

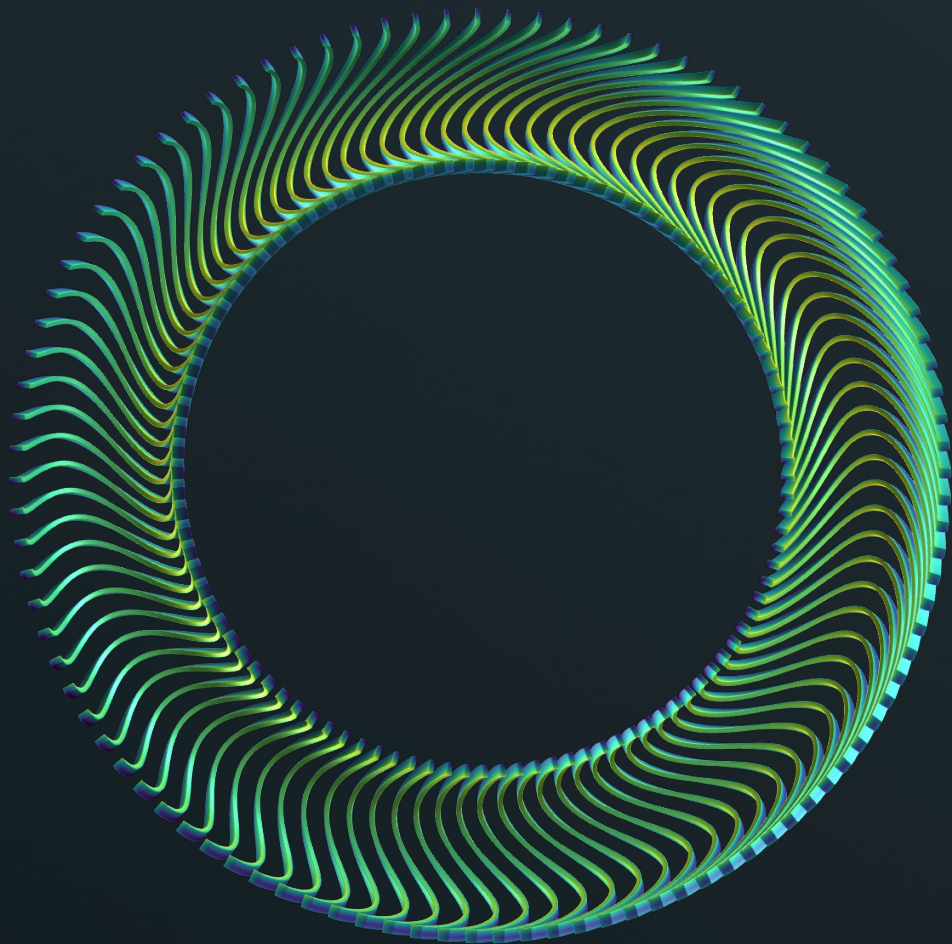


# RAT | An Open-Source Magnet-Design Tool

Autumn 2024



# Today's topics

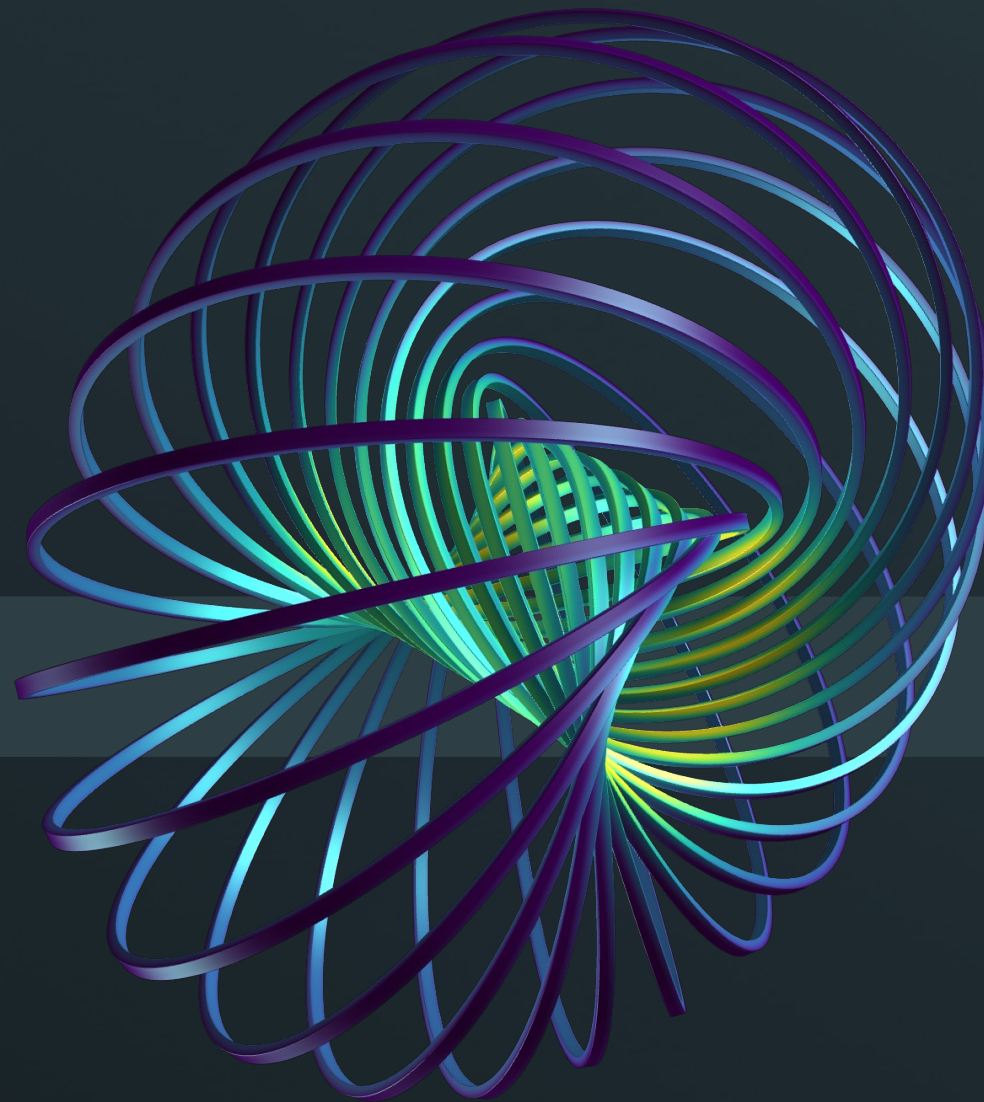


- Introduction
  - What is RAT?
  - How does RAT work?
  - How is RAT validated?
  - How can I get RAT?
- Features
  - Winding algorithm
  - Longitudinal Grading
  - Connectors
  - FreeCAD interface
  - Modeling CCT Magnets
- CMS Modeling Example
  - Building CMS magnet geometry in 10 steps
- CMS Calculations



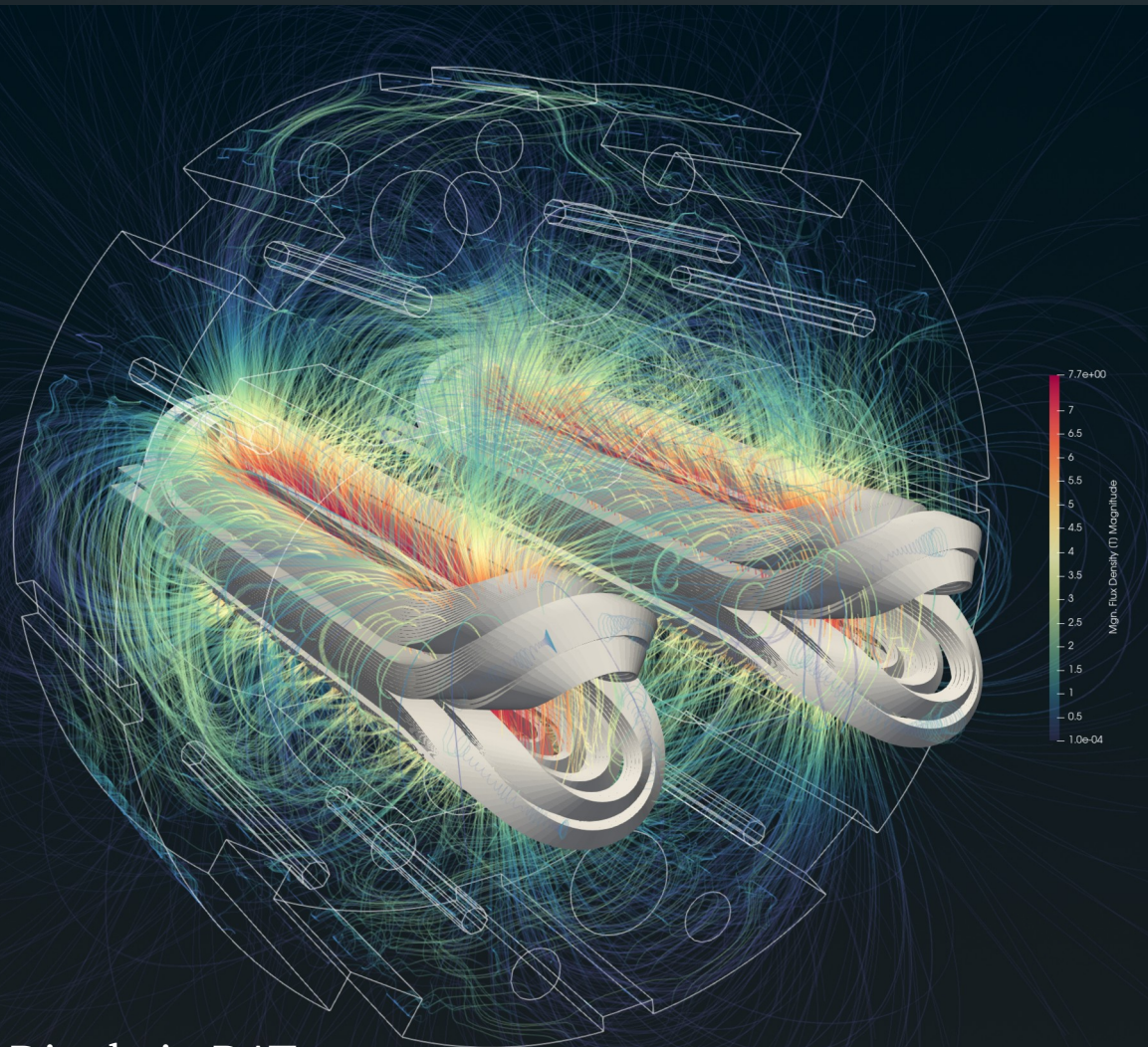
# Introduction

What is RAT and what can it do?





# What is ?



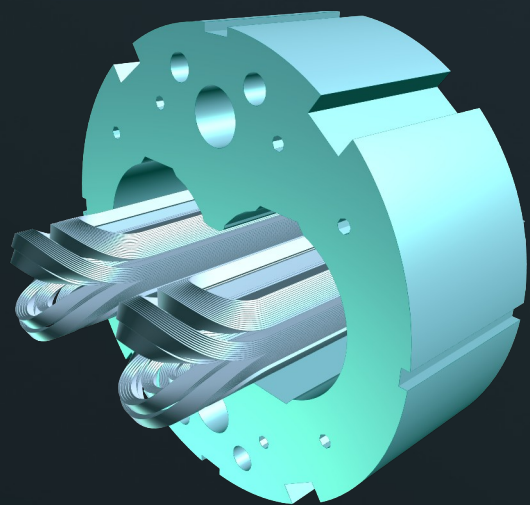
LHC Dipole in RAT

- Software for **3D magneto-statics** including non-linear materials
- Extensive coil/yoke modeler with numerous templates
- Versatile calculation options
- Never ever mesh the air:
  - Biot-Savart with **Fast Multipole Method (FMM)** for coils
  - **Volume Integral Method (VIM)** [1] for HB domains (i.e. Iron)

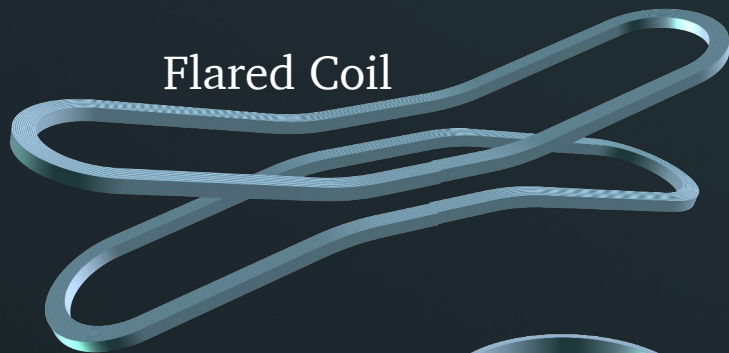
[1] V. Le-Van, G. Meunier, O. Chadebec and J. -M. Guichon, "A Magnetic Vector Potential Volume Integral Formulation for Nonlinear Magnetostatic Problems," in IEEE Transactions on Magnetics, vol. 52, no. 3, pp. 1-4, March 2016, Art no. 7002804, doi: 10.1109/TMAG.2015.2490627.



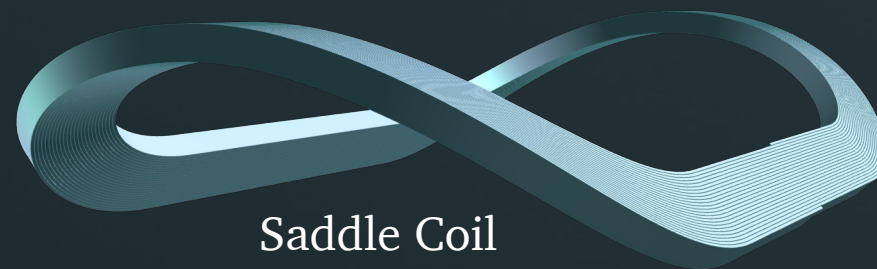
# Many Built in Coil Templates



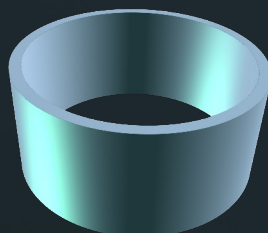
Cos-Theta



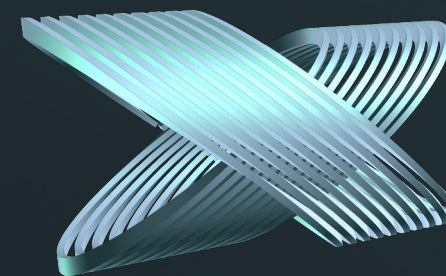
Flared Coil



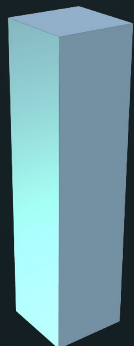
Saddle Coil



Solenoid



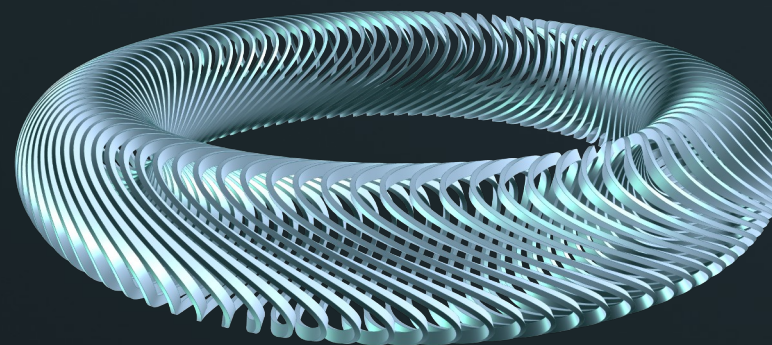
Regular CCT



Bar Magnet



Cloverleaf



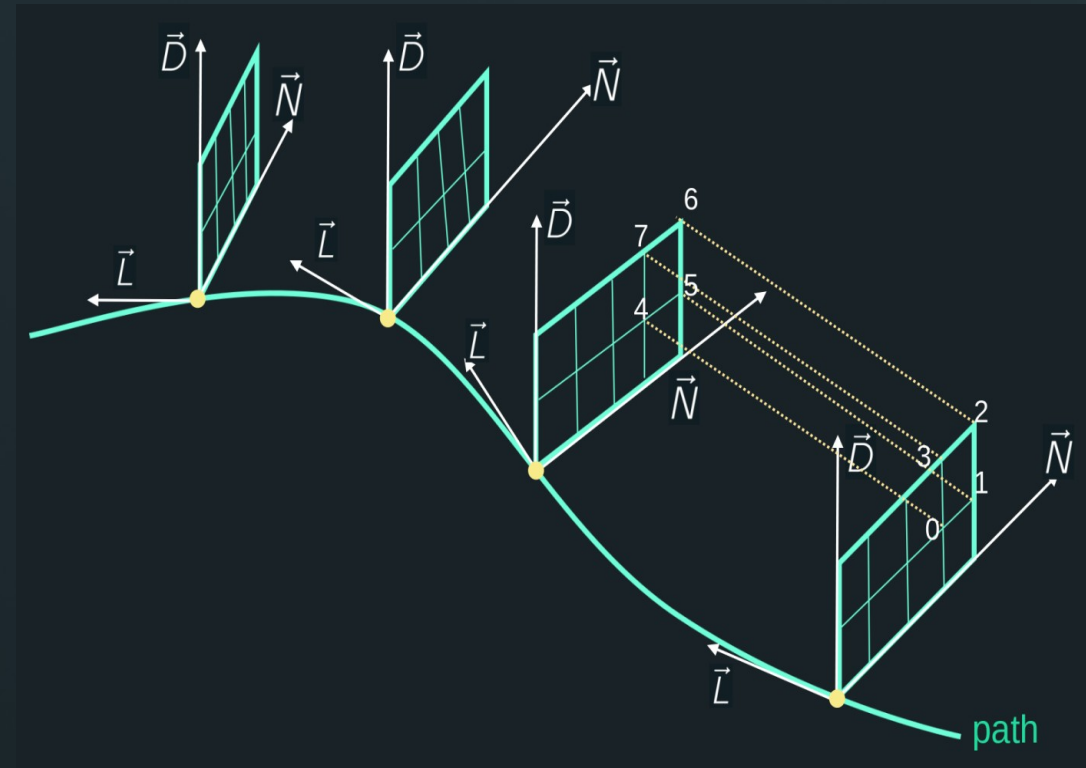
Custom CCT

And More ...



