Notes curl-curl vector-valued equation

Michael Wathen

September 13, 2013

1 Model

Consider the vector-valued equation on the domain $\Omega \in \mathbb{R}^d$ where d is the dimension of the problem (here we use $d = 2, 3$):

$$\nabla_\wedge \nabla_\wedge \vec{u} + c(x,y)\vec{u} = \vec{f}, \quad \text{in} \quad \Omega \tag{1.1}$$

with $\vec{f} \in L^2(\Omega)^d$ as the right hand side. We will be considering Dirichlet boundary conditions:

$$\vec{u}_\wedge \vec{n} = g, \tag{1.2}$$

on the boundary $\partial\Omega$.

This type of system arises in the Maxwell system. In fact, this is the form of the Schur complement of 1st order form of Maxwell's equations.

2 Weak from

To be able to discretise (1.1) using the the finite element method (FEM) we first need to look at the weak form of the equations. We do this by multiplying through by a test function $\vec{v}$ and then integrating over the domain. This gives:

$$\int_\Omega \nabla_\wedge \nabla_\wedge \vec{u} \cdot \vec{v} + c(x,y)\vec{u} \cdot \vec{v}\, dx = \int_\Omega \vec{f} \cdot \vec{v}\, dx,$$

together with integration by parts and Green's Formulas we arrive at the following equation

$$\int_\Omega \nabla_\wedge \vec{u}\, \nabla_\wedge \vec{v} + c(x,y)\vec{u} \cdot \vec{v}\, dx = \int_\Omega \vec{f} \cdot \vec{v}\, dx. \tag{2.1}$$

We define the Stiffness matrix ($\mathcal{K}$) to be $\int_\Omega \nabla_\wedge \vec{u}\, \nabla_\wedge \vec{v}\, dx$, the Mass matrix ($\mathcal{M}$) to be $\int_\Omega c(x,y)\vec{u} \cdot \vec{v}\, dx$ and the right hand side vector (f) as $\int_\Omega \vec{f} \cdot \vec{v}\, dx$ to form the linear equation

$$(\mathcal{K} + \mathcal{M})u = f. \tag{2.2}$$

3 Nédélec elements

Nédélec elements of the first type [Nédélec, 1980] are $H(\text{curl};\Omega)$-conforming vector-valued finite elements that can be used to discretise the weak form of the curl-curl system given in (2.1).

3.1 Nédélec elements of the first kind

Before diving into the full finite element discretisation we need to derive the linear edge basis function. Here we will go through how one would derive both the basis functions on both tetrahedral and quadrilateral elements.

3.1.1 Tetrahedral elements

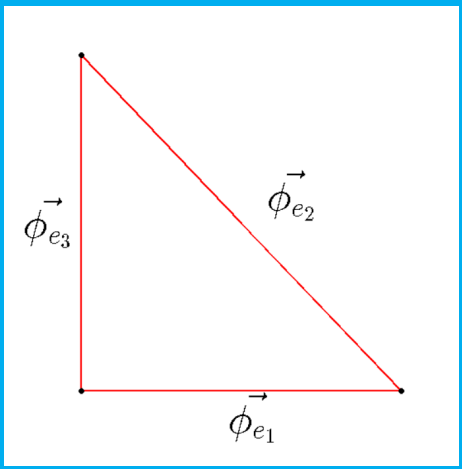


Figure 1: Tetrahedral reference cell

$$\vec{\phi}_{e_1} = \begin{pmatrix} 1-\hat{y} \\ \hat{x} \end{pmatrix}, \ \vec{\phi}_{e_2} = \begin{pmatrix} -\hat{y} \\ \hat{x} \end{pmatrix}, \ \vec{\phi}_{e_3} = \begin{pmatrix} -\hat{y} \\ \hat{x}-1 \end{pmatrix}, \tag{3.1}$$

3.1.2 Quadrilateral elements

$$\vec{\phi}_{e_1} = \begin{pmatrix} 1-\hat{y} \\ 0 \end{pmatrix}, \ \vec{\phi}_{e_2} = \begin{pmatrix} 0 \\ \hat{x} \end{pmatrix}, \ \vec{\phi}_{e_3} = \begin{pmatrix} -\hat{y} \\ 0 \end{pmatrix}, \ \vec{\phi}_{e_4} = \begin{pmatrix} 0 \\ \hat{x}-1 \end{pmatrix}. \tag{3.2}$$

