FOCUSADD Theory

Nick McGreivy

February 2020

The purpose of this document is to specify the mathematical structure of the FOCUSADD code, so that interested parties can understand in detail exactly what the code is doing.

1 Introduction

FOCUSADD is a stellarator coil design code which uses automatic differentiation to compute the derivatives of the coil parameters with respect to an objective function, and uses that information to optimize the function. It can be used to optimize finite-build coils.

2 Reading in the Axis

The axis file specifies the number of Fourier modes of the magnetic axis, `NF`, the field periodicity, `NP`, as well as the sin and cosine coefficients of the axis for each Fourier mode.

The axis is an infinitely thin filament and specified with a toroidal angle, ζ, which ranges from 0 to $2\pi/N_P$. The equations for the position of the axis are

$$\begin{aligned} x(\zeta) &= \sum_{m=0}^{N_{FA}} x_{cm}\cos(m\zeta) + x_{sm}\sin(m\zeta) \\ y(\zeta) &= \sum_{m=0}^{N_{FA}} y_{cm}\cos(m\zeta) + y_{sm}\sin(m\zeta) \\ z(\zeta) &= \sum_{m=0}^{N_{FA}} z_{cm}\cos(m\zeta) + z_{sm}\sin(m\zeta) \end{aligned} \tag{1}$$

Note that the $m = 0$ component of the sin harmonics is automatically zero.

The axis file also specifies 4 integer variables which will be used in initializing the surface: ϵ, a number representing the elongation (ellipticity?) of the magnetic surface; a, the minor radius of the surface; N_{rotate}, the number of rotations of the normal and binormal of the magnetic surface, and `zeta_off`,

which is an integration constant for computing the torsion of the axis. I read these and pass them to the surface, because they are related to the initialization of the surface.

2.1 Frenet-Serret Equations for Axis

Given an axis, the Tangent, Normal, and Binormal are unique (except for cases where the derivative of the curve is zero, but I won't worry about those here). So I compute those in the Axis class.

The tangent vector of a curve parametrized by ζ is

$$\boldsymbol{T} = \frac{d\boldsymbol{r}}{ds} = \frac{d\boldsymbol{r}}{d\zeta} \Big/ \left| \frac{d\boldsymbol{r}}{d\zeta} \right| \tag{2}$$

The normal vector of a curve parametrized by ζ is

$$\boldsymbol{N} = \frac{1}{\left| \frac{d\boldsymbol{T}}{ds} \right|} \frac{d\boldsymbol{T}}{ds} \tag{3}$$

Now we need to find $\frac{d\boldsymbol{T}}{ds}$. Let's find it.

$$\frac{d\boldsymbol{T}}{ds} = \left(\frac{ds}{d\zeta} \right)^{-1} \frac{d}{d\zeta} \left(\frac{d\boldsymbol{r}}{d\zeta} \Big/ \left| \frac{d\boldsymbol{r}}{d\zeta} \right| \right)$$

$$\frac{d\boldsymbol{T}}{ds} = \frac{d^2\boldsymbol{r}}{d\zeta^2} \Big/ \left| \frac{d\boldsymbol{r}}{d\zeta} \right|^2 + \frac{d\boldsymbol{r}}{d\zeta} \left(\left| \frac{d\boldsymbol{r}}{d\zeta} \right| \right)^{-1} \frac{d}{d\zeta} \left(\left| \frac{d\boldsymbol{r}}{d\zeta} \right| \right)^{-1}$$

$$\frac{d\boldsymbol{T}}{ds} = \frac{d^2\boldsymbol{r}}{d\zeta^2} \Big/ \left| \frac{d\boldsymbol{r}}{d\zeta} \right|^2 - \boldsymbol{T} \left(\left| \frac{d\boldsymbol{r}}{d\zeta} \right| \right)^{-2} \frac{d}{d\zeta} \left| \frac{d\boldsymbol{r}}{d\zeta} \right|$$

$$\frac{d}{d\zeta} \left| \frac{d\boldsymbol{r}}{d\zeta} \right| = \frac{d}{d\zeta} \left(x_1^2 + y_1^2 + z_1^2 \right)^{1/2} = (x_1 x_2 + y_1 y_2 + z_1 z_2) / \left(x_1^2 + y_1^2 + z_1^2 \right)^{1/2} = \boldsymbol{T} \cdot \frac{d^2\boldsymbol{r}}{d\zeta^2}$$

