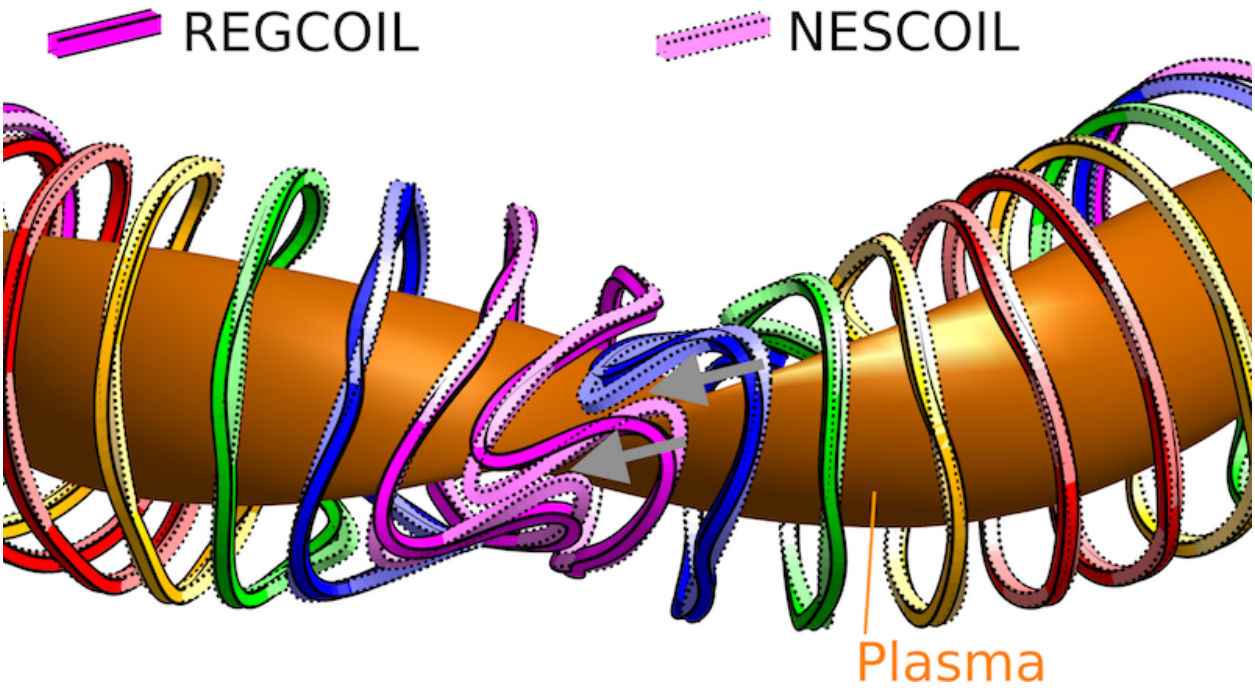
# REGCOIL User Manual




Revised May 15, 2019

# Contents

# CHAPTER 1

# Overview

This program is an implementation of the `regcoil` algorithm described in [1], and was used for the calculations in [1]. This paper is available in the Git repository for this program. In addition, this program can read output files from the `nescoil` code [2], processing the results to compute quantities like the current density and residual magnetic field normal to the target plasma surface.

## 1.1 Required libraries

- `NetCDF` (for writing the output file)
- `BLAS` (for matrix multiplication)
- `LAPACK` (for solving linear systems and for singular value decomposiiton)

Most of these libraries will be available on any high-performace computing system. `BLAS` and `LAPACK` are available on Apple Mac computers (as part of the Accelerate framework) if you install Xcode through the App store.

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

## 1.2 Cloning the repository

The source code for `regcoil` is hosted in a `git` repository at `https://github.com/landreman/regcoil`. You obtain the `regcoil` source code by cloning the repository. This requires several steps.

1. Create an account on `github.com`, and sign in to `github`.
2. Click the icon on the top right to see the drop-down menu of account options, and select the "Settings" page.

3. Click on "SSH and GPG keys" on the left, and add an SSH key for the computer you wish to use. To do this, you may wish to read see the "generating SSH keys" guide which is linked to from that page: https://help.github.com/articles/connecting-to-github-with-ssh/

4. From a terminal command line in the computer you wish to use, enter
`git clone git@github.com:landreman/regcoil.git`
to download the repository.

Any time after you have cloned the repository in this way, you can download future updates to the code by entering `git pull` from any subdirectory within your local copy.

## 1.3 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. Typically the number of threads is set by setting the environment variable `OMP_NUM_THREADS`.

## 1.4 `make test`

To test that your `regcoil` executable is working, you can run `make test`. Doing so will run `regcoil` for some or all of the examples in the `examples/` directories. After each example completes, several of the output quantities 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 `regcoil` will be performed, but the `tests.py` script will be run on any existing output files in the `/examples/` directories.

## 1.5 Units

As in `vmec`, all of `regcoil`'s input and output parameters use SI units: meters, Teslas, Amperes, and combinations thereof.

