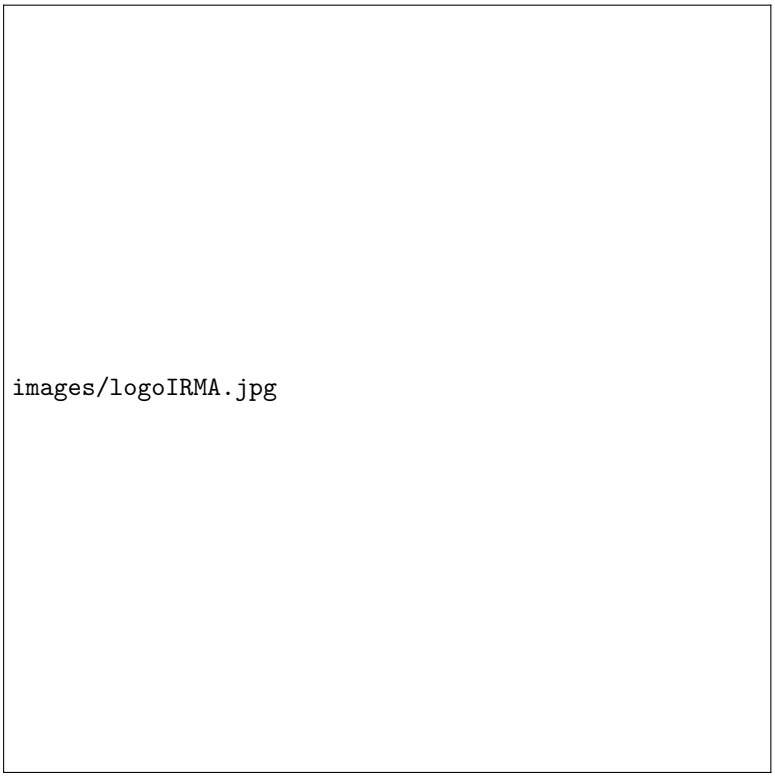
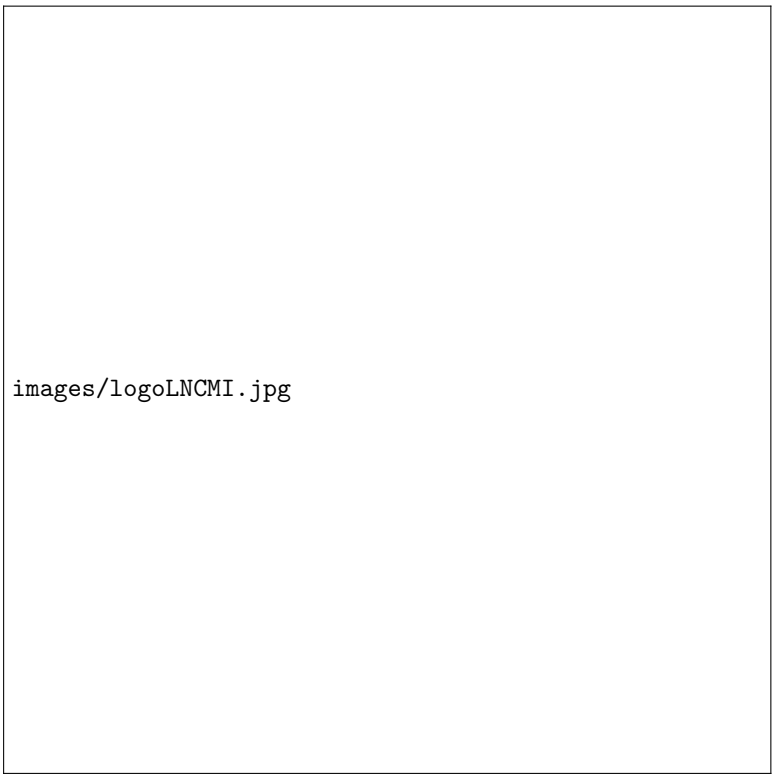
Institut de Recherche Mathématique Avancée (IRMA)
Laboratoire National des Champs Magnétiques Intenses (LNCMI)



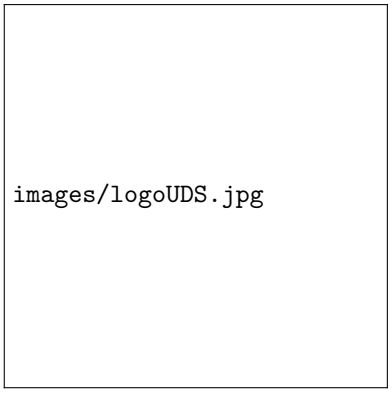


June 2, 2015
Master 2 CSSI :: Internship Report

Coupled modelling and simulation of electromagnets in stationnary and instationnary modes (axisymmetric and 3D)

J. Nadarasa

Supervisors : Ch. Prud'homme, Ch. Trophime, C. Daversin



1 Acknowledgements

First I would like to thank my supervisors Ch. Prud'homme, Ch. Trophime and C. Daversin to have indeed supervised my work through this internship, to have guided me, for their patience and for all the time they awarded to me. A particular thanks to Ch. Prud'homme who offered me this internship subject and for having given me the opportunity to attend the PRACE seminar in March. Then, I would like to thank all the master students, the PhD students and the research engineer V. Huber for their help, their encouragements and support and also for all the time we spent together.

Thanks to all I could achieve this work, it was really a great period of time and experience !

Contents

Part I
Introduction

2 Subject

This is the report of my internship that I did from february to august 2013 at the end of my Master degree, in the IRMA (Institut de Recherche Mathématique Avancée). The subject was entitled "*Coupled modelling and simulation of electromagnets in stationnary and instationnary modes (axisymmetric and 3D)*". It was in collaboration with the Grenoble's LNCMI (Laboratoire National des Champs Magnétiques Intenses).

The scientists of LNCMI work with high level of magnetic fields and make different experiences with them in order to improve the knowledge in fundamental physics :

- scanning and getting properties of specific materials
- localize atoms in the body (IRM)
- supraconductor material
- levitation
- ...

The LNCMI has two branches : one in Toulouse, focusing on pulsed magnetic field; another in Grenoble, on static magnetic field.

Electromagnets To generate the magnetic field they use what is called electromagnets :

Those electromagnets are created from copper material, cut by electro-erosion to the shapes of figures 1 and 2.

Figure 1: longitudinally cooled turn

Figure 2: radially cooled turn

Magnetic field generation Then, a current of about 30 kilo-Ampère is injected into the electromagnet. It induces a massive movement of electrons and thus important Laplace forces. Those forces result directly from the magnetic field.

There is also a phenomenon of Joule heating due to the current intensity. To cool down the temperature inside the electromagnet, scientists put a water-cooling system with 2 circuits : one closed-circuit which is in contact with the electromagnet (and so the water is demagnetized); and another open-circuit water which comes to cool down the first circuit.

The longitudinally cooled turn (1) allows water to travel only in cylinder center hole and the outside, whereas the radially cooled turn (2) allows water to go even in the space between two turns. Thus the latter offers a better cooling properties.

Last but not least, the high level of temperature creates a deformation effect (because the material of the electromagnet is copper) which should be also be taken into account.

Multiphysics modelling From those physics phenomenons, that is to say :

- Electromagnetism
- Heat transfer
- Hydraulics
- Elasticity

we are going to deal with only Electromagnetism, Heat transfer and Elasticity in our electromagnet modelling.

We make the hypothesis of axisymmetrical geometry, which implies that temperature, magnetic field, deformation depend only on the radius r and the height z. Then we make a second hypothesis about the cooling system : it is modelled by forced convection law (see further in the thermic section).

Mathematics We will use the finite element method for the modelling, and use the library Feel++ for the simulation : it is an open-source project for solving PDEs with Galerkin methods; there is a little community arround, composed by researchers, PhD students and students who make it progress. For more details : www.feelpp.org.

3 Technical information : geometry, materials

The considered geometry is a hollow cylinder, r_1 as the inner radius and r_2 as the outer radius and H its height. This cylinder is made in copper. There is also a water cooling system : this water is at T_1 temperature in the central hollow and T_2 in the outside. Here are the main parameters used :

- h_1 and h_2 : heat transfer coefficient in r_1 and r_2
- σ : electrical conductivity
- k : thermal conductivity
- ρ : density of the material
- C_p : specific heat of the material
- λ : Lamé's coefficient (1)
- μ : Lamé's coefficient (2)
- α : thermal dilatation coefficient

Most of them are depending on the temperature or the magnetic potential..., nervertheless at the beginning we consider them as constants; then if we have time, we will use methods (Newton's method) to approach them.

Cylindrical coordinates system As the modelling is done in axisymmetric the main variable is r. And as the geometry is a cylinder, we should better use cylindrical coordinates system instead of cartesian ones. It is very important to keep that in mind because variationnal formulation integrals should be slightly modified : the determinant of the Jacobian of the base switching matrix will intervene. And as we are in cylindrical system, the geometry we have is only a rectangle (because the base switching matrix will transform the rectangle into a cylinder); it is also important.

4 Work

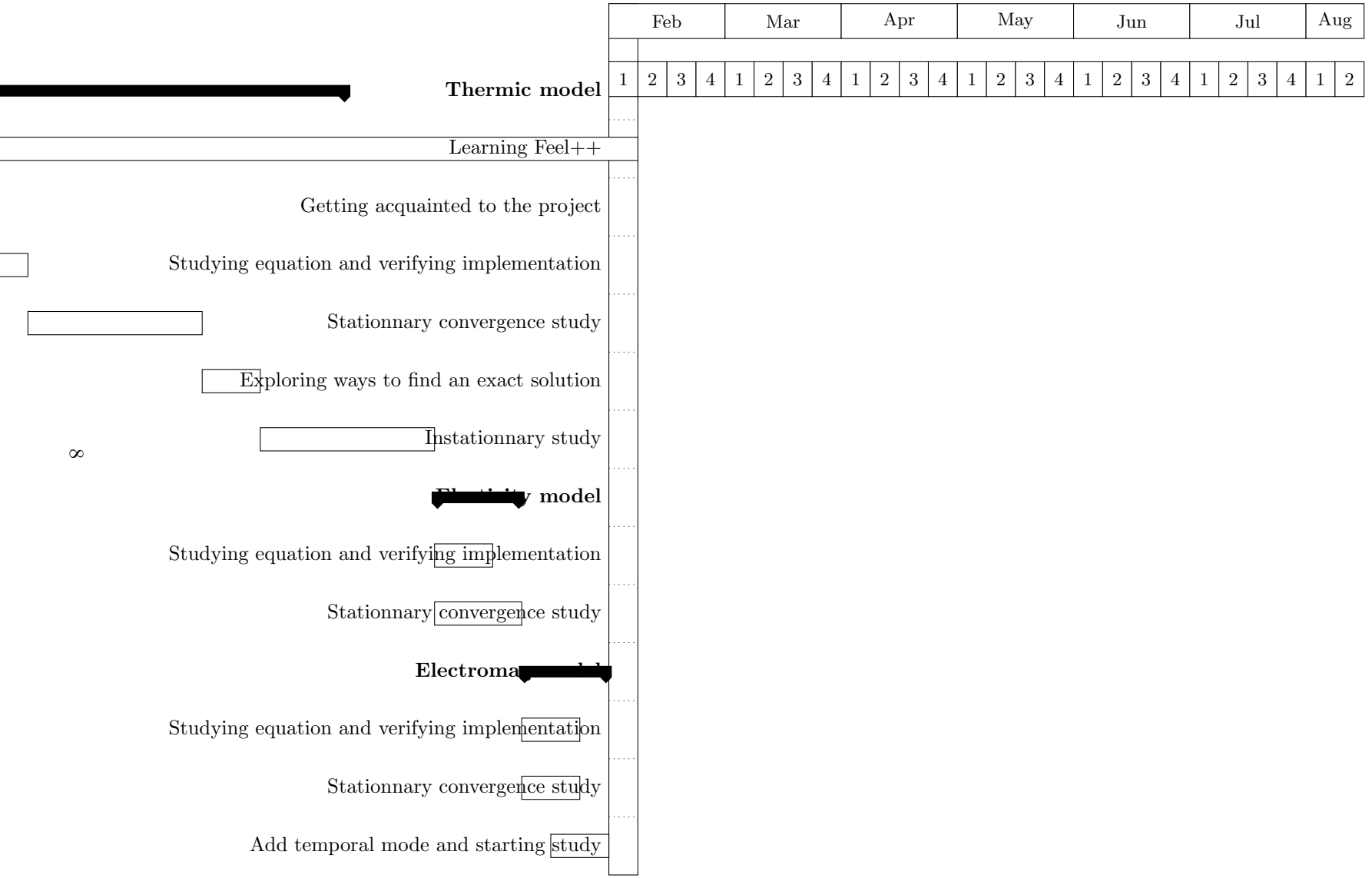
I present here the work in the same order as what I did during the internship : first the thermic model, then the elasticity model and then the electromagnetism one.

