Introduction to DESC and Stellarators

Yigit Gunsur Elmacioglu. Kaya Unalmis

May 13, 2025

Table of Contents

1 Introduction **1**

2 Basis Functions **1**
- 2.1 Fourier Series . . . 1
- 2.2 Zernike Polynomials . . . 2
  - 2.2.1 Zernike Basis Indexing in DESC . . . 3
  - 2.2.2 Symmetry Options . . . 5
- 2.3 Fourier-Zernike Basis . . . 7
- 2.4 Stellarator Symmetry . . . 8

3 Coordinates **10**
- 3.1 Covariant and Contravariant Vectors . . . 10
- 3.2 Vector Operations . . . 13
  - 3.2.1 Dot Product . . . 13
  - 3.2.2 Cross Product . . . 14
- 3.3 Del Operator in Curvilinear Coordinates . . . 14
  - 3.3.1 Gradient . . . 14
  - 3.3.2 Divergence . . . 14
  - 3.3.3 Curl . . . 14
- 3.4 DESC Coordinates . . . 15

4 Fields **17**
- 4.1 Short Discussion about Rotational Transform ι . . . 17
- 4.2 Ideal MHD Equations . . . 18
  - 4.2.1 Magnetic Field . . . 18
  - 4.2.2 Electric Current Density . . . 22
  - 4.2.3 $\mathbf{J} \times \mathbf{B}$ Force . . . 23
- 4.3 Math to compute the rotational transform in DESC . . . 24
  - 4.3.1 Terminology . . . 24
  - 4.3.2 Derivation . . . 24
  - 4.3.3 Other arrangements . . . 25
  - 4.3.4 Axis limit of rotational transform . . . 26
  - 4.3.5 Rotational transform with toroidal stream function . . . 29
  - 4.3.6 Axis limit of rotational transform derivatives . . . 30

5 Optimization **31**
- 5.1 Overview of *lsq-exact* Optimizer . . . 31
- 5.2 Proximal Projection and Force Balance Constraint . . . 34
  - 5.2.1 Creation of the wrapper . . . 36
  - 5.2.2 Build Method . . . 36
  - 5.2.3 _set_eq_state_vector() . . . 36
  - 5.2.4 _feasible_tangents . . . 37
  - 5.2.5 Derivatives . . . 38
  - 5.2.6 _update_equilibrium() . . . 38
  - 5.2.7 JVP . . . 39

6 Profiling DESC **40**
- 6.1 Profiling a simple script with NSight . . . 40
- 6.2 Memory Profiling . . . 43

7 Multi-Device Applications **46**
- 7.1 Installing MPI4JAX . . . 46

8 DON'T FORGET TO INCLUDE **47**

A Relation between Zernike Polynomials and Jacobi Polynomials **47**

B Covariant J Alternative 49

C Relation between Zernike Polynomials and Jacobi Polynomials 50

List of Tables

List of Figures

1 "cos" Symmetry Modes . . . 5
2 "sin" Symmetry Modes . . . 5
3 **Sketch a better figure for this** . . . 10
4 General curvilinear coordinate system [1] . . . 11
5 Surfaces of constant parameters (a) ρ, (b) θ and (c) ϕ [2] . . . 13
6 Rational Transform Iota and Field lines closing on themselves [3] . . . 17

1 Introduction

This document is for getting started with the basic maths used in Stellarators, more specifically the ones used in 3D MHD equilibrium code DESC. It will serve as a cheat sheet for the future. In preparation of this document, the book "An Introduction to Stellarators" by Imbert Gerard et al. is heavily used [3].

2 Basis Functions

2.1 Fourier Series

Any periodic smooth function can be written in Fourier series form. The basis functions for the Fourier series are sines and cosines which form an orthogonal basis that means,

$$\langle sin(m\theta)|sin(n\theta)\rangle = \int_0^T sin(m\theta)sin(n\theta)d\theta \tag{1}$$

$$\langle sin(m\theta)|sin(n\theta)\rangle = \begin{cases} 0 & \text{for } m \neq n \\ T/2 & \text{for } m = n \end{cases} \tag{2}$$

$$\langle cos(m\theta)|cos(n\theta)\rangle = \int_0^T cos(m\theta)cos(n\theta)\,d\theta \tag{3}$$

$$\langle cos(m\theta)|cos(n\theta)\rangle = \begin{cases} 0 & \text{for } m \neq n \\ T/2 & \text{for } m = n \end{cases} \tag{4}$$

$$\langle cos(m\theta)|sin(n\theta)\rangle = \int_0^T cos(m\theta)sin(n\theta)\,d\theta \tag{5}$$

$$\langle cos(m\theta)|sin(n\theta)\rangle = 0 \qquad \text{for all } m, n \tag{6}$$

The last equality results from one being odd and the other being even function, which makes the product odd, and in the given integration boundaries, this will result in 0. Thus, we can write a function in terms of sines and cosines,

$$f(x) = \sum_{n=0}^{\infty} a_n cos(nx) + \sum_{n=0}^{\infty} b_n sin(nx) \tag{7}$$

Since for n =0, sine is always 0, we can write Fourier series in general as,

$$f(x) = a_0 + \sum_{n=1}^{\infty} a_n cos(nx) + \sum_{n=1}^{\infty} b_n sin(nx) \tag{8}$$

Let's define the n^{th} Fourier term as,

$$\mathcal{F}^n(x) = a_n cos(nx) + b_n sin(nx) \tag{9}$$

Further, divide the sine and cosine terms apart by defining,

$$\mathcal{F}^n(x) = cos(nx) \qquad \text{for } n \geq 0 \tag{10}$$

$$\mathcal{F}^n(x) = sin(|n|x) \qquad \text{for } n < 0 \tag{11}$$

2.2 Zernike Polynomials

The Fourier series is very convenient to define periodic functions. But for functions that are defined on a disc, we will need a different way. The Zernike polynomials are a sequence of polynomials that are orthogonal on the unit disc. They play important roles in various optics branches such as beam optics and imaging. We will make use of the properties of Zernike polynomials to define toroidal cross-section. For the Fourier series, we used a single parameter n, for the Zernike Polynomials, we will use 2 parameters l and m. Again, by dividing sine and cosine terms, we can show odd and even Zernike Polynomials as,

$$\mathcal{Z}_l^m(\rho,\theta) = \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \text{ and } 0 \leq \rho \leq 1 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \text{ and } 0 \leq \rho \leq 1 \end{cases} \tag{12}$$

where $\mathcal{R}_l^m$ is the radial part of the Zernike Polynomials and it is defined as,

$$\mathcal{R}_l^m(\rho) = \sum_{s=0}^{(l-m)/2} \frac{(-1)^s(l-s)!}{s!\left(\frac{l+m}{2}-s\right)!\left(\frac{l-m}{2}+s\right)!}\rho^{l-2s} \qquad \text{for } m \geq 0 \text{ and } 0 \leq \rho \leq 1 \tag{13}$$

First few of the radial part Zernike Polynomials are,

$$\begin{aligned} R_0^0(\rho) &= 1 \\ R_1^1(\rho) &= \rho \\ R_2^0(\rho) &= 2\rho^2 - 1 \\ R_2^2(\rho) &= \rho^2 \\ R_3^1(\rho) &= 3\rho^3 - 2\rho \\ R_3^3(\rho) &= \rho^3 \\ R_4^0(\rho) &= 6\rho^4 - 6\rho^2 + 1 \end{aligned}$$

So, we can write, for example, even Zernike Polynomials in full form as,

$$\mathcal{Z}_l^m(\rho,\theta) = cos(m\theta)\sum_{s=0}^{(l-m)/2} \frac{(-1)^s(l-s)!}{s!\left(\frac{l+m}{2}-s\right)!\left(\frac{l-m}{2}-s\right)!}\rho^{l-2s} \qquad \text{for } m \geq 0 \text{ and } 0 \leq \rho \leq 1 \tag{14}$$

Notice that $(l-m)/2$ must be a positive integer. Therefore, possible values for m are $m = \{-l, -l+2, ..., l-2, l\}$. For odd l values, for example $l = 3$, $m = \{-3, -1, 1, 3\}$, and for even l values, for example $l = 4$, $m = \{-4, -2, 0, 2, 4\}$. Moreover, for $\rho = 0$, these polynomials don't depend on θ (as it should be). For m=0, this is trivial [cos(0.θ)=1]. If $|m| > 0$, $l - 2s = l - 2\frac{l-|m|}{2} = |m| > 0$, will always satisfy that there is ρ to some positive power, which in turn when evaluated at $\rho = 0$, gives 0. Therefore, we can say that,

$$\mathcal{Z}_l^m(\rho = 0,\theta) = \mathcal{R}_l^0(0) = \begin{cases} 0 & \text{if m is 0, } l \text{ must be even number, so odd } l \text{ values are 0} \\ (-1)^{l/2} & \text{products of 4 are 1, others -1} \end{cases} \tag{15}$$

The radial part Zernike Polynomials can also be calculated as,

$$\mathcal{R}_l^m(\rho) = (-1)^{(l-m)/2}\rho^m P_{(l-m)/2}^{m,0}(1-2\rho^2) \tag{16}$$

where $P_n^{\alpha,\beta}(\rho)$ is a Jacobi polynomial. This allows numerical simulations to use stable recurrence relations for the Jacobi polynomials. Since Jacobi polynomials have their recursion relation, we can avoid many calculations. Here is the recursion relations for Jacobi polynomials,

$$2n(c-n)(c-2)P_n^{\alpha,\beta}(\rho) = (c-1)[c(c-2)\rho + (a-b)(c-2n)]P_{n-1}^{\alpha,\beta}(\rho) - 2(a-1)(b-1)cP_{n-2}^{\alpha,\beta}(\rho) \tag{17}$$

where,

$$c = 2n+\alpha+\beta, \qquad a = n+\alpha, \qquad b = n+\beta \tag{18}$$

For the calculation of derivatives, we will be using the following relation,

$$\frac{d^k}{dx^k}P_n^{(\alpha,\beta)}(x) = \frac{\Gamma(\alpha+\beta+n+1+k)}{2^k\Gamma(\alpha+\beta+n+1)}P_{n-k}^{(\alpha+k,\beta+k)}(x) \tag{19}$$

Just as in the Fourier series, we can write a function in terms of Zernike polynomials,

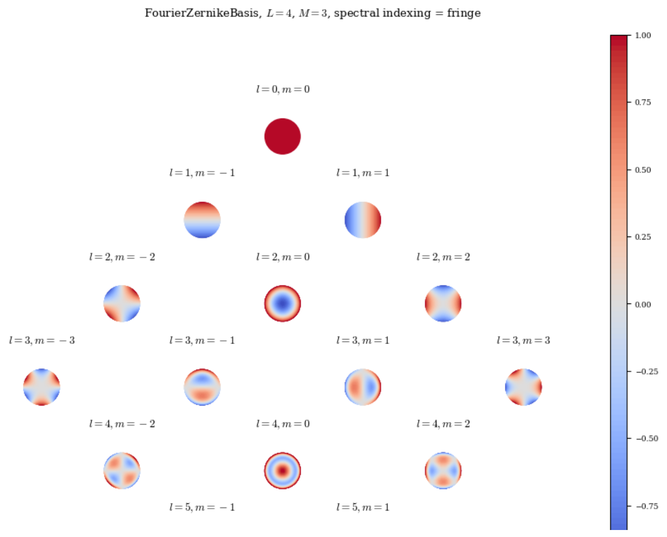
$$f(\rho,\theta) = \sum_{l=0}^{L}\begin{cases} \sum_{|m|=0}^{l} c_l^m \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases} & \text{for } l \text{ even} \\ \sum_{|m|=1}^{l} c_l^m \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases} & \text{for } l \text{ odd} \end{cases} \tag{20}$$

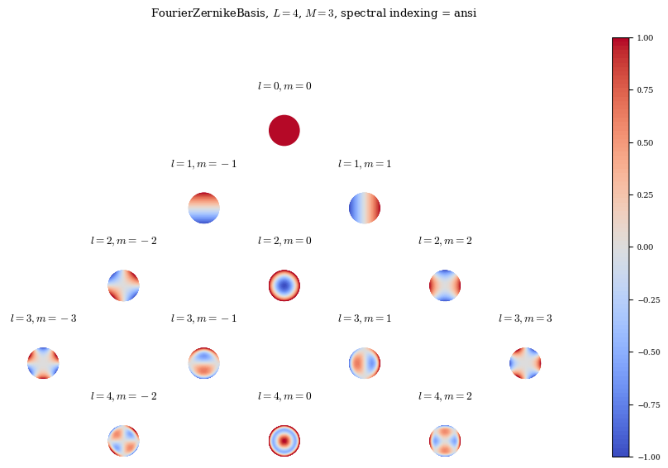
In reality, the upper limit L should go to ∞ but for numerical simulations, we will choose a resolution to express function $f(\rho,\theta)$. For $L = 3$, let's expand the function using Zernike polynomials,

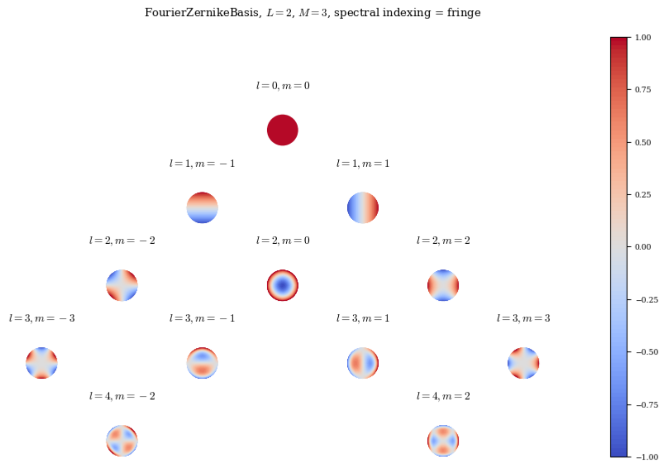
$$\begin{aligned} f(\rho,\theta) = \;& c_0^0\ \mathcal{R}_0^0(\rho) + & (21)\\ & c_1^{-1}\ \mathcal{R}_1^1(\rho)\ \ sin(\theta) + c_1^1\ \mathcal{R}_1^1(\rho)\ cos(\theta) + & (22)\\ & c_2^{-2}\ \mathcal{R}_2^2(\rho)\ sin(2\theta) + c_2^0\ \mathcal{R}_2^0(\rho) + c_2^2\ \mathcal{R}_2^2(\rho)\ cos(2\theta) + & (23)\\ & c_3^{-3}\ \mathcal{R}_3^3(\rho)\ sin(3\theta) + c_3^{-1}\ \mathcal{R}_3^1(\rho)\ sin(\theta) + c_3^1\ \mathcal{R}_3^1(\rho)\ cos(\theta) + c_3^3\ \mathcal{R}_3^3(\rho)\ cos(3\theta) & (24) \end{aligned}$$

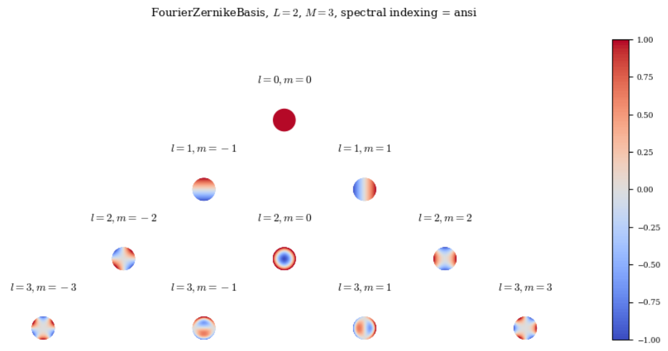
2.2.1 Zernike Basis Indexing in DESC

The previous discussion of possible l and m values was for the general theory. In general, you can select maximum l and m values different as long as max(l)>max(m). Therefore in DESC, when you specify L, it is not the maximum value of l but rather it is the maximum difference between l and m. There are 2 options to construct the pyramid, namely *ansi* and *fringe*. For the *ansi* option, the maximum l value cannot exceed max(L, M) and there is no condition on $l + m$. For *fringe* option, maximum l value can go up to max(L, M) and max(l + m)=max$(L, 2M)$, also the pyramid shouldn't have weird branches, such that a l line can have an element on the edges or if there is an inverse pyramid on top (see the examples, it's hard to explain). Here are some examples,


FourierZernikeBasis, $L=4$, $M=3$, spectral indexing = fringe
$l=0, m=0$
$l=1, m=-1$
$l=1, m=1$
$l=2, m=-2$
$l=2, m=0$
$l=2, m=2$
$l=3, m=-3$
$l=3, m=-1$
$l=3, m=1$
$l=3, m=3$
$l=4, m=-2$
$l=4, m=0$
$l=4, m=2$
$l=5, m=-1$
$l=5, m=1$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


FourierZernikeBasis, $L=4$, $M=3$, spectral indexing = ansi
$l=0, m=0$
$l=1, m=-1$
$l=1, m=1$
$l=2, m=-2$
$l=2, m=0$
$l=2, m=2$
$l=3, m=-3$
$l=3, m=-1$
$l=3, m=1$
$l=3, m=3$
$l=4, m=-2$
$l=4, m=0$
$l=4, m=2$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


FourierZernikeBasis, $L=2$, $M=3$, spectral indexing = fringe
$l=0, m=0$
$l=1, m=-1$
$l=1, m=1$
$l=2, m=-2$
$l=2, m=0$
$l=2, m=2$
$l=3, m=-3$
$l=3, m=-1$
$l=3, m=1$
$l=3, m=3$
$l=4, m=-2$
$l=4, m=2$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


FourierZernikeBasis, $L=2$, $M=3$, spectral indexing = ansi
$l=0, m=0$
$l=1, m=-1$
$l=1, m=1$
$l=2, m=-2$
$l=2, m=0$
$l=2, m=2$
$l=3, m=-3$
$l=3, m=-1$
$l=3, m=1$
$l=3, m=3$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

You can test different cases with the following code block,

```
from desc.basis import FourierZernikeBasis
from desc.plotting import plot_basis
basis = FourierZernikeBasis(L=4, M=3, N=0, spectral_indexing="ansi")
plot_basis(basis);
```

2.2.2 Symmetry Options

"cos" symmetry doesn't have any sin terms, so only $m >= 0$ terms are considered. Similarly, "sin" symmetry doesn't have any cos terms, so only $m < 0$ terms are considered.

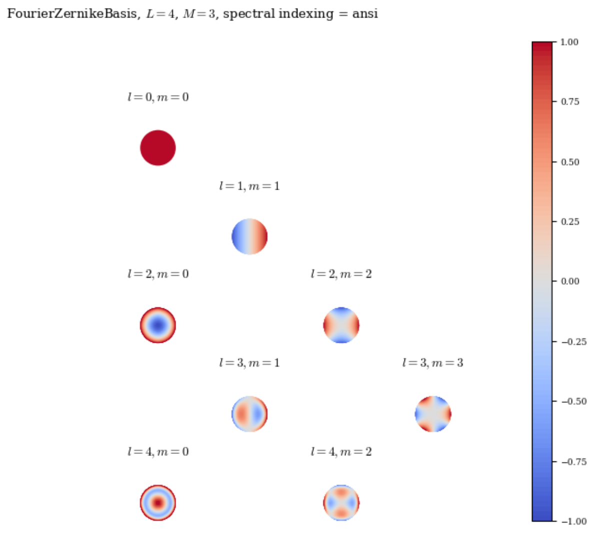


Figure 1: "cos" Symmetry Modes

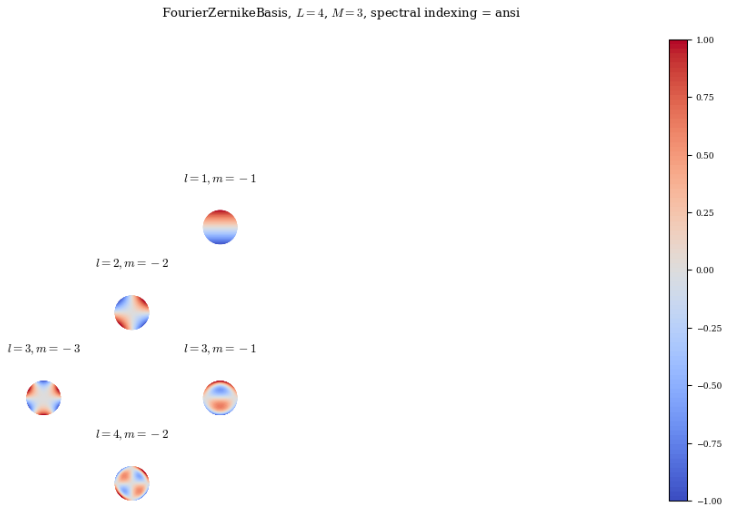


Figure 2: "sin" Symmetry Modes

We can find the total number of Zernike modes for **sym="cos"**, **spectral_indexing="ansi"** and $M = L$,

$$\textit{Total Number of Zernike Modes} = \sum_{l=even}^{M} \left(\frac{l}{2}+1\right) + \sum_{l=odd}^{M} \frac{l+1}{2} \tag{25}$$

So,

$$\begin{aligned}\textit{Total Number of Zernike Modes} &= \begin{cases} \sum_{k=0}^{M/2} \left(\frac{2k}{2}+1\right) + \sum_{k=0}^{M/2-1} \frac{(2k+1)+1}{2} & \text{for } M \text{ even} \\ \sum_{k=0}^{(M-1)/2} \left(\frac{2k}{2}+1\right) + \sum_{k=0}^{(M-1)/2} \frac{(2k+1)+1}{2} & \text{for } M \text{ odd} \end{cases} \\ &= \begin{cases} \left(\frac{M}{2}+1\right) + \frac{(\frac{M}{2}+1)\frac{M}{2}}{2} + \frac{M}{2} + \frac{(\frac{M}{2}-1)\frac{M}{2}}{2} & \text{for } M \text{ even} \\ 2\left(\frac{M-1}{2}+1+\frac{\frac{M-1}{2}(\frac{M-1}{2}+1)}{2}\right) & \text{for } M \text{ odd} \end{cases}\end{aligned} \tag{26}$$

$$\textit{Total Number of Zernike Modes} \quad = \quad \begin{cases} \dfrac{M^2+2M+4}{4} & \text{for } M \text{ even} \\ \dfrac{M^2+4M+3}{4} & \text{for } M \text{ odd} \end{cases} \tag{27}$$

