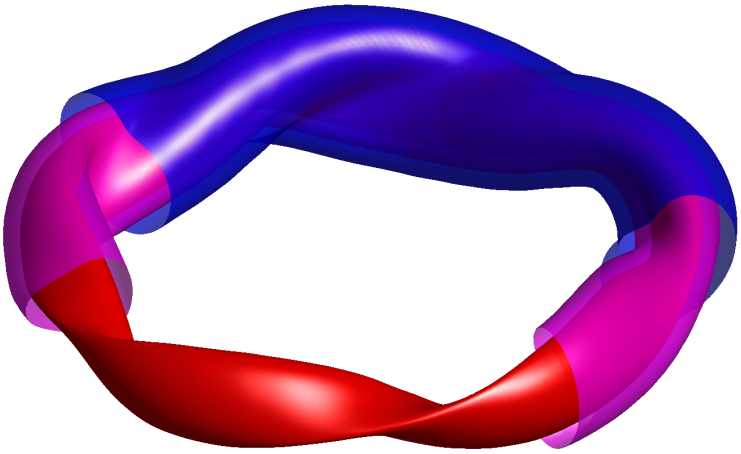
BDISTRIB User Manual



Revised August 25, 2015

Contents

CHAPTER 1

Overview

This program computes efficiency-ordered distributions of magnetic fields as described in [1]. The main method is described on page 12, and some key definitions are given on page 9.

There are some similarities between this program and the `NESCOIL` code used to compute stellarator coils [2]. `NESCOIL` effectively computes the 'inductance' between a plasma surface and a coil surface, meaning the linear relationship by which the current potential on the coil surface gives rise to a normal component of the magnetic field $\mathbf{B}$ on the plasma surface. `NESCOIL` takes the normal component of $\mathbf{B}$ on the plasma surface associated with net poloidal coil current plus plasma current, and effectively multiplies the result by the (pseudo)inverse of the inductance matrix, yielding the current potential required to achieve a magnetic surface (zero normal component of $\mathbf{B}$) on the plasma surface. In `bdistrib`, however, we do not consider net coil current or plasma current, and we compute two inductance matrices instead of one. The two inductance matrices in `bdistrib` refer to the same outer surface (on which the current potential is defined), but to different surfaces on which the normal $\mathbf{B}$ is evaluated. Multiplying one inductance matrix by the (pseudo)inverse of the other gives the so-called 'transfer matrix'. Applying a singular value decomposition (SVD) to this transfer matrix, the singular vectors represent modes of the magnetic field, and the singular values represent the efficiency by which these modes propagate from one toroidal surface to the other.

In `bdistrib` the three toroidal surfaces are named 'plasma', 'middle', and 'outer', in order from the innermost to outermost. The 'plasma' surface can correspond to the outermost surface of a `vmec` equilibrium, or it can be a plain circular toroidal surface. The name of the relevant surface appears as a suffix on most variables.

On each surface we use a poloidal angle u and a toroidal angle v, defined as in `NESCOIL`. The coordinate u lies in the range $[0, 1]$. The surfaces have `nfp` identical toroidal periods, and an increase in v by 1 corresponds to 1 of these periods. Thus, v increases by `nfp` in a complete toroidal revolution. In the output file, there is an array `v` corresponding to one toroidal period, as well as an array `vl` corresponding to all `nfp` toroidal periods. Note that v is proportional to the standard cylindrical angle ϕ: $\phi = 2\pi v/\texttt{nfp}$. Generally, field lines are not straight in the (u, v) coordinates. On the plasma surface, u is identical to the `vmec` poloidal angle divided by 2π, while the `vmec` toroidal angle differs by $2\pi/\texttt{nfp}$compared to v.

In the various variable names in the code and output file, 'r' refers to the position vector, not to a radius. In various arrays with a dimension of length 3, this dimension always corresponds to Cartesian coordinates (x, y, z).

The 'normal' quantities in the code and output file refer to the surface normal vector N = (dr/dv) cross (dr/du) as in the NESCOIL paper. Note that this vector does not have unit magnitude.

