How-To Guide: Adding REGCOIL as a code to be used for an optimization target in STELLOPTV2

Update #1, (2018/2019)

Authors: J. Schmitt, M. Landreman, S. Lazerson

Please send any corrections, additions or comments to jcs0114 at auburn dot edu.

The goal of this document is to outline the steps necessary to add additional optimization variables (free-parameters) and targets to the cost function evaluation of STELLOPT. Two typical use cases are demonstrated here. The first case is the addition of a single term (a scalar) to the cost function, a single free-parameter, and an additional constraint. The cost function measures how well current potentials on the winding surface, constrained by K_RMS, minimize the target Bnorm on the target LCFS. The figure of merit is the residual error in Bnorm and the free parameter is the uiniform separation distance between the taret flux surface and the coil winding surface. The second case add multiple free parameters, (one or more) Fourier coefficients that describe a 'coil winding surface' that surrounds the target flux surface. The cost function and constraint are the same as for the uniform winding surface (minimization of Bnorm on target surface and a desired KRMS). The cost function value is repeated as necessary to attain the required number of target values in the evaluation array required by the Levenberg-Marquardt optimizer.

⇒ An external software package, REGCOIL, is required for this example. You can find the code repository on github at: https://github.com/landreman/regcoil

⇒ The user should be familiar (but not necessarily an expert) with FORTRAN, makefiles, and shell line commands (i.e. BASH) for unix-type operating systems (Linux, Ubuntu, MacOS, CentOS, etc.).

⇒ More advanced cases (linking with C/C++ codes, etc.) are not covered in this document.

Overview - What is STELLOPT going to do with REGCOIL calculations?

Case 1: Uniform Surface Separation

1. STELLOPT will call VMEC and calculate the MHD equilibrium for the specified equilibrium in the INPUT file. *Note: For optimization runs ('stellopt runs') that do not modify the plasma, (i.e. the user is targeting and modifying only REGCOIL variables), the it is advantageous to use 'VMEC2000_ONEEQ' so that it is calculated once and used for the remainder of the loop(s). Otherwise, the MHD calculation will be performed for every variable change, leading to a significant overhead cost.*
2. STELLOPT will call REGCOIL with a specified requirement on the plasma-coil spacing, REGCOIL_SEPARATION, and a desired root-mean-square current density, 'K'. REGCOIL will create a winding surface located REGCOIL_SEPARATION meters away from the plasma. REGCOIL will then find the current potential on that winding surface that has the desired 'K' while minimizing the value of $\chi_B^2 = \int dA\, B_{normal}^2$ on the plasma surface.

Case 2: Fourier (2-D) Description of a Coil Winding Surface

1. See Case 1, Step 1.
2. STELLOPT will manipulate the Fourier coefficients that describe the coil winding surface (2-D arrays given by REGCOIL_RCWS_rbound_c & REGCOIL_RCWS_zbound_z) and perform calls to the REGCOIL code to try to find the current potential on that winding surface that has the desired 'K' while minimizing the value of $\chi_B^2 = \int dA\, B_{normal}^2$ on the plasma surface.

The remainder of this document is organized as follows:

Section 1: Brief overview of STELLOPT (Execution loop, Communicators)
This Section is not complete. See the source code.

Section 2: Modifications to existing STELLOPT functions

Section 3: Additional STELLOPT functions to handle calls to REGCOIL

Section 4: Compiling/Building/Linking STELLOPT with REGCOIL

Section 5: Input file & Namelist entries

Section 1: Brief overview of STELLOPT (Execution loop, Communicators)

The flowchart of the STELLOPT execution loop is shown below. This chart highlights some of the functions in STELLOPT that will be modified to include targets, parameters, and calls to REGCOIL.

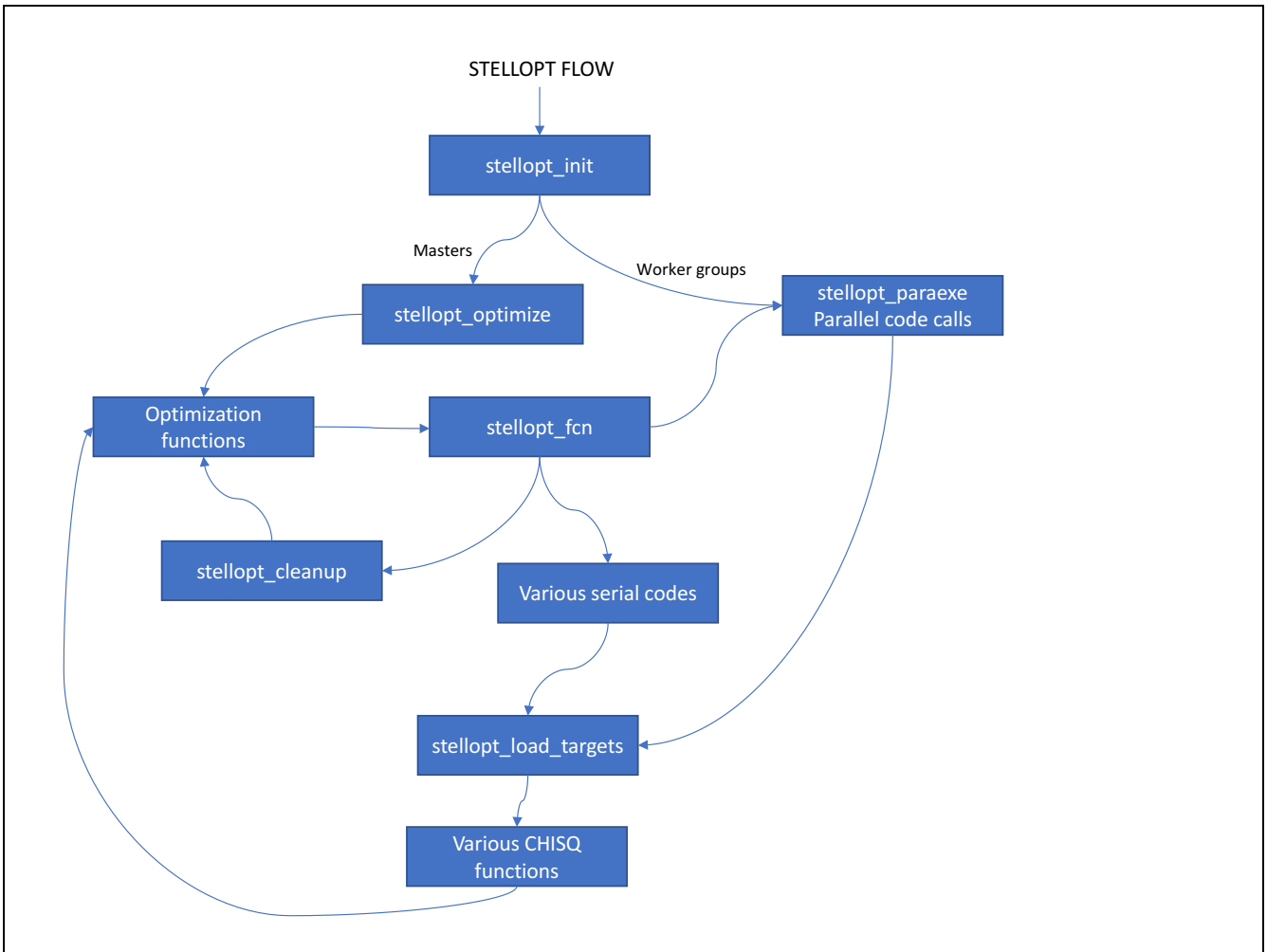

Figure 1: Flowchart of the STELLOPT execution loop.

Section 2: Modifications to existing STELLOPT functions

The following STELLOPT source files were modified:

Modules/stellopt_vars.f90
Modules/stellopt_targets.f90
General/stellopt_init.f90
Modules/stellopt_input_mod.f90
General/stellopt_fcn.f90
General/stellopt_paraexe.f90
General/stellopt_load_targets.f90
General/stellopt_clean_up.f90

A file in LIBSTELL is also modified:
LIBSTELL/Sources/Modules/vparams.f

The lines that were added to each of the STELLOPT source code files are listed below.