For each model, there is the theorical approach with the equations, the variationnal formulation, then a word about the implementation and lastly the visual results and convergence graphs. In the appendix, one can find additionnal information and results. They are unfortunately written in French and could not be translated due to the lack of time.

I didn't have to start my work from nothing, indeed I could rely on the basis of what has been done in a previous internship by C. Daversin, A. Daudin and N. Bernabeu. About the working environment, I had access to 2 HPCs of 24 and 16 threads each.

The Feel++ code here is given to show that a work of implementation has been done and to show the transparency between the variationnal formulation and the implementation.

Remark In the following section you can find a very simple Gantt diagram : the starting of the project had definitely taken a long time as well as the time awarded to the thermic model.


Feb
Mar
Apr
May
Jun
Jul
Aug
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2
Thermic model
Learning Feel++
Getting acquainted to the project
Studying equation and verifying implementation
Stationnary convergence study
Exploring ways to find an exact solution
Instationnary study
∞
model
Studying equation and verifying implementation
Stationnary convergence study
Studying equation and verifying implementation
Stationnary convergence study
Add temporal mode and starting study

Part II
Thermic model

The thermic model is the behaviour of the heat in the electromagnet: it is also called conduction. The study of the temperature will be usefull to prevent materials to be subjected to high level heat; then as it is influenced by magnetism and influences elasticity, we have to know the temperature spreading inside the electromagnet in order to simulate it.

5 Equations

The equation we use has been introduced by Fourier in 1811 and its general form is :

$$\frac{\partial C_p(T)\rho(T)T}{\partial t} - \text{div}\ (k(T)\,\mathbf{grad}\,T) = \sigma\left(\frac{U}{2\pi r}\right)^2, \tag{1}$$

with :

ρ	density of the material,
C_p	heat capacity of the material,
k	thermal conductivity of the material,
σ	electrical conductivity of the material,
U	potential difference applied to the electromagnet.

It can be established by combining thermodynamics laws, applied to a test volume with Fourier law.

Boundary conditions We set the boundary conditions, defined in r_1 and r_2 (see section 3), according to the forced convection model :

$$k\frac{\partial T}{\partial \mathbf{n}} = h(r)\left(T(r) - T_{ambiant}(r)\right), \tag{2}$$

where $\mathbf{n}$ is the inner-pointing normal from the electromagnet. h_1 and h_2 are the heat transfer coefficient in respectively inner and outer side of the cylinder; and T_1 and T_2 reprensent $T_{ambiant}(r_1)$ et $T_{ambiant}(r_2)$.

6 Variationnal formulation

We consider a rectangular domain called Ω and the associated function space $H^1(\Omega)$. This function space could be also $H_0^1(\Omega)$ or $H_g^1(\Omega) = \{u \in H^1(\Omega),\ u = g \text{ on } \partial\Omega\}$; it depends on the boundary conditions. First comes the stationnary case and then the instationnary one. All the parameters (k, ρ and C_p) are constants.

6.1 Stationnary formulation

From (1), the equation we have is :

$$-\text{div}\ (k\,\mathbf{grad}\,T) = -k\Delta T = Q_{th} \tag{3}$$

where $Q_{th} = \sigma\left(\frac{U}{2\pi r}\right)^2$.

By multiplying by a test function $v \in H^1(\Omega)$ (or another function space correctly chosen) and by integrating over the domain Ω, we have :

$$-k\int_\Omega \Delta T v = \int_\Omega \sigma\left(\frac{U}{2\pi r}\right)^2 v \tag{4}$$

according to Green's theorem we have :

$$-k\int_{\Omega}\Delta T v = k\int_{\Omega}\nabla T\cdot\nabla v - k\int_{\partial\Omega}\frac{\partial T}{\partial n}v. \tag{5}$$

According to the boundary condition defined at the beginning, we have :

$$-k\int_{\partial\Omega}\frac{\partial T}{\partial n}v = -\int_{R_1+R_2}(h_1+h_2)Tv + \int_{R_1}h_1T_1v + \int_{R_2}h_2T_2v \tag{6}$$

By replacing and after rearrangement, we have :

Find $T\in H^1(\Omega)$ such that, $\forall v\in H^1(\Omega)$,

$$k\int_{\Omega}\nabla T\cdot\nabla v - \int_{R_1+R_2}(h_1+h_2)Tv = \int_{\Omega}\sigma\left(\frac{U}{2\pi r}\right)^2 v - \int_{R_1}h_1T_1v - \int_{R_2}h_2T_2v \tag{7}$$

6.2 Instationnary formulation

The overall equation is :

$$\rho C_p\frac{\partial T}{\partial t} - k\Delta T = Q_{th} \tag{8}$$

We have to discretize the time derivative in a correct way in order to avoid caring about stabitility conditions. Thus, it is better to use backward eulerian methods; we will begin with the backward Euler's method of order 1. It is indeed very simple to use and to implement.

The time derivative is discretized like :

$$\rho C_p\frac{\partial T}{\partial t} \simeq \rho C_p\frac{T^{n+1}-T^n}{\Delta t} \tag{9}$$

We obtain :

$$\rho C_p\frac{T^{n+1}-T^n}{\Delta t} = F(t,T^{n+1})$$

where

$$F(t,T^{n+1}) = k\Delta T^{n+1} + Q_{th}$$

After rearrangement, we obtain the equation with the discretized time derivative :

$$\frac{\rho C_p}{\Delta t}T^{n+1} - k\Delta T^{n+1} = Q_{th} + \frac{\rho C_p}{\Delta t}T^n \tag{10}$$

By multiplying by a test function $v\in H^1(\Omega)$ and integrating over the domain we obtain equation to solve.

Find $T\in H^1(\Omega)$ such that $\forall v\in H^1(\Omega)$:

$$\frac{\rho C_p}{\Delta t}\int_{\Omega}T^{n+1}v + k\int_{\Omega}\nabla T^{n+1}\cdot\nabla v - \int_{R_1+R_2}(h_1+h_2)T^{n+1}v = \int_{\Omega}Q_{th}v + \frac{\rho C_p}{\Delta t}\int_{\Omega}T^n v - \int_{R_1}h_1T_1v - \int_{R_2}h_2T_2v \tag{11}$$

7 Stationnary mode

7.1 Code

A brief piece of code; it is a simplified version with only the forced convection boundary condition:

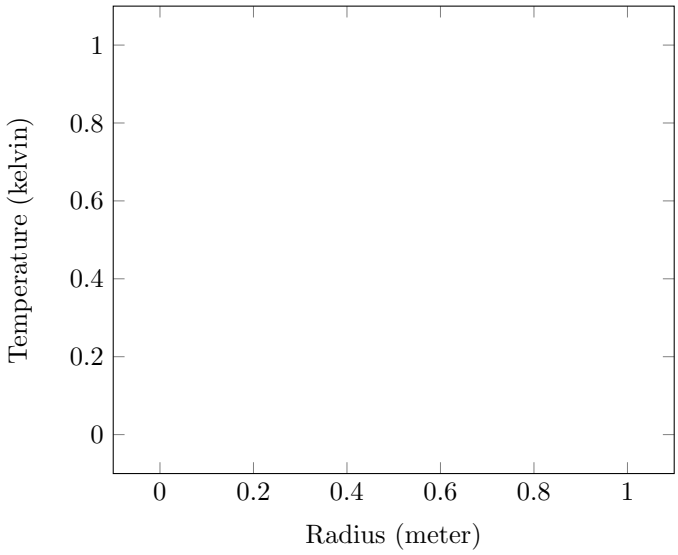
```
1
2  // linear form
3  auto gradU = - *(_mat->ddp()) / (2 * M_PI * Px());
4
5  form1( _test=Th, _vector=FT) += integrate( markedelements( mesh, physical ),
6  Px()*sigma*id(phi_T)*gradU*gradU, _Q<Quadrature_Order>() );
7
8  form1( _test=Th, _vector=FT) += integrate( elements(mesh), Px()*id(phi_T)*idv(Rhs)
       , _Q<Quadrature_Order>() );
9
10 // Robin boundary condition
11 form1( _test=Th, _vector=FT) += integrate( markedfaces( mesh, physical ),
12 -Px()*id(phi_T)*_c->constant()->getExpr() );
13
14 // bilinear form
15 form2( Th, Th, MatrixT ) += integrate( markedelements( mesh, physical),
16 Px()*k*gradt(T)*trans(grad(phi_T)));
17
18 // Robin boundary condition
19 form2( Th, Th, MatrixT ) += integrate( markedfaces(mesh, physical),
20 Px()*_c->coeff_var()->getExpr()*idt(T)*id(phi_T) );
21
22 // solving the system
23 backend_type::build()->solve( _matrix=MatrixT, _solution=T, _rhs=FT );
```

7.2 First outputs

The pre-existent program, called HeatAxi, was already in a working state. Here is a view of what we have in Paraview :

Due to the cooling system in the inner and outer border, it is logical to get lower temperature in the borders and higher in the middle.

We can also plot T depending on r :



7.3 Verification & validation

It *seems* that the program is working well, but we have to prove it by those two well-known steps :

verification : does the implemented model behaves correctly when mesh sizes is changing ?

validation : does the implemented model represent well the real spreading of the temperature ?

Linearity and non-linearity Since the electrical σ and thermal k conducitivity are dependent on the temperature, the model is non linear. But for simplification purpose, we first suppose them being constants : it is the linear model.

Linear : σ and k are considered as constants

Non-linear : σ and k are defined as :

$$\frac{k(T)}{\sigma(T)T} = L = constant \quad \text{Wiedemann-Frantz law} \tag{12}$$

and for σ :

$$\sigma(T) = \frac{\sigma_0}{1 + \alpha(T - T_{ref})} \quad 273K \leq T \leq 373K \tag{13}$$

The non-linearity of σ is already implemented; if we have enough time, we will try to implement the expression of k and make the V&V process again.

7.4 Convergence study

The convergence theory links the L^2 and H^1 norms to the mesh size. That is to say the L^2 error evolves like h^{n+1} and the H^1 error evolves like h^n, where n is the order of the polynomial interpolation; in logarithmic scale, L^2 error should have a slope equal to $n+1$, whereas H^1 error should have a slope equal to n.