1.1 Required libraries

- `LIBSTELL` (for reading `vmec wout` files, and for the `ezcdf` subroutines)
- `NetCDF` (for writing the output file)
- `BLAS/LAPACK` (for matrix multiplication and the SVD subroutine)

If `OpenMP` is available, calculations with the code are parallelized. The plotting and testing functions use `python`, `numpy`, and `scipy`. The plotting routines `bdistribPlot` and `compareSingularValuesPlot` use `matplotlib`.

1.2 Parallelization

The code does not use `MPI`, and so it runs on a single computing node. However, it is possible to use multiple threads on the node to accelerate computations. The multi-threaded parallelization is done in part using `OpenMP` and in part using a multi-threaded `BLAS` routine.

The slowest steps in `bdistrib` occur when assembling the two inductance matrices. For each of these two matrices, there are two slow steps. The first is computation of the magnetic dipole formula between each pair of points on the two toroidal surfaces. The loop for this computation is parallelized using `OpenMP`. The other slow step is the integration of this result against each of the basis functions, which is done using matrix multiplication with the `BLAS` subroutine `DGEMM`. To parallelize this step you can link `bdistrib` with a multi-threaded `BLAS` library, such as the Intel Math Kernel Library (MKL).

1.3 `make test`

To test that your `bdistrib` executable is working, you can run `make test`. Doing so will run `bdistrib` for some or all of the examples in the `examples/` directories. After each example completes, several of the output quantities (typically the singular values of the transfer matrix) will be checked, using the `tests.py` script in the example's directory. The `make test` feature is very useful when making changes to the code, since it allows you to check that your code modifications have not broken anything and that previous results can be recovered.

If you run `make retest`, no new runs of `bdistrib` will be performed, but the `tests.py` script will be run on any existing output files in the `/examples/` directories.

1.4 Questions, Bugs, and Feedback

We welcome any contributions to the code or documentation. For write permission to the repository, or to report any bugs, provide feedback, or ask questions, contact Matt Landreman at `matt.`

landreman@gmail.com

CHAPTER 2

Theory

In this section we detail the theoretical background of the `bdistrib` code. Our discussion closely follows Ref [2] and sections 2.13 - 2.3 of Ref [1].

2.1 Current potential

Consider a surface defined by $\mathbf{r}(u, v)$, i.e. the position vector $\mathbf{r}$ along the surface is parametrized by two coordinates u and v. A vector normal to the surface is

$$\mathbf{N} = \frac{\partial \mathbf{r}}{\partial v} \times \frac{\partial \mathbf{r}}{\partial u}. \tag{2.1}$$

(We will follow the same sign convention as [2].) This vector does not generally have unit magnitude; a unit normal vector is

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{n}|}. \tag{2.2}$$

A divergence-free surface current on the surface must have the form

$$\mathbf{K} = \mathbf{n} \times \nabla \Phi \tag{2.3}$$

for a "current potential" Φ. If we now specialize to the case of a toroidal surface, Φ may be not be singly valued if there is net poloidal or toroidal current on the surface. Here we restrict our attention to the single-valued part of Φ.

2.2 Magnetic dipole formula

Let us now examine the magnetic field produced by the surface current (2.3). We begin with the Biot-Savart law for a surface current:

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int d^2 r' \frac{\mathbf{K}(\mathbf{r}') \times (\mathbf{r} - \mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|^3}. \tag{2.4}$$

We specialize to the case in which $u \in [0,1)$ is the poloidal coordinate, and the toroidal coordinate v increases from 0 to 1 in a single toroidal period, where there are n_p identical toroidal periods. (For instance, $n_p = 5$ for W7-X.) In both `vmec` and `bdistrib`, n_p is called `nfp`. Then any area integral on the toroidal surface (such as the integral in (2.4)) can be expressed as

$$\int d^2r\, Q = \sum_{\ell=1}^{n_p} \int_0^1 du \int_0^1 dv \left| \frac{\partial \mathbf{r}}{\partial u} \times \frac{\partial \mathbf{r}}{\partial v} \right| Q = \sum_{\ell=1}^{n_p} \int_0^1 du \int_0^1 dv |\mathbf{N}| Q \tag{2.5}$$

for any quantity Q. Thus, (2.4) becomes

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \sum_{\ell'=1}^{n_p} \int_0^1 du' \int_0^1 dv' \frac{1}{|\mathbf{r}-\mathbf{r}'|} \left[\mathbf{N}' \times \nabla' \Phi'\right] \times (\mathbf{r}-\mathbf{r}'). \tag{2.6}$$

We use primes to denote quantities on the surface with the current potential, to distinguish these quantities from the coordinates at which $\mathbf{B}$ is evaluated.