Similarly, we can find the total number of Zernike modes for **sym="sin"**, **spectral_indexing="ansi"** and $M = L$,

$$\textit{Total \# of Zernike Modes} \quad = \quad \sum_{l=even}^{M} \frac{l}{2} + \sum_{l=odd}^{M} \left(\frac{l+1}{2} - 1 \right) \tag{28}$$

$$\textit{Total \# of Zernike Modes} \quad = \quad \begin{cases} \dfrac{M^2+3}{4} & \text{for } M \text{ even} \\ \dfrac{M^2+2M}{4} & \text{for } M \text{ odd} \end{cases} \tag{29}$$

If we add the number of modes for sin and cos, we can get the total number of modes for **spectral_indexing="ansi"** and $M = L$,

$$\textit{Total \# of Zernike Modes} = \begin{cases} \dfrac{2M^2+2M+7}{4} & \text{for } M \text{ even} \\ \dfrac{2M^2+6M+3}{4} & \text{for } M \text{ odd} \end{cases} \tag{30}$$

2.3 Fourier-Zernike Basis

We defined Fourier series for periodic functions and Zernike Polynomials for functions defined on a disc. Now, consider a stellarator or a tokamak. We can express any function inside those geometries by Fourier series in the toroidal direction (ζ) and by Zernike Polynomials in a toroidal cross-section ($\zeta =$ constant).

$$f(\rho,\theta,\zeta) = \sum_l \sum_m \sum_n c_{lmn} \mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta) \tag{31}$$

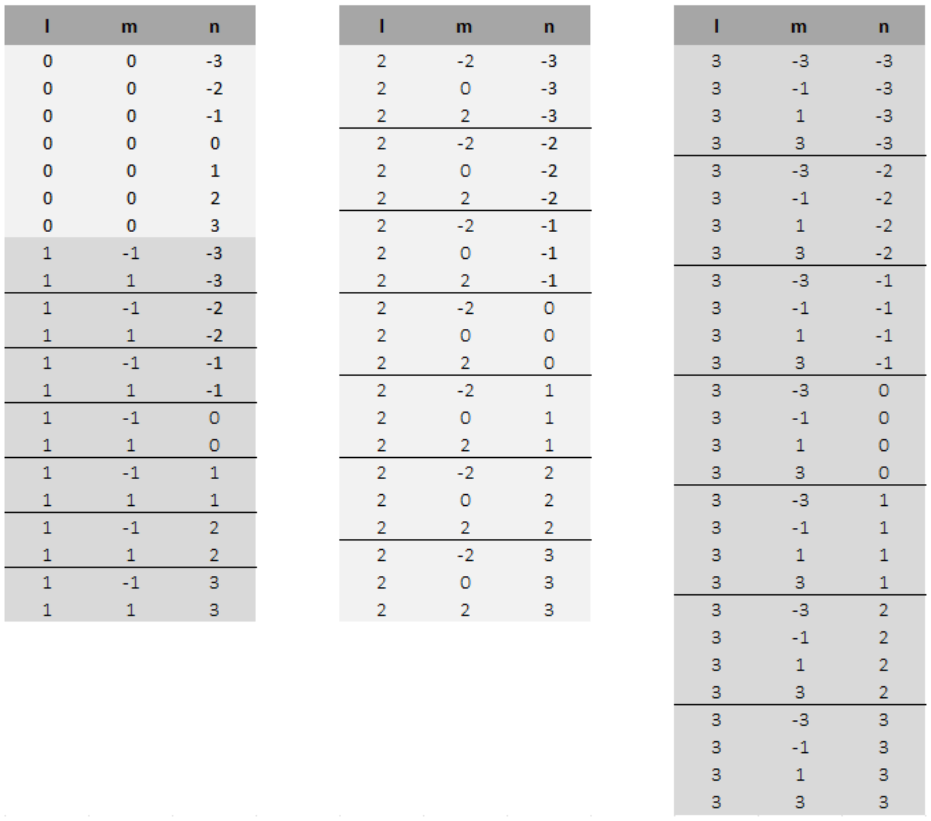
With the previous definitions from 10, 11 and 12, we can write a function depending on ρ, θ and ζ as,

$$f(\rho,\theta,\zeta) = \sum_{n=0}^{N} c_{lmn} cos(n\zeta) \sum_{l=0}^{L} \begin{cases} \sum\limits_{|m|=0}^{l} \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases} & \text{for } l \text{ even} \\ \sum\limits_{|m|=1}^{l} \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases} & \text{for } l \text{ odd} \end{cases} \tag{32}$$

$$+\sum_{n=1}^{N} c_{lmn} sin(n\zeta) \sum_{l=0}^{L} \begin{cases} \sum\limits_{|m|=0}^{l} \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases} & \text{for } l \text{ even} \\ \sum\limits_{|m|=1}^{l} \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases} & \text{for } l \text{ odd} \end{cases} \tag{33}$$

This starts to get a bit messy. Let's represent possible modes (l, m and n combinations) in terms of an array where each row is a mode. For example, take l, we can call it radial resolution from now on, $L = 3$, and take n, we can call it the toroidal resolution, $N = 3$. Corresponding basis function for each mode l, m and n can be written as,

$$\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta) = \begin{cases} \mathcal{R}_l^m(\rho)cos(m\theta)cos(n\zeta) & \text{for } m \geq 0 \text{ and } n \geq 0 \\ \mathcal{R}_l^m(\rho)cos(m\theta)sin(n\zeta) & \text{for } m \geq 0 \text{ and } n < 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta)cos(n\zeta) & \text{for } m < 0 \text{ and } n \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta)sin(n\zeta) & \text{for } m < 0 \text{ and } n < 0 \end{cases} \tag{34}$$



l	m	n
0	0	-3
0	0	-2
0	0	-1
0	0	0
0	0	1
0	0	2
0	0	3
1	-1	-3
1	1	-3
1	-1	-2
1	1	-2
1	-1	-1
1	1	-1
1	-1	0
1	1	0
1	-1	1
1	1	1
1	-1	2
1	1	2
1	-1	3
1	1	3

l	m	n
2	-2	-3
2	0	-3
2	2	-3
2	-2	-2
2	0	-2
2	2	-2
2	-2	-1
2	0	-1
2	2	-1
2	-2	0
2	0	0
2	2	0
2	-2	1
2	0	1
2	2	1
2	-2	2
2	0	2
2	2	2
2	-2	3
2	0	3
2	2	3

l	m	n
3	-3	-3
3	-1	-3
3	1	-3
3	3	-3
3	-3	-2
3	-1	-2
3	1	-2
3	3	-2
3	-3	-1
3	-1	-1
3	1	-1
3	3	-1
3	-3	0
3	-1	0
3	1	0
3	3	0
3	-3	1
3	-1	1
3	1	1
3	3	1
3	-3	2
3	-1	2
3	1	2
3	3	2
3	-3	3
3	-1	3
3	1	3
3	3	3

In addition to basis functions and modes given in the table, there are also coefficients for each mode, let's call them c_{lmn}. So, in total, for $L = N = 3$ case (where L and M are defined as in theory not DESC version), there are 70 modes. In general, the number of modes can be expressed in terms of chosen resolutions as,

$$\begin{aligned} \textit{Total Number of Modes} &= \sum_{n=-N}^{N} \sum_{l=0}^{L} (l+1) \\ &= (2N+1)\left(\frac{L(L+1)}{2} + L + 1\right) \\ &= \frac{1}{2}(2N+1)(L+1)(L+2) \end{aligned} \tag{35}$$

2.4 Stellarator Symmetry

Stellarators have different symmetries which we make use of in numerical simulations. One of them is Stellarator symmetry. If a quantity exhibits this symmetry, then the relation between cylindrical coordinates R, Z and ϕ expressed in flux coordinates is,

$$R(\rho, \theta, \zeta) = R(\rho, -\theta, -\zeta) \tag{36}$$

$$Z(\rho, \theta, \zeta) = -Z(\rho, -\theta, -\zeta) \tag{37}$$

$$\phi(\rho, \theta, \zeta) = -\phi(\rho, -\theta, -\zeta) \tag{38}$$

This means R is an even function with respect to coordinates θ and ζ, whereas Z and ϕ are odd functions. In later chapters, we will use λ instead of ϕ. Therefore, in DESC, the stellarator symmetry implies,

$$R(\rho,\theta,\zeta) = R(\rho,-\theta,-\zeta) \tag{39}$$

$$Z(\rho,\theta,\zeta) = -Z(\rho,-\theta,-\zeta) \tag{40}$$

$$\lambda(\rho,\theta,\zeta) = -\lambda(\rho,-\theta,-\zeta) \tag{41}$$

From this property, we can say that the Fourier-Zernike basis representation of R will only have cos($m\theta$)cos($n\zeta$) and sin($m\theta$)sin($n\zeta$). For Z and λ, allowed terms are sin($m\theta$)cos($n\zeta$) and cos($m\theta$)sin($n\zeta$).

Then, with stellarator symmetry applied, R, Z and λ can be written as,

$$R(\rho,\theta,\zeta) = \sum_l \sum_m \sum_n R_{lmn} \begin{cases} cos(m\theta)cos(n\zeta)\mathcal{R}_l^m(\rho) & m,n \geq 0 \\ sin(m\theta)sin(n\zeta)\mathcal{R}_l^m(\rho) & m,n < 0 \end{cases} \tag{42}$$

$$Z(\rho,\theta,\zeta) = \sum_l \sum_m \sum_n Z_{lmn} \begin{cases} cos(m\theta)sin(n\zeta)\mathcal{R}_l^m(\rho) & m \geq 0, n < 0 \\ sin(m\theta)cos(n\zeta)\mathcal{R}_l^m(\rho) & m < 0, n \geq 0 \end{cases} \tag{43}$$

$$\lambda(\rho,\theta,\zeta) = \sum_l \sum_m \sum_n \lambda_{lmn} \begin{cases} cos(m\theta)sin(n\zeta)\mathcal{R}_l^m(\rho) & m \geq 0, n < 0 \\ sin(m\theta)cos(n\zeta)\mathcal{R}_l^m(\rho) & m < 0, n \geq 0 \end{cases} \tag{44}$$

Special condition at $\rho = 0$, remember property in 15,

$$R(\rho = 0,\theta,\zeta) = \sum_l \sum_n \begin{cases} 0 & \text{odd } l \\ R_{l0n}(-1)^{l/2}cos(n\zeta) & \text{even } l \end{cases} \tag{45}$$

$$Z(\rho = 0,\theta,\zeta) = -\sum_l \sum_n \begin{cases} 0 & \text{odd } l \\ Z_{l0n}(-1)^{l/2}sin(n\zeta) & \text{even } l \end{cases} \tag{46}$$

$$\lambda(\rho = 0,\theta,\zeta) = -\sum_l \sum_n \begin{cases} 0 & \text{odd } l \\ \lambda_{l0n}(-1)^{l/2}sin(n\zeta) & \text{even } l \end{cases} \tag{47}$$

Special condition at $\rho = 1$, radial part of Zernike polynomial is always 1 for $\rho = 1$,

$$R(\rho = 1,\theta,\zeta) = \sum_l \sum_m \sum_n R_{lmn} \begin{cases} cos(m\theta)cos(n\zeta) & m,n \geq 0 \\ sin(m\theta)sin(n\zeta) & m,n < 0 \end{cases} \tag{48}$$

$$Z(\rho = 1,\theta,\zeta) = \sum_l \sum_m \sum_n Z_{lmn} \begin{cases} cos(m\theta)sin(n\zeta) & m \geq 0, n < 0 \\ sin(m\theta)cos(n\zeta) & m < 0, n \geq 0 \end{cases} \tag{49}$$

$$\lambda(\rho = 1,\theta,\zeta) = \sum_l \sum_m \sum_n \lambda_{lmn} \begin{cases} cos(m\theta)sin(n\zeta) & m \geq 0, n < 0 \\ sin(m\theta)cos(n\zeta) & m < 0, n \geq 0 \end{cases} \tag{50}$$

$\rho = 1$ surface is frequently used in DESC as a boundary condition. Class FourierRZToroidalSurface() contains given $\rho = 1$ surface R and Z functions.

3 Coordinates

Magnetic confinement machines have the shape of torus (or donut). Instead of using cartesian coordinate system which has straight lines as axis, due to the curved shaped of the torus, we can use curvilinear coordinates. For example, any function inside this geometry can be defined in terms of cylindrical coordinates with R (radial part), ϕ (toroidal angle) and Z (height). However, as you can see from the right side of Figure 3, there is a more convenient set of curvilinear coordinates that can define any point of this torus which is called flux coordinates.

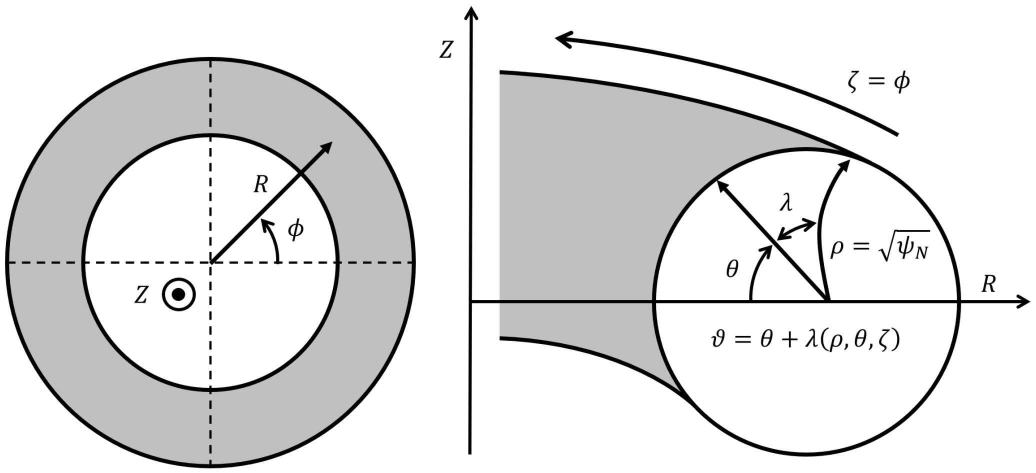


Figure 3: **Sketch a better figure for this**

These coordinates are ρ, θ and ζ. Here, ζ is exactly the same as ϕ in cylindrical coordinates. ρ is similar to the radial coordinate of cylindrical coordinates but since we will try to find more intricate shapes rather than a bunch of circles, instead of defining it as the distance to the center, we will define it as a function of the flux and this will express the index of flux surface. Lastly, θ is the poloidal angle.

3.1 Covariant and Contravariant Vectors

For a chosen set of variables u^i, we can define the constant u^i surfaces and the curves that u^i increase. The surfaces will have a normal vector of $\vec{\nabla} u^i$. And the direction of the curves are $\frac{\partial \vec{R}}{\partial u^i}$. Notice these new parameters and the coordinates that define them don't need to be orthogonal or unit vectors.

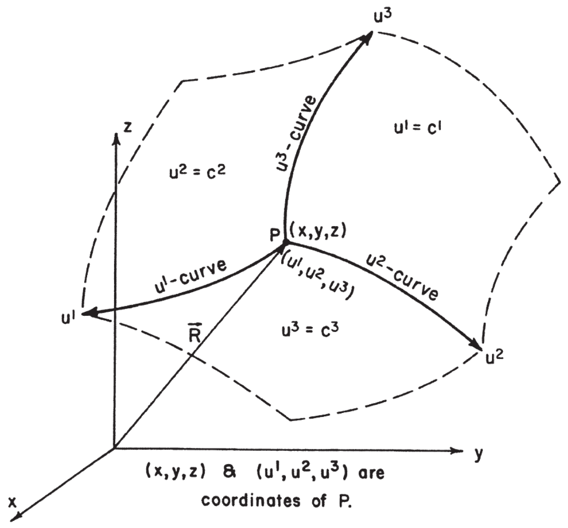


Fig. 2.1. General curvilinear coordinate system with coordinates (u^1, u^2, u^3). The coordinate curves and coordinate surfaces are represented with respect to a Cartesian system with coordinates (x, y, z)

Figure 4: General curvilinear coordinate system [1]

The differential change of u^i can be written as,

$$du^i = \nabla u^i \cdot d\mathbf{R} \tag{51}$$

Since $\vec{R}$ can be written as $\vec{R}(u^1, u^2, u^3)$, we have,

$$d\mathbf{R} = \sum_{j=1}^{3} \frac{\partial \vec{R}}{\partial u^j} du^j \qquad \Rightarrow \qquad du^i = \nabla u^i \cdot \sum_{j=1}^{3} \frac{\partial \vec{R}}{\partial u^j} du^j \tag{52}$$

$$du^i = \nabla u^i \cdot \frac{\partial \vec{R}}{\partial u^i} du^i \qquad \Rightarrow \qquad \nabla u^i \cdot \frac{\partial \vec{R}}{\partial u^i} = \delta^i_j \tag{53}$$

From now on, we will use covariant and contravariant vectors with superscripts and subscripts,

$$\textbf{Contravariant:} \qquad \mathbf{e}_i = \frac{\partial \vec{R}}{\partial u^i} \tag{54}$$

$$\textbf{Covariant:} \qquad \mathbf{e}^i = \nabla u^i \tag{55}$$

$$\mathbf{e}^i \cdot \mathbf{e}_i = \delta^i_j \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases} \tag{56}$$

And the transformation from these two sets of vectors is,

$$\mathbf{e}^1 = \frac{\mathbf{e}_2 \times \mathbf{e}_3}{\mathbf{e}_1 \cdot (\mathbf{e}_2 \times \mathbf{e}_3)} \qquad\qquad \mathbf{e}_1 = \frac{\mathbf{e}^2 \times \mathbf{e}^3}{\mathbf{e}^1 \cdot (\mathbf{e}^2 \times \mathbf{e}^3)} \tag{57}$$

Since these are reciprocal sets of vectors, a vector P can be written as,

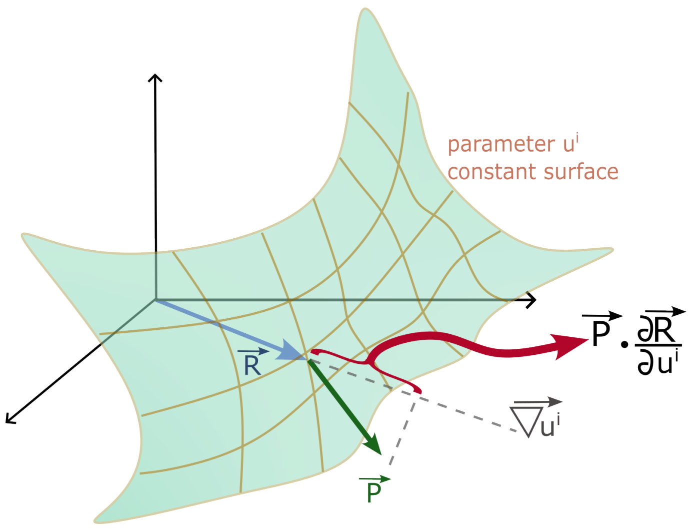
$$\mathbf{P} = (\mathbf{P}\cdot\mathbf{e}_1)\mathbf{e}^1 + (\mathbf{P}\cdot\mathbf{e}_2)\mathbf{e}^2 + (\mathbf{P}\cdot\mathbf{e}_3)\mathbf{e}^3 \tag{58}$$

$$\text{or } \mathbf{P} = (\mathbf{P}\cdot\mathbf{e}^1)\mathbf{e}_1 + (\mathbf{P}\cdot\mathbf{e}^2)\mathbf{e}_2 + (\mathbf{P}\cdot\mathbf{e}^3)\mathbf{e}_3 \tag{59}$$

$$\text{we will use } P_i = \mathbf{P}\cdot\mathbf{e}_i \quad \text{and} \quad P^i = \mathbf{P}\cdot\mathbf{e}^i \tag{60}$$

$$\mathbf{P} = P_1\mathbf{e}^1 + P_2\mathbf{e}^2 + P_3\mathbf{e}^3 \quad \text{or} \quad \mathbf{P} = P^1\mathbf{e}_1 + P^2\mathbf{e}_2 + P^3\mathbf{e}_3 \tag{61}$$

[1]For example in cartesian coordinates, $P_x = \mathbf{P}\cdot\hat{x}$ and $\mathbf{P} = (\mathbf{P}\cdot\hat{x})\hat{x} + (\mathbf{P}\cdot\hat{y})\hat{y} + (\mathbf{P}\cdot\hat{z})\hat{z}$. Note that in orthonormal coordinate systems, the contravariant and covariant basis vectors are the same, implying $\mathbf{e}^i = \mathbf{e}_i = \mathbf{e}_i/\|\mathbf{e}_i\|$, so they are self-reciprocal.



Now, back to the stellarators. We will use contravariant basis vectors for the ρ, θ and ζ as follows,

$$\mathbf{e}_i = \frac{\partial\vec{R}}{\partial u^i} = \frac{\partial}{\partial u^i}(R\hat{R} + Z\hat{Z}) \tag{62}$$

In general, the derivative of the unit vectors is 0, but only the ζ contravariant vector has an exception.

$$\mathbf{e}_\zeta = \frac{\partial}{\partial\zeta}(R\hat{R} + Z\hat{Z}) = \frac{\partial R}{\partial\zeta}\hat{R} + R\frac{\partial\hat{R}}{\partial\zeta} + \frac{\partial Z}{\partial\zeta}\hat{Z} + Z\frac{\partial\hat{Z}}{\partial\zeta} = \frac{\partial R}{\partial\zeta}\hat{R} + R\hat{\phi} + \frac{\partial Z}{\partial\zeta}\hat{Z} \tag{63}$$

We got an additional $R\hat{\phi}$ term due to the partial derivative of $\hat{R}$ with respect to ζ. Using similar steps, one can get the full set of covariant basis vectors as follows,

$$\mathbf{e}_\rho = \begin{bmatrix}\partial_\rho R\\0\\\partial_\rho Z\end{bmatrix} \qquad \mathbf{e}_\theta = \begin{bmatrix}\partial_\theta R\\0\\\partial_\theta Z\end{bmatrix} \qquad \mathbf{e}_\zeta = \begin{bmatrix}\partial_\zeta R\\R\\\partial_\zeta Z\end{bmatrix} \tag{64}$$

If you are confused about the basis vectors being different for each point, think about cylindrical or spherical coordinates. $\hat{\theta}$ and $\hat{\phi}$ change direction for every point in space. Since we use them a lot and they are orthogonal, we tend to ignore this fact. But in general, you can choose any random parameter that has defined constant surfaces. Then, you can create a covariant or contravariant basis using those parameters. For example, the spherical

[1]This seems a bit counter-intuitive because, in orthonormal coordinates systems (that are the most used coordinates), you should take the dot product of the vector with the basis vector you want to find the component of.

coordinates have constant surfaces as shown in Figure 5. the basis vectors are perpendicular to those surfaces. I hope this example didn't even confuse you about spherical and cylindrical coordinates. Give yourself some time, and read this a couple of times. I assure you it will make sense after some point.