LIBSTELL/Sources/Modules/vparams.f

```
!
!     FOR REGCOIL WINDING SURFACE Fourier Series Representation
!
      INTEGER, PARAMETER :: mpol_rcws = 32    ! maximum poloidal mode number (min = -max)
      INTEGER, PARAMETER :: ntor_rcws = 32    ! maximum toroidal mode number (min = -max)
      ! Reserving space for the maximum number of Fourier components
      ! that might be varied/optimized. Each of RC, RS, ZC, ZS may have
      ! spectral components spanning the range of m and n in:
      !       (-mpol_rcws:mpmol_rcws,  -ntor_rcws:ntor_rcws)
      ! (this is slightly different than what is used in nescoil, where
      ! the m<0 components are not used)
      INTEGER, PARAMETER ::  mnprod_x4_rcws = 4 * (2*32+1) * (2*32+1)
```

stellopt_targets.f90

1. Variables for the 'target' and 'sigma' (or weight) were added to the module for the winding surface separation, current density, and chi2_b.
2. Unique integer indices for these targets were also added.
3. Case statements and output expressions were added to the "write_targets" subroutine.

```
STELLOPTV2/Sources/Modules/stellopt_targets.f90

.
.
.
      REAL(rprec) ::  target_regcoil_winding_surface_separation
      REAL(rprec) ::  sigma_regcoil_winding_surface_separation
      REAL(rprec),DIMENSION((2*ntor_rcws+1)*(2*mpol_rcws+1)*4) ::  target_regcoil_chi2_b,
sigma_regcoil_chi2_b
      REAL(rprec) ::  target_regcoil_current_density, sigma_regcoil_current_density

.
.
.

      INTEGER, PARAMETER :: jtarget_regcoil_chi2_b          = 504
      INTEGER, PARAMETER :: jtarget_regcoil_current_density = 5041
.
.
.

      CASE(jtarget_regcoil_chi2_b)
         WRITE(iunit, out_format) 'REGCOIL Chi^2 B'
      CASE(jtarget_regcoil_current_density)
         WRITE(iunit, out_format) 'REGCOIL Current Density on Winding Surface'
```

stellopt_init.f90

1. The variable 'nvars' is incremented if certain logical optimization flags 'lregcoil_winding…', 'lregcoil_current_...' are set to 'true' during the initialization/counting of the variable(s).
2. The variables arrays are initialized: 'vars', 'vars_min', 'vars_max','var_dex', 'diag', 'arr_dex'. This section is made of several nested loops to handle each of the possible (m,n) modes of the winding surface.

```
STELLOPTV2/Sources/General/stellopt_init.f90
.
.
.
       ! Count the number of variables and allocate the array
       iter  = 0
       nvars = 0
       SELECT CASE (TRIM(equil_type))
          CASE('vmec2000', 'flow','animec','satire','paravmec', 'parvmec', 'vboot',
'vmec2000_oneeq')

.
.
.

         CASE('vmec2000','animec','flow','satire','paravmec', 'parvmec', 'vboot',
'vmec2000_oneeq')
          ! REGCOIL options
          IF (lregcoil_winding_surface_separation_opt) nvars = nvars + 1
          IF (lregcoil_current_density_opt) nvars = nvars + 1
          DO m = -mpol_rcws, mpol_rcws
             DO n = -ntor_rcws, ntor_rcws
                IF (lregcoil_rcws_rbound_c_opt(m,n)) THEN
                   nvars = nvars + 1
                END IF
                IF (lregcoil_rcws_rbound_s_opt(m,n)) THEN
                   nvars = nvars + 1
                END IF
                IF (lregcoil_rcws_zbound_c_opt(m,n)) THEN
                   nvars = nvars + 1
                END IF
                IF (lregcoil_rcws_zbound_s_opt(m,n)) THEN
                    nvars = nvars + 1
                END IF
             END DO
          END DO
.
.
.

      END SELECT
.
.
.

      ! Read the Equilibrium Namelist and initalize the var arrays
      SELECT CASE (TRIM(equil_type))
         CASE('vmec2000','animec','flow','satire','paravmec', 'parvmec', 'vboot','
vmec2000_oneeq')
              ! Set some defaults
.
.
.
              ! Now count
              nvar_in=0
              IF (lregcoil_winding_surface_separation_opt) THEN
                 IF (lauto_domain) THEN
                    regcoil_winding_surface_separation_min = &
                        regcoil_winding_surface_separation - &
```

```
                        ABS(pct_domain*regcoil_winding_surface_separation)
                     regcoil_winding_surface_separation_max = &
                         regcoil_winding_surface_separation + &
                         ABS(pct_domain*regcoil_winding_surface_separation)
                  END IF
                  nvar_in = nvar_in + 1
                  vars(nvar_in) = regcoil_winding_surface_separation
                  vars_min(nvar_in) = regcoil_winding_surface_separation_min
                  vars_max(nvar_in) = regcoil_winding_surface_separation_max
                  var_dex(nvar_in) = iregcoil_winding_surface_separation
                  diag(nvar_in)    = dregcoil_winding_surface_separation_opt
                  arr_dex(nvar_in,1) = 1
               END IF
               IF (lregcoil_current_density_opt) THEN
                  IF (lauto_domain) THEN
                     regcoil_current_density_min = &
                         regcoil_current_density - &
                         ABS(pct_domain*regcoil_current_density)
                     regcoil_current_density_max = &
                         regcoil_current_density + &
                         ABS(pct_domain*regcoil_current_density)
                  END IF
                  nvar_in = nvar_in + 1
                  vars(nvar_in) = regcoil_current_density
                  vars_min(nvar_in) = regcoil_current_density_min
                  vars_max(nvar_in) = regcoil_current_density_max
                  var_dex(nvar_in) = iregcoil_current_density
                  diag(nvar_in)    = dregcoil_current_density_opt
                  arr_dex(nvar_in,1) = 1
               END IF
               IF (ANY(lregcoil_rcws_rbound_c_opt) ) THEN
                  DO m = -mpol_rcws,mpol_rcws
                     DO n = -ntor_rcws,ntor_rcws
                        ! IF (m==0 .and. n<=0) CYCLE
                        IF (lregcoil_rcws_rbound_c_opt(m,n)) THEN
                           IF (lauto_domain) THEN
                              regcoil_rcws_rbound_c_min(m,n) = regcoil_rcws_rbound_c(m,n) -
ABS(pct_domain*regcoil_rcws_rbound_c(m,n))
                              regcoil_rcws_rbound_c_max(m,n) = regcoil_rcws_rbound_c(m,n) +
ABS(pct_domain*regcoil_rcws_rbound_c(m,n))
                           END IF
                           nvar_in = nvar_in + 1
                           vars(nvar_in) = regcoil_rcws_rbound_c(m,n)
                           vars_min(nvar_in) = regcoil_rcws_rbound_c_min(m,n)
                           vars_max(nvar_in) = regcoil_rcws_rbound_c_max(m,n)
                           var_dex(nvar_in) = iregcoil_rcws_rbound_c
                           diag(nvar_in)    = dregcoil_rcws_rbound_c_opt(m,n)
                           arr_dex(nvar_in,1) = m
                           arr_dex(nvar_in,2) = n
                        END IF
                     END DO
                  END DO
               END IF
               IF (ANY(lregcoil_rcws_rbound_s_opt) ) THEN
                  DO m = -mpol_rcws,mpol_rcws
                     DO n = -ntor_rcws,ntor_rcws
                        ! IF (m==0 .and. n<=0) CYCLE
                        IF (lregcoil_rcws_rbound_s_opt(m,n)) THEN
                           IF (lauto_domain) THEN
                              regcoil_rcws_rbound_s_min(m,n) = regcoil_rcws_rbound_s(m,n) -
ABS(pct_domain*regcoil_rcws_rbound_s(m,n))
                              regcoil_rcws_rbound_s_max(m,n) = regcoil_rcws_rbound_s(m,n) +
ABS(pct_domain*regcoil_rcws_rbound_s(m,n))
                           END IF
                           nvar_in = nvar_in + 1
                           vars(nvar_in) = regcoil_rcws_rbound_s(m,n)
                           vars_min(nvar_in) = regcoil_rcws_rbound_s_min(m,n)
```

```
                        vars_max(nvar_in) = regcoil_rcws_rbound_s_max(m,n)
                        var_dex(nvar_in) = iregcoil_rcws_rbound_s
                        diag(nvar_in)    = dregcoil_rcws_rbound_s_opt(m,n)
                        arr_dex(nvar_in,1) = m
                        arr_dex(nvar_in,2) = n
                     END IF
                  END DO
               END DO
            END IF
            IF (ANY(lregcoil_rcws_zbound_c_opt) ) THEN
               DO m = -mpol_rcws,mpol_rcws
                  DO n = -ntor_rcws,ntor_rcws
                     ! IF (m==0 .and. n<=0) CYCLE
                     IF (lregcoil_rcws_zbound_c_opt(m,n)) THEN
                        IF (lauto_domain) THEN
                           regcoil_rcws_zbound_c_min(m,n) = regcoil_rcws_zbound_c(m,n) -
ABS(pct_domain*regcoil_rcws_zbound_c(m,n))
                           regcoil_rcws_zbound_c_max(m,n) = regcoil_rcws_zbound_c(m,n) +
ABS(pct_domain*regcoil_rcws_zbound_c(m,n))
                        END IF
                        nvar_in = nvar_in + 1
                        vars(nvar_in) = regcoil_rcws_zbound_c(m,n)
                        vars_min(nvar_in) = regcoil_rcws_zbound_c_min(m,n)
                        vars_max(nvar_in) = regcoil_rcws_zbound_c_max(m,n)
                        var_dex(nvar_in) = iregcoil_rcws_zbound_c
                        diag(nvar_in)    = dregcoil_rcws_zbound_c_opt(m,n)
                        arr_dex(nvar_in,1) = m
                        arr_dex(nvar_in,2) = n
                     END IF
                  END DO
               END DO
            END IF
            IF (ANY(lregcoil_rcws_zbound_s_opt) ) THEN
               DO m = -mpol_rcws,mpol_rcws
                  DO n = -ntor_rcws,ntor_rcws
                     ! IF (m==0 .and. n<=0) CYCLE
                     IF (lregcoil_rcws_zbound_s_opt(m,n)) THEN
                        IF (lauto_domain) THEN
                           regcoil_rcws_zbound_s_min(m,n) = regcoil_rcws_zbound_s(m,n) -
ABS(pct_domain*regcoil_rcws_zbound_s(m,n))
                           regcoil_rcws_zbound_s_max(m,n) = regcoil_rcws_zbound_s(m,n) +
ABS(pct_domain*regcoil_rcws_zbound_s(m,n))
                        END IF
                        nvar_in = nvar_in + 1
                        vars(nvar_in) = regcoil_rcws_zbound_s(m,n)
                        vars_min(nvar_in) = regcoil_rcws_zbound_s_min(m,n)
                        vars_max(nvar_in) = regcoil_rcws_zbound_s_max(m,n)
                        var_dex(nvar_in) = iregcoil_rcws_zbound_s
                        diag(nvar_in)    = dregcoil_rcws_zbound_s_opt(m,n)
                        arr_dex(nvar_in,1) = m
                        arr_dex(nvar_in,2) = n
                     END IF
                  END DO
               END DO
            END IF
.
.
… Followed by other CASE statements
```