Putting it together, we have

$$\frac{d\boldsymbol{T}}{ds} = \frac{d^2\boldsymbol{r}}{d\zeta^2} \Big/ \left| \frac{d\boldsymbol{r}}{d\zeta} \right|^2 - \boldsymbol{T} \left(\boldsymbol{T} \cdot \frac{d^2\boldsymbol{r}}{d\zeta^2} \right) \Big/ \left| \frac{d\boldsymbol{r}}{d\zeta} \right|^2 \tag{4}$$

and

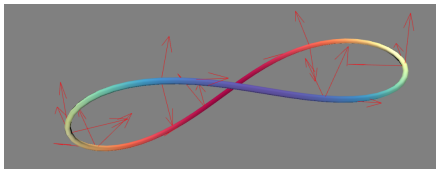
$$\boldsymbol{N} = \frac{1}{\left| \frac{d\boldsymbol{T}}{ds} \right|} \frac{d\boldsymbol{T}}{ds} \tag{5}$$

I can calculate the binormal vector using

$$\boldsymbol{B} = \boldsymbol{T} \times \boldsymbol{N} \tag{6}$$

2.2 Plots of Axis, Frenet Vectors

An example magnetic axis is shown below, with the Tangent, Normal, and Binormal vectors shown as well.



3 Initializing the Surface

3.1 Computing the frame around the axis

We've already calculated the normal and binormal vectors $\boldsymbol{N}$ and $\boldsymbol{B}$ of the axis. Now we rotate the frame of the axis according to

$$\begin{bmatrix} \boldsymbol{v}_1(\zeta) \\ \boldsymbol{v}_2(\zeta) \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \boldsymbol{N}(\zeta) \\ \boldsymbol{B}(\zeta) \end{bmatrix} \tag{7}$$

And we have $\frac{\partial\alpha}{\partial\zeta} = N_{rotate}/2 - (\tau(\zeta) - \langle\tau(\zeta)\rangle)$ where τ is the axis torsion and N_{rotate} is an integer. The axis torsion is given by

$$\tau(\zeta) = \frac{|(\boldsymbol{r}' \times \boldsymbol{r}'') \cdot \boldsymbol{r}'''|}{|\boldsymbol{r}' \times \boldsymbol{r}''|^2} \tag{8}$$

Starting with ζ_{off} as the integration constant at $\zeta = 0$, we can integrate over ζ to compute $\alpha(\zeta)$.

We can compute the curvature as

$$\kappa = \frac{|\boldsymbol{r}' \times \boldsymbol{r}''|}{|\boldsymbol{r}'|^3} \tag{9}$$

3.2 Computing the Surface Position

The position of the surface, for a given poloidal angle θ, toroidal angle ζ, radial coordinate s, minor radius a and ellipticity(?) ϵ is

$$\boldsymbol{r}(s,\theta,\zeta) = \boldsymbol{r}_a(\zeta) + sa\left[\epsilon^{1/2}\cos\theta\boldsymbol{v}_1(\zeta) + \epsilon^{-1/2}\sin\theta\boldsymbol{v}_2(\zeta)\right] \tag{10}$$

3.3 Computing the Normal Vector to the Surface

The normal vector to the surface $\boldsymbol{n}_s$ can be computed as

$$\boldsymbol{n}_s = \frac{d\boldsymbol{r}}{d\theta} \times \frac{d\boldsymbol{r}}{d\zeta} \Big/ \left|\frac{d\boldsymbol{r}}{d\theta} \times \frac{d\boldsymbol{r}}{d\zeta}\right| \tag{11}$$

