

# Rat Graphical User Interface - Manual - *DRAFT*

Little Beast Engineering

20 August 2023

# Contents

<b>Contents</b>	<b>2</b>
<b>1 Introduction</b>	<b>7</b>
1.1 Operating System and Hardware Requirements . . . . .	7
1.2 Customer Support . . . . .	8
1.3 Bug Reports . . . . .	8
1.4 Feature Requests . . . . .	8
<b>2 Getting Started</b>	<b>9</b>
2.1 Downloading the Software . . . . .	9
2.2 Windows Installation . . . . .	9
2.3 Using the Rat GUI for the First Time . . . . .	13
2.3.1 The Floating License Key Model . . . . .	13
2.4 Start using the Rat GUI with an Example Model . . . . .	14
2.5 Calculating Electromagnetostatic Properties . . . . .	15
<b>3 The Rat GUI</b>	<b>16</b>
3.1 Introducing Rat . . . . .	16
3.1.1 Foundational Objects: Cross-Sections and Paths . . . . .	16
3.1.2 Wrapper Models: Making Design Easier . . . . .	16
3.1.3 Object Grouping and Transformations . . . . .	17
3.1.4 Materials and Calculations . . . . .	17
3.1.5 Graphical Overview . . . . .	17
3.2 The Framework of Rat GUI . . . . .	19
3.3 The Viewport . . . . .	19
3.4 The Model Browser . . . . .	20
3.4.1 Editing Options for the Model Tree . . . . .	21
3.4.2 Options for Editing the Calculation Tree . . . . .	24
3.5 The Node Editor . . . . .	25
3.5.1 Drives . . . . .	25
Constant Drive . . . . .	28
Sine Drive . . . . .	28
Linear Drive . . . . .	28
Trapezoid Drive . . . . .	28
Interpolation Table Drive . . . . .	29
Barycentric Rational Interpolation Table Drive . . . . .	29
Fourier Drive . . . . .	29
S-Curve . . . . .	30

	Frenet-Serret RCCT . . . . .	31
3.6	The Processor . . . . .	31
3.7	The Navigation Bar . . . . .	32
3.7.1	Home . . . . .	33
	Export a FreeCAD Macro . . . . .	34
	Exporting an Opera Conductor File . . . . .	34
	Export an STL file . . . . .	34
	Export a Gmsh File . . . . .	34
	Export a Abaqus File . . . . .	34
	Export an Edge Matrix . . . . .	35
	Export a DXF File . . . . .	35
	Import a Solenoid Table . . . . .	36
	Import an Edge Matrix . . . . .	38
3.7.2	View . . . . .	38
3.7.3	Tools . . . . .	40
3.7.4	Settings . . . . .	42
3.7.5	Window . . . . .	43
	The Renderer . . . . .	44
	The Console and Performance Window . . . . .	45
3.7.6	Help . . . . .	45
<b>4</b>	<b>Coils</b> . . . . .	<b>49</b>
4.1	Introduction . . . . .	49
4.1.1	General Coil Settings . . . . .	49
4.1.2	Wrappers versus Customizability . . . . .	51
4.2	The Custom Coil . . . . .	52
4.3	The Edges Coil . . . . .	54
4.4	The Solenoid Coil . . . . .	57
4.5	The Racetrack Model Coil . . . . .	58
4.6	The Rectangle Model Coil . . . . .	59
4.7	The Trapezoid Model Coil . . . . .	61
4.8	The D-Shape Model Coil . . . . .	62
4.9	The Flared-End Racetrack Model Coil . . . . .	63
4.10	The Cloverleaf Model Coil . . . . .	64
4.11	The Canted Cosine Theta Model Coil . . . . .	66
4.12	The Plasma Ring Coil . . . . .	71
<b>5</b>	<b>Meshes</b> . . . . .	<b>74</b>
5.1	Introduction . . . . .	74
5.2	Custom . . . . .	75
5.3	Cube . . . . .	75
5.4	Sphere . . . . .	75
5.5	Cylinder . . . . .	75
5.6	Doughnut . . . . .	75
5.7	Point . . . . .	75
5.8	Line . . . . .	75
5.9	Axes . . . . .	75
5.10	Stick Figure . . . . .	75
5.11	Import . . . . .	75

<b>6</b>	<b>HB-Meshes</b>	<b>76</b>
6.1	General settings . . . . .	77
6.2	Custom HB-Mesh . . . . .	78
6.3	Sphere HB-Mesh . . . . .	79
6.4	JSON HB-Mesh . . . . .	80
6.5	The HB-Curve . . . . .	82
6.5.1	The Vinh Le-Van Fit . . . . .	83
6.5.2	The HB-Curve Table . . . . .	83
6.5.3	The HB-Curve File . . . . .	84
<b>7</b>	<b>Permanent Magnets</b>	<b>87</b>
7.1	Introduction . . . . .	87
7.2	Custom Permanent Magnet . . . . .	87
7.3	Bar Permanent . . . . .	87
7.4	Ring Permanent Magnet . . . . .	87
7.5	Sphere Permanent Magnet . . . . .	87
<b>8</b>	<b>Distmesh Cross-Section</b>	<b>88</b>
8.1	Introduction . . . . .	88
8.2	Building a Cross-Section with the Distmesher . . . . .	89
8.3	Shapes . . . . .	91
8.3.1	Circle Distance Function . . . . .	91
8.3.2	Ellipse Distance Function . . . . .	93
8.3.3	Rectangle Distance Function . . . . .	93
8.3.4	Rounded Rectangle Distance Function . . . . .	95
8.3.5	Polygon Distance Function . . . . .	96
8.3.6	Plasma Distance Function . . . . .	98
8.4	Operations on Distance functions . . . . .	99
8.4.1	The Union Operation . . . . .	100
8.4.2	The Difference Operation . . . . .	100
8.4.3	The Intersect Operation . . . . .	103
8.5	Shape Groups . . . . .	104
8.5.1	Array Distance Function . . . . .	104
8.5.2	Angular Array Distance Function . . . . .	105
8.6	Scale Functions . . . . .	108
<b>9</b>	<b>Cross-Sections</b>	<b>111</b>
9.1	Introduction . . . . .	111
9.2	Point Cross-Section . . . . .	113
9.3	Line Cross-Section . . . . .	114
9.4	Path Cross-Section . . . . .	117
9.5	Rectangle Cross-Section . . . . .	118
9.6	Circle . . . . .	119
9.7	Distmesh Cross-Section . . . . .	120
<b>10</b>	<b>Paths</b>	<b>121</b>
10.1	Circle . . . . .	121
10.2	Polygon . . . . .	121
10.3	Rectangle . . . . .	121
10.4	Trapezoid . . . . .	121

10.5	D-shape . . . . .	121
10.6	Plasma . . . . .	121
10.7	Flared . . . . .	121
10.8	Clover . . . . .	121
10.9	Spiral . . . . .	122
10.10	Canted Cosine Theta Paths . . . . .	122
	10.10.1 Regular CCT . . . . .	122
	10.10.2 Custom CCT . . . . .	122
	10.10.3 Table CCT . . . . .	122
10.11	Axis . . . . .	122
10.12	XYZ File . . . . .	122
10.13	XYZ Table . . . . .	122
<b>11</b>	<b>Path Group</b>	<b>123</b>
11.1	Group . . . . .	123
	11.1.1 Point . . . . .	124
	11.1.2 Straight . . . . .	124
	11.1.3 Arc . . . . .	124
	11.1.4 Super Ellipse . . . . .	124
11.2	Attachment . . . . .	124
11.3	Connect V1 . . . . .	124
11.4	Connect V2 . . . . .	124
11.5	Map . . . . .	124
11.6	Frenet-Serret . . . . .	124
11.7	Unfold . . . . .	124
11.8	Offset . . . . .	124
11.9	Local Offset . . . . .	124
11.10	Cable . . . . .	124
11.11	Stair Cable . . . . .	124
<b>12</b>	<b>Groups</b>	<b>125</b>
12.1	Group . . . . .	125
12.2	Array . . . . .	125
12.3	Angular Array . . . . .	125
12.4	Mirror . . . . .	125
12.5	Path Array . . . . .	125
12.6	Clip . . . . .	125
<b>13</b>	<b>Transformations</b>	<b>126</b>
13.1	Translation . . . . .	126
13.2	Rotation . . . . .	126
13.3	Reflection . . . . .	126
13.4	Bending . . . . .	126
13.5	Reverse . . . . .	126
13.6	Flip . . . . .	126
<b>14</b>	<b>Material properties</b>	<b>127</b>

<b>15 Calculations</b>	<b>128</b>
15.1 Introduction	128
15.1.1 General Calculation Settings	129
15.1.2 GPU Acceleration	129
15.1.3 The Multi-Level Fast Multipole Method	130
15.1.4 Editing Calculations from the Model Browser	132
Add Circuit	132
Add Background	132
15.1.5 Post-Processing	132
15.2 Point(s)	135
15.3 Line	136
15.4 Path	139
15.5 Plane	141
15.6 Mesh Calculation	143
15.6.1 Mesh Calculation Post-Processor Settings	144
Enable Current Sharing	144
Magnetic and electric fields	145
Currents and Power	145
Temperature	147
Mechanics	147
Geometric	148
15.6.2 Field Component	150
15.7 Cartesian Grid Calculation	151
15.8 Polar Grid Calculation	153
15.9 Inductance	153
15.10 Cylindrical Harmonics	155
15.11 Spherical Harmonics	155
15.12 Tracking	155
15.12.1 Add Tracking Mesh	155
15.12.2 Add Emitter	155
15.13 Length	155

# Chapter 1

## Introduction

This manual serves as a guide for the Rat Graphical User Interface (Rat GUI). Before using the Rat GUI, please familiarize yourself with the General Terms and Conditions (Terms, [1]) and the End-User License Agreement (EULA, [2]). By using the Rat GUI, you accept both the EULA and the Terms.

This manual is a work in progress and will be regularly updated. For the latest version of this manual, please visit our website. If you have any suggestions regarding this manual, please let us know via email.

### 1.1 Operating System and Hardware Requirements

Currently, the Rat GUI is supported only on Microsoft Windows [3] starting from Windows 10. The Rat GUI has been tested on different platforms with various hardware architectures. The minimum hardware requirements are summarized in Table 1.1.1. The minimum and recommended hardware specifications assume an architecture without an NVIDIA GPU. The third column provides recommendations if you intend to use GPU acceleration. Note that GPU acceleration is optional and not required to run the Rat GUI, but it is worth mentioning that utilizing a GPU can significantly reduce computation time, depending on the hardware used (typically by a factor of 10 or more).

Table 1.1.1: Hardware Requirements and Recommendations

Hardware	Required	Recommended	Recommended for GPU acceleration
CPU	x86-64 <sup>1</sup>	x86-64 <sup>1</sup>	x86-64 <sup>1</sup>
Number of CPU cores	Four	Six	Six
RAM <sup>2</sup>	8GB	16GB	16GB
GPU	Any	Any	NVIDIA, compute capability > 6.1
GPU memory	2GB	4GB	8GB

<sup>1</sup> The Rat GUI uses OpenBLAS [4] for linear algebra computations, which is suitable for both Intel and AMD processors.

<sup>2</sup> The (GPU) memory requirement depends on the model size. The recommended (GPU) memory should suffice for most applications; however, larger models with smaller element sizes may require additional (GPU) memory.

## 1.2 Customer Support

Our customer support team can be reached via email at [info@rat-gui.ch](mailto:info@rat-gui.ch). We strive to respond to your email as soon as possible, typically within three business days. Our customer support includes assistance with software installation, getting started, and resolving software bugs, all free of charge. Our customer support operating hours are from 11:00 to 19:00 (CET) on business days, excluding public holidays and weekends. To help you get started with the Rat GUI, we provide this manual, tutorials, and our Frequently Asked Questions (FAQ). If you still have questions after consulting these resources, please don't hesitate to email us.

## 1.3 Bug Reports

Although we extensively test the Rat GUI before releasing new versions, we cannot guarantee that it will be free of bugs. If you encounter any bugs, please report them to us via email at [info@rat-gui.ch](mailto:info@rat-gui.ch). We will make every effort to respond to your report within three business days.

## 1.4 Feature Requests

Generally, we do not implement feature requests in the Rat GUI, but we may consider them for future upgrades. Any software customization, configuration, training, or other services beyond the scope of our standard customer support can be requested through a separate statement of work. These services are subject to payment on a time and materials basis, in accordance with Little Beast Engineering's standard rate card.



# Chapter 2

## Getting Started

### 2.1 Downloading the Software

The Rat GUI installer can be downloaded via a link sent to you personally by Little Beast Engineering from the email address [info@rat-gui.ch](mailto:info@rat-gui.ch) or [info@littlebeastengineering.com](mailto:info@littlebeastengineering.com). *You should only use the Rat GUI if you have obtained it directly from Little Beast Engineering!* Make sure to download the latest version that was most recently published. By clicking on one of the download links in the table, the download will automatically start.

If you are interested in trying the Rat GUI, you are invited to contact Little Beast Engineering via email at [info@rat-gui.ch](mailto:info@rat-gui.ch) and request a thirty-day trial license. For more information, please visit our website.

### 2.2 Windows Installation

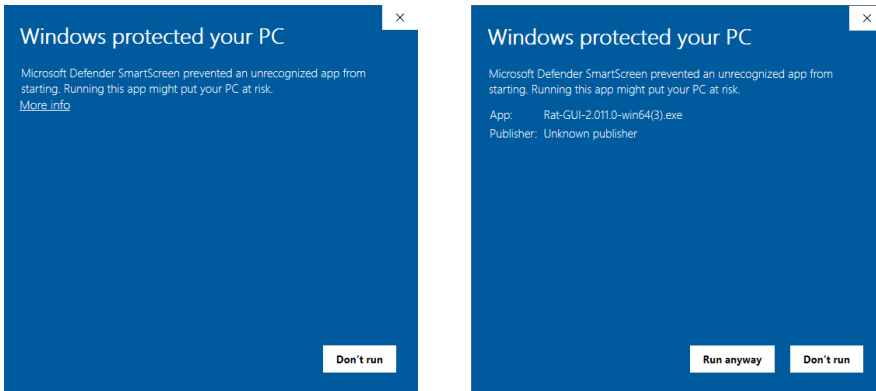
To start the installation, (double) click on the installer icon in the folder where you downloaded the file. If you have previously installed the Rat GUI on your computer, make sure to select the installer with the highest version number. The installer icon is shown in Figure 2.2.1.



Figure 2.2.1: The Rat GUI installer can be started by clicking on the installer icon.

When installing or reinstalling the Rat GUI, Windows Defender might display a pop-up as shown in Figure 2.2.2a. To continue, click on ‘More info’ to expand the message. You should see the name of the Rat GUI executable and that the publisher is listed as unknown, as shown in Figure 2.2.2b. Click on ‘Run anyway’ to proceed with the installation.

In addition, Windows User Account Control might ask if you want to allow the Rat GUI installer executable to make changes to your device. The installer copies the files needed to



(a) Initial window.

(b) More information.

Figure 2.2.2: The Windows Defender message when using the Rat GUI Installer for the first time. Please click ‘More info’ to show information about the software, as shown in the left figure. The installation can be started by clicking on ‘Run anyway’ at the bottom right, as shown in the right figure.

run the Rat GUI to the following directory:

`C:\Program Files\Rat-GUI\` (2.2.1)

Click ‘Yes’ in the window shown in Figure 2.2.3 to continue.



Figure 2.2.3: Windows User Account Control will ask to allow the Rat GUI installer to make changes to your computer. The installer copies the files needed to run the Rat GUI to this directory. Click ‘Yes’ to continue.

If you are updating the Rat GUI to the latest version and you still have the older version of the Rat GUI on your computer, the installer will ask if you want to uninstall the old version, as shown in Figure 2.2.4. Select ‘Yes’ to uninstall the old version first. Choosing ‘No’ will retain the older version, which can be useful if you want to keep multiple versions of the Rat GUI on your system. Selecting ‘Cancel’ will stop the installation process completely. If you choose ‘No’, be careful to specify a different installation location for each version; otherwise, the existing Rat GUI will be overwritten during the installation process. Refer to Figure 2.2.5d for the installation location. Before installing the next version, please ensure that you close the Rat GUI application.

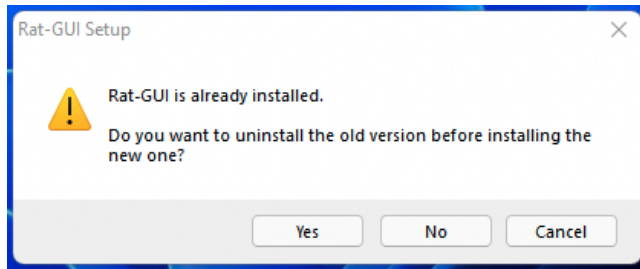


Figure 2.2.4: Message prompt when you already have the Rat GUI installed on your computer. By selecting ‘Yes’, the installer will first delete the old version. If you choose ‘No’, the current version of the Rat GUI will be retained. However, if you do not specify a different installation directory later in the installation process, your current version will be overwritten. See Figure 2.2.5d for the installation location.

Next, the Rat GUI setup window will open, guiding you through the installation process, as shown in Figure 2.2.5a. Each installation step is illustrated in Figure 2.2.5 for reference. To navigate between steps, you can select either ‘Next >’ or ‘< Back’.

On the second page of the Rat GUI setup, you will be asked to read and accept the End-User License Agreement (EULA), as depicted in Figure 2.2.5b. Clicking on ‘I agree’ is necessary to proceed with the Rat GUI installation. If you haven’t received a copy of the EULA with the download link, you can download it from this link. Please note that the Rat GUI is also subject to our General Terms and Conditions (Terms).

Next, the Rat GUI setup will ask if you want to add Rat’s directory to the system PATH, as shown in Figure 2.2.5c. This can be useful if you want to start the Rat GUI from the terminal later. By default, the Rat GUI is not added to your system PATH. During this step, you can also indicate whether you would like to have a Rat GUI Desktop icon.

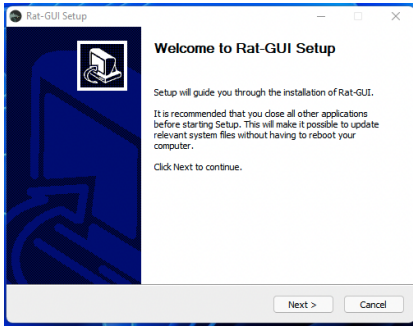
On the following setup page, you can select the installation location, as depicted in Figure 2.2.5d. By default, it is set to:

`C:\Program Files\Rat-GUI\.` (2.2.2)

If you want to keep an older version of the Rat GUI on your system alongside the one you are about to install, you can specify a different location. For example:

`C:\Program Files\Rat-GUI-versionx.x.x\.` (2.2.3)

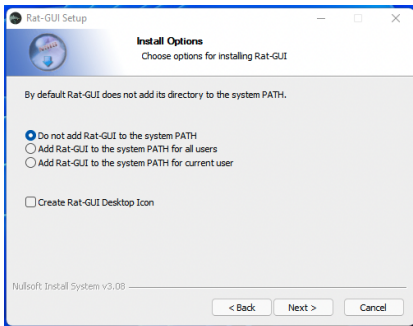
You will then be asked to choose the Start Menu folder where you want to create a shortcut to the Rat GUI. If you don’t want to create a shortcut, you can indicate that below the file list, as shown in Figure 2.2.5e. Once you have made your selection, the installation will begin, as seen in Figure 2.2.5f. This process may take a few minutes. Once the installation is complete, you will receive a notification in the setup window, as shown in Figure 2.2.5g. You can then close the window by clicking on ‘Finish’.



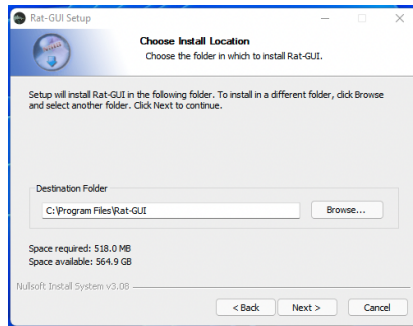
(a) Setup window 1.



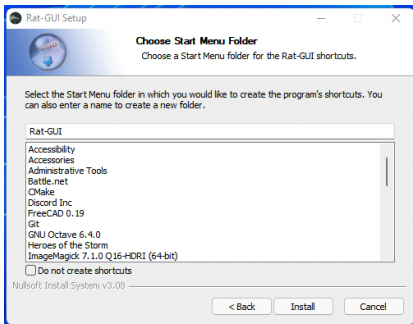
(b) Setup window 2.



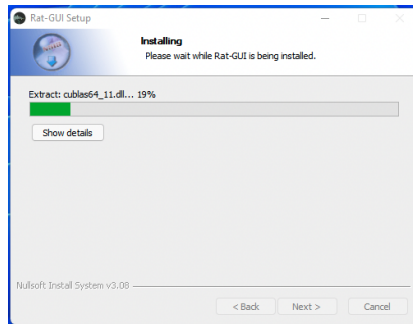
(c) Setup window 3.



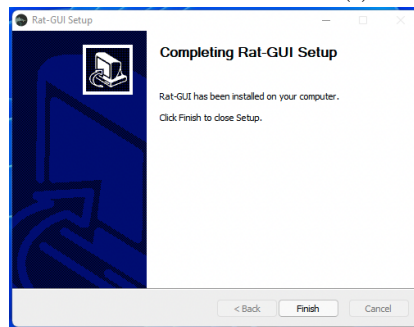
(d) Setup window 4.



(e) Setup window 5.



(f) Setup window 6.



(g) Installation complete.

Figure 2.2.5: The different pages of the Rat GUI installation process.

## 2.3 Using the Rat GUI for the First Time

You can now start the Rat GUI just like any other program on your system. During the startup process, you will first see the splash screen (Figure 2.3.1) and then the Rat GUI interface.



Figure 2.3.1: The splash screen of the Rat GUI.

If you are using the Rat GUI for the first time, you need to enter a valid license key, as shown in Figure 2.3.2. You can use the ‘Tab’ key on your keyboard to navigate between input fields while typing in the key. Once entered, select ‘activate machine’ to start using the Rat GUI. Your license key will be stored on your computer at:

```
C:\Users\Your User Name\AppData\Roaming\Rat-GUI\serial.dat (2.3.1)
```

The license key can be used on multiple platforms and by multiple colleagues. Instead of sharing the key in text form, you can share it by copying the ‘serial.dat’ file to the same location on another computer or to the directory of another user on the same computer.

### 2.3.1 The Floating License Key Model

The floating license key model is designed to allow the license key to be used on multiple platforms and by multiple colleagues. In this model, a check is performed on a remote server to determine if the license key is available for use.

When you activate the Rat GUI with your license key, the software establishes a connection with the license server to verify if the license key is currently being used by another instance of the Rat GUI. If the license key is available, you will be granted access to use the software.

It’s important to note that while the floating license key can be used on multiple platforms and by multiple colleagues, it only allows for one instance of the Rat GUI to be in use at

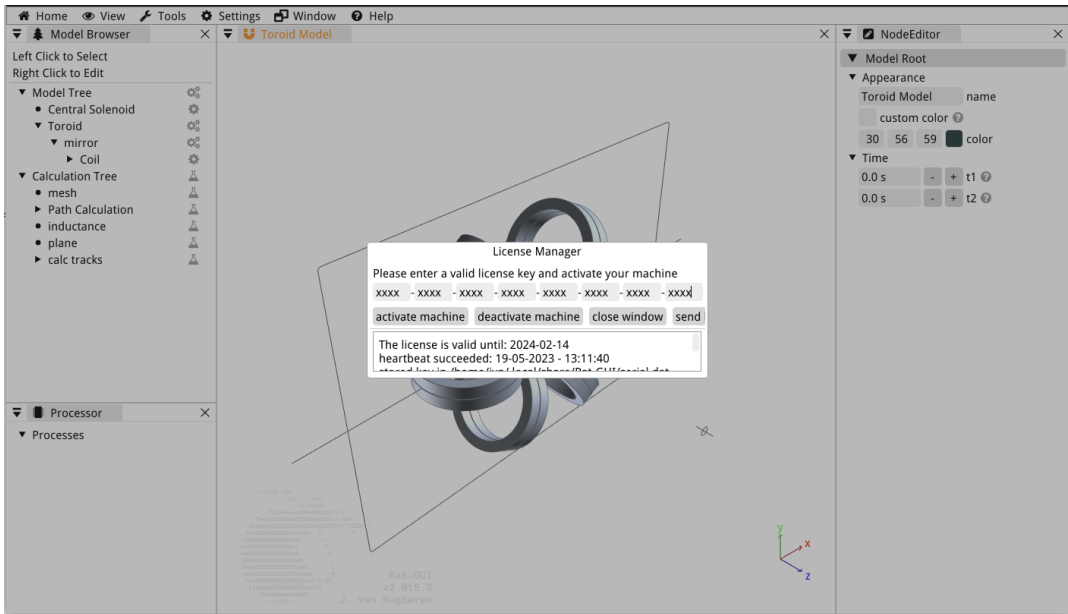


Figure 2.3.2: Entering the license key for the first time.

the same time. This means that if the license key is being used by one person or on one platform, it will not be available for use on another platform or by another colleague until the current instance is released or closed.

By using the floating license key model, you can maximize the utilization of your license across different platforms and allow multiple colleagues to access and use the Rat GUI as needed. Should you require multiple colleagues or platforms to have a running Rat GUI open at the same time, you can expand the licensing capacity by purchasing an additional license through Little Beast Engineering. With the additional license, the existing serial key will remain the same. However, the license server will be configured to allow for two instances of the Rat GUI to be open simultaneously using the same key. Please note that purchasing additional licenses through Little Beast Engineering is a convenient way to extend the capabilities of the Rat GUI without the need to manage multiple serial keys. The existing key remains valid and is used to activate the additional instances allowed by the license server.

## 2.4 Start using the Rat GUI with an Example Model

For a quick start, you can use one of the examples that are pre-programmed into the Rat GUI. They can be accessed by navigating to 'Home > Start From Example,' where you can select the desired example from a list, as shown in Figure 2.4.1. An example model consists of a collection of coils, meshes, ferrites, and calculations that are prepared as a complete Rat Model. This model can be edited using the *node editor* (Section 3.5) by right-clicking on objects in the Model Tree (Section 3.4) and the Calculation Tree (Section 3.4).

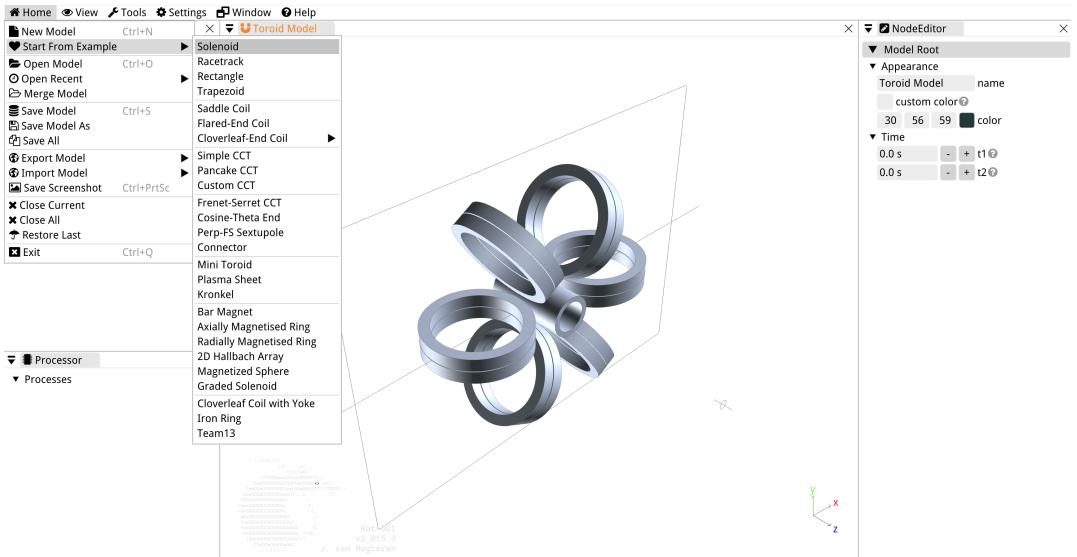


Figure 2.4.1: Quickly start by selecting an example model from the list of built-in examples.

## 2.5 Calculating Electromagnetostatic Properties

Some examples like the Solenoid already contain calculations in the Calculation Tree. If you want to add a calculation, right-click on Calculation Tree in the *model browser*, select ‘Add Calculation’ and click on the calculation of your choice to add it to the Calculation Tree. To run the calculation, select it in the Calculation Tree to make it appear in the *node editor*. There you can change the settings of the calculation and click on ‘Calculate’ to run the calculation and to show the result.

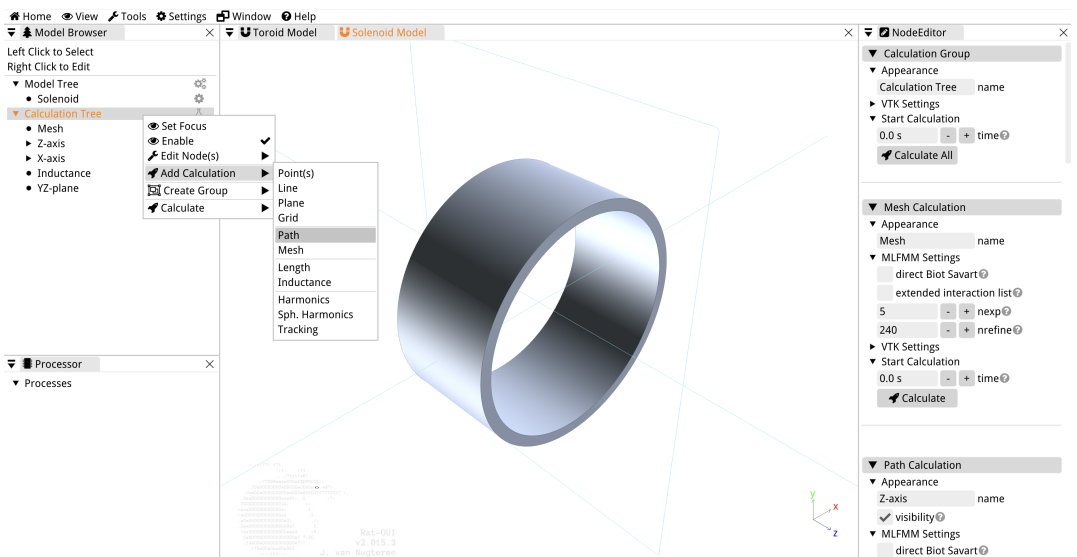


Figure 2.5.1: Add a calculation to the Calculation Tree.

# Chapter 3

## The Rat GUI

In this chapter we will explore the various components that comprise the Graphical User Interface (GUI) of Rat, also known as Rat GUI. The objective is to provide you with a comprehensive understanding of the Rat GUI's structure and functionality, equipping you with the knowledge to navigate and utilize its features effectively.

### 3.1 Introducing Rat

Before explaining the Rat GUI it is important to describe the open source Rat Library that is at the heart of this user interface. Rat, a state-of-the-art software, is tailored specifically for the electromagnetic calculations of magnets. Within its computational domain, magnets are seamlessly represented using a hierarchy of objects, allowing for comprehensive modeling and analysis.

Rat is created using `c++`, an object-oriented programming language. This design choice underpins the frequent references to various tools within the software as “objects.” Typically, in `c++` software, objects serve as input for a pipeline or factory, which then constructs the final model. For instance, when modeling an angular array of six identical D-shaped coils for a toroidal magnet system, there's no need to tediously define each coil. Instead, you can simply define a single D-shaped coil object and feed it into the angular array object, streamlining the entire process. Throughout this introduction to Rat we'll touch upon the diverse objects available, offering a brief overview and pointing you to dedicated chapters for an in-depth exploration.

#### 3.1.1 Foundational Objects: Cross-Sections and Paths

At the core of Rat's modeling paradigm are the Cross-Sections (Chapter 9) and Paths (Chapter 10), the primary building blocks for creating intricate magnetic geometries. Rat offers a plethora of predefined Cross-Sections and Paths, but for those seeking customization, there's the flexibility to import external shapes, curves, or even to define these entities using coordinate-based descriptions. A special class of Cross-Sections is the Distmesh Cross-Section (Chapter 8). A dedicated chapter can be found in Chapter 8.

#### 3.1.2 Wrapper Models: Making Design Easier

For your convenience, Rat introduces predefined Coils, Meshes, HB-Meshes, and Permanent Magnets. These Wrapper Models simplify the design process for standard geometries. What



sets them apart is their user-centric parameters that emphasize ease-of-use, offering a more intuitive way to define the geometry than crafting a unique structure without the need to select a Cross-Section and Path.

However, Rat doesn't restrict those who seek full control. Each high-level object, whether it's a Coil, Mesh, HB-Mesh, or Permanent Magnet, comes with a Custom variant, granting users the freedom to select their preferred Cross-Section and Path. These high-level objects in Rat can be described by the following statements:

- Coil: A volumetric entity capable of conducting current, essentially functioning as an electromagnet (see Chapter 4);
- Mesh: A three-dimensional structure, void of current-carrying capabilities (refer to Chapter 5);
- HB-Mesh: A unique three-dimensional body, crafted from non-linear magnetic materials (more information in Chapter 6);
- Permanent Magnet: An object that stands out with its uniform magnetization (explained in Chapter 7).

### 3.1.3 Object Grouping and Transformations

Rat's versatility shines when dealing with compound geometries. Objects can be effortlessly grouped to create mirrored structures, object arrays, and more (Chapter 12). Beyond grouping, objects can undergo diverse transformations like translations, rotations, and bending (Chapter 13).

Paths, in particular, have an associated special class of Path Groups (Chapter 11). These can serve multiple purposes: from modeling singular cables as opposed to entire coil packs, to devising Custom Paths amalgamated from different Path Sections. Additionally, they offer the flexibility to alter a Path's shape based on intricate mathematical descriptions, such as the manipulation techniques of ribbons (as illustrated by Frenet-Serret formulas) or the smooth integration of current leads.

### 3.1.4 Materials and Calculations

A pivotal feature of Rat is the ability to assign distinct material properties to Coils and Meshes (Chapter 14). These properties are for subsequent calculations, allowing users to obtain parameters like critical current distributions or power densities. These are described in Chapter 15.

### 3.1.5 Graphical Overview

The overarching structure of Rat's functionalities can be pictorially summarized through three distinct graphs, encapsulating the essence of the Rat Model. For a practical understanding and to visualize these core concepts, readers can refer to the examples provided in Figures 3.1.1 to 3.1.4.

This manual offers dedicated chapters on each of these elements, ensuring users have all the information they need to maximize the potential of Rat.

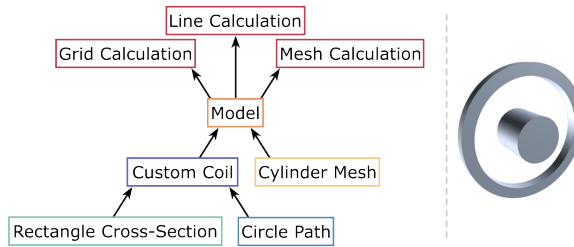


Figure 3.1.1: Modelling a Custom Coil and a Cylinder Mesh with Rat in the Rat GUI.

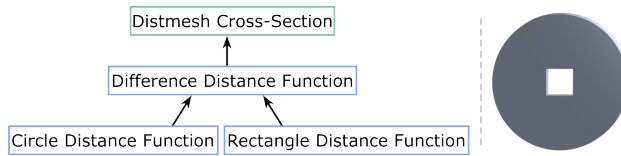


Figure 3.1.2: Modelling a Distmesh Cross-Section with Rat in the Rat GUI.

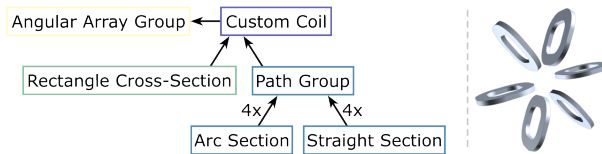


Figure 3.1.3: Modelling a Custom Coil in an Angular Array with Rat in the Rat GUI, obtaining racetrack coils in a toroidal configuration.

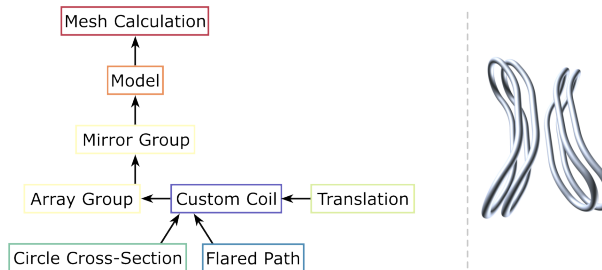


Figure 3.1.4: Modelling a translated Custom Coil inside a an Array Group for one pole, that goes into a Mirror Group to obtain the second pole, with Rat in the Rat GUI.

## 3.2 The Framework of Rat GUI

The Rat GUI consists of five primary components, each serving a distinct purpose within the interface. These components, as depicted in Figure 3.2.1, include the *navigation bar* located at the top, the *model browser* positioned on the top left, the *node editor* situated on the right, the *processor* located on the bottom left, and the *viewport* in the center where your model is rendered.

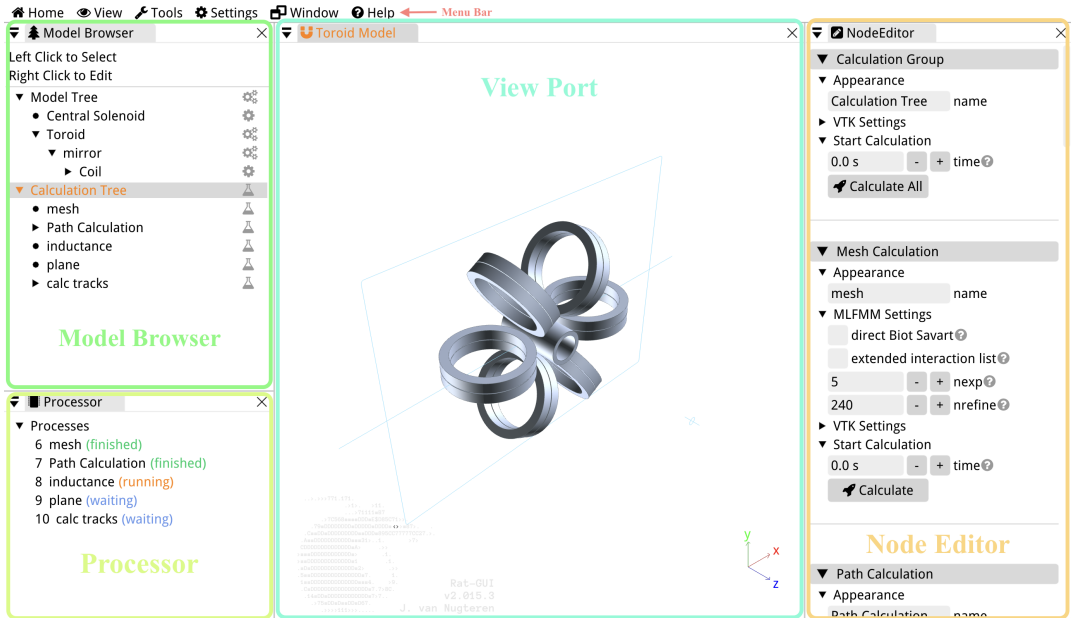


Figure 3.2.1: The different components of the Rat GUI in.

Each component of the Rat GUI can be freely moved to different locations on the screen, enabling you to create a customized layout that suits your preferences. Simply click on the title of a section (e.g., ‘Node Editor’) and drag it to the desired position. If you want to dock a window to a specific position you can use the blue rectangles that appear during the dragging process. This flexible arrangement allows for various configurations, such as side-by-side viewing of models, placing the *node editor* below the *model browser* or vice versa, or having them positioned next to each other.

The same flexibility applies to the display of calculation results, which will appear in separate windows after a calculation has completed. In the event that you accidentally close a window by clicking the ‘x’ next to the title, you can easily reopen it by accessing the ‘Window’ menu in the *navigation bar* and selecting the desired window to display.

For a more comprehensive visual demonstration of these customization options, we invite you to watch a video tutorial available [here](#), which provides a step-by-step walkthrough of the docking process.

## 3.3 The Viewport

The *viewport* in the Rat GUI serves as the central area where your model is displayed. It offers a three-dimensional view of your model and allows for interactive manipulation and

navigation using your mouse.

Using the left mouse button, you can rotate the model by clicking and dragging in any direction. This enables a complete 360-degree view, allowing you to explore the model from various angles. This intuitive rotation feature helps in examining the model from different perspectives.

The right mouse button enables smooth zooming in and out of the model. By holding down the right button and moving the mouse up and down, you can adjust the zoom level and focus on specific areas of interest. This step-less zooming functionality provides precise control over the magnification of the model. You can also zoom in and out on your model with the scroll wheel of the mouse.

With the middle mouse button, or scroll wheel, you can translate the model in the *viewport*. This is called panning of the model. Holding down the middle button and moving the mouse allows you to shift the model horizontally or vertically within the *viewport*, providing flexibility in positioning and navigation. This panning capability allows you to explore different regions of the model without altering the zoom or rotation settings. If you do not have a middle mouse button, you can press and hold **Shift** on your keyboard while holding the left mouse button to use the panning functionality as well.

Additionally, clicking the middle mouse button on the mesh of the model resets the target point of the camera. This action centers the view on the selected point, allowing for convenient focus on specific areas or features. This reset functionality ensures that you can easily navigate to different locations of interest within the model while maintaining control over the camera position. You can reset the camera position with the key bind **Ctrl-R** or by navigating to the view menu (see Section 3.7.2) and clicking ‘Reset View.’ Just like with panning, if you do not have a middle mouse button, use **Shift + click** to reset the target point of the camera in the *viewport*.

When you click the left mouse button on a mesh within the *viewport*, the mesh is selected. This selection is reflected in the *model browser* (refer to Section 3.4), where the selected mesh is highlighted. Simultaneously, the selected mesh appears in the *node editor* (see Section 3.5), allowing you to modify its properties and parameters. If the selected mesh is part of a group (more in Chapter 12), clicking the right mouse button on the mesh will initially select the entire group. By repeatedly clicking on the same coil or mesh, you can navigate down through the Model Tree, revealing lower levels and individual objects within the group. This hierarchical selection process enables you to access and manipulate specific components within complex models, providing granular control over the editing process.

## 3.4 The Model Browser

The *model browser* is set up as a tree graph. A tree graph is a hierarchical structure composed of nodes connected by edges. In a tree graph, the objects or elements are typically referred to as nodes. An edge represents the connection or relationship between nodes. In the Rat GUI a node is an object in your model, for instance a coil, a path, or a transformation. Children of nodes in the Rat GUI are shown below the parent node, and they are indented with respect to the node above. If a parent node has children that are not displayed, you can view them by unfolding the tree with the triangles to the left of the name of the node. Click on the triangles to reveal child-node, or to hide them if you just want to show the parent object in the Model Tree. This works exactly the same in the Calculation Tree.

The topmost node, which does not have any parent nodes, is called the root. In the *model browser* of the Rat GUI there are two roots: the Model Tree and below that the Calculation Tree. Nodes that are connected directly below the root are considered its children. Nodes












that share the same parent are referred to as siblings. Nodes without any children are called leaf nodes or terminal nodes, while nodes with at least one child are referred to as internal nodes.

To build your model in the Rat GUI, you can right-click on either the Model or Calculation Trees. This action opens a menu with options to edit the model. For example, to add coils, right-click on the Model Tree, select ‘Add Coil,’ and then choose from a list of pre-defined coils such as Solenoid or Trapezoid, or add a Custom Coil. Detailed explanations of all coil options are provided in Chapter 4.

To perform calculations in the Rat GUI, you can add a calculation to the Calculation Tree by right-clicking and selecting ‘Add Calculation’ from the options. Calculations are an integral part of a Rat Model, just like coils and meshes for instance. When you save your model, all calculations and their settings are stored in the same file. This feature allows you to create a custom list of standard calculations that you want to perform each time your model changes, without the need to re-add them whenever you restart the Rat GUI. For a comprehensive explanation of each available calculation, refer to Chapter 15. In the next two sections you will find more on how to edit your models in the Rat GUI. Some objects do not have all available edit options in their editing menu when you right-click them to start editing. For instance, you can’t add a Coil to a Solenoid, because this node is a Coil itself.

When adding cross-sections and paths to your models from the *model browser* it might happen that objects have the same name initially. Because of Rectangle Cross-Section and a Rectangle Path are both called `rectangle` by default. If you want to know which one is which you can look at the symbols on the outermost right of the *model browser*.

Table 3.4.1: The symbols in the *model browser*.

Object	Symbol
Model Group	
Coil	
Mesh	
HB-Mesh	
Ferrite	
Cross-Section	
Path	
Material	
Group	
Transformation	
Calculation	

### 3.4.1 Editing Options for the Model Tree

Figure 3.4.1 shows the list of options that are revealed when right-clicking the Model Tree or an node in the Model Tree. The ‘Set Focus’ option will make sure that only the object in focus is displayed in the *viewport*, in its local coordinate system. This can make it easier to edit individual objects in complex models, and since it is shown in its local coordinate system this option can help give insight into how transformations affect the object or how the path or cross-sections of the object are defined with respect to the global coordinate system. With the **Escape** key on the keyboard, the focus can be disabled again, or right-click the object in the Model Tree and choose ‘Unset Focus’.