So studying the convergence of those norms will be good enough to prove the verification (the error is behaving correctly, thus the results are mathematically right). It is also enough to prove the validation because the error is between the approximated temperature and the exact one; this exact function have been found under specific condition, please refer to the appendix for more details.

Some modifications to the program have been made to make the convergence study automatic. First we used optoins like **convergence_data**, **convergence_max** and get results in .dat file which can be plotted in GnuPlot, then Ch. Trophime created a specific class (the Error class) which is doing that in black box.

7.5 Exact solution

To compare with the the approached solution, the exact one must be found. Nevertheless, only the linear will be handled : in fact it is very difficult (maybe impossible) to find the exact solution in the non-linear case because of integrations which cannot be made.

In stationnary mode there is no time derivative term, so the equation becomes :

$$-\operatorname{div}\left(k\,\mathbf{grad}\,T\right) = \sigma\left(\frac{U}{2\pi r}\right)^2, \tag{14}$$

and as we are in cylindrical coordinates system, the temperature is only function of the radius r. The equation becomes :

$$-\frac{1}{r}\frac{\partial}{\partial r}\left(rk\frac{\partial T}{\partial r}\right) = \sigma\left(\frac{U}{2\pi r}\right)^2. \tag{15}$$

After a first integration :

$$rk\frac{\partial T}{\partial r} = -\sigma\left(\frac{U}{2\pi}\right)^2 \ln r + Ak,$$

where A is a constant. So we get :

$$T = A\ln r - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln^2 r + B. \tag{16}$$

By applying the boundary conditions defined below, A and B can be found. As said before, please refer to the appendix for more details.

Without the exact solution

What if the exact solution cannot be found ? What if users have no time to invest in finding the exact solution ?

To allow to verify that the HeatAxi is giving correct results without knowing anything about the exact value of the temperature, error estimators have been implemented by other students: residual based estimator and ZZ's (Zhu and Zienkiewicz) estimator.

They give the sup of the error which can be generated during the computation of the numerical solution; the residual estimator estimates the error $\|u - u_h\|$, whereas the ZZ estimator is for the error $\|\nabla u - \nabla u_h\|$.

Another way to let the user verify that the program is working correctly is to specify manually an exact function. The program will compute the associated right hand side member and use this rhs in finite element method.

7.6 Graphs

All what is needed for the verification-validation process is there, let's continue with graphs got from the automatic convergence study in different norms (L^2 and H^1), types of resolution (linear, newton), polynomial orders (P^1 to P^5).

Parameters value The parameters values are :

- Thermal conductivity k : 377
- DDP U : 0.5
- Electrical conductivity σ : 48e+06
- Heat transfer coefficient h_1 : 60000
- Heat transfer coefficient h_2 : 30000

- Temperature T_1 and T_2 : 303
- L : 2.41e-08

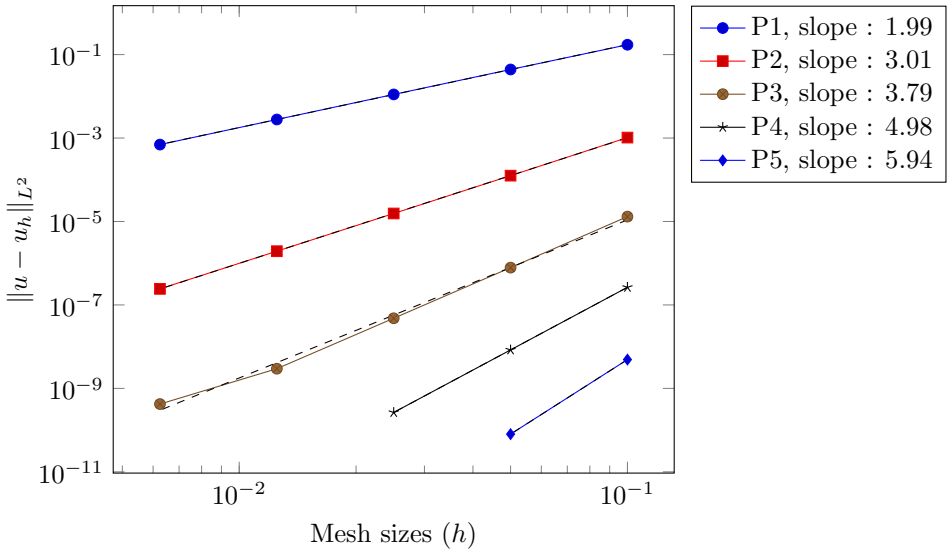
We use a specific geometry for the V&V process : a rectangle 1x0.1 (in fact $r_1 = 1$ and $r_2 = 2$). In fact this geometry and the value of h_1 and h_2 that we have chosen satisfy the condition :

$$h_1 r_1 = h_2 r_2,$$

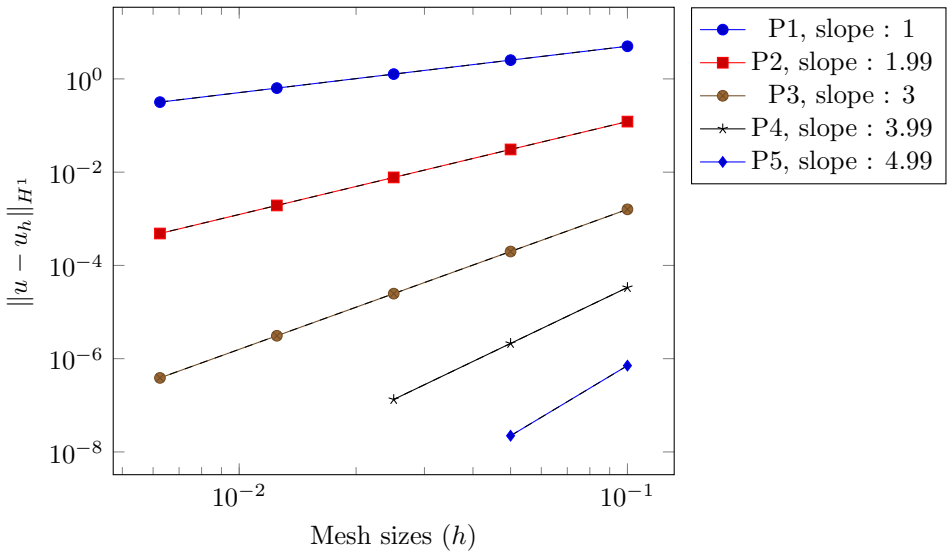
a condition needed to use the exact solution found just before (please refer to the first appendix for more details about this condition).

Warning Dashed plots are for theorical slopes and filled plots are from HeatAxi.

Error convergence - Linear resolution - L2 exact error

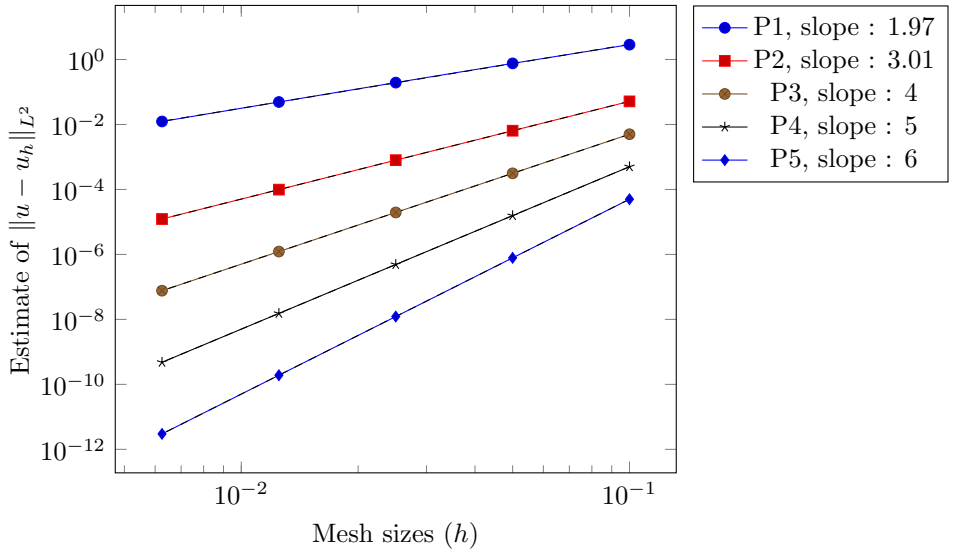


Error convergence - Linear resolution - H1 exact error

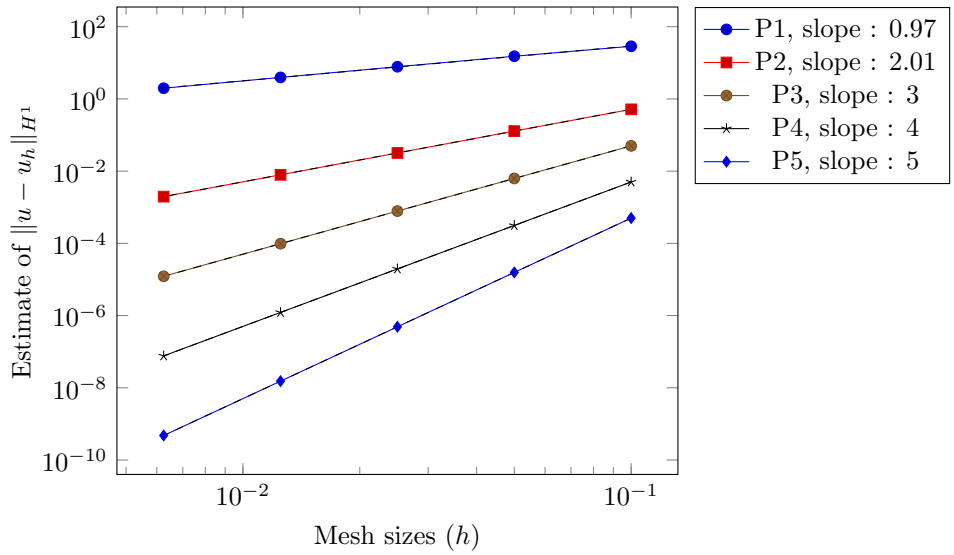


For the non linear case, there is no exact function to compare. We can only rely on the evolution of the residual based error estimators. As there is, for the moment, no theorical explaining for the ZZ estimator for other polynomial order than P1, it cannot be used here.