## 1.6 Plasma current

If there is current inside the plasma, then this current will contribute to the magnetic field normal to the target plasma surface, which the coils must cancel. The contribution of plasma current to the normal field is not computed directly by `regcoil`, but it can be computed using the `bnorm` code which is often distributed with `vmec`. You can then set `load_bnorm=.true.` and specify `bnorm_filename` in the `regcoil` input namelist to load the `bnorm` results into `regcoil`.

## 1.7 Matlab version

Both `fortran` and `matlab` versions of `regcoil` are included in the repository. The `matlab` version is contained in the file `regcoil.m`. For normal use you will want to use the `fortran` version, since it is much faster. The `matlab` version was originally written as a check of the `fortran` version, to verify that two independent implementations of the algorithm in different languages give identical results. The `matlab` version reads in an output file from the `fortran` version and verifies that each significant variable is identical. A few of the features in the `fortran` version are not available in the `matlab` version.

## 1.8 Plotting results

The python program `regcoilPlot` will display many of the output quantities from a single `regcoil` calculation. Results from multiple `regcoil` calculations can be compared using the python program `compareRegcoil`.

You can also make a 3D figure of the shapes of discrete coils using the `matlab` program `m20160811_01_plotCoilsFromRegcoil.m`. Two different sets of coils can be plotted together using the `matlab` program `m20160811_02_compare2CoilsetsFromRegcoil.m`. The latter program was used to generate the figure on the cover of this manual.

## 1.9 Cutting coils

Once a suitable current potential has been computed with `regcoil`, you can 'cut' discrete coils using the python script `cutCoilsFromRegcoil`. You can run this script with no arguments to see a list of the input parameters. This script will generate a coils file suitable for input to the `makegrid` code (distributed with `vmec`), which in turn generates an mgrid file used as input for free-boundary `vmec`.

## 1.10 Output quantities

The output variables are documented using metadata (the '`long_name`' attribute) in the netCDF output files `regcoil_out.<extension>.nc`. To view the available output variables, their annotations, and their values, you can run `ncdump regcoil_out.<extension>.nc | less` from the command line. Some of the most commonly used output quantities which can be found in the netCDF file are `lambda`, `chi2_B`, `chi2_K`, `max_Bnormal`, `max_K`, `chi2_B_target`, and `current_potential`.

## 1.11 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

# Input Parameters

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

## 2.1 General parameters

**general_option**
*Type*: integer
*Default*: 1
*When it matters*: Always
*Meaning*: Determines the overall flow of program execution.

general_option = 1: Compute the current potential for a range of $\lambda$.

general_option = 2: Do not compute the current potential, but rather load the current potential computed by nescoil in the file nescout_filename, compute the $\chi^2_B$ and $\chi^2_K$ for it, and save results. For this setting, Nlambda will be over-written with the number of current potential solutions found in the nescout file.

general_option = 3: Emulate nescoil's truncated singular value decomposition (TSVD) solver. The least-squares problem solved will be minimization of only $\chi^2_B$ (i.e. $\lambda = 0$.) Output quantities will be saved in the same arrays as if $\lambda$ were scanned. For this setting, Nlambda will be over-written with the number of singular values.

general_option = 4: Search for a value of the regularization weight such that a certain target is met. The target is chosen using target_option. Use this value of general_option for running regcoil inside a fixed-boundary plasma shape optimization, in which case chi2_B_target is the objective function you should minimize in the optimization.

general_option = 5: Same as 4, except that before the $\lambda$ search is carried out, the system is solved for $\lambda = 0$ and $\lambda = \infty$ to check whether the current density target is attainable. Thus, this

option takes a little more time than `general_option`=4 but is more robust.

---

**`regularization_term_option`**
*Type*: string
*Default*: `"chi2_K"`
*When it matters*: Always
*Meaning*: Determines which term is used for regularization.

`regularization_term_option = "chi2_K"`: Use $\chi^2_K$ as the regularization term, as described in the Nuclear Fusion paper.

`regularization_term_option = "Laplace-Beltrami"`: Use $\int d^2a(\nabla^2\Phi)^2$ as the regularization term, where the integral is performed over the coil surface, and $\nabla^2$ is the Laplace-Beltrami operator.

---

**`nescout_filename`**
*Type*: string
*Default*: `""`
*When it matters*: Only when `general_option`=2.
*Meaning*: Name of a `nescoil` output file which can be read in for processing.

---

**`symmetry_option`**
*Type*: integer
*Default*: 1
*When it matters*: Always
*Meaning*: Determines whether stellarator symmetry is imposed.

`symmetry_option` = 1: Force the single-valued part of the current potential to be odd in $\theta$ and $\zeta$. This option corresponds to stellarator symmetry.

`symmetry_option` = 2: Force the single-valued part of the current potential to be even in $\theta$ and $\zeta$. I'm not sure why you would ever use this option, but it is available for completeness.

`symmetry_option` = 3: No symmetry in the current potential is imposed.

---

**`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 inductance matrix.

`save_level` = 2: Also do not save the matrix $g$.

