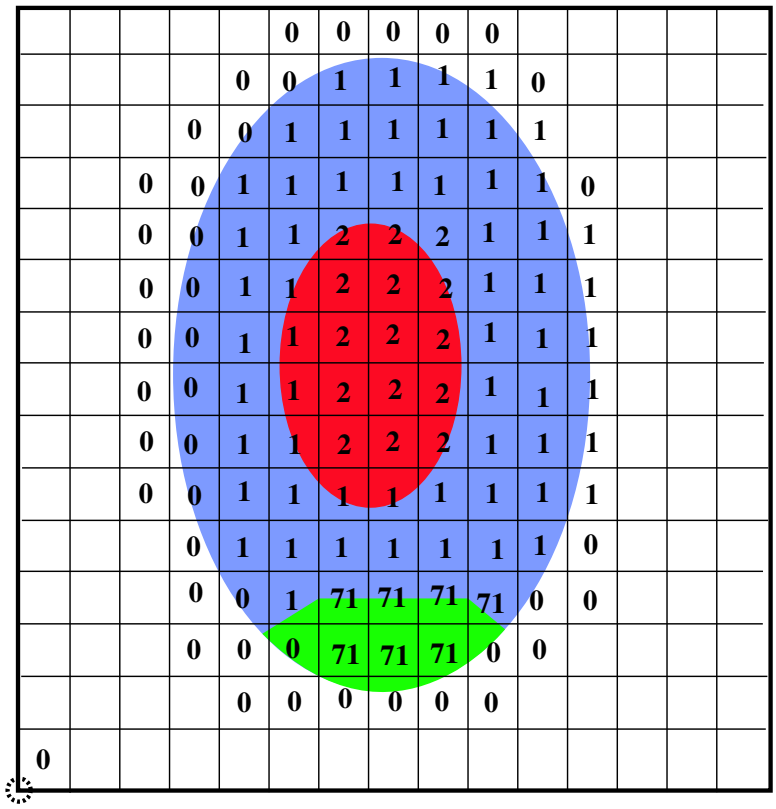

Figure 3: A toroidal cross section of a device imposed on a grid with the red, light blue and green regions being the area inside the LCMS, between the FW and LCMS, and the divertor region, respectively. The numerical tags (*without considering tags for surfaces*) for each cell are given by the tag given to the lower, left grid point which makes up the cell. All cells without explicit tags have tag 0. If surface tags were included in this illustration cells which are close to the boundaries of the three regions would have surface tags, such as 3 and 4 for the FW and LCMS surfaces, instead of the region tags.

grid point, in general they will not.

First the LCMS and FW are parameterized. The Fourier representations of the LCMS and FW are inputs to GEOM. From these the arrays rrc and zzc can be given values for each of their elements (named R_i and Z_i respectively). For example, if the LCMS is parameterized in the j^{th} toroidal cross section by N_θ points in a stellarator symmetric device, the i^{th} element of each array is calculated as

$$R_i = \sum_{m=0,n=0}^{mss,nss} \hat{r}^c_{m,n} \cos(2\pi[\frac{m(i-1)}{N_\theta} + \frac{n(j-1)}{N_\phi}]) + \hat{r}^c_{m,-n} \cos(2\pi[\frac{m(i-1)}{N_\theta} - \frac{n(j-1)}{N_\phi}]) \quad (2)$$

$$Z_i = \sum_{m=0,n=0}^{mss,nss} \hat{z}^s_{m,n} \sin(2\pi[\frac{m(i-1)}{N_\theta} + \frac{n(j-1)}{N_\phi}]) + \hat{z}^s_{m,-n} \sin(2\pi[\frac{m(i-1)}{N_\theta} - \frac{n(j-1)}{N_\phi}]) \quad (3)$$

where $\hat{r}^c_{m,n}$ and $\hat{z}^s_{m,n}$ are the mode m and mode n Fourier coefficients for the radial and vertical coordinate respectively, mss is the maximum m number and nss is the maximal n number.[2] Once this is carried out the surface has been parameterized and integration can be carried out to see if a particular cell is inside or outside of the surface.

The parameterization of regions not defined through Fourier representation (the divertor regions) is similar. Plate geometry is input in cylindrical coordinates, and at each cross section there are only a few points which define the plate. Therefore to parameterize the plate many points inbetween these defining points are linearly interpolated so that the integration steps are fine enough to ensure the accuracy of the integral. Also, a divertor region is made up of several different plates, and they must be connected in order to create a smooth parameterization of the region. This plate parameterization must in turn be connected to a parameterization of the section of the FW that the divertor surface region includes. This effectively means that the connecting plates must have the same coordinate for their adjoining end point. Once both paths, around the plates and on the FW area that encloses the divertor region, have been parameterized they are combined so as to form a closed loop around the divertor region. Integration can then be carried out to see if a particular grid cell lies inside or outside of the divertor region. See figure 4 for an illustration of parameterization of various geometric sections in a particular cross section.

2.2.2 Complex Integration and Tag assignment

Once the surfaces of a region have been parameterized it is possible to tell which grid cells lie inside and outside that region. To determine whether a particular cell lies in a

[2]The Fourier coefficients are functions of the curvilinear coordinate s and the quantities $\frac{(i-1)}{N_\theta}$ and $\frac{(i-1)}{N_\phi}$ equal the curvilinear coordinates u and v respectively.