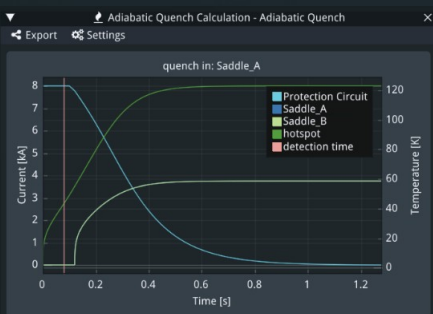
# Modeling Custom Coils in

- All coil/yoke objects in RAT are constructed using:
  - **A path:** a 3D space curve with local coordinate system (Frame)
  - **A Cross Section:** a 2D area defined by a closed perimeter
- The **2D** cross section is meshed:
  - Regular mesh for simple shapes
  - Distmesher for complicated shapes
- The 2D mesh is **extruded** along the path to create a 3D Mesh

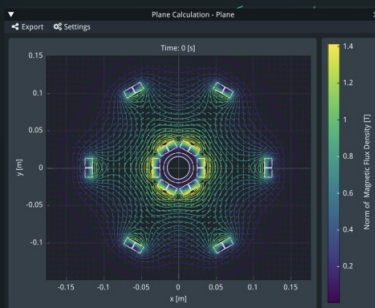




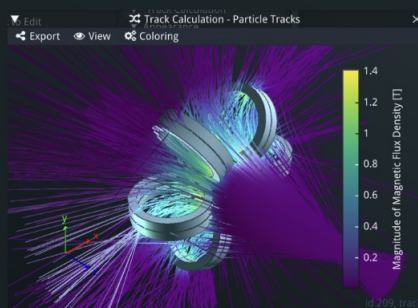
# Many Built in Calculations



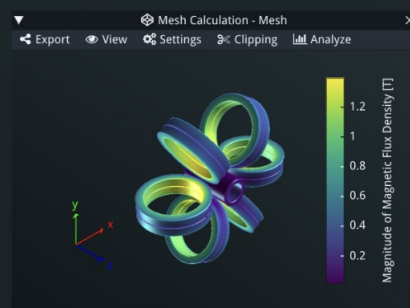
Adiabatic Quench (new)



Plane Calculation



Particle Tracks



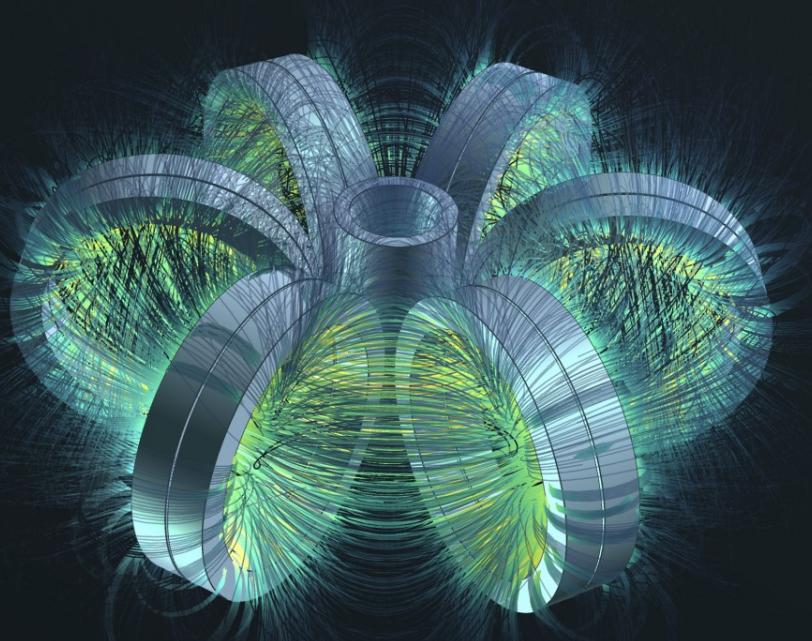
Mesh Calculation



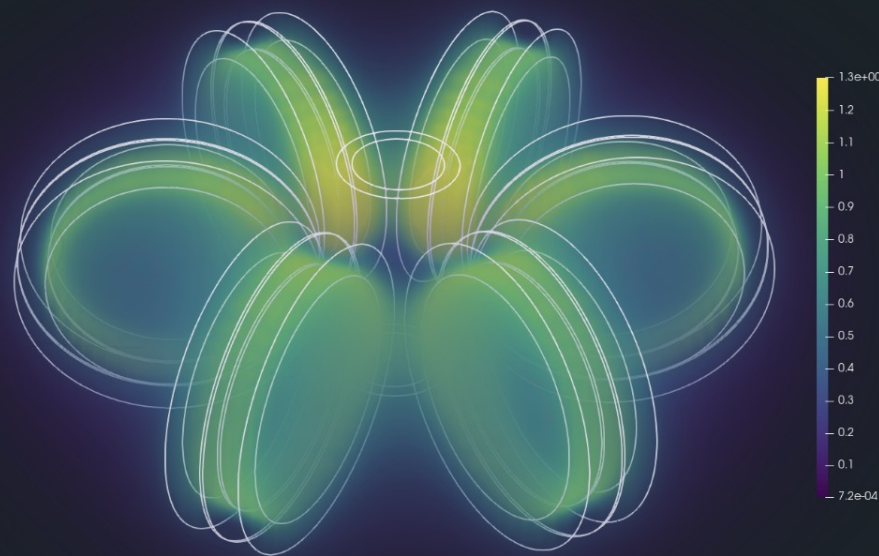
Inductance Calculation



Line Calculation



Field Lines



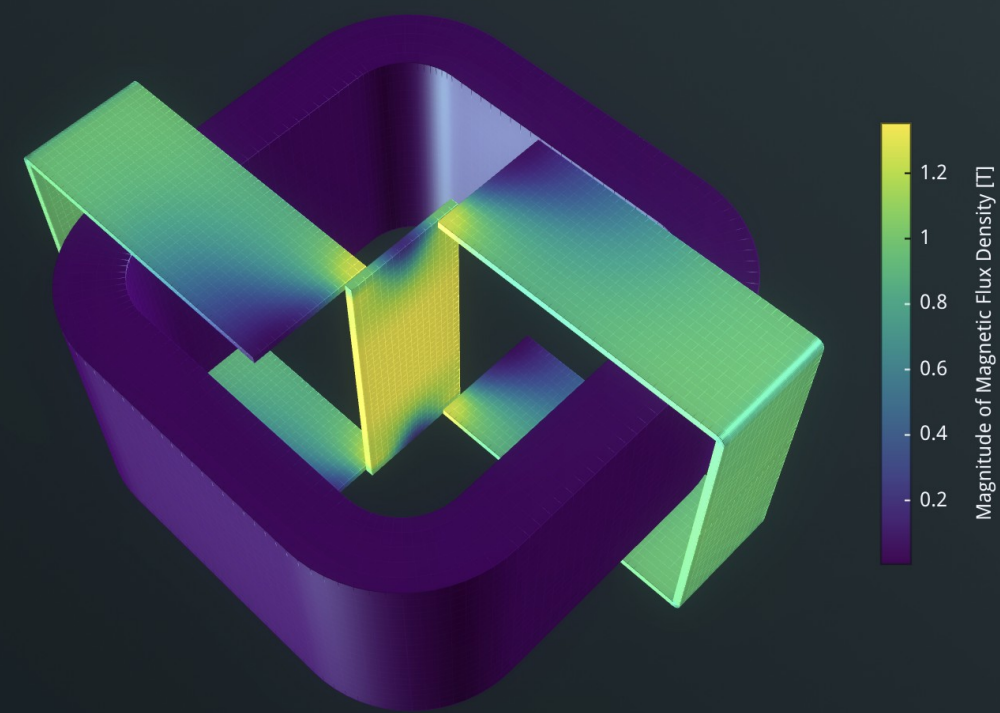
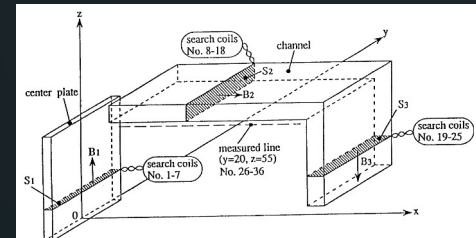
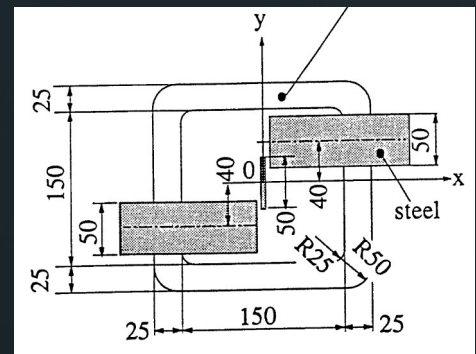
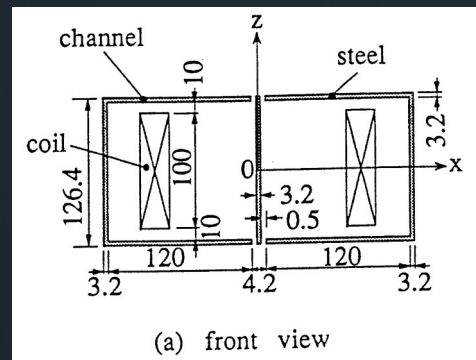
Grid (post in paraview)

And More ...



# How is $\mathbb{R}\Delta \rightarrow$ Validated?

- Numerous regression tests
- Comparing to **Analytical** equations
  - Field on Solenoid axis
  - Field inside magnetized sphere
  - Field of iron sphere in background field
- Comparison to **Soleno**
  - Solenoid field everywhere
- Comparison to **Pre-Calculated** fields
  - Iron ring transformer
- Compumag **TEAM13** problem ->
- All tests are run before updating the code on GIT (next slide)


































# Where can I find $\mathbb{R}\Delta\rightarrow$ ?



**Project-Rat**  

Group ID: 1475626 

-   **Rat-Common**  Contains common files for Rat. ★ 2
-   **Rat-Models**  Coil geometry modeller and field calculation written in C++. ★ 3
-   **Rat-NL**  Nonlinear magnetic materials solver written in C++. ★ 0
-   **Rat-MLFMM**  Matrix based Multi-Level Fast Multipole Method for magn... ★ 4
-   **Materials-CPP**  Implementation in C++ for using material data stored in js... ★ 0
-   **DistMesh-CPP**  Tri- and quad-mesh generator based on distmesh written ... ★ 2
-   **Rat-Docs**  Documentation for Rat. ★ 0
-   **Rat-Template**  Template for setting up a new Rat project. ★ 0

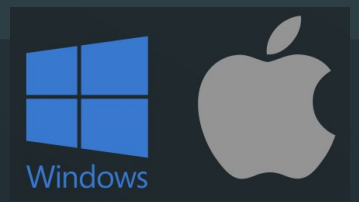
- Open Source Collection of Libraries
  - Permissive **MIT license** model (do what you want!)
  - Calculation Code freely available from GITLAB  
<https://gitlab.com/Project-Rat> 
- Written in object oriented **C++**
- Output in **VTK** File formats (ParaView)





# Graphical User Interface

- Convenient **Graphical User Interface (GUI)** available for sale
- **Supports development of the open source code**

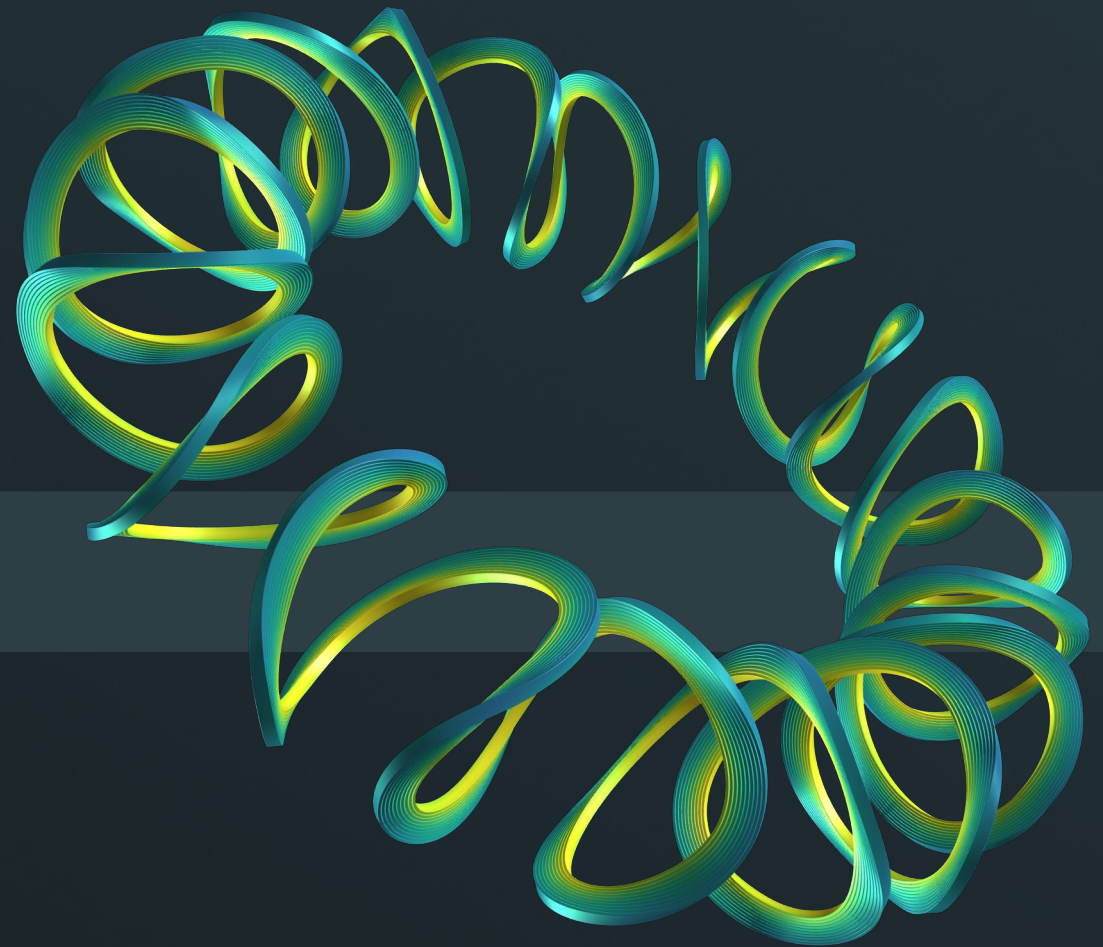


Rat GUI (Developer)

The screenshot displays the Rat GUI (Developer) interface with several active windows:

- Model Editor:** Shows a 3D model of a complex coil structure. The left sidebar contains a 'Model Tree' with categories like 'Central Solenoid', 'Toroid', 'Mirror', and 'Coil'. Below it is a 'Calculation Tree' with 'Mesh', 'Path', 'Inductance', 'Plane', 'Particle Tracks', and 'Field Lines'. A 'Processor' window at the bottom left lists completed tasks: 1 Plane (finished), 2 Mesh (finished), 3 Particle Tracks (finished), 4 Plane (finished), 5 Inductance (finished), and 6 Path (finished).
- Path Calculation - Path:** A graph showing 'Magnetic Flux Density [T]' vs 'Position Along Line [m]'. The y-axis ranges from -1.5 to 1.5, and the x-axis from 0 to 0.5. Three data series are plotted: Bx (blue), By (green), and Bz (yellow).
- Mesh Calculation - Mesh:** A 3D visualization of the coil structure with a color scale for 'Magnitude of Magnetic Flux Density [T]' ranging from 0.2 to 1.2.
- Inductance Calculation - Inductance:** A table showing the 'Inductance Matrix in [mH] // Total Inductance: 32.007'.