Now observe that $\nabla' \Phi' = (\nabla' u') \partial \Phi'/\partial u' + (\nabla' v') \partial \Phi'/\partial v'$. Also observe (suppressing primes temporarily to reduce clutter)

$$\mathbf{N} \times \nabla u = \left(\frac{\partial \mathbf{r}}{\partial v} \times \frac{\partial \mathbf{r}}{\partial u}\right) \times \nabla u = \left(\frac{\partial \mathbf{r}}{\partial u}\right) \underbrace{\left(\frac{\partial \mathbf{r}}{\partial v} \cdot \nabla u\right)}_{=0} - \left(\frac{\partial \mathbf{r}}{\partial v}\right) \underbrace{\left(\frac{\partial \mathbf{r}}{\partial u} \cdot \nabla u\right)}_{=1} = -\frac{\partial \mathbf{r}}{\partial v} \tag{2.7}$$

and

$$\mathbf{N} \times \nabla v = \left(\frac{\partial \mathbf{r}}{\partial v} \times \frac{\partial \mathbf{r}}{\partial u}\right) \times \nabla v = \left(\frac{\partial \mathbf{r}}{\partial u}\right) \underbrace{\left(\frac{\partial \mathbf{r}}{\partial v} \cdot \nabla v\right)}_{=1} - \left(\frac{\partial \mathbf{r}}{\partial v}\right) \underbrace{\left(\frac{\partial \mathbf{r}}{\partial u} \cdot \nabla v\right)}_{=0} = \frac{\partial \mathbf{r}}{\partial u}. \tag{2.8}$$

Thus, we obtain

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \sum_{\ell'=1}^{n_p} \int_0^1 du' \int_0^1 dv' \frac{1}{|\mathbf{r}-\mathbf{r}'|^3} \left[\frac{\partial \mathbf{r}'}{\partial u'} \frac{\partial \Phi'}{\partial v'} - \frac{\partial \mathbf{r}'}{\partial v'} \frac{\partial \Phi'}{\partial u'}\right] \times (\mathbf{r}-\mathbf{r}'). \tag{2.9}$$

Integrating by parts in u' and v', and using the assumption that Φ is single-valued to drop boundary contributions,

$$\begin{aligned} \mathbf{B}(\mathbf{r}) \;=\; & \frac{\mu_0}{4\pi} \sum_{\ell'=1}^{n_p} \int_0^1 du' \int_0^1 dv'\, \Phi' \left[\frac{\partial}{\partial u'} \left(\frac{1}{|\mathbf{r}-\mathbf{r}'|^3} \frac{\partial \mathbf{r}'}{\partial v'} \times (\mathbf{r}-\mathbf{r}')\right)\right. \\ & \left. - \frac{\partial}{\partial v'} \left(\frac{1}{|\mathbf{r}-\mathbf{r}'|^3} \frac{\partial \mathbf{r}'}{\partial u'} \times (\mathbf{r}-\mathbf{r}')\right)\right]. \end{aligned} \tag{2.10}$$

If the outer derivatives act on the $\partial \mathbf{r}'/\partial v'$ and $\partial \mathbf{r}'/\partial u'$ terms, the resulting contributions cancel, so we can pull these $\partial \mathbf{r}'/\partial \ldots$ terms out in front of the outer derivatives:

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \sum_{\ell'=1}^{n_p} \int_0^1 du' \int_0^1 dv'\, \Phi' \left[\frac{\partial \mathbf{r}'}{\partial v'} \times \frac{\partial}{\partial u'} \left(\frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|^3}\right) - \frac{\partial \mathbf{r}'}{\partial u'} \times \frac{\partial}{\partial v'} \left(\frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|^3}\right)\right]. \tag{2.11}$$