stellopt_load_targets.f90

1. This function was modified to include checks for REGCOIL optimization targets “CHI2_B_targets”. If a target is detected (if the sigma is less than ‘bigno’) the it calls ‘chisq_regcoil_chi2_b to calculate its contribution to the cost function, chi^2. The function calculates the cost (chi), not the value of chi^2 – the optimizer handles the squaring ^2 operation.

```
STELLOPTV2/Sources/General/stellopt_load_targets.f90
.
.
.
      ! REGCOIL Coil Optimization (CHI2_B targets)
      IF (ANY(sigma_regcoil_chi2_b < bigno)) THEN
         CALL chisq_regcoil_chi2_b(target_regcoil_chi2_b, sigma_regcoil_chi2_b, ncnt,iflag)
      END IF
.
.
.
```

stellopt_fcn.f90

1. This function was modified to include reading REGCOIL variables of interest from the 'x' variable array.
2. If the flag REGCOIL is define, then stellopt_fcn, the the function will check to see if an of the regcoil chi^2 targets are desired. If so, it will call stellopt_regcoil_chi2_b..

```
STELLOPTV2/Sources/General/stellopt_fcn.f90 b/STELLOPTV2/Sources/General/stellopt_fcn.f90

.
.
.

    ! Save variables
    DO nvar_in = 1, n
.
.
.
       IF (var_dex(nvar_in) == iregcoil_winding_surface_separation) &
              regcoil_winding_surface_separation = x(nvar_in)
       IF (var_dex(nvar_in) == iregcoil_current_density) &
              regcoil_current_density = x(nvar_in)
.
.
.
     IF (var_dex(nvar_in) == iregcoil_rcws_rbound_c)
regcoil_rcws_rbound_c(arr_dex(nvar_in,1),arr_dex(nvar_in,2)) = x(nvar_in)
         IF (var_dex(nvar_in) == iregcoil_rcws_rbound_s)
regcoil_rcws_rbound_s(arr_dex(nvar_in,1),arr_dex(nvar_in,2)) = x(nvar_in)
         IF (var_dex(nvar_in) == iregcoil_rcws_zbound_c)
regcoil_rcws_zbound_c(arr_dex(nvar_in,1),arr_dex(nvar_in,2)) = x(nvar_in)
         IF (var_dex(nvar_in) == iregcoil_rcws_zbound_s)
regcoil_rcws_zbound_s(arr_dex(nvar_in,1),arr_dex(nvar_in,2)) = x(nvar_in)

.
.
.

!DEC$ IF DEFINED (REGCOIL)
         ! Skipping parallelization for now
         ! ctemp_str = 'regcoil_chi2_b'
         ! IF (sigma_regcoil_chi2_b < bigno .and. (iflag>=0)) CALL
stellopt_paraexe(ctemp_str,proc_string,lscreen)
         IF (ANY(sigma_regcoil_chi2_b < bigno)) then
           CALL stellopt_regcoil_chi2_b(lscreen, iflag)
         end if
!DEC$ ENDIF
```

stellopt_vars.f90

1. Added declarations for the REGCOIL specific control parameters and variables (logicals (l*_opt), deltas (d*_opt), mins (*_min), maxes (*_max), number of field periods, nlambda)
2. Added declarations for integer index numbers (which should match those define in stellopt_targets.f90).
3. Case statements were added for handling the print out of each of the optimization variables.

```
STELLOPTV2/Sources/Modules/stellopt_vars.f90

      LOGICAL  ::  lregcoil_winding_surface_separation_opt,&
                   lregcoil_current_density_opt
      REAL(rprec)     ::  dregcoil_winding_surface_separation_opt, &
                          dregcoil_current_density_opt
      REAL(rprec)     ::  regcoil_winding_surface_separation_min, &
                          regcoil_current_density_min
      REAL(rprec)     :: vregcoil_winding_surface_separation_max, &
                          regcoil_current_density_max
      REAL(rprec)                         :: regcoil_winding_surface_separation
      REAL(rprec)                         :: regcoil_current_density
      INTEGER :: regcoil_nlambda, regcoil_num_field_periods
      LOGICAL, DIMENSION(-mpol_rcws:mpol_rcws, &
                         -ntor_rcws:ntor_rcws) :: lregcoil_rcws_rbound_c_opt , &
                                                  lregcoil_rcws_rbound_s_opt, &
                                                  lregcoil_rcws_zbound_c_opt, &
                                                  lregcoil_rcws_zbound_s_opt
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, &
                             -ntor_rcws:ntor_rcws) :: dregcoil_rcws_rbound_c_opt , &
                                                      dregcoil_rcws_rbound_s_opt, &
                                                      dregcoil_rcws_zbound_c_opt, &
                                                      dregcoil_rcws_zbound_s_opt

      ! Regcoil Winding Surface (rcws): Boundary+min/max
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, -ntor_rcws:ntor_rcws) ::
regcoil_rcws_rbound_c, regcoil_rcws_rbound_s
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, -ntor_rcws:ntor_rcws) ::
regcoil_rcws_rbound_c_min, regcoil_rcws_rbound_s_min
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, -ntor_rcws:ntor_rcws) ::
regcoil_rcws_rbound_c_max, regcoil_rcws_rbound_s_max
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, -ntor_rcws:ntor_rcws) ::
regcoil_rcws_zbound_c, regcoil_rcws_zbound_s
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, -ntor_rcws:ntor_rcws) ::
regcoil_rcws_zbound_c_min, regcoil_rcws_zbound_s_min
      REAL(rprec), DIMENSION(-mpol_rcws:mpol_rcws, -ntor_rcws:ntor_rcws) ::
regcoil_rcws_zbound_c_max, regcoil_rcws_zbound_s_max

      CHARACTER(256)  ::  regcoil_nescin_filename
…
      INTEGER, PARAMETER ::  iregcoil_winding_surface_separation   = 5150
      INTEGER, PARAMETER ::  iregcoil_current_density   = 5151
      ! 5152 - 5159 reserved for future REGCOIIL options
      INTEGER, PARAMETER ::  iregcoil_rcws_rbound_c = 5160
      INTEGER, PARAMETER ::  iregcoil_rcws_rbound_s = 5161
      INTEGER, PARAMETER ::  iregcoil_rcws_zbound_c = 5162
      INTEGER, PARAMETER ::  iregcoil_rcws_zbound_s = 5163
.
.
.

         ! REGCOIL cases
         CASE(iregcoil_winding_surface_separation)
            WRITE(iunit,out_format) 'REGCOIL_SEPARATION: Coil winding surface separation'
         CASE(iregcoil_current_density)
            WRITE(iunit,out_format) 'REGCOIL_SEPARATION: Winding surface current density'
         CASE(iregcoil_rcws_rbound_c)
            WRITE(iunit,out_format_2DB) 'REGCOIL_RCWS_rbound_c(',var_dex1,',',var_dex2,'):
REGCOIL Winding Surface Boundary Radial Specification (COS MN)'
```

```
         CASE(iregcoil_rcws_rbound_s)
            WRITE(iunit,out_format_2DB) 'REGCOIL_RCWS_rbound_s(',var_dex1,',',var_dex2,'):
REGCOIL Winding Surface Boundary Radial Specification (SIN MN)'
         CASE(iregcoil_rcws_zbound_c)
            WRITE(iunit,out_format_2DB) 'REGCOIL_RCWS_zbound_c(',var_dex1,',',var_dex2,'):
REGCOIL Winding Surface Boundary Vertical Specification (COS MN)'
         CASE(iregcoil_rcws_zbound_s)
            WRITE(iunit,out_format_2DB) 'REGCOIL_RCWS_zbound_s(',var_dex1,',',var_dex2,'):
REGCOIL Winding Surface Boundary Vertical Specification (SIN MN)'
         ! END of REGCOIL cases
```

