SUN Magnetics

TetraHenry (TTH)

User Manual

v1.0.50

Authors:

Kyle Jackman
Coenrad Fourie

Version: v1.0.50
Date: August 30, 2022

SUN Magnetics (Pty) Ltd
15 De Beer Street
Stellenbosch, 7600
Republic of South Africa

www.sun-magnetics.com

Date: August 30, 2022

Credits

InductEx (the code base, file translators, pre-processor and post-processing algorithms and visualisation tools) is the product of the combined research and development efforts of Coenrad Fourie, Kyle Jackman, Mark Volkmann, Paul le Roux, Rebecca Roberts, Thomas Weighill, Ruben van Staden and Pierre Lötter.

InductEx also builds on the work done by Matton Kamon and Steve Whiteley (FFH), Paul Bunyk and Sergei Rylov (Lmeter), Angus Johnson and Bala Vatti (the polygon clipper unit).

Process-specific support with the assistance of Thomas Ortlepp, Olaf Wetzstein and Jürgen Kunert (FLUXONICS), Oleg Mukhanov, Timur Filippov and Alex Kirichenko (Hypres), Nobuyuki Yoshikawa, Yuki Yamanashi and Kohei Ehara (AIST STP and ADP), Vasili Semenov (Stony Brook University) and Sergey Tolpygo (MIT Lincoln Labs).

Integration into LayoutEditor was done by Jürgen Thies (Juspertor).

Integration into Cadence design environment with the help of Vasili Semenov (Stony Brook University) and Timur Filippov (Hypres).

Quality assurance (by the patient Beta testers and bug reporters): Vasili Semenov (Stony Brook University), Timur Filippov and Alex Kirichenko (Hypres), Pascal Febvre, Romain Collot and Ugur Yilmaz (IMEP-LAHC), Felix Jaeckel and Ruslan Hummatov (University of New Mexico), Xizhu Peng and Naoki Takeuchi (Yokohama National University), Damian Steiger (ETH Zurich), Oliver Brandel (IPHT Jena), Mankit Wong (IBM), Jonathan Stark (ColdLogix) and Steve Kaplan (Sandia National Laboratories), as well as Mark Volkmann, Rodwell Bakolo and Hein van Heerden (in-house).

InductEx is based upon work supported financially by the South African National Research Foundation, grant numbers 69006, 78789 and 93586, with additions developed under contracts W911NF14-C-0089 and W911NF-17-1-0120 with the Intelligence Advanced Research Projects Activity (IARPA)

Contents

List of Figures

List of Tables

1. TetraHenry (TTH)

1.1 Introduction

TetraHenry (*TTH*) [1] is a numerical field solver for inductance extraction and flux trapping analysis of superconducting integrated circuits. The solver uses tetrahedral elements to model multidirectional current flow in complex three-dimensional superconducting volumes. The latest version of TTH supports two-dimensional triangular elements for modeling sheet currents in thin superconducting films. The latest version also supports FastHenry [2] input meshes (filaments/cuboids meshes), which can be used in conjunction with tetrahedral and triangular meshes.

1.2 Input Mesh Formats

TetraHenry supports two input mesh formats: Gmsh's input mesh format [3], see section 1.2.1, and a modified version of FastHenry's input mesh format [2], see section 1.2.2.

1.2.1 Gmsh Mesh Format

For an in-depth description of the Gmsh's input mesh format and the construction of geometric models, see the Gmsh Reference Manual [3]. The current version of TetraHenry supports only version 2.2 of Gmsh's mesh format.

In Gmsh, geometric entities (volumes, surfaces, lines, etc.) can be combined into a "*physical group*" and this *physical group* can be exported to a mesh file with a unique user-defined *physical name* (127 characters max) [3]. Figure 1.1 shows an example of a microstrip line and the corresponding input JSON file. The material properties of the two metal layers/volumes are specified in the JSON file by referencing their corresponding *physical names* ("M0" and "M1"), see section 1.6. The surface terminals of the two ports are also referenced in the JSON file ("P1+","P1-","P2+" and "P2-"). For more information regarding the JSON input format see section 1.3.

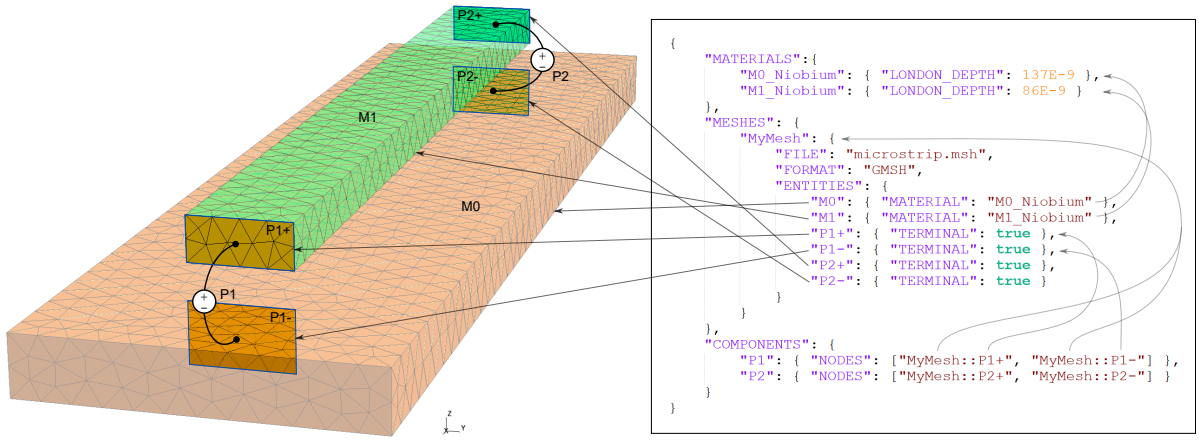


Figure 1.1: An example of a microstrip line with two voltage ports. A section of the corresponding input JSON file is shown on the right.

1.2.2 FastHenry Mesh Format

For an in-depth description of the FastHenry's input mesh format, see the FastHenry User's Guide [4].

In the original FastHenry input mesh format, each filament's material properties are specified within the mesh file. Contrarily, TetraHenry requires material properties to be specified in a separated JSON file. The current version of TetraHenry therefore requires some modifications to the original FastHenry's input mesh format. These modifications allow for the material properties of the filaments (cuboids) to be specified inside the JSON file.

Instead of assigning material properties (`sigma` and `lambda`) to filaments inside the mesh file, each filament must be assigned a *physical name*, e.g. `phys="M1"`, similar to the Gmsh format. *Physical name* can also be assigned to "electrically equivalent" nodes (`.equiv`), which can then be referenced as port terminals inside the JSON file. Listing 1.1 shows and example of a modified version of FastHenry's input mesh format.

The following keywords/properties are new in the modified version:

- `L` - All lines starting with `L` defines a line between two nodes. These lines can be used to define a fluxon path, see "HOLE" under section 1.7.1.

The following keywords/properties are ignored in the modified version:

- `lambda` - The London penetration depth cannot be specified inside the mesh file. Instead, a *physical name* (e.g. `phys="M1"`) must be assigned to each filament. The London penetration depth can then be specified in the JSON file, see section 1.4 and section 1.6.

- `sigma` - The conductivity cannot be specified inside the mesh file. Instead, a *physical name* (e.g. `phys="M1"`) must be assigned to each filament. The conductivity can then be specified in the JSON file, see section 1.4 and section 1.6.

- `.external` - Excitation ports must be specified in the JSON file, see section 1.7.

- `.freq` - Frequency must be specified in the JSON file, see section 1.12.2.

```
        E1 N01 N02 w=1.250000 h=0.220000 nwinc=1 nhinc=2 phys="M1"
        E2 N03 N04 w=1.250000 h=0.220000 nwinc=1 nhinc=2 phys="M1"
        E3 N05 N06 w=1.250000 h=0.220000 nwinc=1 nhinc=2 phys="M1"
        E4 N07 N08 w=1.250000 h=0.300000 nwinc=1 nhinc=2 phys="M0"
        E5 N09 N10 w=1.250000 h=0.300000 nwinc=1 nhinc=2 phys="M0"
        E6 N11 N12 w=1.250000 h=0.300000 nwinc=1 nhinc=2 phys="M0"
        L11 N13 N14 phys="Line1"
        L12 N15 N15 phys="Line1"
        L21 N23 N24 phys="Line2"
        L22 N25 N25 phys="Line2"
        .equiv N01 N03 N05 phys="P1+"
        .equiv N07 N09 N11 phys="P1-"
```

Listing 1.1: An example of a modified version of FastHenry's input mesh format.