Next, note that for any scalar w and vector $\mathbf{a}(w)$,

$$\frac{\partial}{\partial w}\left(\frac{\mathbf{a}}{|\mathbf{a}|^3}\right) = \frac{1}{|\mathbf{a}|^3}\frac{\partial \mathbf{a}}{\partial w} - \frac{3}{|\mathbf{a}|^5}\mathbf{a}\mathbf{a}\cdot\frac{\partial \mathbf{a}}{\partial w}. \tag{2.12}$$

Using this result, and multiplying (2.11) by an arbitrary vector $\mathbf{Y}$,

$$\mathbf{B}\cdot\mathbf{Y} = \frac{\mu_0}{4\pi}\sum_{\ell'=1}^{n_p}\int_0^1 du' \int_0^1 dv'\ \Phi'\,[...] \tag{2.13}$$

2.3 Basis functions

2.4 Fourier basis

2.5 Transfer matrix

2.6 Cylindrical approximation

CHAPTER 3

Input Parameters

In this section we describe all the parameters which can be included in the input namelist.

3.1 General parameters

`transfer_matrix_option`
Type: integer
Default: 1
When it matters: Always
Meaning: Which method is used to compute the transfer matrix.

`transfer_matrix_option` = 1: 3-surface approach: parameterize the solutions to Laplace's equation using a surface current on an outer surface which is outside the control surface.

`transfer_matrix_option` = 2: 2-surface approach: parameterize the solutions to Laplace's equation using monopoles on the control surface. In this case all parameters associated with the outer surface are changed to match the middle surface.

`symmetry_option`
Type: integer
Default: 1
When it matters: Always
Meaning: Determines which set of basis functions is used.

`symmetry_option` = 1: Use $\sin(2\pi[mu + nv])$ basis functions.

`symmetry_option` = 2: Use $\cos(2\pi[mu + nv])$ basis functions.

`symmetry_option` = 3: Use both $\sin(2\pi[mu + nv])$ and $\cos(2\pi[mu + nv])$ basis functions.

---

basis_option_plasma
Type: integer
Default: 1
When it matters: Always
Meaning: Determines which basis functions are used, as well as which weight w is used to define orthogonality. This option controls only the plasma surface. Analogous options `basis_option_middle` and `basis_option_outer` exist for the other two surfaces.

`basis_option_plasma` = 1: Use $w = 1/(n_p|\mathbf{N}|)$, so the basis functions are the Fourier functions $\sqrt{2}\sin(2\pi[mu+nv])$ and/or $\sqrt{2}\cos(2\pi[mu+nv])$ (as determined by `symmetry_option`).

`basis_option_plasma` = 2: Use a constant weight, $w = 1/A$ where A is the surface area. The basis functions will be Fourier functions scaled by $\sqrt{2A/[n_p|\mathbf{N}|]}$.

`basis_option_plasma` = 3: Use a constant weight, $w = 1/A$. The basis functions will be linear combinations of the Fourier functions, as constructed using the Cholesky method.

---

pseudoinverse_thresholds
Type: real array
Default: 1e-12
When it matters: Always
Meaning: To form the pseudoinverse of the outer-to-middle inductance matrix, singular values larger than this threshold will be replaced by their inverse, whereas singular values smaller than this threshold will be replaced by zero. You can supply more than one threshold value in order to compare results for different choices. Good threshold values are in the range 1e-6 to 1e-13, i.e. small compared to 1 but larger than machine precision.

---

check_orthogonality
Type: logical
Default: `.false.`
When it matters: Always
Meaning: If this parameter is set to `.true.`, the integrals $\int d^2a\ wf_if_j$ will be computed for all i and j, to be sure the result is $\delta_{i,j}$ to machine precision.

---

n_singular_vectors_to_save
Type: integer
Default: 12
When it matters: Always
Meaning: Number of columns of U and V to save in the output file, where U and V are the matrices of left and right singular vectors of the transfer matrix, $T = U\Sigma V^T$. Regardless of this parameter, all singular *values* will be saved in the output file.

---

save_level
Type: integer

Default: 3
When it matters: Always
Meaning: Option related determining how many variables are saved in the `netCDF` output file. The larger the value, the smaller the output file.

`save_level` = 0: Save everything.

`save_level` = 1: Do not save the basis functions or K_{mn}.

`save_level` = 2: Also do not save the inductance matrices.