`save_level` = 3: Also do not save the normal vector or derivatives of the position vector.

---

**`load_bnorm`**
*Type*: logical
*Default*: `.false.`
*When it matters*: When `general_option`=1 or 3.
*Meaning*: Whether or not an output file from the `bnorm` code is to be loaded. Set this option to `.true.` if there is significant current in the plasma, meaning the coils will need to cancel the associated magnetic field component normal to the target plasma surface.

---

**`bnorm_filename`**
*Type*: string
*Default*: `""`
*When it matters*: When `general_option`=1 or 3 and `load_bnorm`=`.true.`.
*Meaning*: Output file from the `bnorm` code which contains the magnetic field normal to the target plasma surface associated with current inside the plasma.

---

**`net_poloidal_current_Amperes`**
*Type*: real
*Default*: 1.0
*When it matters*: If `geometry_option_plasma`=0,1, 5 or 7, i.e. if the plasma surface is not a vmec equilibrium.
*Meaning*: The number of Amperes of current the links the coil winding surface poloidally, denoted $G$ in [1]. If the plasma surface is obtained from a vmec equilibrium, then `net_poloidal_current_Amperes` will be determined instead from the `bvco` value in the vmec wout file.

---

**`net_toroidal_current_Amperes`**
*Type*: real
*Default*: 0.0
*When it matters*: Always
*Meaning*: The number of Amperes of current the links the coil winding surface toroidally, denoted $I$ in [1]. Unlike the net poloidal current, this number is never read from a wout file.

## 2.2 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.

---

**`ntheta_plasma`**
*Type*: integer

*Default*: 64
*When it matters*: Always
*Meaning*: Number of grid points in poloidal angle used to evaluate surface integrals on the plasma surface. Often 64 or 128 is a good value. It is resonable and common but not mandatory to use the same value for `ntheta_plasma` and `ntheta_coil`.

---

**`ntheta_coil`**
*Type*: integer
*Default*: 64
*When it matters*: Always
*Meaning*: Number of grid points in poloidal angle used to evaluate surface integrals on the coil winding surface. Often 64 or 128 is a good value. It is resonable and common but not mandatory to use the same value for `ntheta_plasma` and `ntheta_coil`.

---

**`nzeta_plasma`**
*Type*: integer
*Default*: 64
*When it matters*: Always
*Meaning*: Number of grid points in toroidal angle used to evaluate surface integrals on the plasma surface. Often 64 or 128 is a good value. It is resonable and common but not mandatory to use the same value for `nzeta_plasma` and `nzeta_coil`.

---

**`nzeta_coil`**
*Type*: integer
*Default*: 64
*When it matters*: Always
*Meaning*: Number of grid points in toroidal angle used to evaluate surface integrals on the coil winding surface. Often 64 or 128 is a good value. It is resonable and common but not mandatory to use the same value for `nzeta_plasma` and `nzeta_coil`.

---

**`mpol_potential`**
*Type*: integer
*Default*: 12
*When it matters*: Always
*Meaning*: Maximum poloidal mode number to include for the single-valued part of the current potential on the coil winding surface.

---

**`ntor_potential`**
*Type*: integer
*Default*: 12
*When it matters*: Always
*Meaning*: Maximum toroidal mode number to include for the single-valued part of the current potential on the coil winding surface.

---

**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 affects the time required to compute constant-offset surfaces, but does not affect the time for other calculations.

---

**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 affects the time required to compute constant-offset surfaces, but does not affect the time for other calculations.

## 2.3 Geometry parameters for the plasma surface

**geometry_option_plasma**
*Type*: integer
*Default*: 0
*When it matters*: Always
*Meaning*: This option controls how you specify the shape of the target 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. This option exists just for testing purposes.

geometry_option_plasma = 1: Identical to option 0.

geometry_option_plasma = 2: The plasma surface will be the last surface in the full 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 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 $\lambda$ 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.

geometry_option_plasma = 6: The plasma surface will be loaded from an ASCII file, specified by shape_filename_plasma. The first line of this file is ignored. The second line is an integer giving the number of Fourier modes to read. The remaining lines contain $m$, $n$, $rmnc$, $zmns$, $rmns$, $zmnc$.

geometry_option_plasma = 7: The plasma surface and Bnorm information will be loaded from an ASCII file in FOCUS format, specified by shape_filename_plasma. For more information, please look here https://princetonuniversity.github.io/FOCUS/rdsurf.pdf.

---

**shape_filename_plasma**
*Type*: string
*Default*: ""
*When it matters*: Only when geometry_option_plasma is 6 or 7.
*Meaning*: ASCII file from which to read in the plasma shape.

---

**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$-1$.

## 2.4 Geometry parameters for the coil winding surface

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

geometry_option_coil = 0: The coil 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_coil.

geometry_option_coil = 1: Identical to option 0, except the major radius of the coil surface will be set by R0_coil.

`geometry_option_coil` = 2: The coil surface will computing by expanding the plasma surface uniformly by a distance `separation`. The expanded surface will be saved to a local file specified by `nescin_filename`.