1.3 Input File Requirements

TetraHenry requires a single JSON file as input. All settings must be defined using key-value pairs within the JSON file. These key-value pairs can be placed in any order within the JSON file. The specifications for the content of each JSON object is discussed in this section. Each table contains the specifications for a specific JSON object. Each table row defines a single key-value pair and contains the following columns:

- *Key* - Defines the key (option) available for a given JSON object.
- *Value* - Defines the format of the value for the corresponding key.
- *Required (Req.)* - Defines whether the key-value pair must be specified within the given JSON object. If a JSON object is not required, then the key-value pairs within this JSON object are also not required.
- *Description* - A description of the key-value pair.

The excepted formats for input *values* are define below:

- *Object* - The value must be a JSON object. JSON objects are surrounded by curly braces {}.
- *Number* - Can be integer or floating point number.
- *Integer* - Must be an integer number.
- *Complex* - A *number*, e.g. $2.0 = 2.0 + 0.0i$, or an array consisting of two *numbers* representing the real and imaginary components, e.g. $[2.0, 4.5] = 2.0 + 4.5i$.
- *Vector* - An array consisting of three real or complex numbers, i.e. the x, y, z-components of the vector, e.g. $[1.0, 2.0, [3.0, 4.0]] = 1.0\hat{x} + 2.0\hat{y} + (3.0 + 4.0i)\hat{z}$.

- *String* - An arbitrary string of characters, within quotation marks.
- *Fixed String* - A specific predefined string.
- *Array of Strings* - An array of arbitrary strings.
- *Array of Fixed Strings* - An array of predefined strings.
- *Array of Integers* - An array of integers.

```
{
    "VERSION": 1.0,
    "SETTINGS": { },
    "MATERIALS": { },
    "MESHES": { },
    "FIELDS": { },
    "COMPONENTS": { },
    "CONDUCTORS": { },
    "OUTPUT": { },
    "IMPORT": ["default_settings.json", "default_materials.json"]
}
```

Listing 1.2: JSON input path: `/`

Key	Value	Req.	Description
VERSION	*Number*	No	Version of TTH input format. (default: 1.0)
SETTINGS	*Object*	No	Contains input settings.
MATERIALS	*Object*	No	Contains materials and material properties.
MESHES	*Object*	No	Specify imported meshes and mesh entities.
FIELDS	*Object*	No	Specify externally applied fields.
COMPONENTS	*Object*	No	Specify components (voltage sources, resistors, etc.) connected to mesh entities. Components can only be defined when MQS, EMQS or FULL mode is enabled, see section 1.12.1.
CONDUCTORS	*Object*	No	Specify the conductors for capacitance extraction. Conductors can only be defined when EQS mode is enabled, see section 1.12.1.
OUTPUT	*Object*	No	Specify which results should be exported to output files.
IMPORT	*Array of Strings*	No	An array of paths to JSON files. These JSON files will be imported and merge with the input JSON file. Any duplicate key-value pairs within the first imported JSON file will be overwritten by the second imported JSON file, ect. Finally, the input JSON file will override all the duplicate key-value pairs of the imported JSON files. The imported JSON files cannot import their own JSON files, i.e. the IMPORT setting will not work within the imported JSON files.

Table 1.1: JSON input path: /

1.4 Materials

```
"MATERIALS": {
   "MyMaterial": { }
   "Niobium": { },
   "Molybdenum": { },
   "Dielectric_1": {},
   "PEC": { }
}
```

Listing 1.3: JSON input path: `/MATERIALS`

Key	Value	Req.	Description
User defined string, e.g. MyMaterial	*Object*	Yes	The *key* can be any alphanumeric string. This *key* defines the name of the material.

Table 1.2: JSON input path: `/MATERIALS`

1.5 Material Properties

```
"MATERIALS": {
   "MyMaterial": {
       "LONDON_DEPTH": 90E-9,
       "LAMBDA": 90E-9,
       "CONDUCTANCE": 5.95E7,
       "SIGMA": 5.95E7,
       "RESISTIVITY": 1.68E-8,
       "REL_PERMITTIVITY": 1.0,
       "REL_EPS": 1.0,
       "REL_PERMEABILITY": 1.0,
       "REL_MU": 1.0,
       "PEC": false
   }
}
```

Listing 1.4: JSON input path: `/MATERIALS/MyMaterial`

Key	Value	Req.	Description
`LONDON_DEPTH / LAMBDA`	*Number*	No	London penetration depth, λ_L. The superconducting two-fluid model is used. If $\lambda_L = 0$, superconductivity will be disabled. If $\lambda_L \neq 0$ and DC analysis is enabled, the material will be assumed to a PEC. (default: 0.0) [Unit: m]
`CONDUCTANCE / SIGMA`	*Complex*	No	Electrical conductance, σ. If $\sigma = 0$, $\lambda_L = 0$, $\mu_r = 1$ and $\epsilon_r = 1$, the material is assumed to be free-space. (default: 0.0) [Unit: S/m]
`RESISTIVITY / RHO`	*Complex*	No	Resistivity = 1.0 / conductivity, ρ. If $\rho = 0$, $\lambda_L = 0$, $\mu_r = 1$ and $\epsilon_r = 1$, the material is assumed to be free-space. [Unit: Ω·m]
`REL_PERMITTIVITY / REL_EPS`	*Complex*	No	Relative dielectric permittivity, ϵ_r. (default: 1.0)
`REL_PERMEABILITY / REL_MU`	*Complex*	No	Relative magnetic permeability, μ_r. (default: 1.0)
`PEC`	*Boolean*	No	If true, material is a Perfect Electric Conductor. (default: true)

Table 1.3: JSON input path: `/MATERIALS/MyMaterial`

1.6 Meshes

```
"MESHES": {
   "MyMesh": {},
   "Structure1": {},
   "Structure2": {},
   "MyFieldMesh": {}
}
```

Listing 1.5: JSON input path: `/MESHES`

Key	Value	Req.	Description
User defined string, e.g. MyMesh	*Object*	Yes	The *key* can be any alphanumeric string. This *key* defines the name of the imported mesh.

Table 1.4: JSON input path: `/MESHES`

1.6.1 Mesh Properties

```
"MESHES": {
   "MyMesh": {
      "FILE": "microstrip.msh",
      "FORMAT": "GMSH",
      "ENTITIES": { }
```

```
    }
}
```

Listing 1.6: JSON input path: `/MESHES/MyMesh`

Key	Value	Req.	Description
`FILE`	*String*	Yes	Path to imported mesh file.

`FORMAT`	*Fixed String:*
	`"GMSH"`
	
	`"FASTHENRY"`
		Yes	Format of to imported mesh file.
			Gmsh file format. The mesh format must be version 2.2 [3].
			Modified FastHenry file format. This is a modified version of FastHenry's original file format.

`ENTITIES`	*Object*	Yes	Specifies which entities (volumes, surfaces, lines) within the imported mesh file must be imported, see section 1.6.2.

Table 1.5: JSON input path: `/MESHES/MyMesh`

1.6.2 Mesh Entities

```
"MESHES": {
   "MyMesh": {
      "ENTITIES": {
         "MyEntity": {},
         "M0": {},
         "M1": {},
         "P1+": {},
         "P1-": {},
         "Hole_1": {},
         "CutPort": {},
         "C1": {},
         "C2": {}
      }
   }
}
```

Listing 1.7: JSON input path: `/MESHES/MyMesh/ENTITIES`

Key	Value	Req.	Description
User defined string, e.g. MyEntity	*Object*	Yes	The *key* can be any alphanumeric string. This *key* defines the name of the entity (volume, surface or line) that will be imported from the mesh file (`FILE`). The name of entity (*key*) must be identical to the name of the entity within the mesh file. If the imported mesh does not contain an entity with the same name, an error message will be displayed and the program will exit.