stellopt_input_mod.f90

1. Precompiler flags are used to prevent the compilation of sections of the code if REGCOIL is not included in the STELLOPT distribution.
2. The regcoil_variables module is used.
3. Additional variables were addied to the optimum NAMELIST for REGCOIL
4. Variables added for use in loops for REGCOIL winding surface variable processing.
5. Default values to the above variables were added.
6. Several sections of code are added to parse REGCOIL NAMELIST values.
7. A check is performed to see if REGCOIL is linked and supported. An appropriate error message is displayed.
8. Output statements were added to print out the REGCOIL optimization input and output values.

```
STELLOPTV2/Sources/Modules/stellopt_input_mod.f90

!DEC$ IF DEFINED (REGCOIL)
      USE regcoil_variables, ONLY: rc_nfp => nfp, rmnc_coil, rmns_coil, zmns_coil,
zmnc_coil, mnmax_coil, xm_coil, xn_coil, verbose, regcoil_nml
!DEC$ ENDIF
.
.
.

      NAMELIST /optimum/ …
                        lregcoil_winding_surface_separation_opt, &
                        dregcoil_winding_surface_separation_opt, &
                        lregcoil_current_density_opt, &
                        dregcoil_current_density_opt, &
                        target_regcoil_winding_surface_separation, &
                        sigma_regcoil_winding_surface_separation, &
                        target_regcoil_chi2_b, sigma_regcoil_chi2_b, &
                        target_regcoil_current_density, sigma_regcoil_current_density, &
                        regcoil_winding_surface_separation, &
                        regcoil_current_density, &
                        regcoil_nescin_filename, &
                        regcoil_num_field_periods, &
                        lregcoil_rcws_rbound_c_opt, lregcoil_rcws_rbound_s_opt, &
                        lregcoil_rcws_zbound_c_opt, lregcoil_rcws_zbound_s_opt, &
                        dregcoil_rcws_rbound_c_opt, dregcoil_rcws_rbound_s_opt, &
                        dregcoil_rcws_zbound_c_opt, dregcoil_rcws_zbound_s_opt, &
                        regcoil_rcws_rbound_c_min, regcoil_rcws_rbound_s_min, &
                        regcoil_rcws_zbound_c_min, regcoil_rcws_zbound_s_min, &
                        regcoil_rcws_rbound_c_max, regcoil_rcws_rbound_s_max, &
                        regcoil_rcws_zbound_c_max, regcoil_rcws_zbound_s_max

      ! Variables used in regcoil section to parse nescin spectrum
      INTEGER :: imn, m, n
.
.
.

      ! REGCOIL Winding surface options
      regcoil_nescin_filename = ''
      regcoil_num_field_periods = -1.0
      lregcoil_winding_surface_separation_opt    = .FALSE.
      dregcoil_winding_surface_separation_opt    = -1.0
      lregcoil_current_density_opt    = .FALSE.
      dregcoil_current_density_opt    = -1.0
      lregcoil_rcws_rbound_c_opt = .FALSE.
      lregcoil_rcws_rbound_s_opt = .FALSE.
      lregcoil_rcws_zbound_c_opt = .FALSE.
      lregcoil_rcws_zbound_s_opt = .FALSE.
      dregcoil_rcws_rbound_c_opt = -1.0
      dregcoil_rcws_rbound_s_opt = -1.0
      dregcoil_rcws_zbound_c_opt = -1.0
```

```
      dregcoil_rcws_zbound_s_opt = -1.0
.
.
.
! More REGCOIL Options
      target_regcoil_winding_surface_separation = 0.0
      sigma_regcoil_winding_surface_separation = bigno
      regcoil_winding_surface_separation = 1.0
      regcoil_winding_surface_separation_min = 1.0e-3
      regcoil_winding_surface_separation_max = 10.
      target_regcoil_current_density = 0.0
      sigma_regcoil_current_density = bigno
      regcoil_current_density = 8.0e6
      regcoil_current_density_min = 0.0
      regcoil_current_density_max = bigno
      regcoil_rcws_rbound_c_min = -bigno;  regcoil_rcws_rbound_c_max = bigno
      regcoil_rcws_rbound_s_min = -bigno;  regcoil_rcws_rbound_s_max = bigno
      regcoil_rcws_zbound_c_min = -bigno;  regcoil_rcws_zbound_c_max = bigno
      regcoil_rcws_zbound_s_min = -bigno;  regcoil_rcws_zbound_s_max = bigno
      target_regcoil_chi2_b = 0.0
      sigma_regcoil_chi2_b  = bigno
      target_regcoil_current_density = 8.0e6
      sigma_regcoil_current_density  = bigno

.
.
.
!DEC$ IF DEFINED (REGCOIL)
      IF ( ANY(sigma_regcoil_chi2_b < bigno) .and. &
           ( ANY(lregcoil_rcws_rbound_c_opt) .or. ANY(lregcoil_rcws_rbound_s_opt) .or. &
           ANY(lregcoil_rcws_zbound_c_opt) .or. ANY(lregcoil_rcws_zbound_s_opt) ) ) THEN
         rc_nfp = regcoil_num_field_periods
         regcoil_rcws_rbound_c = 0
         regcoil_rcws_rbound_s = 0
         regcoil_rcws_zbound_c = 0
         regcoil_rcws_zbound_s = 0
         IF (myid == master) THEN
            WRITE(6,*) '<----REGCOIL: Reading NESCIN Spectrum from file'
         end if
         !call regcoil_read_nescin_spectrum(regcoil_nescin_filename, (myid == master))
         verbose = (myid == master)
         ! We need to read geometry_option_coil and nescin_filename from the input namelist
before the coil surface can be loaded.
         CALL safe_open(iunit, istat, TRIM(filename), 'old', 'formatted')
         READ(iunit, nml=regcoil_nml, iostat=istat)
         CLOSE(iunit)
         call regcoil_init_coil_surface()
         IF (myid == master) THEN
            WRITE(6,*) '<----REGCOIL: Initializing winding surface with NESCIN Spectrum'
         end if
         !call regcoil_initupdate_nescin_coil_surface((myid == master))
         ! parse the rc_(r/z)mn(c/s)_stellopt arrays and populate the
regcoil_rcws_(r/z)bound_(c/s) 2D arrays
         !do ii = -mpol_rcws,mpol_rcws
         !   do jj = -ntor_rcws,ntor_rcws
         !      regcoil_rcws_rbound_c(ii, jj) = rc_rmnc_stellopt(ii,jj)
         !      regcoil_rcws_rbound_s(ii, jj) = rc_rmns_stellopt(ii,jj)
         !      regcoil_rcws_zbound_c(ii, jj) = rc_zmnc_stellopt(ii,jj)
         !      regcoil_rcws_zbound_s(ii, jj) = rc_zmns_stellopt(ii,jj)
         !   end do
         !end do


         do imn = 1, mnmax_coil
            m = xm_coil(imn)
            n = xn_coil(imn)/(-regcoil_num_field_periods) ! Convert from regcoil/vmec to
nescin convention
```

```
            IF (m < -mpol_rcws .or. m > mpol_rcws .or. n < -ntor_rcws .or. n > ntor_rcws)
THEN
               WRITE(6,*) "Error! (m,n) values in nescin file exceed mpol_rcws or
ntor_rcws."
               WRITE(6,*) "mpol_rcws=",mpol_rcws," ntor_rcws=",ntor_rcws
               WRITE(6,*) "m=",m,"  n=",n
               STOP
            END IF
            regcoil_rcws_rbound_c(m, n) = rmnc_coil(imn)
            regcoil_rcws_rbound_s(m, n) = rmns_coil(imn)
            regcoil_rcws_zbound_c(m, n) = zmnc_coil(imn)
            regcoil_rcws_zbound_s(m, n) = zmns_coil(imn)
         end do

         if (myid==master) then
            WRITE(6,*) '<----STELLOPT_INPUT_MOD: Finished parsing nescoil data and', &
                 ' assigning stellopt variables'
         end if
      END IF
!DEC$ ENDIF
      ! End of REGCOIL winding surface optimization initializion steps
.
.
.
!DEC$ ENDIF
!DEC$ IF DEFINED (REGCOIL)
      IF (myid == master .and. (ANY(sigma_regcoil_chi2_b < bigno) .or. &
                                (sigma_regcoil_current_density < bigno) )) THEN
         WRITE(6,*)         " Stellarator REGCOIL Optimization provided by: "
         WRITE(6,"(2X,A)")
"=============================================================================="
         WRITE(6,"(2X,A)") "=========                                  REGCOIL
========="
         WRITE(6,"(2X,A)") "=========                           (M. Landreman)
========="
         WRITE(6,"(2X,A)") "=========                Matt dot Landreman at gmail dot com
========="
         WRITE(6,"(2X,A)")
"=============================================================================="
         WRITE(6,*)         "    "
      END IF
!DEC$ ELSE
      IF (myid == master .and. (ANY(sigma_regcoil_chi2_b < bigno) .or. &
                                (sigma_regcoil_current_density < bigno) ) ) THEN
         sigma_regcoil_chi2_b = bigno
         sigma_regcoil_current_density = bigno
         WRITE(6,*) '!!!!!!!!!!!!!!!!!!!! WARNING !!!!!!!!!!!!!!!!!!!!!!!!!'
         WRITE(6,*) '  Coil optimization with the REGCOIL'
         WRITE(6,*) '  code has been disabled.  Coil optimziation'
         WRITE(6,*) '  has been turned off.  Contact your vendor for'
         WRITE(6,*) '  further information.'