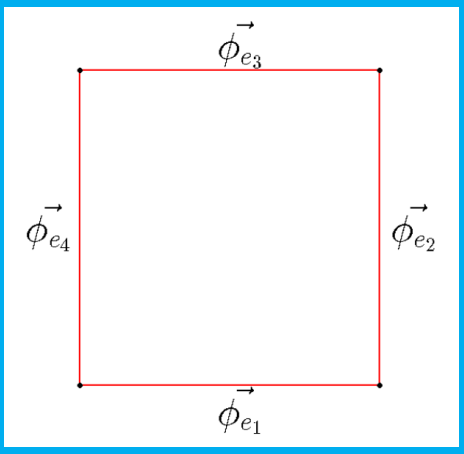


Figure 2: Quadrilateral reference cell

3.2 FEM grid construction

In this subsection we will go over a way to form a uniform quad mesh to be used in the FEM discretisation. Here I will denote $a(=0)$ and $b(=1)$ to be the boundary of the domain and n as the number of grid points. This means that for a uniform grid we will have $(n-1)^d$ cells (recall that d is the dimension of the domain). From here we will be looking at the 2-dimensional problem, hence $d = 2$.

The first things to decide is how to number the cells and edges of the grid. The way this ordering will effect the sparsity pattern of the matrix system given by (2.2) which could cause more fill when using direct methods but do not matter so much when using iterative methods. An example of an ordering one can use is given in figure 3.

Using the ordering from figure 3, one can then store what edge is defined in what cell as a matrix (see (3.3)). Here we have recorded the edge elements in an anticlockwise orientation for each of the cells (the numbers in bold

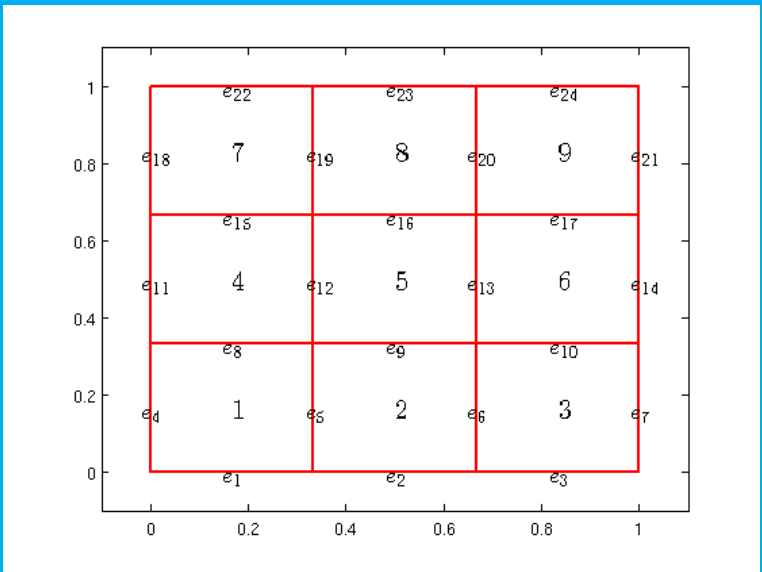


Figure 3: Example Grid

above each column of the matrix NNode).

$$\mathrm{NNode}^T = \begin{array}{c} \begin{array}{ccccccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} \end{array} \\ \begin{pmatrix} 1 & 2 & 3 & 8 & 9 & 10 & 15 & 16 & 17 \\ 5 & 6 & 7 & 12 & 13 & 14 & 19 & 20 & 21 \\ 8 & 9 & 10 & 15 & 16 & 17 & 22 & 23 & 24 \\ 4 & 5 & 6 & 11 & 12 & 13 & 18 & 19 & 20 \end{pmatrix} \end{array} \tag{3.3}$$