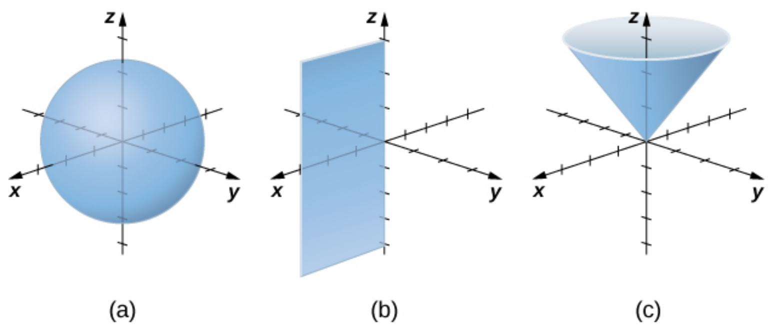


Figure 5: Surfaces of constant parameters (a) ρ, (b) θ and (c) ϕ [2]

And, we will also need the Jacobian of this basis vectors that can be found by,

$$\sqrt{g} = \mathbf{e}_\rho \cdot (\mathbf{e}_\theta \times \mathbf{e}_\zeta) \qquad \text{or} \qquad \sqrt{g} = \begin{vmatrix} \partial_\rho R & \partial_\rho \phi & \partial_\rho Z \\ \partial_\theta R & \partial_\theta \phi & \partial_\theta Z \\ \partial_\zeta R & \partial_\zeta \phi & \partial_\zeta Z \end{vmatrix} \qquad \text{or} \qquad \frac{1}{\sqrt{g}} = \mathbf{e}^\rho \cdot (\mathbf{e}^\theta \times \mathbf{e}^\zeta) \tag{65}$$

$$\begin{aligned} \sqrt{g} &= \begin{bmatrix} \partial_\rho R & 0 & \partial_\rho Z \end{bmatrix} \cdot \begin{vmatrix} \hat{r} & \hat{\phi} & \hat{z} \\ \partial_\theta R & 0 & \partial_\theta Z \\ \partial_\zeta R & R & \partial_\zeta Z \end{vmatrix} \\ &= \begin{bmatrix} \partial_\rho R & 0 & \partial_\rho Z \end{bmatrix} \begin{bmatrix} R\partial_\theta Z \\ \partial_\theta Z \partial_\zeta R - \partial_\theta R \partial_\zeta Z \\ R\partial_\theta R \end{bmatrix} \end{aligned} \tag{66}$$

Finally, we get,

$$\sqrt{g} = R\left(\frac{\partial R}{\partial \rho}\frac{\partial Z}{\partial \theta} + \frac{\partial R}{\partial \theta}\frac{\partial Z}{\partial \rho}\right) \tag{67}$$

with Jacobian, we can rewrite transformations as,

$$\mathbf{e}^i \times \mathbf{e}^j = \varepsilon^{ijk}\frac{\mathbf{e}_k}{\sqrt{g}} \tag{68}$$

$$\mathbf{e}_i \times \mathbf{e}_j = \varepsilon^{ijk}\sqrt{g}\mathbf{e}^k \tag{69}$$

$$\mathbf{e}^1 = \frac{\mathbf{e}_2 \times \mathbf{e}_3}{\sqrt{g}} \qquad \mathbf{e}_1 = \sqrt{g}(\mathbf{e}^2 \times \mathbf{e}^3) \tag{70}$$

3.2 Vector Operations

3.2.1 Dot Product

$$g_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j \tag{71}$$

$$\mathbf{A}\cdot\mathbf{B} = A^i B_j = g_{ij} A^i B^j \tag{72}$$

3.2.2 Cross Product

$$\mathbf{A}\times\mathbf{B} = \sqrt{g}\sum_k (A^i B^j - A^j B^i)\mathbf{e}^k \tag{73}$$

$$= \frac{1}{\sqrt{g}}\sum_k (A_i B_j - A_j B_i)\mathbf{e}_k \tag{74}$$

3.3 Del Operator in Curvilinear Coordinates

Look for D'Haeseleer book [1] page 35 for details.

Del operator for an arbitrary curvilinear coordinate frame is defined as,

$$\nabla \equiv \nabla u^i \equiv \mathbf{e}^i \frac{\partial}{\partial u^i} \tag{75}$$

3.3.1 Gradient

The gradient can be expressed as,

$$\nabla\Phi = \sum_i \frac{\partial\Phi}{\partial u^i}\nabla u^i = \sum_i \frac{\partial\Phi}{\partial u^i}\mathbf{e}^i \tag{76}$$

3.3.2 Divergence

The divergence can be expressed as,

$$\nabla\cdot\mathbf{A} = \frac{1}{\sqrt{g}}\sum_i \frac{\partial}{\partial u^i}(\sqrt{g}A^i) \tag{77}$$

3.3.3 Curl

The curl can be expressed as,

$$\begin{aligned}
\nabla\times\mathbf{A} &= \nabla\times(\sum_j A_j\mathbf{e}^j) && (78)\\
\text{distribute curl} \quad &= \sum_j \left(A_j(\nabla\times\mathbf{e}^j) + \nabla A_j\times\mathbf{e}^j\right) && (79)\\
\text{use } \nabla\times\mathbf{e}^j = 0 \text{ and definition of gradient} \quad &= \sum_j\sum_i \frac{\partial A_j}{\partial u^i}\mathbf{e}^i\times\mathbf{e}^j && (80)\\
\text{use cross product property eq. 68} \quad &= \sum_j\sum_i \frac{\varepsilon^{ijk}}{\sqrt{g}}\frac{\partial A_j}{\partial u^i}\mathbf{e}_k && (81)
\end{aligned}$$

$$\nabla\times\mathbf{A} = \frac{1}{\sqrt{g}}\sum_k\left(\frac{\partial A_j}{\partial u^i} - \frac{\partial A_i}{\partial u^j}\right)\mathbf{e}_k \tag{82}$$

3.4 DESC Coordinates

In DESC, we use flux coordinates (ρ, θ and ζ) to define the flux surface shape in R, Z and λ coordinates. This is the first time I introduce λ, I know it is a bit out of the blue, but for now just accept it, later we will derive it and explain why we need it. So, the flux surface shape can be written as,

$$R(\rho, \theta, \zeta) \tag{83}$$

$$Z(\rho, \theta, \zeta) \tag{84}$$

$$\lambda(\rho, \theta, \zeta) \tag{85}$$

Do not confuse $\vec{\mathbf{R}}$ and R. The former is the whole vector whereas the latter is only the radial distance component in cylindrical coordinates. And, these functions will be expressed in terms of the Fourier-Zernike basis shown by equation 34. Again, do not confuse $\mathcal{R}$ and R. $\mathcal{R}$ is used for the radial part of Zernike Polynomials. Sorry for the similar symbols but this is the convention.

We talked about many different coordinates and functions until now, I know it seems to be confusing. I'll try to explain why we use these briefly. Our overall aim is to find some parameters that make the force error minimum (with the given pressure profile or some other adequate information). To calculate force error, we need to find the magnetic field and electric current density. In the next section, we will see that current density can be derived from magnetic field. So, all we need is to find magnetic field. We could have chosen many different coordinates but since by definition the magnetic field is tangent to flux surfaces, we automatically reduce the number of components to calculate to 2. So, it is convenient to use flux surfaces. And previously, we showed that if you choose a set of surfaces that keep some parameters constant, you can construct a new basis. Since we use flux surfaces, we call these flux coordinates. Here, R and Z in eq 83 and 84 are the same as in equation 67. λ is somehow making our calculations easier. Mainly, the shape of the flux surface is defined by R and Z, but for the direction of the magnetic field on that surface, we need λ, this will be proven. Finally, since we know that our geometry will be some form of a torus, we are using the Fourier-Zernike basis (Fourier for toroidal direction and Zernike for a cross-section in a constant toroidal angle). So, the nature of our problem kind of forces us to use them, like quantum mechanics forces you to use spherical Bessel functions or signal processing forces you to use Fourier series or many other examples. Hard problem brings some fancy mathematical tool to come in handy. Please quote this as my words...

So, very briefly the procedure will be,

- Give the input parameters
- Make an initial guess for R, Z and λ
- Plug them in equation 127 to get $\vec{\mathbf{B}}$
- Calculate Force error from 92
- Use an optimization algorithm to converge on R, Z and λ

In general, we will define the resolution of Fourier-Zernike basis. So, the optimization will try to find coefficients of each mode in equation 31 which are c_{lmn}'s and which make force error minimum. For R, Z and λ, we will use R$_{lmn}$, Z$_{lmn}$ and λ_{lmn}. The abovementioned equations will be derived in the next section.

$$R(\rho, \theta, \zeta) = \sum_{m=-M, n=-N, l=0}^{M,N,L} R_{lmn} \mathcal{Z}_l^m(\rho, \theta) \mathcal{F}^n(\zeta) \tag{86a}$$

$$\lambda(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} \lambda_{lmn} \mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta) \tag{86b}$$

$$Z(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} Z_{lmn} \mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta) \tag{86c}$$

4 Fields

4.1 Short Discussion about Rotational Transform ι

Imagine a magnetic confinement machine with only toroidal magnetic field. Since the magnetic field changes direction, there is a gradient (or some call it curvature, but they both serve the same purpose). From single particle in electromagnetic field demonstrations, you should know that if there is a gradient, there should be a $\nabla \mathbf{B} \times \mathbf{B}$ drift. For the tokamak shape, the gradient $\nabla \mathbf{B}$ is in $\hat{r}$ so radial direction, and the magnetic field itself, as assumed, is in $\hat{\phi}$ direction. From the cross product, this will result in upward or downward motion for ions and electrons. So, the charges will be separated. This separation will induce an electric field in the vertical direction $-\hat{z}$. And again from single particle motion, there will be a $\mathbf{e} \times \mathbf{B}$ drift. From cross product of $-\hat{z} \times \hat{\phi}$, the drift will be in radially outward $\hat{r}$ direction. Therefore, with only a toroidal magnetic field, we cannot confine charged particles.

A poloidal magnetic field is required for confinement, resulting in field lines which twist about flux surfaces. The twist in the field lines is quantified by the rotational transform, ι, which indicates the number of poloidal turns of a field line around the magnetic axis for each toroidal turn around the $\hat{\mathbf{z}}$ axis.

$$\iota = \frac{B^{\theta^*}}{B^{\varsigma}} \tag{87}$$

Even if flux surfaces do not exist, the rotational transform can be defined with respect to the poloidal and toroidal rotation of a field line with respect to the magnetic axis. Often in the tokamak literature, the safety factor $\mathrm{q} = 1/\iota$ is used rather than the rotational transform. If flux surfaces exist, the rotational transform can also be defined in terms of the fluxes as,

$$\iota = \frac{\partial \psi_P / \partial \rho}{\partial \psi_T / \partial \rho} \tag{88}$$

When the rotational transform is rational, $\iota = \mathrm{m/n}$ for m and n integers, a field line closes on itself after n toroidal turns, having completed m poloidal turns. Therefore, a single field line does not cover the entire surface (see Figure 6). Flux surfaces with rational values of ι are known as rational surfaces. When ι is irrational, a given field line comes arbitrarily close to every point on what is called an irrational surface.

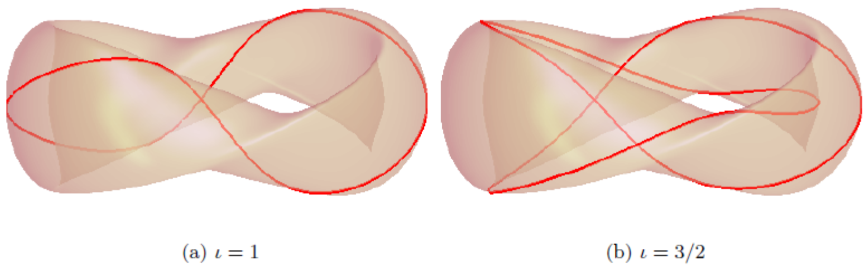

(a) $\iota = 1$ (b) $\iota = 3/2$

Figure 6: Rational Transform Iota and Field lines closing on themselves [3]

4.2 Ideal MHD Equations

In DESC, we use the ideal MHD model for the plasma. So, the equations that we are solving are,

$$\textbf{Force Balance} : \quad \mathbf{J} \times \mathbf{B} = \nabla p \tag{89}$$

$$\textbf{Ampère's Law} : \quad \nabla \times \mathbf{B} = \mu_0 \mathbf{J} \tag{90}$$

$$\textbf{Gauss's Law for Magnetism} : \quad \nabla \cdot \mathbf{B} = 0 \tag{91}$$

We can write **J** from equation 90, and then equation 89 becomes,

$$(\nabla \times \mathbf{B}) \times \mathbf{B} = \mu_0 \nabla p \tag{92}$$

4.2.1 Magnetic Field

Now, let's find the magnetic field in terms of flux coordinates ρ, θ and ζ. Previously, we said that ρ is a flux function but we didn't explain it. In DESC, we are assuming nested flux surfaces. So, we can use this property instead of radial distance. The flux surfaces can be labeled by the total enclosed toroidal flux. In DESC, we will use the following relation for ρ,

$$\rho = \sqrt{\frac{\psi_T}{\psi_{T_{total}}}} \tag{93}$$

We will use the contravariant form for the magnetic field,

$$\mathbf{B} = B^\rho \mathbf{e}_\rho + B^\theta \mathbf{e}_\theta + B^\zeta \mathbf{e}_\zeta \tag{94}$$

Due to the definition of flux surfaces, $\mathbf{B} \cdot \nabla \psi = 0$. Hence, $B^\rho = 0$ and

$$\mathbf{B} = B^\theta \mathbf{e}_\theta + B^\zeta \mathbf{e}_\zeta \tag{95}$$

Substitute **B** in equation 91, and use divergence in non-orthogonal coordinates 77,

$$\frac{1}{\sqrt{g}} \left(\frac{\partial (B^\theta \sqrt{g})}{\partial \theta} + \frac{\partial (B^\zeta \sqrt{g})}{\partial \zeta} \right) = 0 \tag{96}$$

This requires the term in parenthesis to be 0,

$$\frac{\partial (B^\theta \sqrt{g})}{\partial \theta} = -\frac{\partial (B^\zeta \sqrt{g})}{\partial \zeta} \tag{97}$$

Let's try random functions for the solution,

$$B^\theta \sqrt{g} = A(\rho, \theta, \zeta) + a(\rho) \tag{98}$$

$$B^\zeta \sqrt{g} = C(\rho, \theta, \zeta) + c(\rho) \tag{99}$$

The functions that only depend on ρ are added deliberately since these functions will cancel out for both partial derivatives of θ and ζ, such that the equation 97 will be,

$$\frac{\partial A}{\partial \theta} = -\frac{\partial C}{\partial \zeta} \tag{100}$$

Let's choose,

$$A(\rho,\theta,\zeta) = -f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\zeta} \quad \rightarrow \quad \frac{\partial A}{\partial\theta} = -f(\rho)\frac{\partial^2\lambda}{\partial\theta\partial\zeta} \tag{101}$$

$$C(\rho,\theta,\zeta) = f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\theta} \quad \rightarrow \quad \frac{\partial C}{\partial\zeta} = f(\rho)\frac{\partial^2\lambda}{\partial\theta\partial\zeta} \tag{102}$$

which satisfies our condition in the equation 97. Then,

$$B^\theta\sqrt{g} = a(\rho) - f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\zeta} \tag{103}$$

$$B^\zeta\sqrt{g} = c(\rho) + f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\theta} \tag{104}$$

To find $a(\rho)$, $c(\rho)$ and $f(\rho)$, we should look at the toroidal and poloidal fluxes which are given properties for equilibrium calculation. First, we will look at toroidal flux,

$$\psi_T = \int\int B^\zeta\sqrt{g}d\rho d\theta \tag{105}$$

$$= \int\int c(\rho')d\rho' d\theta + \int\left(\int_0^{2\pi}\frac{\partial\lambda}{\partial\theta}d\theta\right)f(\rho')d\rho' \tag{106}$$

$$= \int\int c(\rho')d\rho' d\theta + \int\big(\lambda(2\pi,\zeta) - \lambda(0,\zeta)\big)f(\rho')d\rho' \tag{107}$$

$$= 2\pi\int_0^\rho c(\rho')d\rho' \tag{108}$$

In the third step, we could cancel the second integral because λ is a periodic function. So, $\lambda(2\pi,\zeta) = \lambda(0,\zeta)$.

$$\psi_T = 2\pi\int_0^\rho c(\rho')d\rho' \tag{109}$$

$$c(\rho) = \frac{1}{2\pi}\frac{\partial\psi_T}{\partial\rho} \tag{110}$$

I took the derivative of the first equation in terms of ρ. If we subsitute $c(\rho)$ in B^ζ, we get,

$$B^\zeta = \frac{1}{\sqrt{g}}\left(\frac{1}{2\pi}\frac{\partial\psi_T}{\partial\rho} + f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\theta}\right) \tag{111}$$

We can conduct similar steps for the poloidal flux,

$$\psi_P = \int\int B^\theta\sqrt{g}d\rho d\zeta \tag{112}$$

$$= \int\int a(\rho')d\rho' d\zeta - \int\left(\int_0^{2\pi}\zeta\right)f(\rho')d\rho' \tag{113}$$

$$= \int\int a(\rho')d\rho' d\zeta - \int\big(\lambda(\theta,2\pi) - \lambda(\theta,0)\big)f(\rho')d\rho' \tag{114}$$

$$= 2\pi\int a(\rho')d\rho' \tag{115}$$

Again, I used the periodicity of λ,

$$\psi_P = 2\pi \int_0^\rho a(\rho')d\rho' \tag{116}$$

$$a(\rho) = \frac{1}{2\pi}\frac{\partial \psi_P}{\partial \rho} \tag{117}$$

Remember that $\iota = \dfrac{\partial\psi_P/\partial\rho}{\partial\psi_T/\partial\rho}$. So, $\partial\psi_P/\partial\rho = \iota\ \partial\psi_T/\partial\rho$. Hence, $a(\rho)$ can be written as,

$$a(\rho) = \frac{\iota}{2\pi}\frac{\partial \psi_T}{\partial \rho} \tag{118}$$

If we subsitute $a(\rho)$ in B^θ, we get,

$$B^\theta = \frac{1}{\sqrt{g}}\left(\frac{\iota}{2\pi}\frac{\partial \psi_T}{\partial \rho} - f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\zeta}\right) \tag{119}$$

Finally, the magnetic field can be written as,

$$\mathbf{B} = \frac{1}{\sqrt{g}}\left[\left(\frac{\iota}{2\pi}\frac{\partial \psi_T}{\partial \rho} - f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\zeta}\right)\mathbf{e}_\theta + \left(\frac{1}{2\pi}\frac{\partial \psi_T}{\partial \rho} + f(\rho)\frac{\partial\lambda(\theta,\zeta)}{\partial\theta}\right)\mathbf{e}_\zeta\right] \tag{120}$$

Since there is no constraint on $f(\rho)$, we can take it, for our convenience, as $f(\rho) = \dfrac{f^*(\rho)}{2\pi}\dfrac{\partial\psi_T}{\partial\rho}$

$$\mathbf{B} = \frac{1}{\sqrt{g}}\left[\left(\frac{\iota}{2\pi}\frac{\partial \psi_T}{\partial \rho} - \frac{f^*(\rho)}{2\pi}\frac{\partial \psi_T}{\partial \rho}\frac{\partial\lambda(\theta,\zeta)}{\partial\zeta}\right)\mathbf{e}_\theta + \left(\frac{1}{2\pi}\frac{\partial \psi_T}{\partial \rho} + \frac{f^*(\rho)}{2\pi}\frac{\partial \psi_T}{\partial \rho}\frac{\partial\lambda(\theta,\zeta)}{\partial\theta}\right)\mathbf{e}_\zeta\right] \tag{121}$$

and we can include $f^*(\rho)$ inside λ. Then, we get,

$$\mathbf{B} = \frac{1}{2\pi\sqrt{g}}\frac{\partial \psi_T}{\partial \rho}\left[\left(\iota - \frac{\partial\lambda(\rho,\theta,\zeta)}{\partial\zeta}\right)\mathbf{e}_\theta + \left(1 + \frac{\partial\lambda(\rho,\theta,\zeta)}{\partial\theta}\right)\mathbf{e}_\zeta\right] \tag{122}$$

This last step may seem a bit arbitrary, but look what happens if we create a new coordinate system using $\vartheta = \theta + \lambda(\rho,\theta,\zeta)$. In DESC, this new coordinate is shown by ϑ or (vartheta), but due to its extreme similarity to θ, I will use θ^* instead (s.t. $\theta^* = \theta + \lambda(\rho,\theta,\zeta)$). Let's write the basis vectors in the new coordinate frame,

$$\mathbf{e}_\theta|_{\rho,\zeta} = \frac{\partial \mathbf{R}}{\partial\theta} \qquad \rightarrow \qquad \mathbf{e}_\theta|_{\rho,\zeta} = \frac{\partial \mathbf{R}}{\partial\theta^*}\frac{\partial\theta^*}{\partial\theta} \tag{123}$$

$$\mathbf{e}_\theta|_{\rho,\zeta} = \frac{\partial \mathbf{R}}{\partial\theta^*}\left(1 + \frac{\partial\lambda}{\partial\theta}\right) \qquad \rightarrow \qquad \boxed{\mathbf{e}_\theta|_{\rho,\zeta} = \left(1 + \frac{\partial\lambda}{\partial\theta}\right)\mathbf{e}_{\theta^*}|_{\rho,\zeta}} \tag{124}$$

