**0D model for stellarator coil dimensioning**

A branch for the simsopt code is under development where the Lorentz forces on the coils are determined. This takes into account the self-inflicted forces are the forces originated from the interaction with the rest of the codes. In this code, the coils are treated as 1D filaments, the goal of this module is to obtain the section needed to withstand the Lorentz forces.

# Reading simsopt data

simsopt generates the coil data in txt files with no header. The first three columns are the cartesian coordinates, while the last one is the current. A coil is ended when the current is equal to 0. This data has to be converted into the coil and curve objects that the simsopt uses as input. The curve object has the geometry of the 1D coil but in Fourier terms. The coil is similar but also includes the current.

The xyz2fourier.py file includes routines to convert the input file into a simsopt friendly format. It gets the Fourier decomposition of the curve using the routines found in fourier\_decomp.py and inputs that decomposition into the simsopt function curveXYZfourier. It is important to note that the documentation regarding this function is wrong. It states that the first coefficient you should be adding is the 0 order cosine one when it should be the 1st order sine coefficient (0 order sine coefficient is 0). This is already taken into account in these functions.

Figure 1 shows the comparison of the curve obtained from the simsopt output and from the Fourier decomposition. This proves the validity of the presented scripts.

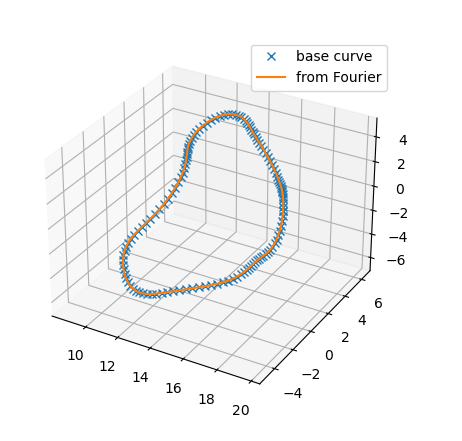


Figure – Curves from the simsopt cartesian output (in blue) and after transforming it into its Fourier decomposition (in orange).

# Lorentz force calculation

The Lorentz force calculation is not part of the main simsopt branch by the time of this writing (12/19/2024). It belongs to the [planar\_coil\_arrays](https://github.com/hiddenSymmetries/simsopt/tree/planar_coil_arrays) branch, this is an open branch, therefore, changes and updates that may render parts of this module outdated may take place. This module is tested with commit ec33cdd8a2df716f696d8688cda6153b79ec235f (from 11/12/2024).

The code calculates the self-induced forces and the mutual forces from the rest of the coils. This is calculated for each of the elements that compose the entire curve. From them, it calculates the torque assuming that the “lever” of the torque is the distance from the center of the element to the center of the coil. I am not so sure that this is the correct way of calculating the torque, more on this in the discussion section.

# Section calculation

The first part of the code estimates the area of conductor needed to provide the target current. Currently, the code is prepared to work with superconductors. This means that there has to be an extra copper conductor for the electricity to flow in case of quenching. The needed cross-section of copper and superconducting material is calculated from the coil current, the critical current density of the material and a safety factor (*sf*) set by the user. The sum of the copper and superconductor cross-sections is referred to as conductor (*cond*) in the code. The cooling necessities are not estimated in this code. Therefore, the coolant cross-section is set by the user, a default value of 30% is used. The addition of the conductor and the coolant cross-section is named as *cable*.

The steel cross-section is obtained by progressively increasing the steel until the von misses stress is lower than the smeared yield limit. The smeared yield limit is calculated as the weighted addition of the yield limit of the participant materials:

Note that all materials have a related safety factor. Therefore, the calculated coil yield limit will also have a certain related safety factor.

Consider superconductors are brittle enough to not have a significant load-bearing capacity. Therefore, their yield limit is set at 0.

# Materials cost

The method *get\_cost* calculates the material cost of the coil. Note that this does not account for manufacturing-engineering costs, just the material.

# Comments, discussion and to do list

* Hard-set values are referenced in the source file
* Superconductor critical current density has been arbitrarily evaluated at 7T and 4.75K.
* Three REBCO models are presented with different costs depending on the source.
* Default steel is SS316. This is a non-magnetic austenitic steel that is widely used in fusion, at the cost of a low yield limit. The assumption of better and more resilient materials for a future reactor could be made.
* *coil\_assembly.py* is the class-based version of *coil\_cross\_section.py*. The former is the intended version with the latter being kept in case it is useful for anyone.
* The stresses obtained from the Lorentz forces are but an approximation. Getting accurate stresses would require knowing the location of the coils supports to, then, calculate the stiffness matrix. The code here presented is intended as an easy check on the feasibility of a design, not as a FEA substitute.