Finite Elements coding with python: Electromagnetics

Presentation · May 2022

DOI: 10.13140/RG.2.2.15505.51040

1 author:

Guillaume Verez
Sonceboz SA
21 PUBLICATIONS **187** CITATIONS

SEE PROFILE

Finite Elements coding

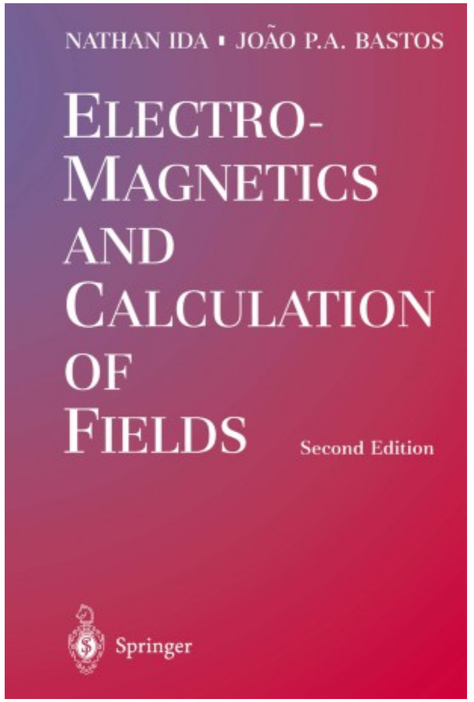
Electromagnetics

Guillaume VEREZ, 2022

Reference

Electromagnetics and calculation of fields

I would like to thank the authors for writing this very practical handbook. Since I have no authorization to copy any material, I will only develop the first example. I suggest anyone interested in developing further the work I'm presenting to buy a copy or find one in a library.



Nathan Ida
Uni. Akron
USA

Joao P.A. Bastos
Uni. Santa Catharina
Brazil

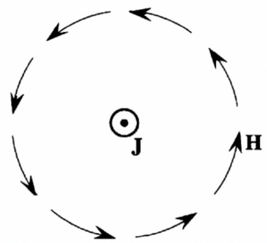
Springer-Verlag
New York 1997

Maxwell's Equations

For a general material

2.1 $$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$$

Expresses the manner by which a magnetic field **H** can create a split into conduction current (associated with **J**) and a time variation of the electric flux density (associated with d**D**/dt).

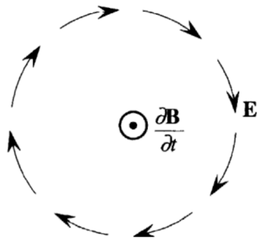


2.2 $$\nabla \cdot \mathbf{B} = 0$$

The magnetic flux is conservative (the magnetic flux entering a volume is equal to the magnetic flux leaving the volume).

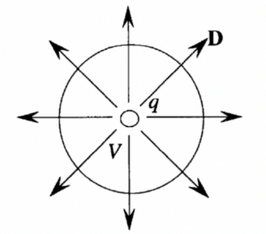
2.3 $$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

Analogous to 2.1, showing that the time derivative of the magnetic flux density **B** is capable of generating an electric field intensity **E**.



2.4 $$\nabla \cdot \mathbf{D} = \rho$$

The electric flux is not conservative (we can easily imagine a volume in which there is a difference between the electric fluxes entering and leaving the volume).



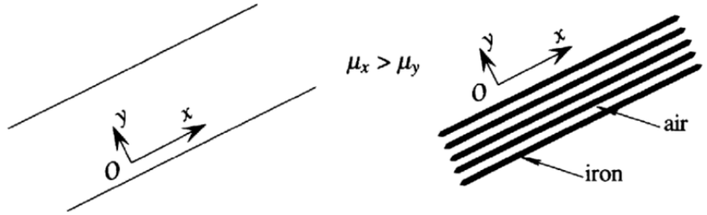
pp. 30-37

Constitutive Equations

For a general material

2.18 $$\mathbf{B} = \|\mu\|\mathbf{H}$$

Introducing the concept of magnetic anisotropy, for a material whose magnetic permeability μ is dominant in a certain direction. For example in a sheet of iron with grain-oriented structure (left) or with a stack of thin insulated sheets (right).



2.19 $$\mathbf{D} = \|\varepsilon\|\mathbf{E}$$

The anisotropy concept can be extended in a similar manner to the electric permittivity ε…

2.20 $$\mathbf{J} = \|\sigma\|\mathbf{E}$$

… and to conductivity σ. In both cases, the nonlinear behavior is normally negligible.

pp. 30-37

Approximations to Maxwell's Equations

Based on the conditions under which we anticipate operating

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \cdot \mathbf{D} = \rho$$

In general media

$$\nabla \times \mathbf{H} = \mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}$$

$$\nabla \cdot \mathbf{H} = 0$$

$$\nabla \times \mathbf{E} = -\mu_0 \frac{\partial \mathbf{H}}{\partial t}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}$$

In media with $\varepsilon=\varepsilon_0$ and $\mu=\mu_0$

The time derivative of the magnetic field intensity is responsible for the generation of **E**.

$$\nabla \times \mathbf{H} = \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}$$

$$\nabla \cdot \mathbf{H} = 0$$

$$\nabla \times \mathbf{E} = -\mu_0 \frac{\partial \mathbf{H}}{\partial t}$$

$$\nabla \cdot \mathbf{E} = 0$$

In vacuum

There are no molecules and there can be no physical support for charges and conduction currents. In addition, the permittivity and permeability can be considered to be constants.

Approximations to Maxwell's Equations

Based on the conditions under which we anticipate operating

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \cdot \mathbf{D} = \rho$$

In general media

The term d**D**/dt is related to "displacement currents". At frequencies of 10^5 Hz with an electric field density of 10^5 V/m, this displacement current density is of the order of 10^{-4} A/mm^2, far smaller than conduction current densities, in the order of 1 A/mm^2.