`geometry_option_coil` = 3: The coil surface will be the 'coil' surface in the `nescoil` 'nescin' input file specified by `nescin_filename`.

`geometry_option_coil` = 4: Similar to option 2, except that the poloidal angle will be changed such that the arclength (with respect to $\theta$) is independent of $\theta$ at each $\zeta$. The coil surface will computing by expanding the plasma surface uniformly by a distance `separation`. The expanded surface will be saved to a local file specified by `nescin_filename`.

---

**`R0_coil`**
*Type*: real
*Default*: 10.0
*When it matters*: Only when `geometry_option_coil` is 1.
*Meaning*: Major radius of the coil surface, when this surface is a plain circular torus.

---

**`a_coil`**
*Type*: real
*Default*: 1.0
*When it matters*: Only when `geometry_option_coil` is 0 or 1.
*Meaning*: Minor radius of the coil surface, when this surface is a plain circular torus.

---

**`separation`**
*Type*: real
*Default*: 0.2
*When it matters*: Only when `geometry_option_coil` is 2.
*Meaning*: Amount by which the coil surface is offset from the plasma surface.

---

**`nescin_filename`**
*Type*: string
*Default*: `"nescin.out"`
*When it matters*: Only when `geometry_option_coil` is 2 (write to) or 3 (read from).
*Meaning*: Name of a `nescin` file, of the sort used with the `nescoil` code. If `geometry_option_coil`=3, the coil surface from this file will be used as the coil surface for `regcoil`. If `geometry_option_coil`=2, `regcoil` will save the uniform-offset surface it computes into a file with this name.

---

**`mpol_coil_filter`**
*Type*: integer
*Default*: 24
*When it matters*: Only when `geometry_option_coil` is 2, 3, or 4.
*Meaning*: Terms in the Fourier series for $R(\theta,\zeta)$ and $Z(\theta,\zeta)$ describing the coil winding surface will be dropped if the poloidal mode number is larger than `mpol_coil_filter`.

---

**ntor_coil_filter**
*Type*: integer
*Default*: 24
*When it matters*: Only when geometry_option_coil is 2, 3, or 4.
*Meaning*: Terms in the Fourier series for $R(\theta,\zeta)$ and $Z(\theta,\zeta)$ describing the coil winding surface will be dropped if the toroidal mode number is larger than ntor_coil_filter. Specify 1, 2, 3, ... rather than nfp, 2×nfp, 3×nfp, etc.

## 2.5 Parameters related to the regularization weight

**Nlambda**
*Type*: integer
*Default*: 4
*When it matters*: Only when general_option = 1, 4, or 5.
*Meaning*: When general_option=1, Nlambda is the number of values of $\lambda$ for which the problem is solved. When general_option=4 or 5, Nlambda is the upper limit on the number of values of $\lambda$ for which the problem is solved.

---

**lambda_max**
*Type*: real
*Default*: 1.0e-13
*When it matters*: Only when general_option = 1.
*Meaning*: Maximum value of $\lambda$ for which the problem is solved.

---

**lambda_min**
*Type*: real
*Default*: 1.0e-19
*When it matters*: Only when general_option = 1.
*Meaning*: Minimum nonzero value of $\lambda$ for which the problem is solved. Note that the problem is always solved for $\lambda = 0$ in addition to the nonzero values.

---

**target_option**
*Type*: string
*Default*: "max_K"
*When it matters*: Only when general_option = 4 or 5.
*Meaning*: Controls which quantity is targeted to determine $\lambda$:

target_option = "max_K": Search for the $\lambda$ value such that the maximum current density over the winding surface equals target_value.

target_option = "chi2_K": Search for the $\lambda$ value such that $\chi^2_K$ equals target_value.

target_option = "rms_K": Search for the $\lambda$ value such that the root-mean-square current density $\left(\int d^2a\ K^2\right)^{1/2}$ (where the integral is over the current winding surface) equals target_value.

`target_option = "max_Bnormal"`: Search for the $\lambda$ value such that the maximum $\mathbf{B} \cdot \mathbf{n}$ over the plasma surface equals `target_value`.

`target_option = "chi2_B"`: Search for the $\lambda$ value such that $\chi^2_B$ equals `target_value`.

`target_option = "rms_Bnormal"`: Search for the $\lambda$ value such that the root-mean-square value of $\mathbf{B}\cdot\mathbf{n}$, i.e. $\left(\int d^2a\ B_n^2\right)^{1/2}$ (where the integral is over the plasma surface) equals `target_value`.
`target_option = "max_K_lse"`: Search for the $\lambda$ value such that $K_{\text{max,lse}}$ (the maximum approximated using the log-sum-exponent norm) equals `target_value`.

$$K_{\text{max,lse}} = \frac{1}{\texttt{target\_option\_p}} \log\left(\frac{\int_{\text{coil}} d^2A \exp\left(\texttt{target\_option\_p}K\right)}{A_{\text{coil}}}\right) \tag{2.1}$$

See `target_option_p`.
`target_option = "lp_norm_K"`: Search for the $\lambda$ value such that $||\boldsymbol{K}||_p$ ( the $L^p$ norm of $K$) equals `target_value`.

$$||\boldsymbol{K}||_p = \left(\frac{\int_{\text{coil}} d^2A\, K^{\texttt{target\_option\_p}}}{A_{\text{coil}}}\right)^{1/p} \tag{2.2}$$

See `target_option_p`.