`save_level` = 3: Also do not save the `drdu`, `drdv`, and `normal` arrays.

`save_level` = 4: Also do not save the singular vectors as functions of (u, v). The vectors will still be saved as amplitudes of basis functions.

3.2 Resolution parameters

For any new set of surface geometries you consider, you should vary the resolution parameters in this section to make sure they are large enough. These parameters should be large enough that the code results you care about are unchanged under further resolution increases.

`nu_plasma`
Type: integer
Default: 64
When it matters: Always
Meaning: Number of grid points in poloidal angle used to evaluate the integral To do: XXX in the outer-to-plasma surface inductance matrix. Often 64 is a good value. It is resonable and common but not mandatory to use the same value for nu_plasma, `nu_middle`, and `nu_outer`.

`nu_middle`
Type: integer
Default: 64
When it matters: Always
Meaning: Number of grid points in poloidal angle used to evaluate the integral To do: XXX in the outer-to-middle surface inductance matrix. Often 64 is a good value. It is resonable and common but not mandatory to use the same value for `nu_plasma`, nu_middle, and `nu_outer`.

`nu_outer`
Type: integer
Default: 64
When it matters: Always
Meaning: Number of grid points in poloidal angle used to evaluate the integral To do: XXX in the outer-to-plasma surface and outer-to-middle surface inductance matrices. Often 64 is a good value. It is resonable and common but not mandatory to use the same value for `nu_plasma`, `nu_middle`,

and nu_outer.

---

nv_plasma
Type: integer
Default: 64
When it matters: Always
Meaning: Number of grid points in poloidal angle used to evaluate the integral To do: XXX in the outer-to-plasma surface inductance matrix. A value of 64 is often good when the number of field periods is more than 1. A higher value like 256 may be needed when the number of field periods is 1. It is resonable and common but not mandatory to use the same value for nv_plasma, nv_middle, and nv_outer.

---

nv_middle
Type: integer
Default: 64
When it matters: Always
Meaning: Number of grid points in poloidal angle used to evaluate the integral To do: XXX in the outer-to-middle surface inductance matrix. A value of 64 is often good when the number of field periods is more than 1. A higher value like 256 may be needed when the number of field periods is 1. It is resonable and common but not mandatory to use the same value for nv_plasma, nv_middle, and nv_outer.

---

nv_outer
Type: integer
Default: 64
When it matters: Always
Meaning: Number of grid points in poloidal angle used to evaluate the integral To do: XXX in the outer-to-plasma surface and outer-to-middle surface inductance matrices. A value of 64 is often good when the number of field periods is more than 1. A higher value like 256 may be needed when the number of field periods is 1. It is resonable and common but not mandatory to use the same value for nv_plasma, nv_middle, and nv_outer.

---

mpol_plasma
Type: integer
Default: 8
When it matters: Always
Meaning: Maximum poloidal mode number to include for the Fourier basis functions on the plasma surface. The value can be positive or zero. It is resonable and common but not mandatory to use the same value for mpol_plasma, mpol_middle, and mpol_outer.

---

mpol_middle
Type: integer
Default: 8
When it matters: Always
Meaning: Maximum poloidal mode number to include for the Fourier basis functions on the middle

surface. The value can be positive or zero. It is resonable and common but not mandatory to use the same value for mpol_plasma, mpol_middle, and mpol_outer.

---

mpol_outer
Type: integer
Default: 8
When it matters: Always
Meaning: Maximum poloidal mode number to include for the Fourier basis functions on the outer surface. The value can be positive or zero. It is resonable and common but not mandatory to use the same value for mpol_plasma, mpol_middle, and mpol_outer.

---

ntor_plasma
Type: integer
Default: 8
When it matters: Always
Meaning: Maximum toroidal mode number to include for the Fourier basis functions on the plasma surface. The value can be positive or zero. It is resonable and common but not mandatory to use the same value for ntor_plasma, ntor_middle, and ntor_outer. You can set the three ntor parameters to zero to examine only axisymmetric modes.

---

ntor_middle
Type: integer
Default: 8
When it matters: Always
Meaning: Maximum toroidal mode number to include for the Fourier basis functions on the middle surface. The value can be positive or zero. It is resonable and common but not mandatory to use the same value for ntor_plasma, ntor_middle, and ntor_outer. You can set the three ntor parameters to zero to examine only axisymmetric modes.