Magnetostatics

$$\nabla \times \mathbf{H} = \mathbf{J}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\mathbf{B} = \|\mu\|\mathbf{H}$$

$$\mathbf{J} = \|\sigma\|\mathbf{E}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

E generated by a magnetic field

Magnetodynamics

$$\nabla \times \mathbf{E} = 0$$

$$\nabla \cdot \mathbf{D} = \rho$$

E generated by the presence of static charges

$$\mathbf{D} = \|\varepsilon\|\mathbf{E}$$

Electrostatics

The neglection of d**D**/dt in the first equation allows decoupling of the system of equations into two parts from which we obtain two systems of equations that can be studied separately.

Electrostatics

Scalar potential

Electrostatics

Scalar potential

Assuming that in the domain under study there are no time-dependent quantities, we can define an electric scalar potential V from which a conservative field intensity can be derived:

$$\mathbf{E} = -\nabla \mathrm{V}$$

This relation is valid for the electrostatic field because:

$$\nabla \times \mathbf{E} = 0$$

And since the curl of the divergence of any scalar function is always zero, this definition is correct.

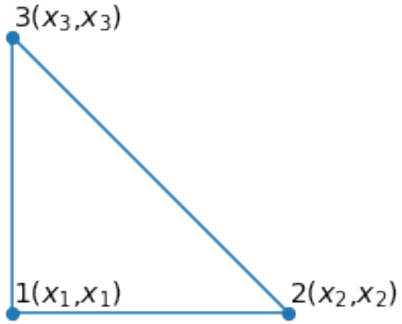
$$\nabla \times (-\nabla \mathrm{V}) = 0$$

For the rest of this electrostatics part: in the assumption that in dielectric materials we can use a scalar permittivity ε, the relation between D and E can be expressed with the use of the permittivity of free space ε_0 and the relative permittivity ε_r.

$$\mathbf{D} = \varepsilon_0 \varepsilon_r \mathbf{E}$$

Finite element technique

Scalar potential



Using first order triangular elements, we assume that the potential V varies linearly (first order) within the triangle:

$$\mathrm{V}(x,y) = a + bx + cy$$

We thus need to find the three unknown a, b and c. For the nodes i (1, 2, 3 in the figure), the potential at each node is:

$$\mathrm{V}_i(x,y) = a + bx_i + cy_i$$

This gives a set of 3 equations (for each i = 1, 2, 3) that, after solving, allows us to express the geometric evolution of the potential:

$$\mathrm{V}(x,y) = \sum_{i=1,2,3} \frac{1}{\mathrm{D}}(p_i + q_i x + r_i y)\mathrm{V}_i$$

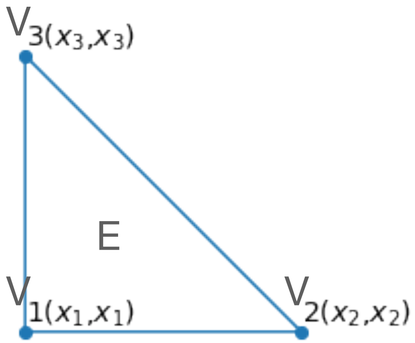
Where:

$$\mathrm{D} = \det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \qquad \begin{cases} p_1 &= x_2 y_3 - x_3 y_2 \\ q_1 &= y_2 - y_3 \\ r_1 &= x_3 - x_2 \end{cases} \qquad \begin{cases} p_2 &= x_3 y_1 - x_1 y_3 \\ q_2 &= y_3 - y_1 \\ r_2 &= x_1 - x_3 \end{cases} \qquad \begin{cases} p_3 &= x_1 y_2 - x_2 y_1 \\ q_3 &= y_1 - y_2 \\ r_3 &= x_2 - x_1 \end{cases}$$

It seems like a lot of parameters, but since x1, y1, x2, y2, x3, y3 are all given by the geometry of the triangle, p1, q1, r1, p2, q2, r2, p3, q3, r3 and D are all known. What it says is that when we create an element, once we find the potential at each node, we will know how the potential will behave on each edge of this element. Note that D is by definition twice the area of the element.

Finite element technique

Electric field



The scalar potential is calculated at each node but the electric field is calculated for each element. Since the potential was defined as $\mathbf{E} = -\nabla \mathrm{V}$

the field can be rewritten in the orthogonal system of coordinates with unit vectors:

$$\mathbf{E} = \mathrm{E}_x \hat{\mathbf{i}} + \mathrm{E}_y \hat{\mathbf{j}}$$

$$\mathbf{E} = -\frac{\partial \mathrm{V}}{\partial x} \hat{\mathbf{i}} - \frac{\partial \mathrm{V}}{\partial y} \hat{\mathbf{j}}$$

Using the results of the previous slide, we can see that the field is constant in an element:

$$\mathrm{E}_x = -\frac{1}{\mathrm{D}} \sum_{i=1,2,3} q_i \mathrm{V}_i \qquad \mathrm{E}_y = -\frac{1}{\mathrm{D}} \sum_{i=1,2,3} r_i \mathrm{V}_i$$

pp. 75-88

Finite element technique

Variational method

We now have to find the potential at each node. Instead of solving directly the physical equation $\nabla^2 \mathrm{V} = 0$, we use the variational method which consists in the minimization of an energy functional. This functional is the energy associated with the electric field intensity in the solution domain where S (= D/2) is the surface of the solution domain:

$$\mathrm{F} = \iint_S \frac{1}{2} \varepsilon \mathrm{E}^2 \mathrm{ds}$$

Minimization of the function is done with respect to the N unknown potentials at the nodes. The energy is minimized when the N equations satisfy the condition:

$$\frac{\partial \mathrm{F}}{\partial \mathrm{V}_n} = 0, n \in [1 \cdots N]$$

The discretization of the domain into M elements allows us to write the energy in the system as the sum of the energy of the M elements:

$$\frac{\partial \mathrm{F}}{\partial \mathrm{V}_n} = \sum_{m=1}^{M} \frac{\partial \mathrm{F}_m}{\partial \mathrm{V}_n}$$