Here, the notation $\mathbf{e}_\theta|_{\rho,\zeta}$ is used to specify at which point the derivative is evaluated. We can follow similar steps for $\mathbf{e}_\zeta$,

$$\mathbf{e}_\zeta|_{\rho,\theta} = \frac{\partial \mathbf{R}}{\partial \zeta}|_{\rho,\theta} \qquad \rightarrow \qquad \mathbf{e}_\zeta|_{\rho,\theta} = \frac{\partial \mathbf{R}}{\partial \zeta}|_{\rho,\theta^*} + \frac{\partial \mathbf{R}}{\partial \theta^*}\frac{\partial \theta^*}{\partial \zeta} \tag{125}$$

$$\mathbf{e}_\zeta|_{\rho,\theta} = \mathbf{e}_\zeta|_{\rho,\theta^*} + \frac{\partial(\theta+\lambda)}{\partial \zeta}\mathbf{e}_{\theta^*}|_{\rho,\zeta} \qquad \rightarrow \qquad \boxed{\mathbf{e}_\zeta|_{\rho,\theta} = \mathbf{e}_\zeta|_{\rho,\theta^*} + \frac{\partial \lambda}{\partial \zeta}\mathbf{e}_{\theta^*}|_{\rho,\zeta}} \tag{126}$$

If we rewrite equation 122 by specifying the previously hidden evaluation points,

$$\mathbf{B} = \frac{1}{2\pi\sqrt{g}}\frac{\partial \psi_T}{\partial \rho}\left[\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\mathbf{e}_\theta|_{\rho,\zeta} + \left(1 + \frac{\partial \lambda}{\partial \theta}\right)\mathbf{e}_\zeta|_{\rho,\theta}\right] \tag{127}$$

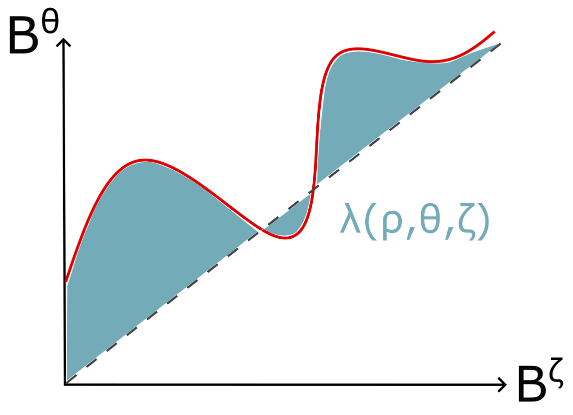
Substitute 124 and 126 in to this equation,

$$\begin{aligned}\mathbf{B} &= \frac{1}{2\pi\sqrt{g}}\frac{\partial \psi_T}{\partial \rho}\left[\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\left(1 + \frac{\partial \lambda}{\partial \theta}\right)\mathbf{e}_{\theta^*}|_{\rho,\zeta} + \left(1 + \frac{\partial \lambda}{\partial \theta}\right)\left(\mathbf{e}_\zeta|_{\rho,\theta^*} + \frac{\partial \lambda}{\partial \zeta}\mathbf{e}_{\theta^*}|_{\rho,\zeta}\right)\right] \\ &= \frac{1}{2\pi\sqrt{g}}\left(1 + \frac{\partial \lambda}{\partial \theta}\right)\frac{\partial \psi_T}{\partial \rho}\left[\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\mathbf{e}_{\theta^*}|_{\rho,\zeta} + \mathbf{e}_\zeta|_{\rho,\theta^*} + \frac{\partial \lambda}{\partial \zeta}\mathbf{e}_{\theta^*}|_{\rho,\zeta}\right] \\ \mathbf{B} &= \frac{1}{2\pi\sqrt{g}}\left(1 + \frac{\partial \lambda}{\partial \theta}\right)\frac{\partial \psi_T}{\partial \rho}(\iota\mathbf{e}_{\theta^*}|_{\rho,\zeta} + \mathbf{e}_\zeta|_{\rho,\theta^*})\end{aligned} \tag{128}$$

If we look at B^{θ^*}/B^ζ ratio,

$$\iota = \frac{B^{\theta^*}}{B^\zeta} \tag{129}$$

As it turned out, if we use θ^* instead of θ, the ratio of the magnetic field components gives rotational transform ι. Therefore, in this new coordinate frame, the field lines appear straight and thus, it is called **straight field line coordinates** or SFL. If we draw the magnetic field in the θ direction versus the magnetic field in the ζ direction, we get a curve like the red solid line in the following Figure,



On the other hand, introducing λ gives us the gray dashed line which has a known slope. THIS FIGURE MIGHT NOT MEAN WHAT I WANT IT TO - LOOK AGAIN

4.2.2 Electric Current Density

We can write electric current density $\mathbf{J}$ in terms of $\mathbf{B}$ using equation 90,

$$\mathbf{J} = \frac{1}{\mu_0}\nabla \times \mathbf{B} \tag{130}$$

We need to introduce the curl operator in curvilinear coordinates to calculate $\mathbf{J}$, which is,

$$\nabla \times \mathbf{B} = \frac{1}{\sqrt{g}}\sum_k \left(\frac{\partial B_j}{\partial u^i} - \frac{\partial B_i}{\partial u^j}\right)\mathbf{e}_k \tag{131}$$

where i, j and k are cyclic ([1,2,3], [2,3,1] or [3,2,1]), and here the indices stand for ρ, θ, and ζ. Plugging in the correct indices and expanding the equation, we can write the current density,

$$J^\rho = \frac{\partial_\theta B_\zeta - \partial_\zeta B_\theta}{\mu_0\sqrt{g}} \tag{132}$$

$$J^\theta = \frac{\partial_\zeta B_\rho - \partial_\rho B_\zeta}{\mu_0\sqrt{g}} \tag{133}$$

$$J^\zeta = \frac{\partial_\rho B_\theta - \partial_\theta B_\rho}{\mu_0\sqrt{g}} \tag{134}$$

where,

$$B_\rho = \mathbf{B}\cdot\mathbf{e}_\rho = (B^\theta\mathbf{e}_\theta + B^\zeta\mathbf{e}_\zeta)\cdot\mathbf{e}_\rho \tag{135}$$

$$B_\theta = \mathbf{B}\cdot\mathbf{e}_\theta = (B^\theta\mathbf{e}_\theta + B^\zeta\mathbf{e}_\zeta)\cdot\mathbf{e}_\theta \tag{136}$$

$$B_\zeta = \mathbf{B}\cdot\mathbf{e}_\zeta = (B^\theta\mathbf{e}_\theta + B^\zeta\mathbf{e}_\zeta)\cdot\mathbf{e}_\zeta \tag{137}$$

Notice that we have to find the covariant components of $\mathbf{B}$ to be able to find contravariant $\mathbf{J}$ components.

$$B_\rho = \frac{1}{2\pi\sqrt{g}}\frac{\partial\psi_T}{\partial\rho}\left[\left(\iota - \frac{\partial\lambda}{\partial\zeta}\right)\mathbf{e}_\theta + \left(1 + \frac{\partial\lambda}{\partial\theta}\right)\mathbf{e}_\zeta\right]\cdot\mathbf{e}_\rho \tag{138}$$

$$B_\theta = \frac{1}{2\pi\sqrt{g}}\frac{\partial\psi_T}{\partial\rho}\left[\left(\iota - \frac{\partial\lambda}{\partial\zeta}\right)\mathbf{e}_\theta + \left(1 + \frac{\partial\lambda}{\partial\theta}\right)\mathbf{e}_\zeta\right]\cdot\mathbf{e}_\theta \tag{139}$$

$$B_\zeta = \frac{1}{2\pi\sqrt{g}}\frac{\partial\psi_T}{\partial\rho}\left[\left(\iota - \frac{\partial\lambda}{\partial\zeta}\right)\mathbf{e}_\theta + \left(1 + \frac{\partial\lambda}{\partial\theta}\right)\mathbf{e}_\zeta\right]\cdot\mathbf{e}_\zeta \tag{140}$$

We will need some dot products like $\mathbf{e}_\rho\cdot\mathbf{e}_\theta$ and $\mathbf{e}_\rho\cdot\mathbf{e}_\zeta$,

$$\mathbf{e}_\rho\cdot\mathbf{e}_\theta = \partial_\rho R\partial_\theta R + \partial_\rho Z\partial_\theta Z \tag{141}$$

$$\mathbf{e}_\rho\cdot\mathbf{e}_\zeta = \partial_\rho R\partial_\zeta R + \partial_\rho Z\partial_\zeta Z \tag{142}$$

$$\mathbf{e}_\theta\cdot\mathbf{e}_\zeta = \partial_\theta R\partial_\zeta R + \partial_\theta Z\partial_\zeta Z \tag{143}$$

$$\mathbf{e}_\theta\cdot\mathbf{e}_\theta = (\partial_\theta R)^2 + (\partial_\theta Z)^2 \tag{144}$$

$$\mathbf{e}_\zeta\cdot\mathbf{e}_\zeta = R^2 + (\partial_\zeta R)^2 + (\partial_\zeta Z)^2 \tag{145}$$

Just for showing it one time, the full equations are,

$$B_\rho = \frac{\frac{\partial \psi_T}{\partial \rho}}{2\pi R \left(\frac{\partial R}{\partial \rho}\frac{\partial Z}{\partial \theta} + \frac{\partial R}{\partial \theta}\frac{\partial Z}{\partial \rho}\right)} \left[\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\left(\frac{\partial R}{\partial \rho}\frac{\partial R}{\partial \theta} + \frac{\partial Z}{\partial \rho}\frac{\partial Z}{\partial \theta}\right) + \left(1 + \frac{\partial \lambda}{\partial \theta}\right)\left(\frac{\partial R}{\partial \rho}\frac{\partial R}{\partial \zeta} + \frac{\partial Z}{\partial \rho}\frac{\partial Z}{\partial \zeta}\right)\right] \tag{146}$$

$$B_\theta = \frac{\frac{\partial \psi_T}{\partial \rho}}{2\pi R \left(\frac{\partial R}{\partial \rho}\frac{\partial Z}{\partial \theta} + \frac{\partial R}{\partial \theta}\frac{\partial Z}{\partial \rho}\right)} \left[\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\left((\frac{\partial R}{\partial \theta})^2 + (\frac{\partial Z}{\partial \theta})^2\right) + \left(1 + \frac{\partial \lambda}{\partial \theta}\right)\left(\frac{\partial R}{\partial \theta}\frac{\partial R}{\partial \zeta} + \frac{\partial Z}{\partial \theta}\frac{\partial Z}{\partial \zeta}\right)\right] \tag{147}$$

$$B_\zeta = \frac{\frac{\partial \psi_T}{\partial \rho}}{2\pi R \left(\frac{\partial R}{\partial \rho}\frac{\partial Z}{\partial \theta} + \frac{\partial R}{\partial \theta}\frac{\partial Z}{\partial \rho}\right)} \left[\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\left(\frac{\partial R}{\partial \theta}\frac{\partial R}{\partial \zeta} + \frac{\partial Z}{\partial \theta}\frac{\partial Z}{\partial \zeta}\right) + \left(1 + \frac{\partial \lambda}{\partial \theta}\right)\left(R^2 + (\frac{\partial R}{\partial \zeta})^2 + (\frac{\partial Z}{\partial \zeta})^2\right)\right] \tag{148}$$

4.2.3 J × B Force

If we take the cross product of **J** and **B** using equation 73 , we can write it as,

$$\begin{aligned}
\mathbf{J} \times \mathbf{B} &= \sqrt{g} \sum_k (J^i B^j - J^j B^i)\mathbf{e}^k && (149)\\
&= \sqrt{g}\left[(J^\theta B^\zeta - J^\zeta B^\theta)\mathbf{e}^\rho \;+\; (J^\zeta B^\rho - J^\rho B^\zeta)\mathbf{e}^\theta \;+\; (J^\rho B^\theta - J^\theta B^\rho)\mathbf{e}^\zeta\right] && (150)\\
&= \sqrt{g}\left[(J^\theta B^\zeta - J^\zeta B^\theta)\mathbf{e}^\rho \;-\; J^\rho B^\zeta \mathbf{e}^\theta \;+\; J^\rho B^\theta \mathbf{e}^\zeta\right] && (151)\\
&= \sqrt{g}(J^\theta B^\zeta - J^\zeta B^\theta)\mathbf{e}^\rho \;-\; \sqrt{g} J^\rho (B^\zeta \mathbf{e}^\theta - B^\theta \mathbf{e}^\zeta) && (152)
\end{aligned}$$

I used $B^\rho = 0$ (see B for alternative). We can write the force error as,

$$\mathbf{F} = \mathbf{J} \times \mathbf{B} - \nabla p \tag{153}$$

Considering that pressure is a pure function of ρ, and using equation 152,

$$\mathbf{F} = -\frac{\partial p}{\partial \rho}\nabla \rho + \sqrt{g}(J^\theta B^\zeta - J^\zeta B^\theta)\mathbf{e}^\rho - \sqrt{g} J^\rho (B^\zeta \mathbf{e}^\theta - B^\theta \mathbf{e}^\zeta) \tag{154}$$

Notice that $\nabla \rho = \mathbf{e}^\rho$,

$$\mathbf{F} = \left(-\frac{\partial p}{\partial \rho} + \sqrt{g}(J^\theta B^\zeta - J^\zeta B^\theta)\right)\mathbf{e}^\rho + \sqrt{g} J^\rho (B^\zeta \mathbf{e}^\theta - B^\theta \mathbf{e}^\zeta) \tag{155}$$

In DESC, we have the following notation for the force error,

$$\mathbf{F} = F_\rho \mathbf{e}^\rho + F_\beta \beta_{\mathbf{DESC}} \tag{156}$$

where,

$$F_\rho = \sqrt{g}(J^\theta B^\zeta - J^\zeta B^\theta) - \frac{\partial p}{\partial \rho} \tag{157}$$

$$F_\beta = \sqrt{g} J^\rho \tag{158}$$

$$\beta_{\mathbf{DESC}} = (B^\zeta \mathbf{e}^\theta - B^\theta \mathbf{e}^\zeta) \tag{159}$$

Equivalently, we can substitute the formula for **J** and write,

$$F_\rho = B^\zeta \frac{\partial_\zeta B_\rho - \partial_\rho B_\zeta}{\mu_0} - B^\theta \frac{\partial_\rho B_\theta - \partial_\theta B_\rho}{\mu_0} - \frac{\partial p}{\partial \rho} \tag{160}$$

$$F_\beta = \frac{\partial_\theta B_\zeta - \partial_\zeta B_\theta}{\mu_0} \tag{161}$$

LIFE IS TOO SHORT TO EXPAND THESE EQUATIONS ! BUT IF I EVER FEEL THAT THIS MIGHT HELP, I CAN EXPAND THEM SOME DAY

4.3 Math to compute the rotational transform in DESC

NOTE: This portion is reformatted, there might be missing symbols etc. Might be better read the original PDF

4.3.1 Terminology

- $\langle Q \rangle_{\sqrt{g}} = (\oiint \mathrm{d}\theta \mathrm{d}\zeta \sqrt{g}\, Q)/(\oiint \mathrm{d}\theta \mathrm{d}\zeta \sqrt{g})$ is the flux surface average of Q.
- $\langle Q \rangle = (\oiint \mathrm{d}\theta \mathrm{d}\zeta\, Q)/(\oiint \mathrm{d}\theta \mathrm{d}\zeta)$ is the flux surface average without $\sqrt{g}$ of Q
- $I = \langle B_\theta \rangle$ is the enclosed net toroidal current in tesla-meters
- `data["I"]` $= \mu_0$ `data["current"]`$/2\pi$
- ψ is the toroidal flux normalized by 2 π
- ρ is the flux surface label
- $\mathrm{d}\psi/\mathrm{d}\rho$ is `data["psi_r"]` $=$ `params["Psi"]_r`$/2\pi$
- λ is the poloidal stream function
- ω is the toroidal stream function
- ι is the rotational transform normalized by 2π

4.3.2 Derivation

Any vector can be written as a contravariant coordinate vector dot product with covariant basis vectors.

$$\mathbf{B} = B^\rho \mathbf{e}_\rho + B^\theta \mathbf{e}_\theta + B^\zeta \mathbf{e}_\zeta \tag{162}$$

Assume nested flux surfaces.

$$\mathbf{B} = B^\theta \mathbf{e}_\theta + B^\zeta \mathbf{e}_\zeta \tag{163}$$

The components of the magnetic field are defined as

$$B^\theta = \frac{\mathrm{d}\psi/\mathrm{d}\rho}{\sqrt{g}}\left(\iota - \frac{\partial\lambda}{\partial\zeta}\right) \tag{164}$$
$$B^\zeta = \frac{\mathrm{d}\psi/\mathrm{d}\rho}{\sqrt{g}}\left(1 + \frac{\partial\lambda}{\partial\theta}\right) \tag{165}$$

Rearranging gives an equation for the rotational transform.

$$\iota = \frac{\sqrt{g}}{\mathrm{d}\psi/\mathrm{d}\rho}B^\theta + \frac{\partial\lambda}{\partial\zeta} \tag{166}$$

This suggests you can chain a relation between ι and I through the common terms B^θ and B_θ in the equations for ι and I, respectively.

$$I = \langle B_\theta \rangle \tag{167}$$
$$= \langle \mathbf{B} \cdot \mathbf{e}_\theta \rangle \tag{168}$$
$$= \left\langle \left(B^\theta \mathbf{e}_\theta + B^\zeta \mathbf{e}_\zeta\right) \cdot \mathbf{e}_\theta \right\rangle \tag{169}$$

Plugging in definitions for the coefficients of the metric tensor g,

$$I = \left\langle B^\theta g_{\theta\theta} + B^\zeta g_{\theta\zeta} \right\rangle \tag{170}$$

Surface average operations are additive homomorphisms.

$$I = \left\langle B^\theta g_{\theta\theta} \right\rangle + \left\langle B^\zeta g_{\theta\zeta} \right\rangle \tag{171}$$

Some rearranging eventually yields ι as a function of I.

$$\left\langle B^\theta g_{\theta\theta} \right\rangle = I - \left\langle B^\zeta g_{\theta\zeta} \right\rangle \tag{172}$$
$$\left\langle \frac{\mathrm{d}\psi/\mathrm{d}\rho}{\sqrt{g}}\left(\iota - \frac{\partial\lambda}{\partial\zeta}\right) g_{\theta\theta} \right\rangle = \tag{173}$$
$$\left\langle \frac{\mathrm{d}\psi/\mathrm{d}\rho}{\sqrt{g}} \iota g_{\theta\theta} \right\rangle - \left\langle \frac{\mathrm{d}\psi/\mathrm{d}\rho}{\sqrt{g}} \frac{\partial\lambda}{\partial\zeta} g_{\theta\theta} \right\rangle = \tag{174}$$

For flux surface functions, surface average operations are the identity operation.

$$\frac{\mathrm{d}\psi}{\mathrm{d}\rho}\left(\iota \left\langle \frac{g_{\theta\theta}}{\sqrt{g}} \right\rangle - \left\langle \frac{\partial\lambda}{\partial\zeta} \frac{g_{\theta\theta}}{\sqrt{g}} \right\rangle\right) = I - \left\langle B^\zeta g_{\theta\zeta} \right\rangle \tag{175}$$
$$\iota = \left(\frac{I - \left\langle B^\zeta g_{\theta\zeta} \right\rangle}{\mathrm{d}\psi/\mathrm{d}\rho} + \left\langle \frac{\partial\lambda}{\partial\zeta} \frac{g_{\theta\theta}}{\sqrt{g}} \right\rangle \right) \Big/ \left\langle \frac{g_{\theta\theta}}{\sqrt{g}} \right\rangle \tag{176}$$

4.3.3 Other arrangements

The following are equivalent rearrangements of (176). I am unsure which are more robust numerically. Small magnitudes in the denominator will be bad. Subtracting similar magnitude terms also results in floating point errors.

$$\iota = \left(\frac{I}{\mathrm{d}\psi/\mathrm{d}\rho} + \left\langle \frac{\partial\lambda}{\partial\zeta} \frac{g_{\theta\theta}}{\sqrt{g}} - \frac{B^\zeta g_{\theta\zeta}}{\mathrm{d}\psi/\mathrm{d}\rho} \right\rangle \right) \Big/ \left\langle \frac{g_{\theta\theta}}{\sqrt{g}} \right\rangle \tag{177}$$

Replacing B^ζ gives

$$\iota = \left(\frac{I}{\mathrm{d}\psi/\mathrm{d}\rho} + \left\langle \frac{\frac{\partial \lambda}{\partial \zeta} g_{\theta\theta} - (1 + \frac{\partial \lambda}{\partial \theta}) g_{\theta\zeta}}{\sqrt{g}} \right\rangle \right) \Big/ \left\langle \frac{g_{\theta\theta}}{\sqrt{g}} \right\rangle \tag{178}$$

Integral form

$$\iota = \left(4\pi^2 I + \frac{\mathrm{d}\psi}{\mathrm{d}\rho} \oiint \mathrm{d}\theta \mathrm{d}\zeta \frac{\frac{\partial \lambda}{\partial \zeta} g_{\theta\theta} - (1 + \frac{\partial \lambda}{\partial \theta}) g_{\theta\zeta}}{\sqrt{g}} \right) \Big/ \left(\frac{\mathrm{d}\psi}{\mathrm{d}\rho} \oiint \mathrm{d}\theta \mathrm{d}\zeta \frac{g_{\theta\theta}}{\sqrt{g}} \right) \tag{179}$$

$$\iota = \left(4\pi^2 \frac{I}{\mathrm{d}\psi/\mathrm{d}\rho} + \oiint \mathrm{d}\theta \mathrm{d}\zeta \frac{\frac{\partial \lambda}{\partial \zeta} g_{\theta\theta} - (1 + \frac{\partial \lambda}{\partial \theta}) g_{\theta\zeta}}{\sqrt{g}} \right) \Big/ \left(\oiint \mathrm{d}\theta \mathrm{d}\zeta \frac{g_{\theta\theta}}{\sqrt{g}} \right) \tag{180}$$

The term $4\pi^2 I$ is an additional term that supplements the zero current version of iota to account for the net toroidal current. It can also be derived (not rigorously) by applying Ampere's law on a toroidal cross-section. This is because $\iota \cong \mathrm{d}\chi/\mathrm{d}\psi$ (change in poloidal flux over toroidal flux) and Ampere's law shows that, for every thin slice of a toroidal cross-section, the toroidal current generates an extra poloidal flux of $\int \mathrm{d}\theta(\vec{B} \cdot \vec{e_\theta})$, which is on average equal to $2\pi I \cong 2\pi\langle\vec{B} \cdot \vec{e_\theta}\rangle$. In (178), this factor of 2π is hidden in $\mathrm{d}\psi/\mathrm{d}\rho$ because we normalized the latter by that same factor. TODO: Can prove this rigorously.