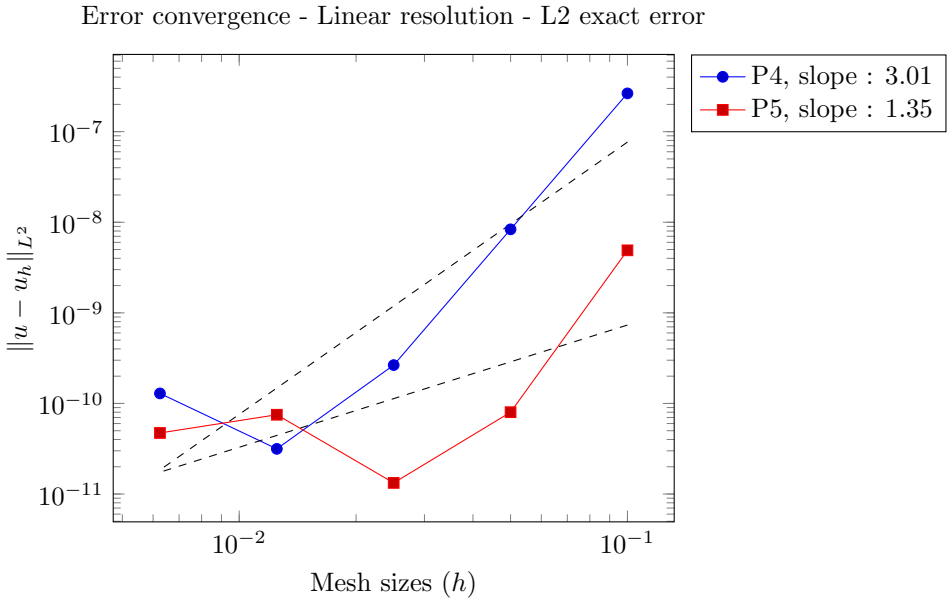
Error convergence - Non Linear resolution - Residu L2 error estimator

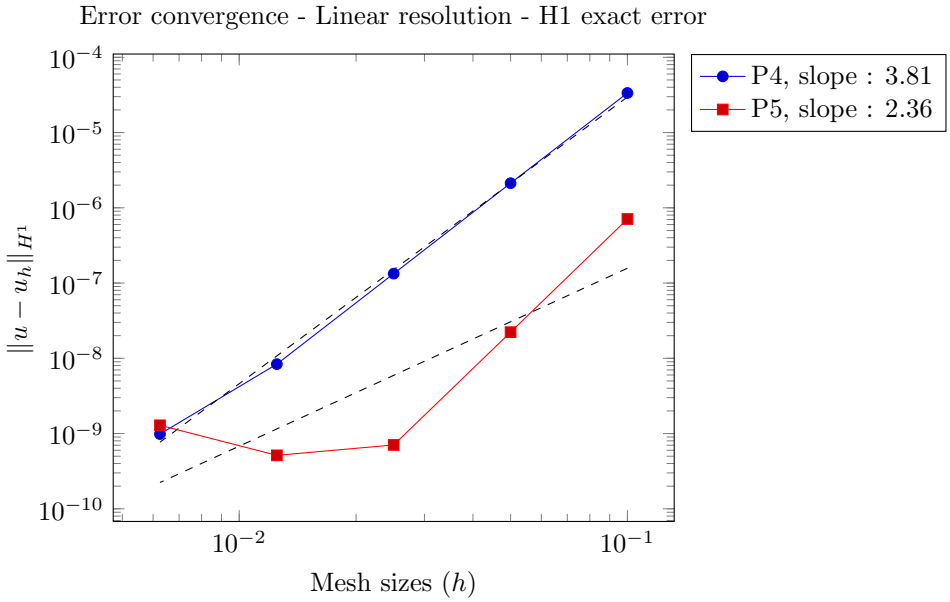


Error convergence - Non Linear resolution - Residu H1 error estimator



Conclusion The slopes for the exact error and also the slopes for the estimated error are optimal and are the best we can get (according to the error convergence theory). For polynomial order 4 and 5, only some points are shown here because of a precision problem. The complete graphs are :



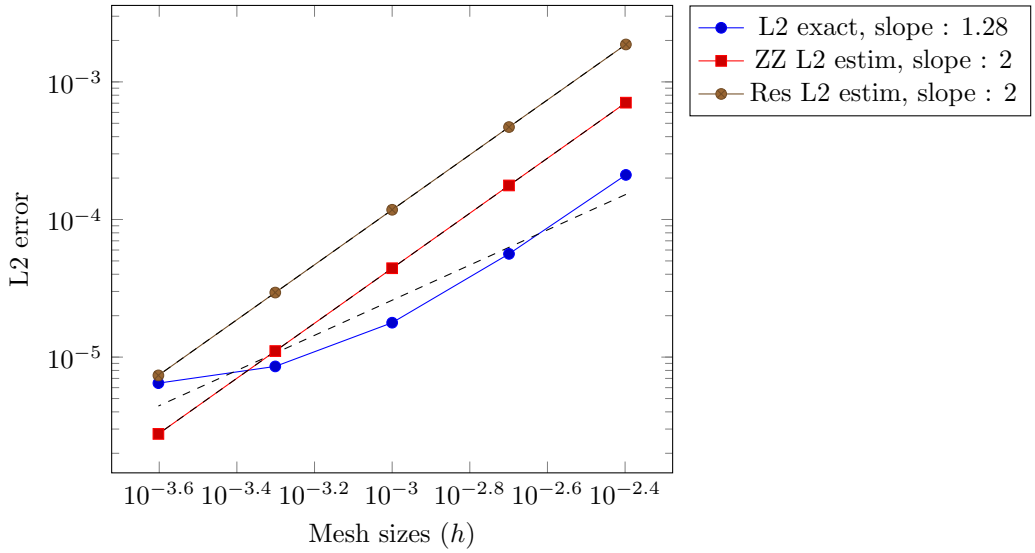


We will meet the problem again and again in other model, so it needs to be clarified here. It is because the precision used here when manipulation numbers is not enough. Indeed, the machine epsilon is arround 10^{-16}; but by combining thousands of thousands of numbers with this precision, the effect is visible much more earlier : for example, here we cannot go below 10^{-11}.

This phenomenon is already appearing in the L^2 error in polynomial order 4, but in P5 it is even more visible.

Another precision problem, this time due to the approximation of π. We put 3.141592 for π in the exact solution in the .cfg file, and we got :

PI precision - Error convergence - Order 1, norm L2, resolution Linear



By giving the "exact" expression of π (GiNaC [1] allows to use it), instead of using an approximation, the problem could be solved.

8 Instationnary Mode

The next step is to put the temporal dimension in the simulation.

8.1 Feel++ implementation

Inside Feel++ there is a class called BDF (*Backward Differentiation Formula*) which helps users to implement some specific finite difference method for time discretization. As most of the operations of discretization are done in blackbox, it is very user-friendly.

We began by implementing "manually" the Backward Euler's method. Then we create another version with BDF. We could compare the two versions of HeatAxi and verify that they gave the same results.

We decided to keep and use BDF : time discretization order, initial/final time, time step ... everything can be changed easily compare to our "manual" version.

Some extra features had to added to take into account the temporal parameter because by default, GiNaC looks only after space variables x, y, z.

```
// initialising the unknown
M_bdf->initialize(T);

// loop which will iterate from initial time to final time
for(M_bdf->start(); M_bdf->isFinished()==false; M_bdf->next() )
{
  // Temperature computing
      // stationnary terms
      form1( ... ) // linear form
      form2( ... ) // bilinear form
```

[1] From *Ginac Is Not A CAS*, is a free computer algebra system and is widely used in Feel++'s API. It provides a wide spectre of mathematical functions and constants. For more details www.ginac.de.

```
11
12      // instationnary terms
13
14         // creating the T(n+1) sequence
15       auto bdf_poly = M_bdf->polyDeriv();
16
17         form1(Th,FT) += integrate(_range=elements(mesh),
18      _expr=cst(rcp)*Px()*idv(bdf_poly)*id(phi_T));
19
20       form2(Th,Th,MatrixT) += integrate(_range=elements(mesh),
21      _expr=cst(M_bdf->polyDerivCoefficient(0))*cst(rcp)*Px()*idt(T)*id(phi_T));
22
23
24      // solving the system
25      solve(...)
26
27      // storing T(n+1) into T(n)
28      // then T(n) into T(n-1)...
29   M_bdf->shiftRight( T );
30 }
```

8.2 Verification & Validation

As for the stationnary mode, those two processes are done by computing the error between the exact solution and the approached one. The solution in instationnary case could not be found. But thanks to the option which allows to give an exact solution and to compute the associated rhs, the v&v process can be done like this :

1 An exact solution is given in the input : $sin(\pi(x-1))sin(\pi(y-1))e^{-t}$

2 The rhs function is computed by HeatAxi

3 The approached T function is computed by HeatAxi

4 Temporal error between the exact and the computed solution

Formula The expression of the temporal error is get from V. Chabannes PhD thesis [**?**] (p. 115) :

$$E_{rr} = \left(\Delta t \sum_{t=t_i}^{t_f} \|u - u^n\|^2_{L^2(\Omega^{tn})} \right)^{\frac{1}{2}} \tag{17}$$

As the BDF method is a finite difference method, in order 1 the error should have a slope equal to 1; in order 2, a slope equal to 2 etc... Practically speaking though it is not so simple.

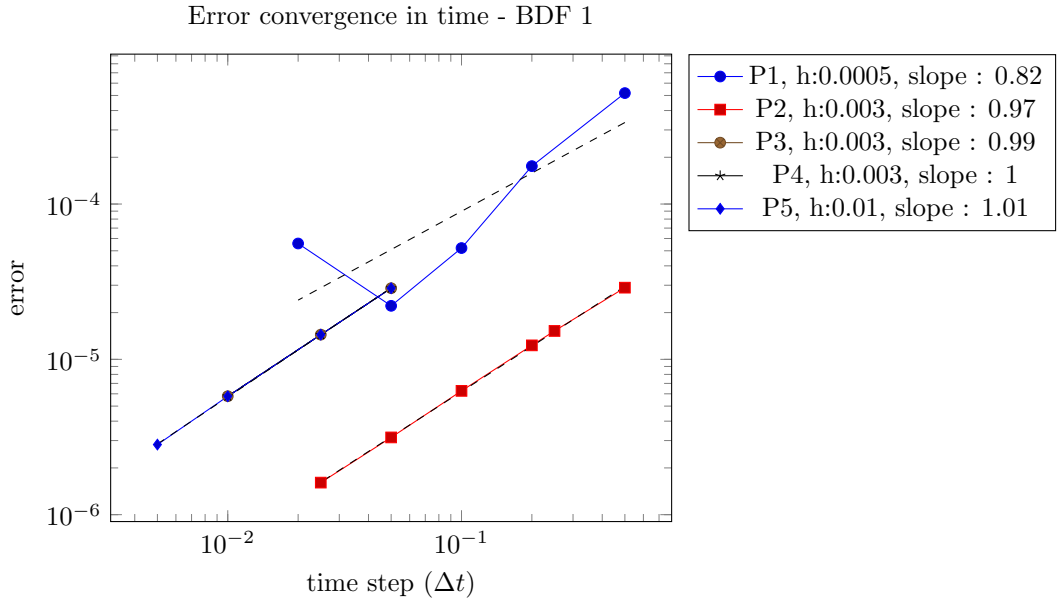
8.3 Graphs

The exact solution here is :

$$20 \sin(\pi(x-1)) \sin(\pi(y-0.05)/0.1)e^{-t}$$

The geometry used here is the same as one used for the stationnary convergence study (see section 7.6) The initial is set to 0. The final time to 0.5 second for P1 and P2; and 0.05 for P3 to P5.

We give only the graph for BDF1 in P1 to P5. For more details please refer to the appendix *BDF : convergence study*. The h notation represents here the mesh size used for the study.



For most of the polynomial orders, the chosen mesh size 0.003 : a higher mesh size implies higher proportion of spatial error and so the error due to the finite difference method won't be visible. Still the P1-BDF1 test slope, done with a mesh size of 0.0005, is not fully satisfying. This means the spatial error is still too high compared to the time derivative discretization error. A contrario, for P5, a mesh size of 0.01 seems to be quite enough.

8.4 Conclusion

The temporal error convergence study was not as easy as the spatial one in stationnay mode. We faced several problems : at the beginning the temporal error went on increasing when the time-step decreases, then we couldn't get a good convergence slope...

After having reviewed the equation, the variationnal formulation, the implementation, the BDF method, we came to the conclusion that if an optimal slope couldn't be reached it is because it is simply not possible to reach always an optimal slope (otherwise it wouldn't be an optimal slope). And that if the temporal error decreases when the time-step decreases, it is enough to prove the convergence of the BDF method and its implementation.

Part III
Elasticity model

9 Introduction

During the magnetic field generation, electromagnets are subjected to internal forces, and effects from the temperature. They will induce physical changement which should be taken in account in our modelling.