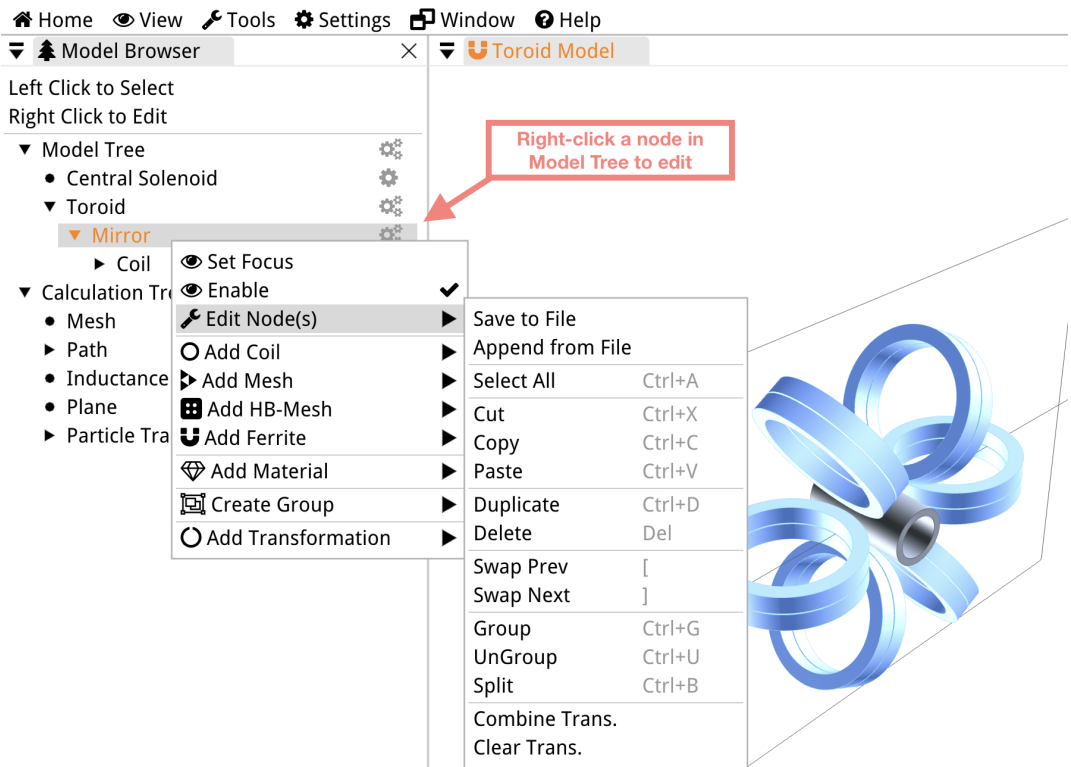


Figure 3.4.1: Right-click on an object in the Model Tree to view the options for editing Rat Models.

By default, all objects in your model are enabled, and when calculations are executed in the Rat GUI, all enabled objects are as input. This is indicated by the check-mark drawn to the right of ‘Enable’ in the list. When an object is enabled, clicking enable will disable it, and the check-mark disappears. To re-enable the object, click enable once more, and the object is available for editing and calculations again.

To edit nodes in the Model Tree quickly, you can use the tools in the ‘Edit Node(s)’ submenu. Except for the first two items, most of these tools are self-explanatory, and can also be found in the Tools in the *navigation bar*, which is explained in Section 3.7.3. The first two options can be used to either save an individual node to a separate file (Save to File) or add any individual node from a file to the current model (Append from File).

There is a number of objects that can be added to a Rat Model. Each of these objects has a dedicated chapter in this manual explaining their function and how to modify and customize them to your needs. The objects are Coils in Chapter 4, Meshes explained in Chapter 5, HB-Meshes in Chapter 6, Ferrites described in Chapter 7, and Materials for when you want to calculate the critical current or other material dependent properties in Chapter 14.

The last two choices in the list are ‘Create Group’ and ‘Add Transformation.’ As their names might suggest, these two have an effect on the object selected in the Tree, such as grouping it into an Array or transforming the object by Rotating it with respect to its original position. Groups are explained in Chapter 12, and the Transformations in Chapter 13.

It may happen that while editing a node, you accidentally input a value or create an object that violates certain conditions, such as having an inner radius larger than the outer radius. In such cases, the object will be visually indicated by turning red or orange, depending on the selected Theme (Section 3.7.4) in the *model browser*. When you hover over the object, an error message will appear, providing specific details about the error and guiding you in resolving the issue. In addition, the variables affected by the error will be highlighted in orange or red in the *node editor* as well.

An example of this is shown in Figure 3.4.2. To address the error, you can check the values of the inner and outer radii or refer to the error message for further guidance. See Figure 3.4.2 for an example of an object that is incomplete and the resulting error message.

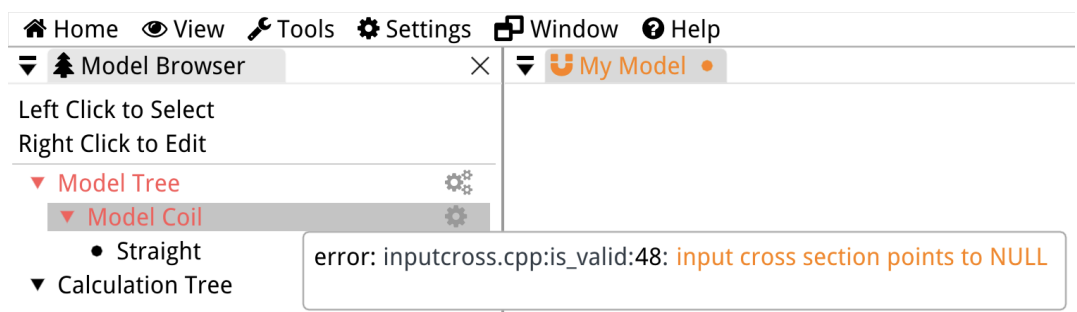


Figure 3.4.2: When an object can't be created because there is something wrong.

Finally, if you attempt to perform a calculation on a Model that contains errors, you will receive an error message in the Rat GUI, indicating that the Model contains invalid objects. In such cases, you will need to revert back to the objects in the Model Tree that are highlighted in orange or red and revise them so that they can be used without issues. This ensures that the Model is error-free and ready for accurate calculations.

### 3.4.2 Options for Editing the Calculation Tree

The Calculation Tree provides various editing options to modify the nodes within it, see Figure 3.4.3. When right-clicking on an object in the Calculation Tree, a menu with editing options is displayed. These options allow you to perform different actions on the nodes within the Calculation Tree.

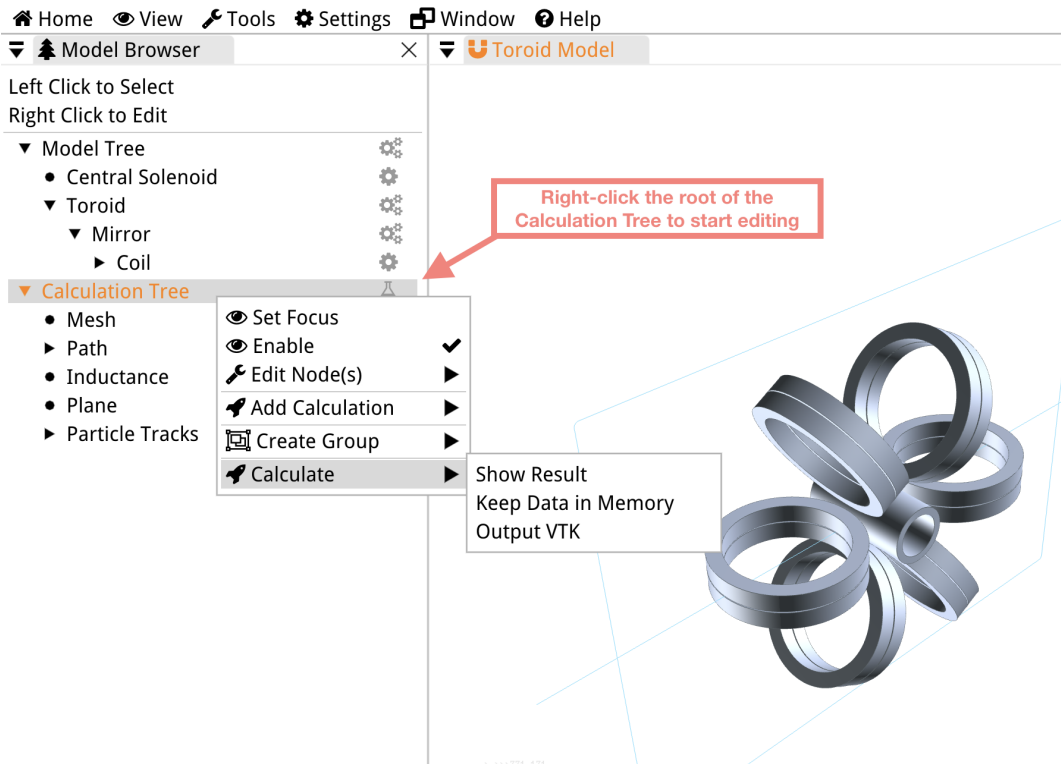


Figure 3.4.3: Right-click on an object in the Calculation Tree to view the options for editing Rat models.

The first three options are the same as for the Model Tree, and they have the same functionality. They allow you to set the focus, enable or disable objects, and edit nodes quickly using the tools in the ‘Edit Node(s)’ submenu.

Adding calculations to the Calculation Tree can be done by selecting the desired calculation from the ‘Add Calculation’ list. You can also group calculations by using the ‘Create Group’ function. This involves selecting the calculations you want to group, right-clicking the selection, and choosing the ‘Create Group’ option. Alternatively, you can create a group first in the Calculation Tree and then add calculations to it.

Additional editing options are available for child nodes in the Calculation Tree. These include adding a background magnetic field to most calculations, incorporating a powering circuit, and, for the Particle Tracking Calculation, including a tracking mesh and an emitter. Detailed explanations of these options can be found in Section 15.1.4.

Please note that calculations can be initiated either by clicking the ‘Calculate’ button in one of the *node editor* or by right-clicking a node in the Calculation Tree, hovering over ‘Calculate,’ and selecting the desired calculation procedure from the list. Further information on calculations can be found in Chapter 15.



## 3.5 The Node Editor

The *node editor* displays all the parameters and settings of the active or selected object in your Rat Model. As explained in the *model browser* and the *viewport* sections, you can select an object by either clicking on its node in the *model browser* or by clicking on the mesh in the *viewport*.

The *node editor* includes different types of fields that allow you to modify various settings. Boolean settings are represented as checkboxes, which can be enabled or disabled. Numeric inputs have fields that can be selected for entering the desired value. Additionally, there are ‘-’ and ‘+’ buttons next to number fields for incrementing or decrementing the value. The step size depends on the Model Scale, as explained in Section 3.7.4.

For settings like object names, you can click on the field to select it and enter the desired name. When setting a directory to read or save files, you can manually type the directory path or use the ‘+’ button on the right of the input field to browse your system.

In some cases, parameters may have a few available options, which are listed on the same line with circular input fields. Each labeled circle represents an option, and you can activate the desired option by clicking on the corresponding circle.

If a setting or a group of settings forms a function that can be graphically displayed, the *node editor* includes a graph for visualization. Hovering over the graph allows you to enlarge it for closer inspection. Examples of such functions include material properties (see Chapter 14) and Drives (refer to Section 3.5.1 for more information).

When a parameter or setting includes a unit, reflecting its physical property, the unit is displayed in the input field of the *node editor*. By default, Rat uses SI units, but this can be customized in the settings as explained in Section 3.7.4. To the right of each field in the *node editor*, you will see a question mark symbol, which serves as a tooltip. A tooltip is a small, informative message or hint that appears when you hover the cursor over the circle with the question mark. It provides additional information or a brief description about the purpose or functionality of that particular setting or parameter. Tooltips are also present next to headers in the *node editor*, providing general information about the object in the Rat GUI.

### 3.5.1 Drives

A Drive is a parameter that can be scaled as a function of time, space, or any other applicable variable. It allows for dynamic variations in a parameter over the course of a simulation, or varying a geometry property of your coil as a function of space. For example, in a calculation involving a Circuit, the current flowing through the Circuit can be defined as a Drive.

The behavior of the Drive can be customized using the drop-down menu next to the word **drive** in the *node editor*. By clicking the downward-pointing triangle next to the drive, you can access a menu that provides a selection of pre-defined functions.

The specific units of the scale function depend on the parameter to which the Drive is applied. For instance, if a Circuit Drive is added to a Calculation, the Drive will generate a function that describes the variation of current over time.

Rat offers several simple and commonly used Drives that are suitable for various scenarios. However, it is important to note that not every Drive is applicable to all parameters. Therefore, in this section, we provide a general introduction to Drives and explain how they are defined. It is ultimately the responsibility of the user to select the appropriate Drive for each specific parameter, considering its characteristics and requirements.

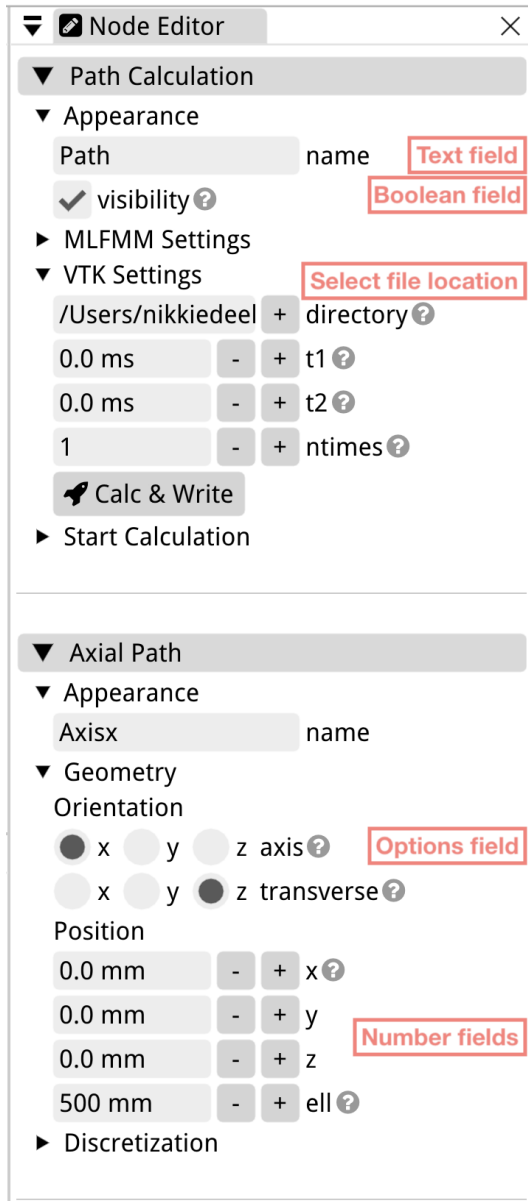


Figure 3.5.1: The different input fields of the *node editor*.

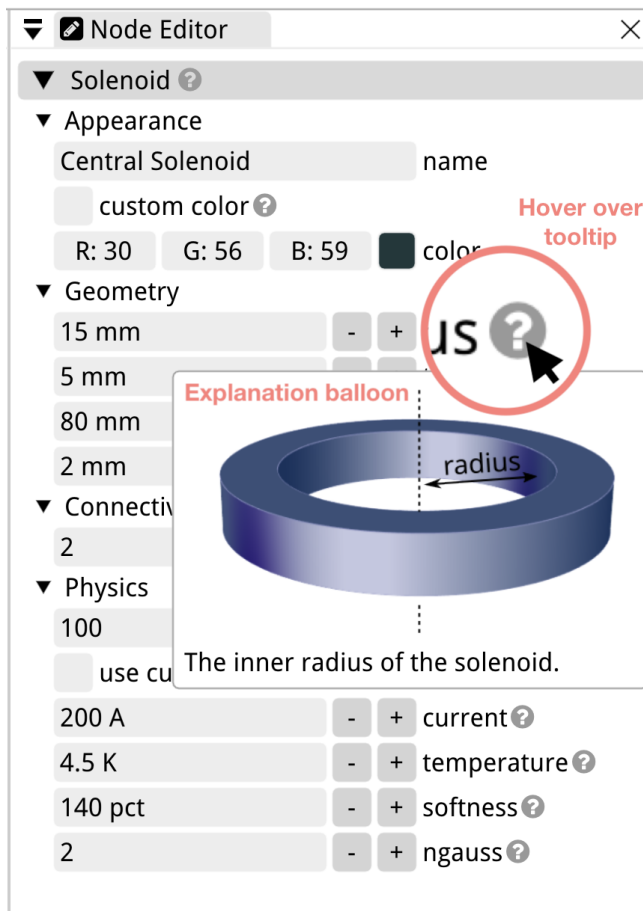


Figure 3.5.2: An example of a tooltip in the Solenoid *node editor*. When the cursor hovers over the question mark next to the **radius** parameter, the explanation balloon appears.

## Constant Drive

The Constant Drive maintains a fixed value and is not influenced by any independent variable. The value of the Constant Drive is determined by the **constant** parameter set in the *node editor*. Mathematically, the Constant Drive can be expressed as:

$$f(x) = C, \quad (3.5.1)$$

where  $C$  represents the value of the **constant**. This means that the Drive function will have a constant output equal to the specified constant value.

## Sine Drive

The Sine Drive allows you to scale a parameter with any independent variable, depending on your requirement. It is defined by the following equation:

$$f(x) = A \sin(2\pi f x + \phi + (v \cdot t)) + b. \quad (3.5.2)$$

In this equation,  $A$  represents the **amplitude** of the sine wave,  $f$  denotes the **frequency**,  $\phi$  represents the **phase shift** of the wave, and  $b$  corresponds to the vertical **offset** or baseline of the wave.

Additionally, the **vphase** setting in the *node editor* of the Sine Drive allows you to introduce a phase shift in the time dimension. This value can be specified in units of  $^\circ/s$ , turning it into a traveling sine wave.

## Linear Drive

The Linear Drive can be defined as a function of an independent variable,  $x$ , depending on the parameter that is being scaled. It is proportional to the slope,  $a$  (**slope** in the *node editor*), and is vertically offset by  $b$  (**offset** in the *node editor*), which is also the intersection with the vertical axis. It can be expressed as:

$$f(x) = ax + b. \quad (3.5.3)$$

## Trapezoid Drive

The Trapezoid Drive is used to define a pulse with a trapezoidal shape as a function of any independent parameter. The trapezoid Drive is defined by several parameters: the **baseline**, **pulse height**, **slope** of the legs, **length** of the plateau (short base of the trapezoid), **horizontal position**, and vertical position or **baseline**. Mathematically, the trapezoid Drive can be expressed as the following system of functions:

$$f(x) = \begin{cases} b & \text{for } x < x_1 \\ a \cdot (x - x_1) + b & \text{for } x_1 \leq x < x_2 \\ A + b & \text{for } x_2 \leq x < x_3 \\ A + a \cdot (x - x_3) + b & \text{for } x_3 \leq x < x_4 \\ b & \text{for } x \geq x_4, \end{cases} \quad (3.5.4)$$

where  $A$  represents the amplitude of the pulse,  $a$  is the slope of the two legs of the trapezoid, and  $b$  is the vertical offset that moves the baseline of the pulse. The length of the plateau, or short base, is determined by  $x_2$  and  $x_3$ , while  $(x_1 + x_4)/2$  is the center of the pulse.

If you want to have a trapezoidal Drive of which the two legs have a different slope, you can use the interpolation table in the next section.

## Interpolation Table Drive

The Interpolation Table Drive is constructed with a user-defined interpolation table, utilizing linear interpolation to draw lines between the points in the table. It can be expressed as:

$$f(x) = y(x_0) + \frac{y(x_1) - y(x_0)}{x_1 - x_0} \cdot (x - x_0), \quad (3.5.5)$$

where  $(x_0, y_0)$  and  $(x_1, y_1)$  are the two coordinates between which interpolation is performed. In the Rat GUI, the Interpolation Table Drive can be created by specifying the number of points to interpolate, `npoints`, and entering the coordinates of the horizontal and vertical axes in the interpolation table. The `pos/time` column represents the horizontal coordinates, and the `value` column represents the vertical coordinates. Alternatively, points can be imported from a file using the 'Import File' button.

## Barycentric Rational Interpolation Table Drive

The Barycentric Rational Interpolation Table Drive is constructed using a user-defined interpolation table with barycentric rational interpolation that results in smooth lines between the points. This technique approximates a function or data points using rational functions and is an extension of classical barycentric interpolation, which utilizes polynomial functions.

In the barycentric rational interpolation, the interpolating function is expressed as a ratio of two polynomial functions. The interpolation formula is as follows:

$$f(x) = \frac{\sum_{j=0}^n \frac{w_j}{x-x_j} y_j}{\sum_{j=0}^n \frac{w_j}{x-x_j}}, \quad (3.5.6)$$

where  $f(x)$  represents the interpolated function at a given point  $x$ ,  $n$  is the number of data points,  $x_j$  and  $y_j$  are the  $x$  and  $y$  coordinates of the  $j$ th data point, respectively, and  $w_j$  is the barycentric weight associated with the  $j$ th data point.

The barycentric weights are defined as:

$$w_j = \frac{1}{\prod_{\substack{k=0 \\ k \neq j}}^n (x_j - x_k)}, \quad (3.5.7)$$

where the denominator is the product of the differences between each pair of distinct data points'  $x$  coordinates. The order of barycentric rational interpolation refers to the degree of the polynomial functions used in the numerator and denominator of the rational function.

In the *node editor* you can select whether your points are given as a function of space or time with the `spatial` checkbox. The the degree of polynomials used can be set with the `order` setting. The `npoints` parameter defines the number of coordinates, and therefore also the number of rows, in your interpolation table. The `pos/time` column represents the horizontal coordinates, and the `value` column represents the vertical coordinates. Alternatively, points can be imported from a file using the 'Import File' button.

## Fourier Drive

The Fourier Drive is created using a Fourier series, as a function of any independent variable  $x$ . The general formula for the Fourier series representation of a periodic function  $f(x)$  with period  $T$  is:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{2\pi n}{T}x\right) + b_n \sin\left(\frac{2\pi n}{T}x\right) \right], \quad (3.5.8)$$

where  $a_n$  and  $b_n$  are the Fourier coefficients representing the amplitudes of the cosine and sine terms, respectively, at the  $n$ th harmonic. The  $a_0$  coefficient represents the DC-component or average value of the function over one period. These coefficients can be calculated using the following formulas:

$$a_0 = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) dx, \quad (3.5.9)$$

$$a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) \cos\left(\frac{2\pi n}{T}x\right) dx, \quad (3.5.10)$$

$$b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) \sin\left(\frac{2\pi n}{T}x\right) dx. \quad (3.5.11)$$

In the user interface, you can set the value of  $n$ , which determines the number of coefficients and therefore the number of rows in the coefficients table in the *node editor*. The first column contains the values of the  $a$  coefficients, while the second column contains the values of the  $b$  coefficients.

## S-Curve

This Drive has the shape of an S-Curve. In Rat, the S-Curve is not based on the sigmoid function. Instead, it is defined by several parameters that are explained in Figure 3.5.3.

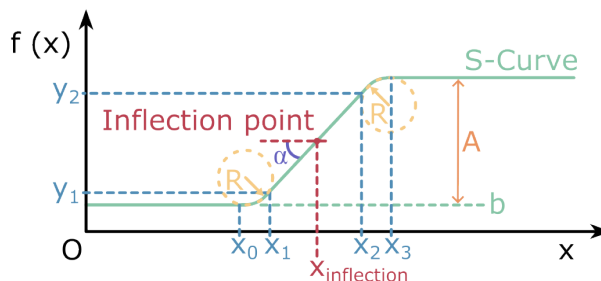


Figure 3.5.3: The S-Curve Drive used in the Rat GUI with definitions of different parameters that defined the shape of the curve.

The S-Curve can be defined with a system of equations, because this Drive is equal to a constant value at the start and end, in the middle, around the inflection point it rises with a constant slope and on either side of the slope the function is an arc with a radius  $R$ . The user can set the horizontal position of the inflection point,  $x_{\text{inflection}}$  with **inflection**,  $R$  with **radius**, the slope,  $a$ , of the linear part with **slope**, the amplitude ( $A$ ) of the Drive with **amplitude** and the vertical offset ( $b$ ) of the curve with **offset**. Based on these settings the

following parameters can be calculated:

$$\alpha = \arctan a \quad (3.5.12)$$

$$h_1 = R - (R \cos \alpha) = y_1 - b \quad (3.5.13)$$

$$h_2 = |A| - 2h_1 = y_2 - y_1 \quad (3.5.14)$$

$$l_1 = R \sin \alpha \quad (3.5.15)$$

$$l_2 = \frac{h_2}{a} \quad (3.5.16)$$

$$x_0 = x_{\text{inflection}} - \frac{1}{2}l_1 - l_2 \quad (3.5.17)$$

$$x_1 = x_{\text{inflection}} - \frac{l_2}{2} = x_0 + l_1 \quad (3.5.18)$$

$$x_2 = x_{\text{inflection}} + \frac{l_2}{2} \quad (3.5.19)$$

$$x_3 = x_{\text{inflection}} + \frac{l_2}{2} + l_1 = x_2 + l_1. \quad (3.5.20)$$

The S-Curve Drive can then be defined with the following system of equations:

$$f(x) = \begin{cases} b & \text{for } x < x_0 \\ b + \frac{A}{|A|}(R - \sqrt{R^2 - (x - x_0) \cdot (x - x_0)}) & \text{for } x_0 \leq x < x_1 \\ b + \frac{A}{|A|}(h_1 + a \cdot (x - x_1)) & \text{for } x_1 \leq x < x_2 \\ b + \frac{A}{|A|}(|A| - R + \sqrt{R^2 - (x - x_2)^2}) & \text{for } x_2 \leq x < x_3 \\ b + A & \text{for } x \geq x_3. \end{cases} \quad (3.5.21)$$

In addition, the **velocity** parameter in the *node editor* is used to control the movement of the S-Curve as a function of time, such as for creating a movie. It determines the velocity shift of the inflection point of the curve, in case the curve should move time.

### Frenet-Serret RCCT

The Frenet-Serret RCCT is a scaling function specifically designed for CCT coils (refer to Section 4.11). It calculates the winding angle to generate a perpendicular Frenet-Serret CCT. The mathematical calculations involved in this process are complex and are described in [reference to be provided].

The settings for this Drive should match the settings for the CCT that you want to model. **npoles** sets the number of CCT poles. The **is skew** checkbox is used to calculate the skew harmonic instead of the main harmonic. The inner radius of the CCT can be set with **radius**, and the winding angle with **alpha**. Finally, the **b** parameter controls the bluntness of the sine and cosine functions in the CCT path. For a more detailed description of CCT coils and CCT paths, please refer to Section 4.11 and Section 10.10.

## 3.6 The Processor

Once you run a calculation or multiple calculations in the Rat GUI, they will appear in the *processor* in the order they were started. The *processor* provides information about the status of each calculation, which can be either **running**, **waiting**, **finished**, **canceled**, or **error**. The process flow in the Rat GUI is such that first a process waits until the previous

process is finished. As soon as this is the case the state goes from **waiting** to **running** and the calculation is performed on your machine. If the program runs into an error the process changes state to **error**, but if it finishes normally it will change to **finished**.

The process flow in the Rat GUI is designed such that a process waits for the previous process to finish before starting. When the previous process is finished, the state changes from "waiting" to "running," and the calculation is performed on your machine. If an error occurs during the calculation, the process state changes to "error." On the other hand, if the calculation completes successfully, the state changes to "finished."

You have the option to cancel a running process by right-clicking its name in the *processor* and selecting 'Cancel.' Additionally, you can start or rerun a process from the same menu. By default, processes are executed consecutively, but you can manually start multiple processes simultaneously in the *processor* while the first one is running.

To quickly see the impact of any changes made to your model, such as modifications to the magnetic field, you can choose to 'Rerun' the calculation from the *processor* menu. During and after a process, you can monitor its progress by opening the logger using the 'Show Log' option from the same menu. Each process has its own logger, which is separate from the *console*.

After a process has finished and the results have been stored in memory or the *post-processor* was closed, you can review the results by right-clicking the process and selecting 'Result.' If you wish to remove processes from the *processor* list, you can use the 'Delete' option or right-click on 'Processes' at the top of the list and choose 'Clear All.' For a visual representation of the information described in this section, refer to Figure 3.6.1.

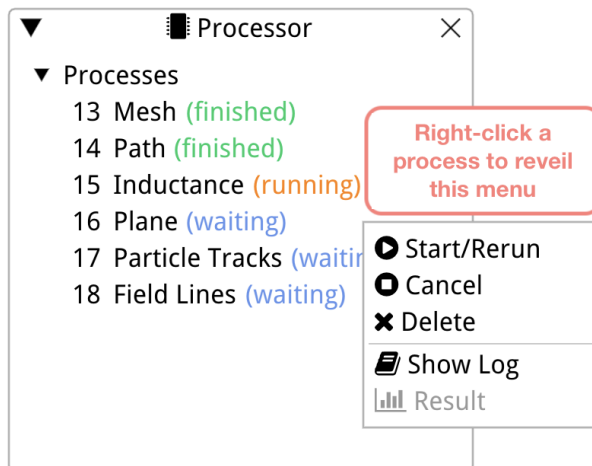


Figure 3.6.1: The *processor* and the options in its menu.

## 3.7 The Navigation Bar

The *navigation bar*, located at the top of the Rat GUI, provides a set of menus that offer various functionalities and options for navigating and customizing the user interface. Each menu in the navigation bar serves a specific purpose and provides a range of tools and settings that are essential for interacting with the Rat GUI effectively. In the following subsections, we will explore each menu in detail, discussing its features, options, and how to utilize them



to enhance your experience with the Rat GUI.

### 3.7.1 Home

In the Home menu the user can find all the usual options for opening, closing, and saving models, that can be found in most graphical user applications. Figure 3.7.1 shows all its available options, and most of them are self-explanatory. If an option has a key-bind it is shown to the right of that option in grey writing, such as ‘Ctrl+S’ for saving your model.

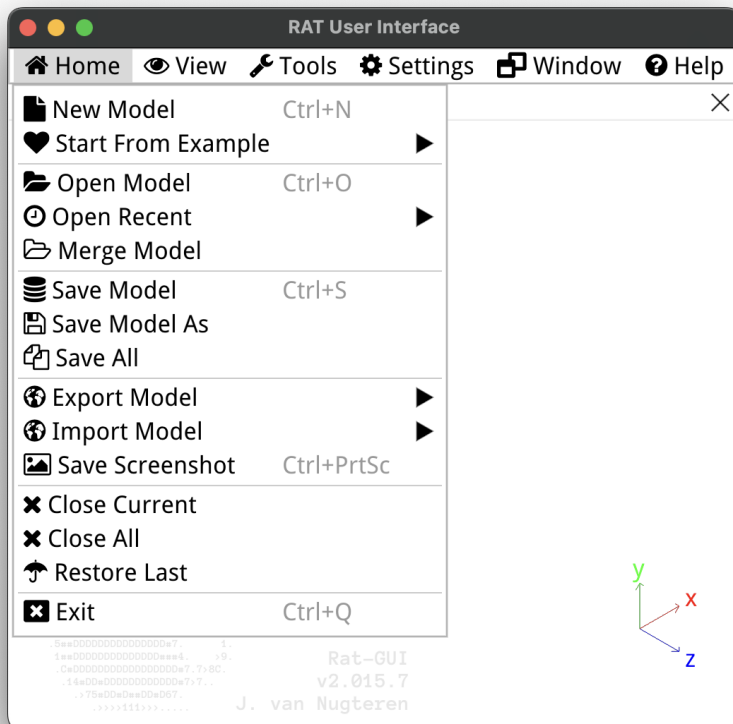


Figure 3.7.1: The Home menu.

Some of the items in this menu, such as ‘Start From Example,’ have more options in a sub-menu. Hover over the ‘Start From Example’ to see its available options. Click on one of the examples in the list to start viewing and using it. The ‘Open Recent’ menu shows you a list of models that you had previously open to allow quick access your own work. Click on one of your models to open them.

The ‘Export Model’ and ‘Import Model’ sub-menus contain a list of options to either export you model to a different file format, or import a model from a different program into the Rat GUI. Rat models can be exported to a FreeCAD macro, an Opera Conductor file, STL, Gmsh, and Abaqus meshes, an Edge matrix, and DXF Edge files. The two available importing options are ‘Solenoid Table’ and ‘Edge Matrix’.

## Export a FreeCAD Macro

FreeCAD is an open-source Computer-Aided Design (CAD) tool [5]. When you export your geometry from Rat to a FreeCAD macro, Rat generates a Python code that can be read by FreeCAD version 0.19 or higher. The macro can be imported into FreeCAD using the ‘Macro’ options in the FreeCAD *navigation bar*. Note that if you have a Rat model with individual cables modeled with many turns and layers, it may take some time for FreeCAD to import the macro.

## Exporting an Opera Conductor File

An Opera conductor file is a specific file format used by the Opera software for defining the geometry, material properties, and current distribution of conductors in electromagnetic simulations [6]. Currently, Rat models can only be exported to Opera conductor files; however, these files cannot be imported back into the Rat GUI.

If you wish to combine features from both Rat and Opera, it is recommended to build your geometry in the Rat GUI and then export it as an Opera conductor file for use in the Opera software. However, please note that if you need to make changes to your model, you should edit the geometry exclusively within the Rat GUI, as it is not possible to import Opera files back into the Rat GUI.

## Export an STL file

An STL (Standard Tessellation Language) file is a commonly used file format in 3D printing and CAD applications. It represents the surface as a raw, unstructured triangulated surface. Each triangle is defined by three points (vertices) in a Cartesian coordinate system and a direction indicator in the form of the normal vector. Together, these triangles create a detailed surface representation of the object. However, an STL file doesn’t include information about color, texture, or other specific details of the object—it focuses solely on the overall shape defined by the triangles. At the moment you can only export your Rat models to STL files, but it is not possible to re-import these files back into the Rat GUI.

## Export a Gmsh File

A Gmsh file is a type of file used in the field of computational physics and engineering, specifically in finite element analysis and mesh generation [7]. Gmsh is a popular open-source software tool used for creating, editing, and visualizing finite element meshes. A Gmsh file is a text-based file that contains information about the geometry, mesh elements, and boundary conditions of a computational model. The Gmsh file format is designed to be compatible with other software tools that utilize finite element analysis. Your Rat Model can easily be exported to Gmsh from the Home menu.

## Export a Abaqus File

The Abaqus input file is a text-based file that serves as the main input for setting up a simulation in the Abaqus software [8]. Using the Rat GUI, you can export your Rat model to the `.inp` file format, which is compatible with Abaqus. The exported file contains information about the model geometry, material properties, boundary conditions, loads, and analysis settings. It is created using Abaqus keywords and commands to define the desired simulation setup.

## Export an Edge Matrix

The edge matrix consists of four groups of three columns, with each group containing the  $x$ ,  $y$ , and  $z$  coordinates of one of the four edges of your coil mesh(es). The edge of a coil (mesh) is defined as boundary of two adjacent faces. The number of rows in the edge matrix is equal to the number of nodes in each of the edge paths. This is illustrated in Figure 3.7.2.

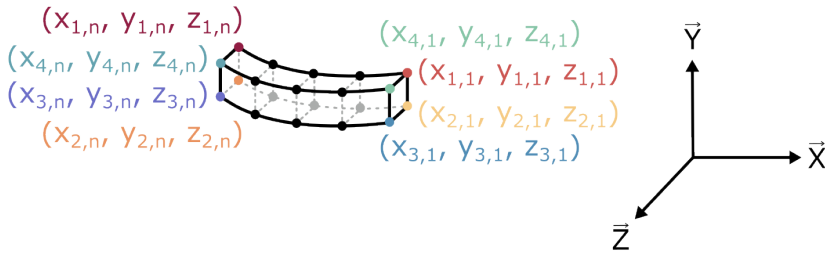


Figure 3.7.2: The coordinates of the four edges of a piece of conductor demonstrating how edge coordinates are defined.

The exported file starts with a header line containing the name of the object. The following lines contain comma-separated coordinates of each edge. The file can include matrices with edges of multiple coils. In that case, the matrices are stored one below the other, separated by an empty line and a header for the next coil. Refer to Figure 3.7.3 for an example of an edge matrix file. To see a real example, try exporting a Rat model to an Edge Matrix and open it with your favorite text editor (e.g., Sublime).

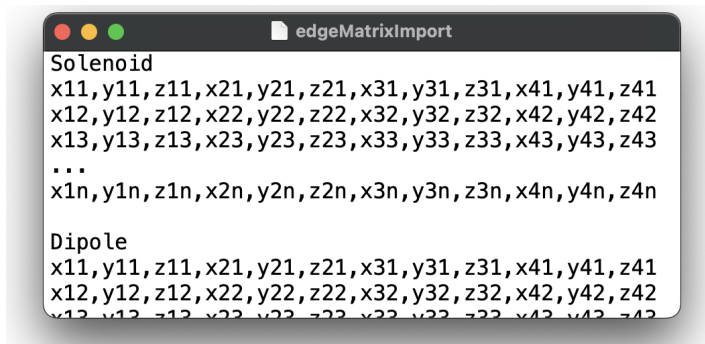


Figure 3.7.3: An example of an edge matrix file. The first parameters  $r_{n,m}$  need to be replaced by the correct values for the edges of each of the coils.

## Export a DXF File

Drawing Exchange Format (DXF) is a widely used file format for exchanging 2D and 3D design data between different CAD software applications. It was developed by Autodesk as a universal format for interoperability among CAD programs [9]. DXF files are saved in a text-based format, making them human-readable and editable using a standard text editor. In the Rat GUI, you can easily export your Rat model to a DXF file by selecting the ‘Export’ option in the Home menu.

## Import a Solenoid Table

A solenoid table can be as simple as four values specifying the inner radius of the solenoid  $R_{\text{in}}$ , the outer radius,  $R_{\text{out}}$ , the lowest z-coordinate,  $Z_{\text{low}}$ , and the highest z-coordinate of the solenoid,  $Z_{\text{high}}$ , assuming its central axis is parallel to the z-axis, refer to Figure 3.7.4a. The file needs to be TXT or CSV formatted and have a header, but this can be any header the user defines as long as it is comma separated. The file can be extended to include the ampere-turns in the solenoid, see Figure 3.7.4b, and you can offset the center of the solenoid in  $x, y, z$  as in Figure 3.7.4c. Note that if you want to specify an offset, you need to also have the ampère-turns in the file, the order of the columns is important.

```

mySolenoid0.csv
Rin,Rout,Zlow,Zhigh
250e-3,300e-3,-5000e-3,5000e-3
450e-3,500e-3,-5000e-3,5000e-3

```

(a) Simplest Solenoid Table.

```

mySolenoid1.csv
Rin,Rout,Zlow,Zhigh,Ampt
250e-3,300e-3,-2500e-3,2500e-3,10000

```

(b) Solenoid Table with Ampere-Turns.

```

mySolenoid2.csv
Rin,Rout,Zlow,Zhigh,Ampt,x,y,z
250e-3,300e-3,-5000e-3,5000e-3,10000,-400e-3,0.0,0.0
450e-3,500e-3,-5000e-3,5000e-3,10000,200e-3,0.0,0.0

```

(c) Solenoid Table with Ampere-Turns and offset.

Figure 3.7.4: Three example files that can be imported as a solenoid table.

Figure 3.7.5 shows an example of importing the a Solenoid Table in the Rat GUI, in that case mySolenoid1.csv from Figure 3.7.4b was imported, refer to the Model Tree in the orange rectangle. The Rat GUI translates the file to a solenoid with inner radius 250 mm, a wall thickness of 50 mm, and a height of 5 m. The ampère-turns show up in the current, see the green rectangle of Figure 3.7.5.

As a simple check the field in the center of the solenoid,  $B_{\text{center}}$ , can be calculated analytically with

$$B_{\text{center}} = \mu_0 \frac{NI}{l} \tag{3.7.1}$$

$$= 1.256 \times 10^{-6} \text{N A}^{-2} \cdot \frac{100 \cdot 100 \text{ A}}{5 \text{ m}} \tag{3.7.2}$$

$$= 0.0025 \text{ T}, \tag{3.7.3}$$

where  $\mu_0$  is the magnetic constant,  $N$  the number of turns,  $I$  the current in the solenoid, and  $l$  the length of the solenoid. As can be seen in Figure 3.7.5 the same results is obtained with Rat.

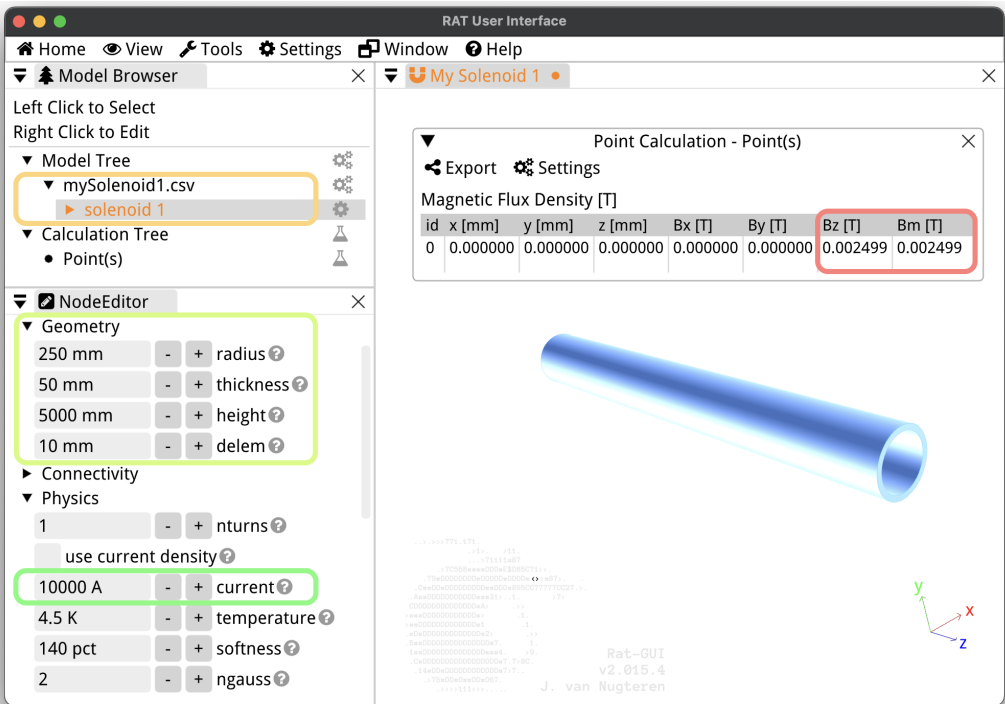


Figure 3.7.5: The results of importing mySolenoid1.csv in the Rat GUI.

## Import an Edge Matrix

The Rat GUI allows you to not only export a Rat model to an Edge Matrix file but also import Edge Matrices. Please refer to Section 3.7.1 for a description of the edge matrix.

When importing an edge matrix into the Rat GUI, you are importing a mesh. However, you still need to set the current in each of the coils. This can be done by adjusting the **current** setting in the yellow rectangle shown in Figure 3.7.6. Additionally, you have the option to increase the number of nodes in the normal (**nnormal**) and transverse (**ntransverse**) directions. The number of nodes in the longitudinal direction is determined by the number of elements in your file and should be equal to the number of rows in the edge matrix. Since an edge matrix allows you to define any shape of coil with a conductor made of four edges, the only way to edit this mesh in the Rat GUI after importing is by manually changing the  $x$ ,  $y$ , and  $z$  values in the *node editor*, as shown in the red rectangle in Figure 3.7.6.

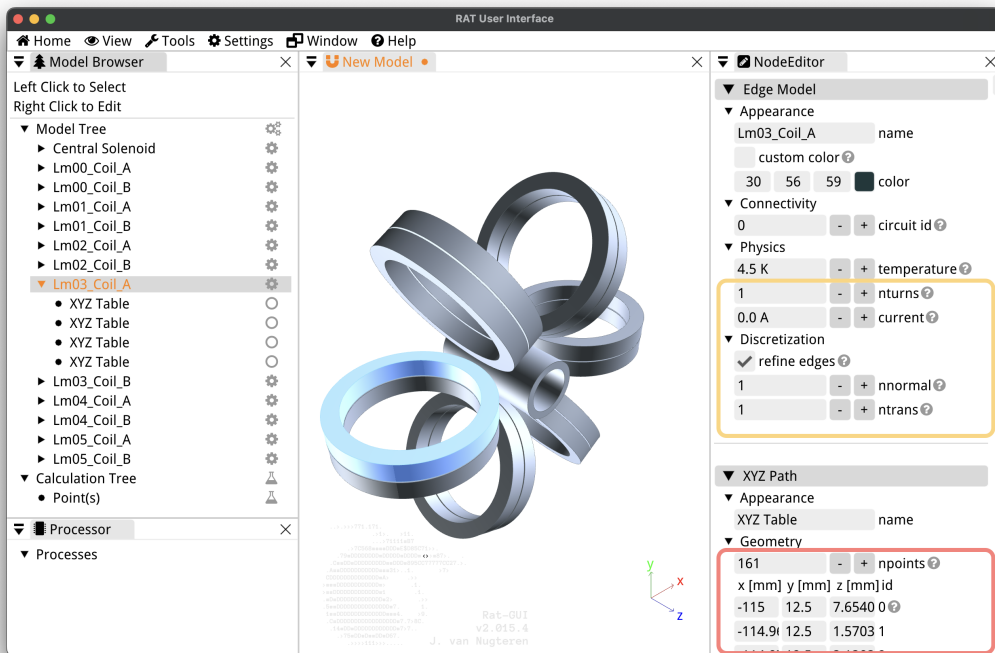


Figure 3.7.6: The result of re-importing the edge matrix of the Mini Toroid example.

### 3.7.2 View

The View menu provides options to customize the visualization of your model in the *viewport*, as shown in Figure 3.7.7. The first feature in the View menu allows the user to cycling through multiple open models. When you have multiple models open simultaneously in the Rat GUI, only the active model, which is the one visible in the Model Tree, can be edited in the *node editor*. The cycle models option allows you to quickly switch between open models, activating a different model each time.

The next set of options is used for zooming in or out on your model. The ‘Zoom Extents’ option adjusts the zoom level and camera position to ensure that the entire model or scene fits within the *viewport* without any clipping or cropping. The ‘Reset View’ option performs

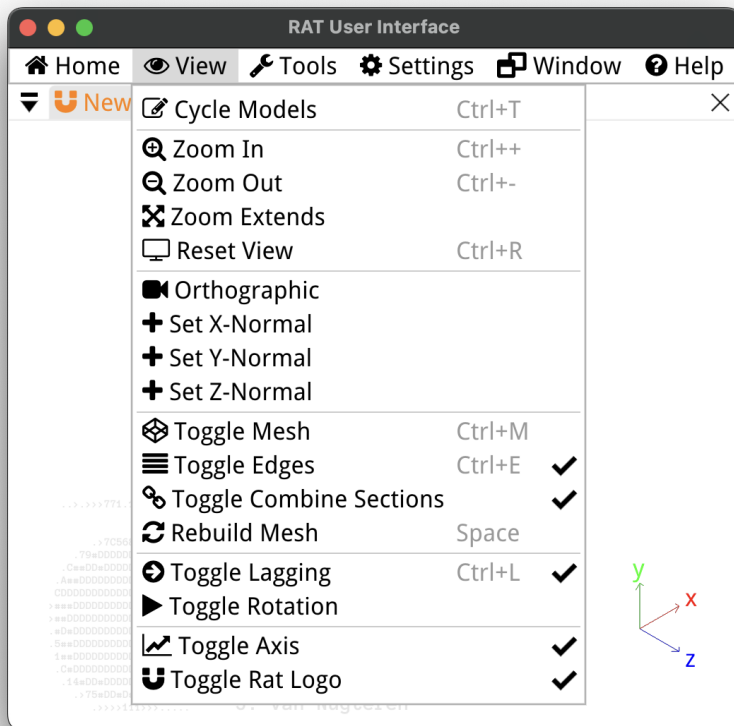


Figure 3.7.7: The View menu.

the same function and also resets the model to its default position in the Rat GUI if it has been rotated in the *viewport*.