      END IF
!DEC$ ENDIF
.
.
.
      ! REGCOIL Options
      ! This section runs if the current density, surface separation or
      ! winding surface are opitmized variables
      !
      IF ((lregcoil_current_density_opt) .or. (lregcoil_winding_surface_separation_opt)
.or.  &
          (ANY(lregcoil_rcws_rbound_s_opt)) .or. (ANY(lregcoil_rcws_rbound_c_opt)) .or. &
          (ANY(lregcoil_rcws_zbound_s_opt)) .or. (ANY(lregcoil_rcws_zbound_c_opt)) ) THEN
         WRITE(iunit,'(A)') '!-----------------------------------------------------------
---------'
         WRITE(iunit,'(A)') '!          REGCOIL OPTIMIZATION'
```

```
         WRITE(iunit,'(A)') '!-----------------------------------------------------------------
---------'
         WRITE(iunit,outflt) 'TARGET_REGCOL_CURRENT_DENSITY',target_regcoil_current_density
         WRITE(iunit,outflt) 'SIGMA_REGCOL_CURRENT_DENSITY',sigma_regcoil_current_density
         WRITE(iunit,outflt) 'REGCOIL_CURRENT_DENSITY',regcoil_current_density

         ! Options for uniform winding surface separations
         IF (lregcoil_winding_surface_separation_opt) THEN
            WRITE(iunit,outflt) &
                   'REGCOIL_WINDING_SURFACE_SEPARATION = ', &
                   regcoil_winding_surface_separation
            WRITE(iunit,outboo) 'LREGCOIL_WINDING_SURFACE_SEPARATION', &
                   lregcoil_winding_surface_separation_opt
            WRITE(iunit,outflt) 'REGCOIL_WINDING_SURFACE_SEPARATION_MIN', &
                   regcoil_winding_surface_separation_min, &
                   'REGCOIL_WINDING_SURFACE_SEPARATION_MAX', &
                   regcoil_winding_surface_separation_max
            IF (dregcoil_winding_surface_separation_opt > 0) &
                 WRITE(iunit,outflt) 'DREGCOIL_WINDING_SURFACE_SEPARATION', &
                          dregcoil_winding_surface_separation_opt
         END IF
         ! end of uniform winding surface separation options

         ! Options for current density optimization - Not completely developted/tested

         IF (lregcoil_current_density_opt) THEN
            WRITE(iunit,onevar) 'LREGCOIL_CURRENT_DENSITY', &
                   lregcoil_current_density_opt, &
                   'REGCOIL_CURRENT_DENSITY_MIN', &
                   regcoil_current_density_min, &
                   'REGCOIL_CURRENT_DENSITY_MAX', &
                  regcoil_current_density_max
            IF (dregcoil_current_density_opt > 0) &
                      WRITE(iunit,outflt) 'DREGCOIL_CURRENT_DENSITY', &
                      dregcoil_current_density_opt
         END IF
         ! end of option for current density optimization

         ! Winding surface component OR separation optimization
         IF ( (ANY(lregcoil_rcws_rbound_s_opt)) .or. (ANY(lregcoil_rcws_rbound_c_opt)) .or.
&
              (ANY(lregcoil_rcws_zbound_s_opt)) .or. (ANY(lregcoil_rcws_zbound_c_opt)) .or.
&
              lregcoil_winding_surface_separation_opt ) THEN
             DO ii = 1,UBOUND(target_regcoil_chi2_b, 1)
                IF (sigma_regcoil_chi2_b(ii) < bigno) THEN
                    WRITE(iunit,"(2(2X,A,I4.3,A,E22.14))") &
                           'TARGET_REGCOIL_CHI2_B(',ii,') = ', target_regcoil_chi2_b(ii), &
                           'SIGMA_REGCOIL_CHI2_B(',ii,') = ', sigma_regcoil_chi2_b(ii)
                END IF
             END DO
         END IF

         ! Options for winding surface (Fourier Series) variation
         IF (  (ANY(lregcoil_rcws_rbound_c_opt)) .or. (ANY(lregcoil_rcws_rbound_s_opt))
.or. &
               (ANY(lregcoil_rcws_zbound_c_opt)) .or. (ANY(lregcoil_rcws_zbound_s_opt)) )
THEN

             ! Boundary components
             ! r-boundary cos components
             DO m = LBOUND(lregcoil_rcws_rbound_c_opt,DIM=1),
UBOUND(lregcoil_rcws_rbound_s_opt,DIM=1)
                DO n = LBOUND(lregcoil_rcws_rbound_c_opt,DIM=2),
UBOUND(lregcoil_rcws_rbound_s_opt,DIM=2)
                   IF(lregcoil_rcws_rbound_c_opt(m,n) ) THEN
```

```
                        WRITE(iunit,'(A)') '! REGCOIL Winding surface R-boundary cos
component'

WRITE(iunit,"(2X,A,I4.3,A,I4.3,A,1X,'=',1X,L1,4(2X,A,I4.3,A,I4.3,A,1X,'=',1X,E19.12))") &
                                'LREGCOIL_RCWS_RBOUND_C_OPT(',m,',',n,')',
lregcoil_rcws_rbound_c_opt(m, n), &
                                'REGCOIL_RCWS_RBOUND_C(',m,',',n,')',
regcoil_rcws_rbound_c(m, n), &
                                'DREGCOIL_RCWS_RBOUND_C_OPT(',m,',',n,')',
dregcoil_rcws_rbound_c_opt(m,n), &
                                'REGCOIL_RCWS_RBOUND_C_MIN(',m,',',n,')',
regcoil_rcws_rbound_c_min(m,n), &
                                'REGCOIL_RCWS_RBOUND_C_MAX(',m,',',n,')',
regcoil_rcws_rbound_c_max(m,n)
                    END IF
                END DO
            END DO

            ! r-boundary sin components
            DO m = LBOUND(lregcoil_rcws_rbound_s_opt,DIM=1),
UBOUND(lregcoil_rcws_rbound_s_opt,DIM=1)
                DO n = LBOUND(lregcoil_rcws_rbound_s_opt,DIM=2),
UBOUND(lregcoil_rcws_rbound_s_opt,DIM=2)
                    IF(lregcoil_rcws_rbound_s_opt(m,n)  ) THEN
                        WRITE(iunit,'(A)') '! REGCOIL Winding surface R-boundary sin
component'

WRITE(iunit,"(2X,A,I4.3,A,I4.3,A,1X,'=',1X,L1,4(2X,A,I4.3,A,I4.3,A,1X,'=',1X,E19.12))") &
                                'LREGCOIL_RCWS_RBOUND_S_OPT(',m,',',n,')',
lregcoil_rcws_rbound_s_opt(m, n), &
                                'REGCOIL_RCWS_RBOUND_S(',m,',',n,')',
regcoil_rcws_rbound_s(m, n), &
                                'DREGCOIL_RCWS_RBOUND_S_OPT(',m,',',n,')',
dregcoil_rcws_rbound_s_opt(m,n), &
                                'REGCOIL_RCWS_RBOUND_S_MIN(',m,',',n,')',
regcoil_rcws_rbound_s_min(m,n), &
                                'REGCOIL_RCWS_RBOUND_S_MAX(',m,',',n,')',
regcoil_rcws_rbound_s_max(m,n)
                    END IF
                END DO
            END DO


            ! z-boundary cos components - not implemented yet
            DO m = LBOUND(lregcoil_rcws_zbound_c_opt,DIM=1),
UBOUND(lregcoil_rcws_zbound_c_opt,DIM=1)
                DO n = LBOUND(lregcoil_rcws_zbound_c_opt,DIM=2),
UBOUND(lregcoil_rcws_zbound_c_opt,DIM=2)
                    IF(lregcoil_rcws_zbound_c_opt(m,n) ) THEN
                        WRITE(iunit,'(A)') '! REGCOIL Winding surface Z-boundary cos
component'

WRITE(iunit,"(2X,A,I4.3,A,I4.3,A,1X,'=',1X,L1,4(2X,A,I4.3,A,I4.3,A,1X,'=',1X,E19.12))") &
                                'LREGCOIL_RCWS_ZBOUND_C_OPT(',m,',',n,')',
lregcoil_rcws_zbound_c_opt(m, n), &
                                'REGCOIL_RCWS_ZBOUND_C(',m,',',n,')',
regcoil_rcws_zbound_c(m, n), &
                                'DREGCOIL_RCWS_ZBOUND_C_OPT(',m,',',n,')',
dregcoil_rcws_zbound_c_opt(m,n), &
                                'REGCOIL_RCWS_ZBOUND_C_MIN(',m,',',n,')',
regcoil_rcws_zbound_c_min(m,n), &
                                'REGCOIL_RCWS_ZBOUND_C_MAX(',m,',',n,')',
regcoil_rcws_zbound_c_max(m,n)
                    END IF
                END DO
            END DO
```

```
            ! z-boundary sin components
            DO m = LBOUND(lregcoil_rcws_zbound_s_opt,DIM=1),
UBOUND(lregcoil_rcws_zbound_s_opt,DIM=1)
                DO n = LBOUND(lregcoil_rcws_zbound_s_opt,DIM=2),
UBOUND(lregcoil_rcws_zbound_s_opt,DIM=2)
                    IF( lregcoil_rcws_zbound_s_opt(m,n) ) THEN
                        WRITE(iunit,'(A)') '! REGCOIL Winding surface Z-boundary sin
component'

WRITE(iunit,"(2X,A,I4.3,A,I4.3,A,1X,'=',1X,L1,4(2X,A,I4.3,A,I4.3,A,1X,'=',1X,E19.12))") &
                                'LREGCOIL_RCWS_ZBOUND_S_OPT(',m,',',n,')',
lregcoil_rcws_zbound_s_opt(m, n), &
                                'REGCOIL_RCWS_ZBOUND_S(',m,',',n,')',
regcoil_rcws_zbound_s(m, n), &
                                'DREGCOIL_RCWS_ZBOUND_S_OPT(',m,',',n,')',
dregcoil_rcws_zbound_s_opt(m,n), &
                                'REGCOIL_RCWS_ZBOUND_S_MIN(',m,',',n,')',
regcoil_rcws_zbound_s_min(m,n), &
                                'REGCOIL_RCWS_ZBOUND_S_MAX(',m,',',n,')',
regcoil_rcws_zbound_s_max(m,n)
                    END IF
                END DO
            END DO
        END IF
        ! end of Options for winding surface (Fourier Series) variation
      END IF  ! End of REGCOIL options
```