We can compute $\frac{d\boldsymbol{r}}{d\theta}$ by taking derivatives of equation 10 to get

$$\frac{d\boldsymbol{r}}{d\theta} = sa\big[-\epsilon^{1/2}\sin\theta\boldsymbol{v}_1 + \epsilon^{-1/2}\cos\theta\boldsymbol{v}_2\big] \tag{12}$$

and

$$\frac{d\boldsymbol{r}}{d\zeta} = \frac{d\boldsymbol{r}_a}{d\zeta} + sa\big[\epsilon^{1/2}\cos\theta\frac{d\boldsymbol{v}_1}{d\zeta} + \epsilon^{-1/2}\sin\theta\frac{d\boldsymbol{v}_2}{d\zeta}\big] \tag{13}$$

Computing $\frac{d\boldsymbol{r}_a}{d\zeta}$ is easy, from equation 1. Now we just need to compute $\frac{d\boldsymbol{v}}{d\zeta}$. From equation 7, we have

$$\begin{bmatrix}\frac{d\boldsymbol{v}_1}{d\zeta}\\ \frac{d\boldsymbol{v}_2}{d\zeta}\end{bmatrix} = \begin{bmatrix}\cos\alpha & \sin\alpha\\ -\sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}\frac{d\boldsymbol{N}}{d\zeta}\\ \frac{d\boldsymbol{B}}{d\zeta}\end{bmatrix} + \begin{bmatrix}-\sin\alpha & \cos\alpha\\ -\cos\alpha & -\sin\alpha\end{bmatrix}\begin{bmatrix}\boldsymbol{N}(\zeta)\\ \boldsymbol{B}(\zeta)\end{bmatrix}\frac{d\alpha}{d\zeta} \tag{14}$$

where $\frac{d\alpha}{d\zeta} = N_{rotate}/2 - \tau(\zeta)$ as before. Now we need to compute $\frac{d\boldsymbol{N}}{d\zeta}$ and $\frac{d\boldsymbol{B}}{d\zeta}$. We can get these from the Frenet-Serret formulas, which are

$$\frac{d\boldsymbol{B}}{ds} = -\tau\boldsymbol{N} = \frac{d\boldsymbol{B}}{d\zeta}\Big/\frac{ds}{d\zeta}$$

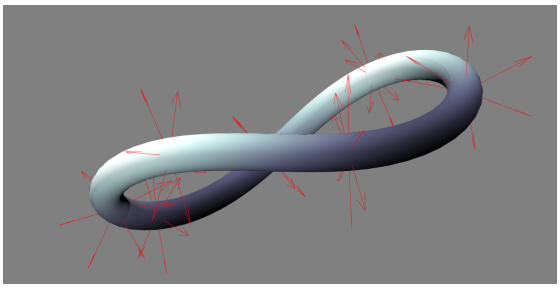
$$\frac{d\boldsymbol{N}}{d\zeta}\Big/\frac{ds}{d\zeta} = -\kappa\boldsymbol{T} + \tau\boldsymbol{B}$$

and $\frac{ds}{d\zeta} = \big|\frac{d\boldsymbol{r}}{d\zeta}\big|$.

It should be mentioned that the normal vector to the surface could be computed using automatic differentiation, but here we compute the derivatives analytically.

3.4 Plots of Surface, Normal to Surface

An example surface is shown below, with some normal vectors to the surface plotted.



4 Initializing the Coils

If we have a HDF5 file which has the coil stored data and metadata, we don't need to initialize the coils from scratch. If we don't have a HDF5 file, then the coils are initialized from scratch in a simple way: they are initialized with the same shape as the surface, except with a larger initial radius than the surface. Mathematically, this is the same as equation 10 except with a larger s, and with ζ evaluated at `NC` points, spaced evenly between 0 and 2π.