By default, the *viewport* in the Rat GUI displays all objects in your model using a perspective view, which mimics how our eyes perceive objects in real life. However, there is also an option to switch to an orthographic view, which displays the 3D object without any perspective.

In an orthographic view, the object is typically depicted as if viewed from an infinite distance, resulting in parallel lines that do not converge towards a vanishing point. This means that the object's size, shape, and proportions remain consistent across the entire image. To switch the *viewport* to an orthographic view, you can select the 'Orthographic' option.

Both perspective and orthographic views allow you to rotate the camera quickly along one of the Cartesian axes. This can be done by using the 'Set X-Normal,' 'Set Y-Normal,' or 'Set Z-Normal' options to align the camera with the desired axis.

The next set of features in the view options controls how the meshes are displayed. By default, the individual elements in the mesh are not indicated with lines or a different color. However, you can enable the display of these elements by toggling the 'Toggle Mesh' option. When enabled, the elements of the mesh are highlighted with lines that are slightly darker than the surface of the mesh. This can help when you are changing the element size of your model. The edges of the mesh on the other hand are by default darker than the mesh surface. You can turn off this highlighting by using the 'Toggle Edges' option. Additionally, certain paths (refer to Chapter 10) are split into sections with the `nsections` option, and these sections can be visualized on the surface of the meshes by toggling the 'Toggle Combine Sections' option.

In rare cases, you may encounter an error while editing your model in the Rat GUI, resulting in the meshes in the *viewport* not being up to date. If such an issue occurs, you can try to resolve it by using the 'Rebuild Mesh' option in the view menu. If you believe you have encountered an error, you can view the associated error message in the *console*, as explained in Section 3.7.5.

As explained in Section 3.3, you can control the camera in the *viewport* of the Rat GUI using your mouse. While holding down a mouse button, the camera moves in response to your mouse movements. Once the mouse button is released, by default, there is a slight delay in the movement of the meshes. You can toggle this lagging effect on or off using the 'Toggle Lagging' option.

Furthermore, there is a 'Toggle Rotation' option available. When activated, it enables a screen-saver mode where the camera continuously rotates horizontally around the meshes. This feature allows for effortless observation of the model from various angles without the need for manual camera rotation. You can continue editing the models from the node editor when it is rotating.

By default, in the two bottom corners of the *viewport*, the orientation of the Cartesian axes is displayed on the right side, and the Rat logo is displayed on the left side. However, you have the option to hide these elements if desired. The 'Toggle Axis' option allows you to hide or show the display of the Cartesian axes, while the 'Toggle Rat Logo' option allows you to hide or show the Rat logo.

### 3.7.3 Tools

The Tools menu provides a range of tools for editing your model. These tools apply to the *model browser*, as explained in Section 3.4, where you can find the Model and Calculation Trees. Similar to most software, you can perform actions such as cutting and pasting objects



to different locations in the Trees, grouping them, swapping the order, or duplicating them. Please refer to Figure 3.7.8 for a complete list of Tools.

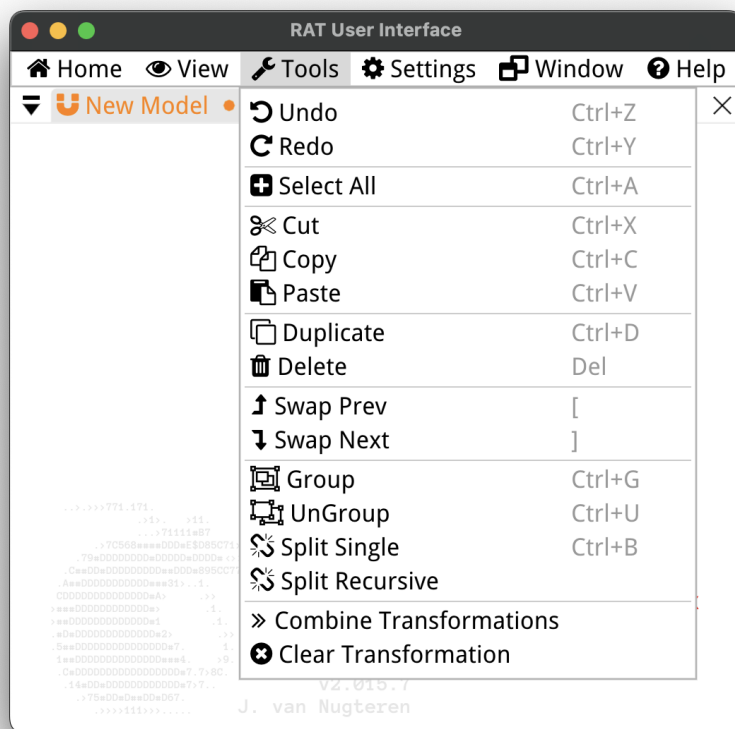


Figure 3.7.8: The Tools menu.

There are four tools in the Rat GUI that require further explanation: Combine Transformations, Clear Transformations, Split (Ctrl+B) and Recursive Split. Sometimes, you may have created multiple transformations for a single object in your model. When you use the Combine Transformations tool on that object, Rat attempts to merge these transformations into a single transformation. For example, if there was a translation of 100 mm in the  $x$ -direction and a separate translation of 250 mm in the  $z$ -direction, these can easily be combined into a single transformation object. On the other hand, when you clear all transformations of an object in the Model Tree, all transformations associated with that object are simultaneously deleted.

The Split tool divides the selected object into its components that are one level below the parent. For instance, a Coil object is split into a Path and a Cross-Section. If transformations were applied to this object or it was part of a group, those transformations are also split out. You can recursively split objects to reach the lowest-level components that make it up. This can be a useful tool if you want to understand how an example coil was created or examine the building blocks of the standard coils in the Rat GUI (see Chapter 4). The Recursive Split tool operates in a similar fashion, but with an extended functionality: it recursively splits all objects to their maximum extent. In other words, if an object's child also has children, the

tool will continue to split until further splitting is no longer possible.

### 3.7.4 Settings

In the settings menu, you can find options to customize the appearance of the Rat GUI, as shown in Figure 3.7.9. The theme of the Rat GUI determines the colors used in the application. The default theme is ‘Classic,’ which is a dark theme, but most figures in this manual are created using the ‘Purewhite’ theme. You can also customize the font used for the text in the Rat GUI. There are four different fonts available in the settings, and you can change the font size for each of them in their respective submenus. The default font is DroidSans with a font size of 16.

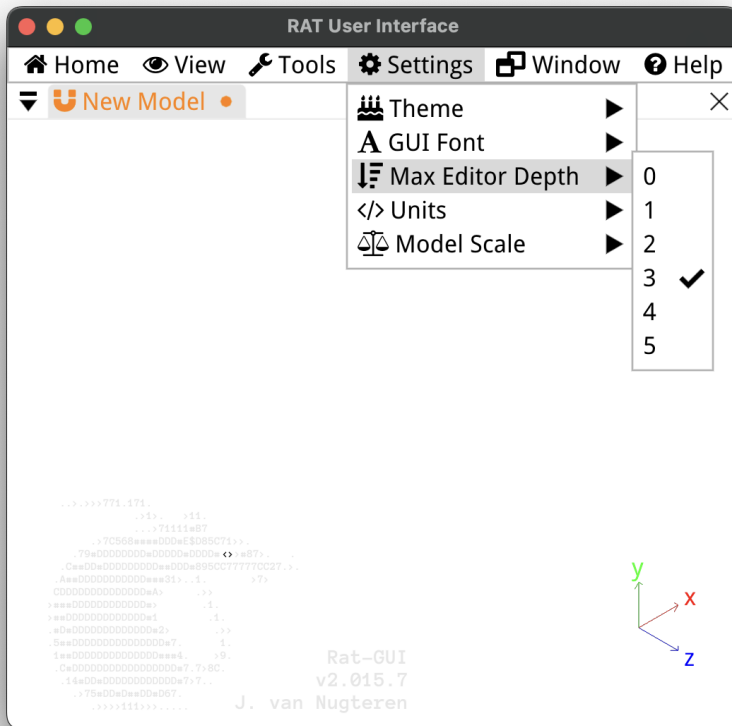


Figure 3.7.9: The Settings menu.

The ‘Max Editor Depth’ setting affects the number of objects displayed in the node editor (Section 3.5) when a root or internal node is selected in the *model browser* (Section 3.4). It determines the maximum number of nodes that are displayed in the *node editor*. For example, if it is set to 3 and node ‘Z’ in the model tree has five child nodes, you will only be able to edit node ‘Z’ and the first two child nodes in the *node editor* when node ‘Z’ is selected. To edit the lower-level nodes, you can select them in the tree or select a parent node that is no more than two levels higher in the tree.

When a model parameter has a unit, that unit is displayed in the *node editor*. By default, most parameters use SI units, such as meters for length. However, if you prefer to

use millimeters instead of meters for setting length parameters, you can change this in the unit settings.

The last setting is the ‘Model Scale.’ It determines the scale of an object when it is initially drawn and also affects the step size when using the plus or minus buttons. By default, ‘Autoscale’ is enabled, which calculates the model scale based on the sizes of the existing coils in your model. If you disable autoscale and set the model scale to meters, for example, the next solenoid you draw will have a radius of 0.5 m.

### 3.7.5 Window

The Window menu has the complete list of available windows in the Rat GUI. By selecting, or deselecting a window in the list, it will be opened or closed respectively. These are the *model browser* (Section 3.4), the *node editor* (Section 3.5), the *processor* (Section 3.6), the *Renderer* (Section 3.7.5), the *Console* and the *Performance* windows. By default the Model browser, the *node editor*, and the *processor* are shown in the Rat GUI, as can be seen in Figure 3.7.10, where these three window options are checked.

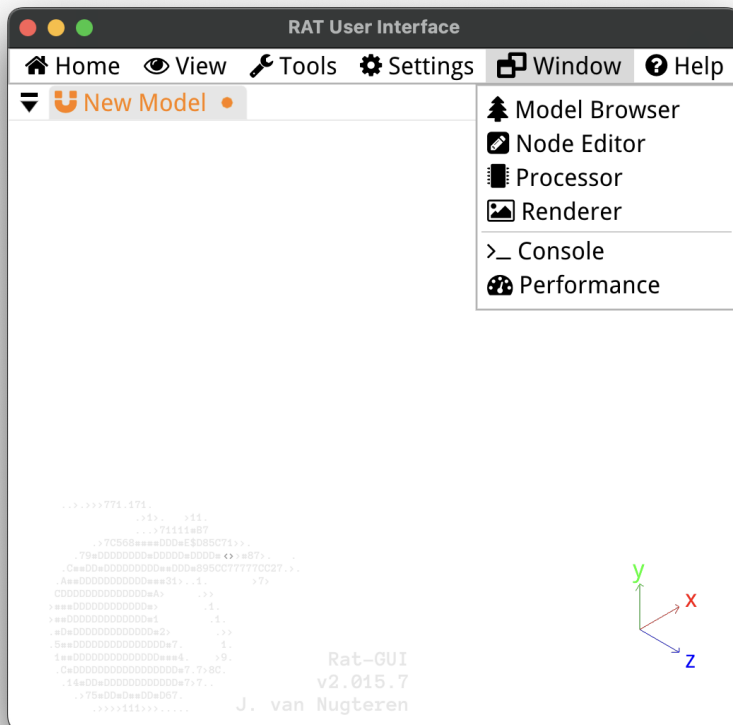


Figure 3.7.10: The Window menu.

## The Renderer

The Renderer window is used to render high-resolution images of your models or render all frames to create a GIF file of a rotating version of your model. If you have multiple models open, you can select the one you want to render from the **viewport** list. By default, it renders the active **viewport**, which is the one with its title highlighted in color.

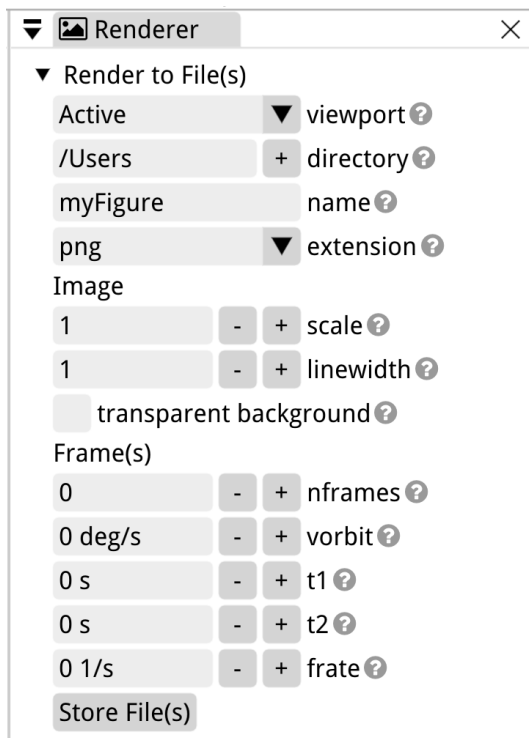


Figure 3.7.11: The Renderer window.

Below the **viewport** selection, you can choose the location on your computer to store the file using the **directory** setting. The filename is specified separately in the **name** setting. The file extension can be selected from the options in the **extension** setting, including bmp, jpg, png, and tga.

The resolution of the rendering can be adjusted using the **scale** factor. The rendering resolution will be the resolution of your screen multiplied by the scale. Increasing the **linewidth** setting will make the edges of the meshes thicker. When the **transparent background** option is selected, the rendering will have no background.

Below the "Frame(s)" options, you can save multiple images of your model rotating from one frame to the next. In Linux, a GIF file is automatically generated. This feature will be available on Windows and MacOS in the future. Currently, on those platforms, individual frames are generated, and it is up to the user to save them as a GIF file.

The total number of frames to be rendered is set with **nframes**. The frame rate, **rframe**, is only used in Linux. It represents the number of frames per second played in the GIF file. On all platforms, the **vorbit** setting controls the rotational speed of the camera that captures the different frames. This rotation happens in the background and cannot be observed in the Rat GUI. You can also specify the start and end times of the virtual recording using  $t_1$

and  $t_2$ , where  $t_1$  is the start time and  $t_2$  is the end time.

For example, to generate 360 frames of your model, with each frame rotated by  $1^\circ$  from the previous one, you can set `nframes` to 360, `vorbit` to  $2^\circ/\text{s}$ ,  $t_1$  to 0s, and  $t_2$  to 30s. If you are using Linux and want to create a GIF of one revolution of your model, taking 10s, set `rframe` to 36 Hz.

### The Console and Performance Window

Both the Console and the Performance window serve diagnostic purposes. Each user mouse click is registered in the Console. All the information in the Console can be exported to a text file by selecting ‘Export > Text File’ from the *navigation bar* of the *console*, refer to Figure 3.7.12. The console settings allow the user to enable or disable auto-scrolling, select the number of lines displayed in the console history, and clear the history.

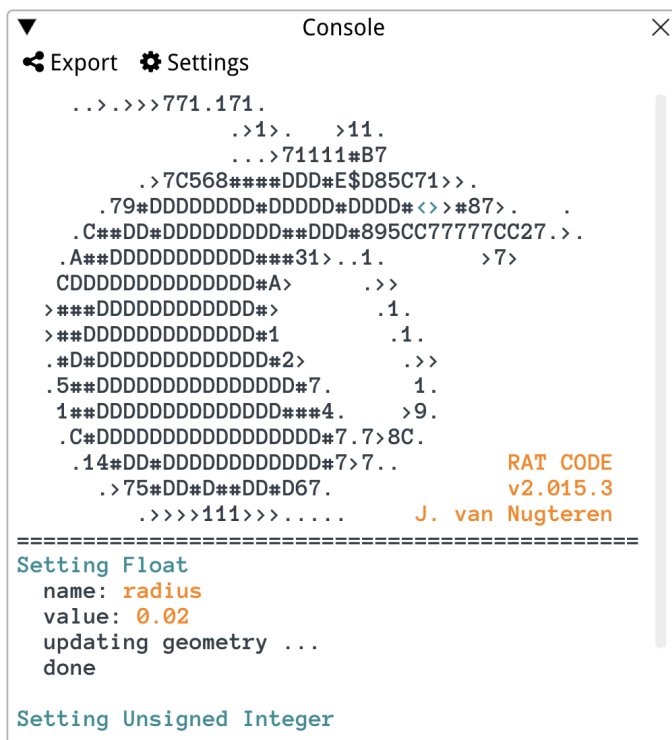


Figure 3.7.12: The *console*.

The Performance window displays the Frames Per Second (FPS), the number of nodes, triangles, and edges in a model, the Model Scale (MSC), the camera target in  $[x, y, z]$  coordinates, the trace (depth in the model tree), and the selection status of an object, see Figure 15.1.3. The Performance window does not have configurable settings and is solely used for diagnostic purposes.

### 3.7.6 Help

In the Help menu you can find useful information about the software, the license and serial key, and links to the websites of Rat GUI and Little Beast Engineering. Please refer to

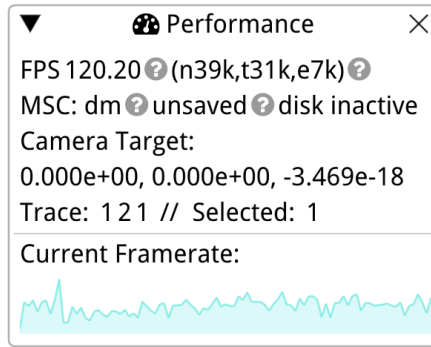


Figure 3.7.13: The Performance window.

Figures 3.7.14 and 3.7.15 for images of the help menu and the About window.

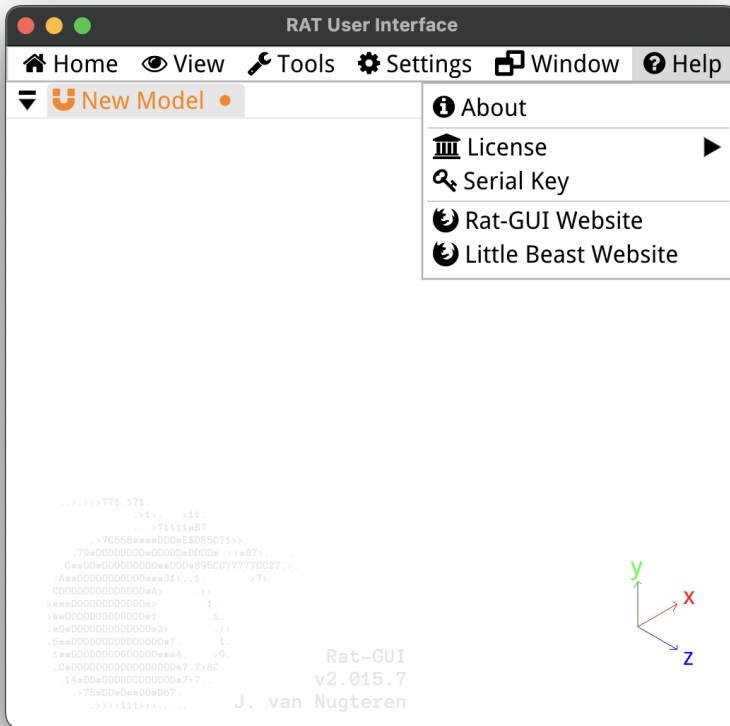


Figure 3.7.14: The Help menu.

Clicking on ‘Serial Key’ in the Help menu opens the License Manager, where you can view your license key and the machine activation status, see Figure 3.7.16. You can change the license key by entering the serial key numbers in the provided text boxes. Note that you first need to deactivate the machine before entering a new serial key. The License Manager allows



Figure 3.7.15: The About window that can be opened via the Help menu.

you to activate or deactivate the software on the current machine using the corresponding buttons. While the License Manager window is open, the Rat GUI cannot be used. To close the window, click the associated ‘Close Window’ button.

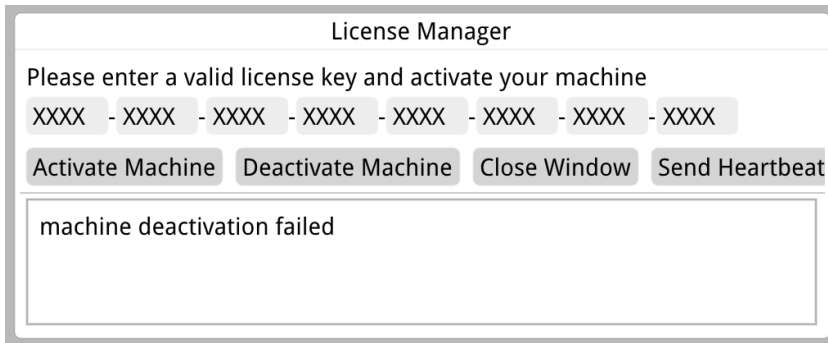


Figure 3.7.16: The License Manager. The serial key in this example is a fictive key of X’s.

The License Manager in Rat GUI uses a heartbeat mechanism to check if the Rat GUI is still running or if it was closed accidentally due to a crash, for instance. Every 10 minutes, the Rat GUI sends a heartbeat signal to the License Server. If the server receives the heartbeat correctly, it knows that the software is still running on your machine. If the server does not receive a heartbeat after 10 minutes, it releases the license for a new Rat GUI instance. The user can manually send a heartbeat by clicking ‘Send Heartbeat’ in the License Manager. However, if the software is functioning correctly, this option is unnecessary.



# Chapter 4

## Coils

### 4.1 Introduction

A Coil in Rat is a three-dimensional object or mesh that is capable of conducting current, and is used to model electromagnetic coils. The geometry of a Coil can be predefined or custom. It can be a standard shape like a Solenoid or a D-shape, or it can be defined by a Path and Cross-Section in the case of a Custom Coil. Alternatively, the coordinates of the edges can be specified by the user for the Edges Coil. The available Coil types and their corresponding sections in this manual are summarized in Table 4.1.1.

Table 4.1.1: Coil objects in the Rat GUI.

Name	Description	Section
Custom	Defined by Path and Cross-Section	Section 4.2
Edges	Defined by $x, y, z$ -coordinates of four edges	Section 4.3
Solenoid	Wrapper Model for solenoid coils	Section 4.4
Racetrack	Wrapper Model for racetrack coils	Section 4.5
Rectangle	Wrapper Model for rectangular coils	Section 4.6
Trapezoid	Wrapper Model for trapezoidal coils	Section 4.7
D-shape	Wrapper Model for D-shape coils	Section 4.8
Flared	Wrapper Model for a racetrack with flared ends	Section 4.9
Clover	Wrapper Model for cloverleaf-shaped coils	Section 4.10
CCT	Wrapper Model for a two-layer straight Canted-Cosine Theta coil	Section 4.11
Plasma Ring	Wrapper Model of a 3D plasma surface	Section 4.12

To add a Coil to your model, right-click on the Model Tree, choose ‘Add Coil,’ and select a Coil type from the list. If you want to inspect the lower-level objects that define a Coil, you can use the Split tool described in Section 3.7.3. Coils can be grouped, and you can apply various transformations to a Coil or a Group of Coils.

#### 4.1.1 General Coil Settings

Coils can be edited in the *node editor*, where all the settings and parameters are displayed when a Coil is selected. These settings are automatically saved in the same file as your

Rat Model. Each Coil has the following types of settings: Appearance, Geometry and Discretization, Connectivity, and Physics settings, as shown in Figure 4.1.1. The specific geometry and discretization settings vary for each Coil and are discussed in the relevant sections of this chapter.

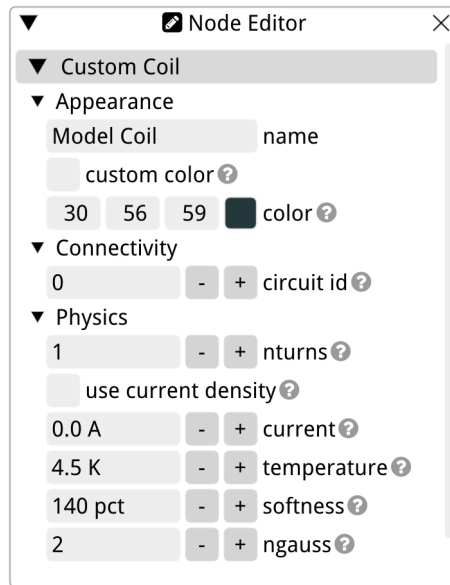


Figure 4.1.1: The general settings in the *node editor* of a Coil.

The appearance settings allow you to name the Coil using the **name** setting and change the color of its mesh in the *viewport*. This feature is particularly useful when using the Rat GUI to render your Model (refer to Section 3.7.5 for more information), as it enables the display of different meshes in different colors. You can specify the Red, Green, and Blue (RGB) values in the corresponding input fields or use the color picker by clicking on the colored square. Double-click on the desired color to activate it, and don't forget to check the **custom color** setting to override the standard mesh color defined in the current theme (see Section 3.7.4).

Connectivity settings include a single parameter called the circuit index, or **circuit id**. The circuit index allows you to group Coils with the same index into circuits, which can be useful for length and inductance calculations. Additionally, a Circuit can be added to a Calculation to drive a time-dependent current in the Coil. In the first case, inductance and length are calculated for circuits rather than individual coils.

The physics settings are used to convert a mesh without physics properties into a magnet model. The first setting is the number of turns (denoted as **nturns**), which represents the total number of turns in the coil, including all layers of conductor windings. For example, if you are designing a Solenoid Coil (refer to Section 4.4) and intend to wind it in two layers with 100 turns each, you would set **nturns** to 200.

The current in the magnet is set using the **current** parameter. By default, the operating current is set in A, but depending on your unit settings you can also use mA for instance, see Section 3.7.4). The **current** refers to the current delivered by your power supply. If you prefer to set the current density instead, please check the **use current density** checkbox located above the **current** parameter and specify the current density in the corresponding input field of the *node editor*. It is important to note that when adding a Powering Circuit

to a calculation later, the `current` setting will be overridden by the value set in that Circuit (refer to Section 15.1.4).

The operating temperature of the magnet can be set as well, and this is especially useful when modelling a superconducting magnet. Use the field labeled with `temperature` and add material properties to the coil to enable calculating temperature margin, or critical current for example. Refer to Chapter 14 for instructions on setting material properties and Section 15.6 for applicable calculations.

The last physics setting is the number of Gauss points used, denoted as `ngauss`. This setting determines the number of line elements within each hexahedron mesh element of the your Coil. The specified number of Gauss points is used in one dimension. The same number of Gauss points is applied in the second dimension, resulting in a total of  $n_{\text{Gauss}}^2$  Gauss points in the cross-section. Line current elements are then drawn from one face of a mesh element to the other, acting as current sources in the magnetic field calculations. This ensures conservation of current between adjacent mesh elements. For a visual representation of the number of Gauss points, refer to Figure 4.1.2.

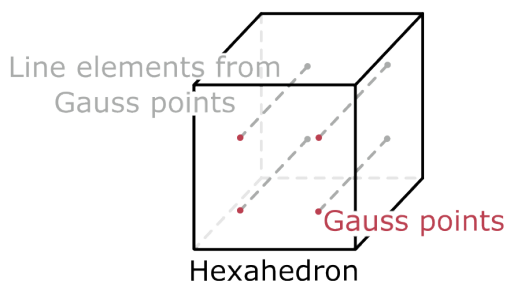


Figure 4.1.2: An illustration of the Gauss points in a hexahedron mesh element and the line current elements that are the sources of the magnetic field calculations.

## 4.1.2 Wrappers versus Customizability

In the remainder of this chapter, you will find a section for each Coil wrapper available in the Rat GUI, as well as for the Custom Coil and Edges Coil, while they are not wrappers. As you will see, most of the wrappers, which are also called Model Coils, have a rectangular coil pack, or Cross-Section, and the main difference lies in the Path used to create the Coil wrapper. The advantage of using a wrapper is that the user only needs to focus on the general shape of the Coil. For example, in a Solenoid Coil, you will simply set the height of the solenoid you are modeling, whereas in the Custom Coil, you would need to adjust the settings of the rectangular Cross-Section to achieve the desired height.

A handy feature in the Rat GUI that you can use on any Coil wrapper is the ‘Split’ function, or simply press `Ctrl+B`. This breaks the wrapper into its Cross-Section and Path, while maintaining the geometry. If you then want to change the Cross-Section from a rectangular shape to a circular shape you can easily do this by right-clicking on the Coil node and selecting ‘Set Input Cross-Section.’ Be careful, you can only go back to the Coil wrapper with the ‘Undo’ function (`Ctrl+Z`). This means that if you change the cross-section after breaking the Coil you can’t remake it into a wrapper and continue editing the wrapper settings.

The discretization settings of wrappers usually only have one parameter: the target element size in each dimension, `delem`. The software uses this value to build the cubic mesh with an element size that is (close to) this value. This parameter is always illustrated in each section in the same figure where the geometry parameters of that specific wrapper are explained.

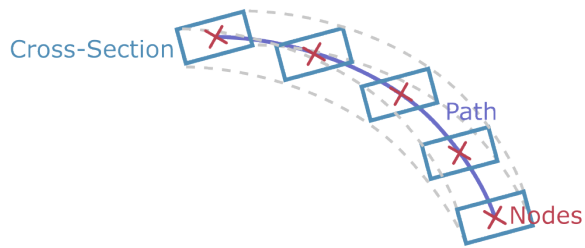
When you use the split command (**Ctrl+B**) on a wrapper Coil, you see with which Cross-Section and Path it was built. In most cases the Cross-Section is rectangular, refer to Section 9.5. Using the split command also allows for more control over the discretization, as it will enable you to set a different element size for each of the three dimensions.

## 4.2 The Custom Coil

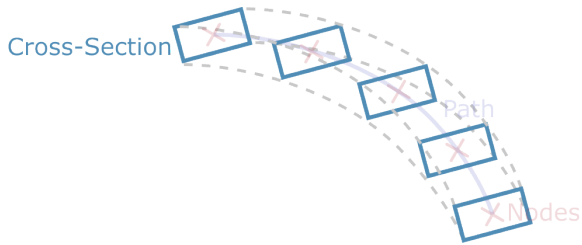
The Custom Coil in Rat is created using a user-selected Cross-Section (refer to Chapter 9) and Path (see Chapter 10). When added to the *model browser*, it will appear red in the Model Tree until you have selected both the Cross-Section and Path. To select the Cross-Section and Path, right-click on the Model Coil in the Model Tree and choose ‘Add Path’ and ‘Add Cross-Section,’ respectively.

In the Rat software a Custom Coil is build by a Cross-Section that is virtually extruded along a Path. This is illustrated in Figure 4.2.1. The red crosses in this figure represent the node of each of the elements of a Coil’s mesh. The user can control the element size with the designated element size parameter of the Cross-Section and Path.

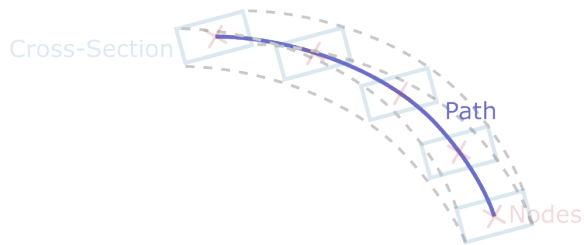
The Custom Coil does not have specific geometry settings, only the general Coil settings (Section 4.1.1). Instead, it relies on the settings of the selected Cross-Section and Path. By editing these underlying components, you can customize the shape and dimensions of the Custom Coil.



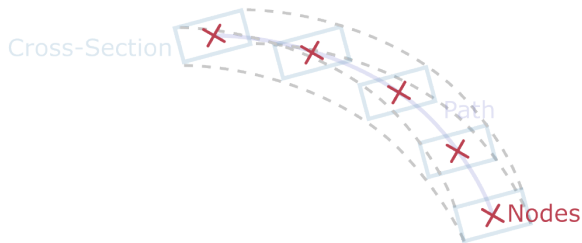
(a) Custom Coil.



(b) Cross-Section.



(c) Path.



(d) Nodes.

Figure 4.2.1: Illustration of a Custom Coil formed by extruding a Cross-Section along a Path. The nodes are positioned on the Path, and each mesh element occupies the volume between successive nodes. Users can adjust the element size using the dedicated element size parameter of both the Cross-Section and Path.

## 4.3 The Edges Coil

The Edges Coil is used when there are no other suitable methods available to model a coil or conductor in the Rat GUI. However, it should be used with caution as debugging Edges Coils can be complicated.

The Edges Coil is constructed with four Paths, one for each edge. Currently, it is only possible to create quadrilateral-shaped coil packs with the Edges Coil or Mesh, hence the requirement for four edges. A Cross-Section cannot be added to the Edges Coil; instead, the software automatically generates the cross-section of the coil pack based on the four provided edges.

When selecting the Edges Coil, it is important to add the four Paths in a specific order. Imagine the cross-section of the coil you are designing as upright and in front of you. The first Path represents the top-left edge, and the other three Paths are added clockwise. This is illustrated in Figure 4.3.1a.

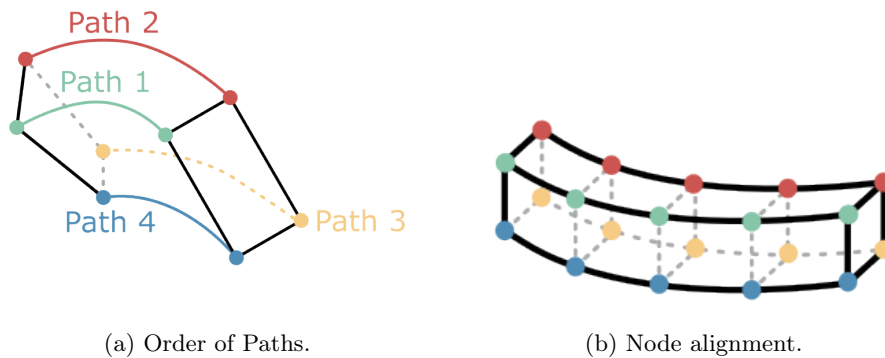


Figure 4.3.1: The order of the Paths of an Edges Coil on the left and the alignment of the of each path on the right.

Another important aspect to consider is that each Path must have the exact same number of nodes, and these nodes must be aligned with respect to each other. Refer to Figure 4.3.1b for an illustration of node alignment. If you are using two Paths for the top two edges with only a difference in radius, for example, you can achieve this by using the Offset Path Group (refer to Section 11.8). This Group allows you to add an offset to the Paths while maintaining the same number of nodes and preserving the alignment with respect to the original position.

An example of an Edges Coil can be downloaded here. In this example, the Edges Coil is modeled using Path Groups, where each Path Groups has an Arc section as child node. For the top two Arcs, a Local Offset Path Group was used to offset the transverse direction with an S-Curve. For each Arc an Offset was used to preserve the node alignment while changing its position.

Each Arc represents only half of a full circle. To achieve the complete round shape, the Edges Coil was placed in a Mirror Group. This configuration maintains the original coil, and with the `anti mirror` setting checked it ensures that the current in the two halves flows from one half to the other.

When performing calculations on the modeled Edges Coil, the magnetic flux density and current density on the surface can be analyzed, as shown in Figure 4.3.3. The magnetic field is highest where the current density is highest, which is influenced by the shape of the coil. The `current` for this coil was set to 500 A, and the `nturns` was set to 50.

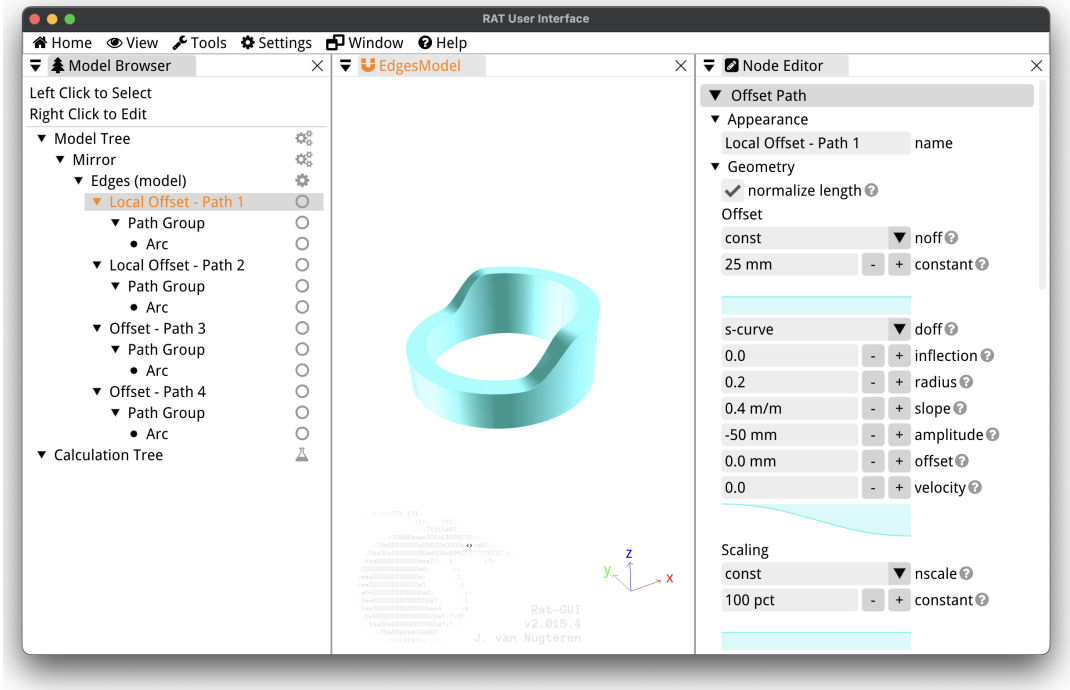
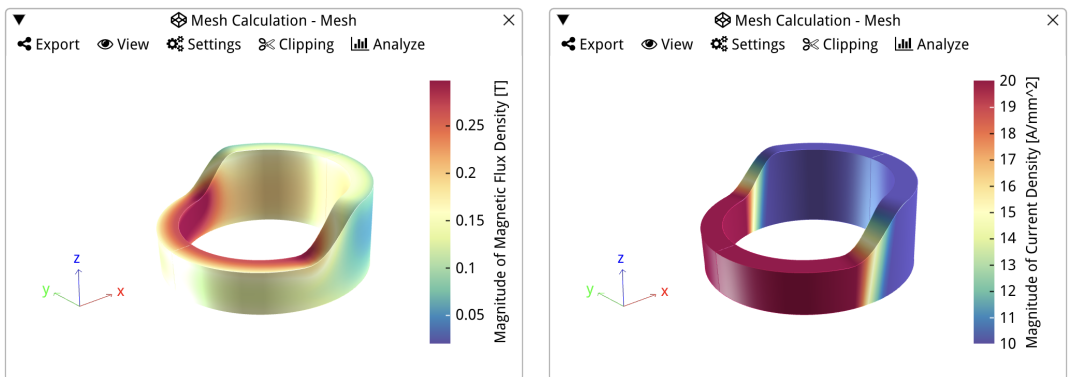


Figure 4.3.2: An example of using the Edges Coil.



(a) The magnetic flux density.

(b) Current density.

Figure 4.3.3: The results of a Mesh Calculation of the Edges Coil in Figure 4.3.2.

The Edges Coil only has general coil settings, which are explained in Section 4.1.1, along with discretization settings. It is not possible to define the **current** in the Edges Coil using current density; only the actual operating current can be specified. An example of the *node editor* of the Edges Coil is shown in Figure 4.3.4.

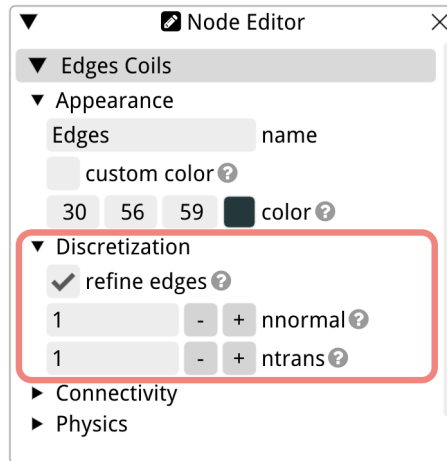


Figure 4.3.4: The *node editor* of the Edges Coil.

The discretization settings include a checkbox for refining the node positions on the edges. This option is typically not used, except in cases where the mesh of the Edges Coil has many non-rectangular elements that require correction. One example is when you use the Edges Coil to defining the ribs of a CCT coil. It is not advisable to use the Edges Coil to model CCTs in the Rat GUI since there are many CCT wrapper models available.<sup>1</sup> In most cases, where the mesh consists mainly of rectangular elements, the **refine edges** option should not be checked.

The other two discretization settings are for determining the number of nodes in the normal direction (**nnormal**) and the number of nodes in the transverse direction (**ntrans**). To inspect the mesh elements and verify whether there are enough nodes in all directions, you can enable mesh visualization by selecting ‘View > Toggle Mesh’ or pressing **Ctrl+M**.

While building the Edges Coil, the mesh will not be drawn in the *viewport* unless all four Paths are complete and free of errors. This means that as long as you have not added all four Paths, the coil will not be visible. Additionally, if the four Paths cannot form a valid shape together due to accidental mistakes, nothing will be shown in the *viewport*. Instead, the Edges Coil will appear red or orange in the *model browser*, indicating that there is a problem with the geometry. It is essential to underline once more that this type of Rat Coil is difficult to debug and should only be used when no other possibility exists to build the Coil in the Rat GUI. If you are going to use it, it is advised to first model the four Paths in four separate Custom Coils (adding a simple Cross-Section to them to visualize the shape of the Paths with respect to each other) to ensure their correctness. You can then copy the Paths of these four Custom Coils to the Edges Coil. However, even with this approach, creating a valid Edges Coil can be challenging due to the complexity of the mesh and geometry. It may require careful adjustments and iterations to ensure a successful result.

<sup>1</sup>CCT wrappers include Section 4.11 and Section 10.10.



## 4.4 The Solenoid Coil

The Solenoid Model Coil is a wrapper model that allows for easy modeling of a solenoid coil in the Rat GUI without the need to build it with a separate Path and Cross-Section. A solenoid is characterized by its long cylindrical form, typically resembling a cylinder or tube. The magnetic field generated by the solenoid is primarily concentrated within the coil due to the uniform winding and the circular ends. As a result, the magnetic field lines follow a path along the central axis of the coil. The field strength is strongest at the center of the coil and gradually diminishes as one moves away from the central axis.

The Solenoid Coil in Rat shares the same general settings as other Coils in the Rat GUI (refer to Section 4.1.1). However, its geometry and discretization settings in the *node editor* are specific to a solenoidal shape. You only need three geometry parameters to create a Solenoid Coil in the Rat GUI: the **radius**, the **thickness**, and the **height**. The **radius** defines the size of the inner radius of the solenoid. The **thickness** represents the measure of the coil pack thickness of the solenoid. The **height** indicates the length of the solenoid's central axis, or its height. These three parameters are illustrated in the Figure 4.4.1 below, featuring an actual solenoid mesh in the Rat GUI.

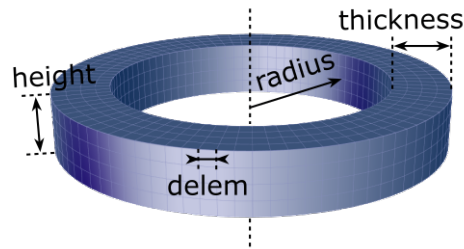


Figure 4.4.1: The definitions of the geometry parameters of the Solenoid Model Coil.

The *node editor* of the Solenoid Coil is shown in Figure 4.4.2. The Solenoid Coil is built with a rectangular Cross-Section and a circular Path. More information on these two Rat objects can be found in Section 9.5 and Section 10.1, respectively. To gain control over the parameters of these internal objects use the Split tool (**Ctrl+D**).

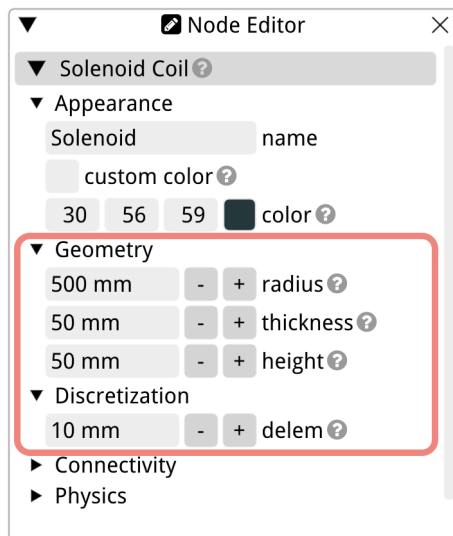


Figure 4.4.2: The *node editor* of the Solenoid Coil.

## 4.5 The Racetrack Model Coil

The Racetrack Model Coil is a wrapper model that allows for easy modeling of a racetrack coil in the Rat GUI without the need to build it with a separate Path and Cross-Section. A racetrack coil is a type of solenoid coil that has a rectangular or square shape with rounded corners, resembling a racetrack or an oval track.

The racetrack coil is used to optimize the magnetic field distribution and improve the magnetic field uniformity along the central axis of the coil. This shape is often preferred in applications where a more uniform magnetic field is required, such as in certain types of particle accelerators, magnetic resonance imaging (MRI) systems, et cetera.

The main advantage of the racetrack coil is that it allows for a more efficient use of the available space, as it can be wound more compactly compared to a traditional cylindrical solenoid. This enables higher magnetic field strengths and better control over the magnetic field profile. The racetrack shape is achieved by combining a rectangular or square cross-section with a circular path for the coil windings. This combination helps to reduce edge effects and magnetic field variations that may occur in purely rectangular or square coils.

The Racetrack Coil in Rat shares the same general settings as other Coils in the Rat GUI (refer to Section 4.1.1). However, its geometry and discretization settings in the *node editor* are specific to a racetrack shape. In the Rat GUI, a Racetrack Coil can be created using four essential geometry parameters: the **radius**, **length**, **thickness**, and **height**. The **radius** determines the size of the inner radius of the curved ends in the racetrack shape. The **length** represents the measurement from the inside of one curved end to the other side, along the long axis of the racetrack. This value equals the length of the straight section plus two times the radius of the curved ends. The **thickness** corresponds to the coil pack size in the radial direction, while the **height** defines the coil pack size in the axial direction. The figure below provides an illustration of these dimensions.