stellopt_clean_up.f90

1. The only modification that is performed is inside of the LEV_CLENAUP or GADE_CLEANUP loop. If the winding surface is varied and the chi_B^2 is a target, then the winding surface is printed out to a file with the prefix 'regcoil_nescout.'.

```
STELLOPTV2/Sources/General/stellopt_clean_up.f90

      IF (ctype == PSO_CLEANUP) THEN
…
!DEC$ IF DEFINED (REGCOIL)
            ! OUTPUT FILES SHOULD BE WRITTEN HERE - Use the regcoil
            ! functions to write the hdf5 output file
            ! This is inside of the PSO loop. Should be
            ! duplicated, or broken out to a subroutine
            ! WRITE *, '<----- REGCOIL Output files missing -----'
!DEC$ ENDIF
…
      ELSE IF ((ctype == LEV_CLEANUP) .or. (ctype == GADE_CLEANUP)) THEN
…
!DEC$ IF DEFINED (REGCOIL)
              ! Currently inside of LEV and GADE cleanup loop, and
              ! 'Keeping the mins' section
              IF ( ANY(sigma_regcoil_chi2_b < bigno) .and. &
                 ( ANY(lregcoil_rcws_rbound_c_opt) .or. ANY(lregcoil_rcws_rbound_s_opt)
.or. &
                   ANY(lregcoil_rcws_zbound_c_opt) .or. ANY(lregcoil_rcws_zbound_s_opt) ) )
THEN
                   CALL copy_txtfile('regcoil_nescout.'//TRIM(proc_string_old),&
                                     'regcoil_nescout.'//TRIM(proc_string))
              END IF
!DEC$ ENDIF
…

      ELSE IF (ctype == LAST_GO) THEN
…
!DEC$ IF DEFINED (REGCOIL)
         ! JCS to do: If needed put regcoil items here.
!DEC$ ENDIF
```

stellopt_paraexe.f90

1. The function call to stellopt_regcoil_chi2_b is added to the list of things to do after the MHD equilibrium is calculated. Order isn't important, so long as VMEC (or other) is finished.

```
STELLOPTV2/Sources/General/stellopt_paraexe.f90

.
.
.
            CASE('regcoil_chi2_b')
               CALL stellopt_regcoil_chi2_b(lscreen,ier)
.
.
.
```

Section 3: Additional STELLOPT functions to handle calls to REGCOIL

The following source files were added to the STELLOPT project:

Chisq/chisq_regcoil_chi2_b.f90
General/stellopt_regcoil_chi2_b.f90

chisq_regcoil_chi2_b.f90
This is a new source file that needed to be created. It is a fairly similar to the other 'chisq_*.f90' source files, in that the entry and exit values of IFLAG and NITER have very specific treatments.

On typical optimization loop, the values of the chi2_Btarget (which are calculated by REGCOIL via a call in stellopt_regcoil_chi2_b2.f90.

```
STELLOPTV2/Sources/Chisq/chisq_regcoil_chi2_b.f90


!-----------------------------------------------------------------------
!     Subroutine:    chisq_regcoil_chi2_b
!     Authors:       J.C. Schmitt (Auburn/PPPL) (jcschmitt@auburn.edu)
!     Date:          2017-2018
!     Description:   Chisq routine(s) for REGCOIL.
!                    This is a 'typical' chisq routine.  In
!                    general all chisq routines should take a target
!                    variable, a sigma variable, an iteration number
!                    and error flag.
!
!                    IFLAG:
!                    On entry, If iflag is less than 1, the code returns
!                    with no further actions.
!                    On entry, if iflag is set to zero, the code should
!                    operate with no screen output.
!                    On entry, if iflag is set to a 1, the code should
!                    output to screen.
!                    On exit, negative iflag terminates execution,
!                    positive iflag, indicates error but continues, and
!                    zero indicates the code has functioned properly.
!
!                    NITER:
!                    On entry, if niter is less than 1 the
!                    code should increment the mtargets value by
!                    the number of sigmas less than bigno.
!                    On entry, if niter is equlal to -2, the value of
!                    target_dex(mtargets) will be set to
!                    jtarget_regcoil_chi2_b
!                    On entry, if niter is 0 or larger, then:
!                       increment mtargets, and
!                       assign targets, sigmas, and vals to the
!                       appropriate quantities from the target and
!                       sigma input arrays.
!
!-----------------------------------------------------------------------
      SUBROUTINE chisq_regcoil_chi2_b(target,sigma,niter,iflag)
!-----------------------------------------------------------------------
!     Libraries
!-----------------------------------------------------------------------

      USE stellopt_runtime
      USE stellopt_targets
      USE stellopt_input_mod
      USE stellopt_vars, ONLY: regcoil_nlambda
      USE vparams, ONLY: mnprod_x4_rcws
!DEC$ IF DEFINED (REGCOIL)
      USE regcoil_variables, ONLY:  chi2_B_target, nlambda, regcoil_nml
```

```
!DEC$ ENDIF
!-----------------------------------------------------------------------
!     Input/Output Variables
!
!-----------------------------------------------------------------------
      IMPLICIT NONE

      REAL(rprec), INTENT(in)    :: target(mnprod_x4_rcws)
      REAL(rprec), INTENT(in)    :: sigma(mnprod_x4_rcws)
      INTEGER,     INTENT(in)    ::  niter
      INTEGER,     INTENT(inout) ::  iflag

!-----------------------------------------------------------------------
!     Local Variables
!
!-----------------------------------------------------------------------
      INTEGER :: iunit, counter, ii

!-----------------------------------------------------------------------
!     BEGIN SUBROUTINE
!-----------------------------------------------------------------------
      IF (iflag < 0) RETURN

!DEC$ IF DEFINED (REGCOIL)
      IF (iflag == 1) THEN
          counter = 0
          DO ii = 1,mnprod_x4_rcws
            IF (sigma(ii) < bigno) counter=counter +1
          END DO
          WRITE(iunit_out,'(A,2(2X,I7))') 'REGCOIL_CHI2_B ', counter, 4
          WRITE(iunit_out,'(A)') 'TARGET  SIGMA  UNUSED  CHI'
      END IF

      IF (niter >= 0) THEN
         ! Now fill in the targets, sigmas and chi_sq
         DO ii = 1, mnprod_x4_rcws
            !IF (sigma(ii) >= bigno) CYCLE
            IF (sigma(ii) < bigno) THEN
              mtargets = mtargets + 1
              targets(mtargets) = target(ii)
              sigmas(mtargets)  = sigma(ii)
              ! The value of the results is in the chi2_B_target variable
              vals(mtargets)    = sqrt(chi2_B_target)
              ! print *, mtargets, vals(mtargets), chi2_B_target, &
              !          target(ii), target_dex(mtargets), sigmas(mtargets), sigma(ii)
              IF (iflag == 1) WRITE(iunit_out,'(4ES22.12E3)') target(ii), &
                                  sigma(ii), 0.0, vals(mtargets)
            END IF
         END DO
      ELSE
         IF (ANY(sigma < bigno)) THEN
            ! Fill in the targets
            DO ii = 1, mnprod_x4_rcws
               IF (sigma(ii) < bigno) THEN
                  mtargets = mtargets + 1
                  IF (niter == -2) THEN
                     target_dex(mtargets)=jtarget_regcoil_chi2_b
                  END IF
               END IF
            END DO
            CALL safe_open(iunit, iflag, TRIM('input.'//TRIM(id_string)), 'old',
'formatted')

            !CALL regcoil_read_input(iunit, iflag)
            READ(iunit, nml=regcoil_nml, iostat=iflag)

            ! save an internal copy of the value of nlambda here (regcoil may
```

```
            ! overwrite it)
            regcoil_nlambda = nlambda
            close(iunit)
            IF (iflag < 0) THEN
               WRITE(6,*) '!!!!!!!!!!!!ERRROR!!!!!!!!!!!!!!'
               WRITE(6,*) '  REGCOIL Namelist not found     '
               WRITE(6,*) '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
            END IF
         END IF
      END IF
!DEC$ ENDIF

      RETURN
!-----------------------------------------------------------------------
!     END SUBROUTINE
!-----------------------------------------------------------------------
      END SUBROUTINE chisq_regcoil_chi2_b
```