For this initialization to work properly, the number of points in zeta on the surface needs to be a multiple of the number of coils. For example, the default settings might have 64 points on the surface and 8 coils.

The central filament of each coil is described with a Fourier series in θ. So the position of the ith coil is described by

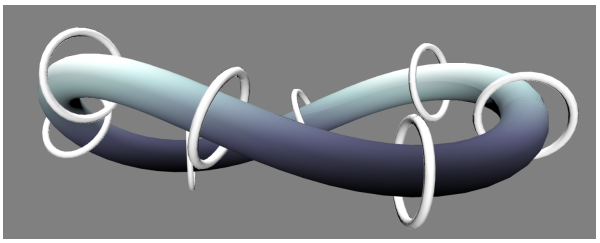
$$\begin{aligned} x^i(\theta) &= \sum_{m=0}^{N_F-1} X^i_{cm}\cos(m\theta) + X^i_{sm}\sin(m\theta) \\ y^i(\theta) &= \sum_{m=0}^{N_F-1} Y^i_{cm}\cos(m\theta) + Y^i_{sm}\sin(m\theta) \\ z^i(\theta) &= \sum_{m=0}^{N_F-1} Z^i_{cm}\cos(m\theta) + Z^i_{sm}\sin(m\theta) \end{aligned} \tag{15}$$

If we need to perform the discrete Fourier transform from real space to the fourier coefficients, it is done as follows:

$$\begin{aligned} X^i_{c0} &= \frac{1}{N_s}\sum_{s=1}^{N_s} x^i(\theta_s)\cos\left(m\theta_s\right) \\ X^i_{cm} &= \frac{2}{N_s}\sum_{s=1}^{N_s} {}^i(\theta_s)\cos\left(m\theta_s\right) \\ X^i_{sm} &= \frac{2}{N_s}\sum_{i=1}^{N_s} x^i(\theta_s)\sin\left(m\theta_s\right) \end{aligned} \tag{16}$$

and so on for y and z. Here, $\theta_s = s/(2\pi N_s)$

An example initial central coil configuration is shown below. A plot of the finite-build coil set is shown below.



4.1 Setting up Finite-Build Coils

The mathematics describing the Frenet Frame of each coil is identical to the mathematics describing the Frenet Frame of the Axis, except each coil varies in θ instead of ζ. For each coil, we compute the Frenet vectors $\boldsymbol{T}^i$, $\boldsymbol{N}^i$, and $\boldsymbol{B}^i$.

The shape of each finite-build coil is assumed to be rectangular. Each finite finite-build is initially aligned at each point with the normal and binormal vectors, but is allowed to rotate relative to these vectors according to an angle $\alpha(\theta)$. Because of the rectangular shape, α needs to be closed in π.

$$\begin{bmatrix} \boldsymbol{v}_1^i(\theta) \\ \boldsymbol{v}_2^i(\theta) \end{bmatrix} = \begin{bmatrix} \cos\alpha^i & \sin\alpha^i \\ -\sin\alpha^i & \cos\alpha^i \end{bmatrix} \begin{bmatrix} \boldsymbol{N}^i(\theta) \\ \boldsymbol{B}^i(\theta) \end{bmatrix} \tag{17}$$

Since α^i needs to be closed in π, then we can parameterize $\alpha(\theta)$ with a Fourier series. This is unlike the surface $\alpha(\zeta)$ where α is found using the integrated torsion.

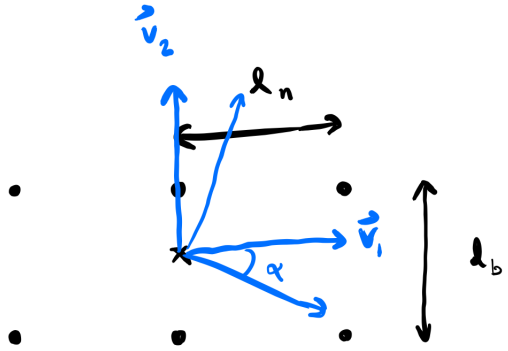
$$\alpha^i(\theta) = \frac{N_R\theta}{2} + \sum_{m=0}^{N_{FR}-1} A_{cm}^i \cos(m\theta) + A_{sm}^i \sin(m\theta) \tag{18}$$

I would imagine that, for ease of engineering, it almost always makes sense to set N_R to zero.