Table 1.6: JSON input path: `/MESHES/MyMesh/ENTITIES`

1.6.3 Entity Properties

```
"MESHES": {
   "MyMesh": {
      "ENTITIES": {
         "MyEntity": {
            "MATERIAL": "PEC",
            "INNER_MATERIAL": "PEC",
            "OUTER_MATERIAL": "Dielectric_1",
            "OUTER_MATERIAL": ["side","bottom","top"],
            "TERMINAL": false,
            "TERMINAL_MODEL": "EQUIVALENT"
            "HOLE": false,
            "BOUNDARY": false,
            "SHEET_THICKNESS": [0, 200E-9],
            "SHEET_MODEL": "THICK",
            "SHEET_LAYERS": 1,
            "SHEET_LAYERS_RATIO": 2
         }
      }
   }
}
```

Listing 1.8: JSON input path: `/MATERIALS/MyMesh/ENTITIES/MyEntity`

Key	Value	Req.	Description
MATERIAL / INNER_MATERIAL	*String*	No	The inner material of this mesh entity. For volume entities, MATERIAL/INNER_MATERIAL refers to the material within the volume entity. For surface entities, MATERIAL/INNER_MATERIAL refers to the material on the side opposite to the direction of the surface's normal vector. This string must point to a material defined under /MATERIALS, see section 1.6.
OUTER_MATERIAL	*String*	No	The outer material of this mesh entity. For volume entities, OUTER_MATERIAL refers to the material outside the volume entity. For surface entities, OUTER_MATERIAL refers to the material on the same side of the direction of the surface's normal vector. This string must point to a material defined under /MATERIALS, see section 1.6.
	Array of Strings		An array of three strings can also be used to define the side, bottom, top materials of a thin film sheet model. The first value in the array defines the material on the outer edges of the thin film. The second value in the array defines the material below the thin film, i.e. opposite side of the normal direction. The third value in the array defines the material above the thin film, i.e. same side of the normal direction.
TERMINAL	*Boolean*	No	If true, the mesh entity is a terminal. A mesh entity must be a terminal if it forms part of the positive/negative node of a component or a cut port (default: false)
TERMINAL_MODEL	*Fixed String:*	No	Define the type of terminal model. (default: "EQUIVALENT")
	"EQUIVALENT"		All the mesh elements of the terminal are electrically shorted.
	"ENHANCED"		A sub-component is connected to each mesh element of the terminal. If a component (e.g. voltage source) is connected between two "ENHANCED" terminals, a set of sub-components (e.g. set of voltage sources) will be connected between each mesh element of the positive and negative terminals. The set of sub-components respresents the full component.
HOLE	*Boolean*	No	If true, the mesh entity is a hole port. Hole ports are defined using a closed path and can only be enabled for line entities. (default: false)

Table 1.7: JSON input path: /MATERIALS/MyMesh/ENTITIES/MyEntity (Part 1)

Key	Value	Req.	Description
BOUNDARY	*Boolean*	No	If true, the mesh entity is an electrical boundary. (default: false)
SHEET_THICKNESS	*Number*	No	The thickness of a thin film (default: 0.0). This setting can only be applied to surface entities. If the thickness is specified by a single *number*, the surface entity represents the center of the film, i.e. the lower and upper parts of a film will be equal to halve of the total thickness, see fig. 1.2. If this value is non-zero, the surface will be treated as a thin film with finite thickness, i.e. a volume. If this value is zero, the surface will be treated as a 2D surface.
	Array of Numbers		An array of two numbers can also be used to specify the lower and upper parts of a film, see fig. 1.2. The first value in the array defines the thickness below the surface entity, i.e. opposite side of the normal direction. The second value in the array defines the thickness above the surface entity, i.e. same side of the normal direction.
SHEET_MODEL	*Fixed String:* "THICK"	No	Sheet current model for a thin film (default: "THICK") Current flows on the top and bottom surfaces of the film.
	"THIN"		Current flows only at the center of the film.
SHEET_LAYERS	*Integer*	No	Defines the number of layers that will be used to subdivide the height of the thin film, similar to the *nhinc* parameter of FastHenry [2]. This setting can only be applied to surface entities with a non-zero sheet thickness. (default: 1)
SHEET_LAYERS_RATIO	*Number*	No	Defines the ratio of adjacent height subdivisions of the thin film, similar to the *rh* parameter of FastHenry [2]. This setting can only be applied to surface entities with a non-zero sheet thickness. (default: 2.0)

Table 1.8: JSON input path: /MATERIALS/MyMesh/ENTITIES/MyEntity (Part 2)

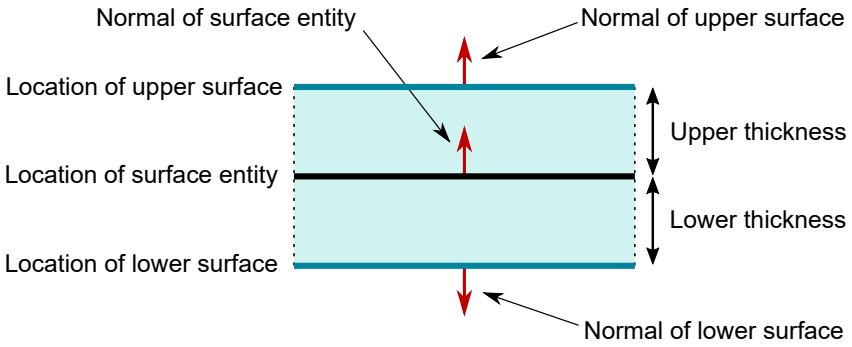


Figure 1.2: Illustration of the cross section of the thin film (sheet model).

1.7 Components

Circuit components can only be defined when mode MQS, EMQS or FULL is enabled, see section 1.12.1. Circuit components can be specified in the `COMPONENTS` object.

```
"COMPONENTS": {
   "MyComponent": { },
   "P1": { },
   "P2": { },
   "V1": { },
   "I1": { },
   "R1": { },
   "L1": { }
}
```

Listing 1.9: JSON input path: `/COMPONENTS`

Key	Value	Req.	Description
User defined string	*Object*	Yes	The *key* can be any alphanumeric string. This *key* defines the name of the component.

Table 1.9: JSON input path: `/COMPONENTS`

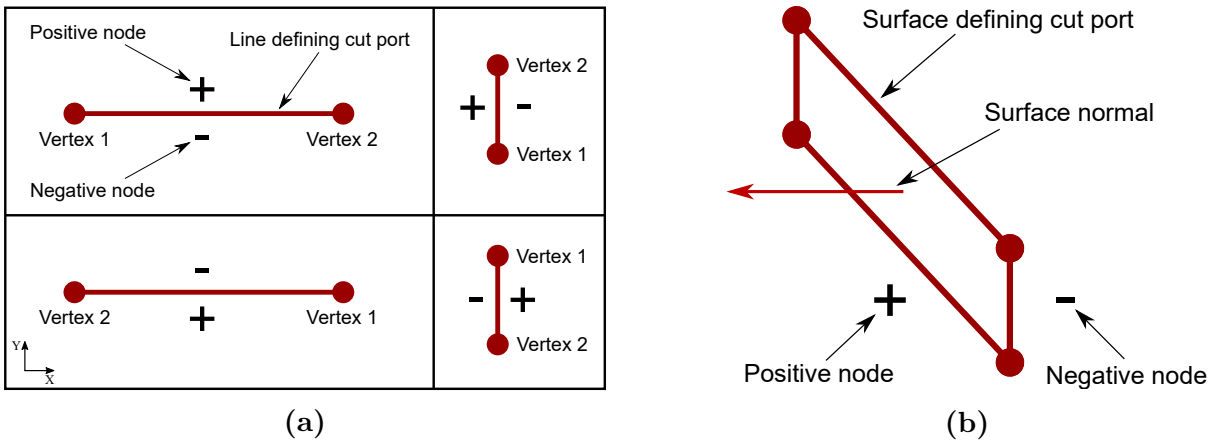
1.7.1 Component Properties

```
"COMPONENTS": {
    "MyComponent": {
        "TYPE": "VOLTAGE",
        "NODES": ["MyMesh::P1+", "MyMesh::P1-"],
        "NODES": ["MyMesh::P2+", ["MyMesh::P21-","MyMesh::P22-"]],
        "VALUE": 1.0,
        "FIXED": false
    },
    "V1": {
        "TYPE": "VOLTAGE",
        "CUT": ["MyMesh::CutPort"],
        "VALUE": 1.0
    },
    "Moat": {
        "TYPE": "PHASE",
        "HOLE": ["MyMesh::Hole_1"]
    }
}
```