Name	Central Solenoid	Lm00_Coil_A	Lm00_Coil_B
Central Solenoid	0.112	-0.000	-0.000
Lm00_Coil_A	-0.000	1.285	0.787
Lm00_Coil_B	-0.000	0.787	1.285
Lm01_Coil_A	-0.000	0.107	0.162
Lm01_Coil_B	-0.000	0.077	0.107
Lm02_Coil_A	-0.000	0.025	0.031
Lm02_Coil_B	-0.000	0.022	0.025
Lm03_Coil_A	0.000	0.014	0.015
Lm03_Coil_B	0.000	0.015	0.014
Lm04_Coil_A	0.000	0.025	0.022
Lm04_Coil_B	0.000	0.031	0.025
Lm05_Coil_A	-0.000	0.107	0.077
Lm05_Coil_B	-0.000	0.162	0.107
Total	0.112	2.658	2.658
- Plane Calculation - Plane:** A 2D plot of 'Norm of Magnetic Flux Density [T]' vs 'x [m]' and 'y [m]' at 'Time: 0 [s]'. The plot shows a central peak with surrounding lobes.
- Track Calculation - Particle Tracks:** A 3D visualization of particle tracks around the coil structure, with a color scale for 'Magnitude of Magnetic Flux Density [T]' ranging from 0.2 to 1.4.



# Features

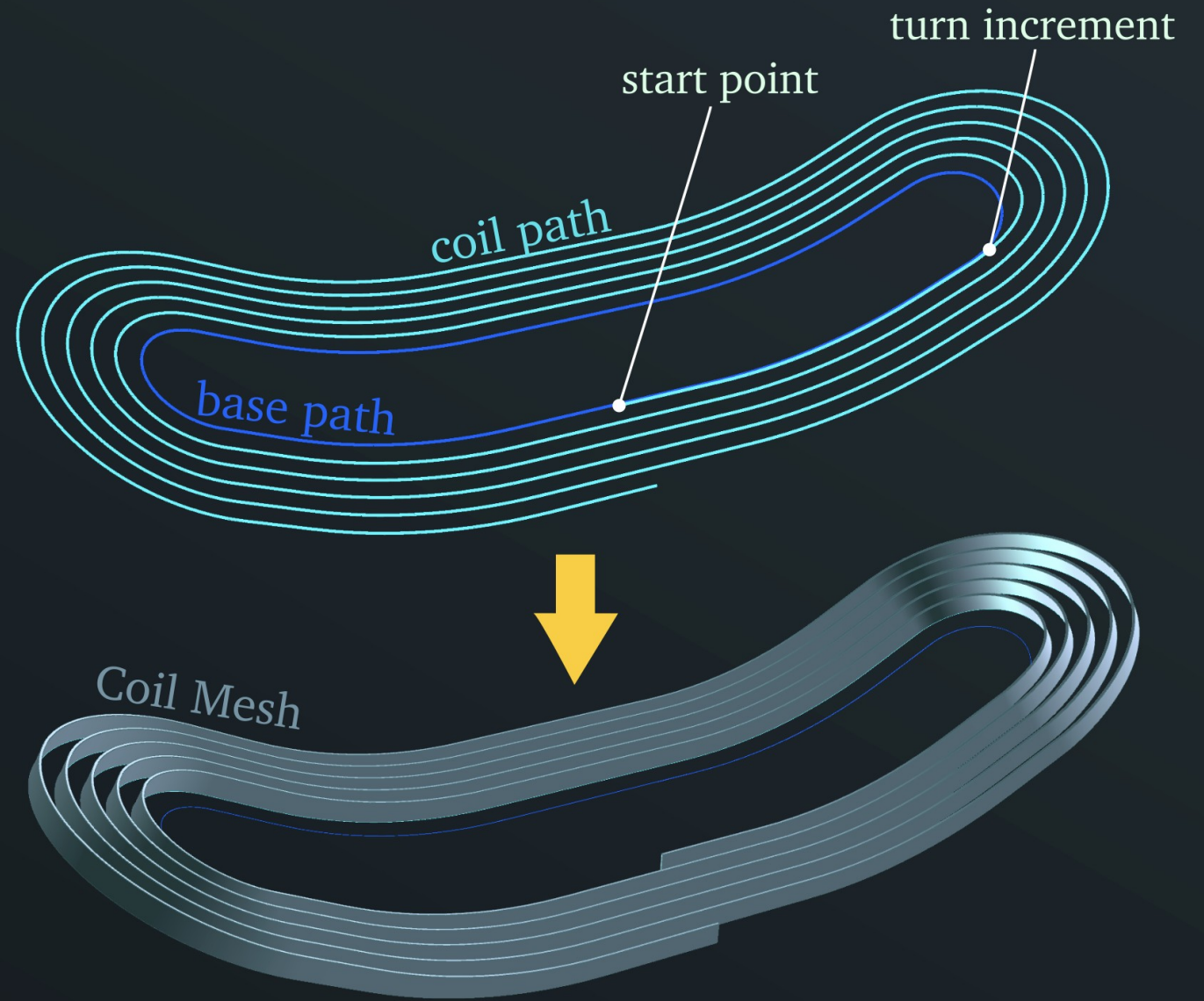
Useful features available in RAT

# Algorithm

The **coiling algorithm** takes a base path and winds a coil path on top of it.

The position of the start of the winding, the increment of the turn and the end of the winding can be specified.

After creating a path for the coil it can be used together with a **cross section** to draw the cable.

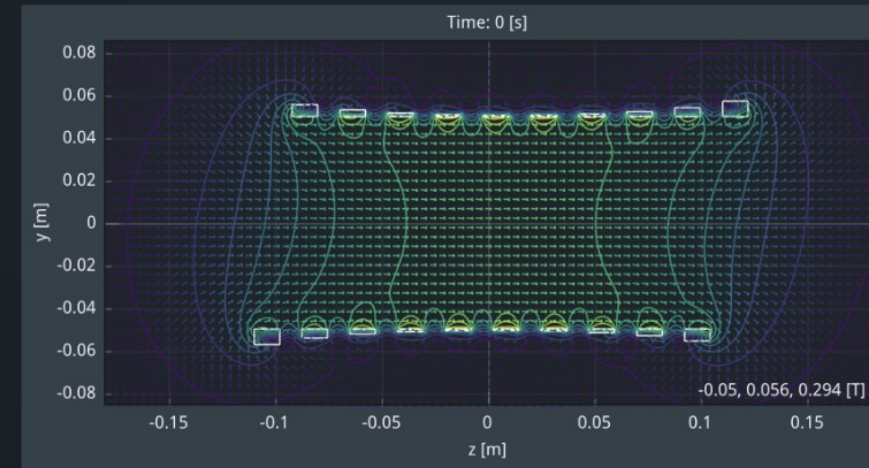
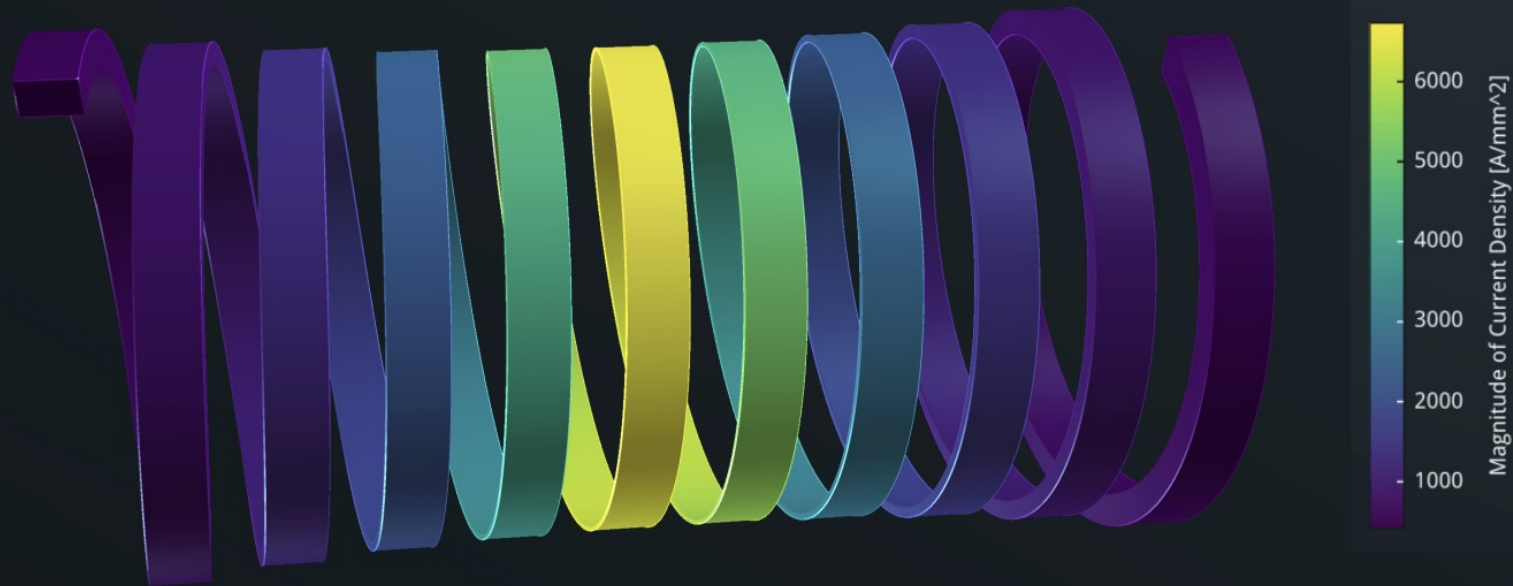


# Longitudinal Grading

Especially in HTS coils it is found that grading the cable along its length by locally changing the number of tapes can save up to **factor 2** of conductor.

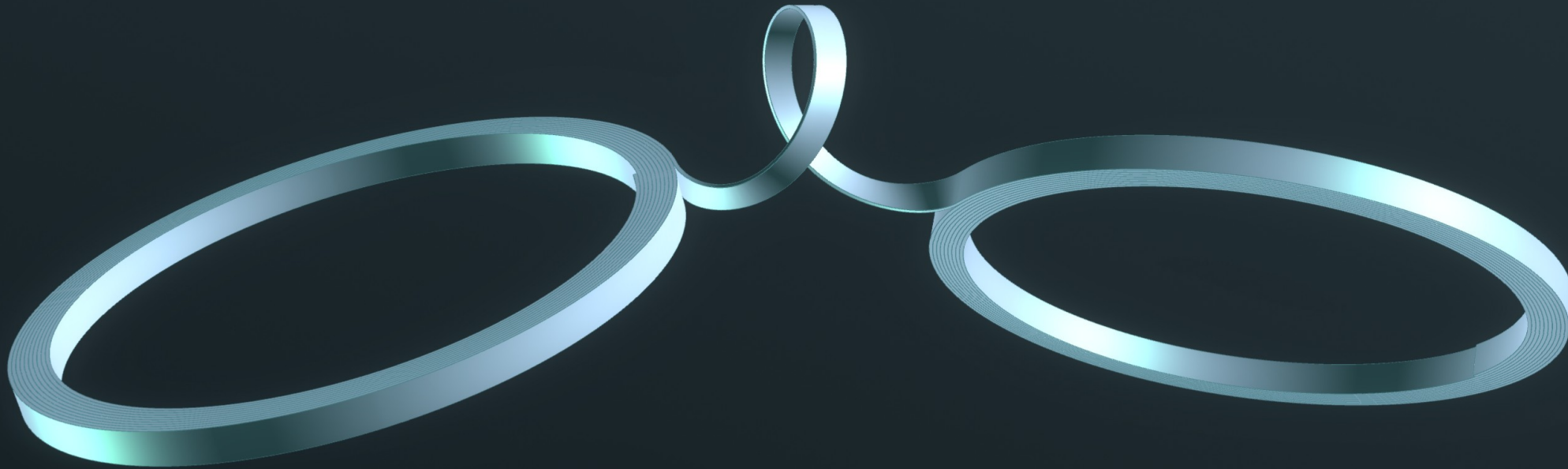
Rat allows cables with varying thickness while correctly preserving current.

See here an example of a **graded HTS solenoid**.





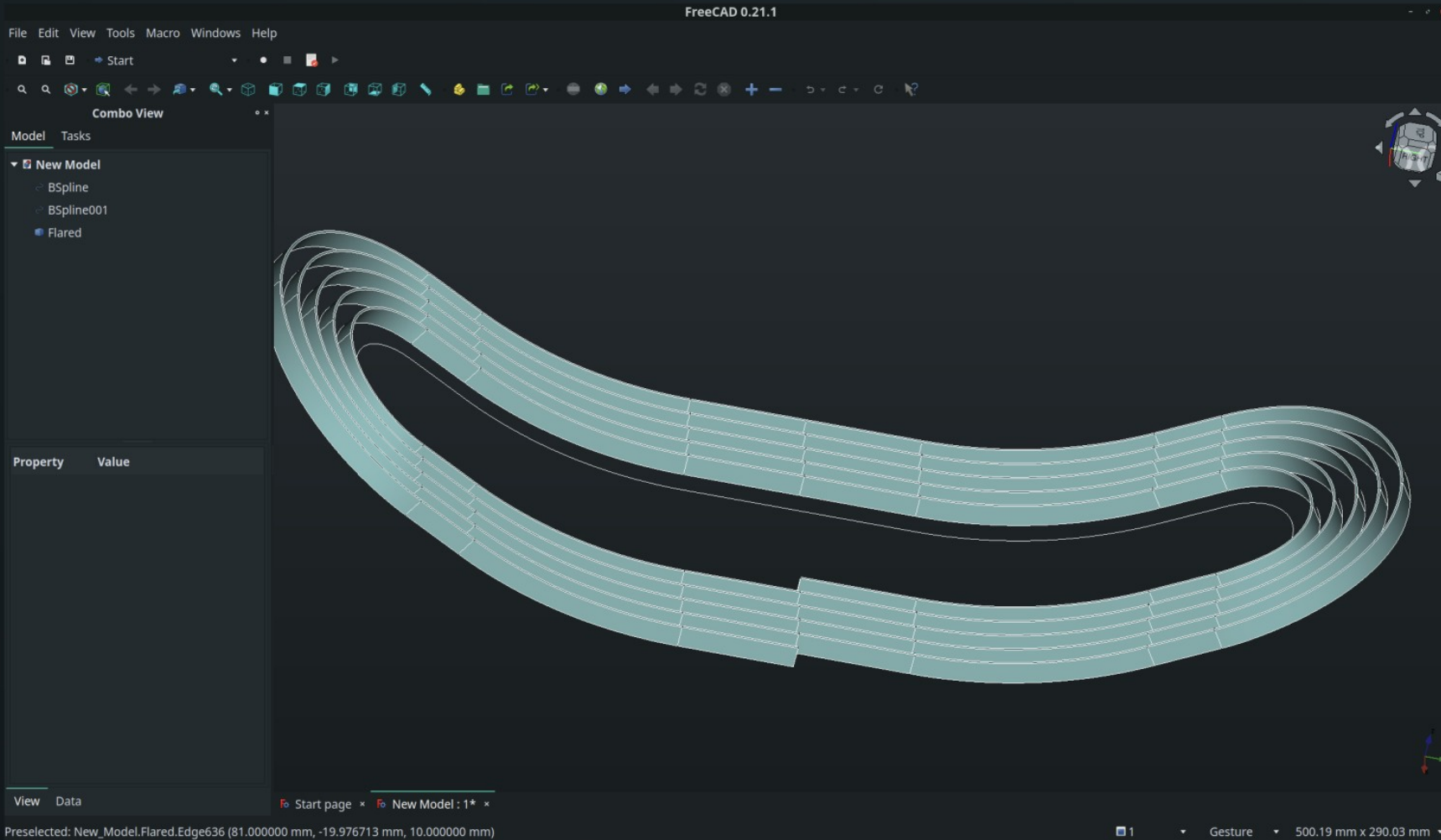
# Connectors



- Make **hard-way bend free** connection between two paths
- Achieved by combining **Bezier** splines and **Frenet-Serret** frames
- The control points for the spline are optimized by minimizing the strain energy while constraining
  - Length of the spline
  - Minimum bending radius
  - Edge regression at specified width
- The order of the spline determines the complexity

# FreeCAD Macro Export

Most coil geometries can be exported to FreeCAD through the use of a **.FCMacro (python script)** file that can be automatically generated by RAT.