3.3 Matrix assemble

Take $c(x, y) = 1$ then the weak form of the problem is

$$\int_\Omega \nabla_\wedge \vec{u}\, \nabla_\wedge \vec{v} + \vec{u} \cdot \vec{v}\, dx = \int_\Omega \vec{f} \cdot \vec{v}\, dx.$$

Defining

$$u = \sum_{i=1}^{n} u_i \vec{\phi}_a \quad \text{and} \quad \vec{v} = \sum_{j=1}^{n} v_j \vec{\phi}_j \tag{3.4}$$

where $\vec{\phi}_k$ (where $k = a$ or b) are the basis functions defined in (3.2) on the reference cell $(\hat{x}, \hat{y})$. Substitute (3.4) into (2.1) gives

$$\sum_{i=1}^{n} u_i \int_{\Omega} \nabla_{\wedge} \vec{\phi}_a \, \nabla_{\wedge} \vec{\phi}_b + \vec{\phi}_a \cdot \vec{\phi}_b \, dx = \sum_{i=1}^{n} \int_{\Omega} \vec{f} \cdot \vec{\phi}_b \, dx, \tag{3.5}$$

for $j = 1, \ldots, n$.

At the moment (3.5) is defined on each quad of size h-by-h. By finding a linear transformation to convert to the reference element (see figure 4) we can define the local stiffness and mass matrices on each quad. Looking

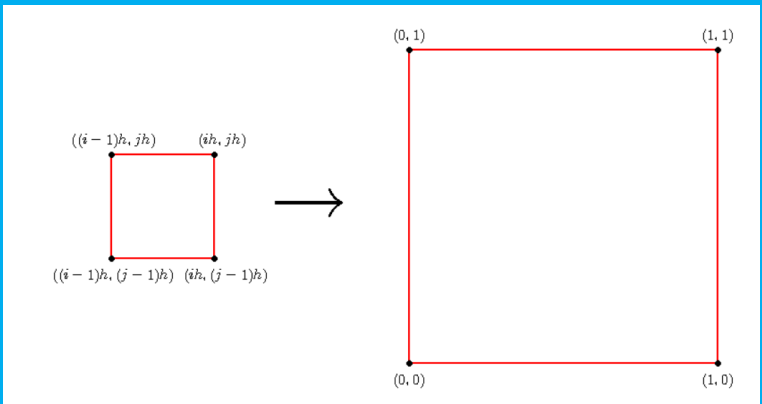


Figure 4: Figure to show conversion to reference cell

for the linear transformation that takes $x : [(i-1)h, ih] \to \hat{x} : [0, 1]$ and

$y : [(j-1)h, jh] \to \hat{y} : [0,1]$ we get that:

$$\begin{aligned} x &= h\hat{x} + (i-1)h, \\ y &= h\hat{y} + (i-1)h. \end{aligned}$$

Using the transformations we can now look at the locally defined stiffness ($\mathcal{K}_h$) and mass ($\mathcal{M}_h$) matrices:

$$\mathcal{K}_h = \int_{(i-1)h}^{ih} \int_{(j-1)h}^{jh} \nabla_\wedge \vec{\phi}_a \, \nabla_\wedge \vec{\phi}_b \, dx \, dy \text{ and } \mathcal{M}_h = \int_{(i-1)h}^{ih} \int_{(j-1)h}^{jh} \vec{\phi}_a \cdot \vec{\phi}_b \, dx \, dy$$

3.3.1 Local Stiffness matrix

Using the linear transformation with $\mathcal{K}_h$ we obtain

$$\mathcal{K}_h = \int_{(i-1)h}^{ih} \int_{(j-1)h}^{jh} \nabla_\wedge \vec{\phi}_a(\bar{x}, \bar{y}) \, \nabla_\wedge \vec{\phi}_b(\bar{x}, \bar{y}) \, dx \, dy$$

where $(\bar{x}, \bar{y}) = \left(\frac{x-(i-1)h}{h}, \frac{y-(i-1)h}{h}\right)$. Hence we get the local stiffness matrix to be:

$$\mathcal{K}_h = \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix} \tag{3.6}$$