---

**`target_option_p`**
*Type*: real
*Default*: 4.0
*When it matters*: Only when `target_option` = `"lp_norm_K"` or `"max_K_lse"`.
*Meaning*: The value of $p$ used for the $L^p$ norm or log-sum-exponent norm of $K$.

---

**`target_value`**
*Type*: real
*Default*: 8.0e6
*When it matters*: Only when `general_option` = 4 or 5.
*Meaning*: The value of the quantity specified by `target_option` that the code will attempt to match by varying $\lambda$.

---

**`lambda_search_tolerance`**
*Type*: real
*Default*: 1.0e-5
*When it matters*: Only when `general_option` = 4 or 5.
*Meaning*: Relative tolerance for the lambda root-finding.

---

## 2.6 Parameters related to adjoint solve

**`sensitivity_option`**
*Type*: integer

*Default*: 1
*When it matters*: Only when one wishes to compute derivatives of output quantities with respect to the coil winding surface parameters. Note that `sensitivity_option > 1` must be used with `general_option` = 1, 4, or 5. `sensitivity_option = 3, 4, 5` should give the same results, although 4 or 5 tend to be more efficient.
*Meaning*:
`sensitivity_option = 1`: Derivatives are not computed.
`sensitivity_option = 2`: Derivative of $\chi^2$ is computed.
`sensitivity_option = 3`: Derivatives of $\chi^2_K$, $\chi^2_B$, and $\chi^2$ are computed. This requires two adjoint solves.
`sensitivity_option = 4`: Derivatives of $\chi^2_K$, $\chi^2_B$, and $\chi^2$ are computed. An adjoint solve is used to compute the derivative of $\chi^2_K$ and the derivative of $\chi^2_B$ is computed from it. This requires one adjoint solve.
`sensitivity_option = 5`: Derivatives of $\chi^2_K$, $\chi^2_B$, and $\chi^2$ are computed. An adjoint solve is used to compute the derivative of $\chi^2_B$ and the derivative of $\chi^2_K$ is computed from it. This requires one adjoint solve.

---

**`fixed_norm_sensitivity_option`**
*Type*: logical
*Default*: false
*When it matters*: When `sensitivity_option` $> 1$.
*Meaning*: If true, derivatives of $\chi^2$, $\chi^2_B$, and $\chi^2_K$ are computed at fixed target (indicated by `target_option`) rather than at fixed $\lambda$. This option must be used with `general_option` $> 3$ and `target_option` = `"chi2_B"`, `"lp_norm_K"` or `"max_K_lse"`.

---

**`sensitivity_symmetry_option`**
*Type*: integer
*Default*: 1
*When it matters*: Symmetry assumed when computing derivative with respect to coil geometry parameters. This does not need to be the same as `symmetry_option`.
*Meaning*:
`sensitivity_symmetry_option = 1`: Only compute derivatives with respect to $r^c_{mn}$ and $z^s_{mn}$ of the winding surface using the `NESCIN` convention. This option corresponds to stellarator symmetry.
`sensitivity_symmetry_option = 2`: Only compute derivatives with respect to $r^s_{mn}$ and $z^c_{mn}$ of the winding surface using the `NESCIN` convention.
`sensitivity_symmetry_option = 3`: No symmetry in the winding surface is imposed.

---

**`nmax_sensitivity`**
*Type*: integer
*Default*: 1
*When it matters*: When `sensitivity_option` $> 1$.
*Meaning*: Derivatives of $\chi^2$, $\chi^2_B$, and $\chi^2_K$ are computed with respect to winding surface Fourier modes with $|n| \leq$ `nmax_sensitivity`.

---

**mmax_sensitivity**
*Type*: integer
*Default*: 1
*When it matters*: When sensitivity_option $> 1$.
*Meaning*: Derivatives of $\chi^2$, $\chi^2_B$, and $\chi^2_K$ are computed with respect to winding surface Fourier modes with $|m| \leq$ mmax_sensitivity.

---

**coil_plasma_dist_lse_p**
*Type*: real
*Default*: 1.0d4
*When it matters*: When sensitivity_option$> 1$.
*Meaning*: If sensitivity_option $> 1$, the derivative of the log-sum-exponent approximation to the coil-plasma distance will be computed. The value of coil_plasma_dist_lse_p determines the scaling, $p$, in the exponent for this form of the distance approximation.

$$d_{\text{min,lse}} = -\frac{1}{p}\log\left(\frac{\int_{\text{coil}} d^2A \int_{\text{plasma}} d^2A \, \exp\left(-p\sqrt{(\boldsymbol{r}_{\text{coil}} - \boldsymbol{r}_{\text{plasma}})^2}\right)}{A_{\text{coil}} A_{\text{plasma}}}\right) \tag{2.3}$$

---

CHAPTER 3

# Winding Surface Optimization with Adjoint REGCOIL

In this section we describe winding surface optimization using the adjoint REGCOIL method.