Listing 1.10: JSON input path: `/COMPONENTS/MyComponent`

Key	Value	Req.	Description
TYPE	*Fixed String:* "VOLTAGE" / "V" "CURRENT" / "I" "PHASE" / "P" "RESISTOR" / "R" "INDUCTOR" / "L" "IMPEDANCE" / "Z" "SHORTCIRCUIT" / "S" / "SHORT"	No	The type of component. (default: VOLTAGE) Voltage source. Current source. Phase source. Resistor. Inductor. Impedance. Electrical short circuit.
NODES	*String Array*	No	An array consisting of two string values. The first and second strings refer to the positive and negative nodes of the component, respectively. The string value must first define the mesh name and then the mesh entity, separated by "::" , e.g. "MyMesh::P1+" points to /MATERIALS/MyMesh/ENTITIES/P1+. If a positive of negative node is connected to multiple mesh entities, an array of strings can also be used to define the positive or negative node.
CUT	*String*	No	The string value refers to the line/surface entity that defines a cut port. The positive and negative nodes of the component depend on the vertices/normal of the line/surface entity, as shown in fig. 1.3. The string value must first define the mesh name and then the mesh entity, separated by "::" , e.g. "MyMesh::CutPort" points to /MATERIALS/MyMesh/ENTITIES/CutPort.
HOLE	*String Array*	No	An array consisting of one string value. The string value refers to the line entity that defines the path of the hole port. The string value must first define the mesh name and then the mesh entity, separated by "::" , e.g. "MyMesh::Hole_1" points to /MATERIALS/MyMesh/ENTITIES/Hole_1.
VALUE	*Number*	No	The value of the component. (default: 1.0, default for phase sources: 2π)
FIXED	*Boolean*	No	Specifies whether the excitation source remain fixed. This setting only applies to voltage, current and phase sources. By default (FIXED = false), an excitation source will be activated independently, while all other excitation sources are disabled (electrically shorted). If FIXED = true, then this excitation source will remain active and will not be disabled (will not be electrically shorted) when activating other excitation sources. (default: false)

Table 1.10: JSON input path: /COMPONENTS/MyComponent



(a) (b)

Figure 1.3: (a) Line entity defining a cut port. (b) Surface entity defining a cut port.

1.8 Conductors

Conductors can only be defined when capacitance extraction (EQS mode) is enabled, see section 1.12.1. Conductors can be specified in the `CONDUCTORS` object.

```
"CONDUCTORS": {
        "MyConductor": {},
        "C1": {},
        "C2": {}
}
```

Listing 1.11: JSON input path: `/CONDUCTORS`

Key	Value	Req.	Description
User defined string	*Object*	Yes	The *key* can be any alphanumeric string. This *key* defines the name of the PEC conductor.

Table 1.11: JSON input path: `/CONDUCTORS`

1.8.1 Conductor Properties

```
"CONDUCTORS": {
   "MyConductor": {
      "SURFACES" = ["MyMesh::C1"],
      "VALUE" = 1.0
   }
}
```

Listing 1.12: JSON input path: `/CONDUCTORS/MyConductor`

Key	Value	Req.	Description
SURFACES	*Array of String*	Yes	An array of string values. These strings refer to the boundary surface entities of the conductor. The string value must first define the mesh name and then the surface mesh entity, separated by "::" , e.g. "MyMesh::C1" points to /MATERIALS/MyMesh/ENTITIES/C1.
VALUE	*Number*	No	The value of the voltage applied to the conductor when excited. Each conductor will be activated independently, while all other excitation conducotors are disabled (electrically shorted). (default: 1.0)

Table 1.12: JSON input path: /CONDUCTORS/MyConductor

1.9 Fields

Externally applied uniform or gradient fields can be specified in the FIELDS object.

```
"FIELDS": {
   "MyField": {},
   "UniformField_Bz": {},
   "GradientField_Gxz": {},
}
```

Listing 1.13: JSON input path: /FIELDS

Key	Value	Req.	Description
User defined string	*Object*	Yes	The *key* can be any alphanumeric string. This *key* defines the name of the externally applied field.

Table 1.13: JSON input path: /FIELDS

1.10 Field Properties

```
"FIELDS": {
   "MyField": {
      "TYPE": "B",
      "UNIFORM": [0.0, 0.0, 1.0],
      "UNIFORM_X": [1.0, 0.0],
      "UNIFORM_Y": [1.0, 0.0],
      "UNIFORM_Z": [1.0, 0.0],
      "GRADIENT_X": [0.0, 0.0, 1.0],
      "GRADIENT_Y": [0.0, 0.0, 1.0],
      "GRADIENT_Z": [0.0, 0.0, 1.0],
```

```
        "GRADIENT_X_X": [1.0, 0.0],
        "GRADIENT_X_Y": [1.0, 0.0],
        "GRADIENT_X_Z": [1.0, 0.0],
        "GRADIENT_Y_X": [1.0, 0.0],
        "GRADIENT_Y_Y": [1.0, 0.0],
        "GRADIENT_Y_Z": [1.0, 0.0],
        "GRADIENT_Z_X": [1.0, 0.0],
        "GRADIENT_Z_Y": [1.0, 0.0]
    }
}
```

Listing 1.14: JSON input path: `/FIELDS/MyField`

Key	Value	Req.	Description

`TYPE`	*String*
	`"H"`
	`"B"`
	`"E"`
	`"D"`
		Yes	The type of the applied field.
			Magnetic field strength. [Unit: A/m]
			Magnetic flux density. [Unit: T]
			Electric field. [Unit: V/m]
			Electric displacement field. [Unit: C·m^{-2}]

`UNIFORM`	*Vector*	No	Uniform field vector. The vector must contain the three components of the field, i.e. the x, y ans z components.
`UNIFORM_X`	*Complex*	No	Uniform field in the x-direction.
`UNIFORM_Y`	*Complex*	No	Uniform field in the y-direction.
`UNIFORM_Z`	*Complex*	No	Uniform field in the z-direction.
`GRADIENT_X`	*Vector*	No	The x-components of a gradient field. The vector must contain the three values: $[G_{xx}, G_{xy}, G_{xz}] = [B_x/dx, B_x/dy, B_x/dz]$.
`GRADIENT_Y`	*Vector*	No	The y-components of a gradient field. The vector must contain the three values: $[G_{yx}, G_{yy}, G_{yz}] = [B_y/dx, B_y/dy, B_y/dz]$.
`GRADIENT_Z`	*Vector*	No	The z-components of a gradient field. The vector must contain the three values: $[G_{zx}, G_{zy}, G_{zz}] = [B_z/dx, B_z/dy, B_z/dz]$.
`GRADIENT_X_X`	*Complex*	No	The G_{xx} component of the gradient field.
`GRADIENT_X_Y`	*Complex*	No	The G_{xy} component of the gradient field.
`GRADIENT_X_Z`	*Complex*	No	The G_{xz} component of the gradient field.
`GRADIENT_Y_X`	*Complex*	No	The G_{yx} component of the gradient field.
`GRADIENT_Y_Y`	*Complex*	No	The G_{yy} component of the gradient field.
`GRADIENT_Y_Z`	*Complex*	No	The G_{yz} component of the gradient field.
`GRADIENT_Z_X`	*Complex*	No	The G_{zx} component of the gradient field.
`GRADIENT_Z_Y`	*Complex*	No	The G_{zy} component of the gradient field.
`GRADIENT_Z_Z`	*Complex*	No	The G_{zz} component of the gradient field.

Table 1.14: JSON input path: `/FIELDS/MyField`

1.11 Output

The output (voltages, currents, charges, etc.) of circuit components/conductors can be defines within the `OUTPUT` object. Three-dimensions VTK plots can also be can be defines within the `OUTPUT` object. These VTK plots can be viewed with an open-source tool called Paraview [5].