stellopt_regcoil_chi2_b.f90
This is a new function. It is the function that generates an appropriate winding surface and calls the various REGCOIL functions to calculate the desired output (cost function) values. In many ways, this duplicates the main 'REGCOIL' function loop (at least the parts required for the optimizations performed here). All variables needed by REGCOIL are passed through memory. Many debugging statements are added in the code..

```
STELLOPTV2/Sources/General/stellopt_regcoil_chi2_b.f90
!-----------------------------------------------------------------------
!     Subroutine:    stellopt_regcoil_chi2_b
!     Authors:       J.C.Schmitt (Auburn/PPPL) jcschmitt@auburn.edu
!     Date:          2017-2018
!     Description:   This subroutine calls the coil regularization code
!                    REGCOIL in 'target sqrt(<K^2>)' mode
!
!-----------------------------------------------------------------------
      SUBROUTINE stellopt_regcoil_chi2_b(lscreen, iflag)
!-----------------------------------------------------------------------
!     Libraries
!-----------------------------------------------------------------------
      USE stellopt_runtime
      USE stellopt_input_mod
      USE stellopt_vars
      USE equil_utils
      use vparams, only: my_mpol => mpol_rcws, my_ntor => ntor_rcws

!DEC$ IF DEFINED (REGCOIL)
      !USE regcoil_auto_regularization_solve
      !USE regcoil_build_matrices
      !USE regcoil_compute_lambda
      !USE regcoil_init_coil_surface
      USE regcoil_init_plasma_mod
      !USE regcoil_initupdate_nescin_coil_surface
      !USE regcoil_prepare_solve
      !USE regcoil_read_bnorm
      !USE regcoil_read_nescin_spectrum
      !USE regcoil_validate_input
      USE regcoil_variables
!DEC$ ENDIF

!-----------------------------------------------------------------------
!     Subroutine Parameters
!        iflag         Error flag
!-----------------------------------------------------------------------
      IMPLICIT NONE
      INTEGER, INTENT(inout) :: iflag
      LOGICAL, INTENT(inout)        :: lscreen

!-----------------------------------------------------------------------
!     Local Variables
!        iunit       File unit number
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!     Local Variables
!        istat         Error status
!        iunit         File unit number
      ! FOR REGCOIL
      INTEGER :: istat, iunit, m, n, ii, imn, nummodes1, nummodes2

!-----------------------------------------------------------------------
!     BEGIN SUBROUTINE
!-----------------------------------------------------------------------
!      IF (iflag < 0) RETURN
      !lscreen = .true.
      IF (lscreen) then
         WRITE(6,'(a)') ' -------------  REGCOIL CALCULATION  ---------'
      ENDIF
```

```
      ! WRITE(6,'(a,a)') '<---- proc_string=', proc_string
      ! WRITE(6,'(a,i4.2)') ' -------------  REGCOIL: iflag=', iflag
!DEC$ IF DEFINED (REGCOIL)
      verbose = lscreen ! Suppress REGCOIL stdout if needed

      ! IF (lscreen) WRITE(6,'(a,a)') '<---- proc_string=', proc_string
      wout_filename = 'wout_'//TRIM(proc_string)//'.nc'
      ! STELLOPT (via lmdif->stellopt_fcn or similar) will modifiy the value of
      ! regcoil_winding_surface_separation, current_density, and/or the
      ! boundary coefficients. Here, the value of the REGCOIL variables
      ! are loaded with the new values, the correct mode of operation is
      ! determiend, control variables are set, and the regcoil-functions
      ! are called
      separation = regcoil_winding_surface_separation
      target_value = regcoil_current_density

      ! Loop over all of the spectral components of the winding surface
      ! and determine the  number of modes

      IF ((ANY(lregcoil_rcws_rbound_s_opt)) .or. (ANY(lregcoil_rcws_rbound_c_opt)) .or. &
          (ANY(lregcoil_rcws_zbound_s_opt)) .or. (ANY(lregcoil_rcws_zbound_c_opt)) ) THEN
         nummodes1 = 0
         DO m = -my_mpol, my_mpol
             DO n = -my_ntor, my_ntor
                IF ( (regcoil_rcws_rbound_c(m,n) .ne. 0) .or. &
                     (regcoil_rcws_rbound_s(m,n) .ne. 0) .or. &
                     (regcoil_rcws_zbound_c(m,n) .ne. 0) .or. &
                     (regcoil_rcws_zbound_s(m,n) .ne. 0) ) THEN
                   nummodes1 = nummodes1 + 1
                END IF
             END do
         END do

         ! Now, write the modes
         CALL safe_open(iunit, istat, TRIM('regcoil_nescout.'// &
                   TRIM(proc_string)), 'replace', 'formatted')
         write(iunit,*) "Number of fourier modes in table"
         write(iunit,*) nummodes1
         write(iunit,*) "Table of fourier coefficients"
         write(iunit,*) "m,n,crc2,czs2,crs2,czc2"
         DO m = -my_mpol, my_mpol
             DO n = -my_ntor, my_ntor
                if ( (regcoil_rcws_rbound_c(m,n) .ne. 0) .or. &
                     (regcoil_rcws_rbound_s(m,n) .ne. 0) .or. &
                     (regcoil_rcws_zbound_c(m,n) .ne. 0) .or. &
                     (regcoil_rcws_zbound_s(m,n) .ne. 0) ) THEN
                   ! These are written in the same order as in a NESCIN
                   ! file: M N RC ZS RS ZC
                   write(iunit,*) m, n, &
                        regcoil_rcws_rbound_c(m,n), regcoil_rcws_zbound_s(m,n), &
                        regcoil_rcws_rbound_s(m,n), regcoil_rcws_zbound_c(m,n)
                        !rc_rmnc_stellopt(m,n), rc_zmns_stellopt(m,n), &
                        !rc_rmns_stellopt(m,n), rc_zmnc_stellopt(m,n)
                END IF
              END DO
          END DO
          DO imn = 1, mnmax_coil
             m = xm_coil(imn)
             n = xn_coil(imn)/(-nfp)
             IF (m < -my_mpol .or. m > my_mpol .or. n < -my_ntor .or. n > my_ntor) THEN
                WRITE(6,*) "Error! (m,n) in regcoil coil surface exceeds mpol_rcws or
ntor_rcws."
                STOP
             END IF
             rmnc_coil(imn) = regcoil_rcws_rbound_c(m,n)
             rmns_coil(imn) = regcoil_rcws_rbound_s(m,n)
             zmnc_coil(imn) = regcoil_rcws_zbound_c(m,n)
```

```
              zmns_coil(imn) = regcoil_rcws_zbound_s(m,n)
           END DO
           CLOSE(iunit)
      END IF

      ! regcoil will overwrite nlambda each time - need to restore it to
      ! the original value here
      nlambda = regcoil_nlambda

      ! This should *almost* be a duplicate of the main code from
      ! regcoil.f90

      ! Validate the input
      ! write(6,'(a)') '<----Validate'
      call regcoil_validate_input()
      ! write(6,'(a)') '<----Compute lambda'
      call regcoil_compute_lambda()

      ! Define the position vector and normal vector at each grid point for
      ! the surfaces:
      ! write(6,'(a)') '<----init_plasma'
      call regcoil_init_plasma()
      ! write(6,'(a)') '<----init coil surfs'
      IF ( (lregcoil_winding_surface_separation_opt) .and. &
           ((ANY(lregcoil_rcws_rbound_s_opt)) .or. (ANY(lregcoil_rcws_rbound_c_opt)) .or. &
            (ANY(lregcoil_rcws_zbound_s_opt)) .or. (ANY(lregcoil_rcws_zbound_c_opt))) )
THEN
        write(6,'(a)') 'K==========<<<<REGCOIL ERROR: Do not optimize both separation AND
Fourier series simultaneously'
      END IF

      IF (lregcoil_winding_surface_separation_opt) then
         ! write(6,'(a)') '<----regcoil_init_coil_surface'
         call regcoil_init_coil_surface()
      END IF

      IF ((ANY(lregcoil_rcws_rbound_s_opt)) .or. (ANY(lregcoil_rcws_rbound_c_opt)) .or. &
          (ANY(lregcoil_rcws_zbound_s_opt)) .or. (ANY(lregcoil_rcws_zbound_c_opt)) ) THEN
         CALL regcoil_evaluate_coil_surface()
      END IF

      ! Initialize some of the vectors and matrices needed:
      ! write(6,'(a)') '<----read bnorm'
      call regcoil_read_bnorm()
      ! write(6,'(a)') '<----build matrices'
      call regcoil_build_matrices()
      call regcoil_prepare_solve()

      ! JCS: I disabled all options except for #5 (for now)
      ! As REGCOIL development continues, future cases can
      ! be handled with case statements here.
      ! write(6,'(a)') '<----select a case'
      select case (general_option)
      !case (1)
      !   call regcoil_solve()
      !case (2)
      !   call regcoil_compute_diagnostics_for_nescout_potential()
      !case (3)
      !   call regcoil_svd_scan()
      !case (4)
      !   call regcoil_auto_regularization_solve()
      case (5)
         ! write(6,'(a)') '<----auto_reg solve'
         call regcoil_auto_regularization_solve()
         ! After regcoil_auto_reg...solve, the value we want should be
         ! in the variable 'chi2_B_target'. Normal termination of regcoil
         ! returns the achieved chi2_B (miniumum). If there is an
```

```
         ! 'error' (too high or too low of current), the chi2_B will
         ! contain the chi2_B that was achieved with infinite
         ! regularization ! (well-spaced apart, straight-ish) coils
      case default
         print *,"Invalid general_option:",general_option
         stop
      END select

      output_filename = 'regcoil_out.'// TRIM(proc_string)//'.nc'
      !call regcoil_write_output() ! For debugging it can be useful to write the
regcoil_out file.

      !=================DEBUG SECTION==========================
      !  This section is for debugging purposes. If
      !  uncommented, the regcoil input files should be written to the
      !  working job directory and an output statement(s) will be displaed
      !  to the screen, regardless of the value of 'lscreen'
      !  - JCS
      ! IF ((ANY(lregcoil_rcws_rbound_s_opt)) .or. (ANY(lregcoil_rcws_rbound_c_opt)) .or. &
      !    (ANY(lregcoil_rcws_zbound_s_opt)) .or. (ANY(lregcoil_rcws_zbound_c_opt)) ) THEN
      ! write(6,'(a)') '<----REGCOIL DEBUG safe_open'
      ! CALL safe_open(iunit, istat, TRIM('regcoil_nescout.'// &
      !           TRIM(proc_string)), 'replace', 'formatted')
      ! write(6,'(a)'), '<----debug write_output'
      ! write(iunit,*), "Number of fourier modes in table"
      ! write(iunit,*), nummodes1
      ! write(iunit,*), "Table of fourier coefficients"
      ! write(iunit,*), "m,n,crc2,czs2,crs2,czc2"

      ! ii = 0
      ! nummodes2 = 0
      ! DO m = -mpol_rcws, mpol_rcws
      !     DO n = -ntor_rcws, ntor_rcws
      !        if ( (regcoil_rcws_rbound_c(m,n) .ne. 0) .or. &
      !             (regcoil_rcws_rbound_s(m,n) .ne. 0) .or. &
      !             (regcoil_rcws_zbound_c(m,n) .ne. 0) .or. &
      !             (regcoil_rcws_zbound_s(m,n) .ne. 0) ) THEN
      !          ii = ii+1
      !          rc_xm_stellopt(ii) = m
      !          rc_xn_stellopt(ii) = n
      !          rc_rmnc_stellopt(ii) = regcoil_rcws_rbound_c(m,n)
      !          rc_rmns_stellopt(ii) = regcoil_rcws_rbound_s(m,n)
      !          rc_zmnc_stellopt(ii) = regcoil_rcws_zbound_c(m,n)
      !          rc_zmns_stellopt(ii) = regcoil_rcws_zbound_s(m,n)
      !          if ( (rc_rmnc_stellopt(ii) .ne. 0) .or.  (rc_rmns_stellopt(ii) .ne. 0)
.or. &
      !               (rc_zmnc_stellopt(ii) .ne. 0) .or.  (rc_zmns_stellopt(ii) .ne. 0) )
THEN
      !             nummodes2 = nummodes2 + 1
      !             write(iunit,*), m, n, &
      !                    rc_rmnc_stellopt(m,n), rc_rmns_stellopt(m,n), &
      !                    rc_zmnc_stellopt(m,n), rc_zmns_stellopt(m,n)
      !          END if
      !      END DO
      ! END DO
      ! if (nummodes1 .ne. nummodes2) then
      !   write(6,'(a)') '<----Hmmm....different number of modes???'
      ! END if
      ! END if
      ! print *, chi2_B_target
      ! print *,"REGCOIL complete. Total time=",totalTime,"sec."
      !=================END OF DEBUG SECTION==========================

      IF (lscreen) WRITE(6,'(a)') ' ---------------------------  REGCOIL CALCULATION DONE
---------------------'
!DEC$ ENDIF
      RETURN
```

```
!-----------------------------------------------------------------------
!     END SUBROUTINE
!-----------------------------------------------------------------------
      END SUBROUTINE stellopt_regcoil_chi2_b
```