9.1 Equations

Here the unknown u is a vector, it represents the displacement induced by the deformation. The parameters λ and μ are called Lamé's coefficients and are defined as follows :

$$\lambda = \frac{Ev}{(1-2v)(1+v)}, \quad \mu = \frac{E}{2(1+v)} \tag{18}$$

- E is called Young modulus, represents the rigidity of the material and varies 10^5 and 10^{10}
- v is called Poisson's coefficient, represents the incompressibility of the material and varies from 0 to 0.5 (a material of 0.5 is perfectly incompressible)

First, the tensor of small deformations :

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \quad 1 \leq i,j \leq 3. \tag{19}$$

Then the stress tensor :

$$\sigma_{ij} = \lambda \delta_{ij} \mathrm{div}\, u + 2\mu\epsilon_{ij}, \quad 1 \leq i,j \leq, \tag{20}$$

where,

$$\mathrm{div}\, u = \sum_{k=1}^{3} \frac{\partial u_k}{\partial x_k} = \sum_{k=1}^{3} \epsilon_{kk} = \mathrm{Tr}(\epsilon) \tag{21}$$

With $u = (u_1, u_2, u_3)^T$, the equilibrium equations is written like :

$$\sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} + f_i = 0, \quad 1 \leq i \leq 3, \tag{22}$$

with the following boundary conditions :

$$\begin{aligned} u_i &= u_{0i} \quad \text{on } \Gamma_1, \\ \sum_{j=1}^{3} \sigma_{ij} n_j &= g_i \quad \text{on } \Gamma_2, \end{aligned} \tag{23}$$

By changing the notation of $\dfrac{\partial u_i}{\partial x_j}$ to $u_{i,j}$, we can rearrange items and have the equilibrium equation written like this :

$$(\lambda+\mu)\sum_{k=1}^{3} u_{k,ki} + \mu \sum_{k=1}^{3} u_{i,kk} + \sum_{k=1}^{3} f_i = 0, \quad 1 \leq i \leq 3 \tag{24}$$

which become with symbolic notation :

$$(\lambda+\mu)\mathbf{grad}\,\mathrm{div}\, u + \mu\Delta u + F = 0 \tag{25}$$

Remark To this equation, we should add the term which represents the dilatation effect :

$$-(3\lambda + 2\mu)\alpha\nabla T,$$

where α is the dilatation parameter; but for the moment we set it to 0.

Warning This is the linear elasticity equation in cartesian coordinates system ! We will switch to cylindrical coordinates system after writing the variationnal formulation. We referred to the article [**?**].

9.2 Variationnal Formulation

For the variationnal formulation, it is better not to use the symbolic notation; we will here deal only with the strong Dirichlet condition (for the boundary Γ_1) that is why the function space is set to $H^1_{u0}(\Omega) = \{u \in H^1(\Omega), u = u_0 \text{ on } \Gamma_1\}$. So by multiplying by $v \in H^1_{u0}(\Omega)$ and integrating over the domain Ω, we have :

$$-\int_\Omega \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} v_i = \int_\Omega f_i v_i, \quad 1 \leq i \leq 3 \tag{26}$$

By applying Green's theorem, we change the left hand side into :

$$\int_\Omega \sum_{j=1}^{3} \sigma_{ij} \frac{\partial v_i}{\partial x_j} - \int_{\partial\Omega} \sum_{j=1}^{3} \sigma_{ij} n_j v_i = \int_\Omega f_i v_i, \quad 1 \leq i \leq 3 \tag{27}$$

then, according to boundary condition on Γ_2 :

$$\int_\Omega \sum_{j=1}^{3} \sigma_{ij} \frac{\partial v_i}{\partial x_j} - \int_{\Gamma_2} g_i v_i = \int_\Omega f_i v_i, \quad 1 \leq i \leq 3 \tag{28}$$

Now we have to change the cartesian coordinates into cylindrical ones; the details can be found in the article [**?**]. We have carefully verified all the calculation made in the article.

Find $u = (u_r, u_z),$ where $u_r,\ u_z \in H^1(\Omega)$, such that $\forall v = (v_r, v_z),$ where $v_r,\ v_z \in H^1(\Omega)$:

$$\begin{aligned}
&\int_\Omega r\lambda \left(\frac{\partial u_r}{\partial r} + \frac{1}{r}u_r + \frac{\partial u_z}{\partial z}\right)\left(\frac{\partial v_r}{\partial r} + \frac{1}{r}v_r\right) + \int_\Omega r\mu\Big[2\left(\frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r} + \frac{1}{r^2}u_r v_r\right) + \left(\frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z} + \frac{\partial u_z}{\partial r}\frac{\partial v_r}{\partial z}\right)\Big] \\
&\qquad - \int_{\Gamma_2} g_r v_r r = \int_\Omega f_r v_r r \\
&\int_\Omega r\mu\left(\frac{\partial u_r}{\partial z}\frac{\partial v_z}{\partial r} + \frac{\partial u_z}{\partial r}\frac{\partial v_z}{\partial r} + 2\frac{\partial u_z}{\partial z}\frac{\partial v_z}{\partial z}\right) + r\lambda\left(\frac{\partial u_r}{\partial r} + \frac{1}{r}u_r + \frac{\partial u_z}{\partial z}\right)\frac{\partial v_z}{\partial z} - \int_{\Gamma_2} g_z v_z r = \int_\Omega f_z v_z r
\end{aligned} \tag{29}$$

9.3 Exact solution

With usual methods, it is not possible to find an exact solution in 2D. Nevertheless, in C. Daversin's note [**?**] we can find an exact solution for an infinite tube, in 1D, without any thermal dilatation :

$$u(r) = C_1 r + \frac{C_2}{r} + u_p(r), \tag{30}$$

where C_1 and C_2 are found with the boundary conditions, and $u_p(r)$ is a particular solution. For more details please refer to her note.

As we are in 2D, the only way to use this exact function to validate our model is to take an enough tall geometry and to plot the displacement in the center of the domain. This will be done if we have enough time.

10 Code

We will base the work arround elasticity on an example available in Feel++ : the feelpp_linelaxi application (doc/manual/solid). We have to verify the implementation and behaviour of the application by doing tests; we may have to add some extra features also.

Remark As for the elasticity, the implementation work was not focused on the mathematical aspect but more on adding HeatAxi features to it, we think it is not useful to give pieces of this code here. Moreover, as there are lots of terms in the linear and bilinear forms, it would take an important place.

So we began with verifying the compilation, the execution and the output given by the linelaxi application. We also checked the implementation of the variationnal formulation in cylindrical coordinates. Everything *seems* to be OK; now we have to prove it mathematically.

11 Convergence study

Again, as for HeatAxi, to prove mathematically that the program is working correctly, we have to show that the error (L^2 and H^1) between an exact function, given by the user, and the approched one, computed with the finite element method is decreasing with a specific slope, when the mesh size is decreasing.

The L^2 and H^1 errors computation is the similar to HeatAxi. Only the equation to compute the rhs is changing here.

Rhs computing Remember, the equation here is :

$$(\lambda+\mu)\mathbf{grad}\,\mathrm{div}\,u+\mu\Delta u+F=0 \tag{31}$$

We should transform those cartesian derivatives into cylindrical ones. The operators in axisymmetric cylindrical system are :

$$\begin{aligned}\mathbf{grad}\,u &= \frac{\partial u}{\partial r}\vec{e_r}+\frac{\partial u}{\partial z}\vec{e_z}\\ \mathrm{div}\,u &= \frac{u_r}{r}+\frac{\partial u_r}{\partial r}+\frac{\partial u_z}{\partial z}\\ \Delta u &= \left(\frac{\partial^2 u_r}{\partial r^2}+\frac{\partial^2 u_r}{\partial z^2}+\frac{1}{r}\frac{\partial u_r}{\partial r}-\frac{u_r}{r^2}\right)\vec{e_r}\\ &\quad \left(\frac{\partial^2 u_z}{\partial r^2}+\frac{\partial^2 u_z}{\partial z^2}+\frac{1}{r}\frac{\partial u_z}{\partial r}\right)\vec{e_z}\end{aligned} \tag{32}$$

where $\vec{e_r}$ and $\vec{e_z}$ are the basis vectors of the cylindrical system.
Thus, we have :

$$\begin{aligned}\mathbf{grad}\ \mathrm{div}\,u &= \left(\frac{1}{r}\frac{\partial u_r}{\partial r}-\frac{u_r}{r^2}+\frac{\partial^2 u_r}{\partial r^2}+\frac{\partial^2 u_z}{\partial r\partial z}\right)\vec{e_r}\\ &\quad \left(\frac{1}{r}\frac{\partial u_r}{\partial z}+\frac{\partial^2 u_r}{\partial r\partial z}+\frac{\partial^2 u_z}{\partial^2 z}\right)\vec{e_z}\end{aligned} \tag{33}$$

Here are the equation for the rhs in each components :

$$-F_r=(\lambda+\mu)\left(\frac{1}{r}\frac{\partial u_r}{\partial r}-\frac{u_r}{r^2}+\frac{\partial^2 u_r}{\partial r^2}+\frac{\partial^2 u_z}{\partial r\partial z}\right)+\mu\left(\frac{\partial^2 u_r}{\partial r^2}+\frac{\partial^2 u_r}{\partial z^2}+\frac{1}{r}\frac{\partial u_r}{\partial r}-\frac{u_r}{r^2}\right) \tag{34}$$

$$-F_z=(\lambda+\mu)\left(\frac{1}{r}\frac{\partial u_r}{\partial z}+\frac{\partial^2 u_r}{\partial r\partial z}+\frac{\partial^2 u_z}{\partial^2 z}\right)+\mu\left(\frac{\partial^2 u_z}{\partial r^2}+\frac{\partial^2 u_z}{\partial z^2}+\frac{1}{r}\frac{\partial u_z}{\partial r}\right) \tag{35}$$

Adding options Several options for specifying the exact function, the rhs function, to compute the rhs or not, the geofile... have been added. We couldn't use the class Error used in HeatAxi, because the unknown u here is a vector not a scalar.

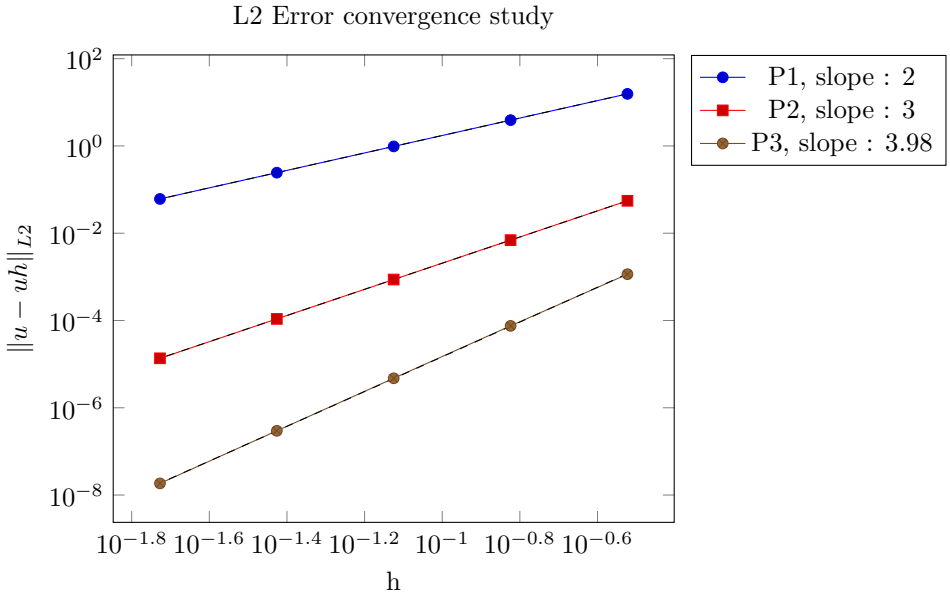
The structure of the overall code is similar to the HeatAxi one.