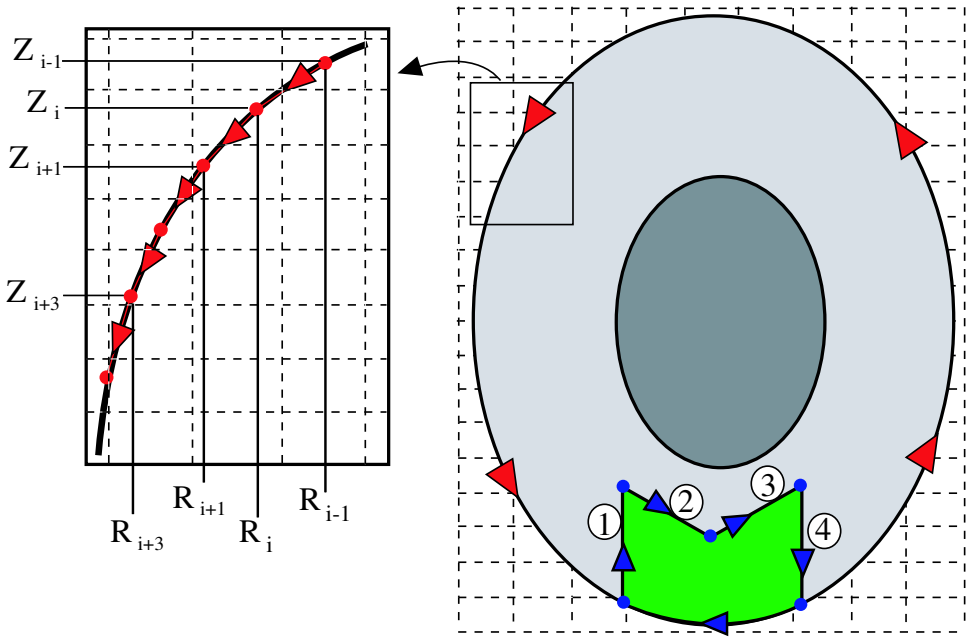

Figure 4: On the right, a cross section of a simple geometry, with the region inside the LCMS colored dark grey, the region between the LCMS and FW in light grey and the divertor region in green. The red arrows on the FW surface indicate the direction of parameterization and hence integration. Similarly the blue arrows indicate parameterization and integration direction of the divertor region. The numbers near the divertor region relate the plate numbers which make up the region, and the blue points are the defining point of the divertor plates. The grid is put in the background for clarity. On the left is a blow up of one section of the FW surface, where individual line integration segments are shown.

particular region with boundary parameterization Ω the complex contour integral

$$\int_{\Omega} \frac{1}{z - z_0} dz \tag{4}$$

with $z = R + iZ$ is carried out, where z_0 is the grid point representing the cell. If z_0 is inside the integral will yield $2\pi i$, if z_0 is outside the integral will yield 0. Therefore for any grid cell the imaginary part of this integral can be evaluated to see it the cell lies inside a given boundary. The discrete version, used for the actual numerical calculation of the imaginary part, is given by an absolute value of the sum

$$Im[\int_{\Omega}] \approx abs[\sum_{i=1}^{N_\theta - 1} \left(\frac{rr_i}{rr_i^2 + zz_i^2} + \frac{rr_{i+1}}{rr_{i+1}^2 + zz_{i+1}^2} \right) dz_i - \left(\frac{zz_i}{rr_i^2 + zz_i^2} + \frac{zz_{i+1}}{rr_{i+1}^2 + zz_{i+1}^2} \right) dr_i] \tag{5}$$

where rr_i is the radial distance of z_0 from R_i, zz_i is the vertical distance of z_0 from Z_i, dr_i is $R_i - R_{i+1}$ and dz_i is $Z_i - Z_{i+1}$.[3]

The tag assignment procedure works as such. First, all grid cells are assigned the tag 0, which is the tag for cells outside the FW. Then for each cell in the grid complex integration is carried out with the boundary being the FW, and cells inside the FW are given the tag 1. Then for all cells the integration is carried out again, this time the boundary being the LCMS, and the cells inside are given the tag 2. After this the integration is carried out for all divertor regions, which are given tags of 70 or higher.

The cells lying close to any surface, such as the FW, LCMS or a divertor region, are given a tag differing from that of the region they are close to, but which also identifies the region they are close to. These cells are considered to be on the "surface" of that region. Each point of the parameterized surface array (the i^{th} point being R_i, Z_i) lies in a grid cell, and the tag of that grid cell is given the corresponding surface tag, which is how the surfaces of regions receive distinct tags. Usually there are several parameterized points which are lie in the same cell. That grid cell will be assigned the same tag for each of these points. Having multiple parameterized points per grid cell ensures there will be no "hole" in any surface. The value of the tags for the FW and LCMS are 3 and 4 respectively while surface tags for the divertor regions have values 11 to 70.

Every grid cell has a numerical tag, which is stored in the array "index". There is another array, called "aindex" which stores a different type of data for every grid cell, although this data is only truly meaningful for cells identified as surface cells of the FW or LCMS. For these surface grid cells "aindex" stores the curvilinear flux coordinate u, which is given by

[3]Currently the code is written such that the discrete integral isn't evaluated from one partition point to the next partition point, but rather from one partition point to 16 points away. This is the meaning of the variable "icompd", which is set to 16. This is done to save computational time. If the integral does not yield a satisfactory answer then the integration is done again skipping every eight points. The multiple of points skipped is decreased by a factor of two until the integration yields satisfactory results.

$$u_i = \frac{(i-1)}{N_\theta} \tag{6}$$

where i runs from 1 to N_θ (the u coordinate relates to the cylindrical coordinates through equations (2) and (3)). This data will allow GOURDON to calculate the position of the field line more easily. All cells which do not have surface tags are given a u coordinate of 0.

After the entire process of assigning tags and u coordinates is finished the two arrays "index" and "aindex" are written in binary to file, which by default is called "fort.10".

2.2.3 GEOM/GOURDON Interface

The GOURDON code uses the output from GEOM in order to see if the current traced step is "close" to the surface of a region. Every time GOURDON moves through an integration step it determines the indices of the corresponding GEOM cell and queries its tag. GOURDON also keeps in memory the coordinate and tag data from the previous step. From this information GOURDON can tell if, in one integration step, a field line crossed, or might have crossed, a boundary.

If the tag of the current integration step is greater than 10, meaning the boundary of a divertor region may have been crossed, the subroutine "REGION" is called, which determines if the boundary was crossed, and if so, where it crossed.

Subroutine REGION works in the following way. Each of the non-planar elements forming divertor and baffle plates is approximated by two triangles (see figure 5), with two defining points having the same toroidal coordinate, phi, and one having a smaller or larger phi-coordinate. Since the GOURDON code knows in which GEOM cells the current and former field line coordinates are lying, GOURDON has a good idea if the field line is close to a plate. If it is close then the subroutine REGION searches for the intersection point of the field line with the plate element using the relations summed up in Appendix 1. For these calculations it is assumed that the field line goes straight from the former to the current coordinate. If the field line does not intersect a triangle then the subroutine is exited to continue the tracing of the field line. Otherwise, a parameter "ksx" is set equal to 1, which, once control is given back to the routine which call REGION, tells GOURDON that this field line tracing should be stopped.

After the optional call to REGION, the subroutine "ORT" is called, which determines if the field line has crossed a surface defined through Fourier representation (such as the FW or LCMS). For each tracing step this subroutine is called but is exited immediately if it can be determined just from the current and former field line position tags that a surface has not been crossed, which saves computational time.