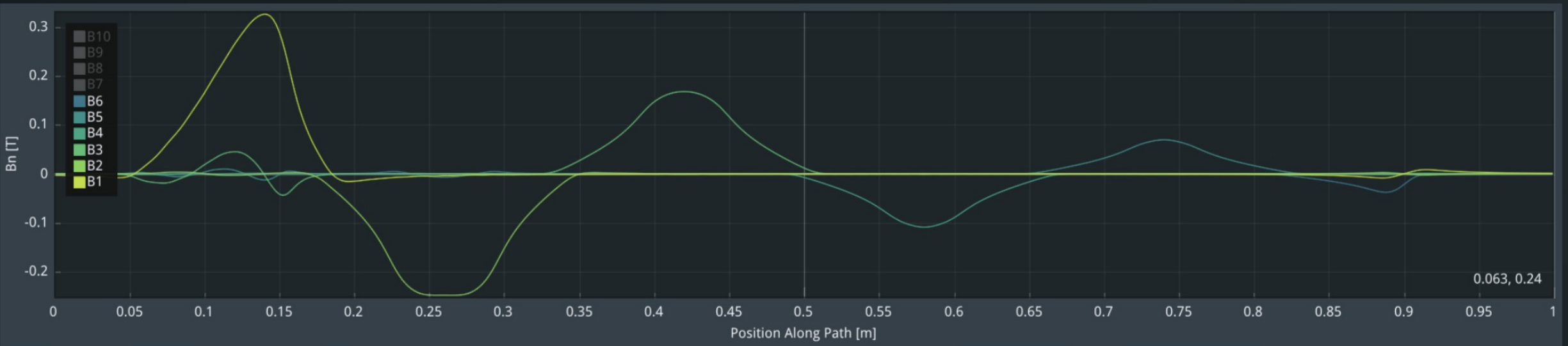
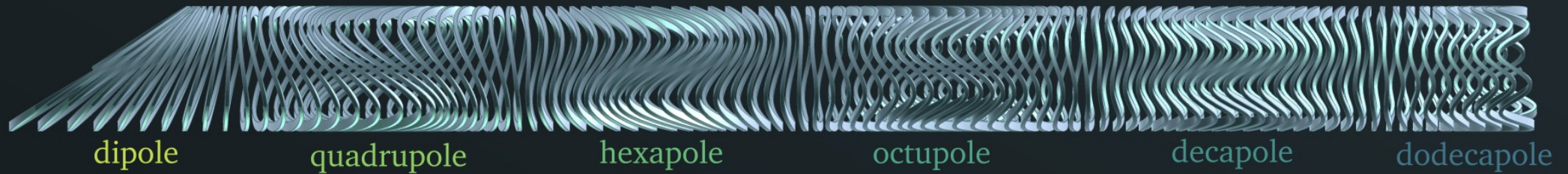


## FCMacro File

```
1 import FreeCAD,Draft,Part,Import
2 doc = App.newDocument("New Model")
3 doc.UndoMode = 0
4 export=[]
5 parts=[]
6 lines=[]
7 group=[]
8 # section from ruled surfaces
9 p0 = FreeCAD.Vector(30.000000000000000,0.000000000000000,0.000000000000000)
10 p1 = FreeCAD.Vector(30.000000000000000,2.000000000000000,0.000000000000000)
11 p2 = FreeCAD.Vector(30.000000000000000,4.000000000000000,0.000000000000000)
12 p3 = FreeCAD.Vector(30.000000000000000,6.000000000000000,0.000000000000000)
13 p4 = FreeCAD.Vector(30.000000000000000,8.000000000000000,0.000000000000000)
14 p5 = FreeCAD.Vector(30.000000000000000,10.000000000000000,0.000000000000000)
15 p6 = FreeCAD.Vector(30.000000000000000,12.000000000000000,0.000000000000000)
16 p7 = FreeCAD.Vector(30.000000000000000,14.000000000000000,0.000000000000000)
17 p8 = FreeCAD.Vector(30.000000000000000,16.000000000000000,0.000000000000000)
18 p9 = FreeCAD.Vector(30.000000000000000,18.000000000000000,0.000000000000000)
19 p10 = FreeCAD.Vector(30.000000000000000,20.000000000000000,0.000000000000000)
20 p11 = FreeCAD.Vector(30.000000000000000,22.000000000000000,0.000000000000000)
21 p12 = FreeCAD.Vector(30.000000000000000,24.000000000000000,0.000000000000000)
22 p13 = FreeCAD.Vector(30.000000000000000,26.000000000000000,0.000000000000000)
23 p14 = FreeCAD.Vector(30.000000000000000,28.000000000000000,0.000000000000000)
24 p15 = FreeCAD.Vector(30.000000000000000,30.000000000000000,0.000000000000000)
25 p16 = FreeCAD.Vector(30.000000000000000,32.000000000000000,0.000000000000000)
26 p17 = FreeCAD.Vector(30.000000000000000,34.000000000000000,0.000000000000000)
27 p18 = FreeCAD.Vector(30.000000000000000,36.000000000000000,0.000000000000000)
28 p19 = FreeCAD.Vector(30.000000000000000,38.000000000000000,0.000000000000000)
29 p20 = FreeCAD.Vector(30.000000000000000,40.000000000000000,0.000000000000000)
30 p21 = FreeCAD.Vector(30.000000000000000,42.000000000000000,0.000000000000000)
31 p22 = FreeCAD.Vector(30.000000000000000,44.000000000000000,0.000000000000000)
32 p23 = FreeCAD.Vector(30.000000000000000,46.000000000000000,0.000000000000000)
33 p24 = FreeCAD.Vector(30.000000000000000,48.000000000000000,0.000000000000000)
34 p25 = FreeCAD.Vector(30.000000000000000,50.000000000000000,0.000000000000000)
35 spline0 = Draft.make_bspline([p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,p21,p22,p23,p24,p25])
36 spline0.ViewObject.LineColor = (1.0,1.0,1.0,0.0)
37 spline0.ViewObject.LineWidth = 0.5
38 lines.append(spline0)
39 # section from ruled surfaces
40 p0 = FreeCAD.Vector(30.000000000000000,50.000000000000000,0.000000000000000)
41 p1 = FreeCAD.Vector(30.000000000000000,51.9758122962931424,0.0097598237108532)
42 p2 = FreeCAD.Vector(30.000000000000000,53.9514317567893045,0.0390383423018337)
43 p3 = FreeCAD.Vector(30.000000000000000,55.9266655645119499,0.0878326982411304)
44 p4 = FreeCAD.Vector(30.000000000000000,57.9013209401235684,0.1561381292856323)
45 p5 = FreeCAD.Vector(30.000000000000000,59.8752051607405988,0.2439479689456681)
46 p6 = FreeCAD.Vector(30.000000000000000,61.8481255787428665,0.3512536471357076)
47 p7 = FreeCAD.Vector(30.000000000000000,63.8198896405755960,0.4780446910107373)
48 p8 = FreeCAD.Vector(30.000000000000000,65.7903049055423708,0.6243087259883541)
49 p9 = FreeCAD.Vector(30.000000000000000,67.7591790645869594,0.7900314769566209)
50 p10 = FreeCAD.Vector(30.000000000000000,69.7263199590623515,0.9751967696671748)
51 p11 = FreeCAD.Vector(30.000000000000000,71.6915355994851069,1.1797865323138981)
52 p12 = FreeCAD.Vector(30.000000000000000,73.6546341842731636,1.4037807972966165)
53 p13 = FreeCAD.Vector(30.000000000000000,75.6154241184653841,1.6471577031699214)
54 p14 = FreeCAD.Vector(30.000000000000000,77.5737140324208667,1.9098934967768821)
55 p15 = FreeCAD.Vector(30.000000000000000,79.5293128004962426,2.1919625355671934)
56 p16 = FreeCAD.Vector(30.000000000000000,81.4820295596992707,2.4933372900999511)
57 p17 = FreeCAD.Vector(30.000000000000000,83.4316737283166674,2.8139883467303894)
58 p18 = FreeCAD.Vector(30.000000000000000,85.3780550245145946,3.1538844104806345)
59 p19 = FreeCAD.Vector(30.000000000000000,87.3209834849098030,3.5129923088940983)
60 p20 = FreeCAD.Vector(30.000000000000000,89.2602694831097097,3.8912769912730072)
61 p21 = FreeCAD.Vector(30.000000000000000,91.1957237482196774,4.2887015400991935)
62 p22 = FreeCAD.Vector(30.000000000000000,93.1271573833153923,4.7052271666372834)
63 p23 = FreeCAD.Vector(30.000000000000000,95.0543818838790031,5.1408132187204423)
64 p24 = FreeCAD.Vector(30.000000000000000,96.9772091561967358,5.5954171839178510)
65 p25 = FreeCAD.Vector(30.000000000000000,98.8954515357165320,6.0689946936839165)
66 p26 = FreeCAD.Vector(30.000000000000000,100.8089218053638234,6.5614995276885857)
67 p27 = FreeCAD.Vector(30.000000000000000,102.7174332138135213,7.0728836183282517)
68 p28 = FreeCAD.Vector(30.000000000000000,104.6207994937166603,7.6030970554172406)
69 p29 = FreeCAD.Vector(30.000000000000000,106.5188348798797762,8.1520880910587401)
70 p30 = FreeCAD.Vector(30.000000000000000,108.4113541273951711,8.7198031446955167)
71 p31 = FreeCAD.Vector(30.000000000000000,110.2981725297205884,9.3061868083391452)
72 p32 = FreeCAD.Vector(30.000000000000000,112.1791059367062644,9.9111818519777923)
```

# Custom CCT

In a Custom CCT Path the harmonic strength  $A_n$  and  $B_n$  can be set along the length of the coil. A superposition of multiple harmonics at the same axial position is possible as well.





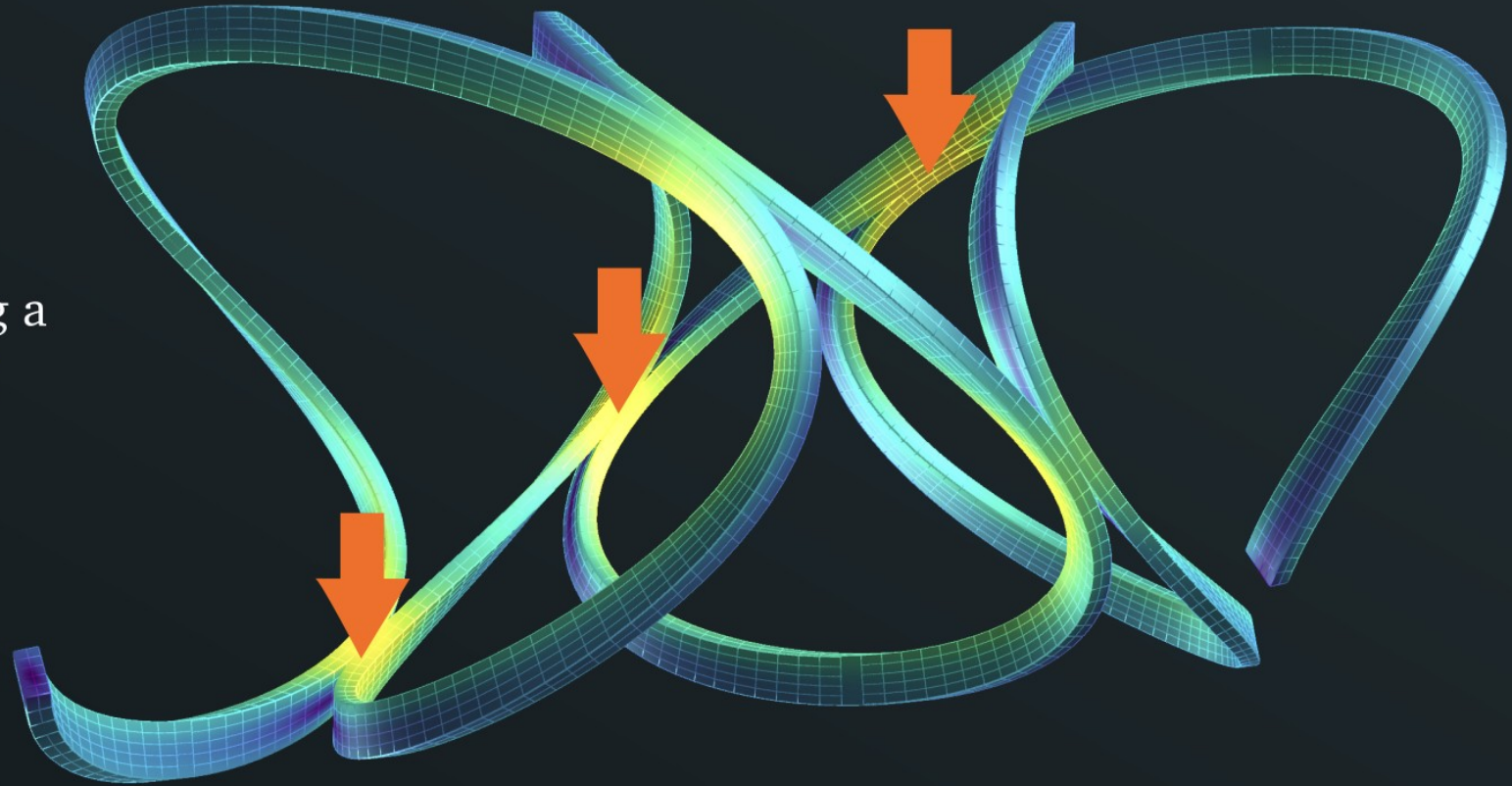
# Controlling CCT Winding Pitch

When using **variable harmonics** along the length of the magnet we would like to maintain **positive spacing** between turns.