Section 4: Compiling/Building/Linking STELLOPT with REGCOIL

The following files were modified to enable compiling/building/linking STELLOPT with REGCOIL functionality:

make.inc
The main stellopt make.inc file has several options for REGCOIL that need to be specified. These can be entered manually the shell (with 'set', 'setenv', or 'export', depending on the shell), or added to a login shell file (i.e. .bashrc, .bash_profile, .cshrc). If the login shell file is modifie, it is a good idea to restart the shell for the settings to be applied.

```
#########################################################################
#            REGCOIL Options
#########################################################################
  LREGCOIL= F
  REGCOIL_DIR = $(REGCOIL_PATH)
  REGCOIL_INC = -I$(REGCOIL_DIR)
  LIB_REGCOIL = libregcoil.a
  REGCOIL_LIB = $(REGCOIL_DIR)/$(LIB_REGCOIL) -fopenmp
```

makestelloptv2
This is the makefile for stelloptv2. It has been modified to remove (clean up) existing REGCOIL.a libraries and to (re)build them, if necessary.

1. The LIB_REGCOIL variable is added the xstelloptv2 dependency list
2. The $REGCOIL_DIR/$LIB_REGCOIL file is removed
3. REGCOIL is built. An archive (.a) library file is created
4. The makefile for REGCOIL needs to be modified to make the libregcoil.a library.

```
STELLOPTV2/makestelloptv2 b/STELLOPTV2/makestelloptv2


xstelloptv2:  $(LIB) $(LIB_VMEC) $(LIB_BEAMS3D) $(LIB_BNORM) $(LIB_BOOTSJ)
$(LIB_BOOZ) $(LIB_COBRA) $(LIB_DIAGNO) $(LIB_DKES) $(LIB_JINV) $(LIB_MGRID)
$(LIB_NEO) $(LIB_GENE) $(LIB_COILOPTPP) $(LIB_REGCOIL) $(LIB_TERPSICHORE)
$(ObjectFiles)
.
.
.

ifdef REGCOIL_DIR
        @rm $(REGCOIL_DIR)/$(LIB_REGCOIL)
.
.
.
#Construct REGCOIL library.
$(LIB_REGCOIL) :
        @cd $(REGCOIL_DIR); make; ar -cruv $(LIB_REGCOIL) *.o


….
```

Additions to the regcoil makefile:

```
lib$(TARGET).a: $(OBJ_FILES)
        ar rcs lib$(TARGET).a $(OBJ_FILES)
```

Debug/STELLOPTV2.dep
These are the changes for the Debug version of the STELLOPTV2 dependencies file.
.

```
STELLOPTV2/Debug/STELLOPTV2.dep

chisq_regcoil_chi2_b.o : \
      stellopt_runtime.o \
      stellopt_targets.o \
      stellopt_input_mod.o \
      stellopt_vars.o \
      equil_vals.o

.
.
.
stellopt_regcoil_chi2_b.o : \
      stellopt_runtime.o \
      stellopt_input_mod.o \
      stellopt_vars.o \
      equil_utils.o
```

Release/STELLOPTV2.dep
These are the changes for the Release version of the STELLOPTV2 dependencies file.

```
STELLOPTV2/Release/STELLOPTV2.dep

chisq_regcoil_chi2_b.o : \
      stellopt_runtime.o \
      stellopt_targets.o \
      stellopt_input_mod.o \
      stellopt_vars.o \
      equil_vals.o

.
.
.
stellopt_regcoil_chi2_b.o : \
      stellopt_runtime.o \
      stellopt_input_mod.o \
      stellopt_vars.o \
      equil_utils.o
```

ObjectList
The ObjectList file for STELLOPTV2 has been updated.

```
STELLOPTV2/ObjectList

.
.
.
chisq_regcoil_chi2_b.o \
.
.
.
stellopt_regcoil_chi2_b.o \
.
.
.
```

Section 5: Input file & namelist entries

See: BENCHMARKS/STELLOPT_TEST/REGCOIL/FOURIER_WINDING_SURFACE