The methods which REGION and ORT use to determine crossings differ. The ORT subroutine takes advantage of the fact that surfaces defined through Fourier representation have a constant s coordinate value in the left-handed curvilinear flux coordinates given by (s, u, v). In this use of the flux coordinates the LCMS has $s = 0$, while the FW has $s = 1$. Points inbetween have s values from 0 to 1 while points outside the FW have s values greater than 1. All ORT has to do to see if a field line crossed a surface is calculate the

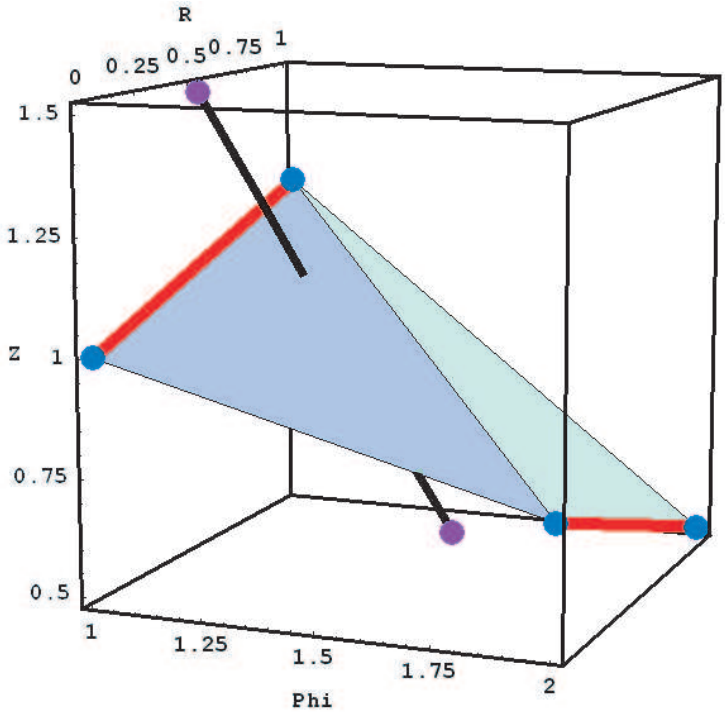

Figure 5: A case in which the field line crosses into a divertor region. The blue dots represent points where the plates are defined, and the red lines are connecting the plate points which are defined in the same cross section/value of ϕ. The purple dots and black line represent the last and current position of the field line and the line segment that connects them. As is shown, two triangles are made from the four plate points, and the field line segment intersects the triangle in the foreground.

s coordinate of the current and former field line coordinate: if the s value passes through 0 or 1 when transitioning from the former s value to the current s value then it is clear that a, and which, surface has been crossed.

The (s, u, v) values corresponding to a given (R, ϕ, Z) point is found in the following manner. Fourier coefficients for a surface with s value inbetween 0 and 1 are obtained by linear interpolation of the Fourier coefficients between two neighboring surfaces themselves defined by Fourier coefficients. For a surface with $s = 0$ (the LCMS) and for another with $s = 1$ (FW) the radial Fourier coefficients of a surface with intermediate s value are

$$\hat{rz}^{c,s}(s) \quad = \quad \hat{rz}_{m,n}(s=0)[1-s] + \hat{rz}_{m,n}(s=1)[s] \tag{7}$$

where $\hat{rz}^{c,s}$ stands for either the radial or vertical Fourier coefficient. The general relation between a cylindrical coordinate (R, ϕ, Z) and a flux coordinate (s, u, v) for a stellarator device is given by the relation

$$R - \sum_{m=0,n=-nss}^{mss,nss} \hat{r}^c_{m,n}(s)\cos(2\pi[mu+nv]) = 0 \tag{8}$$

$$Z - \sum_{m=0,n=-nss}^{mss,nss} \hat{z}^s_{m,n}(s)\sin(2\pi[mu+nv]) = 0 \tag{9}$$

$$\phi = v\frac{2\pi}{N_p} \tag{10}$$

where N_p is the number of periods of the device.[4] The Fourier coefficients $\hat{r}^c_{mn}(s)$ and $\hat{z}^s_{m,n}(s)$ are defined by equation (7). While v is simply given by equation (10) the s and u coordinates have to be computed by searching a zero of the two nonlinear functions of equations (8) and (9). This is done with the C05PBF routine of the NAG library.

If a surface has not been crossed then ORT returns control to the routine which called it. If a surface has been crossed then it calculates the three dimensional coordinates of the intersection of the field line and surface (see Appendix 1 for details). If the FW surface has been crossed the parameter "ksx" is set to 1, which signals to GOURDON that this field line should no longer be traced, and the relevant data is written to arrays. After all field lines have been traced, these arrays, depending on factors such as if field lines or guiding centers were being traced, are written to different output files.

[4] In the code the cosine and sine terms are calculated by evaluating the quantities $\cos(2\pi mu)\cos(2\pi nv) - \sin(2\pi mu)\sin(2\pi nv)$ and $\sin(2\pi mu)\cos(2\pi nv) + \cos(2\pi mu)\sin(2\pi nv)$, respectively.

3 Use of GEOM

GEOM is written in FORTRAN 90 using free allocation of the field dimensions and free format. The code uses the MPI communication library and runs on an IBM Regatta supercomputer.

The directory of GEOM has the following structure. The main directory is called **geom**, the source code is in subdirectory **src**, and the makefile and the object code are in subdirectory **obj**. Subdirectory **exe** contains the executables and the input file.

3.1 Input

The input is a user created file which contains all the relevant structural geometric data (the positions of the FW, divertor and baffle plates) and plasma boundary geometric data (the LCMS) of the device. Also included are the parameters for the grid and the desired number of partitions for the parameterization of the surfaces. Appendix 2 is a sample input file for reference, and figure 6 is a graphic of one of the cross section outputs of this input.

Only devices in which the FW and LCMS are stellarator symmetric or axisymmetric with up-down symmetry can be processed by GEOM, although it would be straight forward to modify the code so that it could run cases without symmetry. The plates can have any symmetry, or none at all.

The first line of the input file, which starts with "stop", is the kind of run it is desired GEOM process. The options for this are "insur", which simply reads in the Fourier coefficients of the surfaces, "beleg", which carries out the tagging process for the FW and LCMS, not regarding if there are divertor regions present or not, and"geom", which characterizes the entire geometry of surfaces, divertor and baffle plates.

The next two lines are for characterizing the grid. The parameter "np" is the number of field periods of the device. If np is greater than 1 then GEOM calculates the tags for the points of one field period and writes the same tags for that period onto the other periods. The parameter "nf" is the number of divisions of the grid desired in the toroidal direction (the number of cross sections that will be calculated), "nr" is the number of divisions in the radial direction and "nz" is the number of divisions in the vertical direction, see figure 2. In the next line, "rnull" is the radial center of the grid, "znull" the vertical, and "ronull" and "zonull" are the radial and vertical half-widths, respectively, meaning the grid starts radially at rnull-ronull and goes to rnull+ronull, while points are defined for every ronull/nr unit of distance. By convention these distances are in meters.

The next two lines relate to the Fourier defined surfaces. The variable "jub" is the number of points that the LCMS will be divided into when the complex contour integral moves along the LCMS boundary. Likewise, "jub1" is the number of points the FW will be divided into. Presently both jub and jub1 must be a multiple of the variable "icompd", which is set to 16 (See the description of this variable for an explanation). The parameter "mss" is the maximal m mode number in the Fourier representation for all of the surfaces, and "nss" is the maximal n node number for all the surfaces.