The extracted results can also be exported to a compact model (JoSIM subcircuit), if mode MQS is enabled, see section 1.11.3.

```
"OUTPUT": {
    "PRINT": { },
    "PLOT": { },
    "MODEL": { }
}
```

Listing 1.15: JSON input path: `/OUTPUT`

Key	Value	Req.	Description
PRINT	*Object*	No	Define output that will be printed to text file, e.g. voltages, currents, charges, etc.
PLOT	*Object*	No	Define VTK output plots, e.g. current density, vector fields, etc.

Table 1.15: JSON input path: `/OUTPUT`

1.11.1 Print

```
"OUTPUT": {
    "PRINT": {
        "VOLTAGE": { }
        "CURRENT": { }
        "CHARGE": { }
        "POWER": { }
    }
}
```

Listing 1.16: JSON input path: `/OUTPUT/PRINT`

Key	Value	Req.	Description
VOLTAGE	*Object*	No	Print voltages of circuit components to file. Mode MQS, EMQS or FULL must be enabled.
CURRENT	*Object*	No	Print currents of circuit components to file. Mode MQS, EMQS or FULL must be enabled.
CHARGE	*Object*	No	Print charges of conductors to file. Mode EQS must be enabled.
POWER	*Object*	No	Print complex power (unit: volt-amps) of the entire structure to file. If PHASE analysis is used, the energy (unit: joules) of the entire superconducting structure will be written to file.

Table 1.16: JSON input path: /OUTPUT/PRINT

Print Voltages

The voltage values of any component (defined under /COMPONENTS) can be exported to a text file.

```
"OUTPUT": {
   "PRINT": {
      "VOLTAGE": {
         "COMPONENTS" = ["P1", "P2"],
         "FILE" = "voltages.txt",
         "FORMAT": "TYPE_1"
      }
   }
}
```

Listing 1.17: JSON input path: /OUTPUT/PRINT/VOLTAGE

Key	Value	Req.	Description
COMPONENTS	*Array of Strings*	No	Array of component names. The voltages over these components will be exported to a text file. These names must correspond to the component names defined under /COMPONENTS, see section 1.7. If this key is not defined, the voltages over all components will be exported to the text file.
FILE	*String*	Yes	The name of the output file.
FORMAT	*Fixed String:* "TYPE_1" "TYPE_2"	No	The output format. (default: "TYPE_1") Original TTH/FFH output format. New TTH/FFH output format.

Table 1.17: JSON input path: /OUTPUT/PRINT/VOLTAGE

Print Currents

The current values of any component (defined under `/COMPONENTS`) can be exported to a text file. **The exported current values represent the positive current direction through the components, i.e. current flows from the positive terminal to the negative terminal**.

```
"OUTPUT": {
   "PRINT": {
      "CURRENT": {
         "COMPONENTS" = ["P1", "P2"],
         "FILE" = "currents.txt",
         "FORMAT": "TYPE_1"
      }
   }
}
```

Listing 1.18: JSON input path: `/OUTPUT/PRINT/CURRENT`

Key	Value	Req.	Description
`COMPONENTS`	*Array of Strings*	No	Array of component names. The current through these components will be exported to a text file. These names must correspond to the component names defined under `/COMPONENTS`, see section 1.7. If this key is not defined, the currents through all components will be exported to the text file.
`FILE`	*String*	Yes	The name of the output file.

`FORMAT`	*Fixed String:*
	`"TYPE_1"`
	`"TYPE_2"`
		No	The output format. (default: `"TYPE_1"`)
			Original TTH/FFH output format.
			New TTH/FFH output format.

Table 1.18: JSON input path: `/OUTPUT/PRINT/CURRENT`

Print Charges

The total charge on any conductor (defined under `/CONDUCTORS`) can be exported to a text file.

```
"OUTPUT": {
   "PRINT": {
      "CHARGE": {
         "CONDUCTORS" = ["C1", "C2"],
         "FILE" = "charges.txt"
         "FORMAT": "TYPE_1"
      }
   }
}
```

Listing 1.19: JSON input path: `/OUTPUT/PRINT/CHARGE`

Key	Value	Req.	Description
`CONDUCTORS`	*Array of Strings*	No	Array of conductor names. The charges on these conductors will be exported to a text file. These names must correspond to the conductors names defined under `/CONDUCTORS`, see section 1.8. If this key is not defined, the charges on all conductors will be exported to the text file.
`FILE`	*String*	Yes	The name of the output file.

`FORMAT`	*Fixed String:*
	`"TYPE_1"`
	`"TYPE_2"`
		No	The output format. (default: `"TYPE_1"`)
			Original TTH/FFH output format.
			New TTH/FFH output format.

Table 1.19: JSON input path: `/OUTPUT/PRINT/CHARGE`

Print Power

The complex power (unit: volt-amps) of the entire layout can be exported to a text file. If PHASE analysis is used, see section 1.12.1, the energy (unit: joules) of the entire superconducting structure will be exported to the text file.

```
"OUTPUT": {
   "PRINT": {
      "POWER": {
         "FILE" = "power.txt",
         "FORMAT": "TYPE_1",
         "ANALYSE_FIELDS": "Bz"
      }
   }
}
```

Listing 1.20: JSON input path: `/OUTPUT/PRINT/POWER`

Key	Value	Req.	Description
`FILE`	*String*	Yes	The name of the output file.

`FORMAT`	*Fixed String:*
	`"TYPE_1"`
	`"TYPE_2"`
		No	The output format. (default: `"TYPE_1"`)
			Original TTH/FFH output format.
			New TTH/FFH output format.

`ANALYSE_FIELDS`	*Array of Strings*	No	Array of field names. These names must correspond to the fields names defined under `/FIELDS`, see section 1.9. If this key is defined, the total energy/power of each excitation/component will be recalculted, but with the specified fields also activated. This total energy/power is calculated as $E = E_V + E_H + E_M$, where E_V is the energy/power of the excitation/component, E_H is the energy/power of the applied field and E_M is the energy/power of the interaction between excitation/component and the applied field.

Table 1.20: JSON input path: `/OUTPUT/PRINT/POWER`

Print Threshold Power

The scaled *threshold power* of each component (defined under `/COMPONENTS`) can be exported to a text file for each applied field (defined under `/FIELDS`). The *threshold power* is scaled according to the field's magnitude and will be referred to as the *threshold scale* (S_C) [unit: dimensionless]. If the energy/power of a trapped fluxon is represented by E_V and the energy/power of an applied field is E_H, the total energy/power of both the fluxon and the applied field will be equal to $E = E_V + E_H + E_M$, where E_M is the energy/power of the interaction between fluxon and the applied field. The *threshold power* represents the point at which the activation of the fluxon does not change the total energy of the system, i.e. the point at which $E_V + E_M = 0$. The *threshold scale* can therefore be calculated as $S_C = -E_V/E_M$. To obtain the *threshold field* (B_C), simply multiply the magnitude of the applied field ($|B|$) with the *threshold scale*: $B_C = S_C|B|$.

```
"OUTPUT": {
   "PRINT": {
      "POWER_THRESHOLD": {
         "FILE": "power.txt",
         "FORMAT": "TYPE_1",
         "ANALYSE_FIELDS": "Bz"
      }
   }
}
```

Listing 1.21: JSON input path: `/OUTPUT/PRINT/POWER_THRESHOLD`

Key	Value	Req.	Description
FILE	*String*	Yes	The name of the output file containing the *threshold scale* (S_C) of each component.
FORMAT	*Fixed String:* "TYPE_1" "TYPE_2"	No	The output format. (default: "TYPE_1") Original TTH/FFH output format. New TTH/FFH output format.
ANALYSE_FIELDS	*Array of Strings*	Yes	Array of field names. These names must correspond to the fields names defined under /FIELDS, see section 1.9. The *threshold scale* (S_C) of each component will be calculated for each of the specified fields.