4.3.4 Axis limit of rotational transform

We know that $\psi \sim \rho^2$ and $\vec{e_\theta} \to \vec{0}$ as $\rho \to 0$. It follows that

$$\lim_{\rho \to 0} \|\mathbf{e}_\theta\| = 0 \implies \begin{cases} \lim_{\rho\to 0} g_{\theta\theta} \\ \lim_{\rho\to 0} \partial g_{\theta\theta}/\partial\rho \\ \lim_{\rho\to 0} g_{\theta\zeta} \\ \lim_{\rho\to 0} \sqrt{g} \\ \lim_{\rho\to 0} I \end{cases} = 0 \tag{181}$$

$$\left.\begin{matrix} \lim_{\rho\to 0} \partial\|\mathbf{e}_\theta\|/\partial\rho \\ \lim_{\rho\to 0}\|\mathbf{e}_\zeta\| \\ \lim_{\rho\to 0}\|\mathbf{e}_\rho\| \end{matrix}\right\} \neq 0 \implies \begin{cases} \lim_{\rho\to 0} \partial^2 g_{\theta\theta}/\partial\rho^2 \\ \lim_{\rho\to 0} \partial\sqrt{g}/\partial\rho \end{cases} \neq 0 \tag{182}$$

Computing the limits of the integrands in (179) shows that (179) is of the indeterminate form 0/0. However, in that form, the denominator approaches zero more rapidly than the numerator, unless a non-basic combination of parameters is justified to be zero. If that justification is made, then the same result obtained below is recovered after three additional applications of l'Hôpital's rule.

It turns out that the rearrangement that groups the toroidal current and the toroidal flux is more amenable to limit analysis than the others. We know that the rotational transform must be finite at the axis.[2] Since the denominator of (178) tends to zero, it follows that ι is finite only if its numerator tends to zero. Hence, as defined in (178), $\lim_{\rho\to 0} \iota$ is apparently of the indeterminate form 0/0.[3]

[2]This statement is equivalent to claiming $\vec{B} \cdot (\vec{e_\zeta} \times \vec{e_\rho}) = 0$ at the magnetic axis.

[3]I assume interchanging the order of the limit and integral operations is valid in (178).

A sufficient condition for interchanging is uniform convergence of the integrand. Other ways to prove validity of the interchange involve finding some region where the integrand is monotonic. I have failed to prove such properties. I think knowledge of relations that depend on the stellarator is required to deduce such properties. For example, uniform convergence of the simplest integrand in the denominator of (178) requires finding a region, dependent on ϵ, where $\|\mathbf{e}_\theta\| < \epsilon\mathbf{e}_\theta/\|\mathbf{e}_\theta\| \cdot (\mathbf{e}_\zeta \times \mathbf{e}_\rho)$ for all $\epsilon > 0$. This task reduces to showing the covariant basis vectors are never orthogonal, with supremum of angle separation less than $\pi/2$. The alternative requires deducing if $g_{\theta\theta}/\sqrt{g}$ is monotonic near $\rho = 0$.

In any case, my justification for the interchange follows from our use of finite sums to compute the integrals. When the integrals are replaced with finite sums, the interchange is valid because the limit of each term to be summed exists. Perhaps more importantly, This suggests we can find the limit of ι without decomposing the basis functions. With that goal, we define

$$\alpha \cong 4\pi^2 \frac{I}{\mathrm{d}\psi/\mathrm{d}\rho} \tag{183}$$

$$\beta \cong \oiint \mathrm{d}\theta\mathrm{d}\zeta \frac{\frac{\partial\lambda}{\partial\zeta} g_{\theta\theta} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) g_{\theta\zeta}}{\sqrt{g}} \tag{184}$$

$$\gamma \cong \oiint \mathrm{d}\theta\mathrm{d}\zeta \frac{g_{\theta\theta}}{\sqrt{g}} \tag{185}$$

Apply l'Hôpital's rule.

$$\lim_{\rho\to 0} \iota = \lim_{\rho\to 0} \frac{\mathrm{d}(\alpha+\beta)/\mathrm{d}\rho}{\mathrm{d}\gamma/\mathrm{d}\rho} \tag{186}$$

The bounds of the integrals do not depend on ρ, so we may differentiate under both integrals using the Leibniz integral rule.

$$\frac{\mathrm{d}\alpha}{\mathrm{d}\rho} = \frac{4\pi^2}{(\mathrm{d}\psi/\mathrm{d}\rho)^2}\left(\frac{\mathrm{d}I}{\mathrm{d}\rho}\frac{\mathrm{d}\psi}{\mathrm{d}\rho} - I\frac{\mathrm{d}^2\psi}{\mathrm{d}\rho^2}\right) \tag{187}$$

$$\frac{\mathrm{d}\beta}{\mathrm{d}\rho} = \oiint \mathrm{d}\theta\mathrm{d}\zeta \frac{\partial}{\partial\rho}\left(\frac{\frac{\partial\lambda}{\partial\zeta} g_{\theta\theta} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) g_{\theta\zeta}}{\sqrt{g}}\right) \tag{188}$$

$$= \oiint \mathrm{d}\theta\mathrm{d}\zeta \left(\frac{\frac{\partial^2\lambda}{\partial\rho\partial\zeta} g_{\theta\theta} + \frac{\partial\lambda}{\partial\zeta}\frac{\partial g_{\theta\theta}}{\partial\rho} - \frac{\partial^2\lambda}{\partial\rho\partial\theta} g_{\theta\zeta} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right)\frac{\partial g_{\theta\zeta}}{\partial\rho}}{\sqrt{g}} - \frac{\frac{\partial\lambda}{\partial\zeta} g_{\theta\theta} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) g_{\theta\zeta}}{\sqrt{g}^2}\frac{\partial\sqrt{g}}{\partial\rho}\right) \tag{189}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}\rho} = \oiint \mathrm{d}\theta\mathrm{d}\zeta \frac{\partial}{\partial\rho}\left(\frac{g_{\theta\theta}}{\sqrt{g}}\right) \tag{190}$$

$$= \oiint \mathrm{d}\theta\mathrm{d}\zeta \left(\frac{\frac{\partial g_{\theta\theta}}{\partial\rho}\sqrt{g} - g_{\theta\theta}\frac{\partial\sqrt{g}}{\partial\rho}}{\sqrt{g}^2}\right) \tag{191}$$

Now, if each of the following limits exist independently, and the limit in the denominator is nonzero,[4] then the following relation holds by the algebraic limit theorem.

$$\lim_{\rho\to 0} \frac{\mathrm{d}(\alpha+\beta)/\mathrm{d}\rho}{\mathrm{d}\gamma/\mathrm{d}\rho} = \frac{\lim_{\rho\to 0}\mathrm{d}\alpha/\mathrm{d}\rho + \lim_{\rho\to 0}\mathrm{d}\beta/\mathrm{d}\rho}{\lim_{\rho\to 0}\mathrm{d}\gamma/\mathrm{d}\rho} \tag{192}$$

Apply l'Hôpital's rule twice to the derivative of the toroidal current supplement term.

$$\lim_{\rho\to 0} \frac{\mathrm{d}\alpha}{\mathrm{d}\rho} = \frac{2\pi^2}{(\mathrm{d}^2\psi/\mathrm{d}\rho^2)^2}\left(\frac{\mathrm{d}^2 I}{\mathrm{d}\rho^2}\frac{\mathrm{d}^2\psi}{\mathrm{d}\rho^2} - \frac{\mathrm{d}I}{\mathrm{d}\rho}\frac{\mathrm{d}^3\psi}{\mathrm{d}\rho^3}\right) \tag{193}$$

Define μ and ν to be the numerator of the integrand in (189) and (191), respectively. Continue with the assumption that the order of limit and integral operations can be interchanged.

$$\lim_{\rho\to 0} \frac{\mathrm{d}\beta}{\mathrm{d}\rho} = \oiint \mathrm{d}\theta\mathrm{d}\zeta \lim_{\rho\to 0} \frac{\mu}{\sqrt{g}^2} \tag{194}$$

$$\lim_{\rho\to 0} \frac{\mathrm{d}\gamma}{\mathrm{d}\rho} = \oiint \mathrm{d}\theta\mathrm{d}\zeta \lim_{\rho\to 0} \frac{\nu}{\sqrt{g}^2} \tag{195}$$

computation shows that the limit we derive here appears to be the continuous extension of (178) to the $\rho = 0$ axis. Since ι is a physical quantity, we should expect ι to be continuous over physical domains. This requires the limit to be a continuous extension.

[4]We will later discover this precondition is true.

Both limits are of indeterminate form 0/0. Recall that l'Hôpital's rule relies on Cauchy's generalized mean value theorem, which is a property proven unique to functions that map $R \to R$. Fortunately, some parts of l'Hôpital's rule have recently been generalized to multivariable functions that map $R^n \to R$ where $n \in N$. Since the singularity is non-isolated and the zero set of the numerator contains the zero set of the denominator inside some small tube surrounding the magnetic axis, we may apply theorem 4, case 1 of the multivariable form of l'Hôpital's rule.

$$\lim_{\rho\to 0} \frac{\mu}{\sqrt{g}^2} = \lim_{\rho\to 0} \frac{\partial\mu/\partial\rho}{\partial\sqrt{g}^2/\partial\rho} = \lim_{\rho\to 0} \frac{\partial^2\mu/\partial\rho^2}{\partial^2\sqrt{g}^2/\partial\rho^2} \tag{196}$$

$$\lim_{\rho\to 0} \frac{\nu}{\sqrt{g}^2} = \lim_{\rho\to 0} \frac{\partial\nu/\partial\rho}{\partial\sqrt{g}^2/\partial\rho} = \lim_{\rho\to 0} \frac{\partial^2\nu/\partial\rho^2}{\partial^2\sqrt{g}^2/\partial\rho^2} \tag{197}$$

The limit of the denominator is

$$\frac{\partial^2\sqrt{g}^2}{\partial\rho^2} = 2\left(\frac{\partial\sqrt{g}}{\partial\rho}\right)^2 + 2\sqrt{g}\frac{\partial^2\sqrt{g}}{\partial\rho^2} \tag{198}$$

$$\lim_{\rho\to 0} \frac{\partial^2\sqrt{g}^2}{\partial\rho^2} = 2\left(\frac{\partial\sqrt{g}}{\partial\rho}\right)^2 \neq 0 \tag{199}$$

Since the above limit is nonzero, we may compute each of the limits $\partial^2\mu/\partial\rho^2$ and $\partial^2\nu/\partial\rho^2$ independent of it.

$$\begin{aligned}
\frac{\partial^2\mu}{\partial\rho^2} = &-\frac{\partial^3\sqrt{g}}{\partial\rho^3}\frac{\partial\lambda}{\partial\zeta}g_{\theta\theta} + \frac{\partial^3\sqrt{g}}{\partial\rho^3}g_{\theta\zeta}\frac{\partial\lambda}{\partial\theta} + \frac{\partial^3\sqrt{g}}{\partial\rho^3}g_{\theta\zeta} \\
&- \frac{\partial\lambda}{\partial\zeta}\frac{\partial^2\sqrt{g}}{\partial\rho^2}\frac{\partial g_{\theta\theta}}{\partial\rho} - g_{\theta\theta}\frac{\partial^2\sqrt{g}}{\partial\rho^2}\frac{\partial^2\lambda}{\partial\rho\partial\zeta} + g_{\theta\zeta}\frac{\partial^2\sqrt{g}}{\partial\rho^2}\frac{\partial^2\lambda}{\partial\rho\partial\theta} \\
&+ \frac{\partial g_{\theta\zeta}}{\partial\rho}\left(\frac{\partial^2\sqrt{g}}{\partial\rho^2}\left(1+\frac{\partial\lambda}{\partial\theta}\right) - 3\sqrt{g}\frac{\partial^3\lambda}{\partial\rho^2\partial\theta}\right) \\
&+ \frac{\partial\sqrt{g}}{\partial\rho}\left(2\frac{\partial^2\lambda}{\partial\rho\partial\zeta}\frac{\partial g_{\theta\theta}}{\partial\rho} + \frac{\partial\lambda}{\partial\zeta}\frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} + g_{\theta\theta}\frac{\partial^3\lambda}{\partial\rho^2\partial\zeta}\right. \\
&\left. -2\frac{\partial g_{\theta\zeta}}{\partial\rho}\frac{\partial^2\lambda}{\partial\rho\partial\theta} - \left(1+\frac{\partial\lambda}{\partial\theta}\right)\frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2} - g_{\theta\zeta}\frac{\partial^3\lambda}{\partial\rho^2\partial\theta}\right) \\
&+ 3\sqrt{g}\frac{\partial^2\lambda}{\partial\rho\partial\zeta}\frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} + 3\sqrt{g}\frac{\partial^3\lambda}{\partial\rho^2\partial\zeta}\frac{\partial g_{\theta\theta}}{\partial\rho} + \sqrt{g}\frac{\partial\lambda}{\partial\zeta}\frac{\partial^3 g_{\theta\theta}}{\partial\rho^3} \\
&+ \sqrt{g}\frac{\partial^4\lambda}{\partial\rho^3\partial\zeta}g_{\theta\theta} - 3\sqrt{g}\frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}\frac{\partial^2\lambda}{\partial\rho\partial\theta} - \sqrt{g}\frac{\partial^3 g_{\theta\zeta}}{\partial\rho^3}\frac{\partial\lambda}{\partial\theta} \\
&- \sqrt{g}g_{\theta\zeta}\frac{\partial^4\lambda}{\partial\rho^3\partial\theta} - \sqrt{g}\frac{\partial^3 g_{\theta\zeta}}{\partial\rho^3}
\end{aligned} \tag{200}$$

$$\frac{\partial^2\nu}{\partial\rho^2} = \sqrt{g}\frac{\partial^3 g_{\theta\theta}}{\partial\rho^3} + \frac{\partial\sqrt{g}}{\partial\rho}\frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - \frac{\partial^2\sqrt{g}}{\partial\rho^2}\frac{\partial g_{\theta\theta}}{\partial\rho} - \frac{\partial^3\sqrt{g}}{\partial\rho^3}g_{\theta\theta} \tag{201}$$

Take the limits.

$$\begin{aligned}
\lim_{\rho\to 0}\frac{\partial^2\mu}{\partial\rho^2} = &\left(1+\frac{\partial\lambda}{\partial\theta}\right)\frac{\partial g_{\theta\zeta}}{\partial\rho}\frac{\partial^2\sqrt{g}}{\partial\rho^2} \\
&+ \left(\frac{\partial\lambda}{\partial\zeta}\frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - 2\frac{\partial^2\lambda}{\partial\rho\partial\theta}\frac{\partial g_{\theta\zeta}}{\partial\rho} - \left(1+\frac{\partial\lambda}{\partial\theta}\right)\frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}\right)\frac{\partial\sqrt{g}}{\partial\rho}
\end{aligned} \tag{202}$$

$$\lim_{\rho\to 0}\frac{\partial^2\nu}{\partial\rho^2} = \frac{\partial^2 g_{\theta\theta}}{\partial\rho^2}\frac{\partial\sqrt{g}}{\partial\rho} \tag{203}$$

Observe that $\lim_{\rho \to 0} \mathrm{d}\gamma / \mathrm{d}\rho \neq 0$ as required by (192).

$$\lim_{\rho \to 0} \frac{\nu}{\sqrt{g}^2} = \frac{\partial^2 g_{\theta\theta} / \partial\rho^2}{2 \partial \sqrt{g} / \partial\rho} \neq 0 \tag{204}$$

Collecting our results, we conclude:

$$\lim_{\rho \to 0} \iota = \frac{4\pi^2 \dfrac{\frac{\mathrm{d}^2 I}{\mathrm{d}\rho^2} \frac{\mathrm{d}^2 \psi}{\mathrm{d}\rho^2} - \frac{\mathrm{d} I}{\mathrm{d}\rho} \frac{\mathrm{d}^3 \psi}{\mathrm{d}\rho^3}}{(\mathrm{d}^2\psi/\mathrm{d}\rho^2)^2} + \oiint \mathrm{d}\theta \mathrm{d}\zeta \left(\dfrac{\left(1 + \frac{\partial\lambda}{\partial\theta}\right) \frac{\partial g_{\theta\zeta}}{\partial\rho} \frac{\partial^2 \sqrt{g}}{\partial\rho^2}}{(\partial\sqrt{g}/\partial\rho)^2} + \dfrac{\frac{\partial\lambda}{\partial\zeta} \frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - 2 \frac{\partial^2\lambda}{\partial\rho\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) \frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}}{\partial\sqrt{g}/\partial\rho} \right)}{\oiint \mathrm{d}\theta \mathrm{d}\zeta \dfrac{\partial^2 g_{\theta\theta}/\partial\rho^2}{\partial\sqrt{g}/\partial\rho}} \tag{205}$$

Written with surface averages instead of integrals

$$\lim_{\rho \to 0} \iota = \frac{\dfrac{\frac{\mathrm{d}^2 I}{\mathrm{d}\rho^2} \frac{\mathrm{d}^2 \psi}{\mathrm{d}\rho^2} - \frac{\mathrm{d} I}{\mathrm{d}\rho} \frac{\mathrm{d}^3 \psi}{\mathrm{d}\rho^3}}{(\mathrm{d}^2\psi/\mathrm{d}\rho^2)^2} + \left\langle \dfrac{\left(1 + \frac{\partial\lambda}{\partial\theta}\right) \frac{\partial g_{\theta\zeta}}{\partial\rho} \frac{\partial^2 \sqrt{g}}{\partial\rho^2}}{(\partial\sqrt{g}/\partial\rho)^2} + \dfrac{\frac{\partial\lambda}{\partial\zeta} \frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - 2 \frac{\partial^2\lambda}{\partial\rho\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) \frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}}{\partial\sqrt{g}/\partial\rho} \right\rangle}{\left\langle \dfrac{\partial^2 g_{\theta\theta}/\partial\rho^2}{\partial\sqrt{g}/\partial\rho} \right\rangle} \tag{206}$$

4.3.5 Rotational transform with toroidal stream function

$$\boxed{\iota = \frac{4\pi^2 \dfrac{I}{\mathrm{d}\psi/\mathrm{d}\rho} + \oiint \mathrm{d}\theta \mathrm{d}\zeta \dfrac{\frac{\partial\lambda}{\partial\zeta} g_{\theta\theta} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) g_{\theta\zeta}}{\sqrt{g}}}{\oiint \mathrm{d}\theta \mathrm{d}\zeta \dfrac{(1 + \frac{\partial\omega}{\partial\zeta}) g_{\theta\theta} - \frac{\partial\omega}{\partial\theta} g_{\theta\zeta}}{\sqrt{g}}}} \tag{207}$$

The behavior of $\partial\omega/\partial\theta$ near the magnetic axis should ensure that γ tends to zero. The quantity $\mathrm{d}\gamma/\mathrm{d}\rho$ in (191) becomes

$$\frac{\mathrm{d}\gamma}{\mathrm{d}\rho} = \oiint \mathrm{d}\theta \mathrm{d}\zeta \left(\frac{\frac{\partial^2\omega}{\partial\rho\partial\zeta} g_{\theta\theta} + \left(1 + \frac{\partial\omega}{\partial\zeta}\right) \frac{\partial g_{\theta\theta}}{\partial\rho} - \frac{\partial^2\omega}{\partial\rho\partial\theta} g_{\theta\zeta} - \frac{\partial\omega}{\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho}}{\sqrt{g}} - \frac{\left(1 + \frac{\partial\omega}{\partial\zeta}\right) g_{\theta\theta} - \frac{\partial\omega}{\partial\theta} g_{\theta\zeta}}{\sqrt{g}^2} \frac{\partial\sqrt{g}}{\partial\rho} \right) \tag{208}$$

Then (197) again requires multiple applications of the $R^n \to R$ l'Hôpital's rule to evaluate. Equation (204) becomes

$$\lim_{\rho \to 0} \frac{\nu}{\sqrt{g}^2} = \frac{\frac{\partial\omega}{\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} \frac{\partial^2\sqrt{g}}{\partial\rho^2}}{2(\partial\sqrt{g}/\partial\rho)^2} + \frac{\left(1 + \frac{\partial\omega}{\partial\zeta}\right) \frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - 2 \frac{\partial^2\omega}{\partial\rho\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} - \frac{\partial\omega}{\partial\theta} \frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}}{2\partial\sqrt{g}/\partial\rho} \neq 0 \tag{209}$$

Then the rotational transform at the magnetic axis is given by

$$\lim_{\rho \to 0} \iota = \frac{4\pi^2 \dfrac{\frac{\mathrm{d}^2 I}{\mathrm{d}\rho^2} \frac{\mathrm{d}^2 \psi}{\mathrm{d}\rho^2} - \frac{\mathrm{d} I}{\mathrm{d}\rho} \frac{\mathrm{d}^3 \psi}{\mathrm{d}\rho^3}}{(\mathrm{d}^2\psi/\mathrm{d}\rho^2)^2} + \oiint \mathrm{d}\theta \mathrm{d}\zeta \left(\dfrac{\left(1 + \frac{\partial\lambda}{\partial\theta}\right) \frac{\partial g_{\theta\zeta}}{\partial\rho} \frac{\partial^2 \sqrt{g}}{\partial\rho^2}}{(\partial\sqrt{g}/\partial\rho)^2} + \dfrac{\frac{\partial\lambda}{\partial\zeta} \frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - 2 \frac{\partial^2\lambda}{\partial\rho\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} - \left(1 + \frac{\partial\lambda}{\partial\theta}\right) \frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}}{\partial\sqrt{g}/\partial\rho} \right)}{\oiint \mathrm{d}\theta \mathrm{d}\zeta \left(\dfrac{\frac{\partial\omega}{\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} \frac{\partial^2\sqrt{g}}{\partial\rho^2}}{(\partial\sqrt{g}/\partial\rho)^2} + \dfrac{\left(1 + \frac{\partial\omega}{\partial\zeta}\right) \frac{\partial^2 g_{\theta\theta}}{\partial\rho^2} - 2 \frac{\partial^2\omega}{\partial\rho\partial\theta} \frac{\partial g_{\theta\zeta}}{\partial\rho} - \frac{\partial\omega}{\partial\theta} \frac{\partial^2 g_{\theta\zeta}}{\partial\rho^2}}{\partial\sqrt{g}/\partial\rho} \right)} \tag{210}$$

Setting ω and its derivatives to zero recovers the results without the toroidal stream function.