First the path is generated with zero pitch causing all turns to coincide

Then the **local pitch** is determined by using a window, **scanning one turn ahead and one turn behind** to find minimum spacing between turns

The pitch then must be numerically integrated to get the z-offset

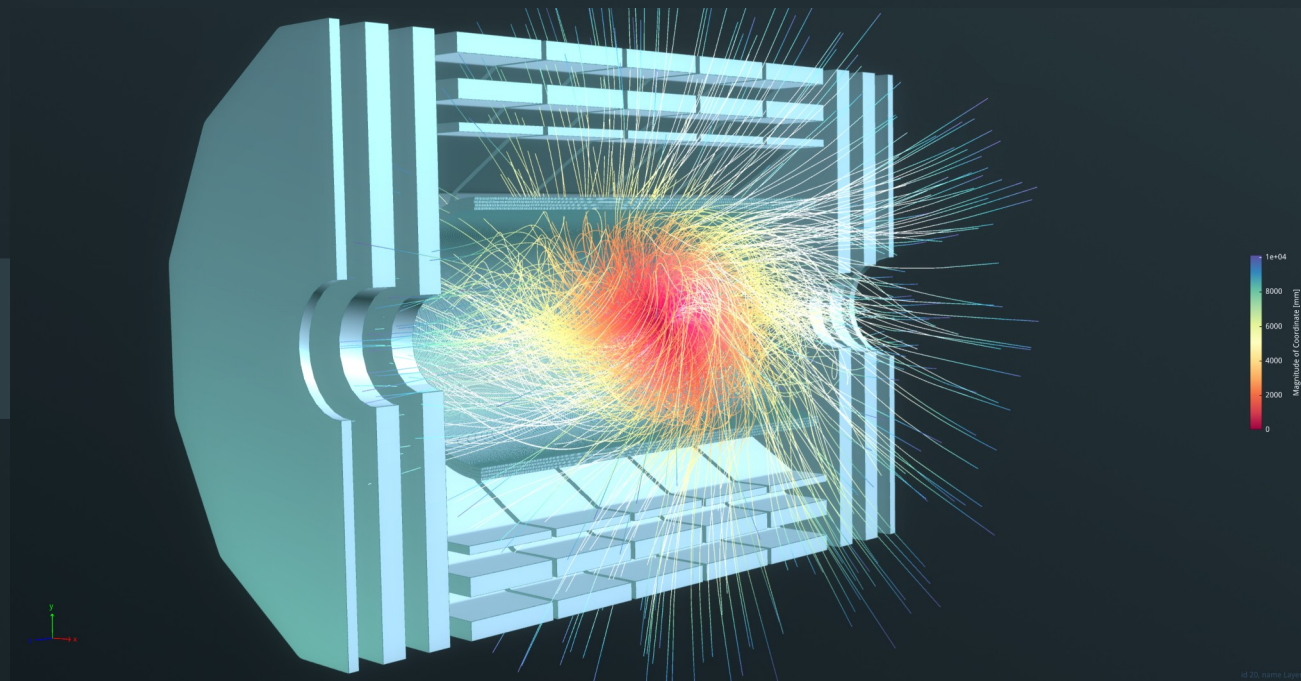




# CMS Example

Modeling CMS Magnet System in RAT

Note: CMS coils modeled here are a loose representation based on info found on internet ...







# Step 1: Modeling the CMS solenoid

## Cross-Section

▼ Rectangle Cross-Section ?

▼ Appearance

Rectangle name

▼ Geometry

0.0 mm - + nc1 ?

300 mm - + nc2 ?

-6250 mm - + dc1 ?

6250 mm - + dc2 ?

▼ Discretization

100 mm - + dnormal ?

100 mm - + dtrans ?

## Path

▼ Circle Path

▼ Appearance

Circle name

▼ Geometry

2950 mm - + radius ?

hardway ?

▼ Discretization

4 - + nsections ?

300 mm - + offset ?

100 mm - + dl ?

## Custom Coil

▼ Custom Coil

▼ Appearance

Solenoid name

custom color ?

30 56 59 color ?

▼ Connectivity

0 - + circuit id ?

▼ Physics

2160 - + nturns ?

use current density ?

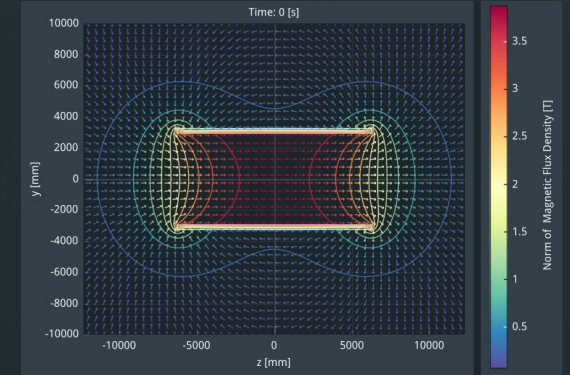
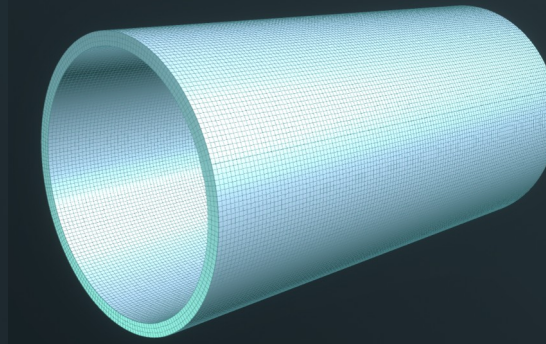
19500 A - + current ?

4.5 K - + temperature ?

140 pct - + softness ?

2 - + ngauss ?

=



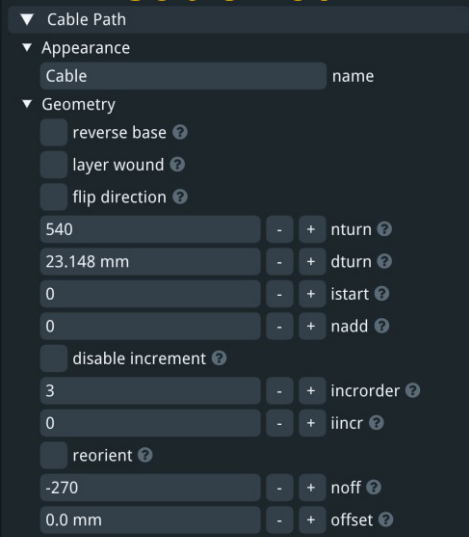
- Use a **Circle** Path
  - Radius 2950 mm
- Use a **Rectangle** Cross Section
  - From 0 to 300 mm in the normal direction
  - From -6250 to +6250 in the transverse direction
- Element sizes set to 100 mm

# Step 2: Adding Turn Detail

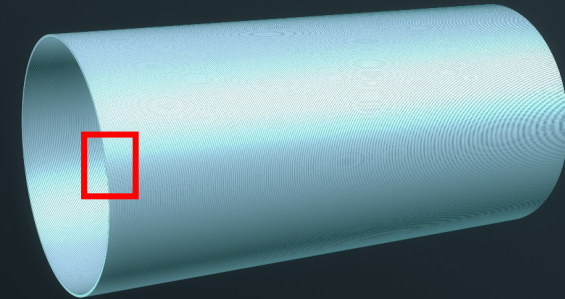
## Circle Path



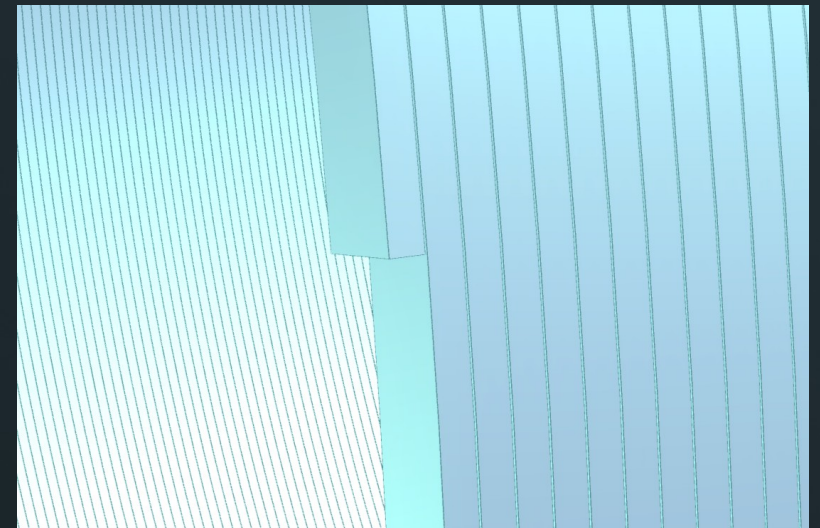
## Cable Path



## Custom Coil



=



- Add the Circle Path to a **Cable Path** to make individual coil windings
- The cross section is now set to the cross section of a cable
- The number of turns is now set to 1
- Other layers are made using increasing circle radii and alternating reverse base



# Step 3: End Cap Yoke Plate Cross Section

## Polygon DF

▼ Reg. Polygon Distance Function

▼ Appearance

Regular Polygon name

▼ Geometry

12 - + nsides ?

6910 mm - + alpha ?

400 mm - + radius ?

## Circle DF

▼ Circle Distance Function ?

▼ Appearance

Circle name

▼ Geometry

0.0 mm - + nc ?

0.0 mm - + dc ?

1100 mm - + radius ?

## Difference DF

▼ Difference Distance Function ?

▼ Appearance

Difference name

## Custom Cross-Section

▼ Distmesh Cross-Section ?

▼ Appearance

Distmesh name

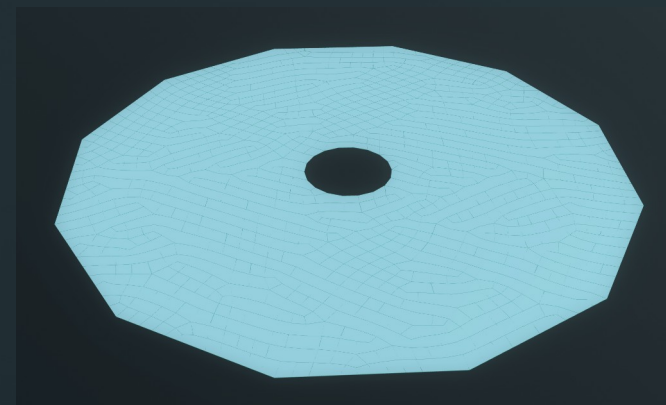
▼ Geometry

5000 - + nodelim ?

500 mm - + h0 ?

1001 - + rng seed ?

20 pct - + rng strength ?



- Use a Distmesh Cross Section to access the built-in mesher
  - Use a **Difference** distance function
  - Add a 12 sided **Polygon** with radius 6910 mm
  - Subtract a **Circle** with radius 1100, 1300 and 1500 mm for the plates respectively
  - Element size 500 mm

# Step 4: Extrude the Mesh along Axis

## Custom Cross-Sec.

▼ Dismesh Cross-Section ?

▼ Appearance

Distmesh name

▼ Geometry

5000 - + nnodelim ?

500 mm - + h0 ?

1001 - + rng seed ?

20 pct - + rng strength ?

## Axial Path

▼ Axial Path

▼ Appearance

Axis name

▼ Geometry

Orientation

x  y  z axis ?

x  y  z transverse ?

Position

0.0 mm - + x ?

0.0 mm - + y ?

7565 mm - + z ?

600 mm - + ell ?

▼ Discretization

400 mm - + dl ?

## Custom Iron Mesh

▼ Custom HB-Mesh ?

▼ Appearance

Plate 1 name

custom color ?

0 62 55 color ?

▼ Geometry

use tetrahedrons ?

▼ Physics

4.5 K - + temperature ?

2 - + vngauss ?

2 - + sngauss ?

## HB-Curve

▼ Vinh Le-Van HB-Curve ?

▼ Appearance

HB-Curve Vinh Le-Van name

▼ Fit Parameters

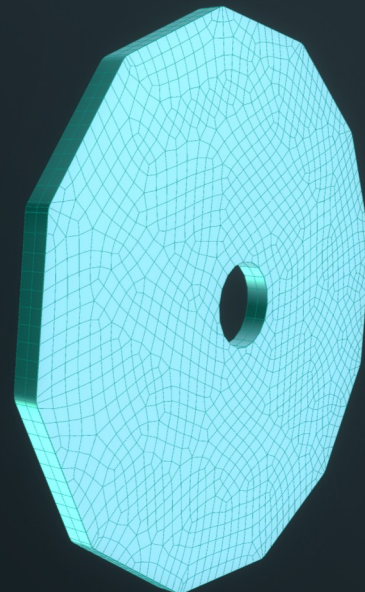
500 H/m - +  $\mu$ r ?

2 T - + Js ?

0.06 MA/m - + H2 ?

100 - + npoints ?

100 pct - + ffill ?



- Use a Custom HB Mesh
  - Add the Dismesh Cross Section that we created in the previous step
  - Add and an axial path along the z-axis of length 600 mm
  - Add an HB curve fit with saturation of 2 T
- Other end-plates are extruded in a similar way



# Step 5: Mirror the Yoke's End Cap Plate

## Custom Iron Mesh

▼ Custom HB-Mesh ?

▼ Appearance

Plate 1 name

custom color ?

0 62 55 color ?

▼ Geometry

use tetrahedrons ?

▼ Physics

4.5 K - + temperature ?

2 - + vngauss ?

2 - + sngauss ?

## Mirror Group

▼ Mirror

▼ Appearance

End Cap Yoke Plates name

custom color ?

30 56 59 color ?

▼ Geometry

keep original ?

anti mirror ?

Origin

0.0 mm - + x ?

0.0 mm - + y

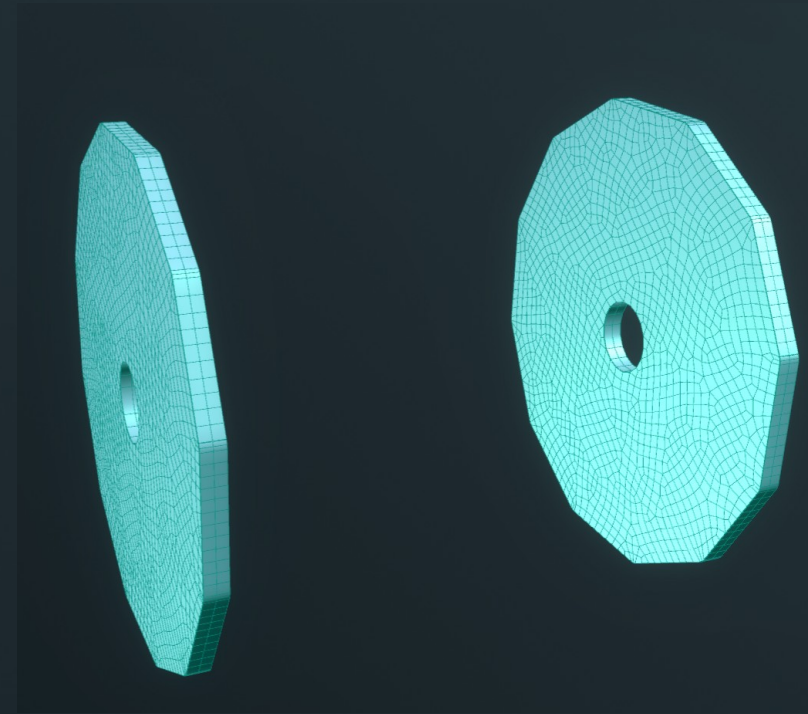
0.0 mm - + z

Plane Vector

0.0 mm - + x ?

0.0 mm - + y

1000 mm - + z



- Using a **Mirror** along the z-axis the end-plate is duplicated to the other side
- All end-plates can be added to the same mirror object





# Step 6: Section of barrel yoke

## Polygon Path

▼ Polygon Path

▼ Appearance

Polygon name

▼ Geometry

12 - + num sides ?

4705 mm - + alpha ?

200 mm - + radius ?

▼ Discretization

0.0 mm - + offset ?

400 mm - + dl ?

## Path Cross-Section

▼ Path Cross-Section ?

▼ Appearance

Cross Path name

▼ Geometry

is line ?

200 mm - + thickness ?

0.0 mm - + offset ?

60 mm - + delem ?



- We could subtract two 12 sided polygons and use the distmesher again
- More regular mesh can be achieved with **Path Cross Section**, which creates a cross section from a path by thickening it
- Other layers made similarly

# Step 7: Extrude the mesh

## Path Cross-Section

▼ Path Cross-Section ?

▼ Appearance

Cross Path name

▼ Geometry

is line ?

200 mm - + thickness ?

0.0 mm - + offset ?

60 mm - + delem ?

## Axial Path

▼ Axial Path

▼ Appearance

Axis name

▼ Geometry

Orientation

x  y  z axis ?

x  y  z transverse ?

Position

0.0 mm - + x ?