## 3.1 Overview

If an adjoint equation is solved in REGCOIL, (sensitivity_option $> 1$), then analytic derivatives of $\chi_B^2$, $\|\boldsymbol{K}\|_2$, or $K_{\max}$ are computed with respect to the Fourier coefficients defining the winding surface using the adjoint method. These derivatives are used for a gradient-based optimization method to find a winding surface which minimizes a user-defined objective function. The target plasma surface is held fixed during the optimization. The user has the option of holding a target function fixed during the optimization (such as $\chi_B^2$, $\|\boldsymbol{K}\|_2$, or $K_{\max}$) to fix the regularization parameter $\lambda$. There are also options to impose constraints, such as on the minimum coil-plasma distance. The NESCIN convention is used, and the result of the optimization is a NESCIN file with the optimal winding surface Fourier coefficients.

## 3.2 Optimization scripts

Additional parameters must be included in the REGCOIL input file outside the regcoil_nml Fortran namelist. The parameters in this namelist are read by either the scipy_optimize or nlopt_optimize python scripts, found in the windingSurfaceOptimization directory. These scripts are called with the REGCOIL input file as an argument. The REGCOIL input file in addition to any geometry files (nescin_filename, efit_filename, bnorm_filename, shape_filename_plasma) must be located in the directory from which these scripts are called.

The nlopt package must be installed in order to call nlopt_optimize. See the nlopt documentation for installation instructions. In general nlopt_optimize should be used if one wants to perform constrained optimization. Once nlopt is installed, ensure that your $PYTHONPATH includes the directory containing libnlopt.so. The directory where this is located is specified by

libdir in the file libnlopt.la. At this time nlopt_optimize has been used with nlopt 2.4.2. The scipy_optimze script utilizes the scipy.optimize package. Details on installation of scipy can be found here. The parameters relevant to each of these scripts are detailed below.

Each time that REGCOIL is called from one of the scripts, an eval_ directory will be created. The script will print the objective function and constraint functions diagnostics with each evaluation to standard output. The compareRegcoilSurface script (found in regcoil/coilOptimizationTools) can be called on two REGCOIL output files to compare the winding surfaces at 2 evaluations.

```
compareRegcoilSurface eval_0/regcoil_out.w7x.nc eval_10/regcoil_out.w7x.nc
```

Several example input files can be found in the adjointRegcoilExamples directory.

While nlopt_optimize and scipy_optimize are serial optimizers, the gradient computation in REGCOIL is performed in parallel with OpenMP. Multithreading is controlled with the OMP_NUM_THREADS environment variable.

To run scipy_optimize or nlopt_optimize from any directory, add the regcoil/coilOptimizationTools directory to your $PATH and to your $PYTHONPATH, and add the regcoil directory to your $PATH.

## 3.3 Required &regcoil_nml namelist parameters

The following items in the &regcoil namelist should be used when running adjoint REGCOIL.

- geometry_option_coil = 3 or 4
  - A nescin_filename must be specified. It is assumed that the $m = 0$ mode only includes $n \geq 0$ modes.
- sensitivity_option $> 1$ denotes an adjoint solve must be performed. If $\chi_B^2$, $\|\boldsymbol{K}\|_2$ or $K_{\text{max}}$ are included in the objective function, sensitivity_option should be $> 2$. If finite difference derivatives are used by setting grad_option = 1 or if $\chi_B^2$, $\|\boldsymbol{K}\|_2$ or $K_{\text{max}}$ are not included in the objective function, sensitivity_option can be set to 1.
- nmax_sensitivity should be set to the largest value of $n$ that should be varied in the NESCIN file. This matters if sensitivity_option $> 1$.
- nmax_sensitivity should be set to the largest value of $m$ that should be varied in the NESCIN file. This matters if sensitivity_option $> 1$.
- sensitivity_symmetry_option should be set to reflect the symmetry desired in the optimized winding surface.
- If the coil-plasma distance is to be included in the objective function or constraints, then coil_plasma_dist_lse_p should be set to the desired value for the log-sum-exponent approximation. A value in the range $10^2$ - $10^4$ is typically sufficient. At very large values the function has very steep gradients, while at small values it does not approximate the minimum function well.

- If the gradients of $\chi_B^2, \|\boldsymbol{K}\|_2$ or $K_{\max}$ are to be computed at fixed target function (specified by `target_option`) (rather than at fixed $\lambda$), `fixed_norm_sensitivity_option` should be $> 1$. The following parameters matter when `fixed_norm_sensitivity_option` $> 1$.
  - `target_option` must be `"max_K_lse"`, `"lp_norm_K"`, or `"chi2_B"`
  - `target_option_p` is a parameter in the norm defined by `target_option`

## 3.4 Coil-winding Surface Optimization Parameters

The parameters related to winding surface optimization are defined in the input file outside the `regcoil_nml` namelist.