---

ntor_outer
Type: integer
Default: 8
When it matters: Always
Meaning: Maximum toroidal mode number to include for the Fourier basis functions on the outer surface. The value can be positive or zero. It is resonable and common but not mandatory to use the same value for ntor_plasma, ntor_middle, and ntor_outer. You can set the three ntor parameters to zero to examine only axisymmetric modes.

---

mpol_transform_refinement
Type: real
Default: 5.0
When it matters: Only when geometry_option_plasma is 4.
Meaning: The number of poloidal mode numbers in the vmec file will be multiplied by this value when transforming from the original poloidal angle to the straight-field-line angle. Since the original vmec angle is chosen to minimize the number of Fourier modes required, more modes are required

in any other coordinate. This parameter does not affect the sets of basis functions, or the time required to assemble the inductance matrices or computed SVDs, but it does affect the time required to compute constant-offset surfaces.

---

ntor_transform_refinement
Type: real
Default: 1.0
When it matters: Only when geometry_option_plasma is 4.
Meaning: The number of toroidal mode numbers in the vmec file will be multiplied by this value when transforming from the original poloidal angle to the straight-field-line angle. Since the original vmec angle is chosen to minimize the number of Fourier modes required, more modes are required in any other coordinate. This parameter does not affect the sets of basis functions, or the time required to assemble the inductance matrices or computed SVDs, but it does affect the time required to compute constant-offset surfaces.

3.3 Geometry parameters for the plasma surface

geometry_option_plasma
Type: integer
Default: 0
When it matters: Always
Meaning: This option controls which type of geometry is used for the plasma surface.

geometry_option_plasma = 0: The plasma surface will be a plain circular torus. The major radius will be R0_plasma. The minor radius will be a_plasma.

geometry_option_plasma = 1: Identical to option 0. (This option exists just for consistency with geometry_option_middle and geometry_option_outer.)

geometry_option_plasma = 2: The plasma surface will be the last surface in the full radial grid of the vmec file specified by wout_ilename. The poloidal angle used will be the normal vmec angle which is not a straight-field-line coordinate. This is typically the best option to use for working with vmec equilibria.

geometry_option_plasma = 3: The plasma surface will be the last surface in the half radial grid of the vmec file specified by wout_filename. The poloidal angle used will be the normal vmec angle which is not a straight-field-line coordinate. This option exists so that the same flux surface can be used when comparing with geometry_option_plasma = 4.

geometry_option_plasma = 4: The plasma surface will be the last surface in the half radial grid of the vmec file specified by wout_filename. The poloidal angle used will be the straight-field-line coordinate, obtained by shifting the normal vmec poloidal angle by vmec's λ quantity. This option exists in order to examine changes when using a different poloidal coordinate compared to geometry_option_plasma = 3.

geometry_option_plasma = 5: The plasma surface will be the flux surface with normalized poloidal flux efit_psiN taken from the efit file specified by efit_filename.

R0_plasma
Type: real
Default: 10.0
When it matters: Only when geometry_option_plasma is 0 or 1.
Meaning: Major radius of the plasma surface, when this surface is a plain circular torus.

---

a_plasma
Type: real
Default: 0.5
When it matters: Only when geometry_option_plasma is 0 or 1.
Meaning: Minor radius of the plasma surface, when this surface is a plain circular torus.

---

nfp_imposed
Type: integer
Default: 1
When it matters: Only when geometry_option_plasma is 0 or 1.
Meaning: When the plasma surface is a plain circular torus, only toroidal mode numbers that are a multiple of this parameter will be considered. This parameter thus plays a role like vmec's nfp (number of field periods), and is used when nfp is not already loaded from a vmec file.

---

wout_filename
Type: string
Default: ''
When it matters: Only when geometry_option_plasma is 2, 3, or 4.
Meaning: Name of the vmec wout output file which will be used for the plasma surface. You can use either a netCDF or ASCII format file.

---

efit_filename
Type: string
Default: ''
When it matters: Only when geometry_option_plasma is 5.
Meaning: Name of the efit output file which will be used for the plasma surface.