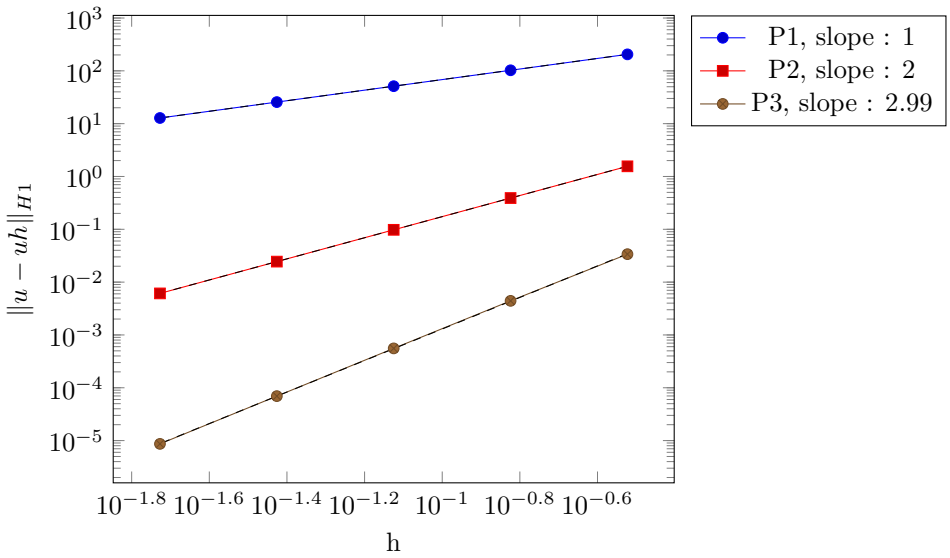
11.1 Results - Verification

We take a rectangular geometry (1 unit large, and 2 units long) and very simple exact solution :

$$\begin{aligned} exact_r &= \sin(\pi x)\sin(\pi z) \\ exact_z &= 10\sin(\pi x)\sin(\pi z) \end{aligned} \tag{36}$$

The rhs on each component is computed by the equation shown below. The boundary conditions, which are Dirichlet's ones, are put to 0 as the exact solution is null on it. Here we go with the graphs :





Conclusion The slopes are good. We didn't do tests in P4 and P5 because here, for the stationnary problem P3 is enough. Using higher polynomial order could be certainly useful in instationnary mode.

11.2 Results - Validation

By specifying an exact solution and computing the associated rhs, we have already proved the validity of this application. Nevertheless, we can also use the exact solution in 1D and show that is also working for it.

12 Thermal dilatation

To what has been done so far, we add a new term, which is the thermal dilatation one. The new equation is :

$$(\lambda + \mu)\mathbf{grad}\,\mathrm{div}\,u + \mu\Delta u - (3\lambda + 2\mu)\alpha(T - T_{ref}) + F = 0, \tag{37}$$

α is the dilatation parameter.

Metal materials (and other type materials) have some parameters which depend on the ambient temperature; in particular, there is a common phenomenon of dilatation : the object volume changes when there is variation of the ambient temperature. The added term models this dilatation.

The changes to do are :

Variationnal form. add the term $-\int_\Omega (3\lambda + 2\mu)\alpha(T - T_{ref})(v_r + v_z)$ to the bilinear form

Code add this term in the code :

```
1 //thermal dilatation
2 form1( _test=Xh, _vector=rhs ) +=
3 integrate( _range=elements( mesh ),
4 _expr= -Px()*dilatation*(3*lambda + 2*mu)*(( idv(dilat_T) - idv(ref_T) )*id( vr
      ) + ( idv(dilat_T) - idv(ref_T) )*id( vz )) );
```

Part IV
Electromagnetism model

13 Introduction

Lastly, we came to the most important point I think of this project : the study of Maxwell's equations, that are the electromagnetism equations. Of course, because the point of this project are electromagnets, and so, magnetic field generation (!).

The electro-magnets are submitted to a certain voltage and current intensity, which will induce a massive movement of electrons from one pole to the other. The moving charges (because electrons have charges) create the wanted magnetic field.

13.1 Equations

As always, we begin with the equation which describes the problem :

$$\begin{aligned} \mathbf{rot}\,(\nu(\mathbf{rot}\,(\vec{A}))\mathbf{rot}\,(\vec{A})) + \sigma\frac{\partial \vec{A}}{\partial t} &= -\sigma\nabla V \quad \text{in } \Omega \\ \operatorname{div}(\sigma\nabla V) &= 0 \quad \text{in } \partial\Omega \end{aligned} \tag{38}$$

- σ is the electrical conductivity
- V is naturally the electrical potential
- ν is the vacuum permittivity, and can depend on $\vec{A}$

$\vec{A}$ is called magnetic potential and its link with the magnetic field will be described further. One particular property of it, is that in axisymmetric, $\vec{A}$ has a component only in u_θ, the others are null : this property will be usefull for the variationnal formulation.

It is important I think to show how to establish this equation : that does help to manipulate it more easily, to verify that is written without errors and also for general knowledge.

In a local way, Ohm's law is written :

$$\vec{j} = -\sigma\nabla V - \sigma\vec{E}_{em} \tag{39}$$

$\vec{E}_{em}$ is the current generated by a changing magnetic environment (it is explained further); it can be expressed like : $\vec{E}_{em} = \dfrac{d\vec{A}}{dt}$.

According to the Maxwell-Ampère's equation, we have :

$$\mathbf{rot}\,(\vec{B}) = \mu\vec{j}, \tag{40}$$

where $\vec{B}$ is the magnetic field.

By replacing $\vec{j}$ in this last equation we have :

$$\mathbf{rot}\,(\vec{B}) = -\mu\sigma\nabla V - \mu\sigma\vec{E}_{em} \tag{41}$$

Another equation, this one from Maxwell-Thomson says that :

$$\operatorname{div}\vec{B} = 0 \tag{42}$$

Vectorial analysis formulas imply that :

$$\operatorname{div}(\vec{F}) = 0 \Longleftrightarrow \vec{F} = \mathbf{rot}\,(\vec{g})$$

which gives :

$$\vec{B} = \mathbf{rot}\,(\vec{A})$$

and so, we have the equation given at the begining :

$$\frac{1}{\mu}\mathbf{rot}\,(\mathbf{rot}\,\vec{A}) = -\sigma\nabla V - \sigma\frac{\partial\vec{A}}{\partial t} \tag{43}$$

as ν is the vacuum permittivity, μ is the vacuum resistivity; and for the moment, we consider all these parameters as constants.

Small modifications As the $\mathbf{rot}\,(\mathbf{rot}\,\vec{A})$ notation cannot be easily manipulated in the variationnal formulation, we should transform it according to vectorial analysis formula :

$$\mathbf{rot}\,(\mathbf{rot}\,\vec{A}) = \mathbf{grad}\,(\mathrm{div}\,\vec{A}) - \Delta\vec{A}$$

Then, with the Coulomb gauge, we get :

$$\mathrm{div}\,\vec{A} = 0$$

So we have :

$$-\nu\Delta A = -\sigma\nabla V - \sigma\frac{\partial\vec{A}}{\partial t} \tag{44}$$

Yet, it is not finished : there is special way of solving for this equation in axisymmetric. According to the notes [**?**] and [**?**], we have to introduce ψ like :

$$\vec{\psi} = \vec{A}r$$

and introduce Δ^* notation like :

$$\Delta^* = \Delta - \frac{2}{r}\frac{\partial\cdot}{\partial r}$$

It would be also useful to read C. Daversin's note [**?**] to understand the axisymmetrical formulation. Consequently by rewritting the equation, we have :

$$-\nu\frac{1}{r}\Delta^*\vec{\psi} = -\sigma\nabla V - \sigma\frac{\partial\vec{A}}{\partial t} \tag{45}$$

The last step :

$$\begin{aligned} -\nu\Delta^*\vec{\psi} &= -\sigma\nabla V r - \sigma r\frac{\partial\vec{A}}{\partial t} \\ &= -\sigma\nabla V r - \sigma\frac{\partial\vec{\psi}}{\partial t} \end{aligned} \tag{46}$$

14 Variationnal formulation

The next step is obviously the establishment of the variationnal formulation and verifying that it is indeed correctly implemented in the HifiMagnet project pre-existent code. The Dirichlet's boundary conditions are written in their strong form.

We consider the domain Ω, the boundary $\delta\Omega$, and $\delta\Omega_D$ the boundary on which we apply Dirichlet boundary conditions. We define the associated function spaces $H^1_g(\Omega) = \{u \in H^1(\Omega), u = g \text{ on } \partial\Omega_D\}$ for the solution, and $H^1_0(\Omega) = \{v \in H^1(\Omega), v = 0 \text{ on } \partial\Omega_D\}$ for the test functions.

14.1 Stationnary

By multiplying with a test function $v_h \in H_0^1(\Omega)$ and integrating over the domain, we have :

$$-\int_\Omega \nu\Delta^*\vec{\psi}v_h = -\int_\Omega \sigma\nabla V r v_h \tag{47}$$

From Green's theorem, the left hand side is modified :

$$\begin{aligned} -\int_\Omega \nu\Delta^*\vec{\psi}v_h &= -\int_\Omega \nu\Delta\vec{\psi}v_h + \int_\Omega \nu\frac{2}{r}\frac{\partial\vec{\psi}}{\partial r}v_h \\ &= \int_\Omega \nu\nabla\vec{\psi}\cdot\nabla v_h - \int_{\partial\Omega}\nu\frac{\partial\vec{\psi}}{\partial n}v_h + \int_\Omega \nu\frac{2}{r}\frac{\partial\vec{\psi}}{\partial r}v_h \end{aligned} \tag{48}$$

Thus, in stationnary, we have to solve :

Find $\psi = rA$, in $H_g^1(\Omega)$, such that $\forall v \in H_0^1(\Omega)$:

$$\int_\Omega \nu\nabla\vec{\psi}\cdot\nabla v_h - \int_{\partial\Omega}\nu\frac{\partial\vec{\psi}}{\partial n}v_h - \int_\Omega \nu\frac{2}{r}\frac{\partial\vec{\psi}}{\partial r}v_h = -\int_\Omega \sigma\nabla V r v_h \tag{49}$$

14.2 Instationnary

We can use the same discretization as in HeatAxi in temporal mode, that is to say an implicit Euler's method. Here, we are showing only the backward Euler's scheme, because the other higher order method will be also written in the same manner.