3.3.2 Local Mass matrix

Now, using the linear transformations as we used to compute $\mathcal{K}_h$, then the local Mass matrix is

$$\mathcal{M}_h = \frac{h^2}{6}\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix} \tag{3.7}$$

3.3.3 Local Load vector

For the load vector (the right-hand-vector) the usual thing to do is to use some sort of quadrature to approxomate the following integral:

$$f_h = \int_{(i-1)h}^{ih}\int_{(j-1)h}^{jh} \vec{f}(x,y) \cdot \vec{\phi}_b\left(\frac{x-(i-1)h}{h}, \frac{y-(i-1)h}{h}\right) dx.$$

Generally one would use a first or second order approximation (midpoint rule or 2-point Gaussian quadrature) for this integral as we are using linear basis functions in the FEM scheme.

3.4 Assembling linear system

Once the local stiffness and mass matrices as well as the local load vector are compute then it is time to assemble the global system

$$\mathcal{A}u = f.$$

where $\mathcal{A} = \mathcal{K} + \mathcal{M}$. This is a fairly simple process in which we loop though all the elements and add in the contributions form the stiffness and mass

matrices into there correct place in the global matrix $\mathcal{A}$ (see the pseudo-code below).

```
for k =1:NumCells do
    el=NNode(k,:)
    for i =1:4 do
        for j =1:4 do
            A(el(i),el(j)) = A(el(i),el(j)) +K_h(i,j) + M_h(i,j)
        end for
        f(el(i)) = f(el(i)) + f_h(i)
    end for
end for
```

3.5 Boundary Conditions

So far we have not said anything about boundary conditions. In finite elements we have "essential boundary conditions" (conditions that we need to manually impose) and "natural boundary conditions" (conditions which are directly imposed by the variational form).