F_m is the energy associated with an element m, which is different from zero in the element m and zero outside this element. Assuming ε and E constant in the element:

$$\begin{aligned} \frac{\partial \mathrm{F}_m}{\partial \mathrm{V}_n} &= \frac{\partial}{\partial \mathrm{V}_n} \iint_S \frac{1}{2} \varepsilon \mathrm{E}^2 \mathrm{ds} \\ &= \frac{1}{2} \varepsilon \frac{\mathrm{D}}{2} \frac{\partial \mathrm{E}_m^2}{\partial \mathrm{V}_n} \end{aligned}$$

Finite element technique

Local matrices

So we need to develop this expression: $\frac{\partial \mathrm{F}_m}{\partial \mathrm{V}_n} = \frac{1}{2}\varepsilon\frac{\mathrm{D}}{2}\frac{\partial \mathrm{E}_m^2}{\partial \mathrm{V}_n}$ using our knowledge on $\mathrm{E_m}$

$$\mathrm{E}_x = -\frac{1}{\mathrm{D}}\sum_{i=1,2,3} q_i \mathrm{V}_i$$

$$\mathrm{E}_y = -\frac{1}{\mathrm{D}}\sum_{i=1,2,3} r_i \mathrm{V}_i$$

1. We express $\mathrm{E_m}$ in orthogonal coordinates, then we develop the square of its modulus.
2. We perform the local derivatives and we obtain for each element:

$$\begin{bmatrix} \frac{\partial \mathrm{F}_m}{\partial \mathrm{V}_1} \\ \frac{\partial \mathrm{F}_m}{\partial \mathrm{V}_2} \\ \frac{\partial \mathrm{F}_m}{\partial \mathrm{V}_3} \end{bmatrix} = \frac{\varepsilon}{2\mathrm{D}} \begin{bmatrix} q_1q_1 + r_1r_1 & q_2q_1 + r_2r_1 & q_3q_1 + r_3r_1 \\ q_1q_2 + r_1r_2 & q_2q_2 + r_2r_2 & q_3q_2 + r_3r_2 \\ q_1q_3 + r_1r_3 & q_2q_3 + r_2r_3 & q_3q_3 + r_3r_3 \end{bmatrix} \begin{bmatrix} \mathrm{V}_1 \\ \mathrm{V}_2 \\ \mathrm{V}_3 \end{bmatrix}$$

[S]
“stiffness” matrix

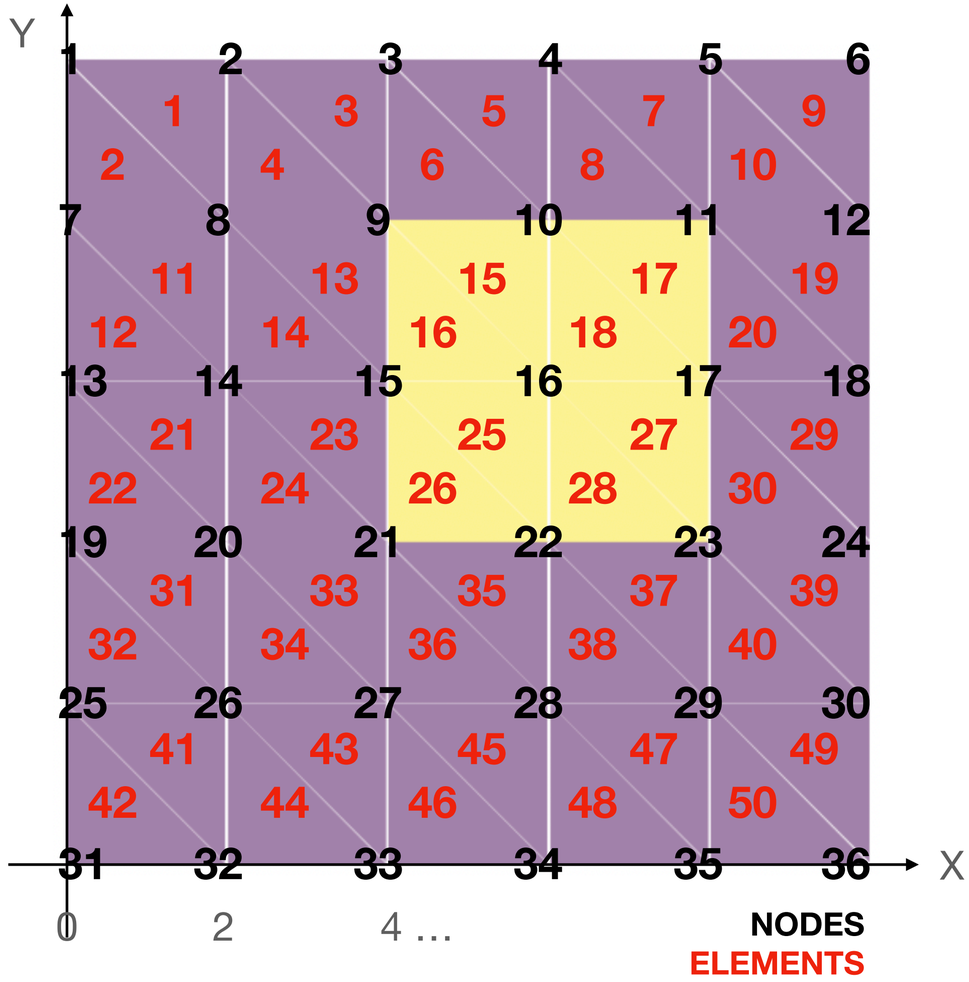
[V]
Vector of unknown potentials

pp. 75-88

Finite element technique

Numerical example

A simple demonstration geometry, 50 elements, 36 nodes, square of material (yellow, 4x4 dimensions) with relative permeability of 5, placed in the air (purple, 10x10 dimensions). The potential is imposed on the top (V = 100) and at the bottom (V = 0).



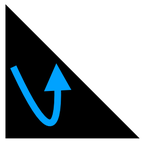
The stiffness matrix of the first element (made of nodes 1, 8, 2) is

$$[S]_1 = \begin{bmatrix} 0.5 & 0 & -0.5 \\ 0 & 0.5 & -0.5 \\ -0.5 & -0.5 & 1 \end{bmatrix}$$

It is the same for every element of air that have the same orientation.