The following block of input relates to the LCMS. The variable "nlcms" is the number of Fourier coefficients used to represent the LCMS. The Fourier coefficients of the LCMS $\hat{r}^c$ and $\hat{z}^s$ are named as "rmnc" and "zmns" respectively. Note that the "rmns" and

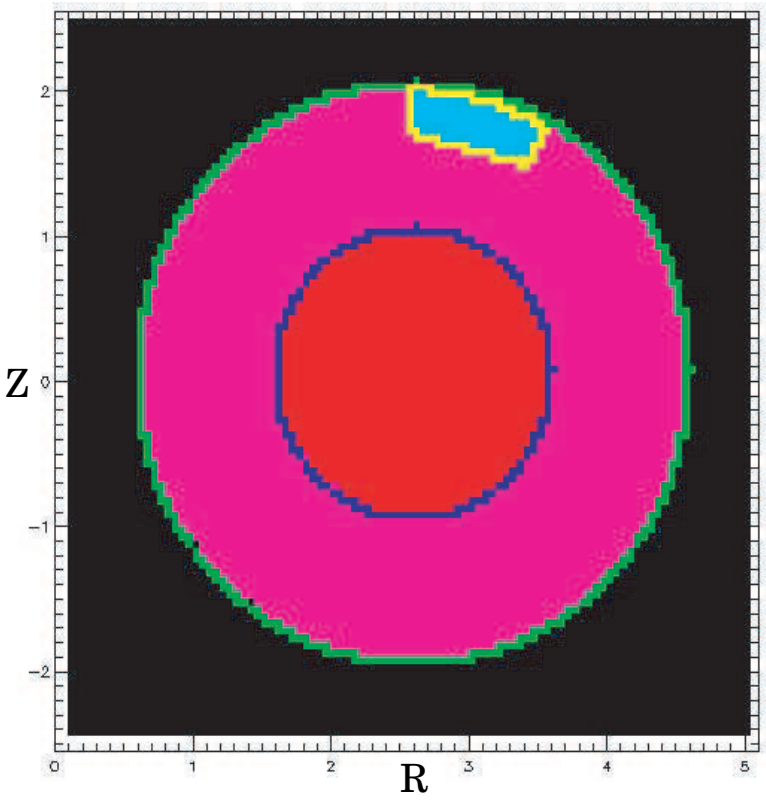

Figure 6: A visual representation of the sample data at the $\phi = 8$ cross section. Each tag is given a different color. The region outside the FW is black, the region between the FW and LCMS is lite red, the region inside the LCMS is red, the divertor region is lite blue, the FW is green, the LCMS is blue and the divertor region surface is yellow.

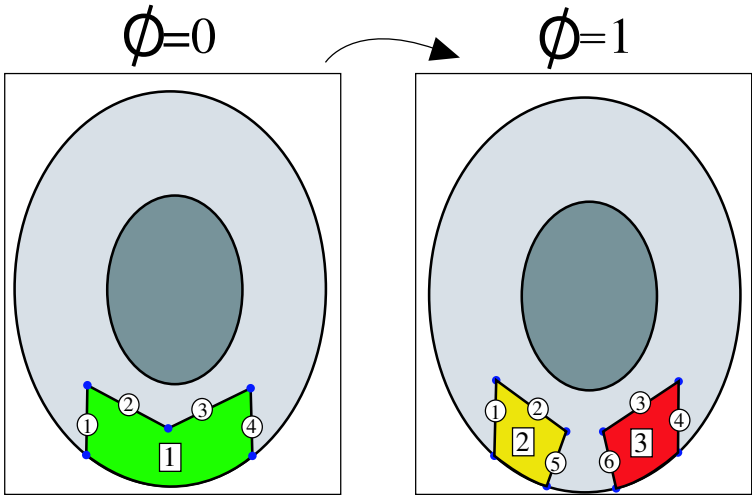

Figure 7: The geometry for two cross sections, one at $\phi = 0$ and the other at $\phi = 1$. For the $\phi = 0$ cross section there is one divertor region named "1", which is made up of plates 1, 2, 3 and 4. In the next cross section some of the defining points for plate 1-4 have changed, and it was decided by the user to make these two separate divertor regions, "2" and "3". To do this connector plates 5 and 6 had to be created to connect plates 2 and 3 to the FW.

"zmnc", which represent $\hat{r}^s$ and $\hat{z}^c$ are required but set to zero since stellarator symmetry or axisymmetry with up-down symmetry is assumed, see Appendix 2 for an example. The coefficients must be put in increasing m value, while put in decreasing n value for any given m value (see Appendix 2). The next block of input relates to the FW. The variable "nwand" is the number of Fourier coefficients used to represent the FW. Again "rmnc" and "zmns" are the nonzero Fourier coefficients.

After the surfaces, the plates are specified. Before any particular plate geometry is detailed, numbers relating to the whole set of plates, which make up the divertor regions, are specified. A divertor region is a closed region defined by divertor, baffle and connector plates, and a section of the first wall. Connector plates are plates which serve to connect the divertor or baffle plates to the FW. Connector plates may or may not actually exist in the device but they are necessary constructs for GEOM to be able to carry out the complex boundary integral for the divertor regions. In turn, divertor regions can split apart, if it is deemed necessary, from cross section to cross section and become unique regions. Also, one plate can make up more than one region. See figure 7.

The parameter "ndd" is the total the number of plates to be read in. The parameter "kto" is the maximum number of toroidal coordinate points a plate may have. This number can be larger than the maximum number of ϕ values given in the input file for a single plate because additional points can be calculated by interpolation. If data is only specified for whole degrees then kto should be at least double the number of points specified for the plate with the highest number of toroidal data points. Similarly, the

parameter "kpo" is the maximal number of (R, Z) points which define any plate. The parameter "npper" is the number of degrees in one period. The parameter "igi" is the number of divertor regions in the entire geometry.

After these parameters, the data relating to each plate are specified. First comes the numerical name of the plate followed by four numbers on the next line. The first of these numbers, called "ndi" in the source code, specifies the number of cross sections at which the plate is defined, while the next parameter, called "ndii" in the source code, is the number of (R, Z) points which define the plate in each cross section. The next number, called "iwid" in the source code, takes on either the value 1 or 2. When this number is 2 this means data is specified every one-half degree. When it is 1, data is only specified for integer degrees, and GEOM linearly interpolates half degree data for that plate. This is to ensure that all plates are defined for every cross section GEOM sets tags for, whether the cross sections advance in the toroidal direction by a whole or half integer degree step. The fourth number, called "ianf" in the source code, can take the values 0, 1 or 2. It is possible for a plate which is generally only defined for whole integer degree values of ϕ to begin (or end) on a half integer value of ϕ, where only the first (or last) step in ϕ is 0.5 degrees. If the fourth number is 2, then the plate ends with a half-integer value of ϕ; if 1, then it begins with a half-integer value of ϕ; and if it is 0, the plate both begins and ends with integer values, irrespective if the ϕ step is one or one-half degree.