The *node editor* of the Racetrack Coil is shown in Figure 4.5.2. The Racetrack Coil is built with a rectangular Cross-Section and a rectangular Path. More information on these

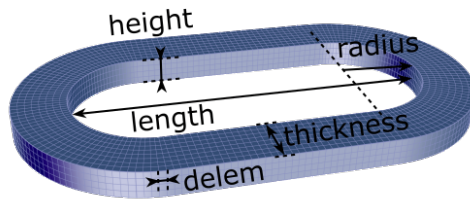


Figure 4.5.1: The definitions of the geometry parameters of the Racetrack Model Coil.

two Rat objects can be found in Section 9.5 and Section 10.3, respectively. To control over the parameters of these internal objects use the Split tool (**Ctrl+D**) to break this Coil up into its constituents.

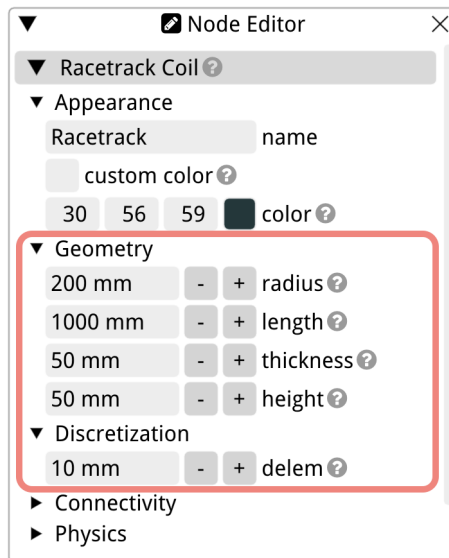


Figure 4.5.2: The *node editor* of the Racetrack Coil.

## 4.6 The Rectangle Model Coil

The Rectangle Model Coil is a wrapper model that allows for easy modeling of a rectangularly shaped coil in the Rat GUI without the need to build it with a separate Path and Cross-Section. The Rectangle Coil is defined by five geometry parameters: the **arc radius**, the **length**, the **width**, the **thickness**, and its **height**. These are all illustrated in Figure 4.6.1. The **arc radius** is the inner radius of the curved corners of the rectangle shape. The **length** is the inner size of the long side, while the **width** is the inner size of the short side. The **thickness** and **height** represent the measures of the two sides of the coil pack.

The *node editor* of the Rectangle Coil is shown in Figure 4.6.2. The Rat Racetrack Coil is built with a rectangular Cross-Section and a rectangular Path. More information on these two Rat objects can be found in Section 9.5 and Section 10.3, respectively. For control over the parameters of these internal objects you can use the Split tool (**Ctrl+D**).

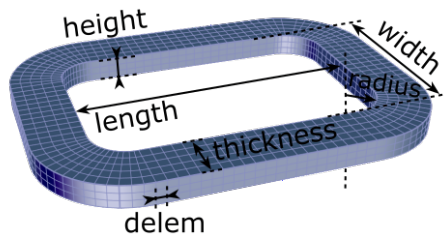


Figure 4.6.1: The definitions of the geometry parameters of the Rectangle Model Coil.

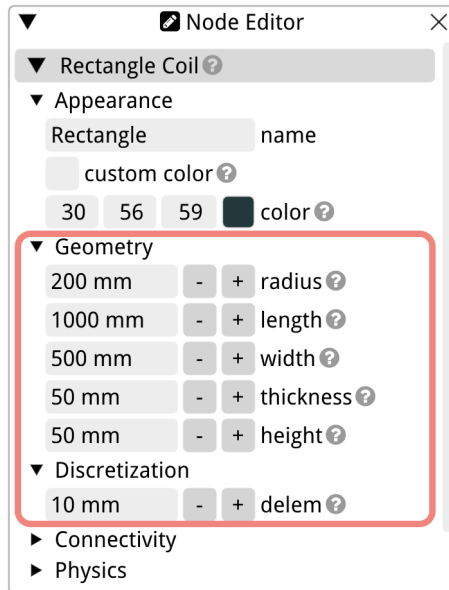


Figure 4.6.2: The *node editor* of the Rectangle Coil.

## 4.7 The Trapezoid Model Coil

The Trapezoid Model Coil is a wrapper model that allows for easy modeling of a trapezoidally shaped coil in the Rat GUI without the need to build it with a separate Path and Cross-Section. It is defined with six geometry parameters: the inner radius of the corners called **arc radius**, the length of **base 1**, the length of **base 2**, the measure of its **altitude**, and the **thickness** and **height** of the coil pack. All these parameters are illustrated below in Figure 4.7.1.

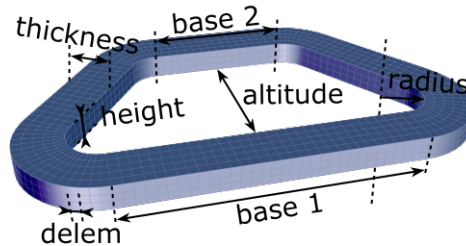


Figure 4.7.1: The definitions of the geometry parameters of the Trapezoid Model Coil.

The *node editor* of the Trapezoid Coil is shown in Figure 4.7.2. The Trapezoid Coil is built with a rectangular Cross-Section and a trapezoidal Path. More information on these two Rat objects can be found in Section 9.5 and Section 4.7, respectively. You can use the Split tool (Ctrl+B) to gain control over its constituents.

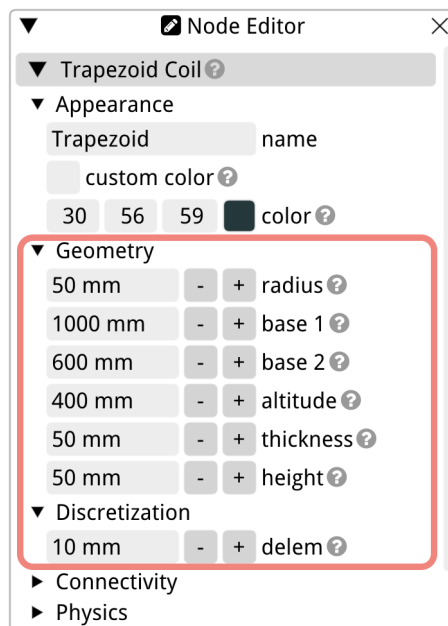


Figure 4.7.2: The *node editor* of the Trapezoid Coil.

## 4.8 The D-Shape Model Coil

The D-Shape Model Coil is a wrapper model that allows for easy modeling of a D-shaped coil in the Rat GUI without the need to build it with a separate Path or Cross-Section. Its Path is a Path Group consisting of a straight section and four Bézier splines. This is explained in more detail in Section 10.5. The Cross-Section used has a rectangular shape.

The D-Shape Coil in the Rat GUI can be defined with four parameters: the **length** of the D (its long side), the **width** of the D (its short side), the coil pack **thickness**, and the coil pack **height**. All these parameters are illustrated in Figure 4.8.1.

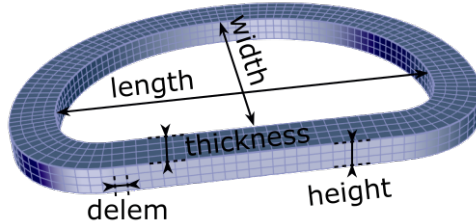


Figure 4.8.1: The definitions of the geometry parameters of the D-Shape Model Coil.

The *node editor* of the D-Shape Coil is shown in Figure 4.8.2. The D-Shape Coil is built with a rectangular Cross-Section (refer to Section 9.5) and a D-Shape Path. A D-shaped object can be made in many ways; thus, the D-shape in Rat is just one example of the many ways in which a D-shape can be drawn in three dimensions. If you want to define your D-shape differently, you could, for instance, use a Path Group, see Section 11.1, and use various arcs and straight sections to draw a D. Alternatively, you can import a Path from an  $x, y, z$ -file with coordinates of your specific D-shape. There is more information on importing Paths from files in Section 10.12, or you can specify your Path in a table. For more information on that topic, refer to Section 10.13.

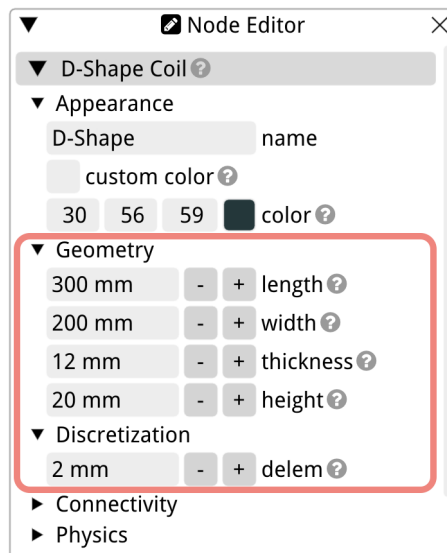


Figure 4.8.2: The *node editor* of the D-Shape Coil.

## 4.9 The Flared-End Racetrack Model Coil

The Flared-End Racetrack Model Coil is a wrapper model that allows for easy modeling of a flared-end racetrack in the Rat GUI without the need to build it with a separate Path and Cross-Section. The Path embedded in this Coil is a Flared Path, which in turn is a Path Group of arcs and straight sections. More information on the Flared Path and the Path Group is found in Section 10.7 and Section 11.1 respectively. The Cross-Section used is rectangular, see Section 9.5.

You need to specify seven geometry parameters to build your symmetric flared-end racetrack in the Rat GUI. These parameters are `length1`, `length2`, `radius1`, `radius2`, `alpha`, the coil pack `thickness`, and the coil pack `height`. All these parameters are illustrated in Figure 4.9.1 below.

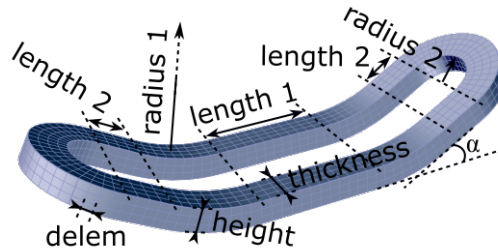


Figure 4.9.1: The definitions of the geometry parameters of the Flared-End Racetrack Model Coil.

The straight section of the racetrack is called `length1`, and the straight section of the flared ends is `length2`. The radius of curvature of the start of the flared ends is called `radius1`, while the radius of the racetrack's ends is `radius2`. The angle between the two straight sections is `alpha`.

The *node editor* of the Flared-End Racetrack Coil is shown in Figure 4.7.2. You can use the Split tool (`Ctrl+B`) to gain control over the Rectangle Cross-Section and the Flared-End Path.

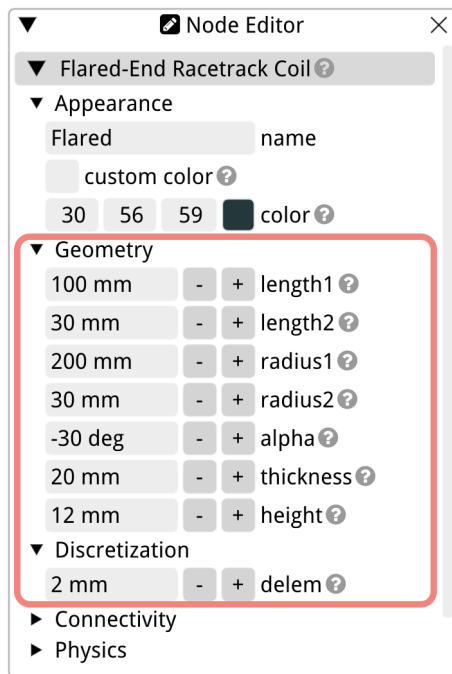


Figure 4.9.2: The *node editor* of the Flared-End Racetrack Coil.

## 4.10 The Cloverleaf Model Coil

The Cloverleaf Coil is a dipole magnet designed for High-Temperature Superconductor tapes (HTS). The coil is wound in a specific manner that prevents damage caused by ‘hardway bend’ tapes. This coil design leverages a Bézier space curve for the winding path, with the Frenet-Serret equations used to compute the orientation of the frame (or the cross-section of the coil pack) along this path.

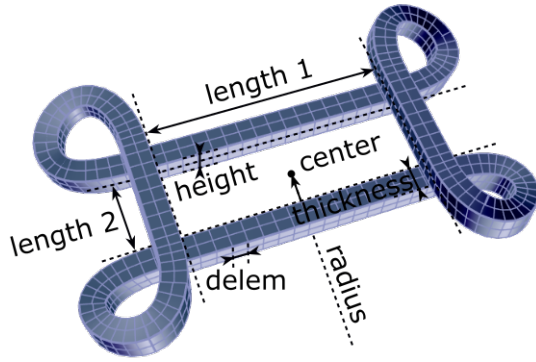
In the Rat GUI, the Cloverleaf Model Coil acts as a wrapper model, enabling straightforward modeling of the cloverleaf coil without requiring the user to construct it with a distinct Path or Cross-Section. Internally, Rat employs a Rectangle Cross-Section (refer to Section 9.5) and a Cloverleaf Path (see Section 10.8) to shape the coil.

To customize the geometry of the cloverleaf coil, the user can adjust nine parameters, all of which are outlined and illustrated in this section and Figure 4.10.1. The parameters **length1** and **length2** denote the lengths of the two straight sections of the cloverleaf. The **bridge height** is measured from the bottom of the lower straight section to the bottom of the upper straight section. This **bridge height** must exceed the coil pack **height** to prevent the bridge from making contact with the lower straight section. Furthermore, the user can define the **thickness** of the coil pack.

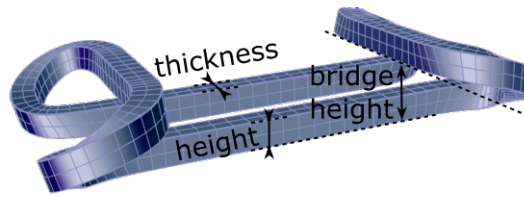
The shape of the cloverleaf coil is also influenced by another parameter, the bridge curvature angle, represented as  $\alpha$  and depicted in Figure 4.10.2. A zero-degree angle implies that the bridge is perpendicular to the straight section beneath it.

In scenarios where a bend magnet design is desirable, such as for fitting it around a beam pipe with a ninety-degree turn, the **radius** setting is used. This **radius** is measured from the center of the magnet, positioned in the middle of both straight sections as shown in Figure 4.10.1a. To keep the magnet straight leave the radius at 0.0 m, which is the default.





(a) Top view.



(b) Side view.

Figure 4.10.1: The definitions of the geometry parameters of the Cloverleaf Model Coil.

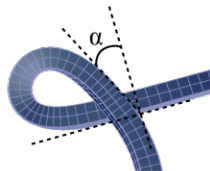


Figure 4.10.2: The bridge angle of the Cloverleaf Model Coil.

The `str12` and `str34` settings, which are not depicted in Figure 4.10.1, affect the shape of the cloverleaf’s leaves. They determine the placement of four of the eight control points used to draw the Bézier space curve. In essence, an increased `str12` value broadens the leaves, while an increased `str34` extends their length. For more detailed information on defining the Cloverleaf Path and the influence of these parameters on the shape of the leaves, please refer to Section 10.8.

Finally, the user can choose to change the winding pack orientation in the cloverleaf’s leaves with the `planar winding` setting at the top of the *node editor*. When this is disabled, a Frenet-Serret frame is assumed for the orientation. However, when enabled, an average between the planar and the Frenet-Serret frames is utilized for the winding pack orientation. All settings discussed here are shown in Figure 4.10.3.

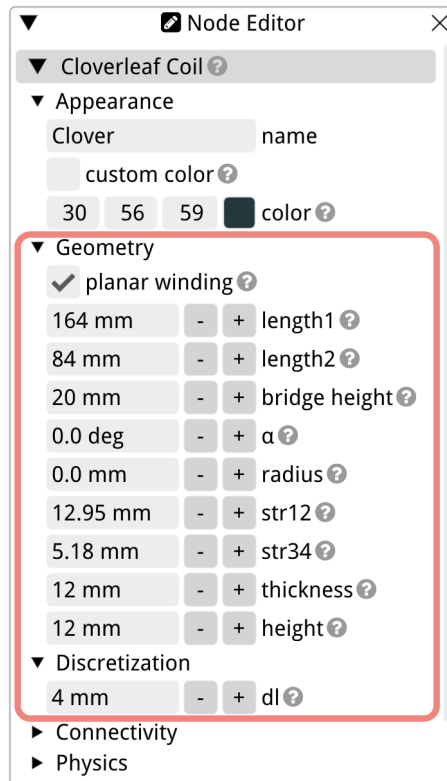


Figure 4.10.3: The *node editor* of the Cloverleaf Coil.

## 4.11 The Canted Cosine Theta Model Coil

A canted cosine theta (CCT) coil is a type of magnet primarily employed in high-energy particle physics applications. The design’s name originates from its characteristic properties: the individual turns of the coil windings are “canted,” or tilted, with respect to the magnet’s central axis, and the coil’s current distribution depends on the azimuthal angle in a cosine-theta manner. This canted winding configuration leads to the magnetic field lines aligning more parallelly to the beam path, providing a more uniform magnetic field. The cosine theta current distribution results in the current density being highest at the top and

bottom of the coil—where it is closest to the magnetic field lines—and lowest on the sides. This pattern of current distribution aids in the creation of a more uniform magnetic field, which is crucial for controlling the path of charged particles in applications such as particle accelerators.

CCT coils offer numerous advantages compared to more conventional magnet designs. The canted design necessitates less conductor material, leading to potential cost savings. Furthermore, the geometry of CCT coils can be adapted to accommodate specific spatial constraints, rendering them a flexible solution for a variety of applications. Additionally, different magnetic field harmonics can be easily combined along the coil’s length.

The Canted Cosine Theta Model Coil serves as a straightforward wrapper model within the Rat GUI, facilitating the easy modeling of a CCT magnet. It is a relatively uncomplicated wrapper: you can only set one field harmonic for the entire coil due to its constituents, which are a Regular CCT Path (refer to Section 10.10.1) and a Rectangle Cross-Section (for more information, see Section 9.5). For more flexibility, you can create a Custom Coil (Section 4.2) and opt for the Custom CCT Path (see Section 10.10.2). This allows you to set multiple harmonics, whose strength can also be varied along the CCT’s central axis.

The CCT Path can be defined by the following equations in the Cartesian coordinate system:

$$x = R \sin(\theta), \tag{4.11.1}$$

$$y = R \cos(\theta), \tag{4.11.2}$$

$$z = A \sin(n\theta) + \frac{\theta\omega}{2\pi}. \tag{4.11.3}$$

Here,  $\theta$  is the running variable, ranging from  $-n_{\text{turn}} \cdot \pi$  to  $n_{\text{turn}} \cdot \pi$ ,  $R$  is the radius,  $n$  is the number of poles,  $A$  is the amplitude, and  $\omega$  represents the pitch. In this Model Coil, the pitch is calculated from the rib thickness,  $d_{\text{rib}}$ , on the mid-plane using:

$$\omega = \frac{d_{\text{cable}} + d_{\text{rib}}}{\sin \alpha}, \tag{4.11.4}$$

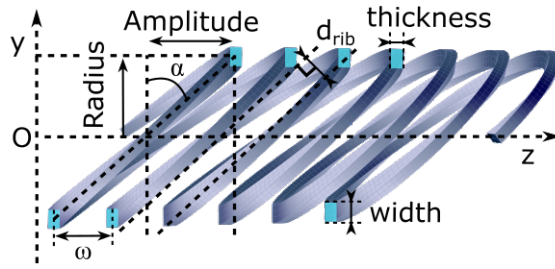
where  $\alpha$  is the skew angle of the coil. The  $d_{\text{rib}}$  parameter can also be defined as the offset between windings,  $d_{\text{turn}}$ , and the cable height, represented as `height` in the node editor, such that  $d_{\text{rib}} = d_{\text{turn}} - \text{height}$ . In the *node editor*, the user can set either the skew angle or the amplitude to define their CCT; they are related to each other through the following equation:

$$A = \frac{R}{n \tan \alpha}. \tag{4.11.5}$$

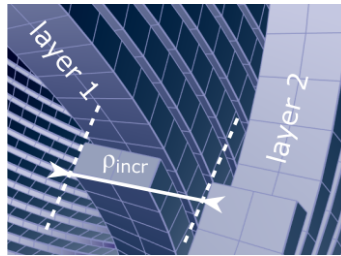
The amplitude and angle relationship applies to the innermost layer. The CCT wrapper can be extended to incorporate multiple layers. In such a case, the Path consists of multiple disconnected sections, but this does not impact the field calculations as the current is assumed to transition from one layer to the next.

All the parameters discussed in this section so far are also illustrated in Figure 4.11.1. However, there are additional parameters that the user can modify in the *node editor*, which are explained following the figure.

The first geometry parameter in the *node editor* of the Canted Cosine Theta Model Coil is the `is skew` boolean. Tick this box to generate field in the skew direction,  $A_n$ , instead of the main,  $B_n$ , direction. It should be noted that this is not equivalent to rotating the coil, as the start and end of the windings remain unchanged with this setting. The harmonics are depicted in Figure 4.11.2.

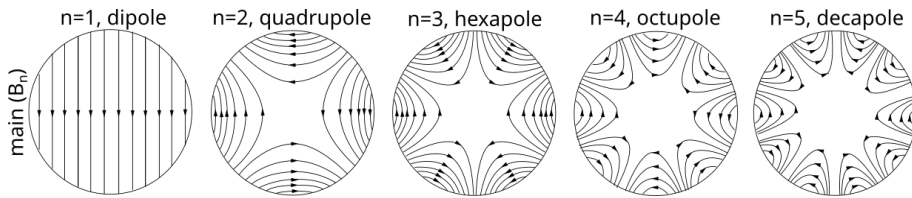


(a) Side view.

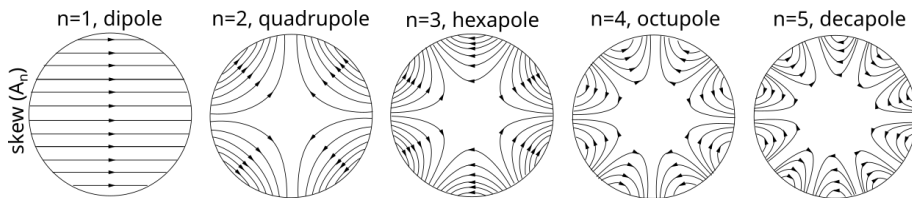


(b) Cross-sectional view.

Figure 4.11.1: The definitions of the geometry parameters of the Canted Cosine Theta Model Coil.



(a) Main harmonics.



(b) Skew harmonics.

Figure 4.11.2: The main harmonics on top and the skew harmonics illustrated at the bottom.

The next parameter in the list is **npoles**, which signifies the number of poles. This sets the magnetic field harmonic generated by the CCT. In Figure 4.11.2, the number of poles for each harmonic is labeled with  $n$ , hence if you assign **npoles** the value 2, a quadrupole is generated for instance.

Following the **radius** and **drib** settings, previously explained in this section, you will discover the **reverse** checkbox. This inverts the canting direction for even layers. On the same line, the **use angle** boolean can be adjusted, which determines if the user sets the **amplitude** of the CCT or the skew angle  $\alpha$ . Altering this setting makes a slight change in the *node editor*, either the **amplitude** or **alpha** setting are presented.

The number of CCT slots in a layer is established with **nt1** and **nt2**, where **nt1** is the number of the initial turn and **nt2** the number of the final turn. The turn number 0 is centered at  $z = 0$ , thus if you set **nt1** to  $-2$  and **nt2** to 7 the coil possesses 10 turns and the third turn is centered at the origin of the global coordinate system.

Next, you can set the **twist**. This parameter twists the coil pack, or CCT slot, with the angle that you assign, at the same frequency as the poles. Maintaining this **twist** angle at  $0^\circ$  keeps the coil pack perpendicular to the former at the peaks of the poles, otherwise the angle that you set is used at the peaks of the poles. Instead of using this twist you can design your CCT based on a Frenet-Serret frame. In that case you will need to model a Custom Coil. You can achieve the Frenet-Serret CCT by either using the Custom CCT Path and checking the **frener-serret** box, or by setting a Regular CCT Path and placing it in a Frenet-Serret Path Group. Refer to Section 10.10.1 and Section 10.10.2 for the CCT Paths and Section 11.6 for the Frenet-Serret Path Group.

The cross-section of the CCT's coil pack is described by **width** and **thickness**. Contrary to the other wrappers, the **width** is defined in the radial direction and the **thickness** in the axial direction, as this is more intuitive for this coil from a coil winding perspective. The **width** becomes the depth of the slot and the **thickness** the width of the slot.

Next in the *node editor* are the settings for CCT layers. For the outer layers, either the amplitude,  $A$ , or the skew angle,  $\alpha$ , can be kept constant. By using the **same**  $\alpha$  setting, you can make this choice. Starting from the second layer, each successive layer's inner radius naturally increases by  $\rho_{\text{incr}}$ . This implies that if you decide to keep the skew angle constant across all layers, the outer layers will stick out slightly at the coil ends. Conversely, if you choose a constant amplitude across all layers, foregoing the skew angle constancy, the outer layers won't extend beyond the innermost layer. However, in this scenario, the skew angle will be altered for the outer layers. One notable advantage of maintaining a constant amplitude is that it optimizes the current density at the coil ends. The effect of using **same angle** can be seen easily with a larger  $\rho_{\text{incr}}$ , as is demonstrated in Figure 4.11.3.

The number of layers of your CCT design can be set with **nlayers**. The distance from the inner radius of an inner layer to the inner radius of the next layer surrounding it is set with  $\rho_{\text{incr}}$ . This parameter is also depicted in Figure 4.11.1b.

This wrapper is far more detailed than the other wrappers in the Rat GUI, and consequently, you will also find parameters to transition from a straight CCT to a curved magnet under 'Bending,' and settings to add current leads on both side of the CCT under 'Current Leads.' This is explained in the upcoming paragraphs. The discretization settings of the Regular CCT Model Coil can be utilized to control the number of nodes along the CCT's path with the number of nodes per turn, **nnpt**. This consequently defines the number of elements in the longitudinal direction. With **delem**, you can set the target element size for the rectangular elements of the cross-section of the CCT.

The bending settings can be used to curve your magnet around the  $y$  axis, in the  $xz$ -plane. This can be achieved either by setting a bending radius, **rbend**, or a bending arc length in

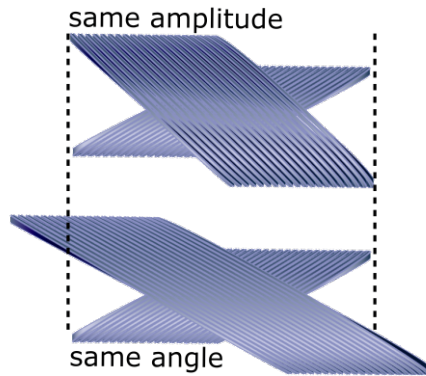


Figure 4.11.3: The effect of using `same angle` for the Canted Cosine Theta Model Coil.

degrees, `arcbend`. To select whether you wish to set radius or arc length, you can use the `use radius` boolean setting on the first line of the bending settings. The final bending option can be used to change the position of the coil with respect to the origin of the global coordinates. By default, the origin is in the middle of the coil, but if you select the `bend origin` box, you can alter this to the center of the bending circle. These bending parameters are also illustrated in Figure 4.11.4.

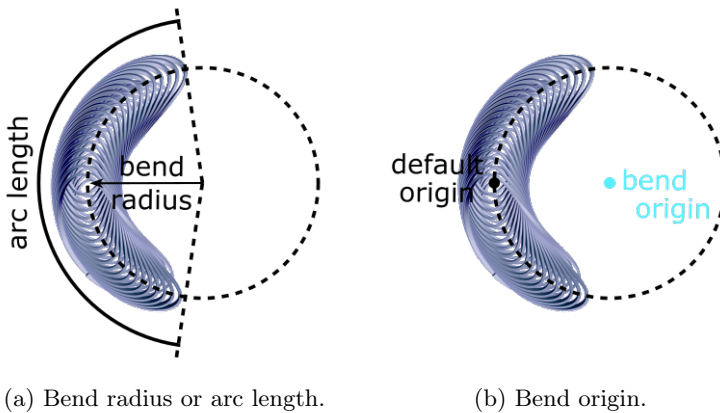
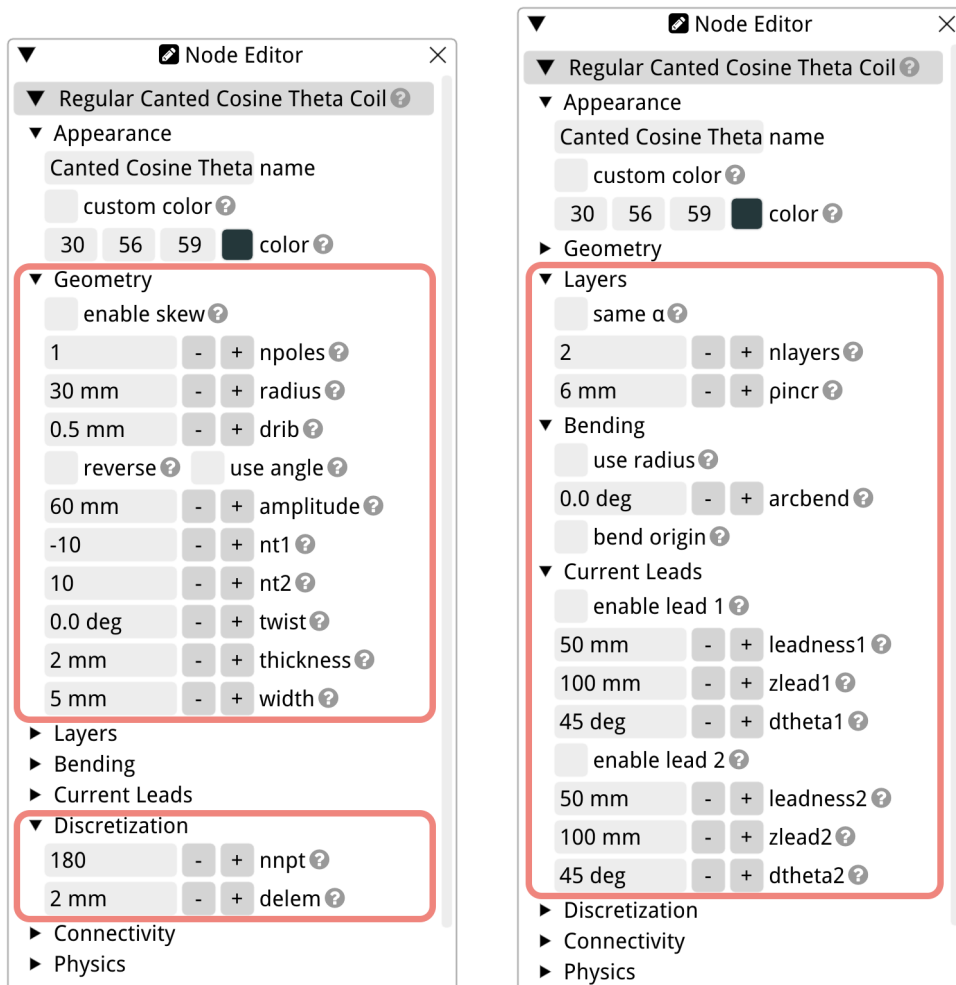


Figure 4.11.4: The definitions of the bending parameters of the Canted Cosine Theta Model Coil.

The final settings are used to model current leads on both ends of the CCT. They have to be enabled with the `enable` checkboxes, there is one checkbox per lead to allow for modeling current leads on one side only. The user can then set the `leadness`, which is the curvature strength for the lead, the `zlead`, which is the start of the lead position with respect to magnetic length, and the azimuthal position of the lead with respect to the coil end, or `dtheta`. The current leads are made with Bézier splines, and these settings impact the control points of the splines.

In general, current leads for CCTs are challenging to model because the orientation of the cross-section of the CCT twists and turns along the CCT path, and hence if you wish to create a current lead at any arbitrary point along the path you would require different splines each time. The current leads in this Regular CCT wrapper are just one solution, and

thus they are not infinitely flexible. All settings discussed in this section are shown in the *node editor* of the Regular CCT Coil in Figure 4.11.1.



(a) Geometry and Discretization.

(b) Layers, Bending and Current Leads.

Figure 4.11.5: The *node editor* of the Regular Canted Cosine Theta Coil.

## 4.12 The Plasma Ring Coil

The Plasma Ring Coil is a wrapper model that allows for easy modeling of a plasma ring in the Rat GUI without the need to build it with a separate Path or Cross-Section. This Model Coil is mainly meant for visualization purposes, to show a volume in your plasma device where the plasma would be. However, it may be used to estimate the magnetic field resulting from a plasma in a fusion device, due to the plasma current. In that case, it should not be viewed as an electromagnetic coil, but a plasma volume with a current running through the ring. It will not resemble a real plasma current in any way since the current density is assumed to be homogeneous in the Plasma Ring Model Coil volume. But it will result in

an order of magnitude estimation of the field. In that case, you need to keep the number of turns, `nturns`, equal to 1. All geometry settings in the *node editor* of the Plasma Ring Coil are shown in Figure 4.12.1.

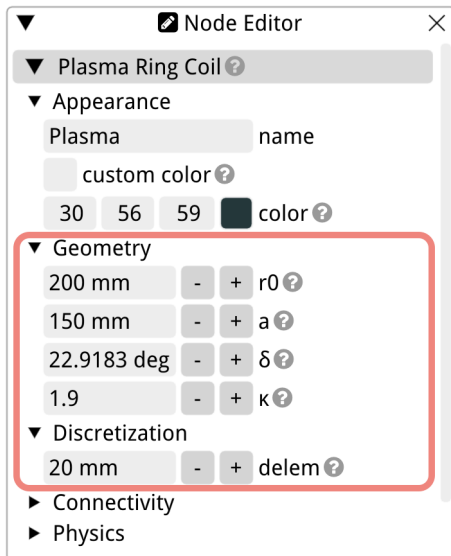


Figure 4.12.1: The *node editor* of the Plasma Ring Coil.

In the Rat a three-dimensional shape, a Coil, or a Mesh, is usually made by extruding a two-dimensional cross-section along a path that is defined in three dimensions. Most Cross-Sections in Rat are made with a simple rectangular mesh; however, the two-dimensional shape of the plasma ring is too complex, and therefore Rat uses a meshing algorithm to mesh this plane before extruding it along a circle with a radius that is equal to the major radius of the poloidal magnetic flux. The shape of the cross-section of the plasma can be calculated with the poloidal magnetic flux function in two dimensions,  $x$  and  $y$ :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \cdot \cos(\theta + \delta \cdot \sin \theta) \\ \kappa \cdot a \cdot \sin \theta \end{bmatrix}. \quad (4.12.1)$$

In Equation (4.12.1),  $x$  and  $y$  represent the Cartesian coordinates in the poloidal plane, which is the plane perpendicular to the toroidal direction. The parameter  $a$  is the minor radius, which represents the distance from the center of the plasma to the magnetic axis, the axis around which the plasma torus is symmetric. The angle  $\theta$  is the poloidal angle, which represents the angular coordinate in the poloidal plane. The  $\delta$  parameter is the triangularity, which determines the shape of the plasma cross-section by modifying the poloidal angle. It moves the foci of the ellipse such that the poloidal magnetic flux surface becomes more or less triangular. Finally,  $\kappa$  is the elongation, which affects the elongation or stretching of the plasma cross-section along the poloidal direction.

All parameters needed to describe the two-dimensional poloidal magnetic flux plane are illustrated in Figure 4.12.2. The discretization settings of the Plasma Ring wrapper are not shown in Figure 4.12.2. But as for all Model Coils in Rat, `delem` is used as the target mesh element size in each direction.

In the Rat GUI, you can set  $a$ ,  $\delta$ , and  $\kappa$ , as well as the major radius  $r_0$ . The major radius is a parameter used to describe the size of a toroidal magnetic confinement device, such as a



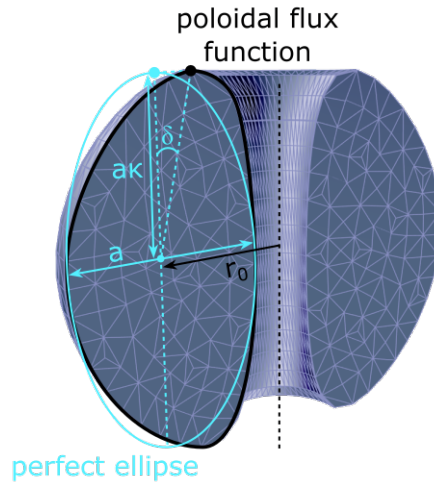


Figure 4.12.2: The definitions of the geometry parameters of the Plasma Ring Coil. The black circumference shows the poloidal magnetic flux function. The cyan circumference a perfect ellipse. The  $\delta$  parameter moves the foci of the ellipse to get a more triangular shape, while the  $\kappa$  parameter determines the elongation.

tokamak, and it refers to the distance from the center of the magnetic confinement device to the center of the torus, measured along the axis of symmetry, as was described before. You can use it to define the aspect ratio,  $A$ , as follows:

$$A = \frac{r_0}{a}. \quad (4.12.2)$$

The aspect ratio provides information about the relative size of the torus (the overall shape of the plasma) compared to the size of the plasma in the poloidal plane (the shape of the plasma cross-section). A high aspect ratio means that the torus is relatively large compared to the cross-sectional size of the plasma, while a low aspect ratio indicates a more compact configuration. In fusion research, aspect ratio is an important parameter that influences the stability and performance of the plasma confinement and is often used to classify different types of tokamaks and other magnetic confinement devices.

# Chapter 5

## Meshes

### 5.1 Introduction

A Mesh in Rat is a three-dimensional object or mesh, and in contrast to the Coil (Chapter 4) it can not be a conductor. It has the same general settings as a Coil (refer to Section 4.1.1), but in the physics settings you can only set the temperature. The available Mesh types and their corresponding sections in this manual are summarized in Table 5.1.1.

Table 5.1.1: Mesh objects in the Rat GUI.

Name	Description	Section
Custom	Defined by Path and Cross-Section	Section 5.2
Cube	A cube	Section 5.3
Sphere	A sphere	Section 5.4
Cylinder	A closed cylinder	Section 5.5
Doughnut	A doughnut-shaped mesh	Section 5.6
Point	A point in space	Section 5.7
Line	A line with the cross-section of a point	Section 5.8
Axes	Three lines, one in each dimension, with a shared origin	Section 5.9
Stick Figure	A stick figure of 1.84 m for showing model scale	Section 4.10
Import	Import an STL file with a mesh	Section 5.11

A Mesh can be used to model structures that are surrounding you Coils, such as mechanical supports or radiation shields. When performing a Mesh Calculation, the magnetic field on the surface of your Mesh, arising from Coils in the same Model are given in the results. See Section 15.6 for more information on the Mesh Calculation. Some Meshes are only there for visualization purposes, such as the Stick Figure and the Axes.

- 5.2 Custom
- 5.3 Cube
- 5.4 Sphere
- 5.5 Cylinder
- 5.6 Doughnut
- 5.7 Point
- 5.8 Line
- 5.9 Axes
- 5.10 Stick Figure
- 5.11 Import

# Chapter 6

## HB-Meshes

HB-Meshes play a pivotal role in modeling objects involving materials with non-linear magnetic characteristics. The hallmark of an HB-Mesh, distinguishing it from standard meshes, is the incorporation of the HB-curve. This curve is essential for achieving accurate magnetic field calculations.

While there are predefined HB-Meshes available, such as the sphere, the versatility of this module is truly showcased through the Custom HB-Mesh. It allows for the creation of virtually any shape by combining the desired Cross-Section and Path. HB-Meshes are particularly suited for modeling intricate structures like iron yokes, capturing detailed features such as holes, keys, and curves. Due to the often complex nature of such objects, the Distmesh Cross-Section offers invaluable tools and techniques for design. Consult Chapter 8 on a detailed explanation of Distmesh Cross-Sections.

This chapter provides an in-depth exploration of the variety of HB-Meshes available and offers guidance on their applications. These are summarized in Table 6.0.1.

Table 6.0.1: Mesh objects in the Rat GUI.

Name	Description	Section
Custom	Defined by Path and Cross-Section	Section 6.2
Sphere	A sphere	Section 6.3
JSON	Import a mesh from a JavaScript Object Notation file that was created in FreeCAD software [5]	Section 6.4

For objects like Coils, Meshes, and Permanent Magnets, Rat employs a straightforward linear integration procedure to compute the electromagnetic properties of the components within your models, i.e.,  $B \propto I$ . This allows the application of the superposition principle to determine the combined field arising from multiple coils in the same space. However, this straightforward approach isn't applicable to non-linear materials. When modeling such materials in Rat, one uses HB-Meshes. This deviation is primarily due to the nonlinear HB relationship inherent in materials like iron. The general settings of Rat's HB-Meshes, explained in the next section, reflect this.

It's crucial to note that in Rat, users are unable to append HB-curves to objects other than the HB-Meshes. The process for modeling non-linear materials in Rat aligns with that for other objects, with the sole exception being the inclusion of the HB-curve. To introduce an HB-Mesh, right-click on the Model Tree within the *model browser* and choose

‘Add HB-Mesh.’ A menu will emerge, allowing users to opt for any of the accessible objects, as outlined in Table 6.0.1. Once an object is selected, it will manifest in the Model Tree. When selected, its attributes can be adjusted via the *node editor*. By default, all HB-Meshes come with an associated child node HB-curve. If users wish to modify this curve, they can simply right-click on the HB-Mesh in the Model Tree (not the curve) and select ‘Add HB-curve.’ Comprehensive details about HB-curves can be found in Section 6.5. Once you have completed modeling your HB-Mesh, you can directly proceed with calculations as detailed in Chapter 15. The software is designed to automatically recognize the presence of non-linear materials and will initiate the appropriate non-linear solver in response.

## 6.1 General settings

Before explaining the general settings of the non-linear solver that is used to calculate the field of the HB-meshes, it is important to understand how the calculation procedure works. The non-linear solver is multi-faceted, focusing on determining the electromagnetic field effects due to coils and permanent magnets, particularly within non-linear domains. Initially, the electromagnetic field is determined due to both the coil and permanent magnets acting on the non-linear parts. From there, the non-linear domain is solved for the vector potential on the edges [10], which is then utilized to derive the magnetization within the elements.

With this foundation, the bound currents are computed based on the magnetization within each element. It’s essential to note that these currents exist both on the surface and within the volume of the mesh. Leveraging the bound currents, both the flux density and vector potential are calculated at all designated target points, encompassing coils, meshes, and more. Alongside this, magnetic charges are also determined, and using these charges, the magnetic field at the target points is subsequently calculated.

A significant step involves the interpolation of magnetization values from the mesh to the target points. One critical observation is the sharp transition of  $\mu_r$  at the iron’s surface, where the majority of bound current flows. This results in a pronounced field transition at this location. The Biot-Savart law is employed to calculate the field resulting from the bound currents through Gaussian quadrature (integration). Given the sharp field transition at the surface, a higher density of Gauss points, denoted as `sngauss`, on the surface is necessary for precise integration. Concluding this procedure, it’s crucial to comprehend that while bound currents contribute to the flux density ( $B$ -field) and vector potential, magnetic charges primarily influence the magnetic field ( $H$ -field), highlighting the importance of the  $B - H$  relationship.

In addition to the number of Gauss points on the surface, `sngauss`, the user can set the number of Gauss points inside the volume with `vngauss`. Finally, just like any other object in Rat and the Rat GUI, you can set assign a temperature to is with the `temperature` parameter in the *node editor*, which is later used to calculate material properties in the Mesh Calculation for instance (refer to Section 15.6). All general settings are shown in an image of the *node editor* in Figure 6.1.1.

In Rat, the default practice is to construct meshes using hexahedron elements. However, certain applications may necessitate the use of tetrahedrons instead. Both the Mesh and the HB-Mesh in Rat offer the option to utilize tetrahedrons, enabled through the general setting `use tetrahedrons`. It’s crucial to understand that if one HB-Mesh is configured with tetrahedrons, all should follow suit. Similarly, if hexahedrons are used for one, they should be used for all. Consistency in the choice of mesh element type across all HB-Meshes is essential.

Each of the HB-Meshes needs an HB-Curve. If it is not automatically included in the

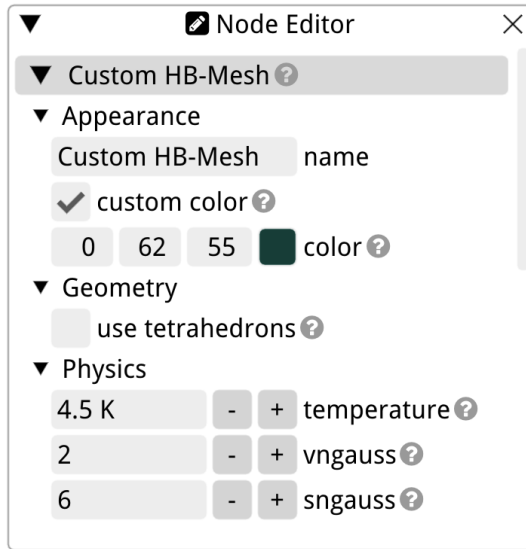


Figure 6.1.1: The *node editor* of the HB-Mesh, with the general setting highlighted.

HB-Mesh the user can easily add it by right-clicking the HB-Mesh in the *model browser* and selecting ‘Add HB-Curve.’ In the menu that appears the user is given three choices. These are explained in Section 6.5.

## 6.2 Custom HB-Mesh

The Custom HB-Mesh in Rat offers flexibility similar to other Custom objects in the software. Users can configure it using a Cross-Section of their choice (as detailed in Chapter 9) and a Path (as outlined in Chapter 10). Upon its addition to the *model browser*, this Custom HB-Mesh is highlighted in red in the Model Tree until both the Cross-Section and Path are selected. To set the Cross-Section and Path, right-click the Model Coil listed in the Model Tree and opt for ‘Add Path’ and ‘Add Cross-Section’, respectively.

In the framework of the Rat software, envision the creation of a Custom HB-Mesh as a process where a Cross-Section undergoes virtual extrusion along a given Path. Figure 6.2.1 aptly illustrates this concept.