4 MATLAB results

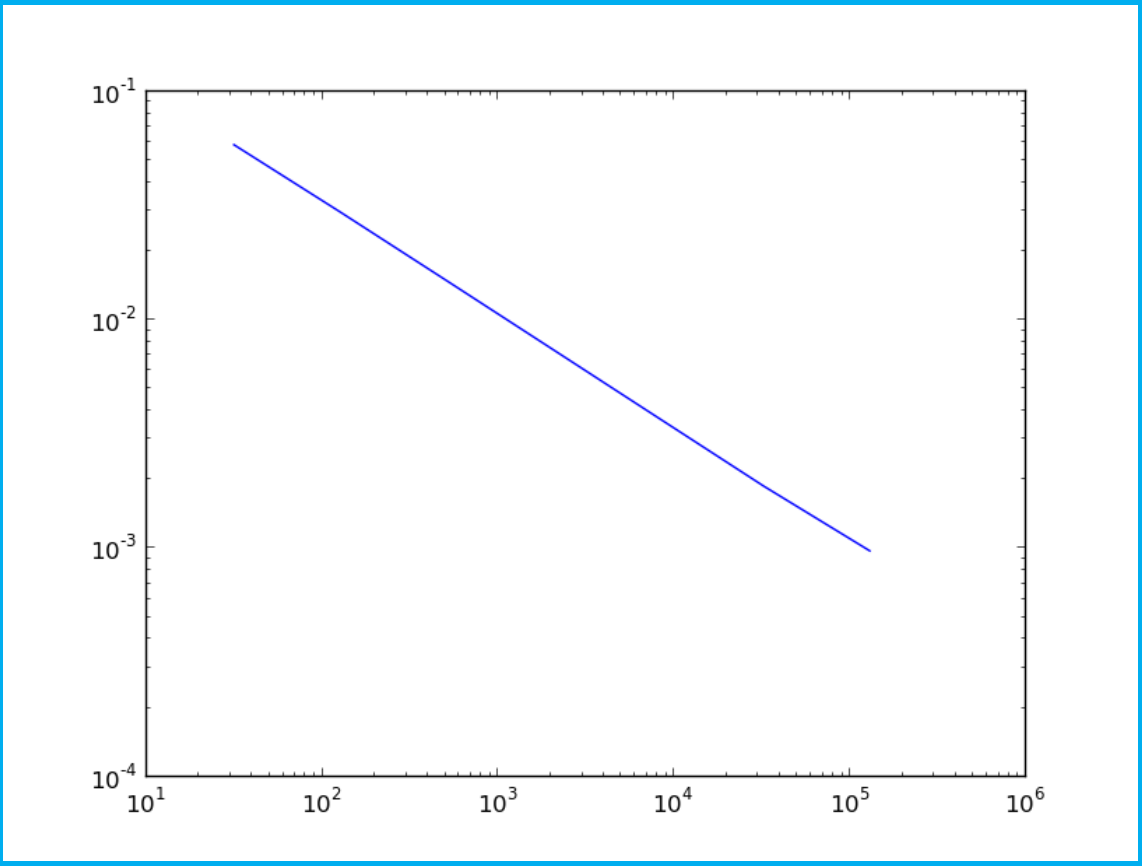
5 FEniCS results

5.1 Convergence results

cycle	# cells	# dofs	L_2-error
0	4	56	0.0575954
1	8	208	0.0292943
2	16	800	0.0147103
3	32	3136	0.00736305
4	64	12416	0.00368252
5	128	49408	0.00184138
6	256	197120	0.000920707
7	512	787456	0.000460355

5.2 Matrix timing results

cycle	# cells	# dofs	Time
0	4	56	0.022646
1	8	208	0.00144291
2	16	800	0.00212812
3	32	3136	0.00621486
4	64	12416	0.0200882
5	128	49408	0.079313
6	256	197120	0.366703
7	512	787456	1.55944
8	1024	$3.14778e{+}06$	6.79219
9	2048	$1.2587e{+}07$	27.8257