The stiffness matrix of the second element (made of nodes 1, 7, 8) is

$$[S]_2 = \begin{bmatrix} 0.5 & -0.5 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & -0.5 & 0.5 \end{bmatrix}$$



It is the same for every element of air that have the same orientation.

For elements of the yellow material it is just 5x[S] since the only difference is the 5 times higher relative permeability.

Finite element technique

Numerical example

We now have to assemble a “total stiffness matrix”, named [SS] of size NODESxNODES by adding the terms of [S] to the line and column numbers corresponding to the node numbers.

At the first iteration, we place $[S]_1$ in an empty [SS]

$$[S]_1 = \begin{bmatrix} 0.5 & 0 & -0.5 \\ 0 & 0.5 & -0.5 \\ -0.5 & -0.5 & 1 \end{bmatrix}$$

Local numbering of any [S]

$$[S] = \begin{bmatrix} (1,1) & (1,2) & (1,3) \\ (2,1) & (2,2) & (2,3) \\ (3,1) & (3,2) & (3,3) \end{bmatrix}$$

Numbering of $[S]_1$ in global node system

$$[S_1] = \begin{bmatrix} (1,1) & (1,8) & (1,2) \\ (8,1) & (8,8) & (8,2) \\ (2,1) & (2,8) & (2,2) \end{bmatrix}$$

$$[SS] = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots & \text{NODES} \end{matrix} \\ \begin{bmatrix} 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -0.5 & 1 & 0 & 0 & 0 & 0 & 0 & -0.5 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ \dots \\ \text{NODES} \end{matrix}$$

pp. 75-88

Finite element technique

Numerical example

At the second iteration, we place $[S]_2$ in $[SS]$

$$[S]_2 = \begin{bmatrix} 0.5 & -0.5 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & -0.5 & 0.5 \end{bmatrix}$$

Numbering of $[S]_2$ in global node system

$$[S_2] = \begin{bmatrix} (1,1) & (1,7) & (1,8) \\ (7,1) & (7,7) & (7,8) \\ (8,1) & (8,7) & (8,8) \end{bmatrix}$$

$$[SS] = \begin{bmatrix} 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -0.5 & 1 & 0 & 0 & 0 & 0 & 0 & -0.5 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

$$[SS] = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots & \text{NODES} \end{matrix} \\ \begin{bmatrix} 1 & -0.5 & 0 & 0 & 0 & 0 & -0.5 & 0 & \dots & 0 \\ -0.5 & 1 & 0 & 0 & 0 & 0 & 0 & -0.5 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -0.5 & 0 & 0 & 0 & 0 & 0 & 1 & -0.5 & \dots & 0 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & -0.5 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ \dots \\ \text{NODES} \end{matrix} \end{array}$$

pp. 75-88

Code

Initialization

An example code is provided in the reference book. It is in FORTRAN77 but I translated it to Python3. You can find the link to the code on my ResearchGate page, and I am giving now an overview.

First, we need to create the geometry, and thus create a matrix (element_mat) containing all the elements that are defined by three nodes. The nodes coordinates are separated and put in two vectors X and Y.

```
n_elem_x = 5                          # 5 elements in the x and y directions
n_elem_y = 5

permittivity_list = [1, 10]           # permittivities of the outside and the material of the square
imposed_potentials_values = [100, 0] # imposed potential at Y = max and Y = 0

element_mat = []
X = []
Y = []

for i in range(n_elem_y + 1):
    for j in range(n_elem_x + 1):
        if i != n_elem_y and j != n_elem_x:
            element_mat.append([i*(n_elem_x+1)+j, (i+1)*(n_elem_x+1)+1+j, i*(n_elem_x+1)+1+j])
            element_mat.append([i*(n_elem_x+1)+j, (i+1)*(n_elem_x+1)+j, (i+1)*(n_elem_x+1)+1+j])
        X.append(10/n_elem_x*j)
        Y.append(10-10/n_elem_y*i)

number_elements = len(element_mat)
number_nodes = (n_elem_y+1)*(n_elem_x+1)

imposed_potentials_nodes = np.zeros((len(imposed_potentials_values), n_elem_x+1), dtype = int)
for i in range(n_elem_x + 1):
    imposed_potentials_nodes[0][i] = i
    imposed_potentials_nodes[1][i] = number_nodes-1-i
```

```
X
```

```
[0.0,
 2.0,
 4.0,
 6.0,
 8.0,
 10.0,
 0.0,
 2.0,
 4.0,
 6.0,
 8.0,
 10.0,
 0.0,
 ...
```

```
Y
```

```
[10.0,
 10.0,
 10.0,
 10.0,
 10.0,
 10.0,
 8.0,
 8.0,
 8.0,
 8.0,
 8.0,
 8.0,
 6.0,
 ...
```

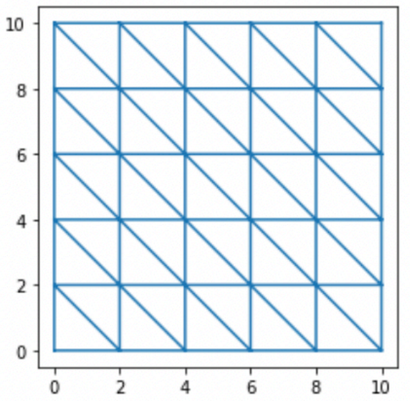
```
element_mat
```

```
[[0, 7, 1],
 [0, 6, 7],
 [1, 8, 2],
 [1, 7, 8],
 [2, 9, 3],
 [2, 8, 9],
 [3, 10, 4],
 [3, 9, 10],
 [4, 11, 5],
 [4, 10, 11],
 [6, 13, 7],
 [6, 12, 13],
 [7, 14, 8],
 ...
```

```
imposed_potentials_nodes
```

```
array([[ 0,  1,  2,  3,  4,  5],
       [35, 34, 33, 32, 31, 30]])
```

```
plt.triplot(X, Y, element_mat)
plt.gca().set_aspect('equal')
```