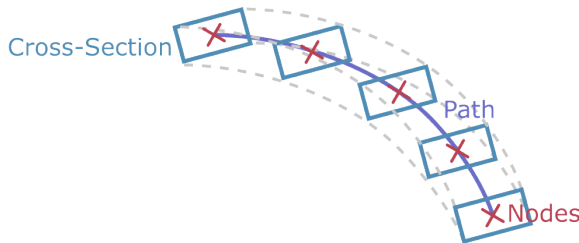


Figure 6.2.1: Demonstration of the extrusion of a Cross-Section along a Path within Rat. More details are shown in Figure 4.2.1.

The Custom HB-Mesh does not possess distinct geometry settings of its own; it solely

adheres to the general HB-Mesh settings as discussed in Section 6.1. The actual customization of its form and dimensions stems from the settings of the appointed Cross-Section and Path.

Lastly, it's worth highlighting the capabilities of the Distmesh Cross-Section. It provides the versatility to craft almost any two-dimensional shape, rendering it highly advantageous for designs like the iron yoke and similar structures.

### 6.3 Sphere HB-Mesh

The Rat software has a specialized HB-Mesh object named the Sphere HB-Mesh. As the object's name suggests, it takes on a spherical form. This is an HB-Mesh wrapper, retaining its fundamental features but complemented by attributes intrinsic to its spherical geometry.

Spherical meshes hold a distinct place in the realm of geometric design due to their inherent characteristics. Unlike many other geometric structures, a sphere cannot be constructed by simply extruding a cross-section along a path; it is a singular entity without separate components for defining its shape. This absence of a discrete Path and cross-section underlines the sphere's simplicity on one hand, and its uniqueness on the other.

In terms of the general HB-Mesh settings (Section 6.1), the Sphere HB-Mesh is aligned with all HB-Meshes. However, where it differentiates is in its specific spherical parameters. Firstly, there's the **radius** parameter. This, as one would surmise, specifies the radius of the sphere, allowing users to scale the sphere to fit their requirements.

The other geometry setting is the **delem**, denoting the target element size. This parameter plays a pivotal role in meshing the sphere, impacting its granularity and detail. As for the HB-curve, by default, it utilizes the Vinh Le-Van fit, a reliable and comprehensive HB-curve that caters to a broad range of applications. All settings of the Sphere HB-Mesh are shown in the *node editor* of Figure 6.3.1.

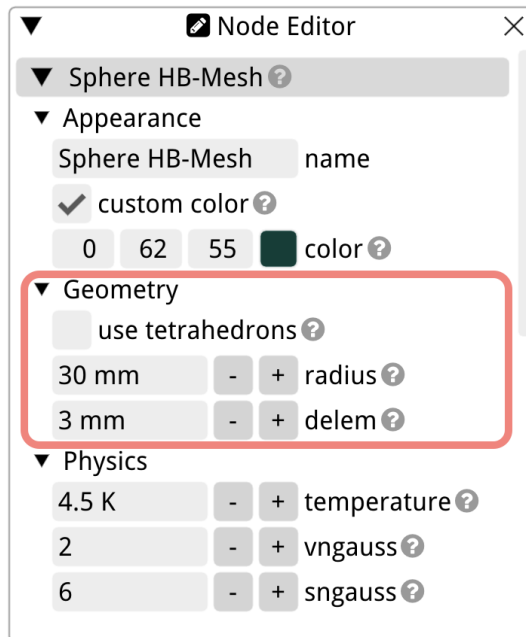


Figure 6.3.1: The node editor interface of the Sphere HB-Mesh in the Rat GUI. It displays both standard HB-Mesh settings and the sphere-specific parameters.

## 6.4 JSON HB-Mesh

The JSON HB-Mesh in Rat provides a pathway for users to seamlessly import mesh structures that have been modeled or converted in FreeCAD [5]. Using FreeCAD, one can either design intricate geometries from scratch or import existing CAD files, and then export these as JSON files for integration into the Rat GUI.

For the purpose of explaining how to create a JSON Mesh in FreeCAD, a simple sphere mesh will be used. Start by opening the software and initiating a new project through the ‘File > New’ menu at the top of the FreeCAD GUI, or by clicking on the **new file** button, refer to Figure 6.4.1a. For modeling your geometry, ensure you are working within the Part Workbench by selecting ‘View > Workbench > Part,’ see Figure 6.4.1b. Once in this environment, you can utilize the various tools available in FreeCAD’s Part Workbench to craft your desired object. In this example a simple sphere was created.

When your design (our sphere) is complete, transition to the FEM Workbench via ‘View > Workbench > FEM,’ shown in Figure 6.4.1b. To create the FEW mesh from your geometry, select the geometry in the Model Tree, it is now highlighted in grey in Figure 6.4.1c. Next, you’ll want to select the ‘FEM Mesh from shape Gmsh’ option to generate a mesh, as shown in Figure 6.4.1c. In the ensuing dialog where the Model Tree is normally shown, adjust the **Max element size** and confirm that the **Element Order** is set to **First Order**. After these selections, click ‘Apply’ followed by ‘OK’ as in Figure 6.4.1d.

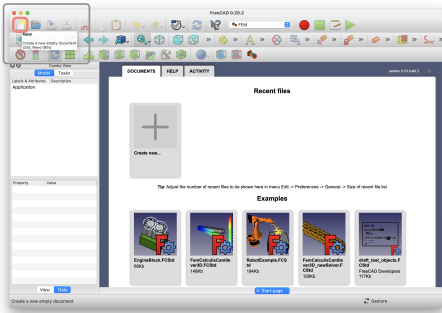
To prepare for the export, you need to hide the original design by selecting it within the Model Tree and pressing the space bar on your keyboard. The mesh is the only object that is still visible as in Figure 6.4.1e. If you wish to make further adjustments to your mesh, FreeCAD offers an array of additional settings under the ‘Data’ section below the Model Tree. If you do opt for changes, always double click the mesh in the Model Tree, apply the alterations, and confirm with OK.

When you’re satisfied with your mesh, select it and proceed to File > Export (Figure 6.4.1f). In the subsequent dialog, choose the **FEM Mesh YAML/JSON (\*.meshyaml \*.meshjson \*.yaml \*.json)** option for file format, which is shown in Figure 6.4.1g. It’s crucial to ensure your file’s extension is set to **.json** before saving, just like in Figure 6.4.1h.

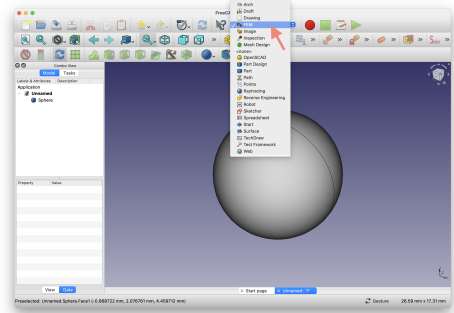
With your mesh now exported from FreeCAD in the JSON format, you can effortlessly introduce it into the Rat GUI. Simply add a JSON HB-Mesh object to the Model Tree of the Rat GUI and load your recently saved .json file.

The *node editor* of the JSON HB-Mesh is shown in Figure 6.4.2. You can select the JSON file exported from FreeCAD with the **file** parameter. There is also a scale parameter that allows the user to quickly change the size of the mesh. Note that FreeCAD produces a mesh with tetrahedron elements. This means that all other HB-Meshes in your Rat Model need to be meshed with tetrahedrons as well. This can be achieved with the use **tetrahedron** checkbox in their respective *node editors*.

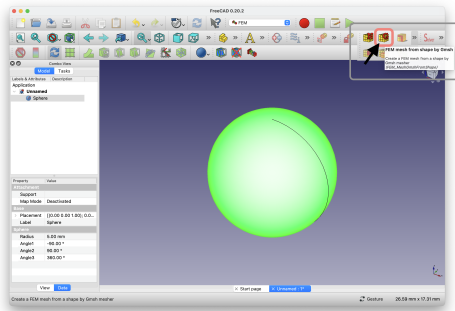




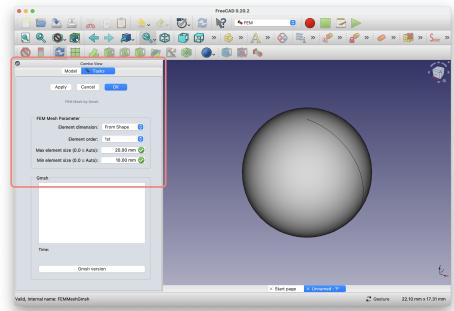
(a) Start from a new file.



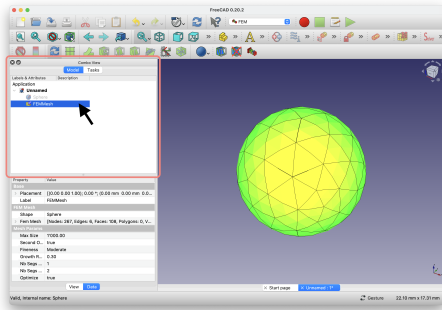
(b) Select a Workbench.



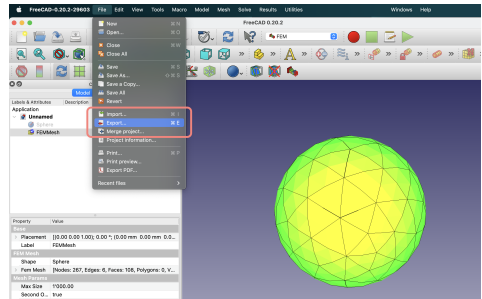
(c) FEM Mesh from shape by Gmsh.



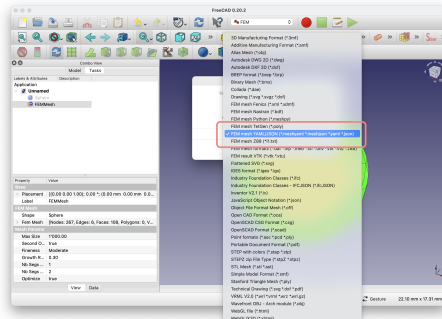
(d) Adjust mesh settings.



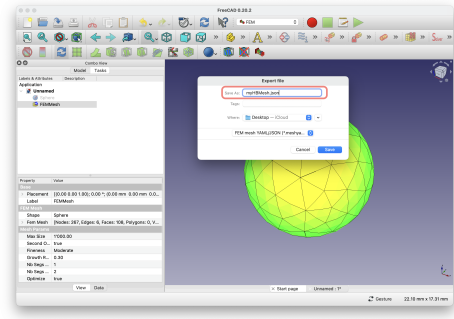
(e) Hide original, select FEMMesh in Model Tree.



(f) Start export process.



(g) Select JSON file type.



(h) Use .json file extension.

Figure 6.4.1: The different steps in the FreeCAD JSON Mesh export.

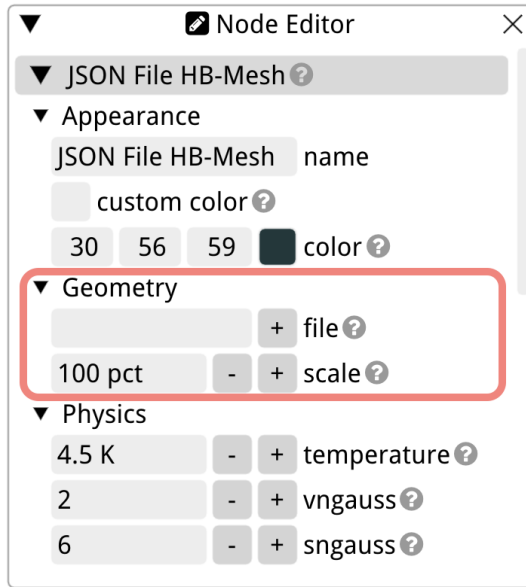


Figure 6.4.2: The *node editor* of the JSON HB-Mesh.

## 6.5 The HB-Curve

The HB-Curve refers to a curve describing the relationship between the magnetic field strength  $\vec{H}$  and the magnetic flux density  $\vec{B}$  for a material. This relationship is essential when dealing with magnetic materials, particularly those that exhibit non-linear behavior like ferromagnetic materials (e.g., iron, nickel, and cobalt).

In the context of non-linear materials, the relationship between  $\vec{H}$  and  $\vec{B}$  is not constant (as it would be in a linear material). Instead, as you increase  $\vec{H}$ ,  $\vec{B}$  also increases, but not necessarily in a linear manner. This non-linear relationship becomes particularly pronounced as the material approaches saturation.

The HB-curve provides a characterization of how the material gets magnetized. Initially, for low  $H$  values,  $\vec{B}$  increases rapidly. As  $H$  continues to increase, the increase in  $\vec{B}$  starts to slow down until the material becomes saturated and  $\vec{B}$  no longer increases substantially.

In Rat, both predefined and user-imported HB-curves can be utilized, all explained in Sections 6.5.1 to 6.5.3 and 13.3. While a set of predefined HB-curves is readily available, users also have the flexibility to input their own magnetic characteristic data. However, it's crucial to approach this with care, as the accuracy of the non-linear calculations hinges significantly on the reliability of these HB-curves.

When importing or defining their HB-curves, users should ensure that the curve exhibits smoothness and continuity. Abrupt jumps or discontinuities can lead to numerical issues in simulations and may not accurately depict the real-world magnetic behavior of materials. Additionally, the derivative of the curve, often represented as  $\mu_r$  or the relative permeability, should be monotonous. A non-monotonous derivative might imply ambiguous and non-physical behavior in the magnetic response of a material to field changes. This is especially pivotal in iterative numerical methods where the curve's tangent is frequently utilized.

Recognizing the regions of high permeability and the onset of saturation in the curve is essential. Materials that saturate will exhibit a drastic drop in permeability, transitioning the

material behavior from a highly magnetic state to a nearly non-magnetic state. The intrinsic non-linear nature of real-world HB-curves demands meticulous representation in simulations, especially when determining the rate of change of magnetic flux density concerning the magnetic field intensity. Users keen on delving deeper into the nuances of HB-curves and their implications are encouraged to refer to the detailed insights provided in the paper “Effective Use of Magnetization Data in the Design of Electric Machines With Overfluxed Regions” [11].

### 6.5.1 The Vinh Le-Van Fit

Vinh Le-Van et al. proposed a nonlinear fit for ferromagnetic materials, as described in [12]. This fit utilizes an arctangent law to describe the relationship between magnetic field intensity ( $\vec{H}$ ) and magnetic flux density ( $\vec{B}$ ), and can be expressed as:

$$\vec{B}(\vec{H}) = \mu_0 \vec{H} + \frac{2J_s}{\pi} \arctan\left(\frac{\pi(\mu_r - 1)\mu_0 \vec{H}}{2J_s}\right) \quad (6.5.1)$$

In the above equation,  $J_s$  denotes the saturation magnetization, typically valued at 2.0 T for iron. The parameter  $\mu_r$  represents the initial relative permeability, with a common value of 500.

In the Rat GUI, users have the flexibility to adjust certain parameters pertaining to this fit. The relative magnetic permeability,  $\mu_r$ , and the saturation magnetization,  $J_s$ , can both be set by the user to ensure compatibility with their specific requirements.

Additionally, the Rat GUI offers settings to influence the software’s utilization of the fit. The software uses the fit and turns it into a table with magnetic field intensity  $\vec{H}$  in one column and the corresponding magnetic flux density  $\vec{B}$  in another. Users can specify the upper bound of the magnetic field,  $H_2$ , to define the maximum value in said table. They can also determine the number of data points or rows within the table using the `npoints` setting.

The last setting in the *node editor* of the Vinh Le-Van fit is the `ffill` setting that allows users to define the fill fraction of the magnetic material in the HB-Mesh under consideration. This can be used in case the material is not homogeneous for instance and only partially of non-linear material.

All parameters are presented in the *node editor* as shown in Figure 6.5.1. When you hover over the HB-curve graph at the bottom of the *node editor*, it enlarges, allowing for a detailed inspection of the curve.

### 6.5.2 The HB-Curve Table

In Rat, there’s an alternative approach to defining the HB-Curve — importing it as a table. Here’s a rundown of the process: The table-based HB-Curve setting starts with the indispensable fill fraction parameter, denoted as `ffill`. This parameter specifies the filling fraction of the magnetic material, especially useful when the object under study isn’t wholly made of a non-linear material.

Next in line is the `data reference` field. This provides information regarding the origin or source of the data—effectively a reference to where the data was procured from or based upon. This is followed by `npoints`, which represents the number of data points within the table. It defines the granularity of the HB data and its consequent representation.

Following this setup, users are presented with input fields for the actual HB Table. The table layout is straightforward: the first column accommodates  $\vec{H}$  values while the second hosts the corresponding  $\vec{B}$  values.

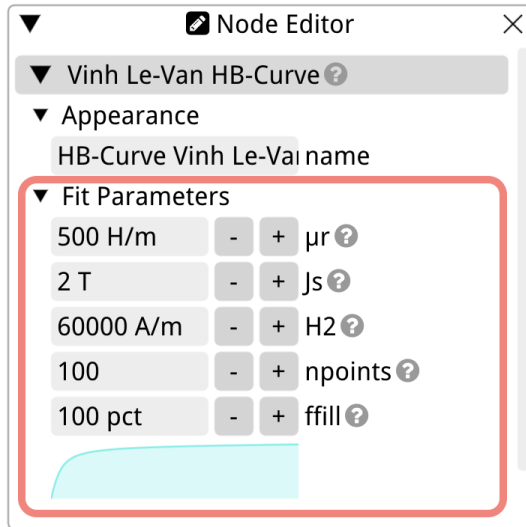


Figure 6.5.1: The *node editor* of the Vinh Le-Van fit in the Rat GUI. Hover over the small graph at the bottom to enlarge it.

Beneath this table, a miniature rendition of the HB-curve is showcased. This graph serves as a visual representation of the table data, allowing users to get a quick glimpse of the material characteristics. For those desiring a more detailed inspection, hovering over this mini graph enlarges it, offering a clearer view.

Lastly, for users who prefer not to manually input each data point, there's an option at the very bottom of the interface: a button that allows the import of the table from a file. When utilizing this feature, ensure that the file format aligns with the one used for the HB File — primarily a comma-separated format with two columns,  $\vec{H}$  and  $\vec{B}$ , and a header. Refer to Figure 6.5.3 for an example file. For your reference the *node editor* of the HB Table is given in Figure 6.5.2.

### 6.5.3 The HB-Curve File

Users have the option to import their own HB data via a file. Simply right-click the HB-Mesh that you want to add the HB-file to and select 'Add HB-Curve > HB File.' This file should be comma-separated and comprise two columns with an accompanying header. While the content of the header is flexible, the first column must contain  $\vec{H}$  values, and the second should have corresponding  $\vec{B}$  values. The number of ( $\vec{H}, \vec{B}$ ) data points is at the user's discretion. Within the *node editor*, this file can be selected using the `file` setting. Either type the full file location and name in the input field or click the '+' button to browse your system.

Once chosen, an HB-curve based on the file data is displayed at the bottom of the *node editor*. For a detailed view, simply hover the mouse over this small curve. Like all HB-Curve types in Rat, the HB-Curve File also offers a fill fraction (`ffill`) parameter. This is particularly useful when the object being modeled isn't entirely comprised of a non-linear material.

An example of an HB File is given in Figure 6.5.3. The *node editor* of the HB File object is shown in Figure 6.5.4.

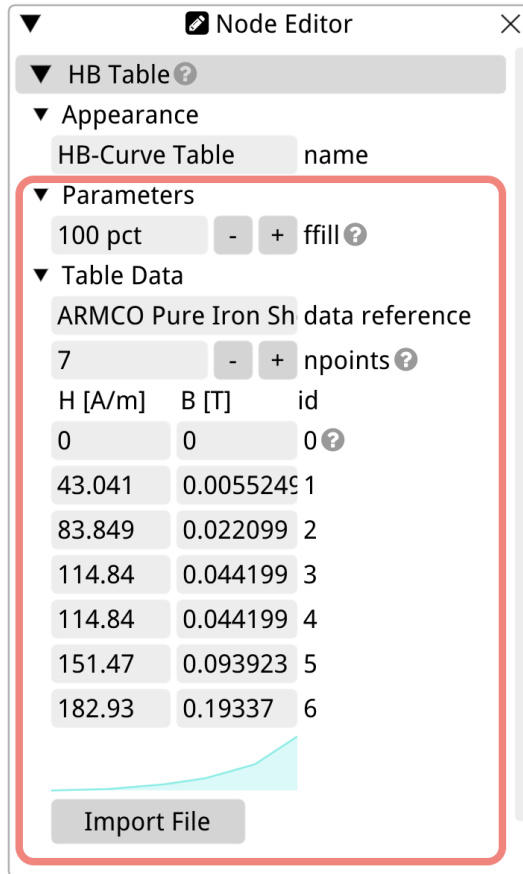


Figure 6.5.2: The *node editor* of the HB Table.

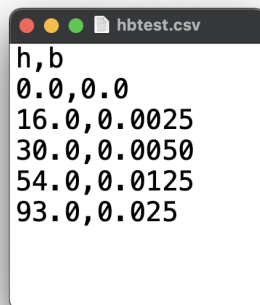


Figure 6.5.3: An example of a file that could be imported as HB File.

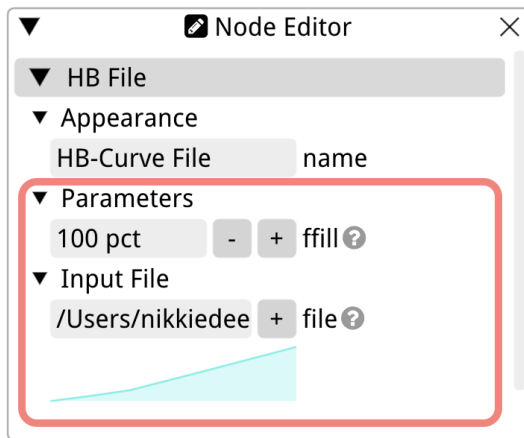


Figure 6.5.4: The *node editor* of the HB File.

## Chapter 7

# Permanent Magnets

7.1 Introduction

7.2 Custom Permanent Magnet

7.3 Bar Permanent

7.4 Ring Permanent Magnet

7.5 Sphere Permanent Magnet

# Chapter 8

## Distmesh Cross-Section

### 8.1 Introduction

Distmesh meshes are used in cases where there is no other way to make your Cross-Section in the Rat GUI. This may be the case when you want to model a U-shaped channel for a cable you are designing, or when you are working on an iron yoke for a dipole with many keyholes and complex shapes. However, whenever possible, a shape should be built using a Cross-Section and a Path - even if the Path originates from a file. This method should be favored over building the same shape with a Distmesh.

The c++ Distmesh in Rat, inspired by [13], employs distance functions to generate various shapes in tandem with the Delaunay triangulation algorithm [14]. These Distance Functions can be combined or subtracted from one another to craft custom forms. Beyond the capabilities of the original Distmesh code, a recombination algorithm has been introduced. This algorithm merges triangles into quadrilaterals. Subsequently, any remaining triangles are divided into quadrilaterals using the Catmull-Clark subdivision algorithm, facilitating the production of pure quad meshes. In Rat, this code is utilized to generate the cross-sections of cables or coils, which are later virtually extruded along their respective paths. Moreover, this code is also instrumental in meshing for non-linear materials.

Distance functions, also known as metric functions, are mathematical tools used to define the distance between pairs of elements in a set. In geometry and computer graphics, distance functions describe shapes or surfaces in space, returning the shortest distance between a given point and the described shape or surface.

For instance, consider a two-dimensional plane with a circle centered at the origin  $(0, 0)$ . The distance function for this circle would take an arbitrary point  $(x, y)$  on the plane as input and returns the distance from that point to the circle. If the point lies outside the circle, this distance is the length of the line segment from the point to the closest point on the circle's circumference. Conversely, if the point lies inside the circle, the distance could be defined as the negative of the length of the line segment from the point to the nearest point on the circle's circumference.

Distance functions can be easily added and subtracted from each other - a process known as an operation, making them ideal for defining custom shapes. Common operations include the union, difference, and intersection. This chapter first describes the general workflow for using the Distmesh. It also have a separate section for each type of distance function used in Rat.



## 8.2 Building a Cross-Section with the Distmesh

Any Custom object in Rat can have a Cross-Section created using the Distmesh. To do this, add a Custom object to the Model Tree, and then choose ‘Set Cross-Section > Distmesh’ to begin creating a two-dimensional custom shape, which is eventually extruded along the Path that is chose by the user.

Initially, the Distmesh object in the Model Tree will appear orange/red, indicating that it lacks the necessary building blocks for rendering. Right-clicking Distmesh in the Tree provides two options: ‘Edit Node(s)’ with the standard tools (see Section 3.4.1 for more information), and ‘Add Distance Function’.

Four types of distance functions are available in the ‘Add Distance Function’ list. The first group comprises distance functions for basic geometric shapes like circles and ellipses. The second group includes distance functions resulting from operations on multiple other distance functions, such as union, difference, and intersection. The third group, featuring arrays and angular arrays, comprises groups of identically shaped distance functions. Finally, the fourth group consists of scale distance functions, which determine the element size of the mesh. The Rat offers two scale functions: Scale and Ones (default).

When modeling custom cross-sections with the Distmesh in the Rat GUI, the typical workflow begins with adding shapes to the Model Tree and customizing them to meet your needs. This includes setting the radii, sizes, and other geometric parameters of the shapes to match your design (explained in Section 8.3). If your design cannot be created with standard shapes - which is likely why you need a Distmesh Cross-Section in the first place - you can add an operation distance function (explained in Section 8.4). Groups of distance functions allow users to replicate shapes across space (explained in Section 8.5). Combining standard shapes with operations and groups should enable you to model any required shape.

In most cases, this will be the end of the process. However, in rare cases where more control over the element size in your Distmesh Cross-Section is required, you can apply a scale function. This is an advanced feature of the distmesh and is generally not necessary. Section 8.6 explains the use case of the scale function and provides instructions for use.

The way that Distmesh works is such that you can only have two distance functions as child nodes for the Distmesh Cross-Section node: one is the shape function and the other is the scale function. However, this does not mean that you can’t combine multiple operations or groups with each other. The shape distance function could be a union of a union and a difference, where that union is a combination of an intersection and a rectangle, and so forth. Combining distance function operations in the correct order might take a bit of practice, but it is overall not too complicated.

Once you have created your first shape distance function, you can right-click on it and choose between ‘Add Distance Function’ (in the case of a union, you need to have at least two child nodes with distance functions) or ‘Create Group.’ The group can be an operation, and it will be placed at the parent level of the node you are including in the group. If that group is another operation, you need to make sure to add a second distance function so that the operation is performed.

An example of combining multiple operations, groups of distance functions, and distance functions themselves is shown in the keyhole example of Figure 8.2.1. This was created by first making a union between two circles. Then, this union was placed in a difference operation along with a rectangle, which created the flat bottom part of the keyhole. That difference was finally placed in another difference operation along with a rectangle. That last rectangle forms the solid block around the keyhole. This keyhole was created using a Custom Coil that has this Distmesh as its Cross-Section and a Path Group’s Straight section as its Path.

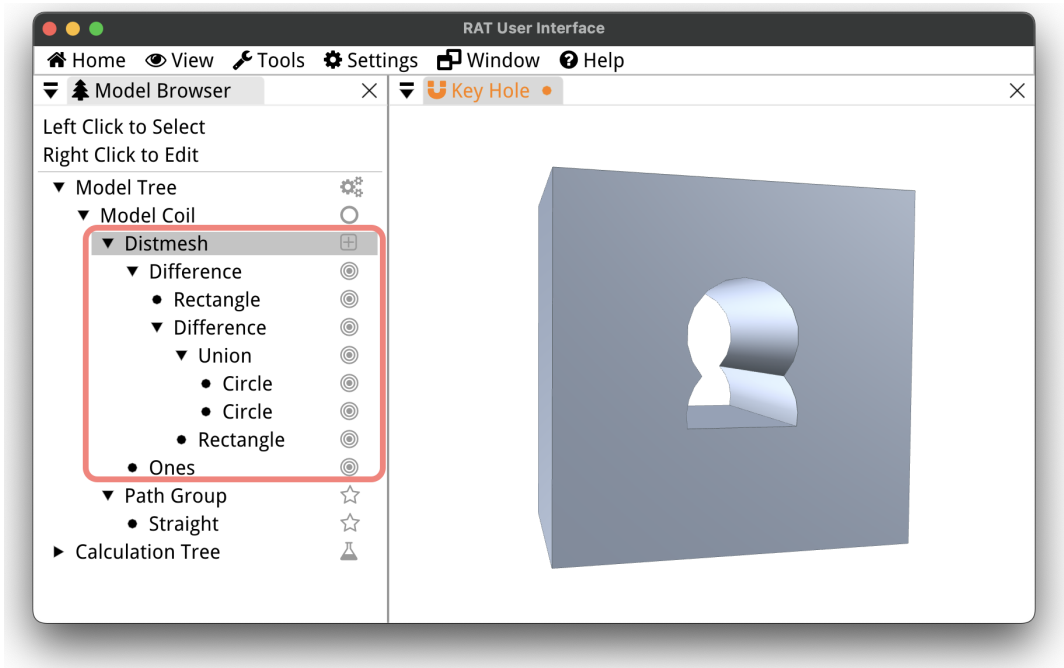


Figure 8.2.1: An example of combining multiple distance functions to model a keyhole.

It is important to note that the order in which the different objects in a union, difference, or intersect operation are placed matters because of how distance functions are defined. This order for each distance function operation is explained in Section 8.4. If the Distmesh object is valid, and the Custom object you are modeling has a Path associated with it, it will be drawn immediately in the *viewport*.

When you've completed the Distmesh Cross-Section by combining multiple Distance Functions, the Path selected for the Custom object you are constructing will pass through the center of that Cross-Section. This central path placement is independent of the relative positioning of individual Distance Functions to the Distmesh origin. For instance, if all Distance Functions were offset by 30 mm from the origin, the Path would still traverse the center of the Distmesh Cross-Section.

Just like any other object in the Rat GUI, the Distmesh Cross-Section has its dedicated *node editor*. Beyond the **name** of the Cross-Section, there are four additional parameters to consider. Given that the shapes created with the Distmesh are intricate, the meshing algorithm is computationally intensive compared to other Cross-Sections. This means there's a higher likelihood to either crash the system or run out of memory. As a safeguard, the Distmesh Cross-Section introduces the **nodelimit** parameter – a limit on the number of nodes for the Custom object you're modeling. Should you surpass the **nodelimit**, the mesh will not render, and a message will appear in the Console notifying you that the node limit has been exceeded. More details about the Console Window can be found in Section 3.7.5.

Instead of decreasing the node limit, you can opt to adjust the target element size of the Distmesh with the **h0** parameter. In that case you can increase **nodelimit** again. The next setting in the *node editor* is the **rng seed** for the random number generator. When this seed is modified, it results in the generation of a different mesh. After updating this setting you can view the mesh changes by using **Ctrl+M**. If for some reason the mesh is not rebuild

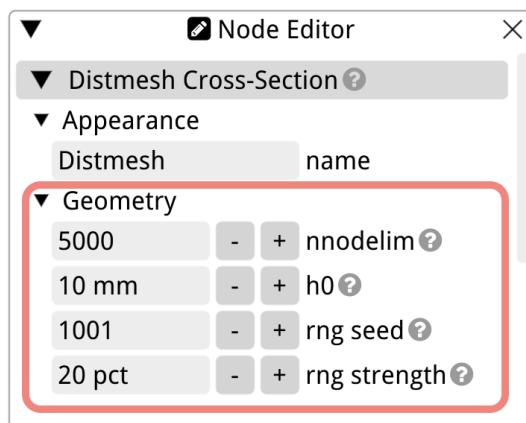


Figure 8.2.2: The *node editor* of the Distmesh Cross-Section.

automatically you can press the space bar on your keyboard to force a rebuild of the mesh. The strength of the variation is set with the last parameter `rng strength`. If this is set to 0% the mesh will not change, even if you vary `rng seed`.

## 8.3 Shapes

The Distmesher has a few standard Distance Functions for commonly used shapes. These include a circle, an ellipse, a rectangle, a rounded rectangle, a polygon, and a poloidal magnetic flux plane for plasma shapes. Each of these are described and their use is explained in the following separate subsections.

Each shape in the Distmesh is defined in a local two-dimensional coordinate system, just like for any other Cross-Section. Since this Cross-Section is a Distmesh Object,  $(0, 0)$  is the origin of the Distmesh. All Distance Functions added to the Distmesh Cross-Section will be oriented with respect to this origin.

### 8.3.1 Circle Distance Function

The first shape in the list of predefined Distance Functions in Rat is the Circle Distance Function. It is defined by the location of its center within its two-dimensional local coordinate system (like any Cross-Section, see Chapter 9) and its `radius`. The `radius` defines the circle's size by representing the distance from its center to its edge. All its parameters in the Rat GUI are shown in Figure 8.3.1.

The normal coordinate, `nc` in the Rat GUI, dictates the normal component of the coordinate of the circle's center. By default, the circle's center aligns with the Distmesh Cross-Section's origin, which is  $(0, 0)$ . However, adjusting the `nc` parameter allows users to offset the circle in the normal direction.

Similarly, the transverse coordinate, `dc`, controls the transverse component of the circle's center. This enables a shift of the circle's center in a direction perpendicular to the normal coordinate, granting flexibility in its placement within the cross-section.

The local coordinate system of the Distmesh and a Circle Distance Function inside it are shown in Figure 8.3.2. The `radius` is shown in this figure as well as the  $(n_c, d_c)$  coordinate.

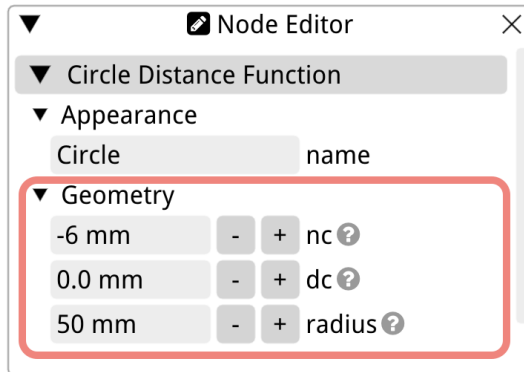


Figure 8.3.1: The *node editor* of the Circle Distance Function.

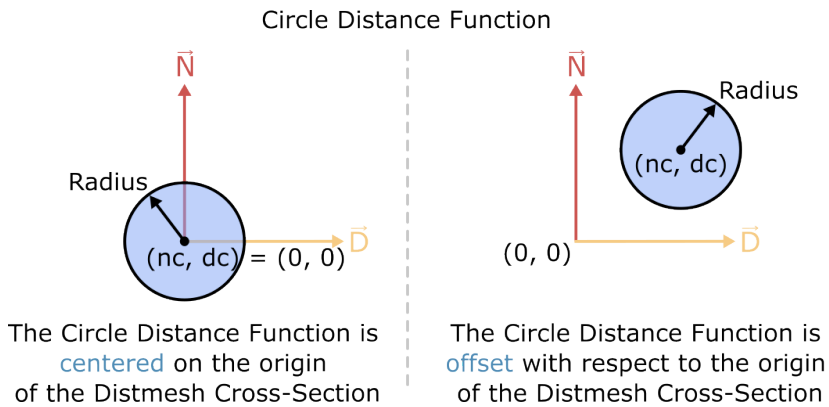


Figure 8.3.2: The local coordinate system of the Distmesh Cross-Section with a Circle Distance Function in it. On the right figure the Circle Distance Function is offset with respect to the origin of the Distmesh Cross-Section.

## 8.3.2 Ellipse Distance Function

The ellipse is another fundamental shape available as a predefined Distance Function in Rat. It is characterized by the position of its center within a two-dimensional local coordinate system of the Dismesh object, much like any Cross-Section (refer to Chapter 9). However, unlike a circle, an ellipse has two distinct radii: one in the normal direction, denoted with **a**, and the other in the transverse direction, called **b**. The **a** and **b** parameters define the ellipse's dimensions, capturing the distances from its center to its boundary along the respective axes. The parameters as they appear in the Rat GUI are presented in Figure 8.3.3.

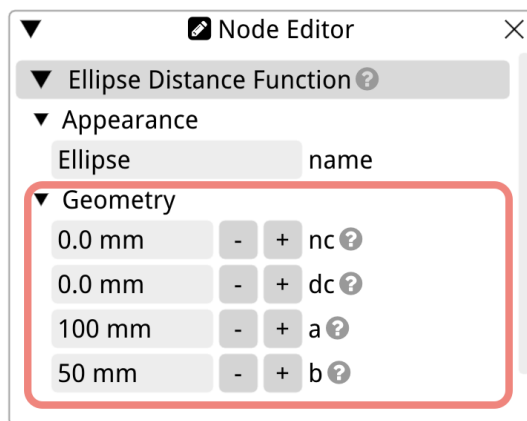


Figure 8.3.3: The *node editor* of the Ellipse Distance Function.

The normal coordinate, denoted as **nc** in the Rat GUI, represents the normal component of the coordinate of the Ellipse Distance Function's center. By default, this center aligns with the origin of the Dismesh Cross-Section, set at (0,0). Tweaking the **nc** parameter offers users the capability to offset the ellipse along the normal direction.

In a similar vein, the transverse coordinate, **dc**, dictates the transverse component of the ellipse's center. This provides an avenue to adjust the ellipse's position perpendicularly to the normal coordinate, allowing more versatile placements within the Dismesh Cross-Section.

The local coordinate system of the Dismesh with an Ellipse Distance Function embedded is depicted in Figure 8.3.4. Both the **a** and **b** radii are highlighted in this representation alongside the  $(n_c, d_c)$  coordinate.

## 8.3.3 Rectangle Distance Function

The Rectangle Distance Function is another predefined shape available in Rat. This function creates a rectangular shape defined by the coordinates of its corners within its local two-dimensional coordinate system, similar to the Circle and Ellipse Distance Functions. This is shown in Figure 8.3.5.

The Rectangle's definition primarily hinges on the specification of its four corner points. The **n1** parameter designates the normal coordinate of the bottom-left and bottom-right corners, while the **n2** parameter specifies the normal coordinate of the top-left and top-right corners. On the transverse axis, the **d1** parameter sets the transverse coordinate for the leftmost corners (both top and bottom), and the **d2** parameter establishes the transverse coordinate for the rightmost corners.

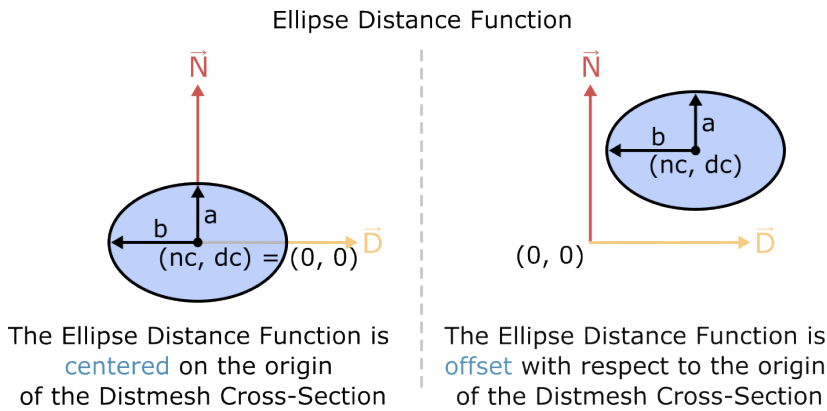


Figure 8.3.4: The local coordinate system of the Distmesh Cross-Section containing an Ellipse Distance Function. On the right, the Ellipse Distance Function is offset relative to the origin of the Distmesh Cross-Section.

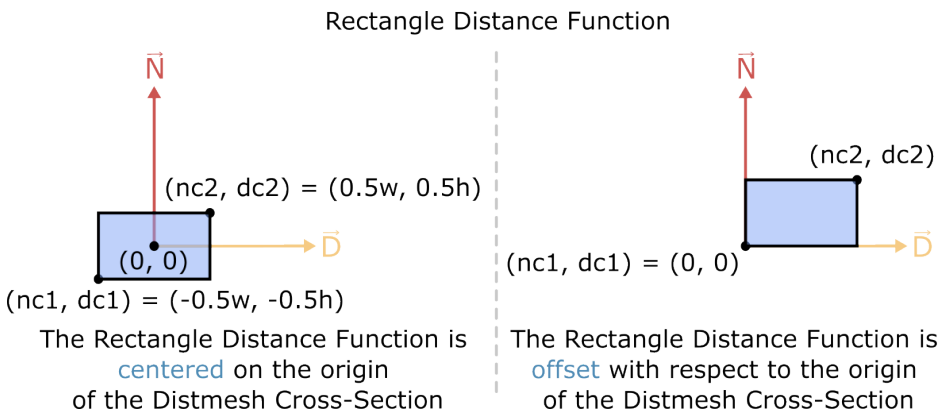


Figure 8.3.5: The local coordinate system of the Distmesh Cross-Section containing a Rectangle Distance Function. On the right, the Rectangle Distance Function is offset relative to the origin of the Distmesh Cross-Section.

This structure provides flexibility, allowing the user to create rectangles of varying dimensions and orientations, as well as customize the placement and orientation of a rectangle within the Distmesh Cross-Section. The *node editor* for the Rectangle Distance Function is shown in Figure 8.3.6. There you will find all the parameters discussed in this subsection.

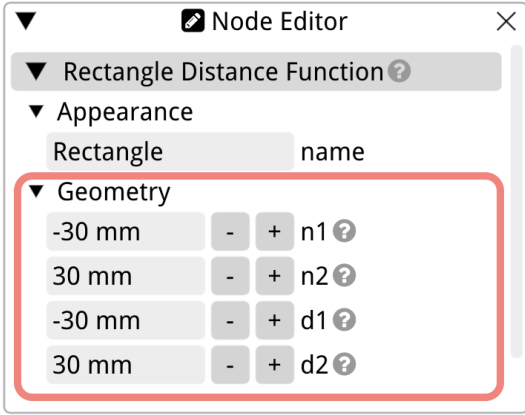


Figure 8.3.6: The *node editor* of the Rectangle Distance Function.

### 8.3.4 Rounded Rectangle Distance Function

The Rounded Rectangle Distance Function is yet another shape available for selection in the Distmesh Cross-Section. Like its counterpart, the Rectangle Distance Function, it is defined by the coordinates of its corners. However, in addition to these parameters, the Rounded Rectangle has a distinct feature: the curvature of its corners. This is visualized in Figure 8.3.7.

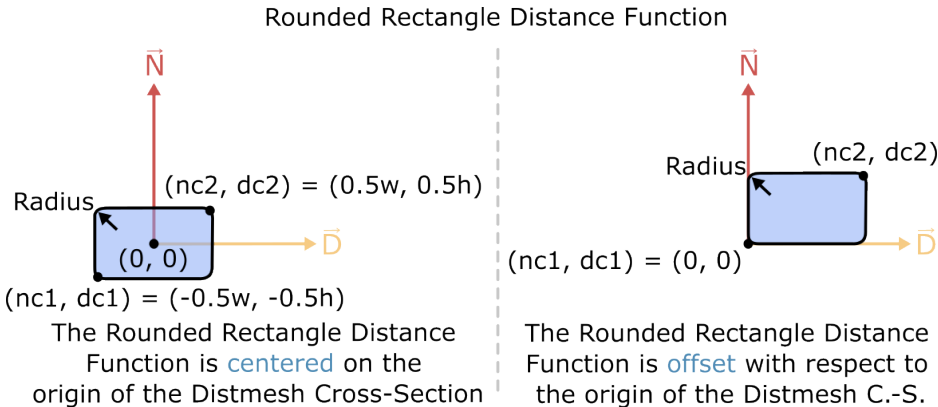


Figure 8.3.7: The local coordinate system of the Distmesh Cross-Section featuring a Rounded Rectangle Distance Function. On the right, the Rounded Rectangle is offset in relation to the origin of the Distmesh Cross-Section.

The primary distinguishing characteristic of the Rounded Rectangle is its **radius**. This parameter determines the radius of curvature for the corners, transforming what would be angular points into smooth, rounded edges.

Defining the corners of the Rounded Rectangle follows a similar approach as the Rectangle Distance Function. The parameters `n1` and `n2` specify the normal coordinates of the bottom and top corners respectively. Meanwhile, `d1` and `d2` determine the transverse coordinates of the leftmost and rightmost corners.

Combining these specifications grants users the ability to craft a wide range of shapes, from near-perfect squares with soft corners to elongated rectangles with subtly rounded edges. The exact placement, dimensions, and degree of curvature can be fine-tuned within the Distmesh Cross-Section, as showcased in the *node editor* for the Rounded Rectangle Distance Function found in Figure 8.3.8.

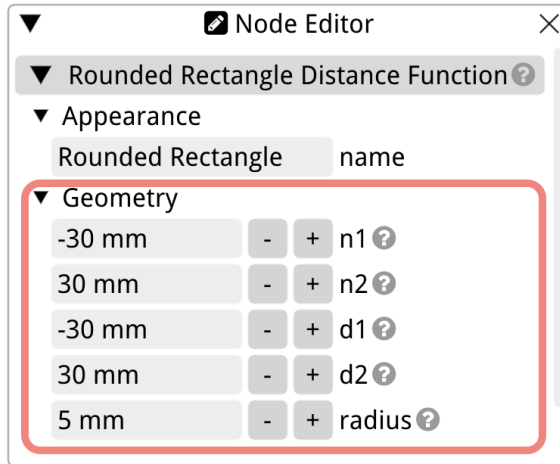


Figure 8.3.8: The *node editor* of the Rounded Rectangle Distance Function.

### 8.3.5 Polygon Distance Function

In the Rat GUI, another versatile shape at the disposal of users is the Polygon Distance Function. Unlike other Distance Functions that may use parameters to define a preset shape like circles or ellipses, the polygon function provides the freedom to create more complex and customizable forms using the vertices or corner coordinates of the desired polygon. The depiction of such a polygon within the Distmesh Cross-Section's local coordinate system is showcased in Figure 8.3.9.

When defining the polygon, the initial step requires users to specify the total number of points, or corners. This number determines the rows available in the subsequent input table. Each row accepts the coordinates for one vertex or corner. The input table has two columns: the normal coordinate and the transverse coordinate. As the user progresses through the table, they should remember to move from one corner to its adjacent corner. This is indicated in Figure 8.3.9 with the coordinate labels  $(n_{c,x}, d_{c,x})$ , where  $x$  is the index of the coordinate. Random ordering of the corners will cause Rat to fail to draw the polygon.

The final adjustment pertains to the polygon's size. Using the scale factor `fscale`, users can either magnify or reduce the polygon's dimensions. This factor, represented in percentages, increases the size when set above 100% or decreases it if the value falls below 100%.