The equation we have is of the form :

$$\sigma\frac{\partial\vec{\psi}}{\partial t} = F(t,\vec{\psi}^{n+1}), \tag{50}$$

where :

$$F(t,\vec{\psi}^{n+1}) = \nu\Delta^*\vec{\psi}^{n+1} - \sigma r\nabla V.$$

The implicit method discretization is :

$$\sigma\frac{\partial\vec{\psi}}{\partial t} = \sigma\frac{\vec{\psi}^{n+1}-\vec{\psi}^n}{\Delta t}, \tag{51}$$

the overall equation in instationnary becomes :

$$\sigma\frac{\vec{\psi}^{n+1}}{\Delta t} - \nu\Delta^*\vec{\psi}^{n+1} = -\sigma r\nabla V + \sigma\frac{\vec{\psi}^n}{\Delta t}, \tag{52}$$

and from that, the variationnal formulation :

Find $\psi = rA$, in $H_g^1(\Omega)$, such that $\forall v \in H_0^1(\Omega)$:

$$\int_\Omega \sigma\frac{\vec{\psi}^{n+1}}{\Delta t}v_h + \int_\Omega \nu\nabla\vec{\psi}\cdot\nabla v_h - \int_{\partial\Omega}\nu\frac{\partial\vec{\psi}}{\partial n}v_h + \int_\Omega \nu\frac{2}{r}\frac{\partial\vec{\psi}}{\partial r}v_h = -\int_\Omega \sigma\nabla V r v_h + \int_\Omega \sigma\frac{\vec{\psi}^n}{\Delta t}v_h \tag{53}$$

15 Implementation in Feel++

The implementation in Feel++ does not add any difficulty. Again, as for the other models, one should not forget the *Px()* term (which represents the r space variable) due to the cylindrical coordinates system. Several blocks of code providing from HeatAxi where copied and pasted, sometimes with small changes. An example is the rhs computation code : we have only modified the equation, the rest remained the same.

```
// linear form
form1( _test=Mh, _vector=F) = integrate( _range = elements(mesh), _expr = - Px()*
    sigma*ddp/( 2* M_PI )*id(v) );
form1( _test=Mh, _vector=F) += integrate( _range = elements(mesh), _expr = - Px()*
    js*Px()*id(v) );

// bilinear form
form2( _test=Mh, _trial=Mh, _matrix=D) = integrate( _range=elements(mesh), _expr=
    nu*Px()*gradt(psi)*trans(grad(v)) );
form2( _test=Mh, _trial=Mh, _matrix=D) += integrate( _range=elements(mesh), _expr=
    nu*2*dxt(psi)*id(v) );

// strong dirichlet condition
form2( _test=Mh, _trial=Mh,_matrix= D) += on( markedfaces(mesh, physical), psi, F,
     _c->constant()->getExpr() );

//solving the system
backend_type::build()->solve( _matrix=D, _solution=psi, _rhs=F );
```

16 Stationnary mode

We chose the exact function :

$$sin(\pi x)sin(\pi y)$$

We let the program computing the rhs, and we make a convergence study by dividing the mesh size four times. The geometry is a rectangle 1x2, centered in the point $(0; 1)$; thus in the boundary, the solution is null.

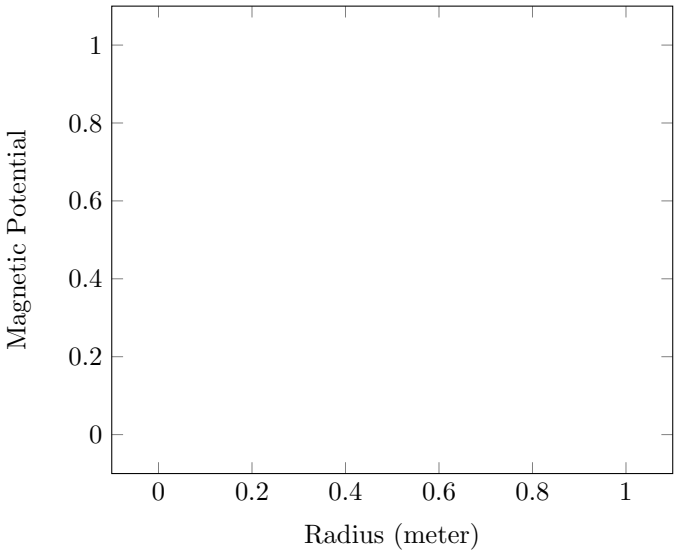
16.1 First outputs

The Paraview output of the domain with the exact function :

images/magneto_A.jpg

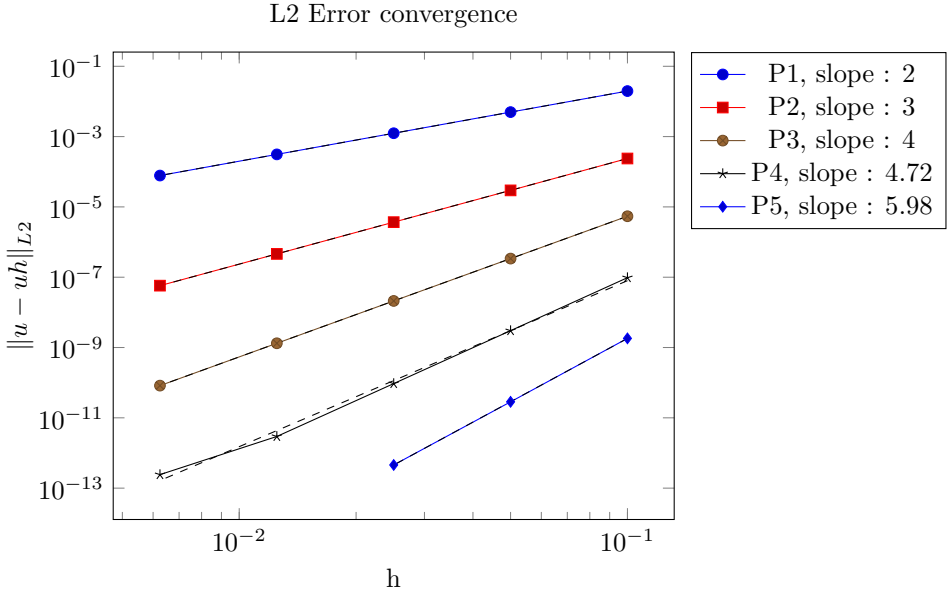
And indeed, on the boundaries, the function is null.

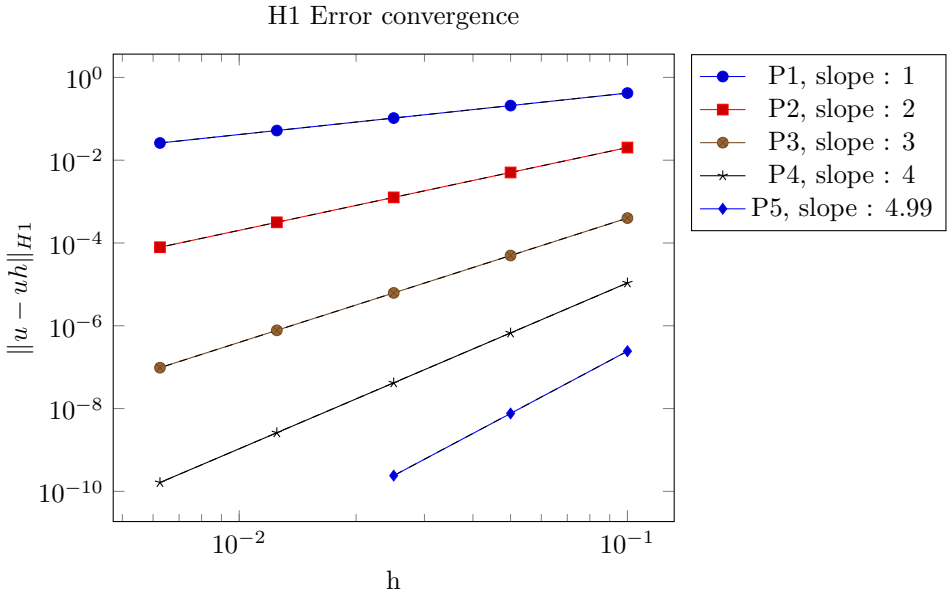
We can also plot the exact magnetic potential and the approximated one :



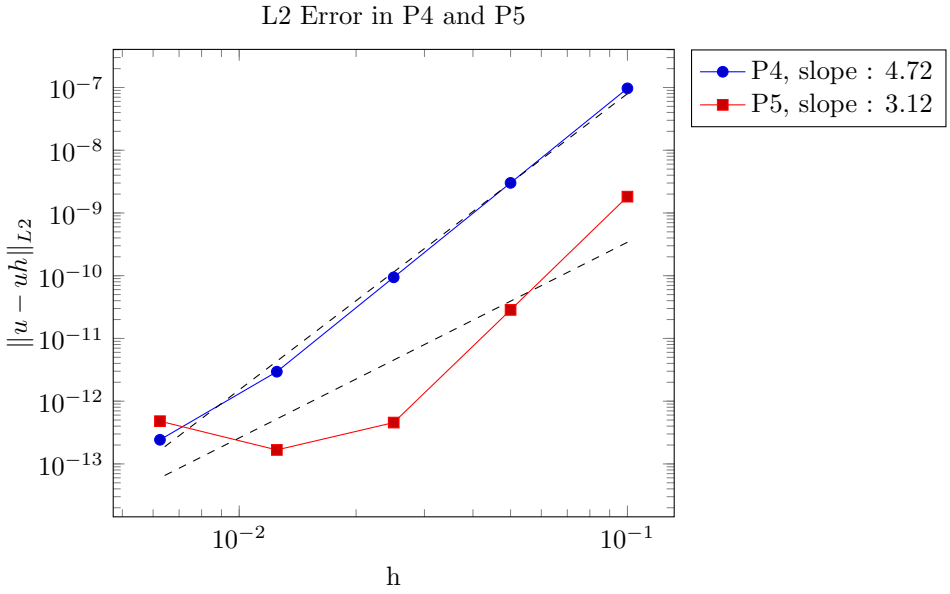
16.2 Convergence study

Then the convergence study graphs : the evolution of the L^2 and H^1 error when the mesh is refined.





For the polynomial order 5, there is only 3 points in the graph instead of 5; it is because the precision issue explained before. Here you can have a look on what is happening :



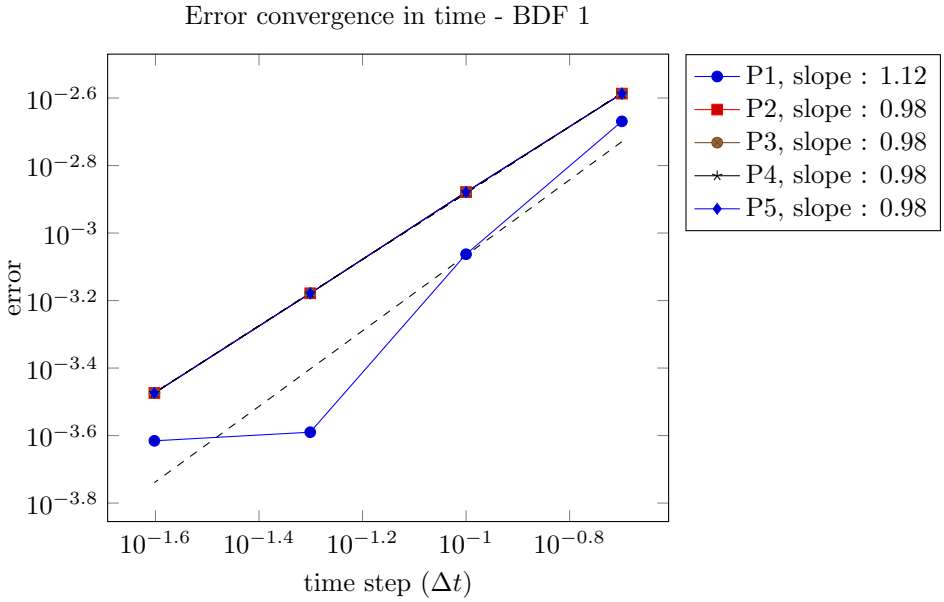
17 Instationnary mode

We chose this time the exact solution :

$$\sin(\pi x)\sin(\pi y)e^{-t} \tag{54}$$

17.1 Convergence study

The geometry is the same as for the stationnary study and the mesh size is 0.02 for P1, P2 and P3; it is 0.05 for P4 and P5. The initial time is set to 0; the final time to 1.