Once we have $\boldsymbol{v}_1^i$ and $\boldsymbol{v}_2^i$, we can compute the position of the $N_n \times N_b$ filaments for each coil, which approximates the finite-build coil. We do this using the following formula for the nth and bth filaments, where n runs from 0 to $N_n - 1$ and b runs from 0 to $N_b - 1$.

$$\boldsymbol{r}_{n,b}^i(\theta) = \boldsymbol{r}_{central}^i + \Big[n - \frac{N_1 - 1}{2}\Big] l_1 \boldsymbol{v}_1^i(\theta) + \Big[b - \frac{N_2 - 1}{2}\Big] l_2 \boldsymbol{v}^i(\theta) \tag{19}$$

Here, l_n is the spacing between the filaments in the $\boldsymbol{v}_1$ direction, and l_b is the spacing between the filaments in the $\boldsymbol{v}_2$ direction.



5 Loss Functions

5.1 Default Loss Function

In the default loss function, the currents in the coils are held fixed and the parameters of the coils are varied. The loss is a weighted sum of the integral of the normal magnetic field over the outer surface of the vacuum magnetic surface and the total length of the coils.

$$\begin{aligned} L &= \sum_{i=1}^{N_c} \int \frac{ds_i}{d\theta} d\theta \\ B^2 &= \int_S (\boldsymbol{B} \cdot \hat{n})^2 dA \\ \text{Loss} &= B^2 + \lambda_L L \end{aligned} \tag{20}$$

where λ_L controls the strength of regulation, encouraging the coils to be shorter in total length.

The normal vector of the surface has fortunately already been calculated. We can calculate the magnetic field at any point on the surface using the Biot-Savart law,

$$\boldsymbol{B}(\boldsymbol{r}) = \sum_{i=1}^{N_c} \sum_{n=1}^{N_1} \sum_{b=1}^{N_2} I_{n,b}^i \mu_0 \oint \frac{d\boldsymbol{l}_{n,b}^i \times (\boldsymbol{r} - \boldsymbol{r}_{n,b}^i)}{|\boldsymbol{r} - \boldsymbol{r}_{n,b}^i|^3} \tag{21}$$

6 Optimizers

One optimizer is simple gradient descent with a fixed learning rate. A second optimizer is `ODEFlow`, which applies a fourth-order Runge-Kutta integrator to the integration of the ODE. The equations for this Runge-Kutta integrator are shown below.

$$\begin{aligned}\boldsymbol{\theta}_1 =& \boldsymbol{\theta}_0 - \frac{\eta}{2}\boldsymbol{\nabla} L(\boldsymbol{\theta}_0) \\ \boldsymbol{\theta}_2 =& \boldsymbol{\theta}_1 - \frac{\eta}{2}\boldsymbol{\nabla} L(\boldsymbol{\theta}_1) \\ \boldsymbol{\theta}_3 =& \boldsymbol{\theta}_2 - \eta\boldsymbol{\nabla} L(\boldsymbol{\theta}_2)\end{aligned} \tag{22}$$

$$\boldsymbol{\theta}_f = \boldsymbol{\theta}_0 - \frac{\eta}{6}(\boldsymbol{\nabla} L(\boldsymbol{\theta}_0) + 2\boldsymbol{\nabla} L(\boldsymbol{\theta}_1) + 2\boldsymbol{\nabla} L(\boldsymbol{\theta}_2) + \boldsymbol{\nabla} L(\boldsymbol{\theta}_3)) \tag{23}$$

A third optimizer is a damped Newton's method, using the Hessian matrix which JAX can calculate automatically.