A unique aspect of this Distance Function is its adaptability. It empowers users to model a wide array of polygons, from simple triangles to intricate many-sided figures. These can



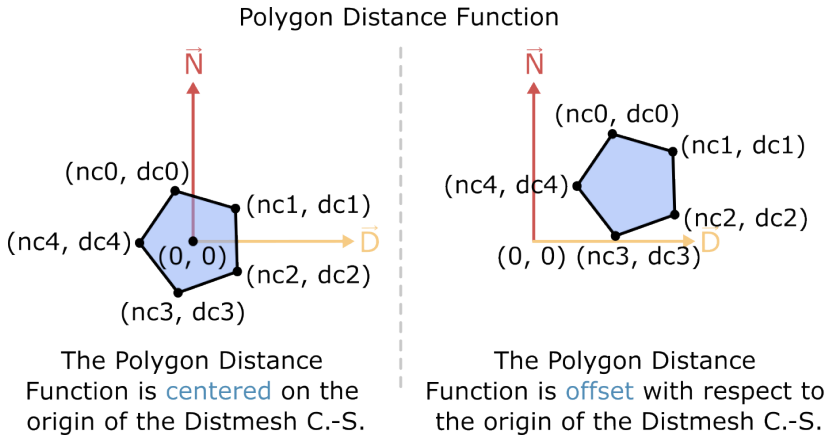


Figure 8.3.9: The local coordinate system of the Distmesh Cross-Section with the Polygon Distance Function. Variations in the polygon’s shape, as determined by the entered corner coordinates, are evident.

be either convex or concave in nature, but it’s crucial to note that the polygon’s final shape strictly depends on the sequence of entered corner coordinates: the corners are input from one corner to the next adjacent corner and so on. As a result, careful attention is advised during input to achieve the desired polygon structure.

The *node editor* specific to the Polygon Shape Distance Function is illustrated in Figure 8.3.10. This figure provides an overview of the settings and parameters of the Polygon Distance Function, in this case four corners of a quadrilateral are set.

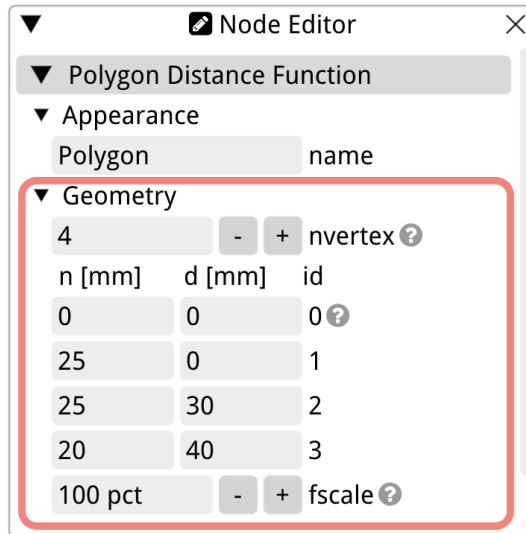


Figure 8.3.10: The *node editor* dedicated to the Polygon Distance Function.

### 8.3.6 Plasma Distance Function

The Plasma Distance Function in Rat offers a modeling approach for the two-dimensional poloidal magnetic flux surface of a magnetic confinement device, capturing its intricate geometrical nuances. This function accommodates the distinctive features of standard plasma confinement devices by considering several descriptive parameters. The shape of the plasma in two dimensions can be calculated with the poloidal magnetic flux function,  $x$  and  $y$ :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \cdot \cos(\theta + \delta \cdot \sin \theta) \\ \kappa \cdot a \cdot \sin \theta \end{bmatrix}, \quad (8.3.1)$$

and the aspect ratio  $A$ :

$$A = \frac{r_0}{a}. \quad (8.3.2)$$

Here, the major radius, denoted as  $r_0$ , is the primary radius of the plasma confinement device, indicating the distance from the torus’s center to the middle of its tube-like cross-section. This is complemented by the minor radius, termed  $a$ , which represents the radius of the elliptical plasma shape.

Further adding to the shape’s complexity, the elongation, known as  $\kappa$ , quantifies the cross-section’s deviation from a perfect circle. An elongation value exceeding 1 suggests the cross-section is stretched, giving the plasma shape a more elliptical appearance in the poloidal plane. Lastly, the triangularity parameter, labeled  $\delta$ , characterizes the unique “d-shap” of the plasma, shedding light on its deviation from being either purely circular or elliptical. All these parameters are illustrated in Figure 8.3.11.

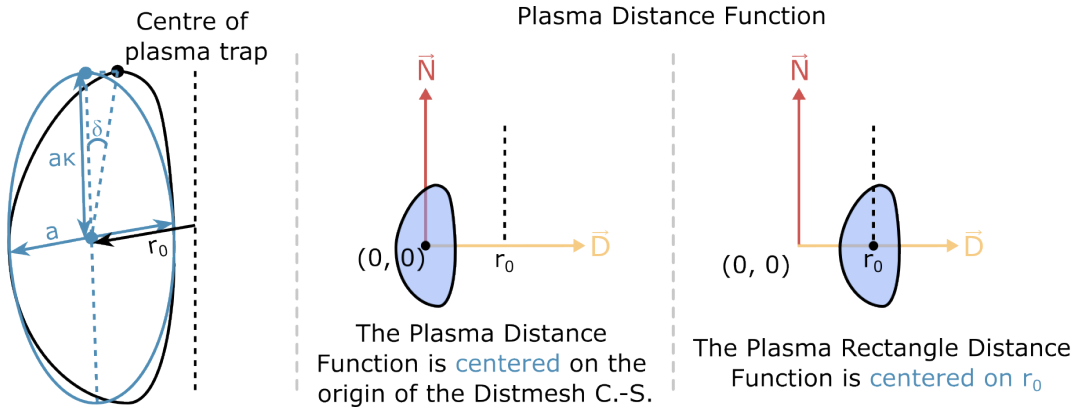


Figure 8.3.11: The definition of the poloidal magnetic flux surface on the left, and the Plasma Distance Function in the local coordinate system of the Distmesh Cross-Section in the middle and right. The right most figure shows the Plasma Distance Function centered on  $r_0$ .

The *node editor* of the Plasma Distance Function, as is shown in Figure 8.3.12, includes all four parameters that define the poloidal magnetic flux surface. Moreover, the first setting available in the *node editor*, referred to as “center,” plays a role in the orientation of the poloidal magnetic flux surface. This option ensures the Distance Function’s alignment with the major radius, in case you are preparing the Cross-Section for extrusion along a circular path.

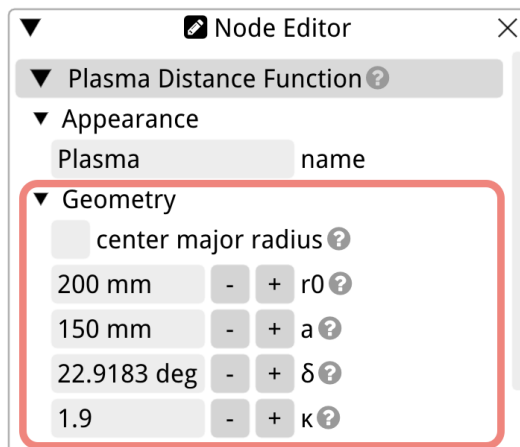


Figure 8.3.12: The *node editor* of the Poloidal Magnetic Flux Surface Distance Function, emphasizing the essential parameters.

## 8.4 Operations on Distance functions

Distance function operations serve as the foundation to combine, modify, or create new geometrical shapes based on the principles of distance functions. Essentially, these operations process multiple distance functions as input to generate a new resultant distance function. The outcome of these operations hinges on the positional relationship of points concerning the given shapes described by the distance functions.

To reiterate, consider a simple scenario involving the distance function for a circle centered at the origin  $(0, 0)$ . This function, for any arbitrary point  $(x, y)$  within its plane, determines the shortest distance from that point to the circle. When the point is exterior to the circle, the determined distance—represented as a positive value—is the length of the line segment extending from that point to the closest position on the circle’s circumference. Conversely, for points situated within the circle, the distance is designated as a negative value, representing the segment length from the point to the nearest location on the circle’s perimeter.

In the context of modeling cross-sections using distance functions, the region of interest is typically represented by points that yield negative distance values. Using our circle example, this encompasses all the points located inside the circle.

The precise outcome of an operation, such as a difference operation on distance functions, depends on which shape is subtracted from another. This distinction becomes crucial as it delineates which points are considered inside the boundary and which ones are outside. The next subsections delve deeper into individual distance function operations, elucidating the mechanism of each operation and emphasizing the significance of the sequence in which these operations are executed.

If you look at the *node editor* of the distance function operations, you will see that they can only be named. Apart from the name, they can’t be customized. All the customization happens in the definition of the shapes that are inside the operations. Those *node editors* are discussed in Section 8.3.

### 8.4.1 The Union Operation

The union operation derives from comparing two distance functions and determining their minimum. By visualizing distance functions as representations of object surfaces, the union operation merges both into a combined object. This new object encompasses all points from the initial shapes, and its corresponding distance function is, essentially, the minimum value between the two originals. For distance functions denoted as  $a(x, y)$  and  $b(x, y)$ , the union, represented by  $c(x, y)$ , is defined by the equation below:

$$c(x, y) = \min(a(x, y), b(x, y)). \tag{8.4.1}$$

From a geometric perspective, this union operation effectively amalgamates two entities into one. However, this shouldn't be mistaken for the arithmetic addition of two surfaces. For instance, uniting two circles yields a single shape encompassing both circles' points without double-counting overlapping areas. This concept is illustrated in Figure 8.4.1. Notably, the sequence in which the shapes are united is inconsequential.

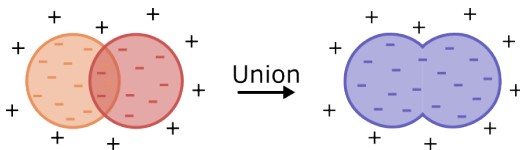


Figure 8.4.1: The union of two circle distance functions, illustrated for clarity with positive and negative signs. The regions marked negative lie within the boundary, therefore these areas will be drawn by the Dismesher.

In the Rat framework, unions offer flexibility. They can be nested — a union comprising another union and an additional shape, for instance. Furthermore, users can craft a union between a differential shape and another entity or between an array and a shape. Moreover, a union is not limited to two shapes; it can integrate three or more, with only the interior points of each shape being rendered. This capability is illustrated in Figure 8.4.2.

### 8.4.2 The Difference Operation

The difference operation  $A - B$  intuitively corresponds to subtracting one shape from another. Imagine you have a block of clay represented by  $A$  and you carve out a shape  $B$  from it. The resulting shape is the difference  $A - B$ .

Technically speaking, this operation captures the intersection of  $A$  with the space not occupied by  $B$ . This means we look for areas where  $A$  exists but  $B$  does not. This can be formulated by combining the distance function for  $A$  and the negated distance function for  $B$ . Using our earlier notation for distance functions  $a(x, y)$  and  $b(x, y)$ , the difference,  $c(x, y)$ , is given by:

$$c(x, y) = \max(a(x, y), -b(x, y)). \tag{8.4.2}$$

Imagine you subtract a small circle from a bigger circle. The resulting shape will look like a doughnut or a ring, with the smaller circle carved out from the center of the bigger one. The sequence in which the shapes are arranged in the operation matters. If you take the difference in the reverse order (small circle minus big circle), the result will be an empty set, because the smaller circle cannot subtract anything from its larger counterpart. You can

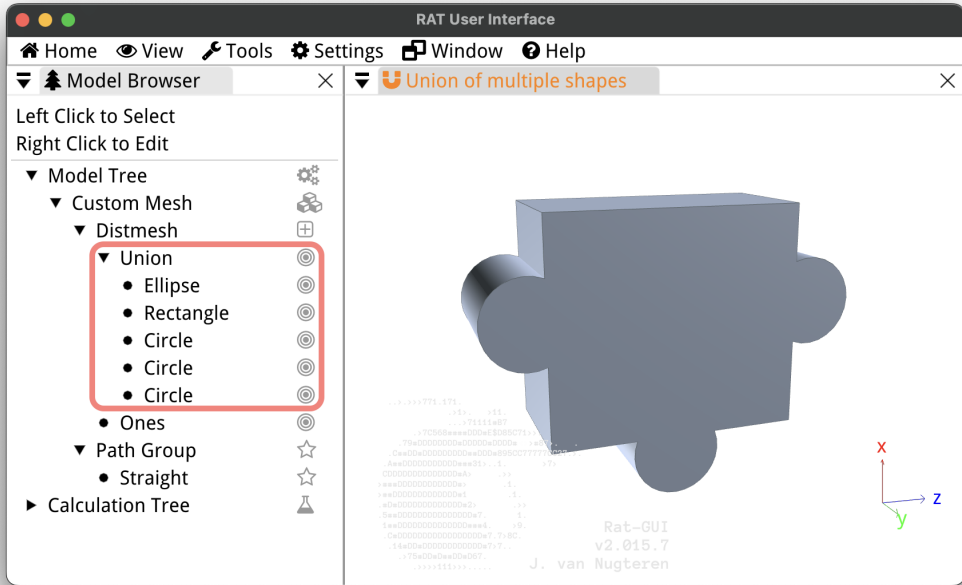


Figure 8.4.2: An example of how multiple shapes are combined in a union in the Rat GUI.

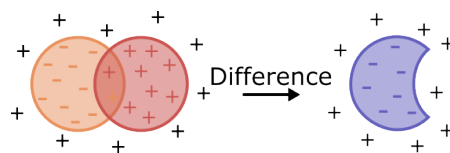


Figure 8.4.3: An illustrative example of how the difference operation acts between two circular distance functions. For visualization, the values of the distance functions might be represented with signs. Shapes in the negative region are considered to be inside the boundary and are depicted.

also create moon-like shapes. In that case you subtract part of one circle from another. This is visualized in Figure 8.4.3.

In Rat, the difference operation offers a diverse set of applications. It's possible to carve out a difference between an array and a distinct shape or even between a conglomerate of shapes (union) and another individual shape. Much like the union operation, the difference can be executed recursively. This paves the way for sophisticated designs where a foundational shape undergoes a succession of subtractions to manifest the desired end product.

It's imperative to emphasize that, unlike the union operation, sequence is pivotal in the difference operation, a fact underscored by its definition in Equation (8.4.2). The order, as it appears in the Model Tree, dictates which shape is pared away from which. The first distance function nestled within the difference object takes on the role of the  $A$  object, and any subsequent shapes listed under the same difference operation are methodically subtracted from  $A$ . Analogous to the union, more than a pair of distance functions can be introduced to the difference; every distance function from the second onward is subtracted from the first. A visual representation of multifaceted difference operations in a Dismesh Cross-Section within the Rat GUI is available in Figure 8.4.4.

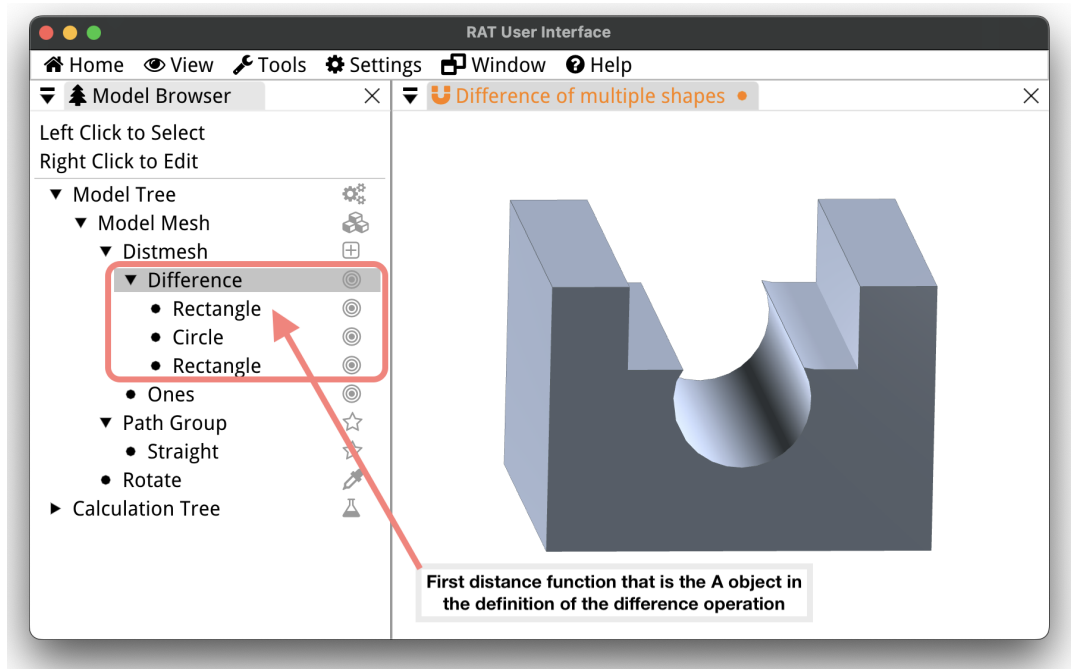


Figure 8.4.4: An example of how multiple shapes interact in a difference operation within the Rat GUI.

A word of caution: when deploying the difference operation, it's crucial to ascertain that the regions designated for subtraction from one shape don't coincide with the boundaries of both shapes involved. Take, for instance, the creation of a U-shaped channel with the Dismesh. The diminutive rectangle assigned to subtract from the larger rectangle should protrude slightly from the edge crafting the U-shape's aperture, as shown in Figure 8.4.5a. Failing to observe this might lead Rat to render an infinitesimally thin line along the original edge, as illustrated in Figure 8.4.5b, potentially complicating the calculations.

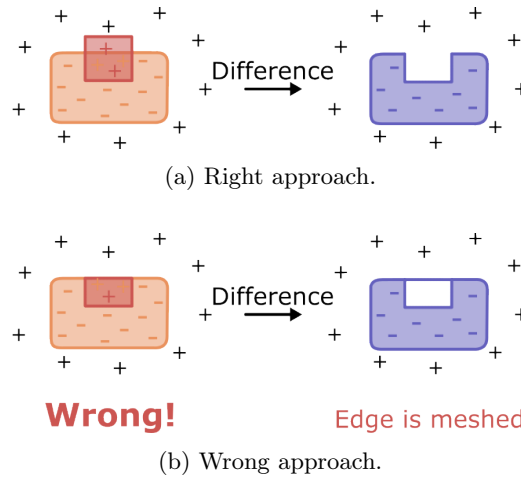


Figure 8.4.5: Drawing a U-shaped channel with the Distmesh Cross-Section, where the smaller rectangle protrudes the edge of the bigger rectangle. The bottom image is an example of what not to do.

### 8.4.3 The Intersect Operation

The intersection operation embodies the shared region of two shapes defined by their distance functions. In mathematical terms, when considering the distance functions of two shapes, the intersection of these shapes is represented by the maximum of the two distance functions. This is because, within the space of distance functions, a point belongs to the intersection if and only if it is inside both shapes. For two distance functions, denoted as  $a(x, y)$  and  $b(x, y)$ , their intersection, represented by  $c(x, y)$ , is formulated as:

$$c(x, y) = \max(a(x, y), b(x, y)). \quad (8.4.3)$$

Visually, an intersection captures the overlap of two objects. If you visualize two circles partially overlapping, the intersection would consist of the shared region between them. The result won't contain any area that's exclusive to just one of the circles. This principle is graphically demonstrated in Figure 8.4.6.

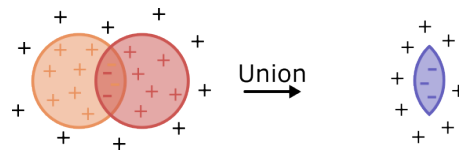


Figure 8.4.6: An example showcasing the intersection of two circle distance functions. Areas exclusive to a single circle are omitted, retaining only the overlapped region.

In Rat, the intersection operation can be applied in a plethora of ways. You might intersect a union of shapes with another shape, or derive the intersection between an array and a separate shape. Just like the union and difference operations, the intersection operation in Rat can be employed recursively. This versatility fosters intricate designs, whereby shapes can be combined in layers to produce more complex geometries.

One salient point to bear in mind is the order of the distance functions in the Model Tree doesn't influence the outcome of the intersection, since the operation is commutative. Yet,

proper organization in the Model Tree can be instrumental in keeping the design process streamlined and intuitive.

## 8.5 Shape Groups

In the realm of intricate design, the need for repeated patterns or symmetrical structures frequently arises. With this in mind, the Rat GUI offers the Shape Group Distance Functions. Currently there are two Shape Groups available: the Array and the Angular Array. These empower users to replicate a particular design multiple times in a straight or circular pattern, and its behavior is governed by a few key parameters explained in their respective subsections.

The Array and the Angular Array are shown in Figure 8.5.1. The left figure shows a two-dimensional Array of a Rectangle Distance Function, where it is copied three times in the horizontal direction, and three times in the vertical direction. On the right of the same figure an Angular Array is shown of the same rectangle, where it is copied six times (seven shapes in total) in a nine-fold symmetric angular array. For this shape, the number of copies was set to seven, even though the symmetry was set to nine to show what the possibilities of the Rat GUI are.

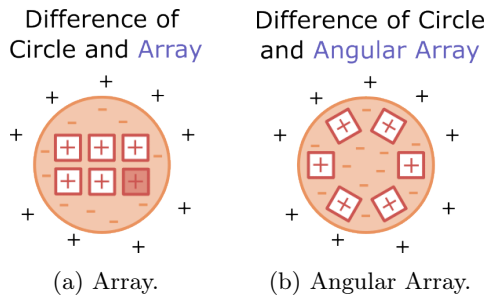


Figure 8.5.1: The two types of Array Distance Functions in Rat.

Both types of Arrays can contain all Distance Functions available in the Rat GUI. For instance, you can place a cross-section that you created with a Difference of a Circle and Rounded-Rectangle Distance Functions inside the Angular Array. As for most features of the Distmesh Cross-Section, the copy (**Ctrl+C**), cut (**Ctrl+X**), and paste (**Ctrl+V**) tools are very handy when moving Distance Functions in or out of different groups. And you can always place previously made Distance Functions inside a group by right-clicking the Distance Function in the *model browser* and selecting ‘Create Group.’

### 8.5.1 Array Distance Function

The Array Distance Function allows users to create structured layouts by replicating shapes (refer to Section 8.3) across two primary dimensions in straight lines. The *node editor* of the Array Distance Function is shown in Figure 8.5.2.

By default, the origin of the array aligns with the origin of the initial shape that is inside the Array Distance Function. However, if you’re looking to center your design, the **center** option is in the *node editor* can be used. Activating this option shifts the origin of the array to its center, giving your layout a more symmetrical appearance.

The number of shape replicas is another essential aspect. With **ndim1**, you can set the number of copies in the first dimension, while **ndim2** lets you specify the copies in the second



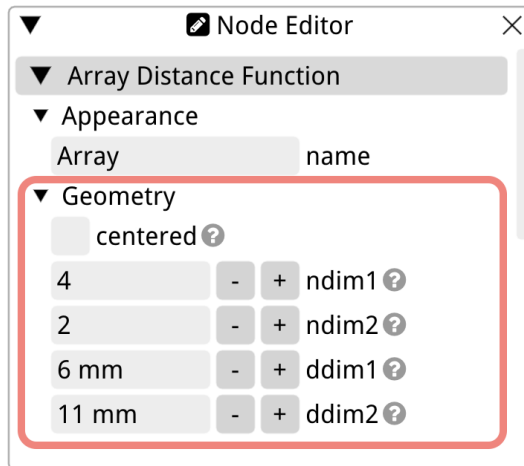


Figure 8.5.2: The *node editor* of the Array Distance Function.

dimension. The distance between each of these shape copies is also adjustable. The `ddim1` parameter controls the spacing of the objects in the array's first dimension. It's worth noting that if this spacing is less than the size of the object in this dimension, you might end up with overlapping shapes. Similarly, `ddim2` adjusts the spacing in the second dimension, with the same caution about potential overlaps.

### 8.5.2 Angular Array Distance Function

The Angular Array feature allows for replicating a particular design multiple times in a circular pattern. The behavior is governed by a few key parameter that are explained in this section. They are also shown in the *node editor* of the Angular Array Distance Function in Figure 8.5.3.

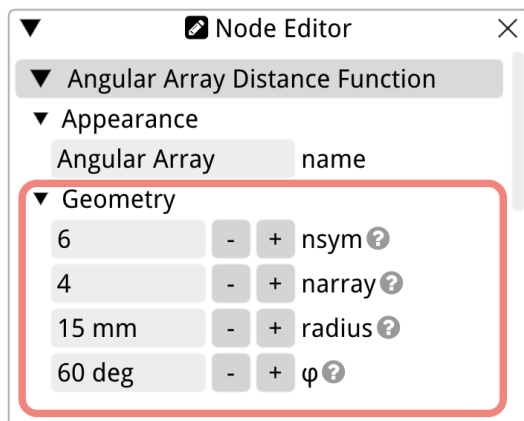


Figure 8.5.3: The *node editor* of the Angular Array Distance Function.

The order of the n-fold symmetry, referred to as `nsym` in the *node editor*, is pivotal as it establishes the rotational symmetry of the angular array. In simple terms, it dictates how often your design is echoed around a circle. For instance, opting for a four-fold symmetry

would result in your design being mirrored four times, with each instance placed at  $90^\circ$  intervals.

Next, we have the number of copies, denoted as `narray`. This value pinpoints the exact count of replications you desire for your design within the array. It's worth noting a special interaction here: if your `nsym` value happens to be less than your `narray` value, the Rat GUI will only draw the number of objects equal to the `nsym` value. The `narray` feature grants users the flexibility to, for example, choose a five-fold symmetry but wish for only three of those designs to be displayed.

Shifting focus to positioning, the starting radius (`radius`) comes into play. This value represents the gap between the circular array's center (or origin) and the beginning point of the array. In effect, the origin of the Distance Function nestled inside the array is determined by this radial measurement, signifying that the commencement of your design drawing will adhere to this specified distance.

Lastly, for those seeking fine-tuned adjustments, there's the angular offset  $\phi$ . This parameter facilitates a rotation of the whole array setup by a designated angle, represented by  $\phi$  degrees, ensuring that your design can achieve that perfect angular placement.

To make the most of the Angular Array feature, begin by settling on the desired symmetry using the `nsym` parameter, which aids in visualizing the design's distribution. Then, dictate the total design replications with `narray`, adjust the positioning using `radius`, and finalize any angular refinements via the `angularOffset` parameter. These tools collectively make the task of producing symmetrical and repetitive designs both precise and adaptable.

An example of how you could use the Angular Array Distance Function is for making star shapes. For instance, take a Difference of a Circle and Rectangle Distance Function, and place the Rectangle in the Angular Array. Set the radius to 0 mm and copy it a few times to end up with a star-shape, such as in Figure 8.5.4.

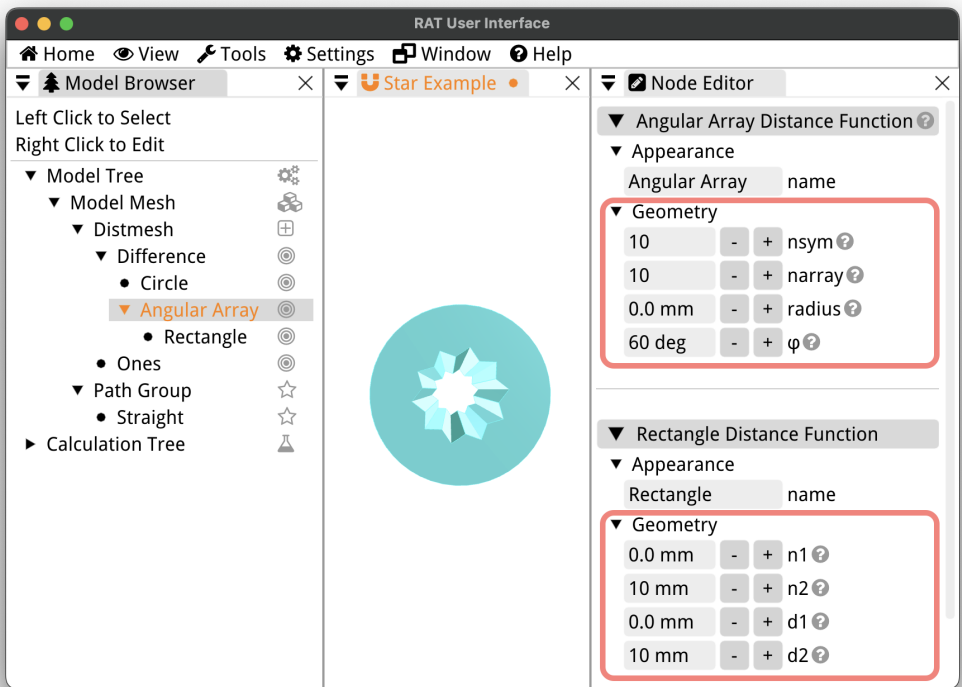


Figure 8.5.4: An example of how a star-shape can be drawn with the Angular Array in the Rat GUI.

## 8.6 Scale Functions

Every Distmesh incorporates a scale function to dictate the size of its mesh elements. By default, this function is named the Ones Distance Function. As implied by its name, the Ones Distance Function assigns a consistent value of one to all elements. The result is a uniformly distributed mesh where every element shares the same target size. For the majority of scenarios, where there's no need for varying element sizes, this uniformity is adequate. Essentially, when there's no demand to refine the mesh in certain areas, the mesh generation process, backed by the Ones Distance Function, becomes straightforward and efficient.

Nevertheless, there are instances where the Ones Distance Function may not be the best fit. This is particularly true when the domain presents diverse geometric features, or certain regions exhibit pronounced gradients in the solution. In such cases, a more adaptive approach to meshing is imperative, one which adjusts the element size in accordance to local geometry or solution nuances. To address this, Rat introduces the Scale Distance Function, accessible via 'Add Distance Function > Scale'. This replaces the Ones Distance Function in the Model Tree by the Scale Distance Function.

Understanding the Scale Distance Function necessitates a grasp of two primary parameters: the `offset` and the `scale`. Both of these are given in the *node editor* of the Scale Distance Function, refer to Figure 8.6.1.

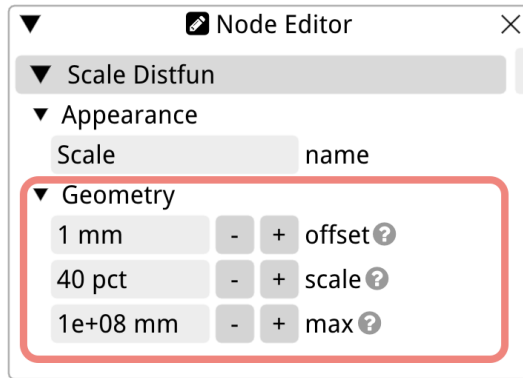


Figure 8.6.1: The *node editor* of the Scale Distance Function.

**Offset:** At its core, the Scale Distance Function remains a distance function. Its value at a boundary essentially defines the offset, which in turn provides insight into the element size right at that boundary. This size subsequently acts as a reference for sizing elements located further away from said boundary.

**Scale:** Ideally set between 10% and 20%, the scale parameter reveals how the element size evolves as you distance yourself from a boundary. For instance, a 10% scale implies that moving a meter away from the boundary results in an elemental size increase of 10 cm.

Another parameter to consider is the maximum elemental size, termed as `max`. This can either be left at its default infinite setting or adjusted to a more restrictive value for enhanced elemental size control.

The Scale Distance Function's forte is in delivering precise meshing control along designated boundaries. An example might involve a structure like an iron yoke with keys. Achieving intricate meshing around the key boundaries demands the replication of Distance Functions for these keys within the Scale Distance Function. This implies that both the

Distmesh Cross-Section and the Scale Distance Function house identical copies of the key Distance Function as child nodes, guaranteeing precision where it matters. This is shown in Figure 8.6.2, where a Union of the Angular Array Distance Functions for the Rectangle Distance Function and the Circle Distance Function are the child nodes of the Scale Distance Function, as well as of the Distmesh Cross-Section. Because this is an example, the `scale` parameter is set to 40 % to exaggerate the difference between the elements at the boundaries of the rectangles and circles and the elements everywhere else.

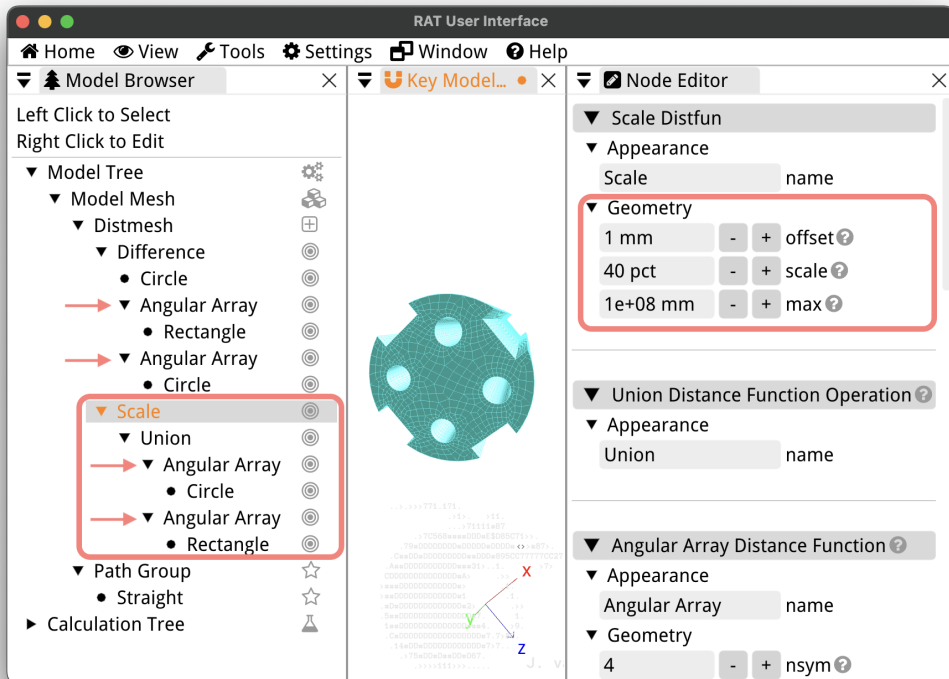


Figure 8.6.2: Illustrating the use of the Scale Distance Function for an iron yoke with keys. The Angular Arrays for the keys and holes are copied inside the Scale Distance Function. The `scale` parameter in the *node editor* is set to 40 % to exaggerate the difference between the elements at the boundaries of the rectangles and circles and the elements everywhere else.

However, a word of caution is warranted. When deploying the Scale Distance Function, it's paramount to designate the minimal anticipated elemental size to both the `offset` and the `h0` in the Distmesh Cross-Section (refer to Figure 8.2.2 in Section 8.2). This value, `h0`, symbolizes the starting size for elements. Due to the inherent mechanics of the Distmesh, the algorithm can only reduce the number of elements, not increase them.

In summation, while the Scale Distance Function furnishes intricate meshing control, its application is most apt for specialized scenarios. Users should be judicious, appraising its ramifications and their project requisites before deployment. Its utilization can be intricate, and in many scenarios, the default Ones Distance Function, resulting in a uniform mesh size, suffices for precise outcomes. Often, simply refining the mesh by decreasing the target elemental size `h0` in the Distmesh Cross-Section is adequate. Thus, it's advisable to first

adjust  $h_0$  before venturing into meshing endeavors with the Scale Distance Function.

# Chapter 9

## Cross-Sections

### 9.1 Introduction

The Rat GUI comes equipped with a range of predefined three-dimensional shapes for modeling Coils, Meshes, HB-Meshes, or Permanent Magnets, which offer an easy start for users. However, for each of these main objects (Coil, Mesh, HB-Mesh, and Permanent Magnet), there is also a “Custom” option available for users who need more flexibility or unique shapes. To utilize this, you add a Custom object of the desired type to the Model Tree, and then add a Cross-Section and Path of your choice to that object.

All Coils and Meshes in Rat are made by extruding a Cross-Section along a Path, as shown in Figure 9.1.1, which is a part of the Custom object illustration in Figure 4.2.1. If you use a pre-defined object like a Solenoid Coil for instance you can always use the Split Tool (**Ctrl+B**) to expose the internal Cross-Section and Path. If you are modeling your design with a Custom object you have full flexibility to choose any of the available Cross-Sections (and Paths) from the start.

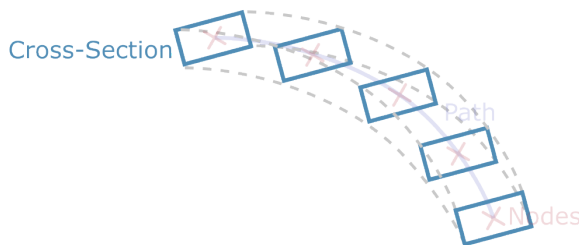


Figure 9.1.1: This illustration is one part of Figure 4.2.1. It is an illustration of how three-dimensional objects, marked by a grey dotted line, are created in Rat. The Cross-Section is always extruded along the Path.

This chapter delves into all the available Cross-Sections, summarized in Table 9.1.1. The first two Cross-Sections are mainly used for visualization purposes. The Rectangle and Circle Cross-Section objects are predefined shapes commonly used. The remaining two, Cross-Section Path and Distmesh, provide the flexibility to model any shape required, such as U-shaped or star-shaped cross-sections, for instance. Each of these Cross-Section objects is described individually in the subsequent sections of this chapter.

All Cross-Sections in Rat are defined within a two-dimensional local coordinate system.

Table 9.1.1: Cross-Section objects in the Rat GUI.

Name	Description	Section
Point	A zero-dimensional point-shaped cross-section, mainly for visualization purposes	Section 9.2
Line	A one-dimensional line cross-section, mainly for visualization purposes	Section 9.3
Path	A two-dimensional cross-section defined by a Path, to make tubular objects for instance	Section 9.4
Rectangle	A two-dimensional rectangular cross-section	Section 9.5
Circle	A two-dimensional circular cross-section	Section 9.6
Distmesh	A two-dimensional Distmesh cross-section defined by distance functions	Section 9.7 Chapter 8

This plane is constructed by two vectors: the first vector, known as the Normal vector,  $\vec{N}$ , and the second vector, called the Transverse vector, or  $\vec{D}$ . Generally, the origin of these local coordinates does not coincide with the origin of the global coordinate system of the Custom object you are modeling. However, with the appropriate transformations applied to your model, alignment can be achieved if necessary. For more information on the available transformations, please refer to Chapter 13.

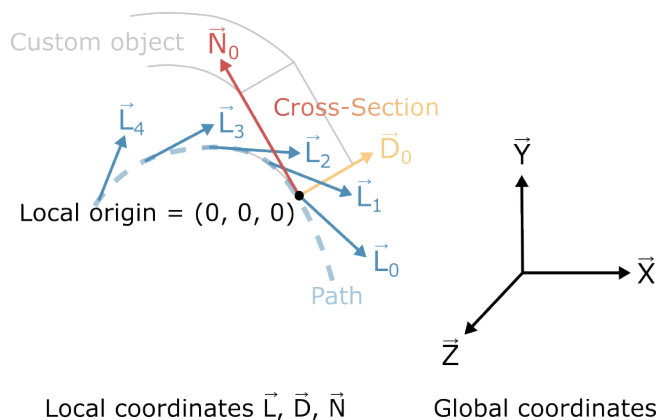


Figure 9.1.2: The local coordinates,  $(\vec{N}, \vec{D})$ , of the Cross-Section, which has a rectangular shape in this illustration. The local coordinates of a Path are denoted with the longitudinal vectors ( $\vec{L}$ ) at each node. The Path Cross-Sections will be extruded such that the Path goes through  $(\vec{N}, \vec{D}, \vec{L}) = (0, 0, 0)$ . In this illustration the Rectangle Cross-Section is not centered on the Path.

The Path always runs through the Cross-Section at  $(\vec{N}, \vec{D}) = (0, 0)$ . So if you want to define your Path in the center of a Rectangular Cross-Section for instance you need to set the coordinates for the bottom left corner of the rectangle to  $(-\frac{1}{2}w, -\frac{1}{2}w)$  and the coordinates of the top right corner of the rectangle to  $(\frac{1}{2}h, \frac{1}{2}h)$ . In that case,  $w$  is the width of the rectangle and  $h$  is its height.



## 9.2 Point Cross-Section

The Point Cross-Section is a one-dimensional point-shaped object that is mainly used for visualization purposes. With this Cross-Section you can easily visualize the center of the Path of your custom object for instance. However, in case you want to add current to your object because it is a Coil object for instance, an area needs to be assumed in order to define the current density along the coil, which can be set in the *node editor* of the Point Cross-Section.

There are four settings to define your Point Cross-Section in the Rat GUI, see Figure 9.2.1. The first two allow for specifying the position of the point within its local two-dimensional plane ( $(\vec{N}, \vec{D})$ , refer to the introduction of this chapter) by using two coordinates: the normal coordinate, *nc*, and the transverse coordinate, *dc*.

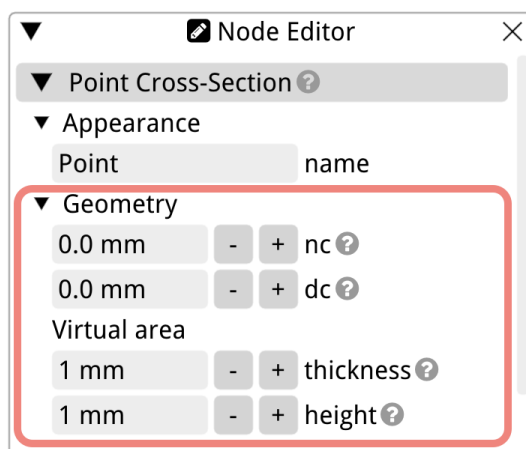


Figure 9.2.1: The *node editor* of the Point Cross-Section, with the four geometry parameters that define its placement in its local coordinate system, and virtual area in case the current density of a Coil is needed.

It makes sense to just keep these to their default  $(0,0)$  position, however, they can be used to offset the Cross-Section when you are trying out your Path's for an Edges Coil for instance, see Section 4.3. In that case you are using the Point Cross-Section just to make your Path shape visible. This is illustrated in .

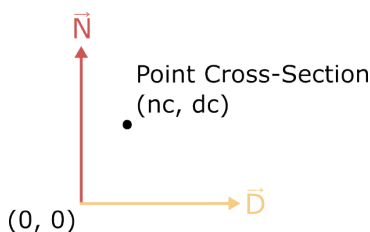


Figure 9.2.2: The local coordinate system of the Point Cross-Section. In this case, the Point is offset with respect to origin where the Path is located.

The other two settings are used in case this Cross-Section is part of a Coil with a current. In that case an area value is needed to calculate the current density and subsequently the

magnetic field produced by the Coil. In such cases, the **thickness** parameter is employed to specify the thickness of that area, and the **height** parameter is used for the height of that area. This Cross-Section has no discretization settings since it is a point.

### 9.3 Line Cross-Section

The Line Cross-Section in Rat is a one-dimensional object used to create an infinitely thin line-shaped cross-section for a Custom Rat object. It results in a Custom object that looks like a surface, primarily intended for visualization purposes. However, if you aim to use it as a Coil that carries current, a presumed cross-sectional area is necessary to define the current density. If you would like to model you Cross-Section as a line with a finite thickness, refer to Section 9.4, create a Path Group and add a Straight Section (Section 11.1.2).

The initial four settings in the *node editor* of the Line Cross-Section determine its start and end points in the two-dimensional local coordinates in which the cross-section is defined. Refer to the last paragraph of Section 9.1 for a detailed explanation of the local coordinates. The coordinates of the start point are represented as **nc1** and **dc1**, while those of the end point are denoted as **nc2** and **dc2**.

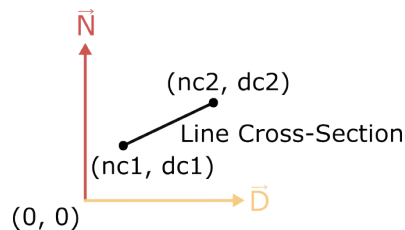


Figure 9.3.1: The local coordinate system of the Line Cross-Section. In this image, the Line is offset with respect to the origin where the Path is situated.

Since the line isn't zero-dimensional, a target element size, **delem**, can be defined. The number of nodes along the Line Cross-Section is determined by the Line's length divided by **delem**, plus one. Use the Toggle Mesh (Ctrl+M) function from the View menu (Section 3.7.2) to view the elements and their sizes. All these geometric parameters are displayed in the *node editor* as shown in Figure 9.3.2.

The final setting comes into play if this Cross-Section is part of a Coil with a current. In this scenario, an area value is required to compute the current density and, consequently, the magnetic field produced by the Coil. Here, the **width** parameter is employed to specify the width of that area. The length of the virtual two-dimensional cross-section is provided by the length of the Line.

Utilizing the Line Cross-Section in conjunction with a Path Group - Straight section (Section 11.1.2) can prove beneficial in forming a straight surface Mesh. If a Mesh Calculation (Section 15.6) is then performed on your model, the resulting *viewport* will showcase the magnetic field on a plane as well as the Coil Mesh. Such a plot isn't possible with a Plane Calculation (Section 15.5) as it generates a two-dimensional plot, whereas the Mesh Calculation provides a three-dimensional *viewport*. This concept is illustrated in Figure 9.3.3.