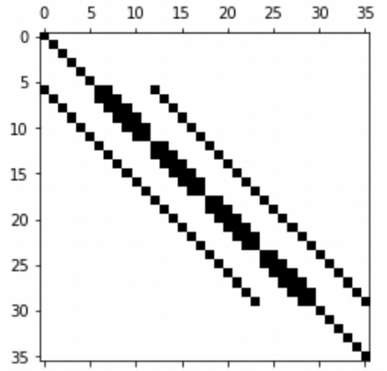
pp. 75-88

Code

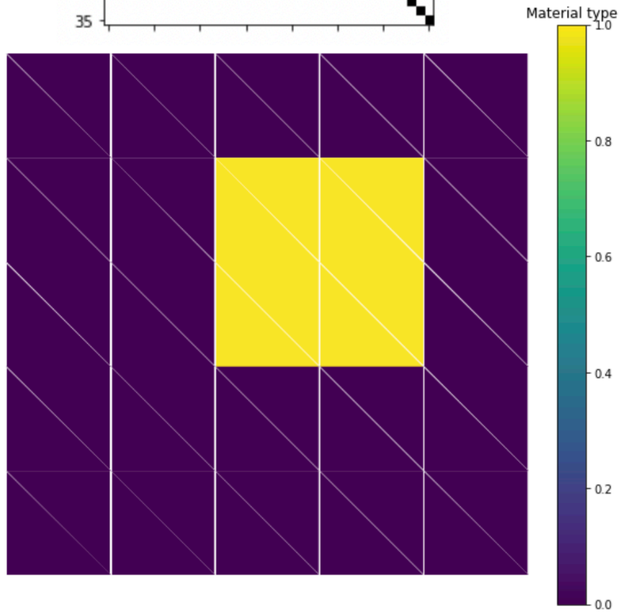
Global stiffness matrix

```
patches = []
color_material = []
globalS = np.zeros((number_nodes, number_nodes))

for i in range(number_elements):
    localS = np.zeros((3, 3))
    current_element_node = [element_mat[i][0], element_mat[i][1], element_mat[i][2]]
    x1 = X[current_element_node[0]]
    y1 = Y[current_element_node[0]]
    x2 = X[current_element_node[1]]
    y2 = Y[current_element_node[1]]
    x3 = X[current_element_node[2]]
    y3 = Y[current_element_node[2]]

    # create visually the element
    polygon = Polygon(np.array([[x1, y1], [x2, y2], [x3, y3]]), False)
    patches.append(polygon)

    # create the material color and assign the element material
    if ((x3 <= 8 and x1 >= 4) and (y1 <= 8 and y2 >= 4)):
        permittivity = permittivity_list[1]
        color_material.append(1.)
    else:
        permittivity = permittivity_list[0]
        color_material.append(0.)

    q1 = y2 - y3
    q2 = y3 - y1
    q3 = y1 - y2
    r1 = x3 - x2
    r2 = x1 - x3
    r3 = x2 - x1
    determinant = x2*y3 + x1*y2 + x3*y1 - x1*y3 - x3*y2 - x2*y1
    localS[0][0] = q1*q1 + r1*r1
    localS[0][1] = q1*q2 + r1*r2
    localS[0][2] = q1*q3 + r1*r3
    localS[1][0] = localS[0][1]
    localS[1][1] = q2*q2 + r2*r2
    localS[1][2] = q2*q3 + r2*r3
    localS[2][0] = localS[0][2]
    localS[2][1] = localS[1][2]
    localS[2][2] = q3*q3 + r3*r3
    localS *= permittivity/(2*determinant)
    # Assemble the localS(3,3) into the global matrix globalS(number_nodes, number_nodes)
    for k in range(3):
        for j in range(3):
            globalS[current_element_node[k]][current_element_node[j]] = globalS[current_element_node[k]][current_element_node[j]] + localS[k][j]
```

```
plt.spy(globalS)
plt.gca().set_aspect('equal')
```



[SS] matrix is diagonal



pp. 75-88

Code

Boundary conditions

Now that the global stiffness matrix is created, we want to solve: $[\mathrm{SS}][\mathrm{V}] = [0]$ (minimization of energy)

However, we want to impose some potential values at the maximum (100) and the minimum (0) Y values. For this, we need to create a “source” vector Q and solve:

$$[\mathrm{SS}][\mathrm{V}] = [\mathrm{Q}]$$

This source vector can, depending on the case, be electric charges, currents or permanent magnets. This vector is called a source vector because it describes the influence of the sources in the element (if any). Generally, these are the sources of the fields - either electric or magnetic, depending on the case.

The Dirichlet boundary conditions can be imposed in a very simple manner using a method based on the following rule: if a value V_n is to be imposed at node n, the diagonal position of row n of the matrix is set to one. All other coefficients in the row are set to zero. In a row n of vector Q, the value of V_n is entered. Performing the product of the vector V with row n of SS gives:

$$0V_1 + 0V_2 + \ldots + 1V_n + \ldots + 0V_{NODES} = V_n$$

```
source_vect = np.zeros(number_nodes)

for i in range(np.size(imposed_potentials_values)): # for all values of imposed potentials in the list
    for j in range(n_elem_x+1): # all the nodes that are on the line where the potential is imposed
        for k in range(number_nodes):
            globalS[imposed_potentials_nodes[i][j]][k] = 0 # zero the coefficients in the corresponding line of globalS
        globalS[imposed_potentials_nodes[i][j]][imposed_potentials_nodes[i][j]] = 1 # then set the diagonal term to 1
        source_vect[imposed_potentials_nodes[i][j]] = imposed_potentials_values[i] # place the imposed potential on the right hand side
```

pp. 75-88

Code

Solving

Scipy is used. We want to solve $[\mathrm{SS}][\mathrm{V}] = [\mathrm{Q}]$ for V, which is equivalent to Ax = b for x.

scipy.sparse.linalg.spsolve(*A, b, permc_spec=None, use_umfpack=True*)