4.3.6 Axis limit of rotational transform derivatives

The limits of the derivatives require similar work to compute. Recall that as $\rho \to 0$ so do $\alpha + \beta$ and γ. For any $x \in \{\alpha + \beta, \gamma\}$, $\mathrm{d}^n x / \mathrm{d}\rho^n$ is finite and nonzero for all $n \in N$.[5] Previously, we showed

$$\iota = \frac{\alpha + \beta}{\gamma} \tag{211}$$

$$\lim_{\rho \to 0} \iota = \lim_{\rho \to 0} \frac{\mathrm{d}(\alpha + \beta)/\mathrm{d}\rho}{\mathrm{d}\gamma/\mathrm{d}\rho} \tag{212}$$

Now, the first derivative breaks from the 0/0 indeterminate form after two applications of the $R \to R$ l'Hôpital's rule.

$$\frac{\mathrm{d}\iota}{\mathrm{d}\rho} = \frac{\frac{\mathrm{d}(\alpha+\beta)}{\mathrm{d}\rho}\gamma - (\alpha + \beta)\frac{\mathrm{d}\gamma}{\mathrm{d}\rho}}{\gamma^2} \tag{213}$$

$$\lim_{\rho \to 0} \frac{\mathrm{d}\iota}{\mathrm{d}\rho} = \lim_{\rho \to 0} \frac{\frac{\mathrm{d}^2(\alpha+\beta)}{\mathrm{d}\rho^2}\gamma - (\alpha + \beta)\frac{\mathrm{d}^2\gamma}{\mathrm{d}\rho^2}}{2\gamma(\mathrm{d}\gamma/\mathrm{d}\rho)} \tag{214}$$

$$= \lim_{\rho \to 0} \frac{\frac{\mathrm{d}^3(\alpha+\beta)}{\mathrm{d}\rho^3}\gamma + \frac{\mathrm{d}^2(\alpha+\beta)}{\mathrm{d}\rho^2}\frac{\mathrm{d}\gamma}{\mathrm{d}\rho} - \frac{\mathrm{d}(\alpha+\beta)}{\mathrm{d}\rho}\frac{\mathrm{d}^2\gamma}{\mathrm{d}\rho^2} - (\alpha + \beta)\frac{\mathrm{d}^3\gamma}{\mathrm{d}\rho^3}}{2(\mathrm{d}\gamma/\mathrm{d}\rho)^2 + 2\gamma(\mathrm{d}^2\gamma/\mathrm{d}\rho^2)} \tag{215}$$

$$= \lim_{\rho \to 0} \frac{\frac{\mathrm{d}^2(\alpha+\beta)}{\mathrm{d}\rho^2}\frac{\mathrm{d}\gamma}{\mathrm{d}\rho} - \frac{\mathrm{d}(\alpha+\beta)}{\mathrm{d}\rho}\frac{\mathrm{d}^2\gamma}{\mathrm{d}\rho^2}}{2(\mathrm{d}\gamma/\mathrm{d}\rho)^2} \tag{216}$$

The second derivative eventually reduces to

$$\frac{\mathrm{d}^2\iota}{\mathrm{d}\rho^2} = \frac{\frac{\mathrm{d}^2(\alpha+\beta)}{\mathrm{d}\rho^2}\gamma^2 - 2\frac{\mathrm{d}(\alpha+\beta)}{\mathrm{d}\rho}\gamma\frac{\mathrm{d}\gamma}{\mathrm{d}\rho} + 2(\alpha + \beta)\left(\frac{\mathrm{d}\gamma}{\mathrm{d}\rho}\right)^2 - (\alpha + \beta)\gamma\frac{\mathrm{d}^2\gamma}{\mathrm{d}\rho^2}}{\gamma^3} \tag{217}$$

$$\lim_{\rho \to 0} \frac{\mathrm{d}^2\iota}{\mathrm{d}\rho^2} = \lim_{\rho \to 0} \frac{2\frac{\mathrm{d}^3(\alpha+\beta)}{\mathrm{d}\rho^3}\left(\frac{\mathrm{d}\gamma}{\mathrm{d}\rho}\right)^2 - 3\frac{\mathrm{d}^2(\alpha+\beta)}{\mathrm{d}\rho^2}\frac{\mathrm{d}\gamma}{\mathrm{d}\rho}\frac{\mathrm{d}^2\gamma}{\mathrm{d}\rho^2} + 3\frac{\mathrm{d}(\alpha+\beta)}{\mathrm{d}\rho}\left(\frac{\mathrm{d}^2\gamma}{\mathrm{d}\rho^2}\right)^2 - 2\frac{\mathrm{d}(\alpha+\beta)}{\mathrm{d}\rho}\frac{\mathrm{d}\gamma}{\mathrm{d}\rho}\frac{\mathrm{d}^3\gamma}{\mathrm{d}\rho^3}}{6(\mathrm{d}\gamma/\mathrm{d}\rho)^3} \tag{218}$$

[5] Showing this requires $n + 1$ applications of the $R^m \to R$ l'Hôpital's rule.

5 Optimization

5.1 Overview of *lsq-exact* Optimizer

In DESC, we solve the inverse equilibrium problem using the pseudo-spectral method. This means, for equilibrium solve, our objective is to find spectral coefficients $\mathcal{R}_{lmn}$, $\mathcal{Z}_{lmn}$ and λ_{lmn}, such that $\mathbf{J}\times\mathbf{B}-\nabla p=0$ is satisfied.

Here, we will go over the fixed-boundary solution process. Once we initialize the equilibrium object, to solve for equilibrium, we need to specify constraints and objectives. In DESC by default, every spectral coefficient is optimizable, so during the optimization process, if nothing is given to the optimizer, it will change any parameter to satisfy the objective. However, this is not what we want. For example, we want the last closed flux surface shape to stay the same. The full state vector in DESC includes basic coefficients like $\mathcal{R}_{lmn}$ but also we have Rb_{mn} which are coefficients of the LCFS that is defined by double Fourier series that are also optimizable by default. To fix the boundary parameters, we have the constraints **FixBoundaryR** and **FixBoundaryZ** that can fix every mode or some user specified modes. These constraints convert Rb_{mn} and Zb_{mn} from optimizable parameter to fixed parameter. The actual constraints that satisfy the boundary is what the user specified it to be are the self-consistency constraints, namely **BoundaryRSelfConsistency** and **BoundaryZSelfConsistency**. These constraints force the boundary shape evaluated at $\rho=1$ to be equal to Rb_{mn} values. Since the radial part of Zernike polynomials evaluated at $\rho=1$ is always 1, the self-consistency constraint ends up being,

$$\sum_l\left(\sum_m\sum_n\mathcal{R}_{lmn}cos(m\theta)cos(n\zeta)\right)=\sum_m\sum_n Rb_{mn}cos(m\theta)cos(n\zeta)$$

$$0Rb_{mn}=\sum_l\mathcal{R}_{lmn}$$

where m and n values on both sides must be equal.

The objectives usually cannot be described as linear operations (at least in DESC). For example, force balance objective calculates $f(\vec{x})=\mathbf{J}\times\mathbf{B}-\nabla p$ at each grid point, and defines the cost as $F(\vec{x})=||f(\vec{x})-0||_2^2$. Due to nonlinearities, we should solve this problem not analytically but through an optimization such that,

$$\vec{x}=\arg\min_{\vec{x}}F(\vec{x})\qquad\text{such that }\mathbf{A}\vec{x}=\vec{b}\tag{219}$$

However, due to the constraints on $\vec{x}$, this is inconvenient to implement using the existing optimizers. What we can do is transform this problem into an unconstrained optimization problem in $\vec{y}$ with the following steps. Assume we have only linear constraints (for fixed boundary force-balance problem this is the case), then one can write it in matrix form,

$$\begin{aligned}\mathbf{A}\vec{x}&=\vec{b}\\ \vec{x}&=\vec{x}_{particular}+\vec{x}_{homogenous}\\ \mathbf{A}\vec{x}_{particular}&=\vec{b}\\ \mathbf{A}\vec{x}_{homogenous}&=0\\ \vec{x}_{homogenous}&=\mathbf{Z}\vec{y}\qquad\text{where }\mathbf{Z}\text{ is the nullspace of }\mathbf{A}\end{aligned}$$

$\mathbf{Z}$ can be elaborated as such,

$$\mathbf{A}\vec{x}_i=0$$

$$\mathbf{Z} = \begin{bmatrix} \vdots & \vdots & \vdots & \dots & \vdots \\ \vec{x}_1 & \vec{x}_2 & \vec{x}_3 & \dots & \vec{x}_N \\ \vdots & \vdots & \vdots & \dots & \vdots \end{bmatrix}$$

$$\mathbf{AZ} = 0$$

$\mathbf{Z}$ is orthogonal matrix such that $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}$, then,

$$\begin{aligned} \vec{x}_{particular} &= \mathbf{A}^{-1}\vec{b} \\ \vec{x}_{homogeneous} &= (\vec{x} - \vec{x}_{particular}) \\ \vec{y} &= \mathbf{Z}^T(\vec{x} - \vec{x}_{particular}) \end{aligned}$$

Notice that the dimensions of $\vec{x}$ and $\vec{y}$ don't need to be the same, actually $\vec{y}$ is smaller than $\vec{x}$, that's why we refer to it as $\vec{x}_{reduced}$, too. Also, we slightly abused the notation by writing $\mathbf{A}^{-1}$, because in general $\mathbf{A}$ is not square, so here $\mathbf{A}^{-1}$ is the left inverse. Thereby, we successfully converted the optimization problem into an unconstrained one, since $\vec{y}$ is unconstrained.

$$\vec{y} = \arg\min_{\vec{y}} ||F(\vec{y})||_2^2 \tag{220}$$

In extttDESC, the user can go back and forth between reduced and full state vectors using the **project** and **recover** functions. The former projects the full state vector to the reduced vector, whereas the latter recovers the full state vector from the reduced state vector.

Although the theory is concrete, in numerical methods, there are some concerns about executing these steps. First of all, taking the inverse of $\mathbf{A}$ is computationally intensive. Instead, first, we will try to reduce the size of $\mathbf{A}$ using the fixed parameter constraints, clean it up a bit, and take the inverse of the much smaller matrix. This is done in the **factorize_linear_constraints()** function. Some of the tricks to simplify $\mathbf{A}$ used in extttDESC can be found in Appendix **??**.

Now, we will focus on the objective function $F(\vec{y})$. Here, the input to this objective is the state vector, and the output is the force error at each grid point. Usually, we use double the number of modes for grid points, so the output vector size is twice the size of the input vector. Ideally, we want every element of the output to be 0. During optimization, we will use Newton's method to find the coefficient values that make the objective as close to 0 as possible. Newton's method is a gradient descent method. It is basically the first 2 terms of the Taylor expansion

$$F(y + \Delta y) = F(y) + F'(y)\Delta y + \mathcal{O}(y^2) \tag{221}$$

For numerical implementation, we will use this equation iteratively, and at each step, we will try to find Δy that minimizes $F(y + \Delta y) = 0$. Another naming convention is to call $F'(y)$ as the Jacobian $J(y)$, since for vector values y, the derivative will be the Jacobian matrix. So, we will write Equation 221 as,

$$F(y + \Delta y) = F(\vec{y}) + \mathbf{J}(\vec{y})\Delta\vec{y}$$

This is the subproblem of our optimization. We need to find the Δy for each step of the main optimization problem. Here is the subproblem,

$$\min_{\Delta y} ||F(y^k) + \mathbf{J}(y^k)\Delta y||_2^2 \tag{222}$$

We used y^k to show that during this subproblem, y^k has a set value. From now on, we will drop the argument from the Jacobian and the objective function, so $F(y^k) \to F$ and $\mathbf{J}(y^k) \to \mathbf{J}$.

The force-balance error function has an absolute value inside of it which makes the values strictly positive. Therefore, the minimum value for the above problem is 0. If we were to solve this problem for Δy by gradient descent (finding the Newton step), we need to take the inverse of $\mathbf{J}$ but it is usually not square. Therefore, we multiply both sides of $F + J\Delta y = 0$ by $\mathbf{J}^T$

$$\Delta\vec{y} = -\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T F$$

Although on paper this is the proper solution to this linear system, numerically there are some limitations that push us from implementing it. In spectral methods, the Jacobian matrix is usually ill-conditioned, which means the highest to lowest eigenvalue ratio is very high ($> 10^6$). Ill-conditioned matrices are sensitive to slight changes in values, and solution to $\mathbf{J}\vec{x} = b$ is sensitive if there is any numerical error such as floating point or round-off. When an ill-conditioned matrix is multiplied by its transpose the resulting matrix is square which is nice but has a condition number even higher than the original one. Therefore, instead of dealing with the $\left(\mathbf{J}^T\mathbf{J}\right)^{-1}$ term and getting extremely error-prone results, we will take the decomposition of the Jacobian. First, let's look at QR decomposition,

$$\begin{aligned}
\mathbf{J} &= \mathbf{QR} \qquad \text{where } \mathbf{Q}^T\mathbf{Q} = \mathbb{I}, \mathbf{Q} \in \mathbb{R}^{m\times n} \text{ and } \mathbf{R} \in \mathbb{R}^{n\times n} \\
\mathbf{QR}\Delta\vec{y} &= -F \\
\mathbf{Q}^T\mathbf{QR}\Delta\vec{y} &= -\mathbf{Q}^T F \\
\mathbf{R}\Delta\vec{y} &= -\mathbf{Q}^T F
\end{aligned}$$

The last equation can be easily solved using back-substitution since R is an upper triangular matrix.

For wide Jacobian,

$$\mathbf{J}^T = \mathbf{QR} \qquad \text{where } \mathbf{Q}^T\mathbf{Q} = \mathbb{I}, \mathbf{Q} \in \mathbb{R}^{n\times m} \text{ and } \mathbf{R} \in \mathbb{R}^{m\times m}$$
$$\mathbf{R}^T\mathbf{Q}^T\Delta\vec{y} = -F \rightarrow \mathbf{R}^T v = -F \rightarrow \text{(solve for v through forward-substitution)} \rightarrow \Delta y = \mathbf{Q}v$$

The above steps use a purely Newton step to reduce the cost function, which is not always the optimum choice. In extttDESC, we have implemented many different optimizers that use better methods than pure gradient descent. For the equilibrium solve, if the user doesn't specify an optimizer explicitly the default is "lsq-exact" which uses linearized least-squares with trust region. The optimization problem for that is,

$$\min_{\Delta y} ||F + \mathbf{J}\Delta y||_2^2$$
$$\text{subject to } ||\Delta y||_2 \leq r_{tr}$$

To do so, we add a regularization parameter α,

$$\min_{\Delta y} ||F + \mathbf{J}\Delta y||_2^2 + \alpha||\Delta y||_2^2$$
$$\text{subject to } ||\Delta y||_2 \leq r_{tr}$$

If we expand the objective function,

$$\begin{aligned}||F + \mathbf{J}\Delta y||_2^2 + \alpha||\Delta y||_2^2 \\ &= \Delta y^T \mathbf{J}^T \mathbf{J}\Delta y + 2F^T \mathbf{J}\Delta y + \alpha \Delta y^T \Delta y + F^T F \\ &= \Delta y^T (\mathbf{J}^T \mathbf{J} + \alpha \mathbb{I})\Delta y + 2F^T \mathbf{J}\Delta y + F^T F \\ &= \Delta y^T \begin{bmatrix} \mathbf{J} & \sqrt{\alpha}\mathbb{I} \end{bmatrix} \begin{bmatrix} \mathbf{J} \\ \sqrt{\alpha}\mathbb{I} \end{bmatrix} \Delta y + 2 \begin{bmatrix} F & 0 \end{bmatrix} \begin{bmatrix} \mathbf{J} \\ \sqrt{\alpha}\mathbb{I} \end{bmatrix} \Delta y + \begin{bmatrix} F & 0 \end{bmatrix} \begin{bmatrix} F \\ 0 \end{bmatrix}\end{aligned}$$

And this can be written in least-squares form,

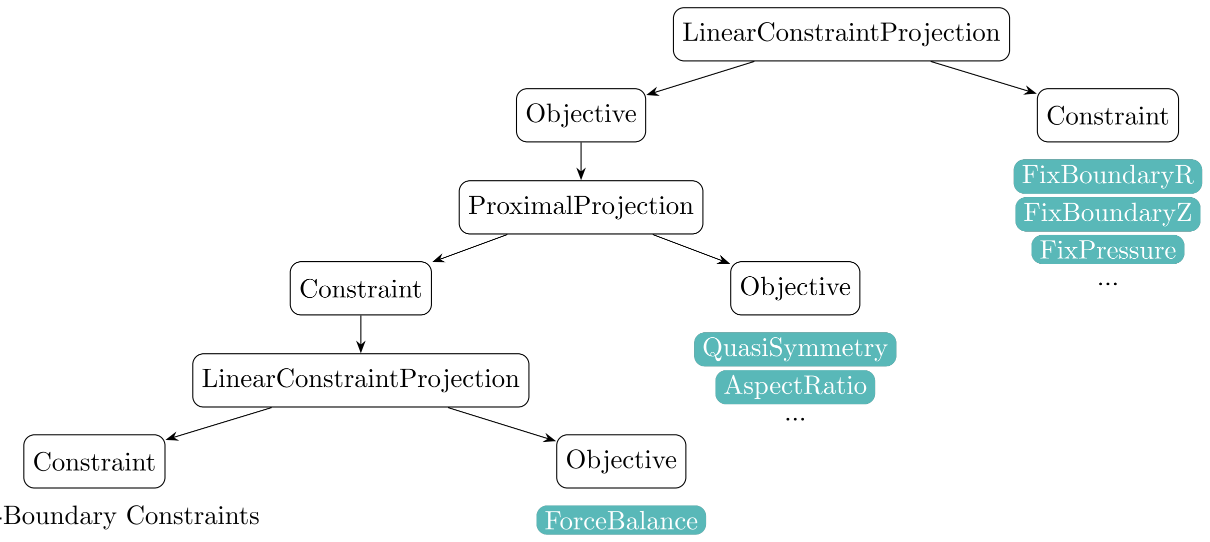
$$\left|\left| \begin{bmatrix} \mathbf{J} \\ \sqrt{\alpha}\mathbb{I} \end{bmatrix} \Delta y + \begin{bmatrix} F \\ 0 \end{bmatrix} \right|\right|_2^2 \tag{223}$$

And if we introduce augmented vector and matrix, F_a and $\mathbf{J}_a$, we can write the regularized trust-region subproblem as,

$$\min_{\Delta y} ||\mathbf{J}_a \Delta y + F_a||_2^2$$
$$\text{subject to } ||\Delta y||_2 < r_{tr}$$

5.2 Proximal Projection and Force Balance Constraint

In `DESC`, we deal with optimization problems of linearly and nonlinearly constrained kind. Among the non-linear constraints, ForceBalance has a special place. When there is an optimization problem with ForceBalance constraint, `DESC` internally wraps the objective and this non-linear constraint with a ProximalProjection objective class. This is then passed to the objective of LinearConstraintProjection.



This optimization has the mathematical notation,

$$\begin{aligned} \min_{c} \quad & g(\mathbf{x},\ \mathbf{c}) \\ \text{subject to} \quad & f(\mathbf{x},\ \mathbf{c}) = 0 \\ & \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{b} \end{bmatrix} \end{aligned} \tag{224}$$

where $\mathbf{x}$ consists of $R_{lmn}, Z_{lmn}, \lambda_{lmn}$, and $\mathbf{c}$ consists of the optimization parameters like Rb_{lmn}, Zb_{lmn} and other optimizable parameters from other *things* such as coils, fields, etc.

The important point to note here is that only Rb_{lmn}, Zb_{lmn} and profiles are included in $\mathbf{c}$ for the equilibrium. Rb_{lmn}, Zb_{lmn} have a simple linear dependence with R_{lmn}, Z_{lmn} through BoundaryRSelfConsistency and BoundaryZSelfConsistency, thus once you know the profiles, one can solve the equilibrium and get what should be $\mathbf{x}$ for those $\mathbf{c}$ values. We will use this dependence for taking the Jacobian of $g(\mathbf{x},\ \mathbf{c})$.

Let's consider the first order Taylor expansion of $g(\mathbf{x},\ \mathbf{c})$ and $f(\mathbf{x},\ \mathbf{c})$,

$$g(x + \Delta x, c + \Delta c) = g(x, c) + \frac{\partial g}{\partial x}\Delta x + \frac{\partial g}{\partial c}\Delta c \tag{225}$$

$$f(x + \Delta x, c + \Delta c) = f(x, c) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial c}\Delta c \tag{226}$$

In Equation 226, since we were in force balance before, and we solved the equilibrium, hence the update is also in force balance, we can remove the first 2 terms. This gives us a relation between Δx and Δc. But note that, this relation only uses the first-order terms and expects a very good force balance; some of the inaccuracies of the ProximalProjection stem from this fact.

$$\Delta x = -\left(\frac{\partial f}{\partial x}\right)^{-}\frac{\partial f}{\partial c}\Delta c \tag{227}$$

Now, we can substitute it in 225,

$$g(x + \Delta x, c + \Delta c) = g(x, c) + \left[\frac{\partial g}{\partial c} - \frac{\partial g}{\partial x}\left(\frac{\partial f}{\partial x}\right)^{-}\frac{\partial f}{\partial c}\right]\Delta c \tag{228}$$

For each step of the optimization, we linearize the objective function as 228, and try to minimize the next value by trust region (the default optimizer *lsqtr* uses this method),

$$\begin{gathered} \min_{\Delta c} ||g + \mathbf{J}\Delta c||_2^2 + \alpha||\Delta c||_2^2 \\ \text{subject to } ||\Delta c||_2 \leq r_{tr} \end{gathered}$$

as explained before for equilibrium. Note that, apart from the calculation of the Jacobian, the optimization process is the same as equilibrium solve or other multi-objective linearly constrained problems, that is why `DESC` doesn't have a dedicated optimizer for this problem, but only a constraint wrapper that executes additional steps for the Jacobian calculation, like updating the equilibrium and taking derivative of the ForceBalance constraint etc.

The above notation is relatively simple, but once we try to code it, there are couple of challenges that make it hard to grasp at first.