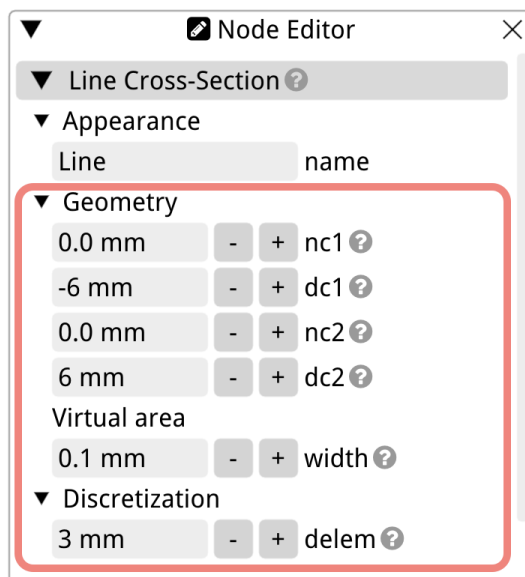
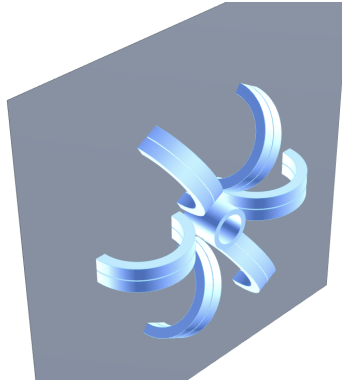
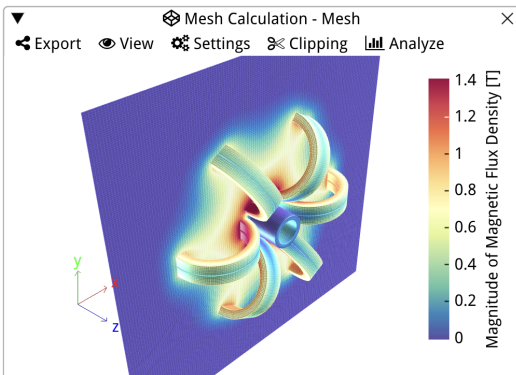


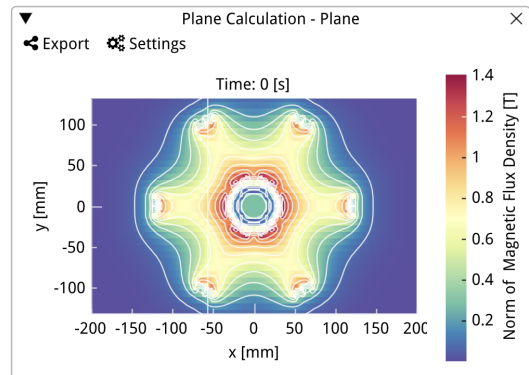
Figure 9.3.2: The *node editor* of the Line Cross-Section, featuring the five geometry parameters that define its placement in its local coordinate system and virtual area if the current density of a Coil is to be calculated.



(a) Surface Mesh (grey) with Line Cross-Section.



(b) Mesh Calculation.



(c) Plane Calculation.

Figure 9.3.3: This figure demonstrates one of the uses of the Line Cross-Section. In (a) a surface is meshes with a Line Cross-Section and a Path Group's Straight section. When you perform a Mesh Calculation on this model you also get the magnetic field on the plane, and your result is a three-dimensional plot that includes the Coil meshes, see (b). This can not be achieved with the Plane Calculation because it only outputs a two-dimensional plot.

## 9.4 Path Cross-Section

The Path Cross-Section in Rat represents a two-dimensional shape, delineated by a user-selected Path from the set of predefined Paths, as listed in Chapter 10. While a Path in Rat is innately infinitesimally thin, as a Cross-Section it can have a thickness. If the `is line` option in the *node editor* is unselected, the Path Cross-Section is attributed with a `thickness`, enabling the modeling of tubular objects, particularly when the path forms a closed loop. Otherwise the user can model an infinitely thin surface, like for the Line Cross-Section, by checking the `is line` box.

A feature of the Path Cross-Section is its ability to be offset relative to the originating Path. This offset functionality is schematically represented in the right half of Figure 9.4.1. It's noteworthy that the Path (not the Path Cross-Section meant in this section) associated with the Custom object invariably intersects with the origin of the local two-dimensional coordinate system.

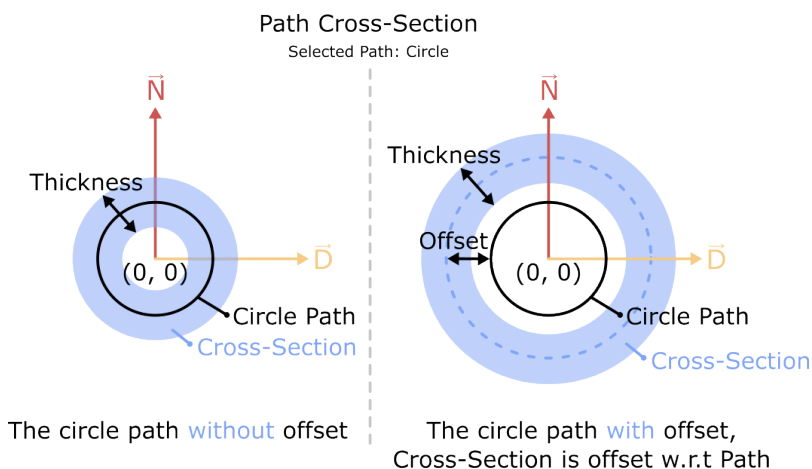


Figure 9.4.1: The local coordinate system of the Path Cross-Section, exemplified using a Circle Path. The Cross-Section's position can be adjusted relative to the chosen Path with the offset. Importantly, the Path of the Custom object being modeled consistently traverses the origin of this local coordinate system.

The *node editor* of the Path Cross-Section, shown in Figure 9.4.2, displays the `thickness` and the `offset` parameters. Another critical parameter, `delem`, determines the target element size of the Cross-Section. It's worth mentioning that the element size corresponding to the Path inside the Cross-Section can be independently defined in its exclusive *node editor*.

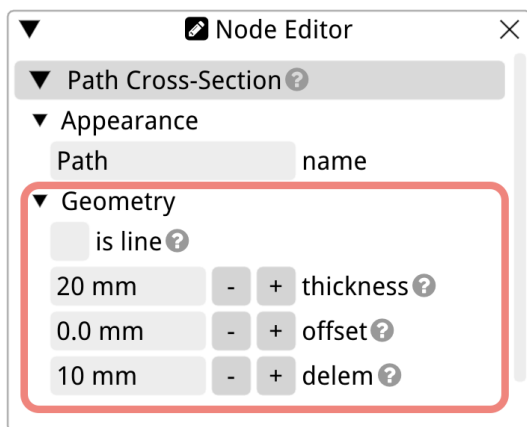


Figure 9.4.2: The *node editor* of the Path Cross-Section.

## 9.5 Rectangle Cross-Section

The Rectangle Cross-Section is commonly used in Rat's Coil Models (Chapter 4), as many magnets are wound with rectangularly shaped conductors or have a rectangular coil pack. The shape of the rectangle is defined by the coordinates of the bottom left and top right corners. This concept is illustrated in Figure 9.5.1.

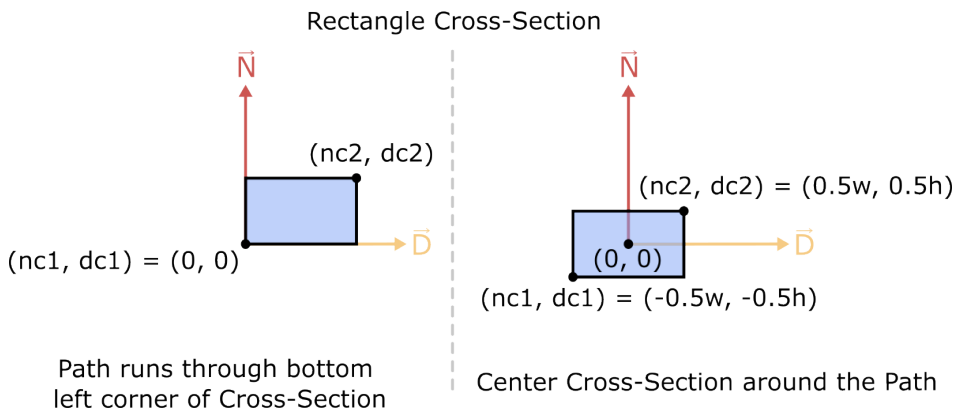


Figure 9.5.1: The local coordinate system of the Rectangle Cross-Section. On the left, the Path will run through the bottom left corner of the Cross-Section. On the right, the rectangle is centered on the origin where the Path is located.

The *node editor* of the Rectangle Cross-Section is displayed in Figure 9.5.2. The four coordinates in the geometry settings are used to position the rectangle in its local coordinate system. The coordinate of the bottom left corner of the rectangle is determined by **nc1** in the normal direction and **dc1** in the transverse direction. The normal and transverse coordinates of the top right corner are **nc2** and **dc2**, respectively.

When modeling with Custom objects in the Rat GUI, you have more flexibility regarding the element sizes in all directions. For the Rectangle Cross-Section, this flexibility is enabled by the **dnormal** and **dtrans** settings. Here, **dnormal** refers to the element size in the normal direction, and **dtrans** is the size of the element along the transverse direction. The

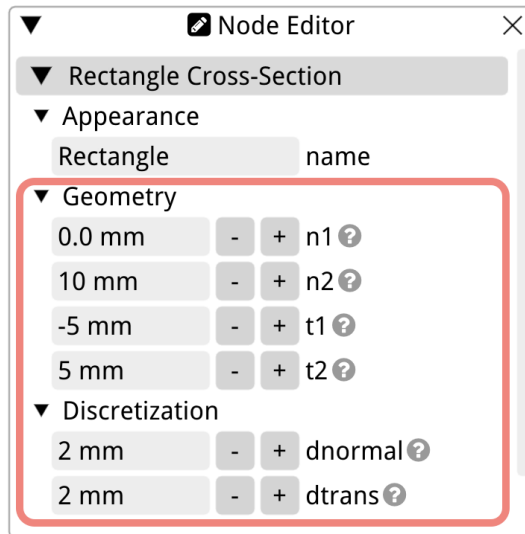


Figure 9.5.2: The *node editor* of the Rectangle Cross-Section, featuring the six geometry parameters that define its geometry in its local coordinate system and its element size.

longitudinal element size of the Custom object, whether it's a Coil or a Mesh, is set in the *node editor* of the Path chosen by the user. Use the Toggle Mesh (Ctrl+M) function from the View menu (Section 3.7.2) to view the elements and their sizes.

## 9.6 Circle

The Circle Cross-Section can be used when modelling a magnet with circular conductors, for instance. In Rat, it is defined by the coordinates of its center and its radius. This is shown in Figure 9.6.1, where  $(nc, dc)$  is the position of the center of the circle.

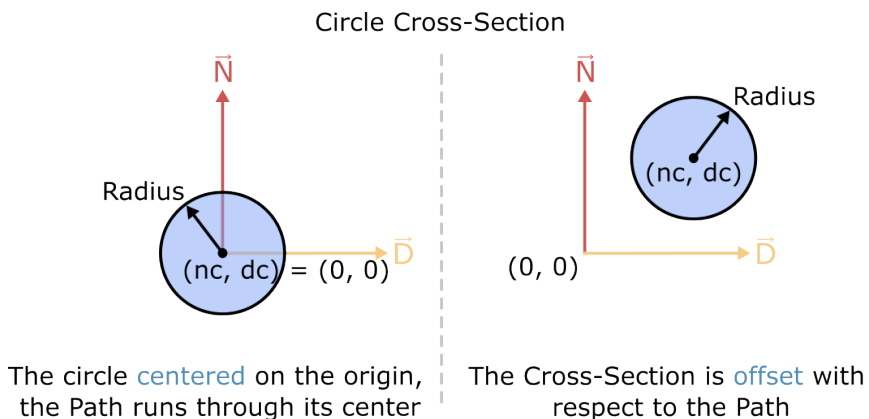


Figure 9.6.1: The local coordinate system of the Circle Cross-Section. On the left, the Path will run through the center of the Cross-Section. On the right, the rectangle is offset with respect to the Path.

Figure 9.6.2 shows the *node editor* of the Circle Cross-Section. Below the input fields for `nc` and `dc`, the user can set the radius of this shape. There is also one discretization setting, `delem`.

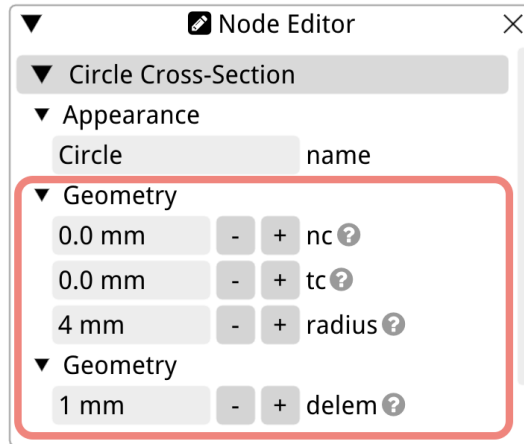


Figure 9.6.2: The *node editor* of the Circle Cross-Section, featuring the four geometry parameters that define its geometry in its local coordinate system and its element size.

To mesh the Circle Cross-Section, a multi-technique adaptive meshing approach was taken, where a rectangular orthogonal mesh is used in the center of the circle, overlaid with butterfly subdivision refinement for the circular outer parts. This technique is a blend of structured and unstructured meshing methods, which ensures optimal representation of the circle geometry while balancing computational efficiency. Therefore, you can only set one target element size, as the center of the circle comprises square elements. The user can set the target element size for the Circle Cross-Section with the `delem` setting in the *node editor*. The element size of a Custom object in the longitudinal direction is set in the *node editor* of the Path that is used to model it. Use the Toggle Mesh (Ctrl+M) function from the View menu (Section 3.7.2) to view the elements and their sizes.

## 9.7 Distmesh Cross-Section

The Distmesh Cross-Section stands out as a unique feature within Rat. It empowers users to create virtually any conceivable shape by leveraging and combining Distance Functions. Whether it's star-shaped forms, U-shapes, or cross-sections featuring holes, the Distmesh Cross-Section makes it feasible. However, its high degree of flexibility also means it's more intricate compared to the other Cross-Sections introduced earlier. As such, to provide in-depth guidance on its utilization, a dedicated Distmesh Chapter has been included in this manual, which can be referenced in Chapter 8.



# Chapter 10

## Paths

### 10.1 Circle

### 10.2 Polygon

### 10.3 Rectangle

### 10.4 Trapezoid

### 10.5 D-shape

### 10.6 Plasma

### 10.7 Flared

This is secretly a path group, you can split it?

### 10.8 Clover

strengths: so the first control point is at exactly  $L$  vector times that length away from start point. second cp is at  $L$  vector time second length away

## **10.9 Spiral**

### **10.10 Canted Cosine Theta Paths**

#### **10.10.1 Regular CCT**

#### **10.10.2 Custom CCT**

#### **10.10.3 Table CCT**

### **10.11 Axis**

### **10.12 XYZ File**

file keeps the like to the file, so you don't need to reload the file everytime you update it. In case you have the Gui open, and in the mean time you edit the file and save it (under the same name) then you can redraw the mesh in the gui with the space bar.

### **10.13 XYZ Table**

# Chapter 11

## Path Group

### 11.1 Group

Path group matches L vector of two path sections , so each section starts at the coordinate  $(0,0,0)$  and the path group sticks them together matching the L-vector at the interface.

- 11.1.1 Point
- 11.1.2 Straight
- 11.1.3 Arc
- 11.1.4 Super Ellipse
- 11.2 Attachment
- 11.3 Connect V1
- 11.4 Connect V2
- 11.5 Map
- 11.6 Frenet-Serret
- 11.7 Unfold
- 11.8 Offset
- 11.9 Local Offset
- 11.10 Cable
- 11.11 Stair Cable

# Chapter 12

## Groups

There are two types of groups: the first one acts on the whole model or on the coil level. The second type of group acts on the path level only. FIX GROUPS OF PATHS (NOT THE PATH GROUP).

### 12.1 Group

### 12.2 Array

### 12.3 Angular Array

### 12.4 Mirror

### 12.5 Path Array

### 12.6 Clip

# Chapter 13

## Transformations

13.1 Translation

13.2 Rotation

13.3 Reflection

13.4 Bending

13.5 Reverse

13.6 Flip

## Chapter 14

# Material properties

# Chapter 15

## Calculations

### 15.1 Introduction

A calculation in Rat takes a Model as input and produces data as output. In the Rat GUI, a calculation operates on the Model Tree and is organized within the Calculation Tree. The available calculation and their corresponding sections in this manual are summarized in Table 15.1.1.

Table 15.1.1: Calculation types in the Rat GUI.

Name	Description	Section
Point(s)	Field on one or more coordinates	Section 15.2
Line	Magnetic field along a line	Section 15.3
Path	Magnetic field along a (curved) path	Section 15.4
Plane	Magnetic field on a plane	Section 15.5
Mesh	Magnetic field on the surface of coil mesh(es)	Section 15.6
Cartesian Grid	Magnetic field in a three-dimensional box	Section 15.7
Polar Grid	Magnetic field in a three-dimensional box	Section 15.7
Length	Length of the path, typically conductor length	Section 15.13
Inductance	Inductance and mutual inductance of coils	Section 15.9
Cylindrical Harmonics	Cylindrical field harmonics for field quality of accelerator magnets	Section 15.10
Spherical Harmonics	Spherical field harmonics for field quality of NMR/MRI magnets	Section 15.11
Tracking	Calculate particle trajectories influenced by the magnetic field	Section 15.12

To add a Calculation to your model, right-click on the Calculation Tree, choose ‘Add Calculation,’ and select a Calculation from the list. Calculations can be executed using three different methods. First, the buttons ‘Calculate,’ ‘Calculate All,’ and ‘Show Results’ perform the calculations and display the results immediately in a pop-up window in the Rat GUI. Second, to perform the calculations and store the results as VTK files, use ‘Calc & Write’ and ‘Output VTK.’ These methods do not open a pop-up window with results. The same applies to the third option, ‘Keep Data in Memory,’ which starts the calculation without



writing the output to a file or visualizing it in the Rat GUI. To view the results later, use the ‘Results’ option in the Processes menu, accessible by right-clicking a process (see Section 3.6 for more on the *processor*). The progress of your calculation can be monitored in the logger: right-click on the process of your calculation and select ‘Show Log’.

### 15.1.1 General Calculation Settings

The calculations can be edited just like the objects in the Model Tree. When you select the Calculation Tree or a specific calculation within it, all its settings and parameters will show up in the *node editor*. These settings, along with the Calculation Tree itself, are automatically stored in the same file as a Rat Model.

Within the *node editor* of the Calculation Tree (refer to Figure 15.1.1), you can set a different **name** for the Calculation Tree, adjust the VTK settings for the entire tree, and specify a time for calculating the magnetic properties. This feature is particularly useful when the model includes a time-dependent variable. By setting the **time** parameter, you can perform the calculations specifically for a particular time instance.

The VTK settings are used if you intend to write the results to VTK files [15]. These files can be opened using ParaView [16] for further data processing. Each calculation can have separate VTK settings, or you can choose to apply the same VTK settings to all calculations by configuring them for the entire Calculation Tree (see Figure 15.1.1). The file location is specified using the **directory** parameter. If your model includes a time-dependent variable, you also have the option to store separate VTK files containing results at different time steps. The time range for storing results can be defined using the **t1** (start time) and **t2** (stop time) parameters, while the number of time steps to be stored can be set with **ntimes**.

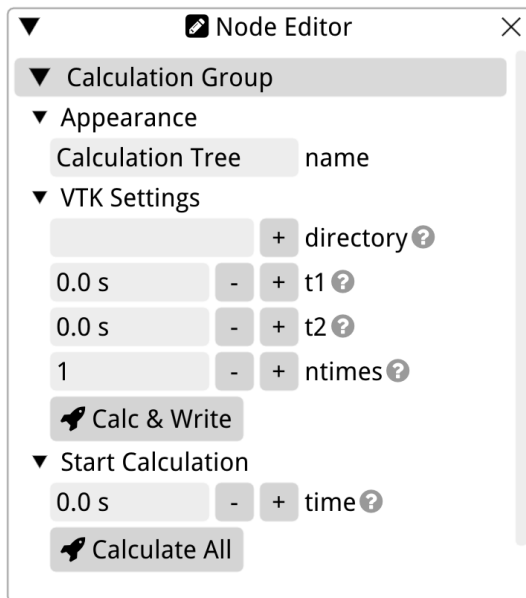


Figure 15.1.1: The settings of the Calculation Tree.

### 15.1.2 GPU Acceleration

GPU acceleration, which involves utilizing the graphics card to speed up calculations, can

be enabled for each calculation in the CUDA settings (refer to Figure 15.1.2 below). If you have one or more NVIDIA graphics cards, they will appear in the list. By checking the box next to the card's name, it will be utilized for calculating the results. CUDA settings can be configured individually for each calculation. If you have multiple graphics cards, you can assign different cards to different calculations. This allows you to run calculations simultaneously on different cards when initiating all calculations from the Calculation Tree. Refer to 'Calculate All' in Figure 15.1.1.

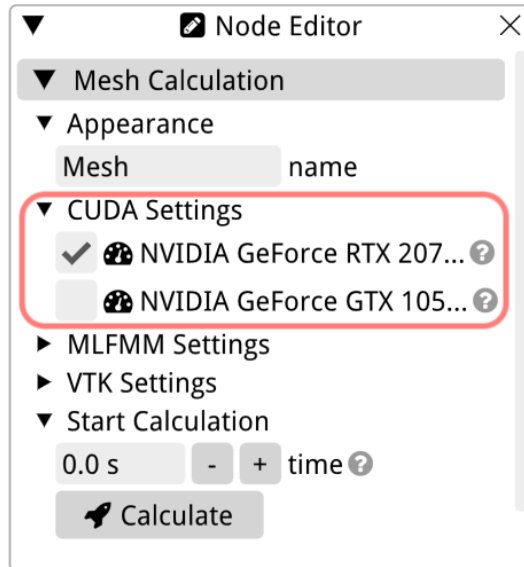


Figure 15.1.2: The GPU Acceleration settings in the *node editor*.

### 15.1.3 The Multi-Level Fast Multipole Method

Except for the Length Calculation, all calculations in RAT use the Multi-Level Fast Multipole Method (MLFMM) [17]. Most MLFMM settings are hidden in the Rat GUI, and the accessible settings in the GUI generally do not need to be changed. However, there are cases where you may want to modify MLFMM settings, and those settings are explained in the next paragraph. For more information on the Multi-Level Fast Multipole Method, please refer to the following references: [17, 18].

The first checkbox in the MLFMM settings turns off MLFMM and uses the direct Biot-Savart method, see Figure 15.1.3. This changes the complexity of the computations from  $\mathcal{O}(N + M)$  to  $\mathcal{O}(N \cdot M)$  because the direct Biot-Savart calculates the field from each source  $n$  on each target  $m$ , which is not the case for MLFMM. Here,  $N$  is the total number of sources and  $M$  is the total number of targets. The second checkbox allows for using the **extended interaction list** in the Multipole to Localpole step. It slightly improves the accuracy of MLFMM, but in most cases, it is not necessary. This option is checked by default when maximum accuracy is required, such as in harmonics calculations. When you model includes non-linear materials it is advised to always check this box due to the sharp transition sharp transition of  $\mu_r$  at the edge of non-linear materials. For more information consult Section 6.1.

The **nexp** parameter sets the number of expansions of the Legendre polynomials that

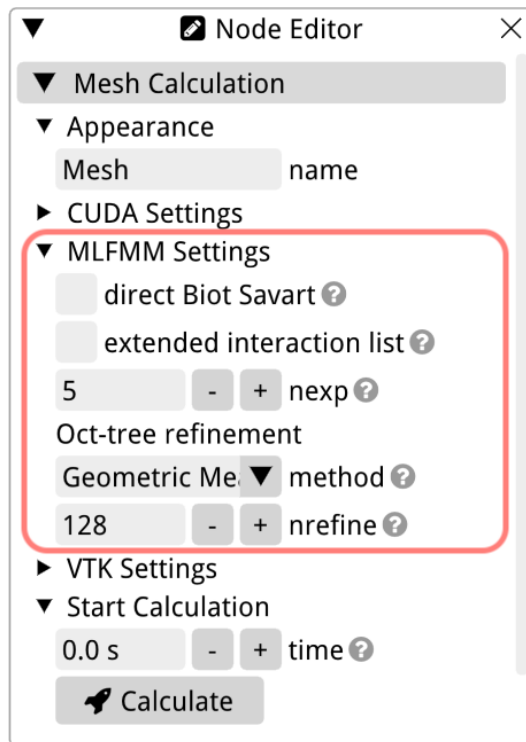


Figure 15.1.3: The MLFMM settings in the *node editor*.

describe the multipoles. A higher value of `nexp` results in more accurate results but also increases the computation time. Below `nexp`, you can set the refinement `refine method` and the refinement target `nrefine`.

During the oct-tree setup step, the number of levels in the tree is increased until the average number of sources and targets inside each box falls below `nrefine`. The setting here determines how to combine the average values. The following options are available:

1. both: both average number of sources and the average number of targets must fall below the refinement target;
2. either: or the average number of sources or the average number of targets must fall below the refinement target;
3. average: the average number of particles weighted by the number of source and target boxes must fall below the refinement target;
4. average2: the average number of particles without weighting must fall below the refinement target;
5. times: the average number of sources times the average number of targets, i.e., the average number of interactions during S2T step, must fall below the refinement target;
6. geometric mean: the geometric mean of the average number of sources and average number of targets must fall below the refinement target.

The parameter `nrefine` is the target number of sources and targets in the oct-tree boxes. The algorithm stops dividing the parent boxes into smaller ones depending on the refinement method and the value for `nrefine`. Adjusting this setting can be useful, for example, when you notice that the S2T step in MLFMM is taking too long. In such cases, you can lower `nrefine`. Conversely, if the M2L step is slow, you can increase this setting. To find the duration of each step, consult the logger of your calculation, as detailed in Section 3.6.

Note that, when using GPU (CUDA) instead of CPU, the refinement target can be increased as GPU's are highly efficient at direct S2T calculations. Also note that the grid size can never be smaller than the element size (when using elements). If you are modeling ferrites or other non-linear magnetic materials, it is important to pay special attention to the `nexp` and `nrefine` parameters. Further details on this topic can be found in Chapter 7.

### 15.1.4 Editing Calculations from the Model Browser

You can edit the Calculation Tree in the same manner as you would edit the Model Tree, refer to Section 3.4.1. As always, right-click the item you want to edit and select the appropriate option from the menu that appears.

When you right click an Calculation in the Tree to edit it there are two options that are only applicable to Calculations, namely the 'Add Background' and 'Add Circuit' options. See the appropriate subsections below. The Tracking Calculation has to additional dedicated options, which are explained in Section 15.12.

#### Add Circuit

#### Add Background

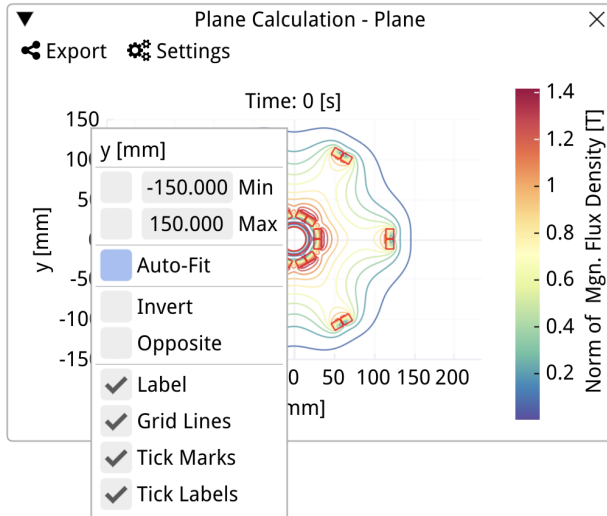
### 15.1.5 Post-Processing

When a calculation is completed, you can view the results in Rat GUI. If you started the calculation using the 'Calculate,' 'Calculate All,' or 'Show Results' options, the results will

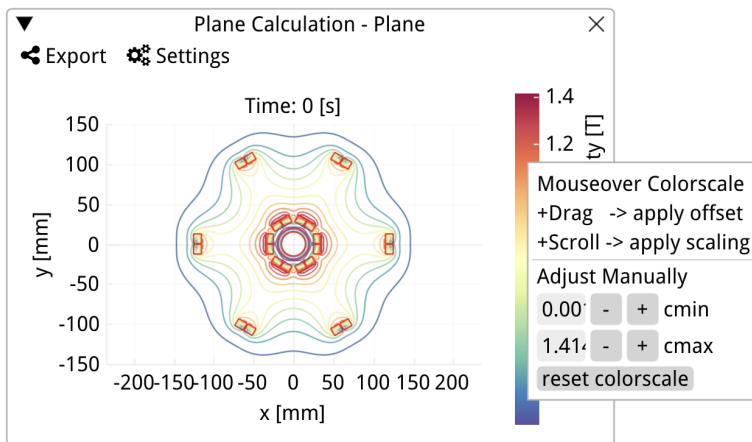
be automatically displayed in a pop-up window. Alternatively, you can analyze the results by right-clicking on the calculation in the *processor* and selecting ‘Result’ (refer to Section 3.6). Each calculation has its dedicated *post-processor* window that presents the results. These windows offer various settings and plotting options specific to each calculation, which are discussed in the relevant sections below.

Some of the *post-processors* feature two- or three-dimensional graphs. The three-dimensional image can be inspected from all angles with the mouse, just like in the *viewport*, see Section 3.3. In two-dimensional plots, you can zoom in or out of the results by scrolling while your pointer is positioned on the graph. The location of your mouse determines the area of the image that is zoomed, similar to the zooming-behavior in the *viewport*. This zooming behavior also applies to graph axes and the colorbars.

By right-clicking on the axes or colorbar, you can access the settings specific to that axis or bar, as illustrated in Figure 15.1.4. These settings allow you to customize the plotting range and change the appearance of the axes. If the plot includes a colorbar, you can drag it by right-clicking and holding your mouse button while moving the mouse up and down. This action modifies the range of values displayed on the colorbar and in the plot. Additionally, you can double-click on the axis or colorbar with the right mouse button to reset it back to the default settings.



(a) Axis settings.

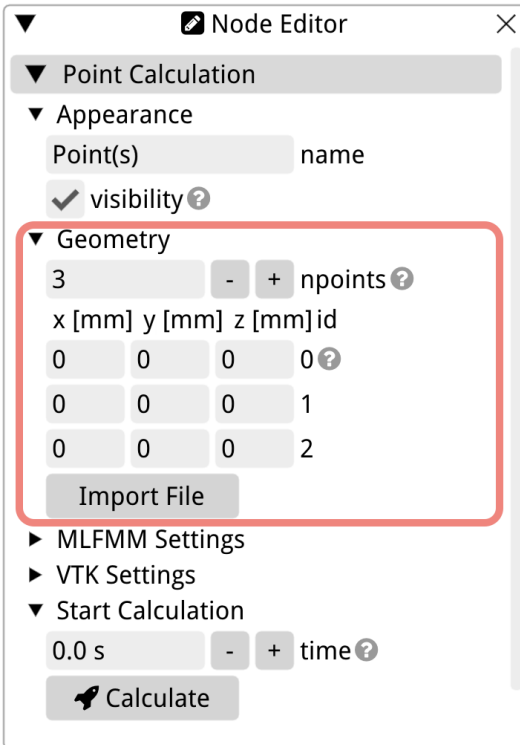


(b) Colorbar settings.

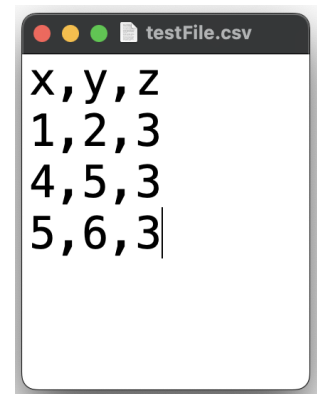
Figure 15.1.4: The settings for axes and colorbars in *post-processors*. These settings can be accessed by right-clicking on an axis or colorbar.

## 15.2 Point(s)

The Point(s) Calculation is used to calculate the magnetic field at a specific point or a list of points in your model space. Each point is defined by its Cartesian coordinates. Use the `npoints` parameter in the geometry settings to specify the number of points for which you want to calculate the magnetic field. Then, set the coordinates for each point in the table below `npoints`. For each component ( $x$ ,  $y$ , and  $z$ ), there is a separate line with three input fields. If you wish to visualize the points in the model, you can enable the visibility checkbox below the calculation's name (and above `npoints`). This visibility setting does not affect the calculation itself.



(a) Settings.



(b) Coordinates file.

Figure 15.2.1: The settings for the Point Calculation on the left and an example file with coordinates on the right.

If you have a large list of points, you also have the option to import them from a file. The file should have three comma-separated columns with a header. The header must be the following line: ' $x,y,z$ '. The file extension can be either `.csv`, `.txt`, or `.xyz`. An example of a file that can be imported for a Point Calculation is shown in Figure 15.2.1b. Click on the 'Import File' button to import your own files.

Below the button for importing files with points, you will find the MLFMM settings and the VTK settings, explained in Section 15.1.3 and Section 15.1.1, respectively. At the bottom of the Point Calculation settings, you can initiate the calculation by clicking on the 'Calculate' button. Like all calculations, if your model includes a time-dependent parameter, you can adjust the `time` setting to select a specific moment in time for the calculation.

The *post-processor* of the Point Calculation displays the results in a table, as shown in Figure 15.2.2. The table consists of several columns: the first column lists the *id* of each point, followed by three columns for the *x*, *y*, and *z* coordinates of each point. The last four columns present the results for the magnetic flux density  $B_x$ ,  $B_y$ ,  $B_z$ , and its magnitude  $\|\vec{B}\|$ .

The screenshot shows a window titled "Point Calculation - Point(s) Calculation" with a close button (X) in the top right. Below the title bar are two buttons: "Export" (with a share icon) and "Settings" (with a gear icon). The main content is a table titled "Mgn. Flux Density [T]". The table has seven columns: "id", "x [mm]", "y [mm]", "z [mm]", "Bx [T]", "By [T]", and "Bz [T]". There are three rows of data, with the second row highlighted in grey.

id	x [mm]	y [mm]	z [mm]	Bx [T]	By [T]	Bz [T]
0	1.000	2.000	5.000	0.000	0.000	0.287
1	8.000	4.000	2.000	0.001	-0.000	0.289
2	9.000	9.000	9.000	-0.001	-0.001	0.289

Figure 15.2.2: Results of the Point Calculation displayed in the *post-processor*.

The Point Calculation *post-processor* menu provides options for exporting data and adjusting the *post-processor* settings. You can export the data to a ‘VTK Unstructured file,’ which can be analyzed using ParaView, as well as a comma-separated-value (CSV) file. The settings allow you to display different field types, change the number representation in the table to either floating-point or scientific notation, and adjust the precision to control the number of decimals displayed in the table.

The available field types in the Point Calculation *post-processor* include the Vector Potential in Vs/m, the Magnetic Field in A/m, the Magnetic Flux Density in T, and the Magnetization in A/m. You can customize the units for these fields in the GUI’s settings located in the *navigation bar*, as described in Section 3.7.4. Both the field type and the unit are provided in the header of the table, as well as in the header of each column.

## 15.3 Line

The Line Calculation is used to calculate the magnetic field along a straight line in your model space. This line is defined by setting the start and end coordinates. Both coordinates are Cartesian and can be found in the geometry settings of the Line Calculation. For both the start and end coordinate there is a separate field for the *x*, *y*, and *z* components.

Below the coordinate settings, you can specify the number of nodes along the line using **nnodes**. The magnetic field will be calculated at these nodes, which correspond to the targets of the magnetic field calculation. The number of nodes can be defined in the discretization settings.

The Line Calculation has its own individual settings for MLFMM and VTK, which are explained in Section 15.1.3 and Section 15.1.1 respectively. To initiate the calculation, you can either click the ‘Calculate’ button at the bottom of the Line Calculation settings or right-click on the calculation in the Calculation Tree and select one of the three options in the Calculate menu.

The *post-processor* of the Line Calculation displays the results in a two-dimensional graph. By default, the graph shows the three components of the magnetic flux density (T) as a



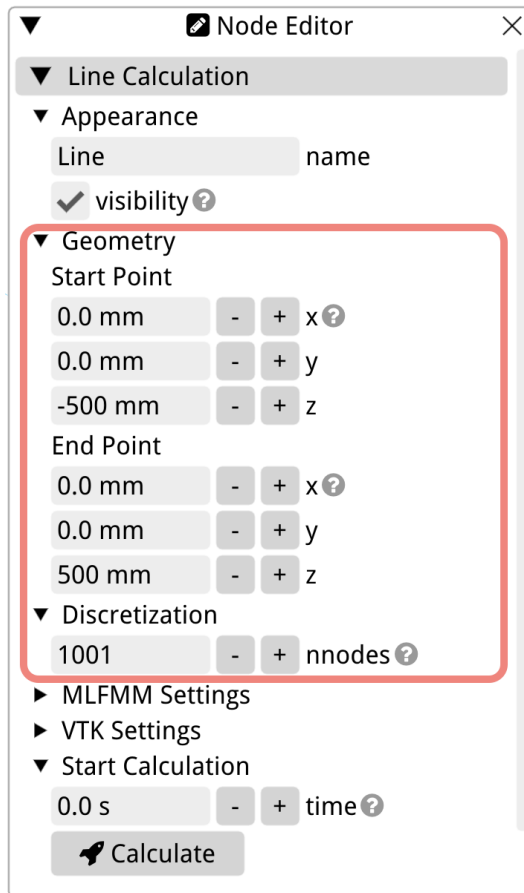
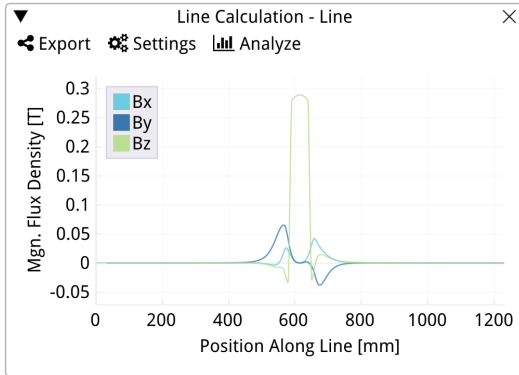
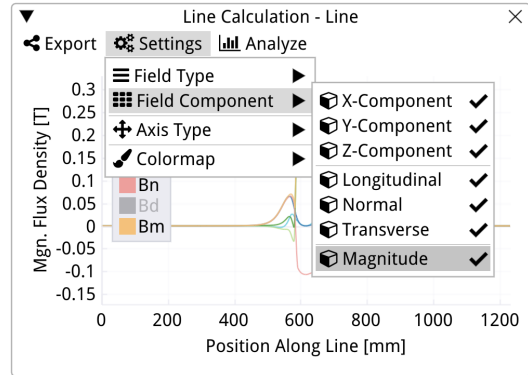


Figure 15.3.1: The settings of the Line Calculation.

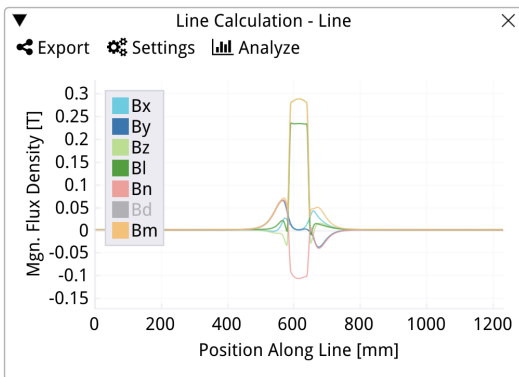
function of the position along the line, as shown in Figure 15.3.2. The results can be exported to a VTK Table, a VTK Unstructured, or a CSV Text File. In the settings tab, you can choose the field type to display in the graph, select the field components, choose the horizontal axis type, and adjust the color map.



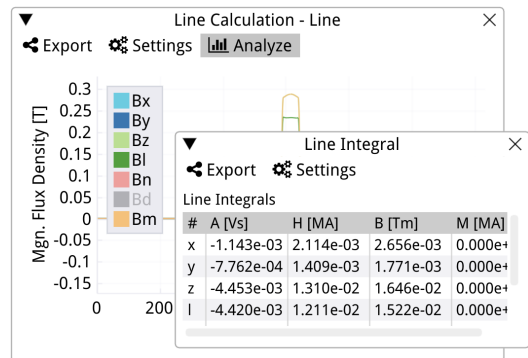
(a) Default settings.



(b) Custom settings.



(c) Field component selection.



(d) Line integral.

Figure 15.3.2: The *post-processor* of the Line Calculation, showing the default result on the top left, how to plot more field components on the top right, the customized *post-processor* with all field components on the bottom left ( $B_d$  disabled in legend), and the results of the Line Integral Analysis on the bottom right.

Just like in all other magnetic field calculations in the Rat GUI, you can plot the vector potential, magnetic field, flux density, and magnetization in the graph by selecting the field type in the settings. For each of these field types, you can plot the  $x$ ,  $y$ , and  $z$  components, as well as the longitudinal ( $l$ ), normal ( $n$ ), and transverse ( $d$ ) components, and the magnitude ( $m$ ). This can be done using the Field Component setting. You can enable or disable specific graphs in the plot by clicking on the corresponding label in the plot's legend. Refer to Figure 15.3.2b where the  $B_y$  component is disabled, and therefore greyed-out, in the legend.

The horizontal axis in the Line Calculation *post-processor* can be customized in the settings. By default, it displays the position along the line, from the start point to the end point. You can also choose to display the results as a function of the line position from the end

point to the start point by selecting ‘Axis Type > Position Along Line Rev.’ Additionally, you can set the origin of the horizontal axis to match the middle of the line by choosing ‘Axis Type > Position Along Line Mid.’ In the same group of settings, you can change the Axis Type to plot the field as a function of the line’s  $x$ ,  $y$ , or  $z$  components. You can also customize the colors

The last option in the Line Calculation *post-processor*’s menu is Analyze. This initiates a trapezoidal integration of each field component along the length of the line. For example, the line integral of  $B_x$  is defined as:

$$I_{B_x} = \int_L B_x \cdot dL. \tag{15.3.1}$$

Here,  $dl$  represents the line element size that can be calculated by dividing the total length of the line by **nnodes** (refer to Figure 15.3.1). After the calculation is finished, the results of these integrals are displayed in a table. The columns in the table correspond to the vector potential  $A$ , magnetic field  $H$ , flux density  $B$ , and magnetization  $M$  from left to right. Each row represents a different field component ( $x$ ,  $y$ ,  $z$ ,  $l$ ,  $n$ ,  $d$ ,  $m$ ).

To customize the displayed field types, you can right-click on the table header and uncheck the fields that should not be displayed. The settings of the table also allow you to change the precision and notation of the values.

## 15.4 Path

This calculation is used to determine the magnetic field along a user-defined path. Users can choose from various available path types in Rat, as described in Chapter 10. By default, the Path Calculation is created with an Axis Path. To view the Path on which the magnetic field will be calculated, unfold the Path Calculation in the Calculation Tree by clicking the triangle on the same line. Refer to Figure 15.4.1 for an example.

The *node editor* for the Path Calculation is divided into two sections, as is shown in Figure 15.4.2. The first section is for configuring the Path Calculation itself, while the second section is for configuring the specific Path within the Path Calculation. This structure is consistent with a Coil’s Path, where each tree level has its own *node editor* section. The Path Calculation *node editor* provides standard settings such as **name**, **visibility**, MLFMM Settings, and VTK settings. The *node editor* for the available path options are described in detail in Chapter 10. The calculation results are displayed in the same *post-processor* window as the Line Calculation *post-processor*, as explained in Section 15.3.

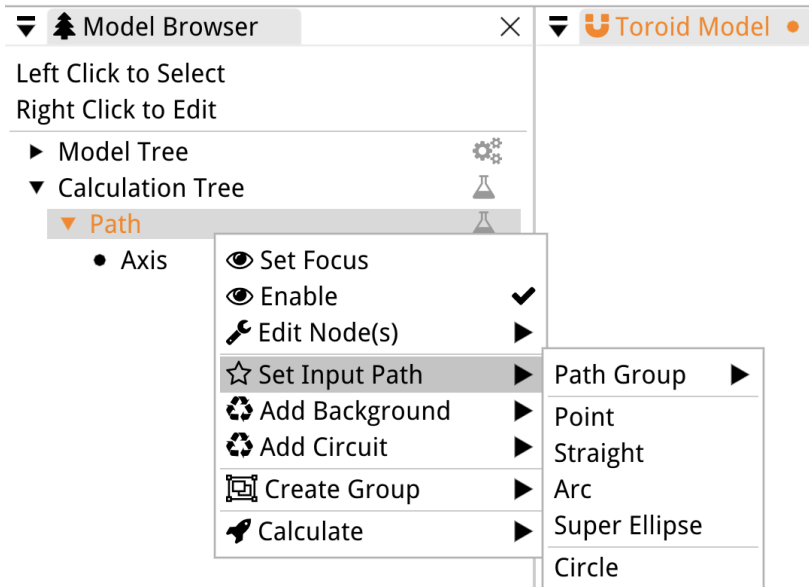


Figure 15.4.1: Set the input path for the Path Calculation by right-clicking on the Path Calculation in the Calculation Tree.

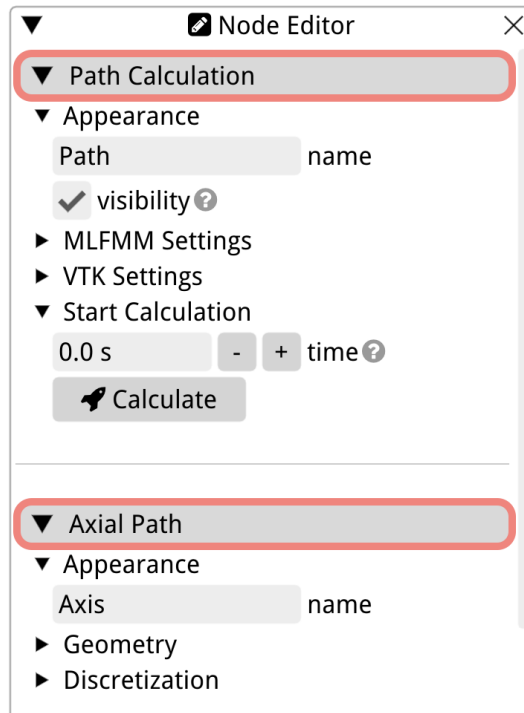


Figure 15.4.2: The settings of the Path Calculation.

## 15.5 Plane

The Plane Calculation is used to calculate the magnetic field on a plane in the model space. The plane is defined by its normal vector direction, size, and the offset of the plane's center with respect to the origin of the model. The plane set by the user is shown in the *viewport* as its outline when the *visibility* setting is checked in the *node editor*.

The normal vector direction (*normal*) can be chosen in the geometry settings of the Plane Calculation by selecting either *x*, *y*, or *z*. Right below that, you can set the length of the plane's two sides with *length1* and *length2*. The offset with respect to the plane's center in each dimension can be set individually by specifying the offset for *x*, *y*, and *z* respectively under the 'Offset' section.