Table 1.21: JSON input path: /OUTPUT/PRINT/POWER_THRESHOLD

1.11.2 Plot

```
"OUTPUT": {
   "PLOT": {
      "J": { },
      "M": { },
      "Q": { },
      "H": { },
      "B": { },
      "E": { },
      "D": { }
   }
}
```

Listing 1.22: JSON input path: /OUTPUT/PLOT

Key	Value	Req.	Description
J	*Object*	No	Plot electric current density of to VTK file. Mode MQS, EMQS or FULL must be enabled. [Unit: A/m^2]
M	*Object*	No	Plot magnetic current density of to VTK file. Mode MQS, EMQS or FULL must be enabled. [Unit: V/m^2]
Q	*Object*	No	Plot electric charge density to VTK file. Mode EQS must be enabled. [Unit: C/m^2]
H	*Object*	No	Plot magnetic field to VTK file. Mode MQS, EMQS or FULL must be enabled. [Unit: A/m]
B	*Object*	No	Plot magnetic flux density to VTK file. Mode MQS, EMQS or FULL must be enabled. [Unit: T]
E	*Object*	No	Plot electric field to VTK file. Mode EQS, MQS, EMQS or FULL must be enabled. [Unit: V/m]
D	*Object*	No	Plot electric displacement field to VTK file. Mode EQS, MQS, EMQS or FULL must be enabled. [Unit: C/m^2]

Table 1.22: JSON input path: `/OUTPUT/PLOT`

Plot Current Density

The electric current density (J) of all current-carrying volumes can be exported to a VTK file. Mode MQS, EMQS or FULL must be enabled. The settings described below are identical for plotting the magnetic current density (M).

```
"OUTPUT": {
   "PLOT": {
      "J": {
         "FILE" = "J.vtk",
         "FORMAT": "VTK"
      }
   }
}
```

Listing 1.23: JSON input path: `/OUTPUT/PLOT/J`

Key	Value	Req.	Description
FILE	*String*	Yes	The name of the output VTK file.
FORMAT	*Fixed String:* "VTK" "FASTHENRY"	No	The output format. (default: "VTK") VTK file format. Original FastHenry current density format. This format will only export the current density of filaments.

Table 1.23: JSON input path: `/OUTPUT/PLOT/J`

Plot Charge Density

The electric charge density (Q) of all conductors and dielectrics surfaces can be exported to a VTK file. Mode EQS must be enabled.

```
"OUTPUT": {
   "PLOT": {
      "Q": {
         "FILE" = "Q.vtk"
      }
   }
}
```

Listing 1.24: JSON input path: `/OUTPUT/PLOT/Q`

Key	Value	Req.	Description
`FILE`	*String*	Yes	The name of the output VTK file.

Table 1.24: JSON input path: `/OUTPUT/PLOT/Q`

Plot Magnetic Flux Density

The magnetic flux density (B) can be calculated over a user-defined mesh entity and exported to a VTK file. Mode MQS, EMQS or FULL must be enabled. The settings described below are identical for plotting the H-field.

```
"OUTPUT": {
   "PLOT": {
      "B": {
         "FILE" = "B.vtk",
         "ENTITIES" = ["MyFliedMesh::Rect", "MyFliedMesh::Box"]
      }
   }
}
```

Listing 1.25: JSON input path: `/OUTPUT/PLOT/B`

Key	Value	Req.	Description
ENTITIES	*Array of Strings*	Yes	Array of mesh entity names. The magnetic flux density will be calculated over these mesh entities. The names must correspond to the mesh entities defined under /MESHES, see section 1.6.2. Each string value must first define the mesh name and then the mesh entity, separated by "::" , e.g. "MyFliedMesh::Box" points to /MATERIALS/MyFliedMesh/ENTITIES/Box.
FILE	*String*	Yes	The name of the output VTK file.

Table 1.25: JSON input path: /OUTPUT/PLOT/B

Plot Electric Field

The electric field can be calculated over a user-defined mesh entity and exported to a VTK file. Mode EQS, MQS, EMQS or FULL must be enabled. The settings described below are identical for plotting the D-field.

```
"OUTPUT": {
   "PLOT": {
      "E": {
         "FILE" = "E.vtk",
         "ENTITIES" = ["MyFliedMesh::Rect", "MyFliedMesh::Box"]
      }
   }
}
```

Listing 1.26: JSON input path: /OUTPUT/PLOT/E

Key	Value	Req.	Description
ENTITIES	*Array of Strings*	Yes	Array of mesh entity names. The electric field will be calculated over these mesh entities. The names must correspond to the mesh entities defined under /MESHES, see section 1.6.2. Each string value must first define the mesh name and then the mesh entity, separated by "::" , e.g. "MyFliedMesh::Box" points to /MATERIALS/MyFliedMesh/ENTITIES/Box.
FILE	*String*	Yes	The name of the output VTK file.

Table 1.26: JSON input path: /OUTPUT/PLOT/E

1.11.3 Compact Model

A compact model of the extracted results can be generated if mode MQS is enabled, see fig. 1.4. This compact model can then be exported to a JoSIM/JSIM/WRPSICE subcircuit. The pins of the subcircuit can be connected to specified excitation ports (voltage sources), defined under `/COMPONENTS`. Excitation ports (voltage sources) can also be replaced with Josephson Junctions within the subcircuit.

```
"OUTPUT": {
   "MODEL": {
      "FILE": "dcsfq.cir",
      "SUBCKT": "dcsfq",
      "FORMAT": "JOSIM",
      "PORTS": ["PVDD", "Pa", "Pq", "Moat1"],
      "FIELDS": ["Bx", "By", "Bz"],
      "JUNCTIONS": ["J1", "J2", "J3"],
      "JJ_MODEL": "jjmit",
      "JJ_SCALE": 1E12,
      "THRESHOLD_K": 1E-4,
      "COMMANDS": [".model jjmit jj(rtype=1, vg=2.8mV, cap=0.07pF,
         ↪  r0=160, rn=16, icrit=0.1mA)"]
   }
}
```

Listing 1.27: JSON input path: `/OUTPUT/MODEL`

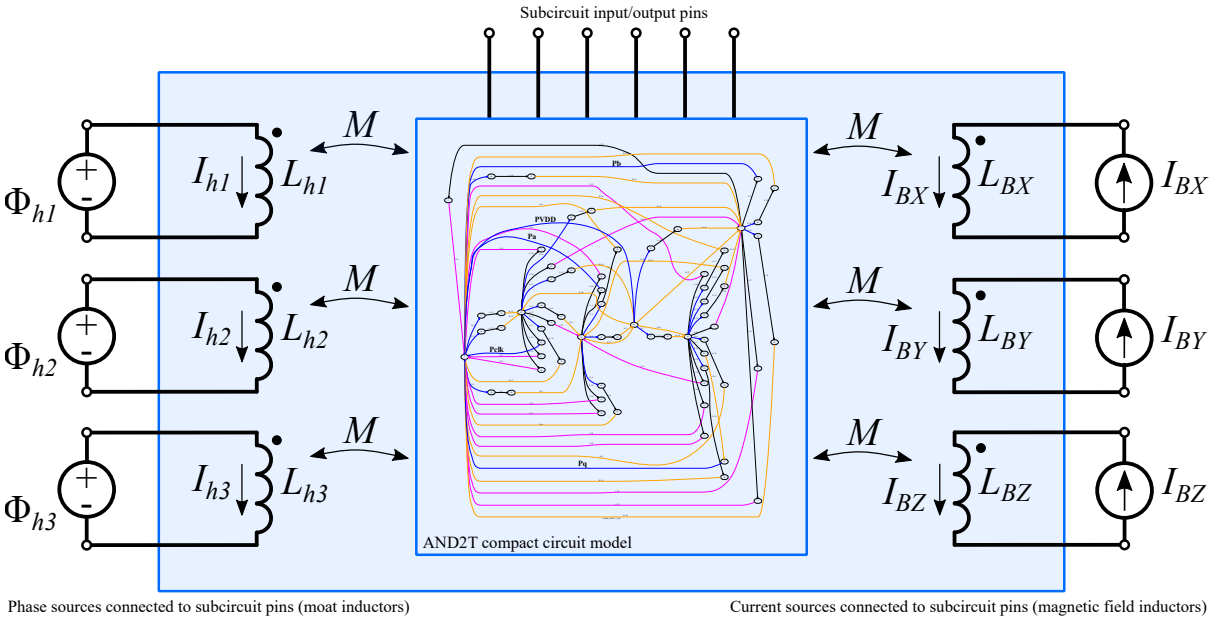


Figure 1.4: Example of a subcircuit model with moat (fluxon) and external magnetic field inductors. These inductors couple to all the inductors of the compact circuit model.

