rdcoils

Initialize the coils data with Fourier series.

[called by: focus.] [calls: .]

0.1 overview

1. If case_coils=1, then the Fourier series will be used for represent the coils.
2. The basic equations about the Fourier representation is,

$$x = X_{c,0} + \sum_{n=1}^{N} [X_{c,n} \cos(nt) + X_{s,n} \sin(nt)] , \tag{1}$$

$$y = Y_{c,0} + \sum_{n=1}^{N} [Y_{c,n} \cos(nt) + Y_{s,n} \sin(nt)] , \tag{2}$$

$$z = Z_{c,0} + \sum_{n=1}^{N} [Z_{c,n} \cos(nt) + Z_{s,n} \sin(nt)] , \tag{3}$$

0.2 Initilization

There are several ways to initialize the coils data.

1. case_init = 1 : Toroidally placing Ncoils circular coils with a radius of init_radius and current of init_current. The ith coil is placed at $\zeta = \frac{i-1}{Ncoils} \frac{2\pi}{Nfp}$.
2. case_init = 0 : Read coils data from **ext.focus** file. The format is as following. This is the most flexible way, and each coil can be different.

```
# Total number of coils
          16
#------------1--------------------------------
#coil_type symm coil_name
   1 0 Mod_001
#Nseg current Ifree Length Lfree target_length
  128 9.844910899889484E+05 1 5.889288927667147E+00 1 1.000000000000000E+00
#NFcoil
 4
#Fourier harmonics for coils ( xc; xs; yc; ys; zc; zs)
 3.044612087666170E+00 8.531153655332238E-01 4.194525679767678E-02 2.139790853335835E-02
     3.243811555342430E-03
 0.000000000000000E+00 3.542408058492299E-16 -9.108712738922674E-16 1.841880477639364E-16
     -1.172175996642087E-16
-4.456021385977147E-15 8.545613874434043E-16 -3.133154295448265E-16 1.764367073160815E-16
    -1.187904023667544E-16
 0.000000000000000E+00 -5.425716121023922E-02 -8.986316303345250E-02 -2.946386365076052E-03
     -4.487052148209031E-03
-4.293247278325474E-17 -1.303273952226587E-15 7.710821807870230E-16 -3.156539892466338E-16
    9.395672288215928E-17
 0.000000000000000E+00 9.997301975562740E-01 2.929938238054118E-02 2.436889176706748E-02
     1.013941937492003E-03
#-----------2--permanent magnet---------------
#coil_type symm coil_name
   2 0 dipole_01
# Lc ox oy oz Ic I mt mp
  1 0.0 0.0 0.0 1 1.0E6 0.0 0.0
#-----------3--backgound Bt Bz----------------
#coil_type symm coil_name
   3 0 bg_BtBz_01
# Ic I Lc Bz (Ic control I; Lc control Bz)
  1 1.0E6 0 0.0
   .
   .
   .
```

3. case_init = -1 : Get coils data from a standard coils.ext file and then Fourier decomposed (normal Fourier tansformation and truncated with $NFcoil$ harmonics)

0.3 Discretization

1. Discretizing the coils data involves massive triangular functions in nested loops. As shown in Eq.(**??**), the outside loop is for different discrete points and for each point, a loop is needed to get the summation of the harmonics.
2. To avoid calling triangular functions every operations, it's a btter idea to allocate the public triangular arrays.

$$cmt(iD, iN) = \cos(iN\,\frac{iD}{D_i}2\pi); iD = 0, coil(icoil)\%D;\; iN = 0, coil(icoil)\%N \tag{4}$$

$$smt(iD, iN) = \sin(iN\,\frac{iD}{D_i}2\pi); iD = 0, coil(icoil)\%D;\; iN = 0, coil(icoil)\%N \tag{5}$$

3. Using the concept of vectorization, we can also finish this just through matrix operations. This is in **fouriermatrix**.

```
subroutine fouriermatrix(xc, xs, xx, NF, ND)
nn(0:NF, 1:1) : matrix for N; iN
tt(1:1, 0:ND) : matrix for angle; iD/ND*2pi
nt(0:NF,0:ND) : grid for nt; nt = matmul(nn, tt)
xc(1:1, 0:NF) : cosin harmonics;
xs(1:1, 0:NF) : sin harmonics;
xx(1:1, 0:ND) : returned disrecte points;

xx = xc * cos(nt) + xs * sin(nt)
```

4. Actually, in real tests, the new method is not so fast. And parallelizations are actually slowing the speed, both for the normal and vectorized method.

rdcoils.f90 last modified on 020-01-21 16:33:55.; Focus subroutines;