## 3.5 Winding Surface Optimization

The following objective function is used when `nlopt_optimize` or `scipy_optimize` is called.

$$f = \texttt{scale\_factor}\Bigg( - \alpha_V V_{\text{coil}} + \alpha_S S_p - \alpha_D d_{\min} + \alpha_B \chi_B^2 + \alpha_K \|\boldsymbol{K}\|_2 + \alpha_{D,\tanh}\left(1 + \tanh\left(\left(d_{\min} - \texttt{d\_min\_target}\right)/\texttt{alpha\_D\_tanh\_scale}\right)\right)\Bigg) \tag{3.1}$$

Here $S_p$ is the spectral width,

$$S_p = \sum_{m,n} m^p \left( (r_{mn}^c)^2 + (z_{mn}^s)^2 \right), \tag{3.2}$$

$d_{\min}$ is the minimum coil-plasma distance,

$$d_{\min} = \min\left( \sqrt{\left(\boldsymbol{r}_{\text{coil}} - \boldsymbol{r}_{\text{plasma}}\right)^2} \right), \tag{3.3}$$

and $\|\boldsymbol{K}\|_2$ is the root-mean-squared current density,

$$\|\boldsymbol{K}\|_2 = \sqrt{\chi_K^2 / A_{\text{coil}}}. \tag{3.4}$$

The coefficients in $f$ are defined by the user.

**`alphaV`**
*Type*: float
*Default*: 0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called.
*Meaning*: Scaling factor for $V_{\text{coil}}$ in the objective function.

---

**alphaS**
*Type*: float
*Default*: 0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called.
*Meaning*: Scaling factor for $S_p$ in the objective function.

---

**alphaD**
*Type*: float
*Default*: 0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called.
*Meaning*: Scaling factor for $d_{\text{min}}$ in the objective function.

---

**alphaB**
*Type*: float
*Default*: 0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called.
*Meaning*: Scaling factor for $\chi^2_B$ in the objective function.

---

**alphaK**
*Type*: float
*Default*: 0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called.
*Meaning*: Scaling factor for $\|\boldsymbol{K}\|_2$ in the objective function.

---

**alphaD_tanh**
*Type*: float
*Default*: 0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called.
*Meaning*: Scaling factor for the tanh function in the objective function, which acts as a 'wall' in parameter space when $d_{\text{min}}$ reaches `d_min_target`. The scaling is set by `alphaD_tanh_scale`.

---

**alphaD_tanh_scale**
*Type*: float
*Default*: 1.0
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called and `alphaD_tanh_scale` is non-zero.
*Meaning*: Sets the scale length for the tanh function in the objective function. When this value is large, the gradients are less sharp.

---

**d_min_target**
*Type*: float
*Default*: 0.1
*When it matters*: When `nlopt_optimize` or `scipy_optimize` is being called and `alphaD_tanh_scale` is non-zero.

*Meaning*: Sets the location of the 'wall' in parameter space due to the tanh function.

---

**scaleFactor**
*Type*: float
*Default*: 1
*When it matters*: When nlopt_optimize or scipy_optimize is being called.
*Meaning*: Scaling factor for objective function.

---

### 3.5.1 Scipy Optimize

The following parameters are read if scipy_optimize is being called.
**scipy_optimize_method**
*Type*: string
*Default*: CG
*When it matters*: When scipy_optimize is being called.
*Meaning*: The method used by scipy_optimize. The following gradient-based methods are available: CG, BFGS, Newton-CG, L-BFGS-B, TNC, SLSQP, dogleg, and trust-ncp. See the scipy.optimize.minimize documentation for more information.

---

**grad_option**
*Type*: integer
*Default*: 1
*When it matters*: When scipy_optimize is being called.
*Meaning*: When grad_option $== 1$, the gradients computed by REGCOIL are used by scipy.optimize.minimize. If grad_option $== 0$, a gradient function handle is not passed to scipy.optimize, and finite differencing is used.

---

**maxiter**
*Type*: integer
*Default*: 1000
*When it matters*: When scipy_optimize is being called.
*Meaning*: Maximum number of iteration to be taken by scipy.optimize.minimize.

---

**norm**
*Type*: integer
*Default*: 2
*When it matters*: When scipy_optimize is being called.
*Meaning*: Order of norm of the gradient used by scipy.optimize.minimize to determine successful termination.

---

**gtol**
*Type*: float
*Default*: $10^{-5}$
*When it matters*: When scipy_optimize is being called.
*Meaning*: Tolerance for gradient norm required for termination.

---

**nmax**
*Type*: integer
*Default*: none
*When it matters*: When `scipy_optimize` is being called.
*Meaning*: Maximum $n$ for Fourier modes of coil winding surface parameterization in `nescin_filename`.