0.0 mm - + y

0.0 mm - + z

2536 mm - + ell ?

▼ Discretization

400 mm - + dl ?

## Custom Iron Mesh

▼ Custom HB-Mesh ?

▼ Appearance

Layer 1 name

custom color ?

0 62 55 color ?

▼ Geometry

use tetrahedrons ?

▼ Physics

4.5 K - + temperature ?

2 - + vngauss ?

2 - + sngauss ?

## HB-Curve

▼ Vinh Le-Van HB-Curve ?

▼ Appearance

HB-Curve Vinh Le-Van name

▼ Fit Parameters

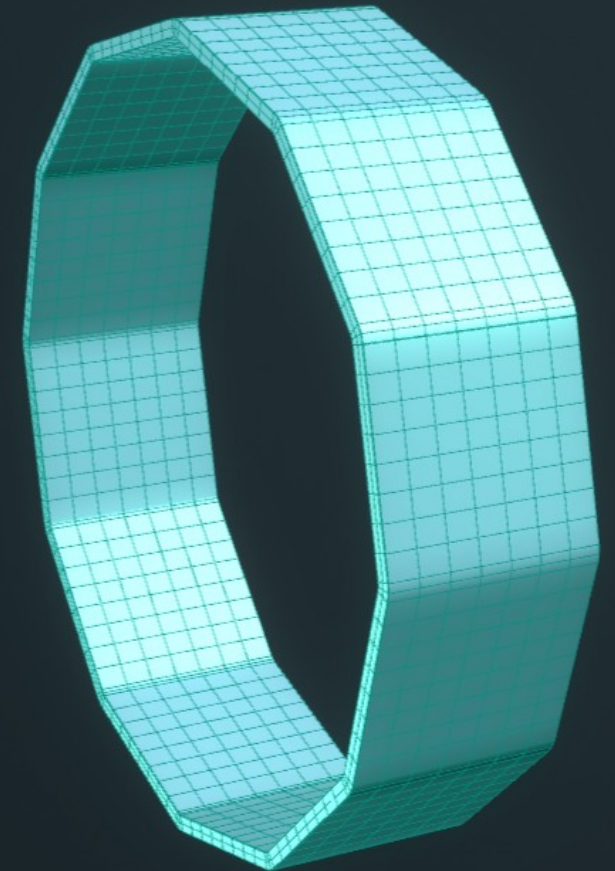

500 H/m - +  $\mu$  ?

2 T - +  $J_s$  ?

0.06 MA/m - +  $H_2$  ?

100 - + npoints ?

100 pct - + ffill ?



- Use a **Custom HB Mesh**
  - Add the Path based Cross Section that we created in the previous step
  - Add and an axial path along the z-axis of length 2536 mm
  - Add an HB curve fit with saturation of  $2 T$
- Other layers are extruded in a similar way



# Step 8: array of Barrel yoke section

## Custom Iron Mesh

## Array of Meshes

▼ Custom HB-Mesh ?

▼ Appearance

Plate 1 name

custom color ?

0 62 55 color ?

▼ Geometry

use tetrahedrons ?

▼ Physics

4.5 K - + temperature ?

2 - + vngauss ?

2 - + sngauss ?



▼ Array

▼ Appearance

Barrel Yoke name

custom color ?

30 56 59 color ?

▼ Geometry

centered ?

alternate ?

1 - + nx ?

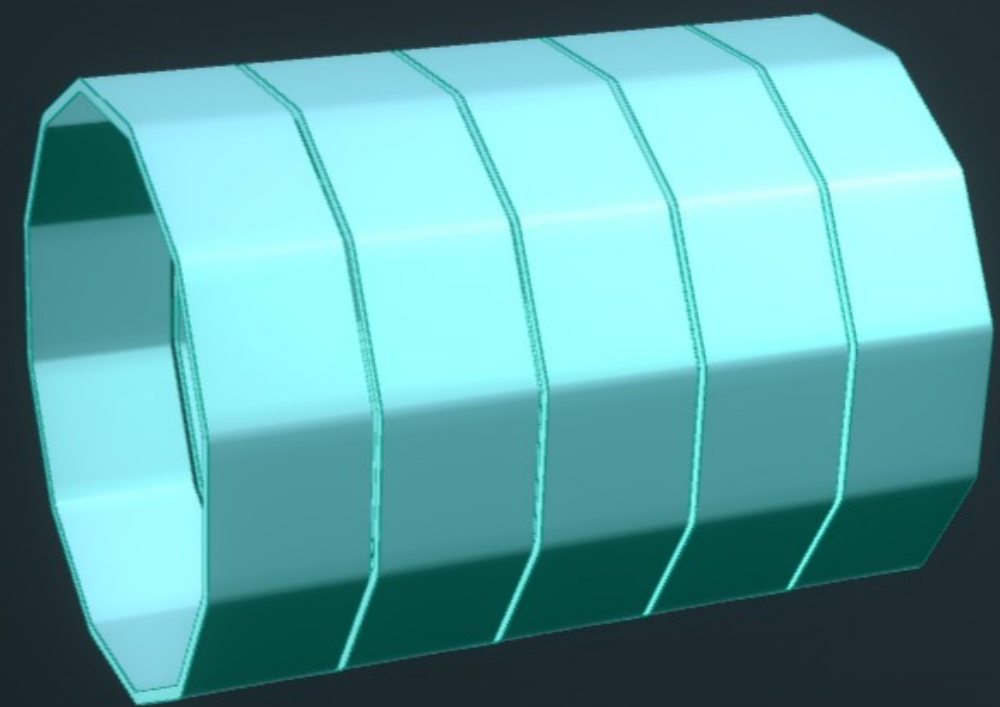
1 - + ny ?

5 - + nz ?

0.0 mm - + dx ?

0.0 mm - + dy ?

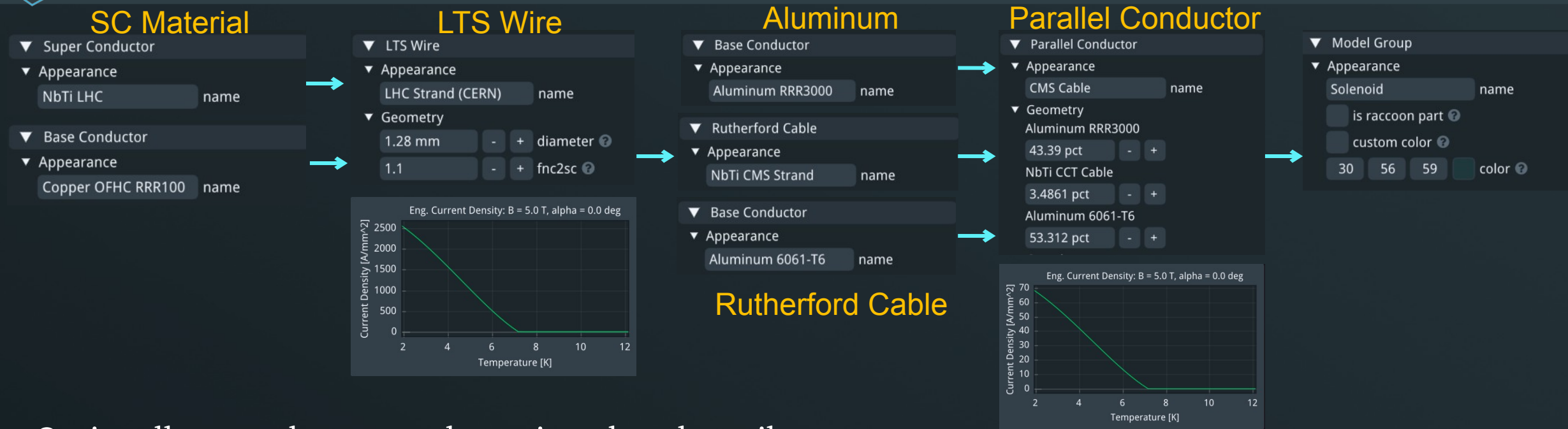
2686 mm - + dz ?



- Using the **Array** along the z-axis the ring is duplicated 5 times
- All layers are included in the same array



# Step 9: Add the CMS Conductor



- Optionally a conductor can be assigned to the coils
  - Base conductor has **k**, **rho**, **Cp** and **d**
  - Superconductor also has **Jc** and **N**
- All conductors can be combined using
  - **Parallel** or **Series** conductors
  - LTS wire and HTS tape are just wrappers around parallel conductor
- This allows for calculation of **margins/critical current** on coil etc.

# Step 10: Combine the Models

## Solenoid

Model Group

Appearance

Solenoid name

custom color ?

30 56 59 color ?

## End Caps

Mirror

Appearance

End Cap Yoke Plates name

custom color ?

30 56 59 color ?

Geometry

keep original ?

anti mirror ?

Origin

0.0 mm - + x ?

0.0 mm - + y ?

0.0 mm - + z ?

Plane Vector

0.0 mm - + x ?

0.0 mm - + y ?

1000 mm - + z ?

## Barrel Yoke

Array

Appearance

Barrel Yoke name

custom color ?

30 56 59 color ?

Geometry

centered ?

alternate ?

1 - + nx ?

1 - + ny ?

5 - + nz ?

0.0 mm - + dx ?

0.0 mm - + dy ?

2686 mm - + dz ?

## Combined Model

Model Group

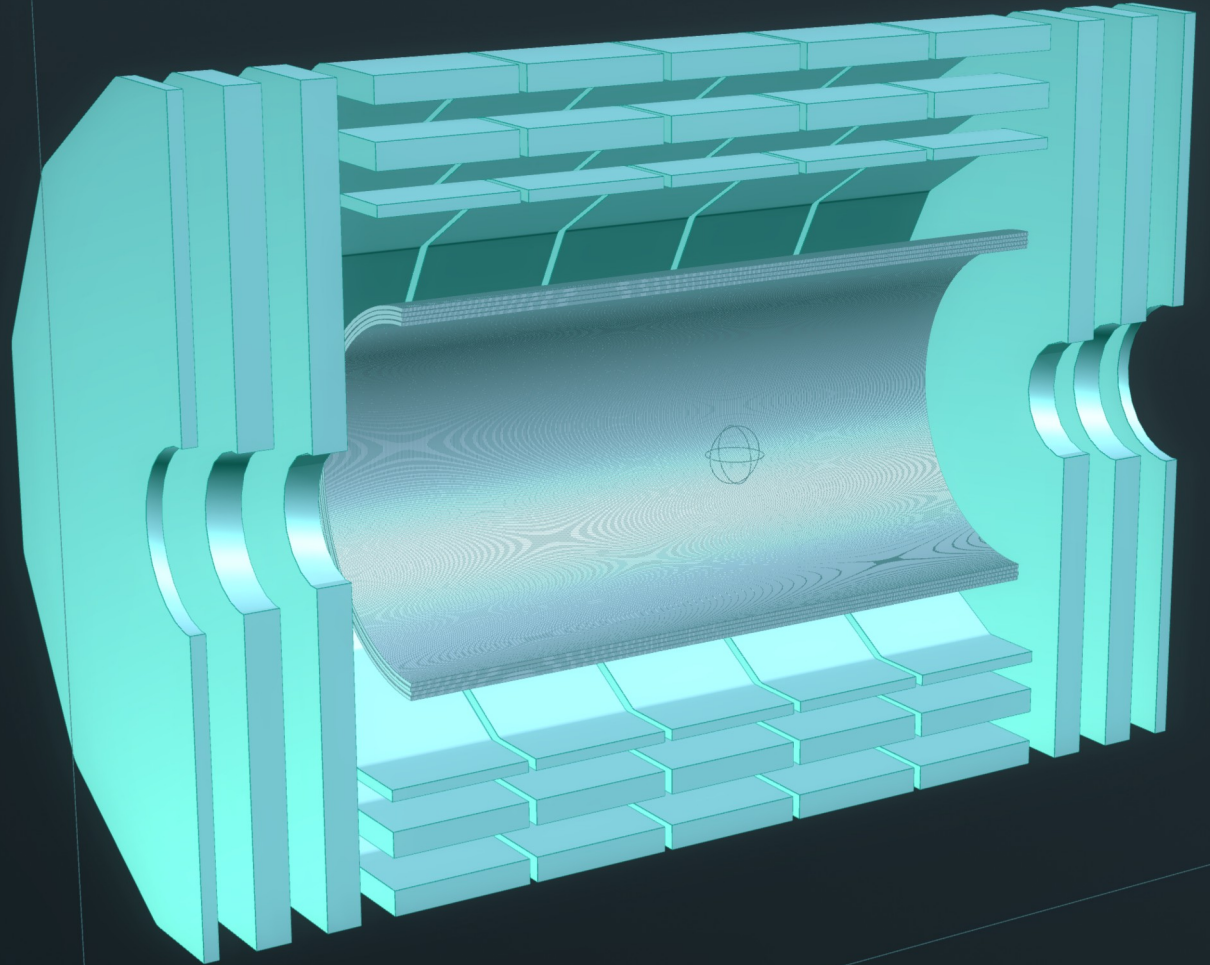
Appearance

Model Tree name

custom color ?

30 56 59 color ?

=



- Using a Model Group all objects can be combined into a single model, ready for calculations