The following data are the coordinate data for the plate. There are "ndi" chunks of data, which specify the data for one ϕ value of the cross section, with each being "ndii+1" lines long. The first lone number is the ϕ value in degrees for that cross section. The following numbers are the radial, vertical and curvilinear u coordinates of the point. The u coordinate is only meaningful when the point lies on the FW, in which case this plate is connected to the FW and this number takes the u value of the wall at that point. Otherwise this number is set to -100.

After all of the data for the plates is entered, two arrays, which relate the divertor region name to cross sections and plate names, must be specified. The first, named "ireg1" in the source code, has eight columns. The first is the number of the cross sections, of which there are "nf" of, which starts at 0 and runs to nf-1. It is important to note that this is not necessarily the ϕ values for the cross section. The second column indicates how many regions are in any particular cross section. Columns three through eight specify the numerical names of the regions in that cross section. As the code is written there are up to six regions in a cross section. This parameter can be changed by tweaking the source code, if necessary. If there are less than six regions in a particular cross section then the superfluous elements in the row are set to 0.

Then next array, called "ireg2" in the source code, relates the region name to the plates which make it up. The first column is the region name and the second is the number of plates which make up that region. Columns three through six are the numerical plate names of the plates that make up the region. As written, up to four plates can make up a region, which is relatively straight forward to change if necessary. If the region is made up of less than four plates the superfluous elements in the row are set to 0.

For quick reference a list and description of the input variables ordered by their role is given here. The input quantities are read in via the subroutines geom.f90, insur.f90, iregin.f90 and pquerd.f90.

Device and Mode

- **stop**: A character which describes the type of job it is desired GEOM run. If "insur" is specified then GEOM only reads in the Fourier coefficients for the FW and LCMS. If "beleg" is specified GEOM creates a tagged map of the grid by ignoring the plate data (i.e. all tags will be from 0 to 4). If "geom" is specified GEOM creates a tagged map considering the entire geometry.
- **np**: An integer which specifies the number of field periods of the device.

Grid Parameters

- **rnull**: A real number which specifies the radial center of the grid.
- **ronull**: A real number which specifies the half-width of the computational box.
- **znull**: A real number which specifies the vertical center of the grid.
- **zonull**: A real number which specifies the half-height of the computational box.
- **nf**: An integer which specifies the number of toroidal partitions of the computational grid.
- **nr**: An integer which specifies the number of radial partitions of the computational grid.
- **nz**: An integer which specifies the number of vertical partitions of the computational grid.

Parameterization

- **jub**: An integer which specifies the number of points that the LCMS will be dividing into when the complex contour integral moves along the LCMS boundary. Should be an integer multiple of the variable "icompd", see section 2.2.
- **jub1**: An integer which specifies the number of points that the FW will be dividing into when the complex contour integral moves along the FW boundary. Should be an integer multiple of the variable "icompd", see section 2.2.

Fourier Coefficients

- **m**: An integer which is the poloidal mode number.
- **n**: An integer which is the toroidal mode number.
- **nlcms**: An integer which specifies the number of Fourier coefficients which define the LCMS.

- **nwand**: An integer which specifies the number of Fourier coefficients which define the FW.
- **mss**: An integer which specifies the maximal m-mode number in the Fourier representation for all of the surfaces.
- **nss**: An integer which specifies the maximal n-mode number in the Fourier representation for all of the surfaces.
- **rmnc**: A real number which is the radial symmetric Fourier coefficient for a given surface and (m,n) mode.
- **rmns**: A real number which is the radial asymmetric Fourier coefficient for a given surface and (m,n) mode. This term should be set to zero in the input file since GEOM can only process LCMS and FW geometry which is stellarator symmetric or axisymmetric with up-down symmetry.
- **zmnc**: A real number which is the vertical asymmetric Fourier coefficient for a given surface and (m,n) mode. This term should be set to zero in the input file since GEOM can only process LCMS and FW geometry which is stellarator symmetric or axisymmetric with up-down symmetry.
- **zmns**: A real number which is the vertical symmetric Fourier coefficient for a given surface and (m,n) mode.

Plates

- **ndd**: An integer which specifies the number of plates in the geometry.
- **kpo**: An integer which specifies the maximal number of (R, Z) coordinates which make up a point.
- **kto**: An integer which specifies the maximal number of cross sections which could be processed.
- **npper**: An integer which specifies the number of degrees in one period.
- **igi**: An integer which specifies the number of divertor regions present in the geometry.
- **ndi(ndd)**: An array which holds the number of defining toroidal points for every plate as an indexed function of plate name. Specified before the particular plate data in the input file.
- **ndii(ndd)**: An array which holds the number of defining coordinate points for every plate as an indexed function of plate name. Specified before the particular plate data in the input file.
- **iwid**: An integer which specifies whether half-degree data should be interpolated for a particular plate. Specified before the particular plate data in the input file.

- **ianf**: An integer which specifies whether toroidal plate data begins, ends or neither begins nor ends with half-degree data. Specified before the particular plate data in the input file.
- **ireg1(8,npper-1)**: The first large input array, specified in the input file after all plate data has been entered. Relates the cross section number to the divertor regions in the cross section.
- **ireg2(6,igi)**: The second large input array, specified in the input file after all plate data has been entered. Relates the divertor region numbers to their constituting plates.

3.2 GEOM Output

As written there are four main output files called "fort.10", "fort.17", "fort.26" and "output". The fort.10 file are the index and aindex arrays written in binary, the fort.17 file are the index and aindex arrays *without considering the plates* written in binary and the fort.26 file is the index array written in ASCII. The output file is essentially a log recording if certain subroutines were called. A full run of GEOM (the variable "stop" in the input file the value "geom") gives an output file which only has the date and time at which the job ran, whereas a run which did not consider the plate geometry ("stop"="beleg") would have the text "'STOP: beleg'" before the date and time.

3.3 Compiling and Running GEOM

In the subdirectory **obj_ibm** is the make file which creates the executable GEOM in the subdirectory **exe**.

The job is submitted by executing the commands text file, which for the example input file named "my_in" of Appendix 2, is named "geom.e". In the first line of this file the text string following "poe" is what the GEOM executable is to be called, and "<my_in>>" specifies the path name of the input file (beginning in the subdirectory **exe**). The output file name is given by the next string, here "output". The number of processors used to run the job is given after the text "-procs".