---

efit_psiN
Type: real
Default: 0.98
When it matters: Only when geometry_option_plasma is 5.
Meaning: Value of normalized poloidal flux at which to select a flux surface from the efit input file. A value of 1 corresponds to the last closed flux surface, and 0 corresponds to the magnetic axis.

---

efit_num_modes
Type: integer

Default: 10
When it matters: Only when geometry_option_plasma is 5.
Meaning: Controls the number of Fourier modes used to represent $R(\theta)$ and $Z(\theta)$ for the shape of the plasma surface. Each of these functions will be expanded in a sum of functions $\sin(m\theta)$ and $\cos(m\theta)$, where m ranges from 0 to efit_num_modes-1.

3.4 Geometry parameters for the middle surface

geometry_option_middle
Type: integer
Default: 0
When it matters: Always
Meaning: This option controls which type of geometry is used for the middle surface.

geometry_option_middle = 0: The middle surface will be a plain circular torus. The major radius will be the same as the plasma surface: either R0_plasma if geometry_option_plasma is 0 or 1, or Rmajor_p from the vmec wout file if geometry_option_plasma is 2. The minor radius will be a_middle.

geometry_option_middle = 1: Identical to option 0, except the major radius of the middle surface will be set by R0_middle.

geometry_option_middle = 2: The middle surface will computing by expanding the plasma surface uniformly by a distance separation_middle.

geometry_option_middle = 3: The middle surface will be the 'coil' surface in the NESCOIL 'nescin' input file specified by nescin_filename_middle.

R0_middle
Type: real
Default: 10.0
When it matters: Only when geometry_option_middle is 1.
Meaning: Major radius of the middle surface, when this surface is a plain circular torus.

---

a_middle
Type: real
Default: 1.0
When it matters: Only when geometry_option_middle is 0 or 1.
Meaning: Minor radius of the middle surface, when this surface is a plain circular torus.

---

separation_middle
Type: real
Default: 0.2
When it matters: Only when geometry_option_middle is 2.
Meaning: Amount by which the middle surface is offset from the plasma surface.

---

nescin_filename_middle
Type: string
Default: ''
When it matters: Only when geometry_option_middle is 3.
Meaning: Name of a NESCOIL nescin input file. The coil surface from this file will be used as the middle surface for bdistrib.

3.5 Geometry parameters for the outer surface

Note that all parameters in this namelist are ignored when transfer_matrix_option = 2.

---

geometry_option_outer
Type: integer
Default: 0
When it matters: Always
Meaning: This option controls which type of geometry is used for the outer surface.

geometry_option_outer = 0: The outer surface will be a plain circular torus. The major radius will be R0_outer. The minor radius will be a_outer.

geometry_option_outer = 1: Identical to option 0.

geometry_option_outer = 2: The outer surface will computing by expanding the plasma surface uniformly by a distance separation_outer.

geometry_option_outer = 3: The outer surface will be the 'coil' surface from the NESCOIL 'nescin' input file specified by nescin_filename_outer.

geometry_option_outer = 4: The outer surface will be the 'coil' surface from the NESCOIL 'nescin' input file specified by nescin_filename_outer, expanded uniformly by a distance separation_outer.

R0_outer
Type: real
Default: 10.0
When it matters: Only when geometry_option_outer is 1.
Meaning: Major radius of the outer surface, when this surface is a plain circular torus.

---

a_outer
Type: real
Default: 1.5
When it matters: Only when geometry_option_outer is 0 or 1.
Meaning: Minor radius of the outer surface, when this surface is a plain circular torus.

---

separation_outer
Type: real
Default: 0.4
When it matters: Only when geometry_option_outer is 2.
Meaning: Amount by which the outer surface is offset from the plasma surface. (Not the offset between the outer and *middle* surfaces!).

---

nescin_filename_outer
Type: string
Default: ''
When it matters: Only when geometry_option_outer is 3.
Meaning: Name of a NESCOIL nescin input file. The coil surface from this file will be used as the outer surface for bdistrib.

References

[1] A. Boozer. *Nucl. Fusion*, **55**, 025001 (2015).

[2] P. Merkel. *Nucl. Fusion*, **27**, 867 (1987).