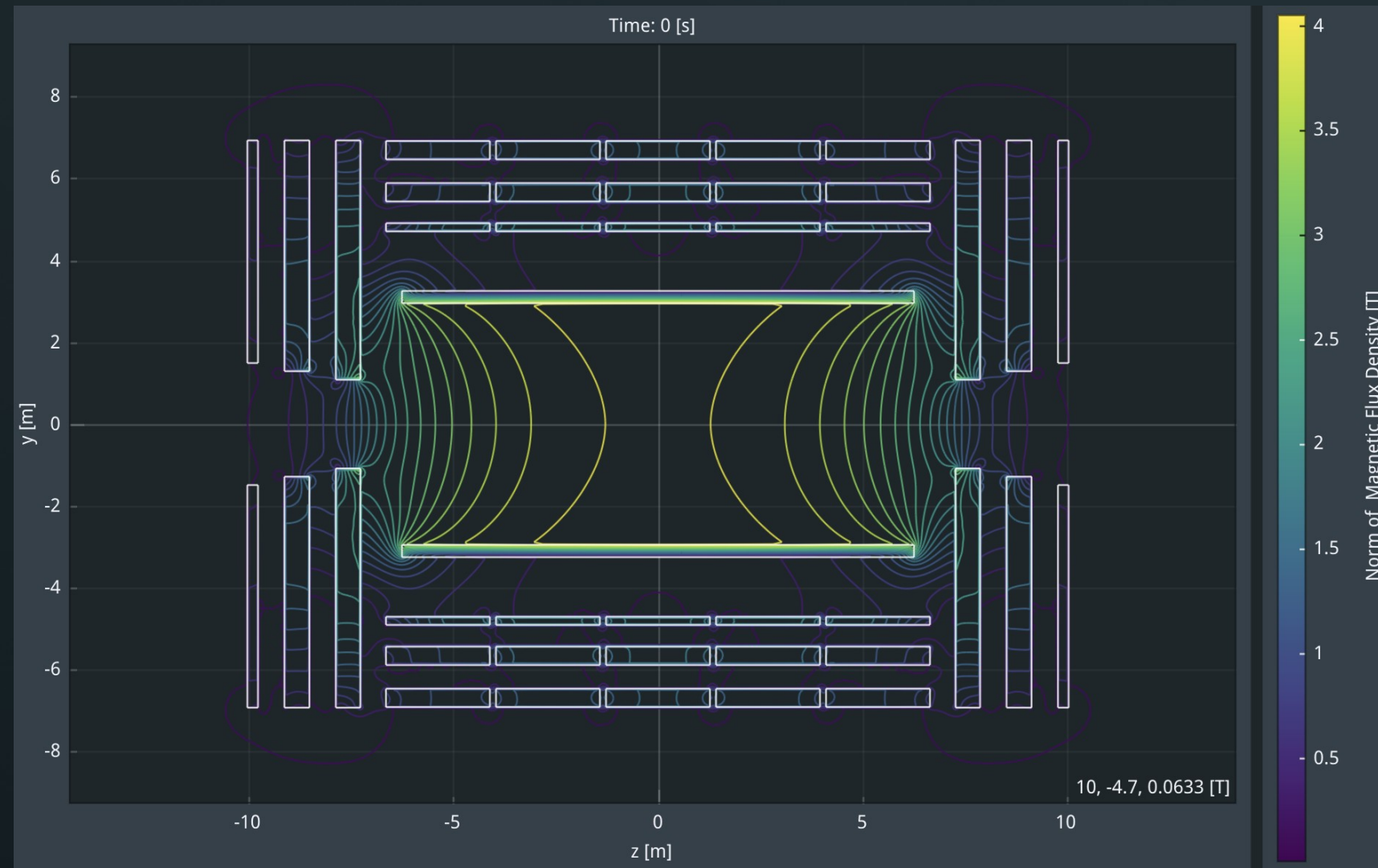


# CMS Calculations

Example calculations on the CMS model



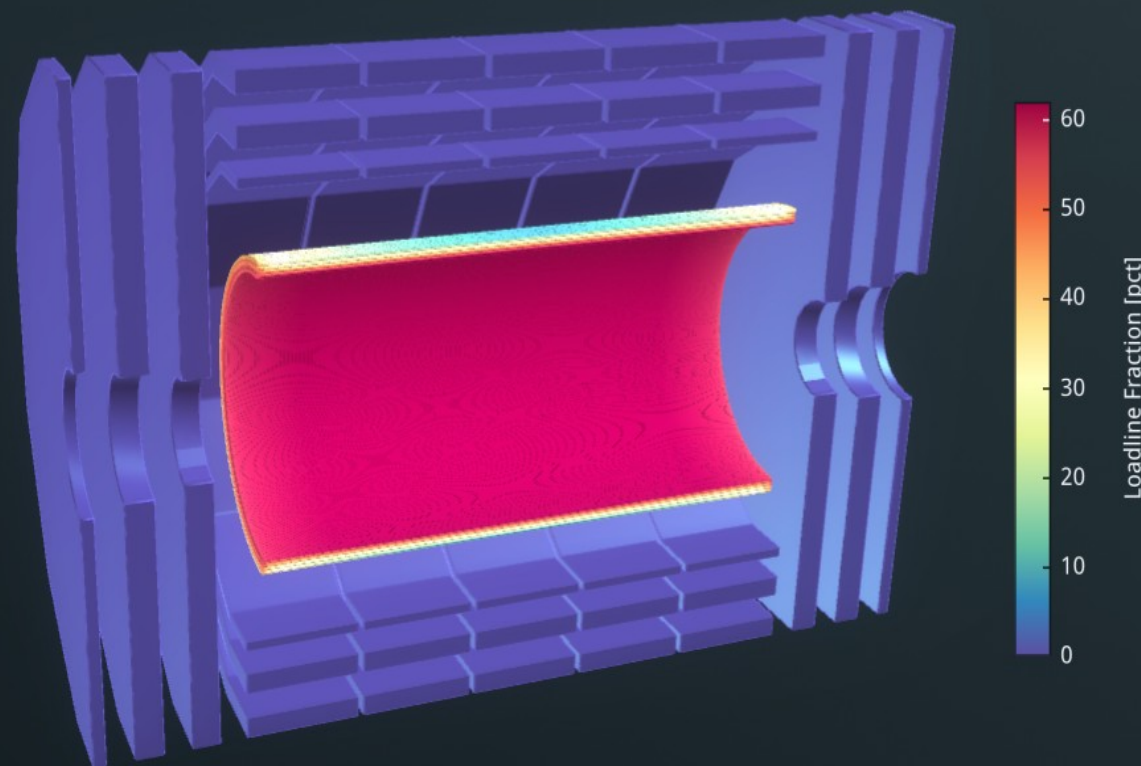
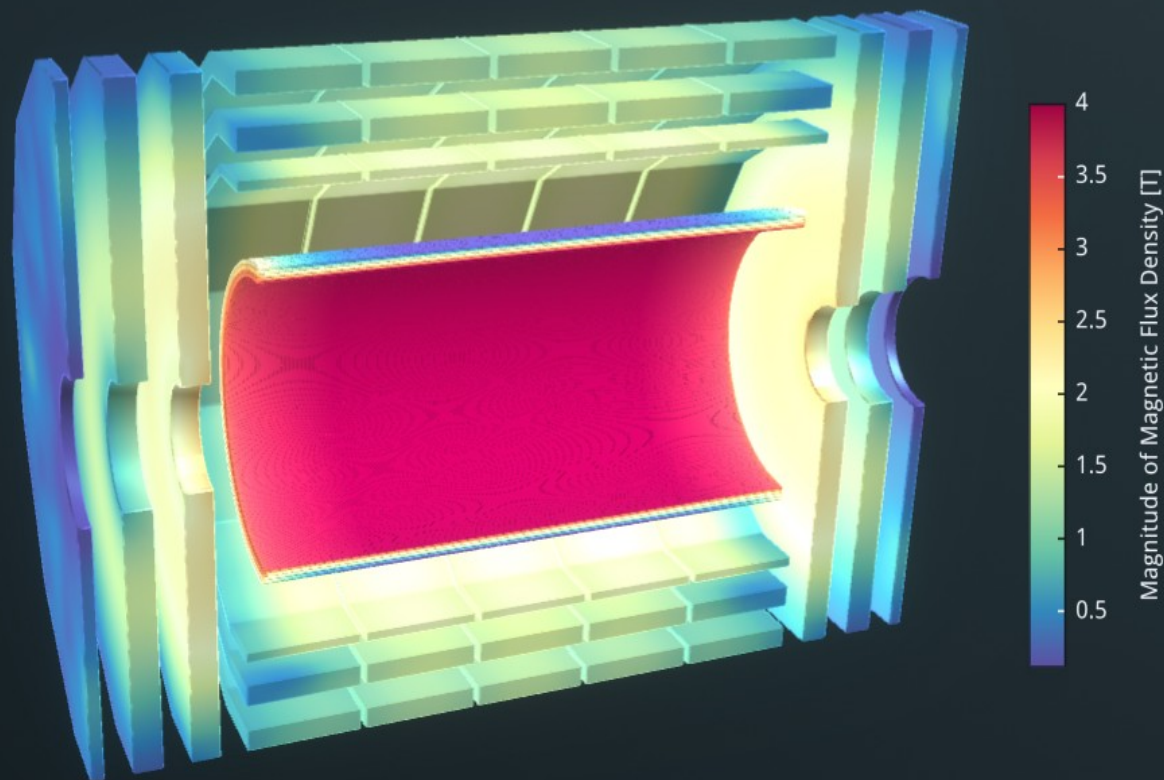
# Plane Calculation



- Connect the model to **Plane Calculation**
- Plane is defined by
  - orientation YZ
  - Size **24** by **18 m**
  - Number of pixels **800** by **600**
- Field at center **4.0 T**



# Mesh Calculation

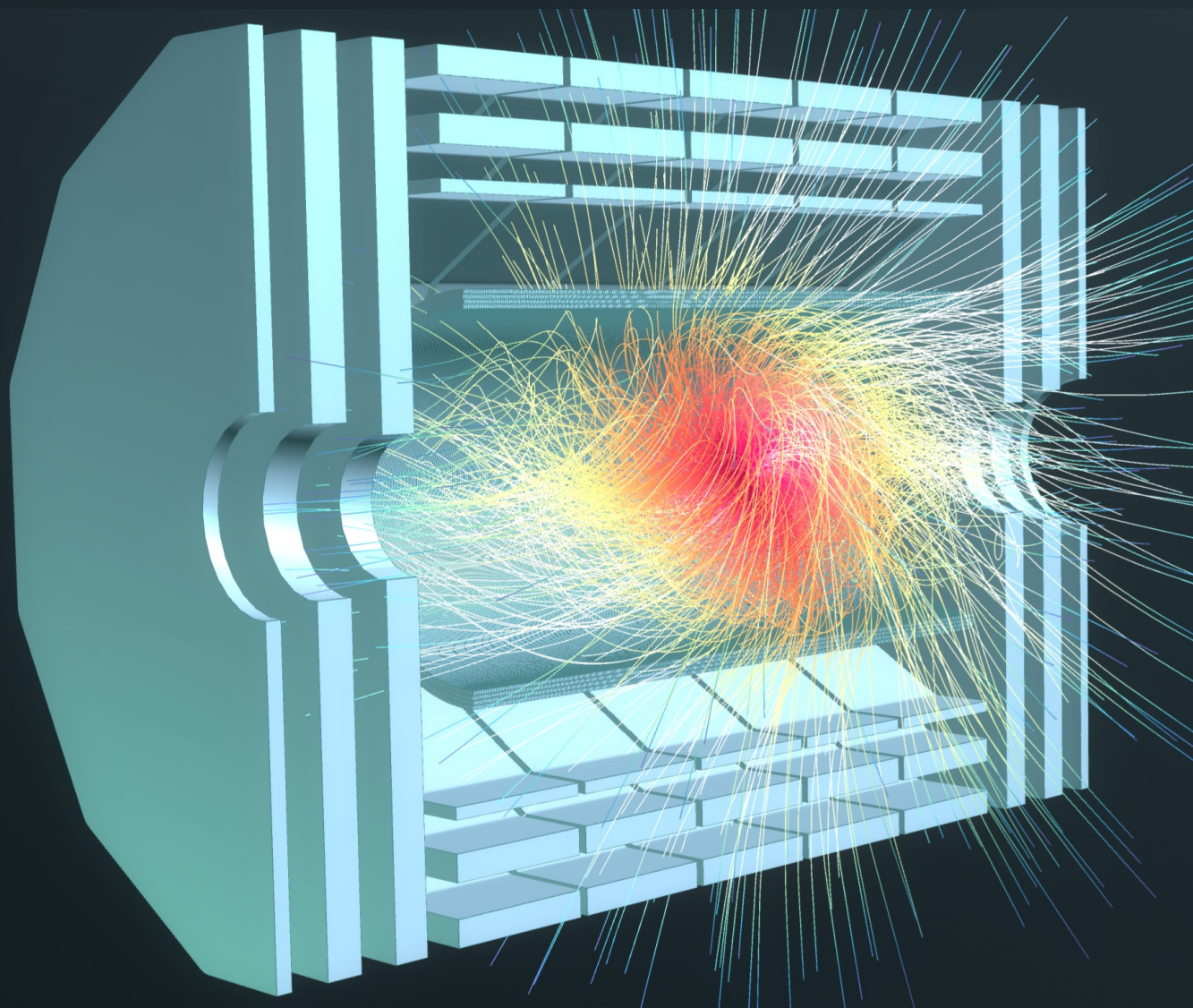


- Mesh calculation allows calculating field on and in conductor
- Because we included the **material** properties we also get **loadline fraction** and more





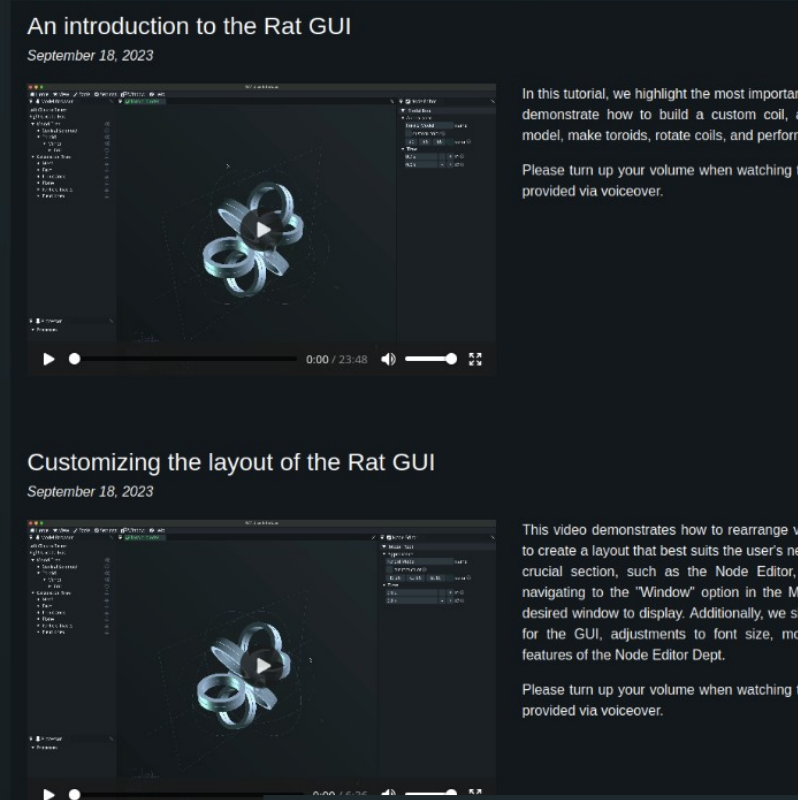
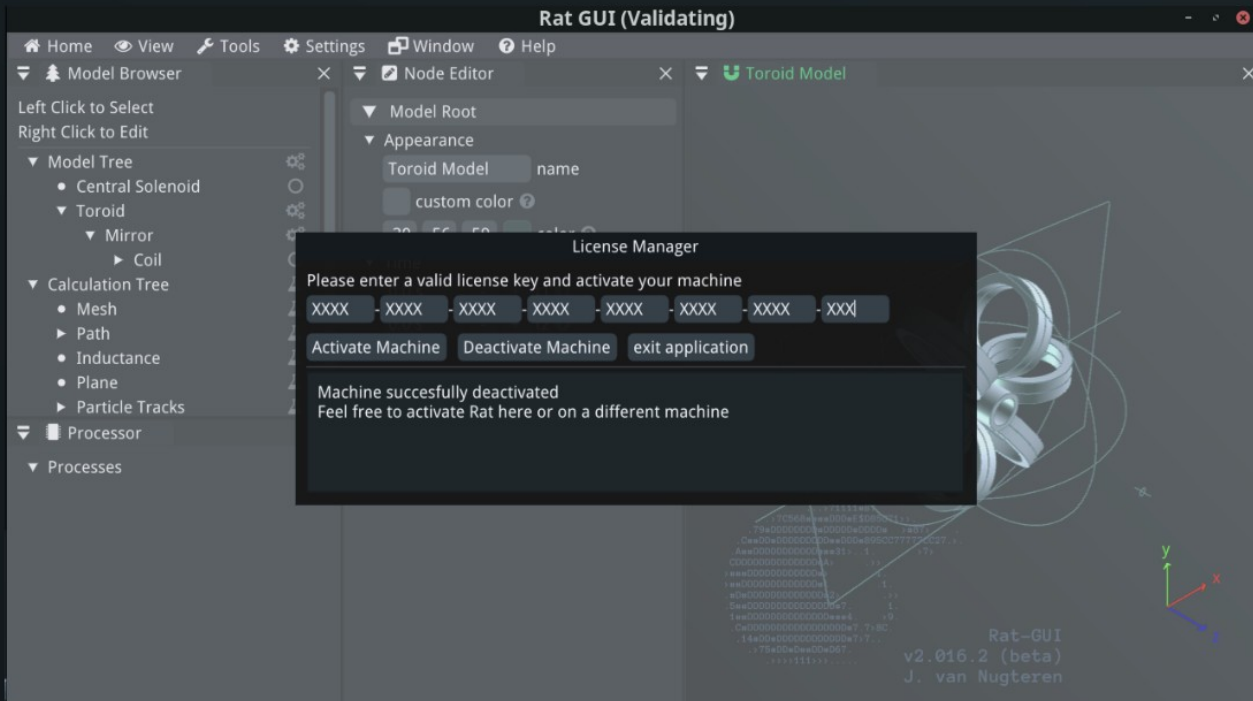
# Particle Tracking



- Uses **pre-calculated FMM** to calculate the field at the target positions
- Uses B-field to calculate particle trajectory based on
  - rest mass
  - electric charge
  - particle energy / momentum
- Changing B-field causes E-field, this is included in calculations
- Particle interactions are currently not taken into account

# Try GUI Today

1. After the presentation ask Nikkie for a trial key or send an e-mail to: [info@rat-gui.com](mailto:info@rat-gui.com)
2. Download RAT from: <https://rat-gui.com/download.html>
3. Install latest RAT on your computer (circumvent microsoft unidentified developer nonsense).
4. Type the key in the XXXX boxes indicated in the screenshot and hit activate.
5. You can use RAT for free for **30 days**.
6. The key can be active on **one PC** at a time.