Here is a simple example of a GEOM command script

```
poe GEOM <my_in >>output -procs 1
date >> output
```

4 Source Code

In this section the subroutines and modules used in both GEOM and the GEOM-related code in GOURDON will be briefly described.

4.1 GEOM Subroutines and Modules

4.1.1 Subroutines

- **beleg.f90**: The purpose of this subroutine is to assign tag values to all grid cells on the basis of whether or not the points are inside, outside or "close" (as explained in section 2.2) to the FW or LCMS, and write the data to various output files. It does not take into account whether or not divertor regions are present, and therefore only assigns values 0, 1, 2, 3 or 4 to grid points. This subroutine contains code or other subroutines which parameterize the surfaces, carries out complex integration on the surfaces and writes data to the output file.

- **belegp.f90**: The purpose of this subroutine is to assign tag values to grid cells on the basis of whether or not the points are inside or on the surface of (as explained in section 2.2) a divertor region, and write the data to various output files. If the grid cell in not close to a divertor region then the tag is not changed. Also, this subroutine contains code or other subroutines which parameterize the surfaces and boundaries of the divertor regions, carries out complex integration on these boundaries and writes data to the output file. This subroutine is called only when the text "geom" is inserted on the second line of the input file.

- **compdiv.f90**: This subroutine carries out the complex integration over either the FW or LCMS. It outputs the value of the integration and requires the coordinate of the grid point for which the integral will be carried out (z_0 in equation (4)).

- **compdivg.f90**: This subroutine is similar to "comdivg.f90"; it calculates the contour integral for a certain grid point z_0. The difference is that "compdivg.f90" performs the integration for a divertor region, not the FW or LCMS. It requires the parameterization of the plates which make up the divertor region and the parameterization of the part of the FW which completes the boundary of the divertor region.

- **geom.f90**: This is the main program of GEOM. The first block of the input file, up to the specifications of the LCMS, are read in here.

- **insur.f90**: This subroutine reads in the surfaces defined by Fourier coefficients, such as the FW and LCMS, from the input file. It also divides the coefficients in which $n = 0$ by two. This is due to how the coefficients are used in the code. The $n = 0$ coefficients are counted twice in a sum, and to counter this these coefficients are divided by two.

- **iregin.f90**: This subroutine reads in the two large arrays at the end of the input file (see section 3.1). The first array relates the cross section number to the divertor

regions it contains. The second relates the divertor region number to the plates which make up that divertor region.

- **pquerd.f90**: This subroutine reads in data of the divertor geometry and interpolates half-degree coordinate data for a given plate if no such data exists. This interpolation is linear, depending on the plate coordinates from the two closest integer degree data. It also begins to parameterize the path for the integration on the plate. The parameterization is finished, by making much finer partitions, in the subroutine "belegp".

- **surfb.f90**: This subroutine parameterizes either the FW or LCMS. It produces two arrays, "rrc" and "zzr", which are the radial and vertical coordinates of the parameterization, respectively. Stellarator symmetry or axisymmetry with up-down symmetry of the FW and LCMS is assumed.

- **surfwl.f90**: Similar to subroutine "surfb.f90", this subroutine parameterizes the part of the FW which makes up part of the boundary of a given divertor region. It produces arrays "rrw" and "zzw", which are the radial and vertical coordinates of the parameterization, respectively. Stellarator symmetry or axisymmetry with up-down symmetry of the FW is assumed.

- **winkel.f90**: This subroutine calculates part of the expression necessary to transform the Fourier representation of the FW and LCMS into cylindrical coordinates so that these surfaces can be parameterized and grid points nearby can be assigned the tags of these surfaces. It is in this part of the expression where the indexes of the parameterized lines serve as input for the transformation. This subroutine also allocates many of the arrays which are used for parameterization, complex integration, transformation from Fourier representation to cylindrical coordinates and also for storing of the tag and u coordinate data.

4.1.2 Modules

- **mod_compl.f90**: Contains various arrays relating to complex integration, all of which are allocated in subroutine "winkel.f90".

- **mod_cosind.f90**: Contains various arrays relating to the transformation from Fourier representation to cylindrical coordinates, all of which are allocated in subroutine "winkel.f90".

- **mod_dim.f90**: Creates constants which are read in from the input file.

- **mod_fourier.f90**: Contains various arrays which hold the Fourier representation of the FW and LCMS, all of which are allocated in subroutine "insur.f90".

- **mod_index1.f90**: Contains various arrays and constants which relate to the assigning of processors to cross sections and to the tags and u coordinate data of the grid points, all of which are allocated in subroutine "winkel.f90".

- **mod_plate.f90**: Contains various arrays relating to plate specifications and parameterization, all of which are allocated in subroutine "iregin.f90".

- **mpp_functions.f90**: This module establishes a MPP job environment to execute the parallelized GEOM code. This relates to all of the "call barrier, call broadcast" code blocks used to transmit data that processor 0 has calculated to all other processors. The "call broadcast(var)" command sends the variable "var" from the current processor to all other processors. The "call barrier()" command does not allow any processors to take on another task until the current processor is finished with its task. The "call MPI_Recv(a,b, MPI_DOUBLE_PRECISION, k..." and "call MPI_Send(a,b, MPI_DOUBLE_PRECISION, 0..." commands are also part of the parallelized structure, of which the first command tell the current processor to receive variables "a" and "b" from processor "k", while the second commands tells the all other processors to send variables "a" and "b" to processor 0.

  This parallelized environment is initialized with the "call init_MPP_environment()" command near the beginning of the "geom.f90" source code, and ended with the "call mpi_finalize(errorcode)" command at the end of the "geom.f90" source code.

4.2 Parallelization in GEOM

GEOM is a parallelized code which is parallelized in the toroidal direction. When a job is run with more than one processor each processor computes and assigns the tag for the grid cells in a number of different cross sections. For example, if there are 20 toroidal cross sections of a device, and four processors are used, each processor will compute the tag data for five cross sections. Below is a list and short description of the most important variables related to parallelization which appear in the source code.

Variables of Parallelization

- **npes**: An integer set by the user in the commands text "geom.e" which is the number of processors being used to execute the job. Appears in subroutines beleg, belegp and mpp_functions.
- **pid**: A processor dependent integer which is the identification number of the current processor. This number takes values from 0 to npes-1. Appears in subroutines beleg, belegp, geom, iregin, mpp_functions and pquerd.
- **imax**: An integer equal to MAX[(nf+npes-1)/npes, 1]. It is, essentially, the number of cross sections that each processor will need to compute. This is defined in this non intuitive manner since it is possible that the number of processors does not divide evenly into the number of cross sections to compute. See discussion of variable ipek for more information. Appears in subroutines/modules beleg, belegp and mod_index1
- **ipek(imax+1)**: A processor dependent array which holds the cross section numbers for a given processor to compute. With a size "imax+1" it is ensured that all cross sections are processed. If the number of cross sections to be computed does not divide evenly into the number of processors being used, npes, some of the last elements of the array for a few of the processors will not be accessed. The elements of this array vary from processor to processor because of the the inclusion of the processor id

number, pid, in the assigning of the array in subroutine beleg. For example, the array could have elements (0,3,7,11) for one processor, meaning this processor will compute the tagged map for cross sections 0, 3, 7 and 11, while having elements (2,5,9,13) for another processor. Appears in subroutines beleg and belegp.