Solve the sparse linear system Ax=b, where b may be a vector or a matrix.

```
potential_vec = scipy.sparse.linalg.spsolve(csc_matrix(globalS), source_vect)
```

```
def electric_field_intensity(element_number):
    # E = - grad(V)
    current_element_node = [element_mat[element_number][0], element_mat[element_number][1], element_mat[element_number][2]]
    x1 = X[current_element_node[0]]
    y1 = Y[current_element_node[0]]
    x2 = X[current_element_node[1]]
    y2 = Y[current_element_node[1]]
    x3 = X[current_element_node[2]]
    y3 = Y[current_element_node[2]]
    q1 = y2 - y3
    q2 = y3 - y1
    q3 = y1 - y2
    r1 = x3 - x2
    r2 = x1 - x3
    r3 = x2 - x1
    potential = [potential_vec[current_element_node[0]], potential_vec[current_element_node[1]], potential_vec[current_element_node[2]]]
    determinant = x2*y3 + x1*y2 + x3*y1 - x1*y3 - x3*y2 - x2*y1
    EX = -(q1*potential[0] + q2*potential[1] + q3*potential[2])/determinant
    EY = -(r1*potential[0] + r2*potential[1] + r3*potential[2])/determinant
    return EX, EY

EMOD_vec = []
for i in range(number_elements):
    EX, EY = electric_field_intensity(i)
    EMOD = np.sqrt(EX**2 + EY**2)
    EMOD_vec.append(EMOD)
```

$$\mathrm{E}_x = -\frac{1}{\mathrm{D}} \sum_{i=1,2,3} q_i \mathrm{V}_i$$

$$\mathrm{E}_y = -\frac{1}{\mathrm{D}} \sum_{i=1,2,3} r_i \mathrm{V}_i$$

We calculate the modulus of E in each element

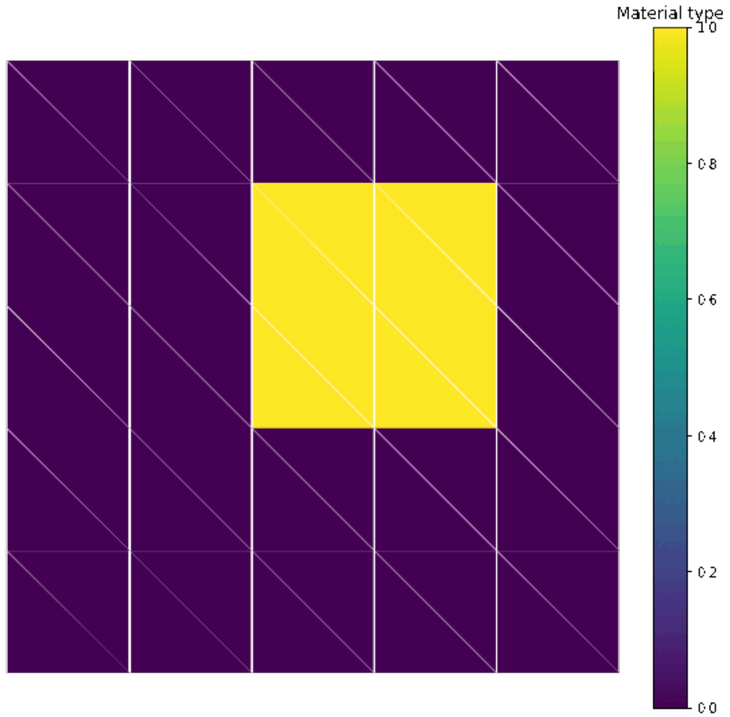
pp. 75-88

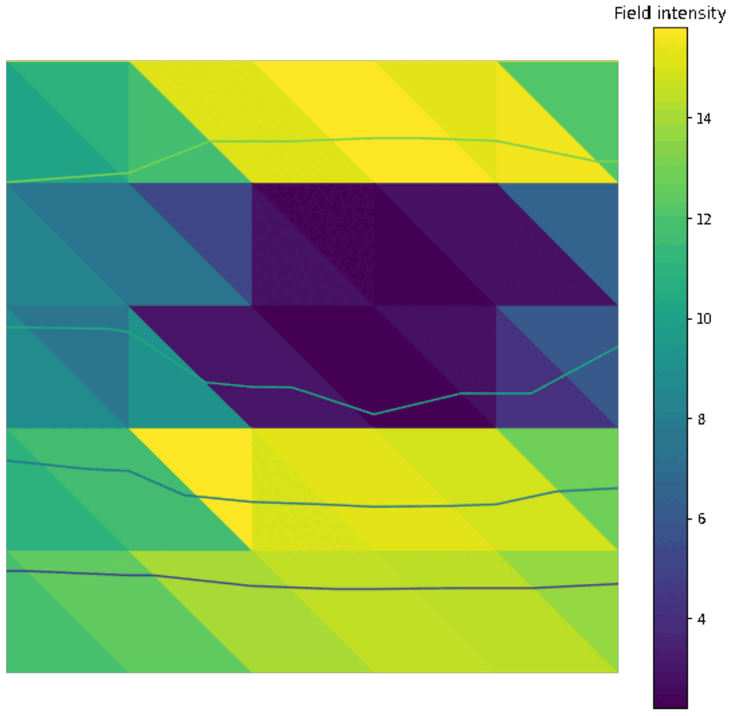
Code

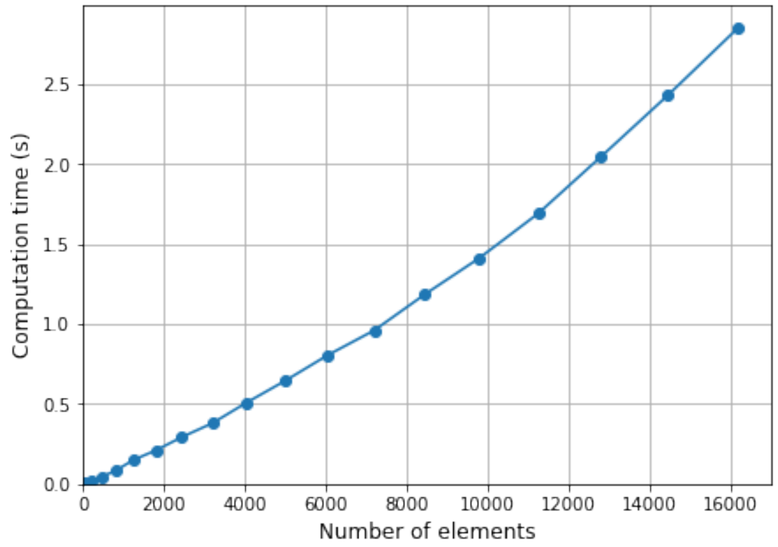
Post-processing

We can plot both the modulus of E in each element (constant, so only one color), and the scalar potential, which give these line evolutions, higher value on top, close to the boundary condition of V = 100, and lower value down, close to V = 0.