<https://rat-gui.com/tutorials.html>

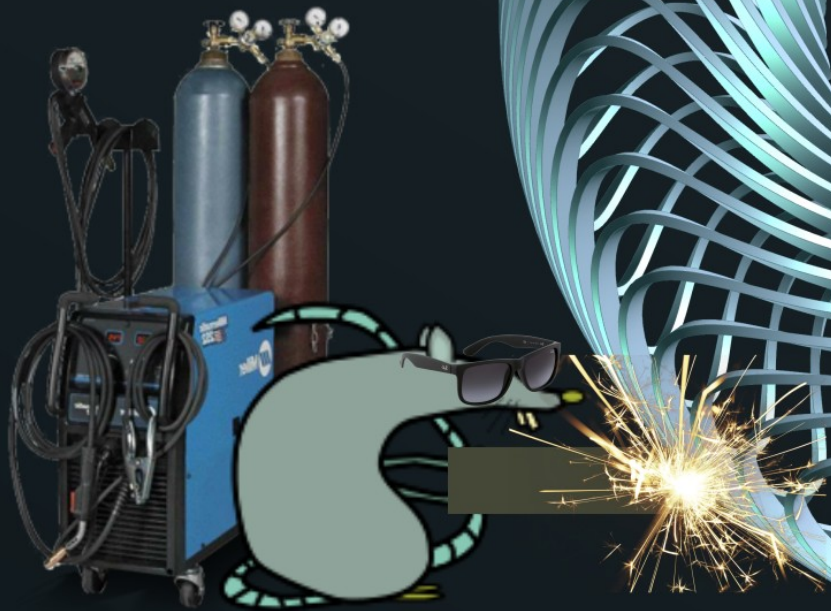
For more information and source  
code:

<https://gitlab.com/Project-Rat>

<https://rat-gui.com/>

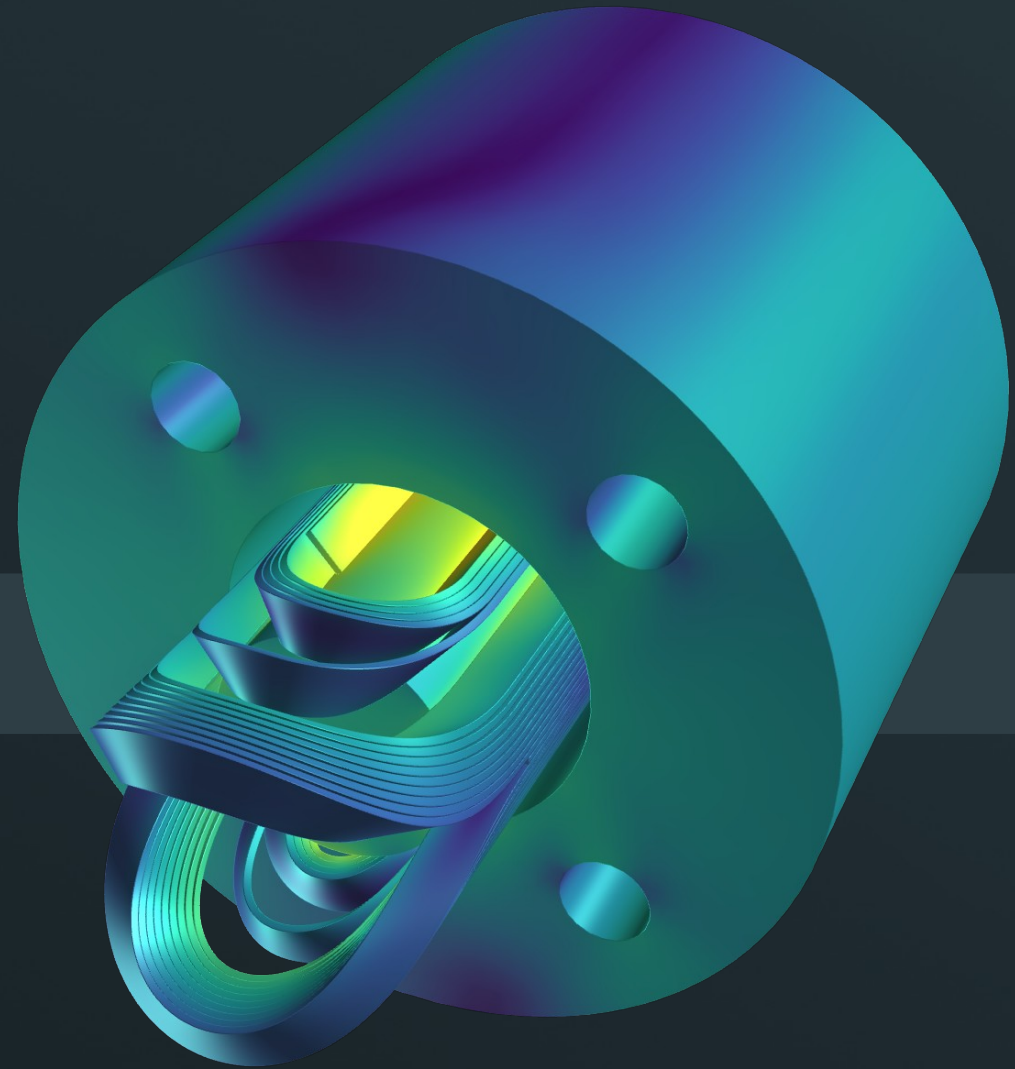


Thanks





# Back-up slides



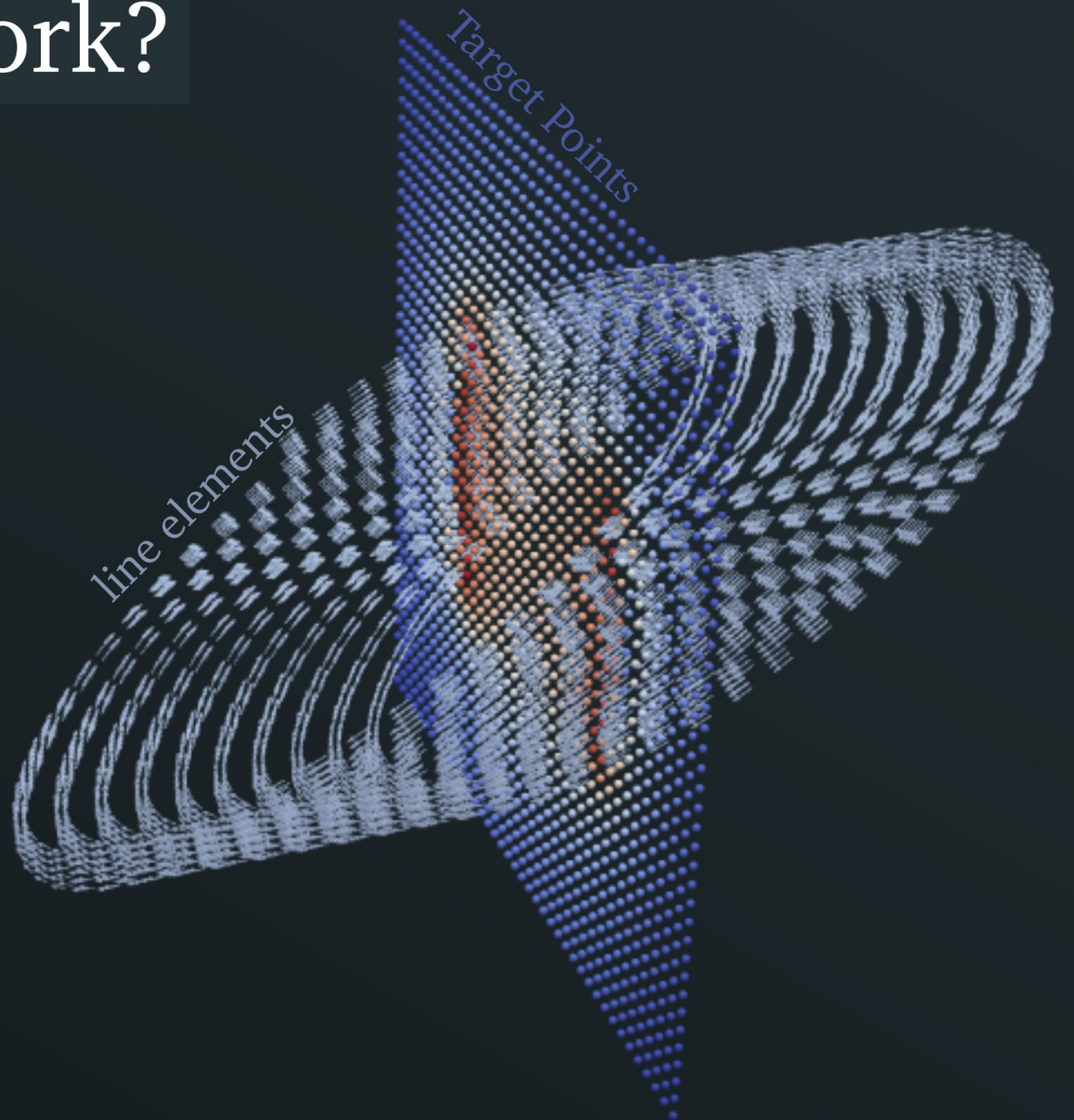
# How does $\mathcal{R}\Delta$ Work?

For a magnetic field calculation the coil is split into  $N$  line elements

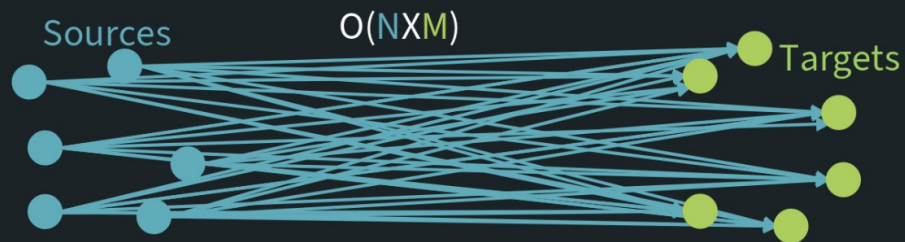
The  $M$  target points are defined by the type of calculation, for instance a plane

Integrating directly from the Biot-Savart equation results in  $O(NM)$  complexity

This will take a long time when  $N$  and  $M$  are large



## Direct Biot Savart Method



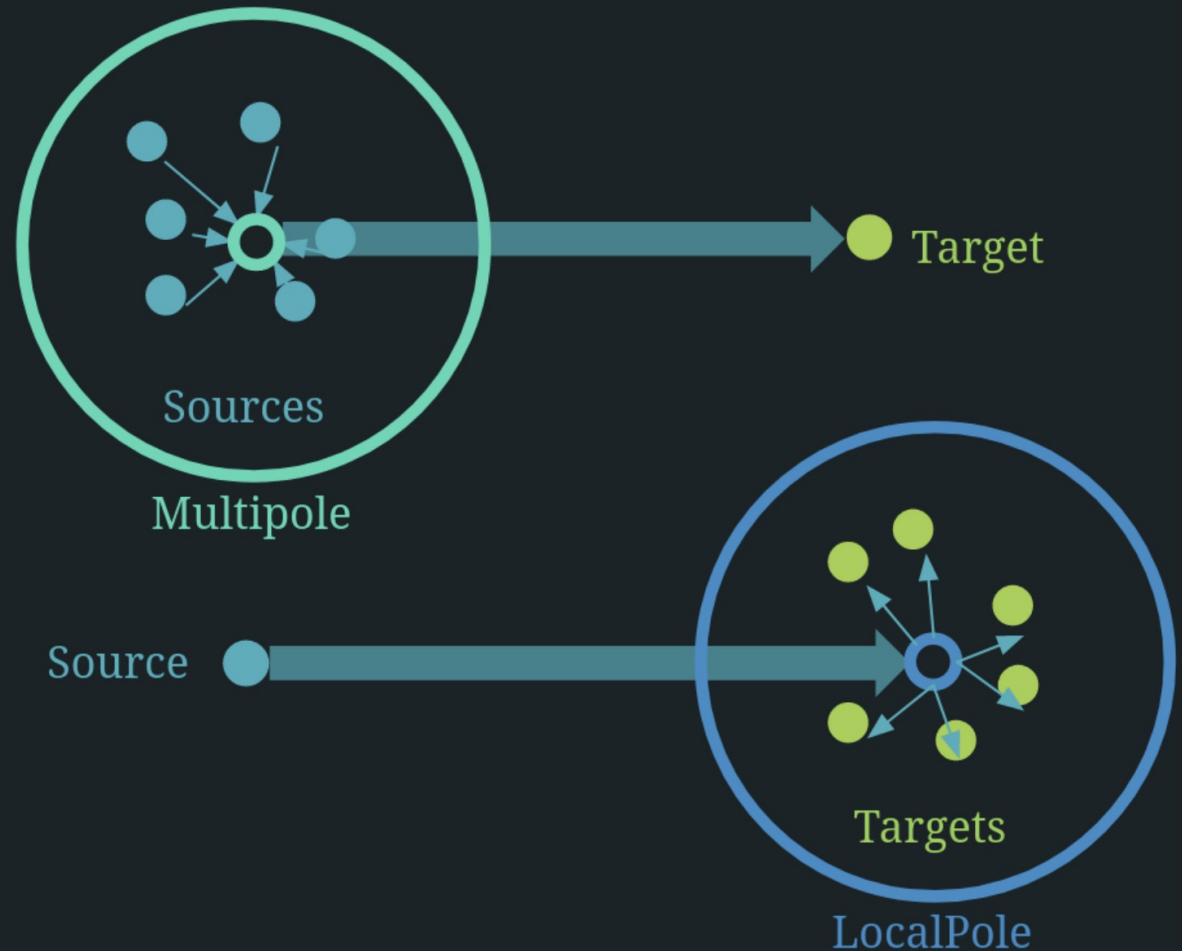
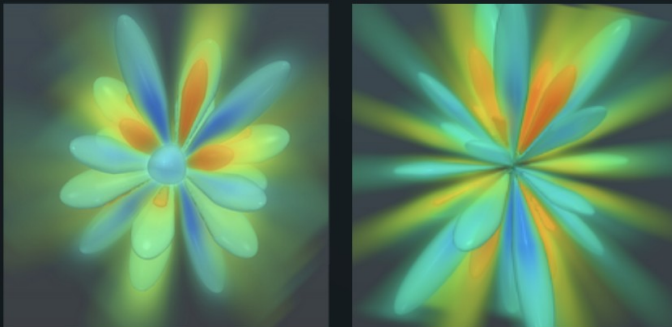
# Fast Multipole Method (I)

The multipole method uses **multipoles** and **localpoles** to represent the field of the sources.

**Multipoles** represent the field of sources inside a sphere at any target point outside the sphere.

**Localpoles** represent the field of sources outside a sphere at any target point inside the sphere.

In this sense they are essentially opposites.



[1] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. J. Comput. Phys. 73, 325–348 (1987).

[2] J. Kurzak and B. M. Pettitt. Fast multipole methods for particle dynamics. Mol Simul. 2006 ; 32(10-11): 775–790. doi:10.1080/08927020600991161.

# Fast Multipole Method (II)

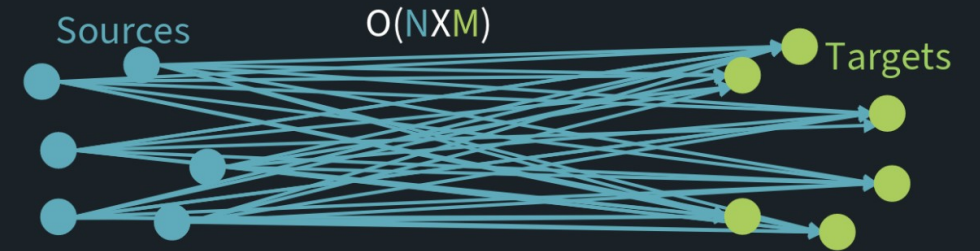
Consider a system with  $N$  sources and  $M$  targets  
Each line represents a field evaluation.

Straight forward Biot-Savart integration leads to complexity  $O(NM)$ .

By using the **Multipoles** and **Localpoles** as a “middle-man” the complexity is reduced to  $O(N+M)$ .

This is the essence of the MLFMM.

Direct Biot Savart Method



Multi-Level Fast Multipole Method (MLFMM)