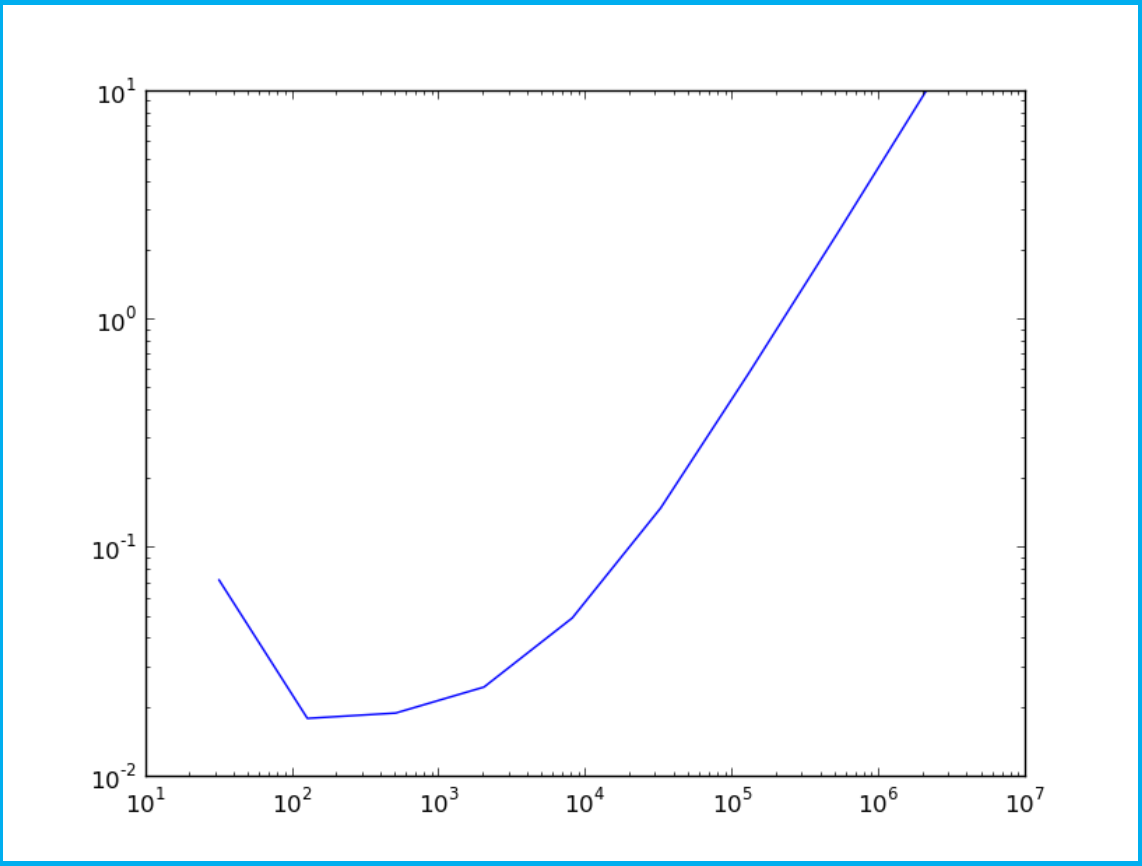


5.3 None zero boundary conditions

6 deal.II results

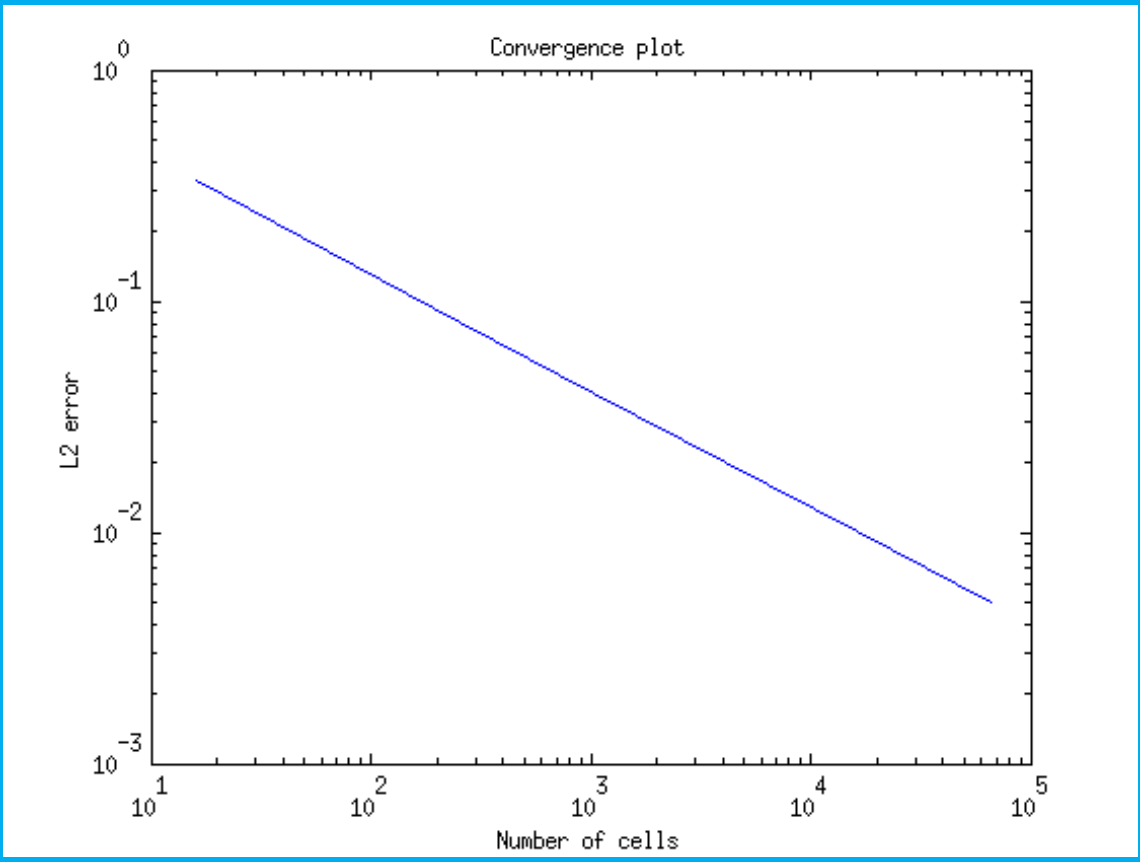
6.1 Convergence results

cycle	# cells	# dofs	L_2-error
0	16	40	3.3545e-01
1	64	144	1.6312e-01
2	256	544	8.0546e-02
3	1024	2112	4.0129e-02
4	4096	8320	2.0046e-02
5	16384	33024	1.0021e-02
6	65536	131584	5.0101e-03