4.3 GEOM related Modules, Subroutines and Variables in GOURDON

The subroutines "ort.f90" and "region.f90" form the heart of the GEOM-related code in GOURDON. In these it is calculated if the field line crosses a surface or divertor region, respectively, and if so, what are the coordinates of intersection. These subroutines are called one after the other in the large GOURDON subroutines called "fieldline.f90" and "guiding.f90", which trace field lines or guiding centers.

In the block of code before ORT and REGION are called the location of the grid cell which the field line is current in are calculated and the tag of that grid cell is queried, which if it has the appropriate value, begins the detailed investigation if and where the field line crossed the Fourier representation defined surface or the divertor region surface.

Many of the GEOM related subroutines, modules and arrays are described below.

4.3.1 Subroutines

- **insur.f90**: Same function as in GEOM.
- **iregin.f90**: Same function as in GEOM.
- **lage.90**: This subroutine calculates the curvilinear s and u coordinates of the current position of the field line. See section 2.2.3, GEOM/GOURDON interface, and Appendix 1 for more information. Appears in subroutine ort.
- **ort.f90**: This subroutine (if the grid cell in which the field line is currently in has a surface tag, such as the tag from the FW or LCMS) calculates if the field line truly crossed the surface, and if it did, where it crossed. See section 2.2.3, GEOM/GOURDON Interface, for details.
- **position.f90**: This subroutine is called when it has been determined by subroutine ort that a field line crosses the FW or LCMS boundary. It calculates the coordinates of intersection. See section 2.2.3, GEOM/GOURDON interface, and Appendix 1 for more information. Appears in subroutine ort.
- **pquerd.f90**: Same function as in GEOM.
- **readin.f90**: Reads in the first block of the GEOM input file.
- **region.f90**: This subroutine, if the grid cell in which the field line is currently in has a divertor region boundary tag, calculates if the field line truly crossed the divertor surface, and if it did, where it crossed. See section 2.2.3, GEOM/GOURDON Interface, for details.

4.3.2 Variables

Tracing "nparam" field lines, information about their possible intersection with LCMS, FW and divertor and baffle plates are stored in the following arrays:

GOURDON Arrays Concerning Field Line/Surface Intersection

- **dlfield(6, nparam)**: The first dimension of this array stores the coordinates (phi,R,Z) of the intersection point of a field line or guiding center with the LCMS, velocity of the guiding center at the intersection point, and length of the path and time the traced guiding center needed to reach the intersection point. Note, that velocity and time are only defined for the guiding center. The second dimension stores the field line number.

- **wfield(7, nparam)**:The first dimension of this array stores coordinates (phi,R,Z) of the intersection point of a field line or guiding center with the FW, velocity of the guiding center at the intersection point, and length of the path and time the traced guiding center needed to reach the intersection point. Note, that velocity and time are only defined for the guiding center. Further, the u-coordinate of the intersection point is stored as the last quantity. The second dimension stores the field line number.

- **aplat(3, nparam)**:The first dimension of this array stores coordinates (phi,R,Z) of the intersection point of a field line or guiding center with a divertor or baffle plate. The second dimension stores the field line number.

- **iplat(3, nparam)**:The first dimension of this array stores number of the divertor region, "ignum", plate number, "ipnum", and number of the triangle, "ize", of the intersection points for field lines that intersect divertor or baffle plates. The second dimension stores the field line number.

4.3.3 Modules

- **mod_dim.f90**: Same as in GEOM.

5 Appendix 1: Finding intersection points

5.1 Triangles

The problem is to find the criteria for the intersection of a given line segment with a given triangle, and if they do intersection, to find the coordinates of intersection, $\vec{I}$.

Let the general triangle, illustrated in figure 8, be composed of arbitrary points

$$\begin{array}{rcl} \vec{r1} & = & (r1_x, r1_y, r1_z) \\ \vec{r2} & = & (r2_x, r2_y, r2_z) \\ \vec{r3} & = & (r3_x, r3_y, r3_z) \end{array} \tag{11}$$

Define the vector $\vec{a}$ to be $\vec{a} = \vec{r1} - \vec{r2}$, and vector $\vec{b}$ to be $\vec{r2} - \vec{r3}$.

Given the points $\vec{f1}$ and $\vec{f2}$ which define the line segment define $\vec{f} = \vec{f2} - \vec{f1}$ and $\vec{r} = \vec{r1} - \vec{f1}$. With the definition of the following parameters

$$\begin{array}{rcl} \lambda & = & (r_x + \alpha a_x + \beta b_x)/(f_x) \\ \alpha & = & \frac{(r_y f_x - r_x f_y) + \beta(b_y - f_x - b_x f_y)}{a_x f_y - a_y f_x} \\ \beta & = & \frac{(r_y f_x - r_x f_y)(a_x f_z - a_z f_x) - (r_z f_x - r_x f_z)(a_x f_y - a_y f_x)}{(b_y f_x - b_x f_y)(a_x f_z - a_z f_x) - (b_z f_x - b_x f_z)(a_x f_y - a_y f_x)} \end{array} \tag{12}$$

it can be shown that the criteria for intersection are

$$0 \leq \alpha, \beta, \lambda \leq 1 \quad \text{and} \quad \alpha + \beta \leq 1 \tag{13}$$

and that the intersection point is given by

$$\vec{I} = \vec{f1} + \lambda(\vec{f2} - \vec{f1}) = \vec{r1} + (\alpha\vec{a}) + (\beta\vec{b}) \tag{14}$$

5.2 Surfaces

The problem is to, given two points $\vec{X1}$ and $\vec{X3}$ with coordinates (x_1, y_1, z_1) and (x_3, y_3, z_3) with a connecting segment that intersects a surface defined by Fourier coefficients with known s coordinate, find the intersection coordinates of the surface and line segment.

Since it is know that the intersection point lies on the line connecting $\vec{X1}$ and $\vec{X3}$ the vector, call $\vec{X2}$, from $X1$ to the intersection point can be written as

$$\vec{X2} = a(x_3 - x_1, y_3 - y_1, z_3 - z_1) \tag{15}$$

where a is a scalar, and the coordinates of the intersection are given by $\vec{X1} + \vec{X2}$. Since the s coordinate and the Fourier coefficients of the surface are known, the components of the vector $\vec{X2}$ can be represented in the following manner.