- Compute functions never use Rb_{lmn}, Zb_{lmn}. So, in JAX tracing sense, their derivative is always 0. We need to account the self-consistency relations to take the proper derivatives with respect to these parameters.

- In general, $g(\mathbf{x},\ \mathbf{c})$ have multiple *things* that it optimizes, such as Equilibrium, Coils etc. Each sub-objective of $g(\mathbf{x},\ \mathbf{c})$ has a dedicated *thing* to compute the cost. We need to be able to split the state vector and pass them to the correct objectives.
- Due to trust-region, we may sometimes try same set of optimizable parameters multiple times, we should ideally store the combinations and omit doing same computations.

Our journey of weirdness starts now! Hopefully once we document what is going on here, we will be able to simplify it later!

5.2.1 Creation of the wrapper

The first instance of ProximalProjection is in the _maybe_wrap_nonlinear_constraints() where we take in the constraints and if there is ForceBalance, we create a wrapper ProximalProjection. Another difference is that we don't add the self-consistency constraints if we are using proximal. Then, ProximalProjection is passed to LinearConstraintProjection. Normally, factorize_linear_constraints() computes the null-space for the whole set of optimizable parameters, but for ProximalProjection, we do something different and take some of the parameters out of the optimization, these are $[R_{lmn}, Z_{lmn}, \lambda_{lmn}, Ra_n, Za_n]$ (I guess since we don't add self-consistency stuff, those columns are already 0, anyway). ProximalProjection compared to other ObjectiveFunction classes doesn't include $[R_{lmn}, Z_{lmn}, \lambda_{lmn}, Ra_n, Za_n]$ in the state vector, so the dim_x attribute of ProximalProjection is not the usual state vector size, but the size without these variables.

5.2.2 Build Method

The build() method, as usual creates the constant arrays, and some useful objects for the class. The summary of the operations are,

- Create and build LinearConstraintProjection for the internal fixed-boundary Equilibrium solves
- Create unique list of *things* (Equilibrium, Coil, Surface etc) and form _things_per_objective_idx that stores which *thing* belongs to which sub-objective
- Set _eq_idx to the index of Equilibrium in the list of *things*
- Form the required transformation, _dxdc, in _set_eq_state_vector() from optimizable parameters Rb_{lmn}, Zb_{lmn} and profiles of Equilibrium to full state vector of the Equilibrium with Rb_{lmn}, Zb_{lmn} set to 0
- Form the feasible tangent directions matrix, _feasible_tangents used to be called _unfixed_idx_mat. This matrix, when multiplied by the state vector that consists reduced state vector of the Equilibrium and all the optimizable parameters of other *things*, will return the full state vector with all parameters of each *thing*
- Create history arrays

5.2.3 _set_eq_state_vector()

The pure purpose of this function is to create _dxdc that helps us to use Rb_{lmn}, Zb_{lmn} as optimization parameters but still be able to take proper derivatives. As explained above, our compute functions never use Rb_{lmn} and Zb_{lmn}, so, their derivative is 0. But we know that changing Rb_{lmn}, Zb_{lmn} is equivalent to changing R_{lmn}, Z_{lmn}. Actually, Rb_{lmn}, Zb_{lmn} are dummy variables we introduced, because they are some linear combination of R_{lmn}, Z_{lmn}. Let's see an example where we have Rb_{lmn}, Zb_{lmn}, p_l and ι_l as optimization parameters,

$$dxdc@\begin{bmatrix} Rb_{lmn} \\ Zb_{lmn} \\ p_l \\ \iota_l \end{bmatrix} \rightarrow \begin{bmatrix} R_{lmn} \\ Z_{lmn} \\ \textbf{zeros\_like}(\lambda_{lmn}) \\ \textbf{zeros\_like}(Rb_{lmn}) \\ \textbf{zeros\_like}(Zb_{lmn}) \\ p_l \\ \iota_l \end{bmatrix} \tag{229}$$

To do so, we will use the self-consistency relation defined in BoundaryRSelfConsistency constraint. We can obtain

$$A_R R_{lmn} = Rb_{lmn} \tag{230}$$

We need the other way around, so, we will take the pseudo-inverse

$$R_{lmn} = A_R^{\dagger} Rb_{lmn} \tag{231}$$

We do the same for Zb_{lmn}. Since profiles don't have straight relation, we will use the identity matrix. So, the matrix _dxdc has the form,

$$dxdc = \begin{bmatrix} A_R^{\dagger} & 0 & 0 \\ 0 & A_Z^{\dagger} & 0 \\ 0 & 0 & \mathbb{I} \end{bmatrix} \tag{232}$$

5.2.4 _feasible_tangents

Let's consider that our *things* are Equilibrium, Coil, and Surface, in this order. The structure of the _feasible_tangents is,

$$\_feasible\_tangents = \begin{bmatrix} D@Z & 0 & 0 \\ 0 & \mathbb{I} & 0 \\ 0 & 0 & \mathbb{I} \end{bmatrix} \tag{233}$$

where D stores the scaling coefficients and Z is the null-space of the fixed-boundary problem such that $AZy = 0$ with y reduced state vector. Given that we find the particular solution, Zy will always satisfy the constraints and hence we try to find y that minimizes the objective.

Multiplication of _feasible_tangents with the state vector where all except Equilibrium have full state vectors, but Equilibrium has the reduced state vector (not just Rb_{lmn}, Zb_{lmn} but the actual reduced state vector from null-space, y) looks like,

$$\begin{bmatrix} D@Z & 0 & 0 \\ 0 & \mathbb{I} & 0 \\ 0 & 0 & \mathbb{I} \end{bmatrix} \begin{bmatrix} y_{eq} \\ x_{coil} \\ x_{surf} \end{bmatrix} = \begin{bmatrix} x_{eq} \\ x_{coil} \\ x_{surf} \end{bmatrix} \tag{234}$$

5.2.5 Derivatives

Again, we are aiming to compute the following Jacobian,

$$\frac{dg}{dc} = \frac{\partial g}{\partial c} - \frac{\partial g}{\partial x}\left(\frac{\partial f}{\partial x}\right)^{-}\frac{\partial f}{\partial c} \tag{235}$$

This can be thought of in JVPs. Let's assume we know the full set of derivatives of the cost function with respect to every parameter and call it J. Then, the above derivative can be computed by a couple of JVPs in proper tangent directions. In _jvp, we will,

1. Update the Equilibrium to be in force balance
2. Find the proper tangent direction
3. Take the Jacobian vector product (JVP)

Note that this doesn't include evaluating the full Jacobian!

5.2.6 _update_equilibrium()

The optimizer only knows about y values (optimization variable after linear constraints are applied and the space is reduced). However, before calculating the Jacobian or the cost, LinearConstraintProjection recovers the full state vector. For non-proximal objective functions, this is literally the full state vector. But for proximal, the recover() step only returns the Rb_{lmn}, Zb_{lmn} and profiles of the Equilibrium as well as whole state vectors of other *things*. This means $\mathbf{x}$ of ProximalProjection, the arguments to its compute_() and jac_() methods is,

$$\begin{bmatrix} \begin{bmatrix} Rb_{lmn} \\ Zb_{lmn} \\ p_l \\ \iota_l \end{bmatrix} \\ x_{coil} \\ x_{surf} \end{bmatrix} \tag{236}$$

_update_equilibrium() method takes in this vector $\mathbf{x}$, and does following,

- Checks if we have seen this vector before
- If not, it perturbs the equilibrium using the difference from the previous vector. Then, the equilibrium is resolved for force balance.
- Takes the full state vector of the Equilibrium stores as xeq, and the full state vector of the whole *things* stores as xopt.

_get_tangent()
This function is vectorized, so it will take in a single 1D array and return the proper tangent direction.
In _proximal_jvp_f_pure(), we compute,

$$df\,dc = \left(\frac{\partial f}{\partial x}\right)^{-}\frac{\partial f}{\partial c} \tag{237}$$

Note that f is the ForceBalance and hence doesn't have any dependence on parameters of other *things*. So, we first divide the v into each component for different *thing*, and take the derivative with respect to Rb_{lmn}, Zb_{lmn}, and profile params.

However, as we said previously, the compute functions don't have those parameters, so we need to find their corresponding $R_{lmn}, Z_{lmn}, L_{lmn}$ and profile params using dxdc, see Equation 229.

$$\frac{\partial f}{\partial c} = jvp(dxdc@dc, xf, constants) \tag{238}$$

where xf is the state vector of the Equilibrium.

For the x part, we take the derivative using the feasible_tangents of the equilibrium sub-problem using the LinearConstraintProjection we created earlier,

$$\frac{\partial f}{\partial x} = jvp(feasible\_tangents, xf, constants) \tag{239}$$

And the inversion is carried out by Singular Value Decomposition (SVD). Note that in general, we get ill-conditioned matrices, therefore we add a small regularisation, by adding the smallest eigenvalue to all of the eigenvalues. **However, I am not sure if this is true for all cases, for well-conditioned problems, this can mess up stuff!**

Note that above $dfdc$ is the derivative of Equilibrium with respect to some of the Rb_{lmn}, Zb_{lmn}, and profile params (the ones that are free, so in the y_{eq}). But the full differentiation also includes other optimization parameters from other optimizable objects. That is why, we concatenate $dfdc$ with zeros of their size.

$dfdc$ can be tranformed into the size of the full state vector by multiplication of feasible_tangents as shown in Equation 234. So, we got,

$$\frac{\partial g}{\partial x}\left(\frac{\partial f}{\partial x}\right)^{-}\frac{\partial f}{\partial c} = jvp(feasible\_tangents@dfdc, xg, constants) \tag{240}$$

The last part we need is the direction in the optimization parameters. Since variables of other optimizable are all in c, we don't need to change the part of v that was passed to ProximalProjection. However, as noted before, we need to find corresponding R_{lmn}, Z_{lmn} for Rb_{lmn}, Zb_{lmn} by multiplying the Equilibrium part of v by dxdc which will give us the tangent in the full space, as shown in Equation 229.

5.2.7 JVP

We found the proper tangent direction for the derivative. All we need to do is to compute the derivative using JVPs based on the ***blocked*** or ***batched*** method of the ObjectiveFunction.

6 Profiling DESC

Note: This is adapted from the document created during the 2024 NVidia Hackathon, see original here.

6.1 Profiling a simple script with NSight

For profiling, in addition to usual DESC dependencies, you will need **nvtx** which you can install using pip,

```
pip install nvtx
```

Here's a simple test script that we use for profiling equilibrium solve.

```
from desc import set_device
set_device("gpu")
import numpy as np

from desc.equilibrium import Equilibrium
from desc.geometry import FourierRZToroidalSurface
from desc.objectives import (
    ObjectiveFunction,
    ForceBalance,
    get_fixed_boundary_constraints,
)
from desc.optimize import Optimizer
from desc.plotting import plot_1d, plot_section, plot_surfaces
from desc.profiles import PowerSeriesProfile
import sys
import nvtx

with nvtx.annotate("surface 2D Init"):
    surface_2D = FourierRZToroidalSurface(
        R_lmn=np.array([10, -1]),
        Z_lmn=np.array([1]),
        modes_R=np.array([[0, 0], [1, 0]]),
        modes_Z=np.array([[-1, 0]]),
        NFP=5,
    )
with nvtx.annotate("Equilibrium Init"):
    eq = Equilibrium(
        surface=surface_2D,
        sym=True,
        L=int(sys.argv[1]),
        M=int(sys.argv[1]),
        N=int(sys.argv[1]),
        L_grid=2*int(sys.argv[1]),
        M_grid=2*int(sys.argv[1]),
        N_grid=2*int(sys.argv[1]),
    )
with nvtx.annotate("Objectives"):
    objective = ObjectiveFunction(ForceBalance(eq=eq))
with nvtx.annotate("Constraints"):
    constraints = get_fixed_boundary_constraints(eq=eq)
with nvtx.annotate("optimizer"):
    optimizer = Optimizer("lsq-exact")
with nvtx.annotate("solve"):
    eq, solver_outputs = eq.solve(
        objective=objective, constraints=constraints, optimizer=optimizer, maxiter=10, ftol=0, gtol=0, xtol=0, verbose=3
    )
```

We run this script with NSight on the Della cluster.

```
module purge
module load anaconda3/2024.6
conda activate desc-env
nsys profile --trace=nvtx python3 <test_script.py> 4  > trace0.out
```

Of course for running this on GPU, you will need to ask for a GPU in your slurm. For more slurm options, check Princeton Research cluster page.

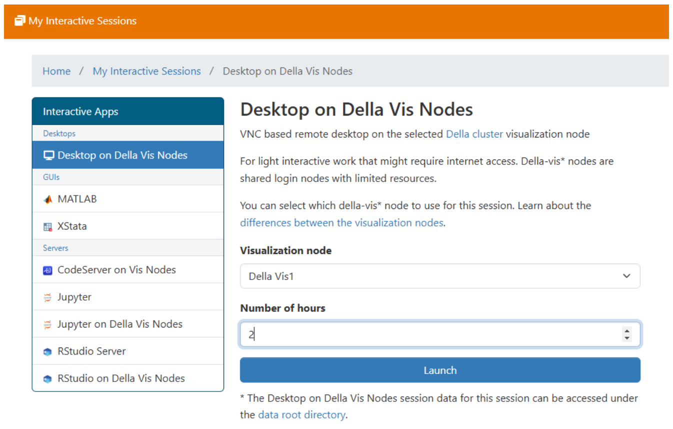
```
#!/bin/bash
#SBATCH --job-name=profile        # create a short name for your job
#SBATCH --nodes=1                # node count
#SBATCH --cpus-per-task=32        # cpu-cores per task (>1 if multi-threaded tasks)
#SBATCH --mem=64G          # memory per cpu-core (4G is default)
#SBATCH --time=00:15:00          # total run time limit (HH:MM:SS)
#SBATCH --gres=gpu:1
#SBATCH --constraint=gpu80

module purge
module load anaconda3/2024.6
conda activate desc-env

nsys profile --trace=nvtx,cuda,cusolver --output=your-name-for-the-report --force-overwrite=true python test_script.py 12
```

Here, –**constraint=gpu80** is optional (asks a GPU with 80GB memory). –**trace=nvtx,cuda,cusolver** part optionally traces cuda and cusolver activities, for example QR or SVD calls can be seen with these settings. –**output** is used for giving a name for the report. –**force-overwrite** is used to overwrite the existing file (note that for the overwrite to happen successfully, you should close the file from Nsight systems, which we will talk in shortly). For more details on Nsight options, check NVidia page.

To run the NSight GUI, you must log on the Della visual node.


My Interactive Sessions
Home / My Interactive Sessions / Desktop on Della Vis Nodes
Interactive Apps
Desktops
Desktop on Della Vis Nodes
GUIs
MATLAB
XStata
Servers
CodeServer on Vis Nodes
Jupyter
Jupyter on Della Vis Nodes
RStudio Server
RStudio on Della Vis Nodes
Launch

Desktop on Della Vis Nodes

VNC based remote desktop on the selected Della cluster visualization node

For light interactive work that might require internet access. Della-vis* nodes are shared login nodes with limited resources.

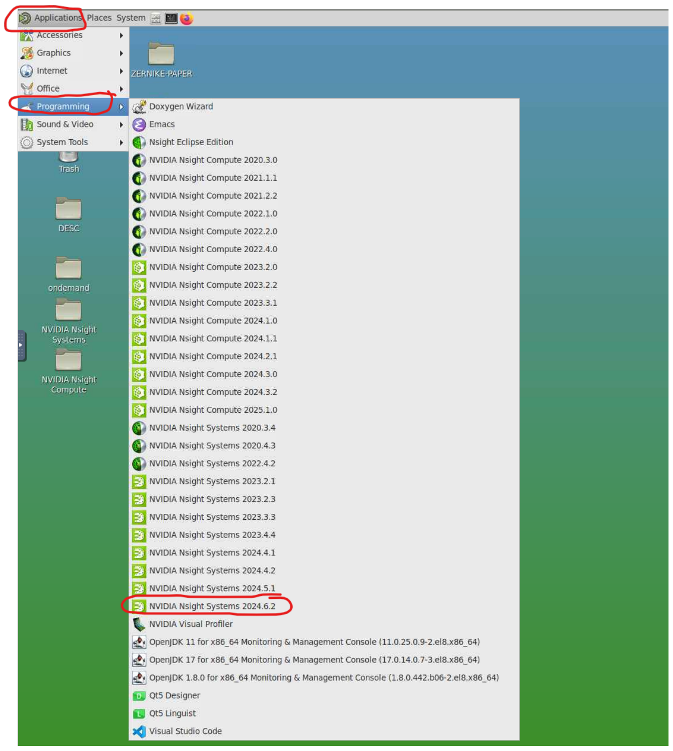
You can select which della-vis* node to use for this session. Learn about the differences between the visualization nodes.

Visualization node: Della Vis1

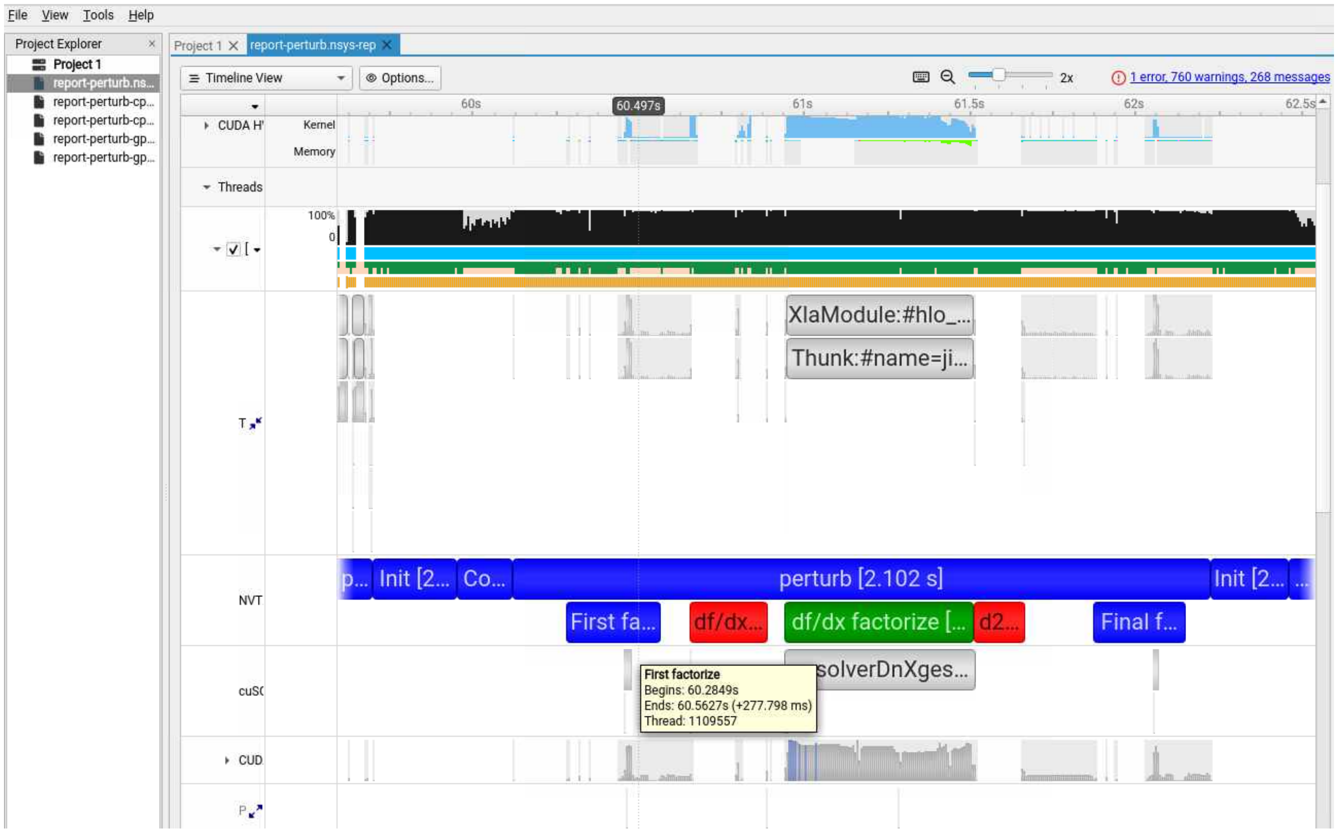
Number of hours: 2

* The Desktop on Della Vis Nodes session data for this session can be accessed under the data root directory.

Then, open the Nsight systems software,


Applications Places System
Accessories
Graphics
Internet
Office
Programming
Sound & Video
System Tools
ZERNIKE-PAPER
Trash
DESC
ondemand
NVIDIA Nsight Systems
NVIDIA Nsight Compute
Doxygen Wizard
Emacs
Nsight Eclipse Edition
NVIDIA Nsight Compute 2020.3.0
NVIDIA Nsight Compute 2021.1.1
NVIDIA Nsight Compute 2021.2.2
NVIDIA Nsight Compute 2022.1.0
NVIDIA Nsight Compute 2022.2.0
NVIDIA Nsight Compute 2022.4.0
NVIDIA Nsight Compute 2023.2.0
NVIDIA Nsight Compute 2023.2.2
NVIDIA Nsight Compute 2023.3.1
NVIDIA Nsight Compute 2024.1.0
NVIDIA Nsight Compute 2024.1.1
NVIDIA Nsight Compute 2024.2.1
NVIDIA Nsight Compute 2024.3.0
NVIDIA Nsight Compute 2024.3.2
NVIDIA Nsight Compute 2025.1.0
NVIDIA Nsight Systems 2020.3.4
NVIDIA Nsight Systems 2020.4.3
NVIDIA Nsight Systems 2022.4.2
NVIDIA Nsight Systems 2023.2.1
NVIDIA Nsight Systems 2023.2.3
NVIDIA Nsight Systems 2023.3.3
NVIDIA Nsight Systems 2023.4.4
NVIDIA Nsight Systems 2024.4.1
NVIDIA Nsight Systems 2024.4.2
NVIDIA Nsight Systems 2024.5.1
NVIDIA Nsight Systems 2024.6.2
NVIDIA Visual Profiler
OpenJDK 11 for x86_64 Monitoring & Management Console (11.0.25.0.9-2.el8.x86_64)
OpenJDK 17 for x86_64 Monitoring & Management Console (17.0.14.0.7-3.el8.x86_64)
OpenJDK 1.8.0 for x86_64 Monitoring & Management Console (1.8.0.442.b06-2.el8.x86_64)
Qt5 Designer
Qt5 Linguist
Visual Studio Code

Here is an example report, created for perturb function,



First factorize
Begins: 60.2849s
Ends: 60.5627s (+277.798 ms)
Thread: 1109557

Note that you can add **nvtx** annotations inside the source code to profile specific places of the code. Unfortunately, you cannot annotate inside a jitted function. However, you can temporarily disable jit and profile like that.