Increasing the number of elements permits to converge and visualize better, at the cost of computation time.







pp. 75-88

Magnetostatics

Vector potential

Magnetostatics

Vector potential

If we wish to calculate the field in a domain with current sources, the scalar potential formulation is not applicable. However, it is possible to use the vector potential A, related to B by:

$$\mathbf{B} = \nabla \times \mathbf{A}$$

For the rest of this magnetostatics part: we will assume a scalar permeability μ and thus reluctivity ν

$$\mathbf{B} = \mu \mathbf{H}$$
$$\mathbf{H} = \nu \mathbf{B}$$

In the same fashion than before, a matrix equation is created using $\nabla \times \mathbf{H} = \mathbf{J}$ with the energy functional to minimize:

$$\mathrm{F} = \iint_S \left[\int_0^{\mathrm{B}} \mathrm{H} \mathrm{dB} - \mathrm{JA} \right] \mathrm{ds}$$

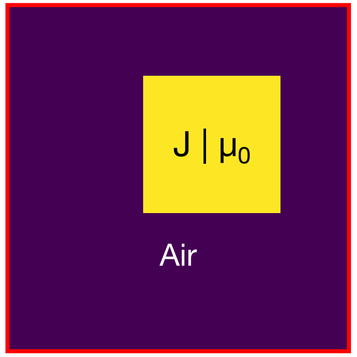
$$\frac{\partial \mathrm{F}}{\partial \mathrm{A}_n} = 0, n \in [1 \cdots N]$$