---

**mmax**
*Type*: integer
*Default*: none
*When it matters*: When `scipy_optimize` is being called.
*Meaning*: Maximum $m$ for Fourier modes of coil winding surface parameterization in `nescin_filename`.

---

**nmax**
*Type*: integer
*Default*: none
*When it matters*: When `nlopt_optimize` is being called.
*Meaning*: Maximum $n$ value for winding surface Fourier modes in `nescin_filename`.

---

**mmax**
*Type*: integer
*Default*: none
*When it matters*: When `nlopt_optimize` is being called.
*Meaning*: Maximum $m$ value for winding surface Fourier modes in `nescin_filename`.

### 3.5.2 NLOPT Optimize

The following parameters are read if `nlopt_optimize` is being called.

**constraint_min**
*Type*: integer
*Default*: 0
*When it matters*: When `nlopt_optimize` is being called.
*Meaning*: `constraint_min` = 0: No constraint on a minimum coil-plasma distance is enforced.
`constraint_min` = 1: Minimum coil-plasma distance is constrained to be $\leq$ `d_min`.

---

**d_min**
*Type*: float
*Default*: 0.2
*When it matters*: When `nlopt_optimize` is being called and `contraint_min` = 1.
*Meaning*: Minimum coil-plasma allowed for optimized winding surface.

---

**constraint_max_K**
*Type*: integer
*Default*: 0
*When it matters*: When `nlopt_optimize` is being called.
*Meaning*: `constraint_max_K` = 0: No constraint on $\max K$.

constraint_max_K = 1: Maximum current density is constraint to be $\leq$ max_K

---

**max_K**
*Type*: float
*Default*: 7.1e6
*When it matters*: When nlopt_optimize is being called and contraint_max_K = 1.
*Meaning*: Maximum current density allowed during winding surface optimization.

---

**constraint_rms_K**
*Type*: integer
*Default*: 0
*When it matters*: When nlopt_optimize is being called.
*Meaning*: constraint_rms_K = 0: No constraint on $\|K\|_2$.
constraint_rms_K = 1: Maximum current density is constraint to be $\leq$ rms_K

---

**rms_K**
*Type*: float
*Default*: 2.36e6
*When it matters*: When nlopt_optimize is being called and contraint_rms_K = 1.
*Meaning*: Maximum current density allowed during winding surface optimization.

---

**nlopt_method**
*Type*: string
*Default*: none
*When it matters*: When nlopt_optimize is being called.
*Meaning*: Algorithm used for gradient based winding surface optimization. The following options are supported.

- nlopt.G_MLSL_LDS
- nlopt.LD_LBFGS
- nlopt.LD_MMA
- nlopt.LD_SLSQP
- nlopt.LD_CCSAQ
- nlopt.LD_TNEWTON_PRECOND_RESTART
- nlopt.LD_VAR1

---

**omega_min**
*Type*: float
*Default*: -7
*When it matters*: When nlopt_optimize is being called.
*Meaning*: Minimum value for $r^c_{mn}$ or $z^s_{mn}$ allowed for winding surface optimization.

---

**omega_max**
*Type*: float
*Default*: 7
*When it matters*: When `nlopt_optimize` is being called.
*Meaning*: Maximum value for $r^c_{mn}$ or $z^s_{mn}$ allowed for winding surface optimization.

---

**constraint_tol**
*Type*: float
*Default*: 1e-6
*When it matters*: When `nlopt_optimize` is being called and `constraint_min` =1 or `constraint_max_K` = 1 or `constraint_rms_K`.
*Meaning*: Tolerance allowed for constraint equation to be satisfied.

---

**ftol_rel**
*Type*: float
*Default*: 1e-6
*When it matters*: When `nlopt_optimize` is being called.
*Meaning*: Optimization will stop when the relative change in the objective function $f$ is less than `ftol_rel` in successive steps.

---

## 3.6 General considerations and tips

- The results of the optimization vary widely with the input parameters. It is suggested that a user perform low-resolution optimizations with varying parameters (such as $\alpha_B$, $\alpha_S$, $\alpha_{\max K}$, and $\alpha_V$). To begin, one can run the optimization for a few evaluations to ensure that it is descending in the desired direction.

- If `general_option` = 4 or 5 (a $\lambda$ search is performed for a target function), it is not always possible to obtain a solution for $\lambda$ if the current density is too low or high. In this case, the optimization scripts adjust the `target_current_density` and will print a message to standard output. If you see that the `target_current_density` is readjusted several times, it is probably a good idea to begin the optimization with a different `target_current_density` .

- The SLSQP and CCSAQ algorithms in `nlopt` can be sensitive to the selection of `omega_min` and `omega_max`, as these set the initial step size for the optimization. If the winding surface wanders to far from the initial surface, these should be adjusted.

# References


[1] M. Landreman. An improved current potential method for fast computation of stellarator coil shapes. *Nucl. Fusion*, **57**, 046003 (2017).

[2] P. Merkel. Solution of stellarator boundary value problems with external currents. *Nucl. Fusion*, **27**, 867 (1987).