6.2 Memory Profiling

Jax by default preallocates 75% of the available memory to the Jax process.

```
XLA_PYTHON_CLIENT_ALLOCATOR=platform
```

This will disable JAXs memory preallocation and force it to use CUDAs allocator: while slower, it does make it easier to understand where memory is being allocated, especially when combined with the –**cuda-memory-usage=true** option to **nsys** profile.

See Jax documentation.

Tracing Instrument code with (NVTX page)

Recommended to use –**trace=cuda,nvtx,cusolver** with **nsys** profile JAX uses cuSOLVER for the linear algebra operations Set the option jax.config.update("jax_traceback_in_locations_limit", -1) This will give deeper XLA traces (see Jax documentation)

Starting and stopping instead of profiling the whole app

These functions start and stop the profiler, add them to the top of the file where you want to control profiling:

```python
# profile stuff
import nvtx
from ctypes import cdll
libcudart = cdll.LoadLibrary('libcudart.so')

def cudaProfilerStart(enabled=True):
    if enabled:
        libcudart.cudaProfilerStart()

def cudaProfilerStop(enabled=True):
    if enabled:
        libcudart.cudaProfilerStop()
```

And then cudaProfilerStart where you want the profile trace to start. Add cudaProfilerStop where you want the profile trace to stop, or it'll end automatically at the end. When using start or stop add –capture-range cudaProfilerApi To the nsys profile command otherwise it'll keep profiling the whole application Goals for the hackathon

Memory profiling (Can't get any useful info from Perfetto or NSight. Maybe use tensorboard on a TPU to get an idea? Is there another software or any way to learn memory consumption on a GPU?) Try float32 or bfloat and see what parts can work with mixed precision Sharding? It's not like a shared-data neural network. Instead we have multiple objectives that use the same data (sort of). We want to split the objectives and share the same equilibrium data. Splitting objectives over multiple GPUs (Scaling tests?) Writing a small bit of code in native XLA (bad idea, last resort)

Things to try

Call nvidia-smi from different modules inside DESC Install pynvml pynvml pip install pynvml In python code

```python
import pynvml
# Initialize
pynvml.nvmlInit()

...
# get device handle
nvml_handle = pynvml.nvmlDeviceGetHandleByIndex(<device id>)
```

```
    ...
    # get total memory usage (as seen in nvidia-smi)
    all_mem_gb = pynvml.nvmlDeviceGetMemoryInfo(nvml_handle).used / (1024.0 * 1024.0 * 1024.0)
```

Here is an example script we use to look at the memory consumption,

```
#!/usr/bin/env python3
import subprocess
import time
import threading
import matplotlib.pyplot as plt


def monitor_vram(duration, interval, vram_usage_list, timestamps):
    end_time = time.time() + duration
    while time.time() < end_time:
        result = subprocess.run(
            ["nvidia-smi", "--query-gpu=memory.used", "--format=csv,noheader,nounits"],
            stdout=subprocess.PIPE,
        )
        output = result.stdout.decode("utf-8").strip()
        vram_usage = int(output.split()[0])
        vram_usage_list.append(vram_usage)
        timestamps.append(time.time())
        time.sleep(interval)


if __name__ == "__main__":
    duration = 200  # duration to monitor in seconds
    interval = 0.1  # interval between checks in seconds
    vram_usage_list = []
    timestamps = []

    # create threads for monitoring VRAM and running GPU code
    vram_thread = threading.Thread(
        target=monitor_vram, args=(duration, interval, vram_usage_list, timestamps)
    )

    # start the threads
    vram_thread.start()

    # res = 15
    # run without blocking (Popen)
    # subprocess.Popen(['python', "DESC_profiler_new.py", "5"])
    subprocess.Popen(["python", "DESC_profiler_eq_solve_w7x_mem.py"])

    # wait for the thread to finish
    vram_thread.join()

    # write the VRAM usage to a file
    with open("yge_vram_usage.txt", "w") as file:
        for usage in vram_usage_list:
            file.write(f"{usage}\n")

    plt.figure(figsize=(15, 7))
    # plot the VRAM usage
    times = [t - timestamps[0] for t in timestamps]
    import numpy as np
```

```
vram_usage_list = np.array(vram_usage_list)
times = np.array(times)

# this is kind of ad-hoc way of clipping the intended section
idx = np.where(vram_usage_list > 1000)
idx = np.arange(len(vram_usage_list))

plt.plot(times[idx], vram_usage_list[idx], "-or", ms=2)
plt.xlabel("Time (s)", fontsize=20)
plt.ylabel("VRAM Usage (MiB)", fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title("GPU VRAM Usage Over Time")
plt.grid(True)
plt.tight_layout()
plt.savefig(f"yge_test0_maxiter2_qr.png", dpi=1000)
# plt.show()
```

7 Multi-Device Applications

7.1 Installing MPI4JAX

The installation on the website does not work as is, also for computing clusters in Princeton we already have C compilers for openmpi, so we will use them. The instructions are slightly complicated but as of February 18th 2025, it works. First, we will get openmpi from cluster modules, install jax with GPU support. Then, we will use the hacky way of installing mpi4py and mpi4jax as explained in the Princeton cluster page, note that that page use the trick only for mpi4py but we will use it for mpi4jax too, so, follow these instructions to revert the changes.

```
module load anaconda3/2024.6
conda create -n desc-mpi python=3.12
conda activate desc-mpi
module load openmpi/gcc/4.1.6
# install jax
pip install -U jax['cuda12']
```

For Python versions newer than 3.8, we have to use following as explained in Princeton Research Computing page.

```
# install mpi4py
export MPICC=$(which mpicc)
cd /home/<YOUR_NETID>/.conda/envs/desc-mpi/compiler_compat
rm -f ld
ln -s /usr/bin/ld ld
pip install mpi4py --no-cache-dir
# move back to main folder, we don't want to install it to /compiler_compat
cd ~
# To install `mpi4jax`, we need to grab the source code
# install mpi4jax
git clone https://github.com/mpi4jax/mpi4jax
cd mpi4jax
```

Comment out a couple of lines in ‘setup.py‘ in mpi4jax directory (inside get_sycl_path()):

```
elif os.path.exists("/opt/intel/oneapi/compiler/latest/"):
        _sycl_path = "/opt/intel/oneapi/compiler/latest/"
```

and then back in the command line,

```
cd ..
python -c 'import mpi4jax' # check that it works
pip install -e .  # install mpi4jax to the environment, add it to the path
# Now that we are done with openMPI stuff, revert some changes previously made
cd /home/<NetID>/.conda/envs/desc-mpi/compiler_compat
rm -f ld
ln -s ../bin/x86_64-conda-linux-gnu-ld ld
# move back to main folder
cd ~
# install DESC as usual
```

mpi4jax hello_world example: `https://mpi4jax.readthedocs.io/en/latest/usage.html`

Note: I believe that we can skip cloning mpi4jax by setting some environment variable, it should basically use the proper compiler, maybe try to set “CMPLR_ROOT”? If we ever gonna make this a user feature, it needs to be less complicated as it can be. See the issue for future fixes.

8 DON'T FORGET TO INCLUDE

- Better explain optimization
- Make better figure for coordinates
- Make a plot using Dario's script for covariant and contravariant basis vectors

References

[1] W. D. D'haeseleer, W. N. G. Hitchon, J. D. Callen, and J. L. Shohet, *Flux Coordinates and Magnetic Field Structure: A Guide to a Fundamental Tool of Plasma Theory*. Springer Science & Business Media, Dec. 2012.

[2] "5.7: Cylindrical and Spherical Coordinates," Jan. 2019.

[3] L.-M. Imbert-Gerard, E. J. Paul, and A. M. Wright, "An Introduction to Stellarators."

A Relation between Zernike Polynomials and Jacobi Polynomials

For the special case of $\beta = 0$, $x = 1 - 2\rho^2$ and $\alpha = m$, Jacobi Polynomials can be written using equation **??**,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} (-1)^s \binom{n+m}{n-s}\binom{n}{s} \rho^{2s}(1-\rho^2)^{n-s} \tag{241}$$

Now, let's use binomial theorem to expand $(1-\rho^2)^{n-s}$,

$$(1-\rho^2)^{n-s} = \sum_{k=0}^{n-s} (-1)^k \binom{n-s}{k} \rho^{2k} \tag{242}$$

Substitute this in eq 266,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} (-1)^s \binom{n}{s}\binom{n+m}{n-s} \rho^{2s} \sum_{k=0}^{n-s} (-1)^k \binom{n-s}{k} \rho^{2k} \tag{243}$$

Now, rearrange the terms,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} \sum_{k=0}^{n-s} (-1)^{(s+k)} \binom{n+m}{n-s}\binom{n}{s}\binom{n-s}{k} \rho^{2(s+k)} \tag{244}$$

Substitute $j = s+k$, hence $k = j - s$,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} \sum_{j-s=0}^{j-s=n-s} (-1)^j \binom{n+m}{n-s}\binom{n}{s}\binom{n-s}{j-s} \rho^{2j} \tag{245}$$

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} \sum_{j=s}^{n} (-1)^j \binom{n+m}{n-s}\binom{n}{s}\binom{n-s}{j-s} \rho^{2j} \tag{246}$$

Here, we can change the order of summation, it is better to use table to find new limits,

	0	1	2	$\cdots$	$n-1$	n
$s=0$	$\times$	$\times$	$\times$	$\cdots$	$\times$	$\times$
$s=1$		$\times$	$\times$	$\cdots$	$\times$	$\times$
$s=2$			$\times$	$\cdots$	$\times$	$\times$
$\vdots$				$\ddots$	$\vdots$	$\vdots$
$s=n-1$					$\times$	$\times$
$s=n$						$\times$

Each $\times$ represents a valid pair (s, j). This can be re-written in terms of summation over j first, then s,

	0	1	2	$\cdots$	$n-1$	n
$j=0$	$\times$					
$j=1$	$\times$	$\times$				
$j=2$	$\times$	$\times$	$\times$			
$\vdots$				$\ddots$		
$j=n-1$	$\times$	$\times$	$\times$	$\cdots$	$\times$	
$j=n$	$\times$	$\times$	$\times$	$\cdots$	$\times$	$\times$

Since the (s, j) pairs are the same, the nested summation can be written as,

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} \rho^{2j} \sum_{s=0}^{j} (-1)^j \frac{(n+m)!}{(n-s)!(m+s)!} \frac{n!}{s!(n-s)!} \frac{(n-s)!}{(j-s)!(n-j)!} \tag{247}$$

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} \rho^{2j} \sum_{s=0}^{j} (-1)^j \frac{(n+m)!}{(n-s)!(m+s)!} \frac{j!}{s!(j-s)!} \frac{n!}{j!(n-j)!} \tag{248}$$

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} (-1)^j \binom{n}{j} \rho^{2j} \sum_{s=0}^{j} \binom{n+m}{n-s} \binom{j}{s} \tag{249}$$

Now, we need to use a property of the binomial coefficients. Consider,

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k \tag{250}$$

$$(1+x)^{n+m}(1+x)^j = \left(\sum_{k=0}^{n+m} \binom{n+m}{k} x^k \right) \left(\sum_{k=0}^{j} \binom{j}{k} x^k \right) = \left(\sum_{s=-m}^{n} \binom{n+m}{n-s} x^{n-s} \right) \left(\sum_{k=0}^{j} \binom{j}{k} x^k \right) \tag{251}$$

$$(1+x)^{n+m+j} = \sum_{k=0}^{n+m+j} \binom{n+m+j}{k} x^k \tag{252}$$

For x^γ coefficient, we have

$$\binom{n+m+j}{\gamma} x^\gamma = \sum_{k=0}^{\gamma} \binom{j}{k} \binom{n+m}{\gamma-k} x^\gamma \tag{253}$$

In previous step, I used $n-s+k=\gamma$ and $s=n+k-\gamma$, hence $n-s=\gamma-k$. Now, let's substitute $\gamma=n$,

$$\binom{n+m+j}{n} = \sum_{k=0}^{j} \binom{n+m}{n-k}\binom{j}{k} \tag{254}$$

We can finally use this relation to simplify eq 274,

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} (-1)^j \rho^{2j} \binom{n}{j}\binom{n+m+j}{n} \tag{255}$$

Lets's multiply last equation by ρ^m and $(-1)^n$,

$$(-1)^n \rho^m P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} (-1)^{j+n} \rho^{2j+m} \binom{n}{j}\binom{n+m+j}{n} \tag{256}$$

Substitute $j=n-s$,

$$(-1)^n \rho^m P_n^{m,0}(1-2\rho^2) = \sum_{n-s=0}^{n-s=n} (-1)^{2n-s} \rho^{2n+m-s} \binom{n}{n-s}\binom{2n+m-s}{n} \tag{257}$$

$$(-1)^{\frac{l-m}{2}} \rho^m P_{\frac{l-m}{2}}^{m,0}(1-2\rho^2) = \sum_{s=0}^{(l-m)/2} (-1)^s \rho^{l-2s} \binom{\frac{l-m}{2}}{s}\binom{l-s}{\frac{l-m}{2}} \tag{258}$$

$$(-1)^{\frac{l-m}{2}} \rho^m P_{\frac{l-m}{2}}^{m,0}(1-2\rho^2) = \sum_{s=0}^{(l-m)/2} (-1)^s \frac{\frac{l-m}{2}!}{s!(\frac{l-m}{2}-s)!} \frac{(l-s)!}{\frac{l-m}{2}!(\frac{l+m}{2}-s)!} \rho^{l-2s} \tag{259}$$

$$\mathcal{R}_l^m(\rho) = (-1)^{\frac{l-m}{2}} \rho^m P_{\frac{l-m}{2}}^{m,0}(1-2\rho^2) = \sum_{s=0}^{(l-m)/2} \frac{(-1)^s (l-s)!}{s!\left(\dfrac{l+m}{2}-s\right)!\left(\dfrac{l-m}{2}-s\right)!} \rho^{l-2s} \tag{260}$$

which is exactly equivalent to the radial part of the Zernike Polynomials.

B Covariant J Alternative

If we use equation 74 , we need to find covariant components of **J** too, which are as shown above,

$$J_\rho = \mathbf{J}\cdot\mathbf{e}_\rho = (J^\rho\mathbf{e}_\rho + J^\theta\mathbf{e}_\theta + J^\zeta\mathbf{e}_\zeta)\cdot\mathbf{e}_\rho \tag{261}$$

$$J_\theta = \mathbf{J}\cdot\mathbf{e}_\theta = (J^\rho\mathbf{e}_\rho + J^\theta\mathbf{e}_\theta + J^\zeta\mathbf{e}_\zeta)\cdot\mathbf{e}_\theta \tag{262}$$

$$J_\zeta = \mathbf{J}\cdot\mathbf{e}_\zeta = (J^\rho\mathbf{e}_\rho + J^\theta\mathbf{e}_\theta + J^\zeta\mathbf{e}_\zeta)\cdot\mathbf{e}_\zeta \tag{263}$$

Hence,

$$\mathbf{J}\times\mathbf{B} = \frac{1}{\sqrt{g}}\sum_k (J_iB_j - J_jB_i)\mathbf{e}_k \tag{264}$$

$$= \frac{1}{\sqrt{g}}\Big[(J_\theta B_\zeta - J_\zeta B_\theta)\mathbf{e}_\rho \;+\; (J_\zeta B_\rho - J_\rho B_\zeta)\mathbf{e}_\theta \;+\; (J_\rho B_\theta - J_\theta B_\rho)\mathbf{e}_\zeta\Big] \tag{265}$$

C Relation between Zernike Polynomials and Jacobi Polynomials

For the special case of $\beta = 0$, $x = 1 - 2\rho^2$ and $\alpha = m$, Jacobi Polynomials can be written using equation **??**,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} (-1)^s \binom{n+m}{n-s} \binom{n}{s} \rho^{2s} (1-\rho^2)^{n-s} \tag{266}$$

Now, let's use binomial theorem to expand $(1-\rho^2)^{n-s}$,

$$(1-\rho^2)^{n-s} = \sum_{k=0}^{n-s} (-1)^k \binom{n-s}{k} \rho^{2k} \tag{267}$$

Substitute this in eq 266,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} (-1)^s \binom{n}{s} \binom{n+m}{n-s} \rho^{2s} \sum_{k=0}^{n-s} (-1)^k \binom{n-s}{k} \rho^{2k} \tag{268}$$

Now, rearrange the terms,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} \sum_{k=0}^{n-s} (-1)^{(s+k)} \binom{n+m}{n-s} \binom{n}{s} \binom{n-s}{k} \rho^{2(s+k)} \tag{269}$$

Substitute $j = s + k$, hence $k = j - s$,

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} \sum_{j-s=0}^{j-s=n-s} (-1)^j \binom{n+m}{n-s} \binom{n}{s} \binom{n-s}{j-s} \rho^{2j} \tag{270}$$

$$P_n^{m,0}(1-2\rho^2) = \sum_{s=0}^{n} \sum_{j=s}^{n} (-1)^j \binom{n+m}{n-s} \binom{n}{s} \binom{n-s}{j-s} \rho^{2j} \tag{271}$$

Here, we can change the order of summation, it is better to use table to find new limits,

	0	1	2	$\cdots$	$n-1$	n
$s=0$	$\times$	$\times$	$\times$	$\cdots$	$\times$	$\times$
$s=1$		$\times$	$\times$	$\cdots$	$\times$	$\times$
$s=2$			$\times$	$\cdots$	$\times$	$\times$
$\vdots$				$\ddots$	$\vdots$	$\vdots$
$s=n-1$					$\times$	$\times$
$s=n$						$\times$

Each $\times$ represents a valid pair (s, j). This can be re-written in terms of summation over j first, then s,

	0	1	2	$\cdots$	$n-1$	n
$j=0$	$\times$					
$j=1$	$\times$	$\times$				
$j=2$	$\times$	$\times$	$\times$			
$\vdots$				$\ddots$		
$j=n-1$	$\times$	$\times$	$\times$	$\cdots$	$\times$	
$j=n$	$\times$	$\times$	$\times$	$\cdots$	$\times$	$\times$

Since the (s, j) pairs are the same, the nested summation can be written as,

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} \rho^{2j} \sum_{s=0}^{j} (-1)^j \frac{(n+m)!}{(n-s)!(m+s)!} \frac{n!}{s!(n-s)!} \frac{(n-s)!}{(j-s)!(n-j)!} \tag{272}$$

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} \rho^{2j} \sum_{s=0}^{j} (-1)^j \frac{(n+m)!}{(n-s)!(m+s)!} \frac{j!}{s!(j-s)!} \frac{n!}{j!(n-j)!} \tag{273}$$

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} (-1)^j \binom{n}{j} \rho^{2j} \sum_{s=0}^{j} \binom{n+m}{n-s} \binom{j}{s} \tag{274}$$

Now, we need to use a property of the binomial coefficients. Consider,

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k \tag{275}$$

$$(1+x)^{n+m}(1+x)^j = \left(\sum_{k=0}^{n+m} \binom{n+m}{k} x^k \right) \left(\sum_{k=0}^{j} \binom{j}{k} x^k \right) = \left(\sum_{s=-m}^{n} \binom{n+m}{n-s} x^{n-s} \right) \left(\sum_{k=0}^{j} \binom{j}{k} x^k \right) \tag{276}$$

$$(1+x)^{n+m+j} = \sum_{k=0}^{n+m+j} \binom{n+m+j}{k} x^k \tag{277}$$

For x^γ coefficient, we have

$$\binom{n+m+j}{\gamma} x^\gamma = \sum_{k=0}^{\gamma} \binom{j}{k} \binom{n+m}{\gamma-k} x^\gamma \tag{278}$$

In previous step, I used $n-s+k=\gamma$ and $s=n+k-\gamma$, hence $n-s=\gamma-k$. Now, let's substitute $\gamma=n$,

$$\binom{n+m+j}{n} = \sum_{k=0}^{j} \binom{n+m}{n-k} \binom{j}{k} \tag{279}$$

We can finally use this relation to simplify eq 274,

$$P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} (-1)^j \rho^{2j} \binom{n}{j} \binom{n+m+j}{n} \tag{280}$$

Lets's multiply last equation by ρ^m and $(-1)^n$,

$$(-1)^n \rho^m P_n^{m,0}(1-2\rho^2) = \sum_{j=0}^{n} (-1)^{j+n} \rho^{2j+m} \binom{n}{j} \binom{n+m+j}{n} \tag{281}$$

Substitute $j=n-s$,

$$(-1)^n \rho^m P_n^{m,0}(1-2\rho^2) = \sum_{n-s=0}^{n-s=n} (-1)^{2n-s} \rho^{2n+m-s} \binom{n}{n-s} \binom{2n+m-s}{n} \tag{282}$$

$$(-1)^{\frac{l-m}{2}} \rho^m P_{\frac{l-m}{2}}^{m,0}(1-2\rho^2) = \sum_{s=0}^{(l-m)/2} (-1)^s \rho^{l-2s} \binom{\frac{l-m}{2}}{s} \binom{l-s}{\frac{l-m}{2}} \tag{283}$$

$$(-1)^{\frac{l-m}{2}} \rho^m P_{\frac{l-m}{2}}^{m,0}(1-2\rho^2) = \sum_{s=0}^{(l-m)/2} (-1)^s \frac{\frac{l-m}{2}!}{s!(\frac{l-m}{2}-s)!} \frac{(l-s)!}{\frac{l-m}{2}!(\frac{l+m}{2}-s)!} \rho^{l-2s} \tag{284}$$

$$\mathcal{R}_l^m(\rho) = (-1)^{\frac{l-m}{2}} \rho^m P_{\frac{l-m}{2}}^{m,0}(1-2\rho^2) = \sum_{s=0}^{(l-m)/2} \frac{(-1)^s (l-s)!}{s! \left(\dfrac{l+m}{2} - s\right)! \left(\dfrac{l-m}{2} - s\right)!} \rho^{l-2s} \tag{285}$$

which is exactly equivalent to the radial part of the Zernike Polynomials.