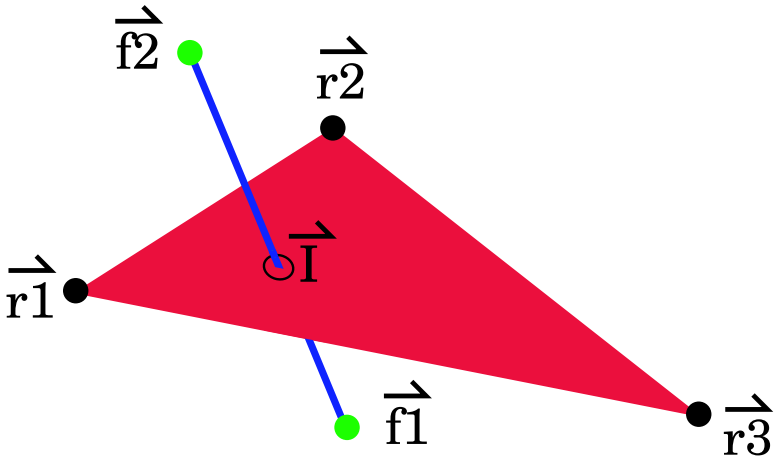

Figure 8: An illustration of a line segment intersecting a triangle.

$$\begin{array}{rcl} x_2 & = & \left(\sum_{m=0,n=0}^{mss,nss} \hat{r}^c_{m,n} \cos(2\pi[mu+nv])\right) \cos(v\frac{2\pi}{N_p}) \\ y_2 & = & \left(\sum_{m=0,n=0}^{mss,nss} \hat{r}^c_{m,n} \cos(2\pi[mu+nv])\right) \sin(v\frac{2\pi}{N_p}) \\ z_2 & = & \sum_{m=0,n=0}^{mss,nss} \hat{z}^s_{m,n} \sin(2\pi[mu+nv]) \end{array} \tag{16}$$

where N_p is the number of periods of the device. Combining equations (15) and (16) results in a systems three equations and three unknowns, a, u and v. Once found a can be plugged into equation (15) to find $\vec{X}2$ and the intersection coordinates given by $\vec{X}1 + \vec{X}2$.

6 Appendix 2: Sample Input File

Here is an example of a simple input for GEOM which has one divertor region made from three plates. The LCMS, FW and outer wall are all circles of radius 1.0, 2.0 and 2.2 meters respectively.

```
stop:             geom
                  np=  24     nf=  30      nr=  100    nz=  100
                  rnull= 2.500  znull= 0.000   ronull= 2.50   zonull= 2.50   [m]
                  jub=  80000  jub1=  3200
                  mss=  2      nss=  0
plasma_surface:   nlcms= 2
   m     n          rmnc         rmns          zmnc         zmns
   0     0       2.5000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   1     0       1.0000E+00   0.0000E+00   0.0000E+00   1.0000E+00
first_wall:       nwand= 2
   m     n          rmnc         rmns          zmnc         zmns
   0     0       2.5000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   1     0       2.0000E+00   0.0000E+00   0.0000E+00   2.0000E+00
ndd kto kpo  npper  igi
 5   60  3    30     1
plate:    1
      5       2      1      0
   6.000
      2.50000    2.00000       0.2500
      2.50000    1.70000    -100.0000
   7.000
      2.50000    2.00000       0.2500
      2.50000    1.70000    -100.0000
   8.000
      2.50000    2.00000       0.2500
      2.50000    1.70000    -100.0000
   9.000
      2.50000    2.00000       0.2500
      2.50000    1.70000    -100.0000
  10.000
      2.50000    2.00000       0.2500
      2.50000    1.70000    -100.0000
plate:    2
      5       2      1      0
   6.000
      2.50000    1.70000    -100.0000
      3.25000    1.29900    -100.0000
   7.000
      2.50000    1.70000    -100.0000
```

```
       3.30000    1.38564    -100.0000
    8.000
       2.50000    1.70000    -100.0000
       3.35000    1.47224    -100.0000
    9.000
       2.50000    2.00000    -100.0000
       3.40000    1.55885    -100.0000
   10.000
       2.50000    2.00000    -100.0000
       3.45000    1.64545    -100.0000
plate:    3
       5       2      1      0
    6.000
       3.25000    1.29900    -100.0000
       3.50000    1.73200       0.1666
    7.000
       3.30000    1.38564    -100.0000
       3.50000    1.73200       0.1666
    8.000
       3.35000    1.47224    -100.0000
       3.50000    1.73200       0.1666
    9.000
       3.40000    1.55885    -100.0000
       3.50000    1.73200       0.1666
   10.000
       3.45000    1.64545    -100.0000
       3.50000    1.73200       0.1666
    0   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0
    2   0   0   0   0   0   0   0
    3   0   0   0   0   0   0   0
    4   0   0   0   0   0   0   0
    5   0   0   0   0   0   0   0
    6   0   0   0   0   0   0   0
    7   0   0   0   0   0   0   0
    8   0   0   0   0   0   0   0
    9   0   0   0   0   0   0   0
    10  0   0   0   0   0   0   0
    11  0   0   0   0   0   0   0
    12  1   1   0   0   0   0   0
    13  1   1   0   0   0   0   0
    14  1   1   0   0   0   0   0
    15  1   1   0   0   0   0   0
    16  1   1   0   0   0   0   0
    17  1   1   0   0   0   0   0
```

18	1	1	0	0	0	0	0
19	1	1	0	0	0	0	0
20	1	1	0	0	0	0	0
21	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0
1	3	1	2	3	0		

7 Acknowledgements

The authors would like to thank Dr. Don Steiner for editing this document.

References

[1] Strumberger, E., *Nuclear Fusion*, **40**, 1697, 2000.

[2] Hirshman, S.P., Lee, D.K., *Comput. Phys. Commun.*, **39**, 161, 1986

[3] Hirshman, S.P., van Rij, W.I., Merkel, P., *Comput. Phys. Commun.*, **43**, 143, 1986

[4] Hirshman, S.P., Meier, H.K., *Phys. Fluids*, **28**, 1387, 1985

[5] Strumberger, E., Merkel, P., Schwarz, E., Tichmann, C., Laboratory Report IPP 5/100, Garching, 2002, http: //www.ipp.mpg.de/netreports/ipp-report_5_100.ps

[6] Strumberger, E., *Nuclear Fusion*, **37**, 19, 1997

[7] Nuhrenberg, J., Strumberger, E., *Contributions to Plasma Physics*, **32**, 204, 1992

[8] Gourdon, C., "Programme Optimise de Calculs Numeriques Dans les Configurations Magnetique Toriodales",CEN Fontenay aux Roses, 1970