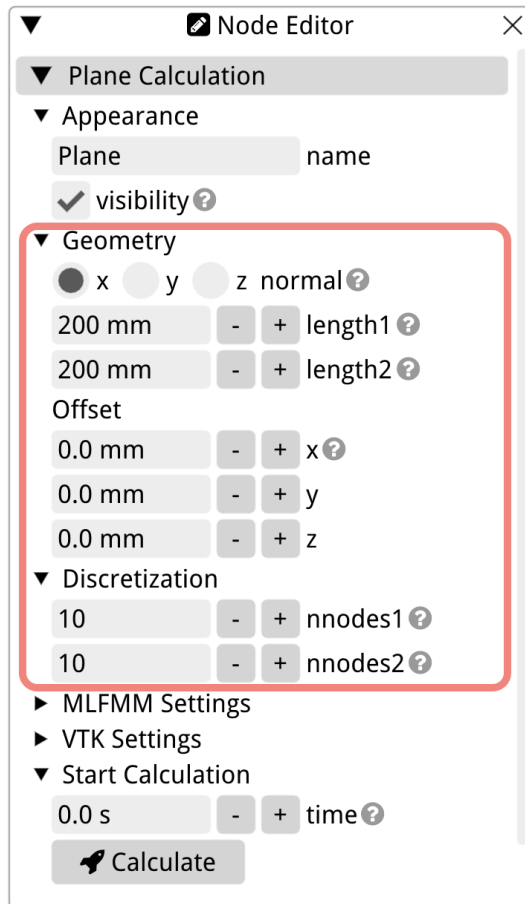


Figure 15.5.1: Plane Calculation settings.

The total number of nodes on the plane, which corresponds to the targets of this magnetic field calculation, is defined by the number of nodes in one dimension multiplied by the number of nodes in the other dimension. You can set the number of nodes in each dimension individually using *nnodes1* (for *length1*) and *nnodes2* under the discretization settings. All other settings for this calculation are explained in the introduction of this chapter, Section 15.1.

The results of the Plane Calculation are shown in a two-dimensional contour plot, and when coil(s) cross the plane in the model, they are shown as an outline in the plot as well, refer to Figure 15.5.2. When you mouse over the plot, the values are shown at the bottom left as ‘ $x$ -coordinate,  $y$ -coordinate,  $B$ , [T].’ The data can be exported to CSV, Drawing Exchange Format (DFX), and VTK unstructured file formats.

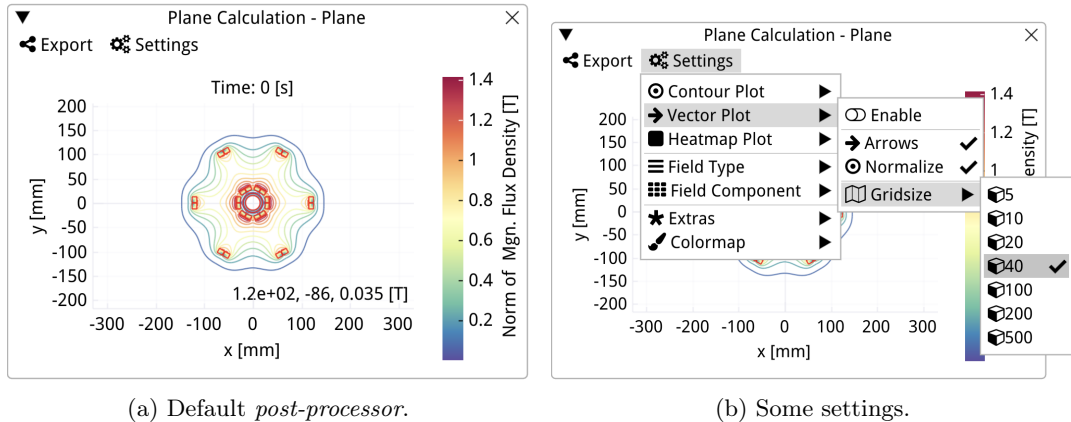


Figure 15.5.2: The *post-processor* of the Plane Calculation, showing the default result on the left and the settings on the right.

The settings of the *post-processor* window can be used to visualize the data in different forms: a contour plot (default), a vector plot, or a heatmap. These three options can be combined, and they are enabled with the ‘Enable’ option in each of the respective plot option sub-settings.

When the contour plot is enabled, the user can set the number of iso-surfaces by selecting them from the list in ‘Contour Plot > Number of Iso-Surfaces.’ In the same contour plot settings, the user can also select whether to draw the 5 G contour, the 30 G contour, and the 1 ppm contour. A check mark to the right of the setting indicates whether an option is active or not.

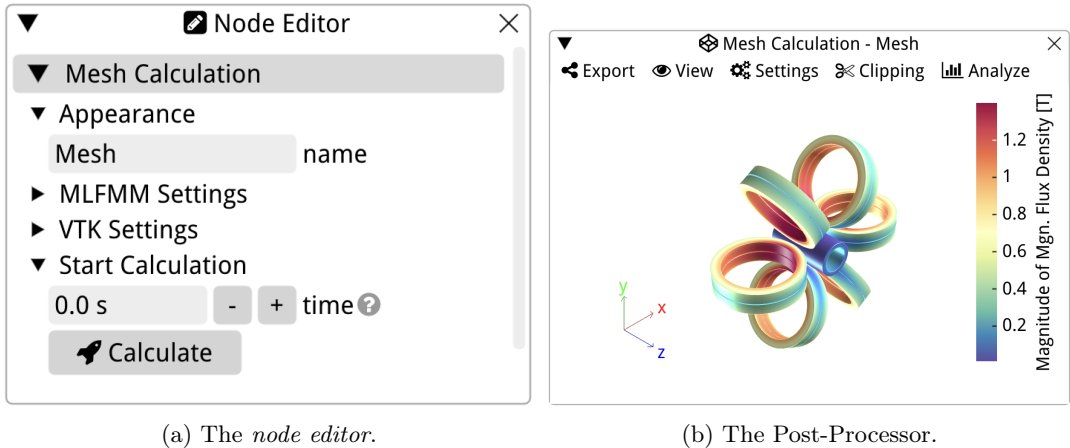
The vector plot draws lines or arrows to indicate the strength of the field. In the sub-settings, you can choose whether the vectors are displayed as arrows (check mark visible) or as lines (no check mark). The ‘Normalize’ option normalizes the vectors in the plot such that they are all the same length. The last sub-setting is the grid size, which determines the number of arrows drawn: the higher the grid size, the more arrows are drawn.

When the heatmap is enabled, the entire graph is colored based on the field strength. If the heatmap is used in combination with either of the two other plotting options, the contours and/or vectors will be drawn on top of the heatmap in white to remain visible. Select ‘Heatmap Plot > Enable’ to use it.

For this magnetic field calculation, the user can also select the field type and field component in the settings of the *post-processor*. Below the field settings, you can find some extra draw options. The first extra option, Coil Cross-Section, draws the coil cross-sections in the plot on top of the data if they cut across the plane. The other two extra options draw a blue dot at the location with the lowest field value (Indicate Min) and a red dot at the location of the highest field value (Indicate Max). Lastly, the user can also select a different color map in the *post-processor* settings. When the heatmap is disabled, the color map will be used to color the contour lines and/or vectors in the plot.

## 15.6 Mesh Calculation

The Mesh Calculation module is utilized to compute the magnetic field on the surface of the coils and meshes in your model. It takes all the coils and meshes in your model as input. In the *node editor* of the Mesh Calculation, you will find the standard calculation settings: name, MLFMM settings, and VTK settings. For a detailed explanation of these settings, please refer to Section 15.1.



(a) The *node editor*.

(b) The Post-Processor.

Figure 15.6.1: The *node editor* (left) and the *post-processor* (right) of the Mesh Calculation.

The output of the Mesh Calculation is visualized in a *post-processor* window, displaying the three-dimensional mesh of your model with colors representing the value of the plotted (physical) property. By default, the magnitude of the magnetic flux density,  $||\vec{B}||$ , is shown. You can modify this property by accessing the ‘Settings > Physical Property’ option. Detailed explanations of each property available for plotting on your mesh surfaces can be found in Section 15.6.1. If you have specified the material properties of your coils (refer to Chapter 14), additional data such as critical current fraction, thermal margin, loadline fraction, and more will also be available for plotting. The Mesh Calculation data can be exported from the *post-processor* to a CSV file, a CSV force density table, and a VTK unstructured file using the Export menu located at the top left corner.

Since the Mesh Calculation *post-processor* also functions as a *viewport* (refer to Section 3.3), you can control the view of your meshes using the mouse, similar to the main *viewport* of the Rat GUI that shows model itself. The View menu in the *post-processor* provides options to change the view. In addition to the standard view options found in the View menu of the Rat GUI (see Section 3.7.2), there are two additional options available in this *post-processor*. These options allow you to toggle the display of three axes that indicate the orientation of the meshes and to toggle the colorbar.

The Clipping menu enables you to create a cross-section of the results along a user-selected plane. You can apply the clip by enabling clipping with the checkbox at the top of the menu, and then define the location of the clipping plane using the other options in that menu. The clipping plane is defined by its normal vector direction, which is the vector perpendicular to the plane itself. To quickly set the normal vector of the plane, you can click the X-Normal, Y-Normal, or Z-Normal buttons. Alternatively, you can manually set the vector direction by entering the values of the  $x$ ,  $y$ , and  $z$  components in the corresponding boxes. The plane can be offset from the origin using the *offset* setting. By default, the program clips in the same

direction as the vector that was set. However, you can invert the clipping side by using the **invert** checkbox at the end of this menu.

In the last menu of this *post-processor*, you will find options for further analysis of the data. The ‘Peak Values’ table displays the minimum value’s position ( $[r_x, r_y, r_z]$ ), the lowest value ( $v_{\min}$ ), the maximum value’s position ( $[R_x, R_y, R_z]$ ), and the highest value ( $v_{\max}$ ) for each coil in the model. The row with the highest value is highlighted in green, while the row with the lowest value is highlighted in yellow. By selecting the ‘Analyze > Integrated Forces’ option, a table is displayed showing the integrated forces of each coil in the model. The table presents the  $F_x$ ,  $F_y$ , and  $F_z$  components of the integrated force vector, followed by its magnitude, with each coil on a separate row. Both the peak values and the integrated force densities can be displayed in floating or scientific notation with a custom precision by adjusting the settings of the tables. The data in the tables can be exported to CSV format using the Export option.

The last option in the analyze menu is a histogram that plots the volume fraction of the physical property displayed in the Mesh Calculation *post-processor*. By default, the histogram shows the distribution for all coils in the data set. However, you can use the settings menu to select the specific mesh for which you want to view the histogram. The histogram data can be exported to a CSV file for further analysis or documentation.

### 15.6.1 Mesh Calculation Post-Processor Settings

The Mesh Calculation *post-processor* provides several settings to change the visualization Mesh Calculation’s data. These settings allow you to change the physical property being plotted on the mesh surface, select the component to display if the property is a field or vector, and modify the colormap used for visualization. Below, each of the physical properties are explained in detail.

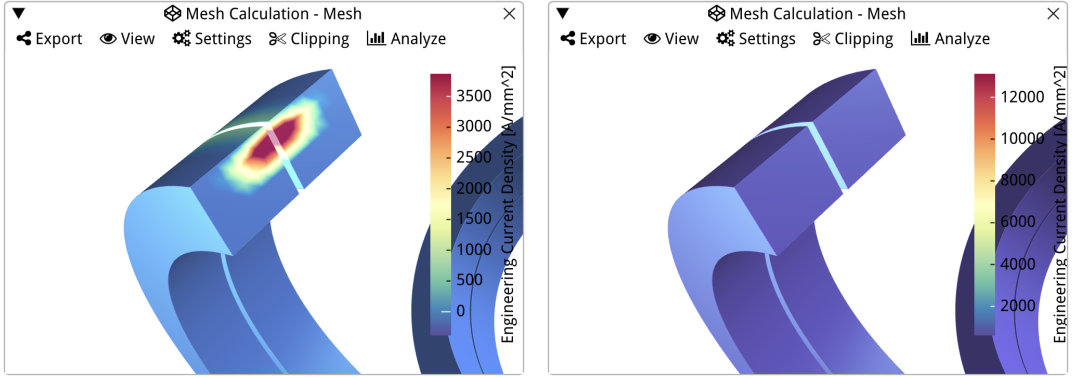
#### Enable Current Sharing

The first item in the list of physical properties is the option to enable current sharing. This setting is particularly relevant when modeling a cable consisting of superconducting wires or tapes, where the current can redistribute due to the absence of insulation between individual wires or tapes. By enabling current sharing, the software accounts for this redistribution phenomenon.

In Rat, the magnetic field is calculated assuming a homogeneous current density in the coil pack. Based on the resulting magnetic field, the critical current is then calculated for each element in the model. Since the magnetic field strength varies across the coil, the critical current also varies accordingly. Without current sharing, the engineering current density is plotted, as shown in Figure 15.6.2a.

Let’s consider a scenario where you are modeling a cable composed of High-Temperature Superconducting (HTS) tapes that are soldered together, allowing current redistribution within and between the tapes. In this case, the coil will not quench when the operating current  $I$  equals the lowest critical current within the coil pack. Instead, the current will first redistribute, and the quench will occur only when the operating current reaches the critical current across the entire coil pack. This can be modeled in the Rat GUI when you enable the current sharing, and it is shown in Figure 15.6.2b. Note that to utilize this option, you need to model the coil as individual turns, for instance by using a Cable Path (refer to Section 11.10), where a turn comprises all tapes or wires that are not electrically insulated from each other.





(a) No current sharing.

(b) Current sharing enabled.

Figure 15.6.2: An example where the engineering current density is plotted in a mesh calculation to demonstrate the effect of using ‘Enable Current Sharing’ in the *post-processor* settings.

When current sharing is enabled in the Mesh Calculation *post-processor* settings, the critical current density is calculated using the following equation:

$$J_{c,cs} = \frac{1}{A} \iint_A J_e(T, B, \alpha) dA, \quad (15.6.1)$$

where  $J_{c,cs}$  is the critical current density with current sharing enabled,  $A$  is the cross-sectional area of the turn,  $J_e$  represents the engineering current density,  $T$  denotes the temperature of the coil,  $B$  corresponds to the magnetic field, and  $\alpha$  represents the magnetic field angle.

### Magnetic and electric fields

Below the current sharing option, the user can choose to plot different types of magnetic fields and the electric field. Similar to other calculations in Rat, you can plot the vector potential, the magnetic field, the magnetic flux density, and the magnetization. Please note that magnetization is only valid for ferrites.

When one of the field types is selected, you can also choose to plot individual field components using the ‘Field Component’ settings. These settings are explained in more detail in Section 15.6.2.

The electric field is only valid when the coils or meshes have material properties because it depends on the resistance of the coil. Therefore, if the material properties are specified for the coils or meshes in the model (refer to Chapter 14), the electric field can be plotted as well.

### Currents and Power

Next in the menu you find plots options for different current densities and the power density. The Current Density,  $\vec{J}$ , is defined as

$$\vec{J} = \frac{\vec{I}}{A}, \quad (15.6.2)$$

where  $I$  is the operating current that the user sets for each coil individually in the coil’s physics settings, and  $A$  is the cross-sectional area where the current flows. In case the user

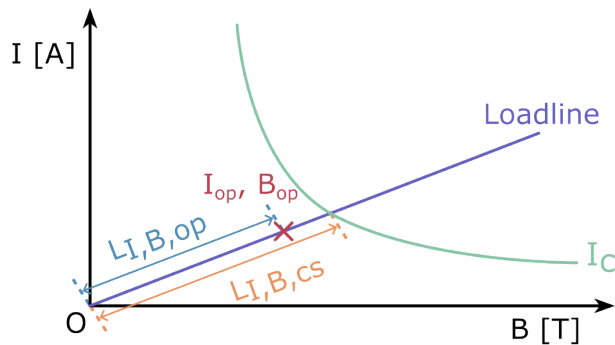
models individual turns the cross-section of a turns is used. When you model the coil such that the whole coil pack is extruded along the Path, which means that the coil's Cross-Section equals all the cross-sections of the turns and layers of a coil, the area is calculated by taking the area of the Cross-Section divided by the number of turns,  $n_{\text{turns}}$ . The  $n_{\text{turns}}$  parameter is also set by the user in the physics setting of a coil, see Chapter 4.

The Engineering Current Density, denoted as  $J_e$ , is another option available in the menu. It is defined as:

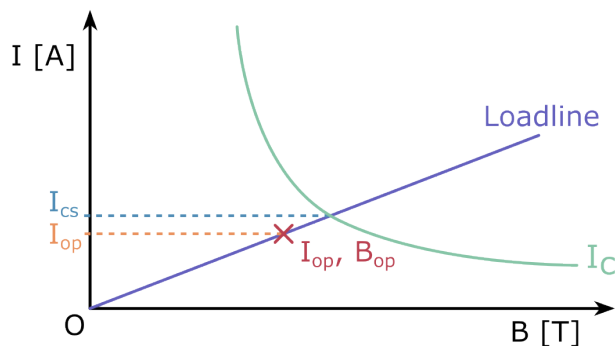
$$J_e = \frac{I_c}{A}, \quad (15.6.3)$$

where  $I_c$  represents the critical current, and  $A$  is the same area as for the current density  $\vec{J}$  (see Equation (15.6.2)). It is important to note that  $J_e$  should only be considered when the coil has superconducting material properties.

The Loadline Fraction and the Critical Current Fraction are additional properties that are only used for superconducting coils. The Loadline Fraction is defined as the length of the loadline from the origin to the point of operation ( $L_{I,B,op}$ ) divided by the length of the loadline from the origin to the current sharing point ( $L_{I,B,cs}$ ). The Critical Current Fraction is calculated as the ratio of the current ( $I$ ) to the critical current ( $I_c$ ). Illustrations of the Loadline Fraction and the Critical Current Fraction are shown in Figure 15.6.3.



(a) Loadline Fraction.



(b) Critical Current Fraction.

Figure 15.6.3: An illustration of the definition of the fraction of the loadline on the left, which is calculated by dividing  $L_{I,B,op}$  by  $L_{I,B,cs}$ , and an illustration of the definition of the critical current fraction, which is calculated by dividing  $I$  by  $I_c$ .

Except for the current density, all options mentioned so far are scalars. The last option in the block of current-like properties is the Power Density, and this is also a scalar. It is defined as current ( $I$ ) times voltage ( $V$ ) and it is only valid when the coil has material properties.

## Temperature

The next options in the list are temperature and temperature margin. The temperature of each coil can be set by the user in the physics settings using the `temperature` parameter. This allows you to specify the temperature at which the coil operates.

The temperature margin, on the other hand, is only used for superconducting coils and is applicable when material properties are set. It is defined as the difference between the current sharing temperature ( $T_{cs}$ ) and the temperature of the coil ( $T_{op}$ ). The temperature margin indicates the margin between the operating temperature and the temperature at which the coil quenches. An illustration of the temperature margin is shown in Figure 15.6.4.

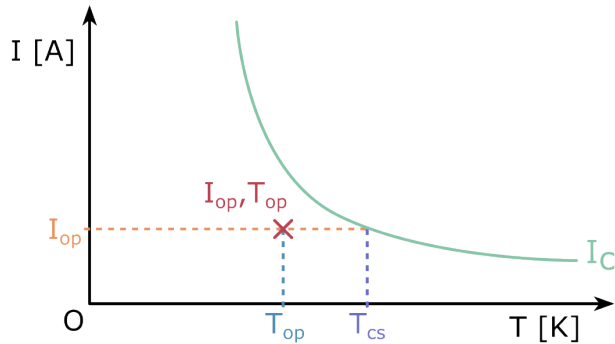


Figure 15.6.4: An illustration of the temperature margin, calculated as the difference between  $T_{cs}$  and  $T_{op}$ .

## Mechanics

Below the temperature options, you will find a few mechanical properties that can be plotted on the coil meshes. The property Curvature,  $\kappa$ , is calculated as one over the radius of the coil or a coil section in case the curvature varies along the coil. Next on the list is the Force Density. This is calculated based on the Lorentz force,  $F_L$ , which is caused by the magnetic field:

$$\vec{F}_L = \iiint_V (\vec{J} \times \vec{B}) dV, \quad (15.6.4)$$

where  $\vec{J}$  is the current density,  $\vec{B}$  is the magnetic field, and  $V$  is the volume of the coil.

The Lorentz force density is used to calculate the Pressure Vector, as shown in Equation (15.6.5), which can also be plotted in the *post-processor*. The Pressure Vector is a two-dimensional vector with the pressure in the normal direction,  $P_n$ , and the pressure in the transverse direction,  $P_d$ . These components are defined in the local coordinate system on the coil and are calculated using Equation (15.6.6) and Equation (15.6.7), where  $\vec{N}$  and

$\vec{D}$  are the normal and transverse vectors, respectively.

$$\vec{P} = \begin{pmatrix} P_n \\ P_d \end{pmatrix}, \quad (15.6.5)$$

$$P_n = \int \vec{F}_n \cdot \vec{N} \, dn, \quad (15.6.6)$$

$$P_d = \int \vec{F}_d \cdot \vec{D} \, dd. \quad (15.6.7)$$

For the pressure vector calculation in Rat, it is assumed that the coil is infinitely pliable, and that all the Lorenz force is acting on an infinitely stiff wall surrounding the coil. The Lorenz force is integrated along the coil, which means that the direction of the force might be the same as the normal and/or transverse vectors of the coil for one coil section, but in the opposite direction for a different part of the coil. The result of the integration then looks like the blue curve  $P_{n,0}$  in Figure 15.6.5.

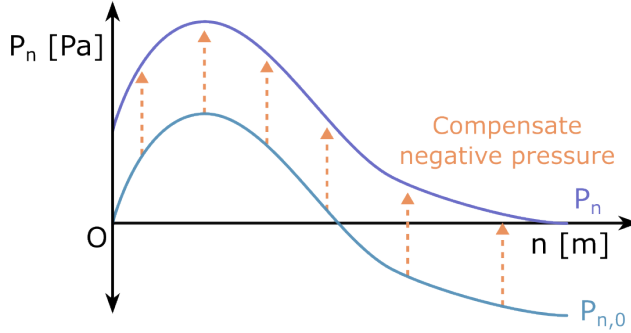


Figure 15.6.5: An illustration of the pressure vector integration and compensation for negative pressure.

Negative pressure, as in  $P_{n,0}$  in Figure 15.6.5, is non-physical. Therefore, a correction is performed where the curve is shifted such that the minimum value of the curve is at zero pressure. This zero pressure point can be interpreted as either the point where the coil is separating from or pushing into the imaginary wall, or a point in the coil where the material is pulled apart. This depends on whether the zero point is at the wall location (start or end of integration) or in the middle of the coil. This is demonstrated with imaginary material elements and a wall in Figure 15.6.6. The result of correcting for negative pressure,  $P_n$ , is shown in purple in Figures 15.6.5 and 15.6.6.

The last mechanical property that can be plotted with the results of the Mesh Calculation is the Von Mises stress,  $\sigma_V$ . In Rat, the pressure vector is used to calculate the principal plane stress as defined in Equation (15.6.8).

$$\sigma_V = \sqrt{P_n^2 + P_d^2 - P_n P_d} \quad (15.6.8)$$

## Geometric

The last four options in the list of Physical Properties of the Mesh Calculation *post-processor* are different geometric properties. The first geometry option, Index, displays the index of the Mesh that is used in Rat to identify a mesh or coil. If you export the data to VTK, you can use this information to identify the same meshes in ParaView.

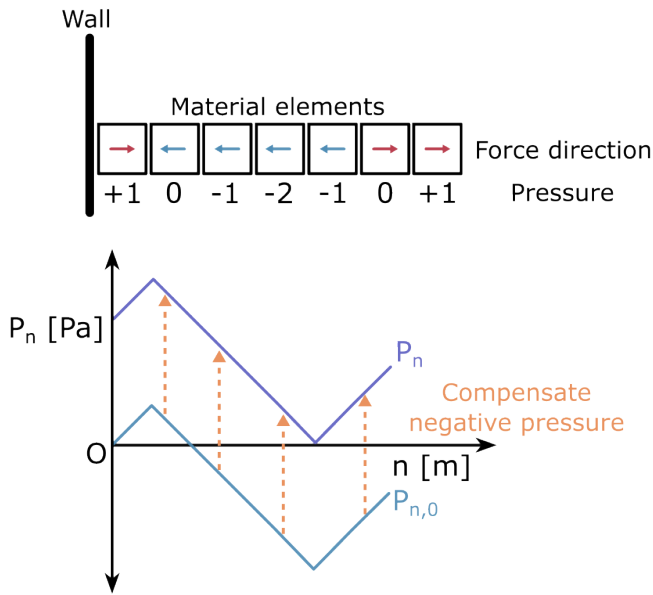


Figure 15.6.6: A demonstration of the pressure vector integration for a row of material elements that are pushing into a wall.

The Longitudinal ( $\vec{L}$ ), Normal ( $\vec{N}$ ), and Transverse ( $\vec{T}$ ) options plot the direction of the components of the local coordinate system of the conductor. The local coordinates are explained in Chapters 9 and 10, and they are illustrated in Figure 15.6.7. These plots can be used to visualize how Rat defines the winding direction of your coils. This can be particularly useful when interpreting results of coils wound with HTS tapes that have a critical current dependent on the field angle with respect to the conductor orientation.

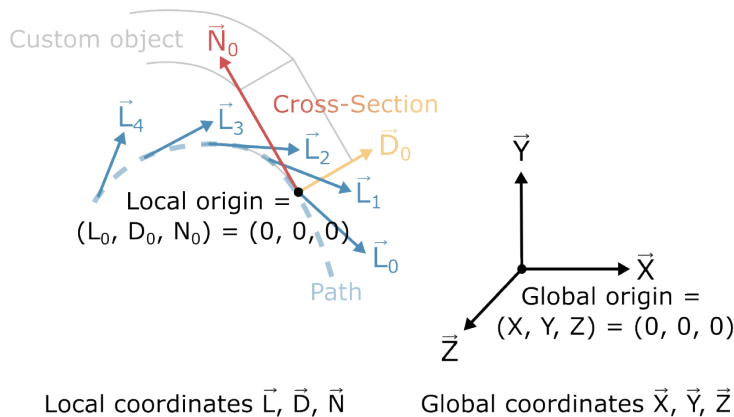


Figure 15.6.7: An illustration of the local coordinates of the conductor with respect to the global Cartesian coordinates.

Plotting  $\vec{L}$ ,  $\vec{N}$ , or  $\vec{T}$  only makes sense when you also select a component in the Field Component settings. These can be selected by choosing ‘Settings > Field Component’ in the *post-processor*. To demonstrate this, let’s look at a plot of  $\|\vec{L}\|$  and its three components

expressed in the global Cartesian system in Figure 15.6.8. By examining the  $x_L$ ,  $y_L$ , and  $z_L$  plots together, you can observe that  $\vec{L}$  for the left-most coil follows the azimuthal direction of that coil. This is also the winding direction assumed by Rat. It is important to note that all these vectors and components are unit vectors, and their value is therefore equal to one.

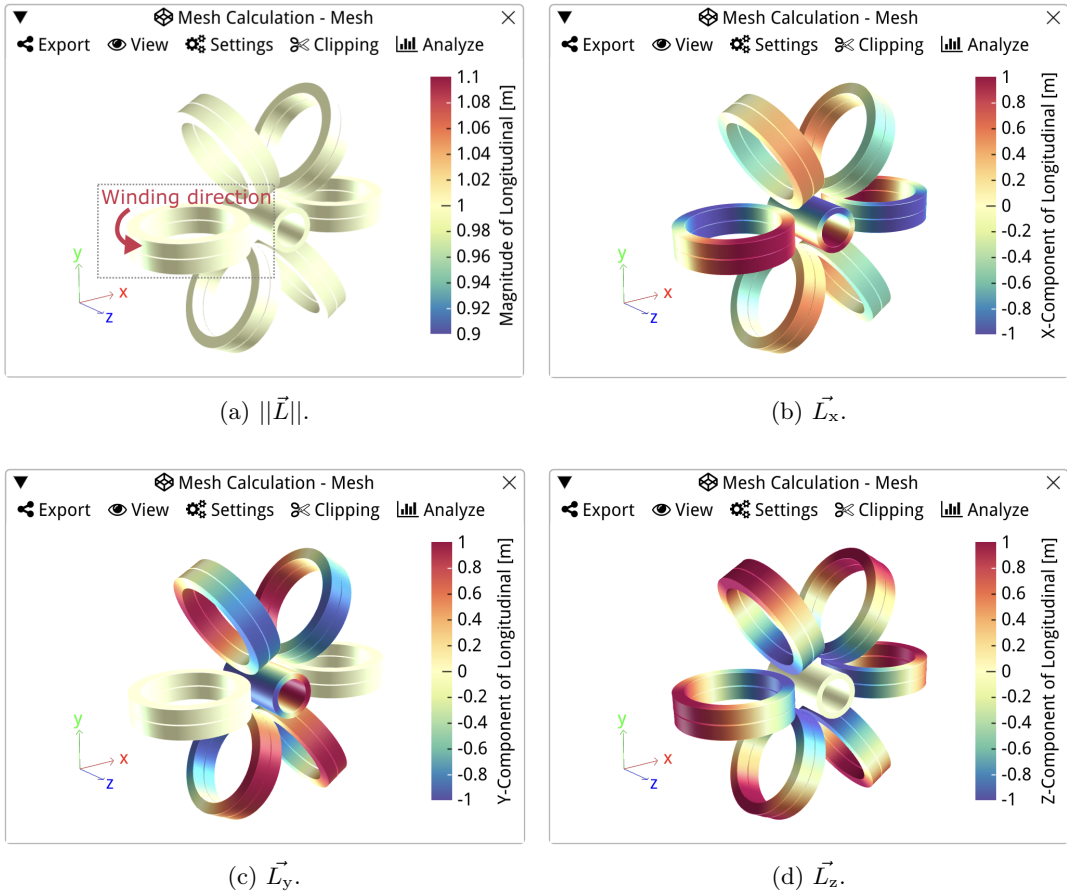


Figure 15.6.8: Plots of the Longitudinal vector,  $\vec{L}$ . The top left plot displays the magnitude of  $\vec{L}$ , the top right plot shows its X-component, the bottom left plot shows the Y-component, and the bottom right plot shows the Z-component.

## 15.6.2 Field Component

The Field Component settings can be used in combination with properties that are a field or a vector. The  $X$ ,  $Y$ , and  $Z$  components give information about the global coordinate system, and the  $L$ ,  $N$ , and  $D$  components about the global coordinate system of a conductor in a coil. The magnitude of a field or vector can be plotted as well and selected in this menu.

The last four settings in this list are angles with respect to the magnetic fields direction. These can be interesting to plot when you are designing HTS coils where the critical current of magnet depends on the magnetic field angle. These angles are defined by the following

equations:

$$\alpha = \arctan \sqrt{(\vec{L} \cdot \vec{B})^2 + (\vec{D} \cdot \vec{B})^2} / (\vec{N} \cdot \vec{B}) \quad (15.6.9)$$

$$\alpha_{\text{long}} = \arctan2 |(\vec{L} \cdot \vec{B}), (\vec{N} \cdot \vec{B})| \quad (15.6.10)$$

$$\alpha_{\text{trans}} = \arctan2 |(\vec{D} \cdot \vec{B}), (\vec{N} \cdot \vec{B})| \quad (15.6.11)$$

$$\alpha_{\text{plane}} = \arctan2 |(\vec{D} \cdot \vec{B}), (\vec{L} \cdot \vec{B})| \quad (15.6.12)$$

$$(15.6.13)$$

where  $\alpha$  is the magnetic field angle,  $\alpha_{\text{trans}}$  the transverse angle,  $\alpha_{\text{long}}$  is the longitudinal angle,  $\alpha_{\text{plane}}$  the in-plane angle, and  $\vec{B}$  is the magnetic field.

## 15.7 Cartesian Grid Calculation

The Grid Calculation is used to calculate the magnetic field on a three-dimensional rectangular grid that the user can place in the model space. By defining the sizes of the three dimensions of the grid and specifying the center point, users can customize the grid according to their requirements. These settings are specified in the Cartesian coordinates of the model. The dimensions of the grid box are represented by `sizex`, `sizey`, and `sizez` in the *node editor*, as shown in Figure 15.7.1. The center of the box, or its offset, can be set using the *x*, *y*, and *z* values.

The number of targets for the magnetic field calculation can be set in the discretization settings. Users can specify the number of nodes separately for each side of the grid. The `nnodesx` represents the number of nodes along the *x*-axis, `nnodesy` along the *y*-axis, and `nnodesz` along the *z*-axis of the grid. The total number of nodes is the product of these three settings.

The results of the Grid Calculation are presented as a three-dimensional isosurfaces plot in a *viewport*, refer to Figure 15.7.2. Users have the flexibility to choose the field type that the isosurfaces represent using the ‘Iso Field Type’ option in the *post-processor* settings of the Grid Calculation. An isosurface follows the field lines, where none of the selected Iso Field Type’s field lines intersect with an isosurface. Users can also change the component of the Iso Field Type using the ‘Iso Field Component’ setting. Additionally, the user can select the number of isosurfaces using the ‘Number of Isosurfaces’ option.

The color of the isosurface is determined by the ‘Field Type’ and ‘Field Component’ selection. If the Iso Field Type and the Field Type are the same, the surface has a uniform color. However, if they differ, the surface location indicates where the Iso Field Type is constant, while the color of the surface represents the value of the Field Type. Users can further customize the Grid Calculation plot by selecting a colormap.

The Grid Calculation *post-processor’s viewport* can be manipulated just like the other *Viewports* in the Rat GUI. Users can control the orientation of the grid and meshes using the mouse. Additional view options can be found under the ‘View’ section of the *post-processor navigation bar*. The data can also be exported to the CSV format and a VTK Grid.

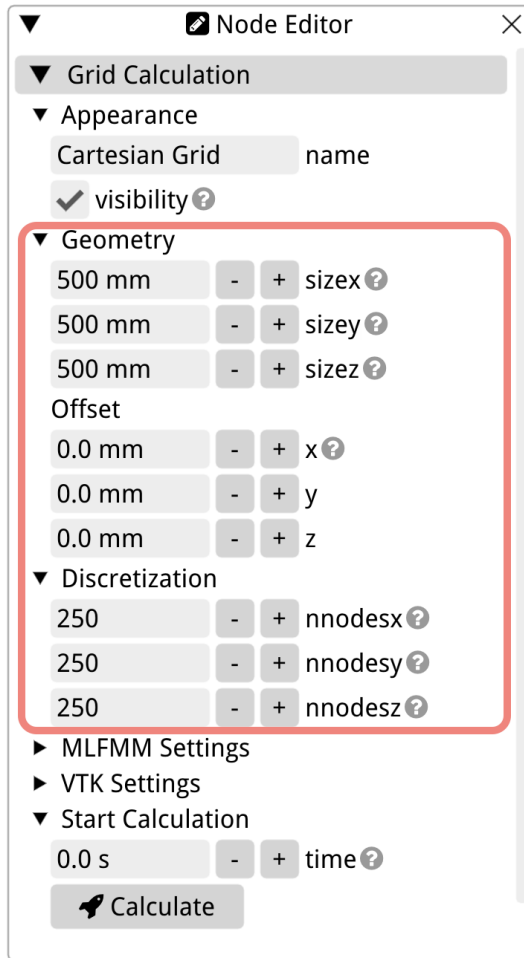


Figure 15.7.1: The *node editor* of the Cartesian Grid Calculation.

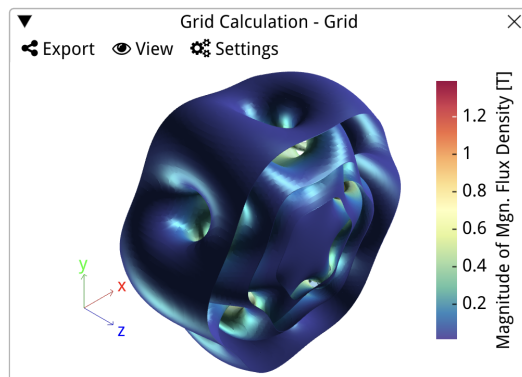


Figure 15.7.2: The *post-processor* of the Grid Calculation.



## 15.8 Polar Grid Calculation

### 15.9 Inductance

Another calculation that can be added to the Calculation Tree is the Inductance Calculation, which computes the inductance matrix and the stored magnetic energy matrix for all the coils in your model. If you have multiple coils connected to different circuits, you can group them using the `group circuits` checkbox in the *node editor*, refer to Figure 15.9.1. To ensure proper grouping, make sure to assign the same `circuit id` for coils within the same circuit. For more details on this topic, please see Section 4.1.

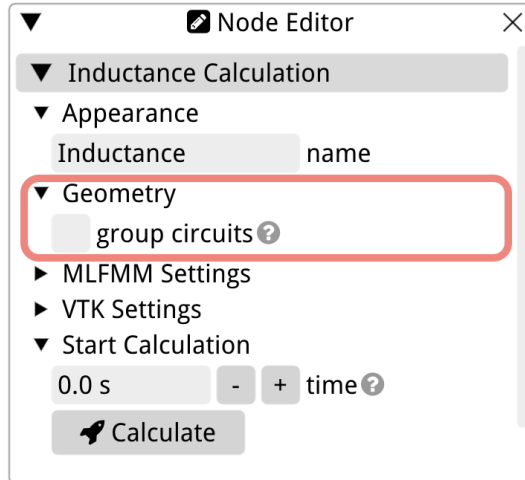


Figure 15.9.1: The *node editor* of the Inductance Calculation.

The *post-processor* for the Inductance Calculation consists of three tabs: the Inductance Matrix, the Energy Matrix, and the Inductance Heatmap. Both the inductance and energy matrices display the names of the coils in the column and row headers. The last row of the matrices represents the total inductance or stored magnetic energy of each column. An example of the *post-processor* for the Mini Toroid is shown in Figure 15.9.2.

The diagonal elements of the inductance matrix represent the self-inductance,  $L$ , of each coil, while the off-diagonal values indicate the mutual inductance,  $M_{1,2}$ , between coils 1 and 2. The stored magnetic energy,  $E_{\text{mag}}$ , can be calculated using the formula:

$$E_{\text{mag}} = \frac{1}{2} M_{1,2} I_1 I_2, \quad (15.9.1)$$

where  $I_1$  represents the current in coil 1 and  $I_2$  represents the current in coil 2. For values on the diagonal,  $M_{1,2}$  equals  $L$ , and  $I_1$  equals  $I_2$  and they correspond to the current of the respective coil.

The data can be exported in CSV and VTK table formats. In the settings of the *post-processor*, you have the option to modify the notation (floating or scientific) and precision (the number or decimal) of the values in the tables. The colormap options are specifically used in the Inductance Heatmap visualization.

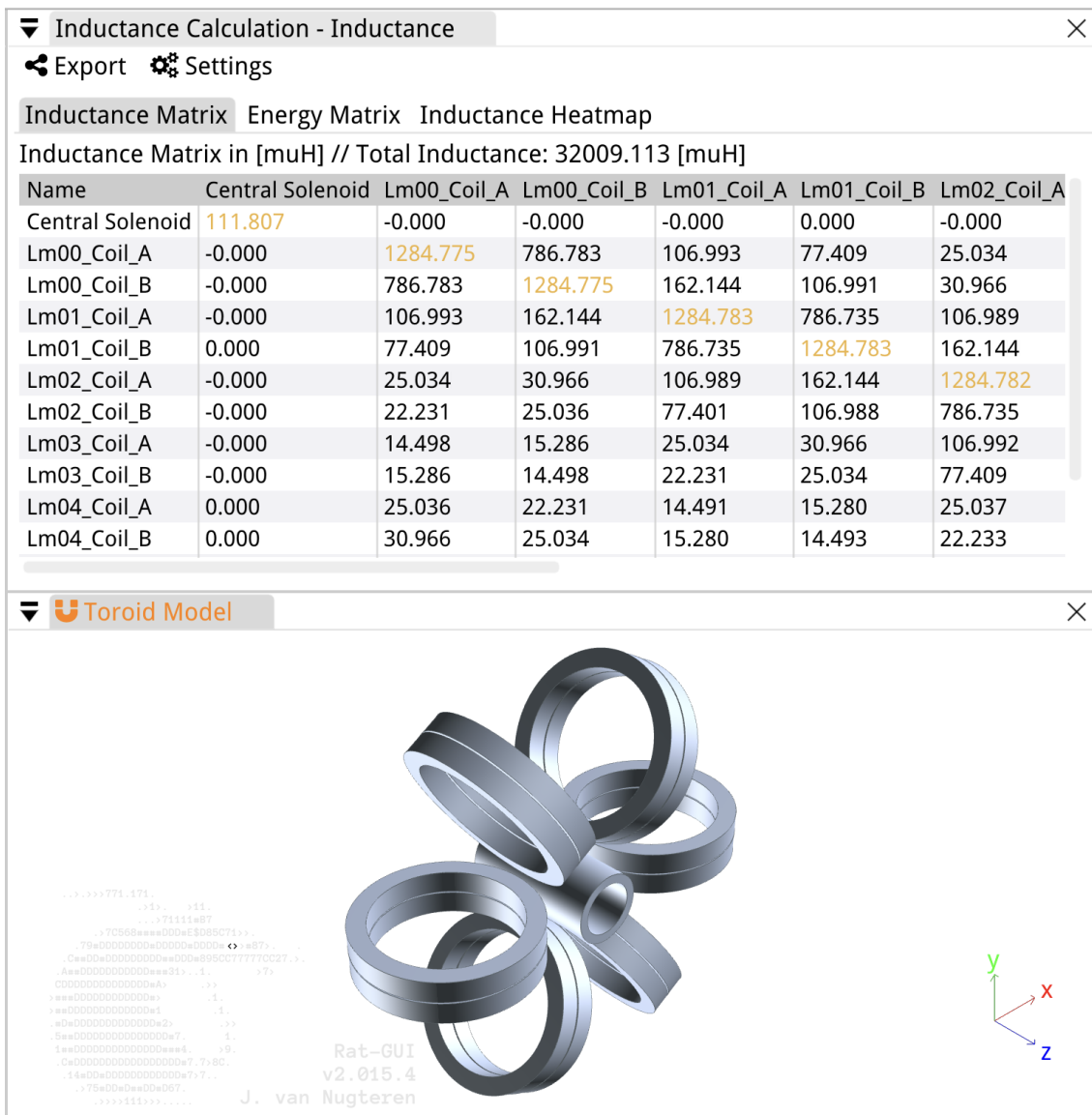


Figure 15.9.2: The *post-processor* (top) of the Inductance Calculation for the Mini Toroid Example (bottom).

15.10 Cylindrical Harmonics

15.11 Spherical Harmonics

15.12 Tracking

15.12.1 Add Tracking Mesh

15.12.2 Add Emitter

15.13 Length

# Bibliography

1. Little Beast Engineering Sàrl. *Rat GUI General Terms and Conditions* online. Jan. 29, 2022. [https://rat-gui.ch/licenses/GTandC\\_RAT\\_GUI.pdf](https://rat-gui.ch/licenses/GTandC_RAT_GUI.pdf) (2023).
2. Little Beast Engineering Sàrl. *Rat GUI General End User License Agreement* online. Jan. 22, 2022. [https://rat-gui.ch/licenses/EULA\\_RAT\\_GUI.pdf](https://rat-gui.ch/licenses/EULA_RAT_GUI.pdf) (2023).
3. Microsoft. *Microsoft Windows* online. 2023. <https://www.microsoft.com/en-us/windows> (2023).
4. Xianyi, Z. *et al. OpenBLAS - An optimized BLAS library* online, GitHub. 2023. <https://www.openblas.net/> (2023).
5. FreeCAD Community. *FreeCAD* online. 2023. <https://www.freecad.org/index.php> (2023).
6. Dassault Systèmes. *Opera: Electromagnetic and Electromechanical Simulation* online. 2023. <https://www.3ds.com/products-services/simulia/products/opera/> (2023).
7. Geuzaine, C. & Remacle, J.-F. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *Internation Journal for Numerical Methods in Engineering* **79**, 1309–1331 (May 7, 2009).
8. Dassault Systèmes. *Abaqus: Finite Element Analysis for Mechanical Engineering and Civil Engineering* online. 2023. <https://www.3ds.com/products-services/simulia/products/abaqus/> (2023).
9. Autodesk Inc. *Autodesk* online. 2023. <https://www.autodesk.com/> (2023).
10. Le Van, V., Meunier, G., Chadebec, O. & Guichon, J.-M. A Magnetic Vector Potential Volume Integral Formulation for Nonlinear Magnetostatic Problems. *IEEE Transactions on Magnetism* **52**. <https://hal.science/hal-01385525> (Mar. 2016).
11. Rao, D. K. & Kuptsov, V. Effective Use of Magnetization Data in the Design of Electric Machines With Overfluxed Regions. *IEEE Transactions on Magnetism* **51**, 1–9 (2015).
12. Le Van, V., Meunier, G., Chadebec, O. & Guichon, J.-M. A Volume Integral Formulation Based on Facet Elements for Nonlinear Magnetostatic Problems. *IEEE Transactions on Magnetism* **51**. <https://hal.science/hal-01177829> (July 2015).
13. Persson, P.-O. & Strang, G. A Simple Mesh Generator in MATLAB. *SIAM Review* **46**, 329–345. eprint: <https://doi.org/10.1137/S0036144503429121>. <https://doi.org/10.1137/S0036144503429121> (2004).
14. Bourke, P. Triangulate: Efficient Triangulation Algorithm Suitable for Terrain Modelling or An Algorithm for Interpolating Irregularly-Spaced Data with Applications in Terrain Modelling. Presented at Pan Pacific Computer Conference, Beijing, China. <http://paulbourke.net/papers/triangulate/> (Jan. 1989).

15. Schroeder, W., Martin, K. & Lorensen, B. *The Visualization Toolkit* 4th ed. ISBN: ISBN 978-1-930934-19-1. <https://vtk.org/> (2023) (2006).
16. Ahrens *et al.* *ParaView: An End-User Tool for Large Data Visualization* (ed Elsevier) ISBN: ISBN-13: 9780123875822. <https://www.paraview.org> (2023) (2005).
17. Greengard, L. & Rokhlin, V. A fast algorithm for particle simulations. *Journal of Computational Physics* **73**, 325–348. ISSN: 0021-9991. <https://www.sciencedirect.com/science/article/pii/0021999187901409> (1987).
18. Kurzak, J. & Pettitt, B. M. Fast multipole methods for particle dynamics. *Molecular Simulation* **32**. PMID: 19194526, 775–790. <https://doi.org/10.1080/08927020600991161> (2006).