pp. 347-352

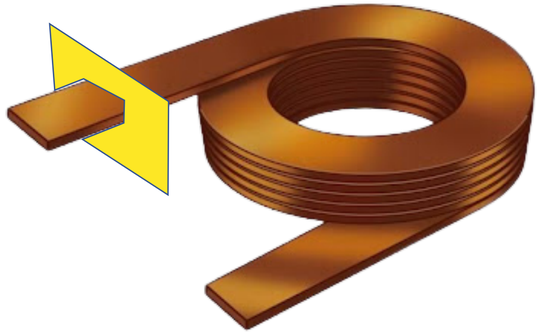
Magnetostatics

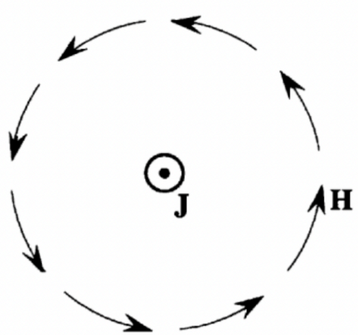
Magnet and current density sources

Current density, 100 A/mm²

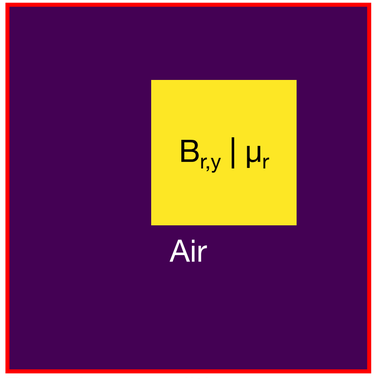


Boundary condition

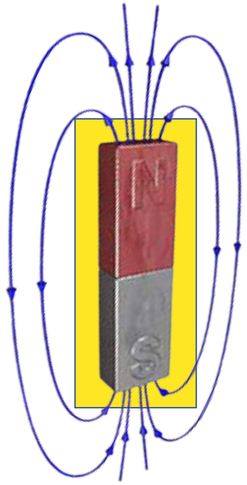




Remanent flux density, 1.3 T, +Y direction



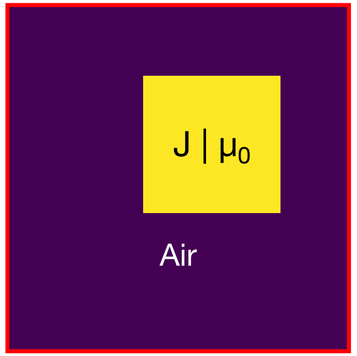
Boundary condition



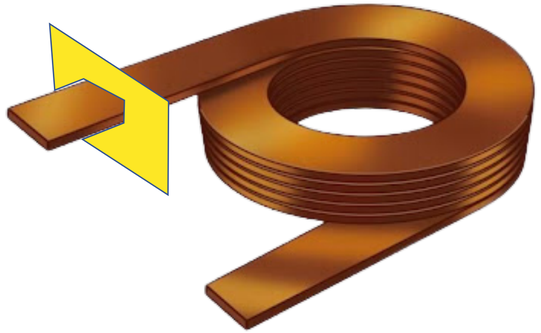
Magnetostatics

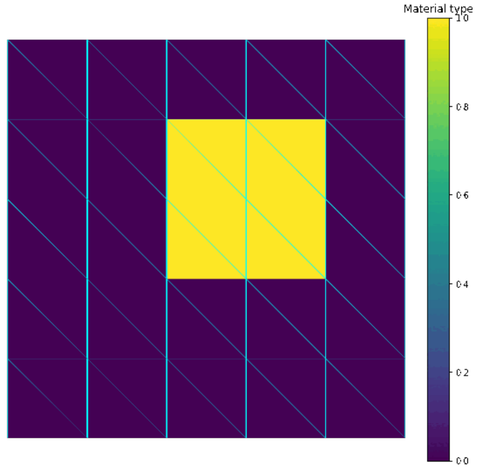
Magnet and current density sources

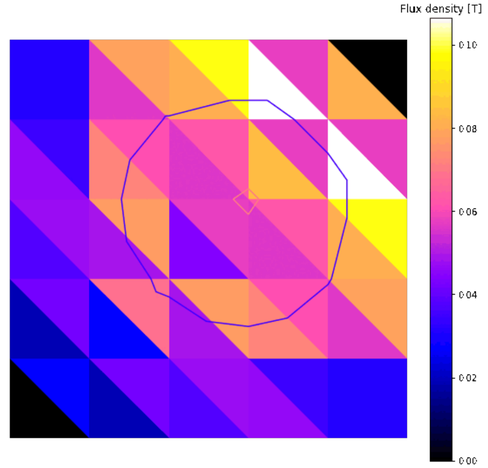
Current density, 100 A/mm²



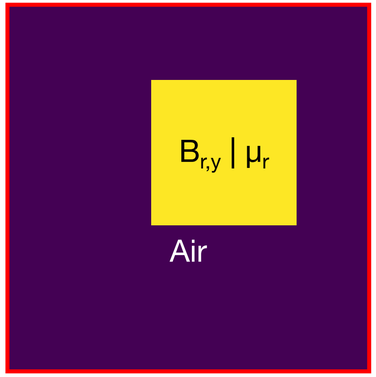
Boundary condition



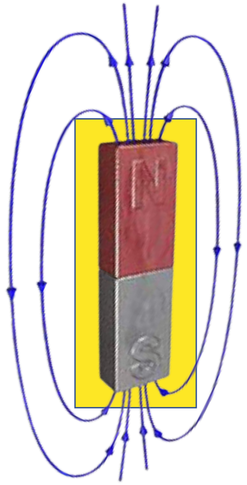


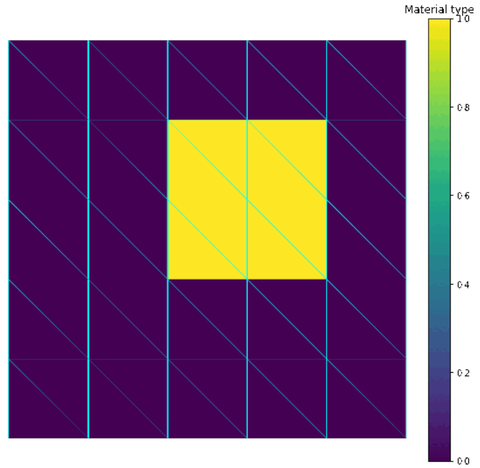


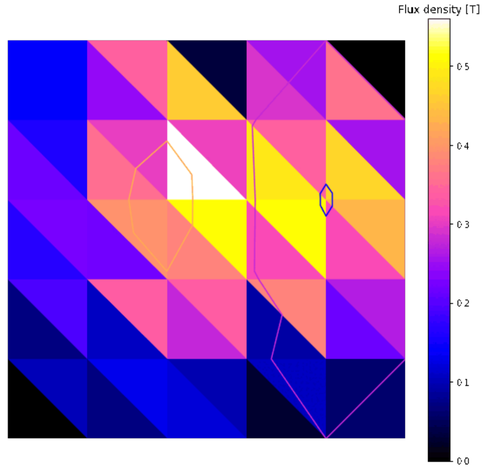
Remanent flux density, 1.3 T, +Y direction



Boundary condition



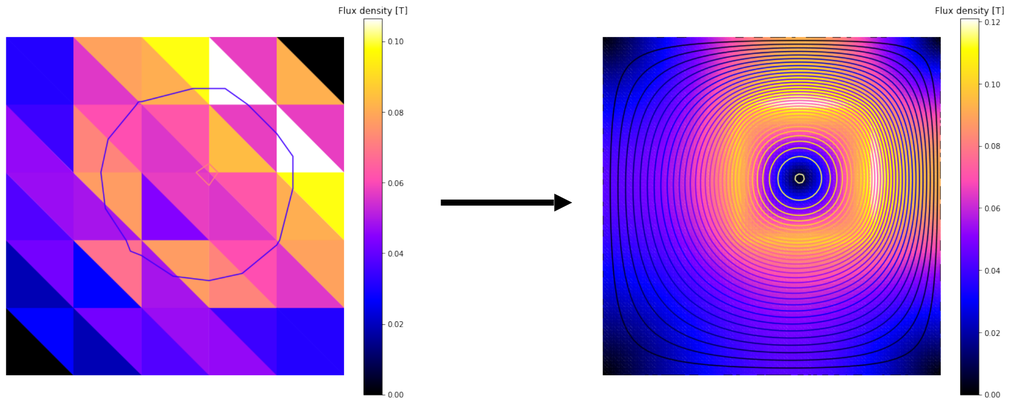


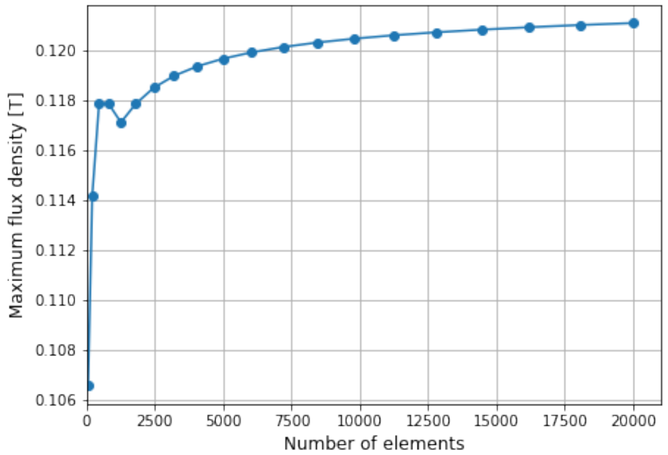


Magnetostatics

Magnet and current density sources

Current density, 100 A/mm²





Remanent flux density, 1.3 T, +Y direction