Key	Value	Req.	Description
FILE	*String*	Yes	The name of the output file.
SUBCKT	*String*	Yes	Name of the subcircuit.
FORMAT	*Fixed Strings* JOSIM JSIM WRSPICE	No	Format of the subcircuit. (default: "JoSIM") Compatible with JoSIM simulator. Compatible with JSIM simulator. Compatible with WRSpice simulator.
PORTS	*Array of String:*	No	Array of component names. The voltage/phase sources will be removed and thier positive and negative terminals will be connected to the pins of the subcircuit. Only components defined as voltage/phase sources under /COMPONENTS, see section 1.7, will be accepted.
JUNCTIONS	*Array of Strings*	No	Array of component names. The voltage or phase sources will be removed and thier positive and negative terminals will be connected to Josephson junctions within the subcircuit.Only components defined as voltage/phase sources under /COMPONENTS, see section 1.7, will be accepted.
FIELDS	*Array of Strings*	No	Array of field names. Each field is modelled as an inductor in the subcircuit. The positive and negative terminals of each field inductor are connected to the subcircuit pins. Each field can be activated by connecting a current source to the pins of the corresponding field inductor. The amplitude of the field can be adjusted by adjusted the magnitude of the current through the field inductor (a current of 1 A will produce the same field magnitude specfied in /FIELDS). The field names must correspond to the field names defined under /FIELDS, see section 1.9.
JJ_MODEL	*String*	No	The model name of the Josephson junctions. Only one junction model is allowed. The properties of the junction model can be defined with the COMMANDS key and the .model control. The model parameters of each junction will be scaled according to the area of the junction's terminals. The area of the junction can be scaled using JJ_{SCALE}. The final area is calculated as $area = A_{term} \times JJ_{SCALE}$, where A_{term} is the are of the terminal in μm^2.
JJ_SCALE	*Number*	No	Linear scale of junction area, JJ_{SCALE}. (default: 1.0)
COMMANDS	*Array of Strings*	No	A list of string controls that will be included at the end of the subcircuit.

Table 1.27: JSON input path: /OUTPUT/MODEL (Part 1)

Key	Value	Req.	Description
THRESHOLD_K	*Number*	No	The threshold of magnetic coupling values. If the absolute value of a magnetic coupling is below this threshold, it will be ignored and not included in the subcircuit. (default: 1E-4)
SVD_METHOD	*Fixed Strings*	No	SVD method for calculating compact model solutions. (default: "BDC")
	BDC		Recursive divide and conquer strategy which makes if very fast for large problems.
	JACOBI		Jacobi is numerically very accurate, but very slow for large problems.
	SPARSEQR		Sparse QR factorization.

Table 1.28: JSON input path: /OUTPUT/MODEL (Part 2)

1.12 Settings

```
"SETTINGS": {
   "MODE": {},
   "FREQUENCY": {},
   "PATHS": {},
   "FMM": {},
   "SOLVER": {},
   "QUADRATURE": {},
   "PRECONDITIONER": {},
   "JUNCTION": {}
}
```

Listing 1.28: JSON input path: /SETTINGS

Key	Value	Req.	Description
MODE	*Object*	Yes	Contains simulation mode settings.
FREQUENCY	*Object*	No	Contains frequency settings AC analysis.
PATHS	*Object*	No	Contains paths to license file and other binaries.
FMM	*Object*	No	Contains Fast Multipole Method (FMM) settings.
SOLVER	*Object*	No	Contains solver (GMRES, etc.) settings.
QUADRATURE	*Object*	No	Contains numerical quadrature settings.
PRECONDITIONER	*Object*	No	Contains preconditioner settings.

Table 1.29: JSON input path: /SETTINGS

1.12.1 Mode

```
"SETTINGS": {
   "MODE": {
```

```
3         "TYPE": "MQS",
4         "ANALYSIS": "AC"
5     }
6 }
```

Listing 1.29: JSON input path: `/SETTINGS/MODE`

Key	Value	Req.	Description
`TYPE`	*Fixed String:*	Yes	Specify simulation type (default: `"MQS"`).
	`"EQS"`		Electro-quasistatic: For capacitacne extraction.
	`"MQS"`		Magneto-quasistatic: For inductance and resistance extraction.
	`"EMQS"`		Electro-magneto-quasistatic: For impedance extraction.
	`"FULL"`		Full-wave: For full-wave impedance extraction.
`ANALYSIS`	*Fixed String:*	Yes	Specify analysis mode. (default: `"AC"`)
	`"AC"`		Frequency-based analysis. Works with MQS, EMQS and FULL modes.
	`"DC"`		DC-based analysis. Works with EQS and MQS modes.
	`"PHASE"`		Phase-based analysis. Works with MQS mode (no resistors allowed).

Table 1.30: JSON input path: `/SETTINGS/MODE`

1.12.2 Frequency

Frequency can be specified when using AC analysis, see section 1.12.1. A frequency sweep (section 1.12.2) or an array of frequencies (section 1.12.2) can be specified. The default frequency is 10 GHz. If `DC` or `PHASE` analysis is used, the frequency will be set to 0 Hz.

Frequency Sweep

```
1 "SETTINGS": {
2     "FREQUENCY": {
3         "START": 1E9,
4         "STOP": 10E9,
5         "NUM": 10,
6         "LOGSCALE": false
7     }
8 }
```

Listing 1.30: JSON input path: `/SETTINGS/FREQUENCY`

Key	Value	Req.	Description
START	*Number*	Yes	Start frequency. (default: 10E9)
STOP	*Number*	Yes	Stop frequency. (default: 10E9)
NUM	*Integer*	No	Number of steps between start and stop frequencies. (default: 1)
LOGSCALE	*Boolean*	No	Use log-scale steps between start and stop frequencies. (default: false)

Table 1.31: JSON input path: /SETTINGS/FREQUENCY

Frequency Array

```
"SETTINGS": {
   "FREQUENCY": [1E9,2E9,3E9,4E9,5E9,6E9,7E9,8E9,9E9,10E9]
}
```

Listing 1.31: JSON input path: /SETTINGS/FREQUENCY

Key	Value	Req.	Description
-	*Array*	Yes	Array of frequencies.

Table 1.32: JSON input path: /SETTINGS/FREQUENCY

1.12.3 Paths

```
"SETTINGS": {
   "PATHS": {
      "OUTPUT": "output",
      "LICENSE": "ix_license.txt"
   }
}
```

Listing 1.32: JSON input path: /SETTINGS/PATHS

Key	Value	Req.	Description
OUTPUT	*String*	No	Path to output directory. (default: "output")
LICENSE	*String*	No	Path to license file. (default: "ix_license.txt")

Table 1.33: JSON input path: /SETTINGS/PATHS

1.12.4 Fast Multipole Method (FMM)

```
"SETTINGS": {
   "FMM": {
      "ORDER": 2,
```

```
        "THREADS": "MAX",
        "DEPTH": 6,
        "MAXDEPTH": 10,
        "NNBRS": 2,
        "ADAPTIVE": true
    }
}
```

Listing 1.33: JSON input path: `/SETTINGS/FMM`

Key	Value	Req.	Description
`THREADS`	*Integer*	No	Number of parallel threads used when constructing FMM matrices. (default: maximum)
`ORDER`	*Integer*	No	Expansion order. (default: 2)
`NNBRS`	*Integer*	No	Number of neighboring cubes used for direct interactions. (default: 2)
`DEPTH`	*Integer*	No	Fixed depth of the FMM octree. This setting will be ignored if `ADAPTIVE` = true. (default: automatic)
`MAXDEPTH`	*Integer*	No	Maximum depth of the FMM octree. (default: 10)
`ADAPTIVE`	*Boolean*	No	If true, determine the optimal depth of the FMM octree automatically. If false, the value of `DEPTH` will be for the depth of the FMM octree. (default: true)

Table 1.34: JSON input path: `/SETTINGS/FMM`

1.12.5 Solver

```
"SETTINGS": {
    "SOLVER": {
        "TYPE": "GMRES",
        "TOL": 1E-4,
        "ABSTOL": 1E-3,
        "THREADS": "MAX",
        "MAXITER": 200,
        "MINRESTART": 100
    }
}
```