6.2 Matrix timing results

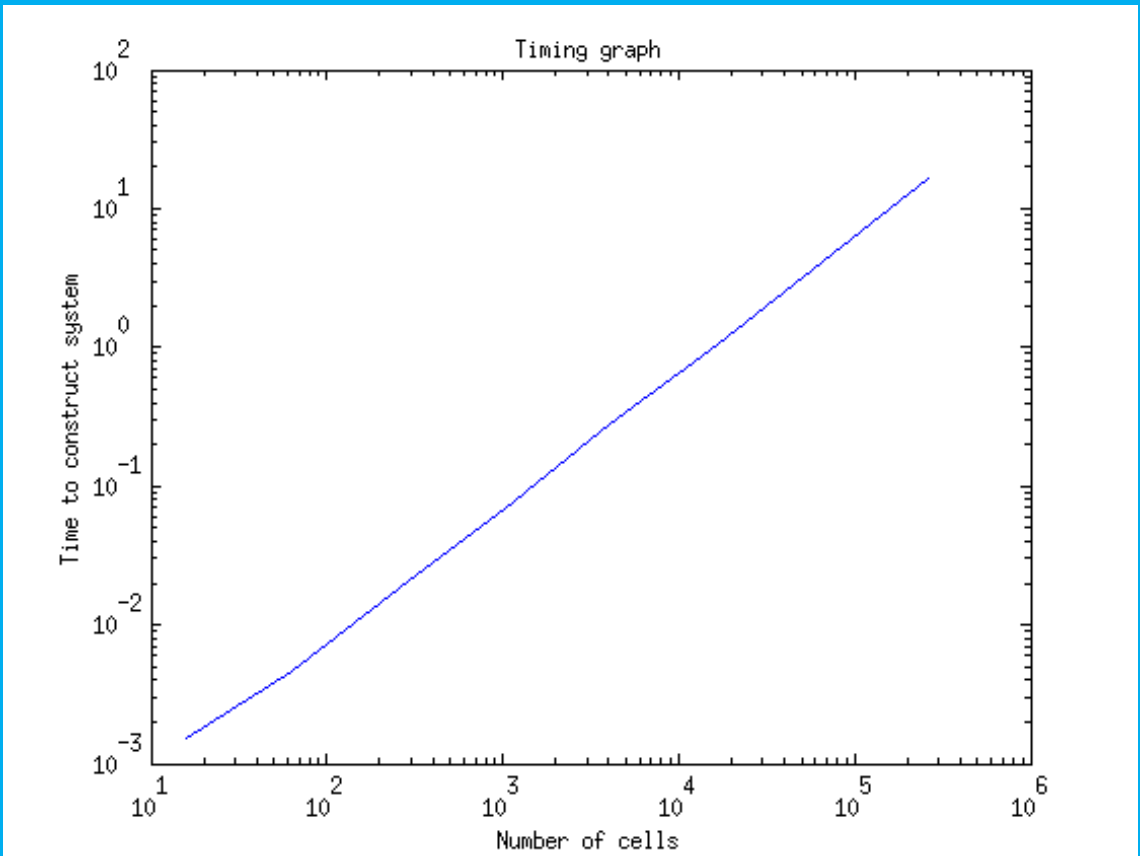
cycle	# cells	# dofs	Time
0	16	40	0.0016
1	64	144	0.0045
2	256	544	0.0173
3	1024	2112	0.0662
4	4096	8320	0.2582
5	16384	33024	1.0823
6	65536	131584	4.1593
7	262144	525312	16.8618
8	1048576	2099200	67.1105



6.3 None zero boundary conditions

References

[Nédélec, 1980] Nédélec, J.-C. (1980). Mixed finite elements in 3. *Numerische Mathematik*, 35(3):315–341.


Timing graph
Time to construct system
Number of cells
10^2
10^1
10^0
10^-1
10^-2
10^-3
10^1
10^2
10^3
10^4
10^5
10^6