Listing 1.34: JSON input path: `/SETTINGS/SOLVER`

Key	Value	Req.	Description
TYPE	*Fixed String:* "GMRES" "FGMRES" "DIRECT"	No	Iterative solver type. (default: "GMRES") Use GMRES iterative solver. Use flexible GMRES iterative solver. Use direct solver (Warning: requires a lot of memory).
TOL	*Number*	No	Mininum relative tolerance of solution. (default: 1E-4)
ABSTOL	*Number*	No	Solution absolute tolerance of solution. (default: 1E-4)
THREADS	*Integer*	No	Maximum parallel threads. (default: maximum)
MAXITER	*Integer*	No	Maximum solver iterations. (default: 100)
MINRESTART	*Integer*	No	Restart iterations of FGMRES solver. (default: 100)

Table 1.35: JSON input path: /SETTINGS/SOLVER

1.12.6 Numerical Quadrature

```
"SETTINGS": {
   "QUADRATURE": {
      "SURFACE": [1, 6],
      "VOLUME": [1, 6]
   }
}
```

Listing 1.35: JSON input path: /SETTINGS/QUADRATURE

Key	Value	Req.	Description
SURFACE	*Array of Integer*	No	Array of two integer values. The two values respectively specify the lowest and highest quadrature orders for the surface numerical integration. (default: [1, 6])
VOLUME	*Array of Integer*	No	Array of two integer values. The two values respectively specify the lowest and highest quadrature orders for the volume numerical integration. (default: [1, 6])

Table 1.36: JSON input path: /SETTINGS/QUADRATURE

1.12.7 Preconditioner

The GMRES and FGMRES iterative solvers use a preconditioning method in order to speed up convergence. The settings shown in table 1.37 and table 1.38 correspond to some of the options available for the LU factorization routines in the SuperLU library [6].

```
"SETTINGS": {
   "PRECONDITIONER": {
      "ENABLED": true,
      "TYPE": "ILU",
      "SPARSITY": "DIAGONAL",
      "EQUIL": false,
      "COLPERM": "MMD_AT_PLUS_A",
      "ITERREFINE": "NOREFINE",
      "SYMMETRICMODE": false,
      "SYMPATTERN": false,
      "PIVOTGROWTH": true,
      "CONDITIONNUMBER": true,
      "PRINTSTAT": true,
      "DIAGPIVOTTHRESH": 0.1,
      "FILLRATIO": 30,
      "ILU_DROPRULE": ["BASIC", "AREA"],
      "ILU_DROPTOL": 1E-4,
      "ILU_FILLFACTOR": 10.0,
      "ILU_NORM": "INF_NORM",
      "ILU_MILU": "SILU",
      "ILU_MILU_DIM": 3.0,
      "ILU_FILLTOL": 1E-2
   }
}
```

Listing 1.36: JSON input path: `/SETTINGS/PRECONDITIONER`

Key	Value	Req.	Description
ENABLED	*Boolean*	No	If true, enable preconditioning method. (default: true)
TYPE	*Fixed String:*	No	Type of factorization of preconditioner matrix. (default: "ILU" for MQS, "ILU"/"LOCAL_INVERSE" for EQS, "LU" for EMQS/FULL)
	"LU"		Use LU factorization.
	"ILU"		Use incomplete LU factorization (ILU).
	"LOCAL_INVERSE"		Preconditioner is formed by inverting a sequence of reduced matrices, associated with the leaf-level cubes of the FMM octree [7].
SPARSITY	*Fixed String:*	No	Sparsity pattern of the preconditioner matrix. (default: "DIAGONAL" for MQS/EMQS/FULL, "CUBE" for EQS)
	"DIAGONAL"		Preconditioner matrix contains diagonal entries.
	"CUBE"		Preconditioner matrix contains near-field interactions between elements.
COLPERM	*Fixed String:*	No	Type of ordering for the columns of the preconditioner matrix [6]. (default: "MMD_AT_PLUS_A")
	"NATURAL"		Natural ordering [6].
	"MMD_ATA"		Multiple Minimum Degree (MMD) applied to the structure of A^TA [6].
	"MMD_AT_PLUS_A"		Multiple Minimum Degree (MMD) applied to the structure of $A^T + A$ [6].
	"COLAMD"		Column Approximate Minimum Degree [6].
ITERREFINE	*Fixed String:*	No	Choose iterative refinement method. (default: "NOREFINE")
	"NOREFINE"		Disable iterative refinement.
	"SINGLE"		Single refinement [6].
	"DOUBLE"		Double refinement [6].
	"EXTRA"		Maximum refinement [6].
EQUIL	*Boolean*	No	If true, equilibrate the matrix [6]. (default: false)
SYMMETRICMODE	*Boolean*	No	If true, use the symmetric mode [6]. (default: false)
SYMPATTERN	*Boolean*	No	If true, assume matrix has the symmetric pattern [6]. (default: false)
PIVOTGROWTH	*Boolean*	No	If true, compute the reciprocal pivot growth [6]. (default: true)
CONDITIONNUMBER	*Boolean*	No	If true, compute reciprocal condition number [6]. (default: true)
PRINTSTAT	*Boolean*	No	If true, print statistics [6]. (default: true)

Table 1.37: JSON input path: /SETTINGS/PRECONDITIONER (Part1)

Key	Value	Req.	Description
DIAGPIVOTTHRESH	*Number*	No	Acceptable diagonal pivot threshold [6]. [min,max] = [0.0, 0.1] (default: 0.1)
FILLRATIO	*Number*	No	An estimated of the number of non-zeros for L and U, compared to the non-zeros in the pre-conditioner matrix [6]. (default: 30)
ILU_DROPRULE	*Fixed String Array:* "NODROP" "BASIC" "PROWS" "COLUMN" "AREA" "DYNAMIC" "INTERP" "SECONDARY"	No	List of dropping rules for ILU [6]. (default: ["BASIC", "AREA"]) No dropping rule [6]. Basic dropping rule [6]. Supernodal based ILUTP [6]. Variant of ILUTP [6]. Variant of ILUTP [6]. Dynamic adjusted threshold [6]. Interpolation mode [6]. ["PROWS", "COLUMN", "AREA"].
ILU_DROPTOL	*Number*	No	Dropping threshold for ILU [6]. [min,max] = [0.0, 1.0] (default: 1E-4)
ILU_FILLFACTOR	*Number*	No	Fill ratio upper bound for ILU [6]. (default: 10.0)
ILU_NORM	*Fixed String:* "ONE_NORM" "TWO_NORM" "INF_NORM"	No	Norm used for ILU. (default: "INF_NORM") Infinity-norm [6]. 1-norm [6]. 2-norm [6].
ILU_MILU	*Fixed String:* "SILU" "SMILU_1" "SMILU_2" "SMILU_3"	No	Version of modified ILU [6]. (default: "SILU") Disable MILU[6]. MILU type 1 [6]. MILU type 2 [6]. MILU type 3 [6].
ILU_MILU_DIM	*Number*	No	Dimension of PDE for ILU [6]. (default: 3.0)
ILU_FILLTOL	*Number*	No	Zero pivot perturbation threshold for ILU [6]. (default: 1E-2)

Table 1.38: JSON input path: /SETTINGS/PRECONDITIONER (Part2)

References

[1] K. Jackman and C. J. Fourie, "Tetrahedral modeling method for inductance extraction of complex 3-d superconducting structures," *IEEE Transactions on Applied Superconductivity*, vol. 26, no. 3, pp. 1–5, Apr. 2016. DOI: 10.1109/TASC.2016.2522299.

[2] M. Kamon, M. J. Tsuk, and J. K. White, "Fasthenry: A multipole-accelerated 3-d inductance extraction program," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 9, pp. 1750–1758, Sep. 1994. DOI: 10.1109/22.310584.

[3] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009. DOI: 10.1002/nme.2579. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579.

[4] M. Kamon, L. Silveira, C. Smithhisler, and J. White, "Fasthenry user's guide," *Research Laboratory of Electronics, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, 1996.

[5] U. Ayachit, *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.

[6] X. S. Li, J. W. Demmel, J. R. Gilbert, L. Grigori, M. Shao, and I. Yamazaki, "Superlu users' guide," *Lawrence Berkeley National Laboratory*, 1999.

[7] K. Nabors, S. Kim, and J. White, "Fast capacitance extraction of general three-dimensional structures," *IEEE Transactions on Microwave Theory and Techniques*, vol. 40, no. 7, pp. 1496–1506, 1992. DOI: 10.1109/22.146331.