17.2 Conclusion

Due to the lack of time to get results, we couldn't finalize the BDF convergence study. We made it only for BDF order 1. The graphs show a very good convergence slope.

Part V
Coupling and Conclusion

18 Coupling

18.1 Introduction

The last part of this project is to couple the 3 models and get One big model. For that there are 2 types of coupling :

Monolithic it consists in solving at the same time the 3 models

Partitioned it consists in solving each model one after another

We will use the partitioned coupling method, because something similar has already been done (C. Daversin's code) ; it would be easy to begin with that and to understand better.

18.2 Heat transfer and elasticity

We present here only the coupling between the heat transfer model and the elasticity model. In fact, the domain for those 2 models are the same, but for the electromagnetism model the domain is something widen, so this problem should be handled differently. We didn't had enough time to solve this point.

As we chose the partitioned method, the first question is what model should we first compute ? Heat transfer model equation does not depend on the value of the displacement of the deformation, but in the equation of the linear elasticity the thermal dilatation intervenes. We must compute the thermic model then inject the found temperature into the elasticity model before solving it.

Then all what we need is to call each model's function and then export the results so it can be viewed in Paraview. Here is the squeletton of the code (big pieces of codes are omitted) :

```
// mesh creation
...

// materials and boundary condition
...
std::vector<condition_ptrelt> boundary_conditions_T = conditions["temperature"];
std::vector<condition_ptrelt> boundary_conditions_U = conditions["elasticity"];

// function space for the temperature (1D scalar space)
auto Xh_T = space_type::New(mesh);

// function space for elasticity (2D scalar space)
auto Xh_U = El_space_type::New(mesh);

// HeatAxi model
element_type T(Xh_T);

T = heataxi_model.temperature_model(mesh, Xh_T, materials, boundary_conditions_T,
    T_old, Rhs_T, M_bdf_T);

// Elasticity model
El_element_type U(Xh_U);

```

```
23 U = linelaxi_model.elasticity_model(mesh, Xh_U, U, V, Rhs_el, T, ref_T, materials,
       boundary_conditions_U);
24
25 // Export results
26 ...
```

As you can see, we only call the function used in the different implementation, nothing else has to be done. About the V&V process, as the models were verified and validated separately, the coupling was also in the same time verified and validated.

It is not true : the coupling has to be verified and validated, even if the two associated models has been already validated

The coupling work is not finished, electromagnetism model has to be added and then the BDF structure.

19 Conclusion

19.1 Achieved work

The aim was to create a complete 3D model of electromagnets in axisymmetric. So far, we didn't manage to complete the objectives, we only did :

- Heat transfer model : stationnary and instationnary, but only the linear model
- Heat transfer equation exact solution : exploration of several ways for solving this equation in 2D axisymmetric
- Elasticity model : stationnary mode only
- Electromagnetism model : stationnary and instationnary, but only the linear model
- Coupling : the coupling was done between the heat transfer and the elasticity models

19.2 Difficulties

The first main difficulty was the Feel++ API. Yet, after six months, I am wondering why it was so difficult to start; it might be the lack of experience. For sure, it was a very good exercice to learn how to use a library, to contribute somehow to it and help other people.

The other difficulty was the mathematical side : even if during the master class I learned FEM methods and other discretisation methods, it is another thing when it comes to implement a complete model. It is there one can see if the theorical aspect was understood or not. To my opinion, I have better understood the FEM method during this internship than before, but also other points like :

- the real role of error estimation : it is their study which says if the implementation is correct or not. It is not just another *useless* and *annoying* theorem !
- the finite difference method : especially backward Euler's methods (with the BDF structure of Feel++).
- quadrature of integrals : in HeatAxi we fix the quadrature order manually, but for polynomial order 4 and 5, the quadrature order was not high enough : we couldn't obtain a good convergence slope. We found it after several hours of debugging.
- Dirichlet's boundary conditions : the real difference between weak and strong Dirichlet's conditions and how to implement them.

19.3 Perspectives

There is always something to add, to improve and to change, particularly for projects like this one :

HeatAxi it is important to add the non-linear mode for the parameters k, ρ and C_p in order to have a complete model of the heat transfer.

Elasticity to complete the model we need to add the double time derivative : $\frac{\partial u^2}{\partial^2 t}$. Maybe with a finite difference discretization.

Electromagnetism finalize the temporal study similarly to what has been done for HeatAxi.

Coupling add electromagnetism model to the coupling of HeatAxi/Elasticity.

Coupling-instationnary make the coupling working in instationnary and validate the results.

Then, in a practical view, it would be useful to add some Doxygen documentation which will help other people to understand the code easily when they have to work on it.

19.4 General conclusion

It was for sure a great experience : I learned plenty of things about as well the mathematics as relational. I wish I could have started more rapidly and so could be able to finish the entire project or at least add the temporal mode to the elasticity model. Nevertheless I am glad to see that what we have done is correct (that's to say, the convergence study both in stationnary and in instationnary) and working fine.

I would like also to say some words about the PRACE seminar I attended in the last week of march in the Maison de la Simulation (Saclay, France). It was about parallel solvers for linear algebra. The purpose is definitely something new and for me it was only an overview of this branch of mathematics/computer science. The exchanges with other PhD students and researchers were really benefitful.

During this seminar I realized although high performance computing is interesting, powerful and useful, what I prefer most and what interest me a lot in applied mathematics is the modelling and simulation sides !

Part VI
Appendix

A Appendix A : Calcul 1D du champ de température dans les aimants Bitter

Dans ce document, nous allons établir l'expression analytique de la température dans un aimant axisymétrique de section rectangulaire parcouru par une distribution de courant de type Bitter.

A.1 Equation de la chaleur

L'équation de la chaleur s'écrit simplement sous la forme

$$\rho C_p \frac{\partial T}{\partial t} - \text{div } (k \, \mathbf{grad}\, T) = \sigma \left(\frac{U}{2\pi r} \right)^2, \tag{55}$$

avec

ρ	la masse volumique du matériau,
C_p	la capacité thermique du matériau,
k	la conductivité thermique du matériau,
σ	la conductivité électrique du matériau

et

U	la différence de potentiel appliquée aux bornes de l'aimant.

Ici nous nous intéresserons qu'à la solution stationnaire dans un domaine cylindrique creux. Nous supposerons de plus que T est uniquement fonction de r.

Sous ces hypothèses, (A.3) devient alors :

$$-\frac{1}{r}\frac{\partial}{\partial r}\left(rk \frac{\partial T}{\partial r} \right) = \sigma \left(\frac{U}{2\pi r} \right)^2. \tag{56}$$

On montre facilement que :

$$rk\frac{\partial T}{\partial r} = -\sigma \left(\frac{U}{2\pi} \right)^2 \ln r + Ak,$$

où A désigne une constante. Soit finalement :

$$T = A \ln r - \frac{\sigma}{2\,k} \left(\frac{U}{2\pi} \right)^2 \ln^2 r + B. \tag{57}$$

Les constantes d'intégration A et B sont déterminées par les conditions aux limites appliquées sur les rayons internes et externes, respectivement r_1 et r_2, de l'aimant.

Température moyenne

Par définition la température moyenne $< T >$ est donnée par :

$$< T >= \frac{1}{\pi \left(r_2^2 - r_1^1 \right)} \int_{r_1}^{r_2} T(r)\, r dr.$$

Soit :

$$< T >= \frac{1}{\pi \left(r_2^2 - r_1^2 \right)} \frac{1}{2} \left[r^2 \left(B + A(\ln r - \frac{1}{2}) - \frac{\sigma}{2k} \left(\frac{U}{2\pi} \right)^2 (\ln^2 r - \ln r + \frac{1}{2}) \right) \right]_{r_1}^{r_2} \tag{58}$$

Température maximum

La température sera maximale pour r tel que $\frac{\partial T}{\partial r} = 0$. Pour trouver la position du maximum de température il nous faut donc résoudre l'équation :

$$A - \frac{\sigma}{k} \left(\frac{U}{2\pi} \right)^2 \ln r = 0$$

d'où :

$$r = e^{\frac{A\,k}{\sigma\left(\frac{U}{2\pi}\right)^2}}. \tag{59}$$

Finalement on obtient :

$$T_{max} = \frac{A^2\,k}{2\sigma\,\left(\frac{U}{2\pi}\right)^2} + B. \tag{60}$$

A.2 Convection forcée

Dans cette partie nous supposerons que les conditions aux limites sont du type :

$$k\frac{\partial T}{\partial \mathbf{n}} = h(r)\left(T(r) - T_{ambiant}(r)\right), \tag{61}$$

avec $\mathbf{n}$ la normale unitaire extérieur à l'aimant. Par abus de notation h_1, h_2 désigneront respectivement $h(r_1)$ et $h(r_2)$. De même nous posons T_1, T_2 pour $T_{ambiant}(r_1)$ et $T_{ambiant}(r_2)$.

En appliquant ces conditions en $r1$ et $r2$ il vient immédiatement :

$$\begin{cases} A(-k - h_2 r_2 \ln r_2) - B h_2 r_2 & = -\dfrac{\sigma}{2\,k}\left(\dfrac{U}{2\pi}\right)^2 \ln r_2\,(h_2 r_2 \ln r_2 + 2k) - h_2 r_2 T_2, \\ A(k - h_1 r_1 \ln r_1) - B h_1 r_1 & = -\dfrac{\sigma}{2\,k}\left(\dfrac{U}{2\pi}\right)^2 \ln r_1\,(h_1 r_1 \ln r_1 - 2k) - h_1 r_1 T_1 \end{cases}$$

D'où :

$$A = \frac{\sigma}{2k}\left(\frac{U}{2\pi}\right)^2 2\frac{\frac{1}{2}\ln(r_1 r_2)\ln\frac{r_2}{r_1} + k\left(\frac{\ln r_2}{h_2 r_2} + \frac{\ln r_1}{h_1 r_1}\right)}{k\left(\frac{1}{h_2 r_2} + \frac{1}{h_1 r_1}\right) + \ln\frac{r_2}{r_1}} \tag{62}$$

$$+\frac{T_2 - T_1}{k\left(\frac{1}{h_2 r_2} + \frac{1}{h_1 r_1}\right) + \ln\frac{r_2}{r_1}}. \tag{63}$$

On obtient que :

$$T = B - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2\left[\ln^2\left(\frac{r}{r_0}\right) - \ln^2(r_0)\right] + \frac{T_2 - T_1}{k\left(\frac{1}{h_2 r_2} + \frac{1}{h_1 r_1}\right) + \ln\frac{r_2}{r_1}}\ln r, \tag{64}$$

avec :

$$\ln r_0 = \frac{\frac{1}{2}\ln(r_1 r_2)\ln\frac{r_2}{r_1} + k\left(\frac{\ln r_2}{h_2 r_2} + \frac{\ln r_1}{h_1 r_1}\right)}{k\left(\frac{1}{h_2 r_2} + \frac{1}{h_1 r_1}\right) + \ln\frac{r_2}{r_1}}. \tag{65}$$

Remarquons que cette expression peut aussi s'écrire :

$$\ln r_0 = \frac{1}{2}\ln(r_1 r_2) + \frac{k}{2}\,\frac{\left(\frac{1}{h_2 r_2} - \frac{1}{h_1 r_1}\right)}{k\left(\frac{1}{h_2 r_2} + \frac{1}{h_1 r_1}\right) + \ln\frac{r_2}{r_1}}\,\ln\frac{r_2}{r_1}. \tag{66}$$

A.2.1 Calcul de B dans le cas où $T_2 = T_1$

Dans le cas où $T_2 = T_1 = T_f$ on a simplement:

$$T = B - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2\left[\ln^2\left(\frac{r}{r_0}\right) - \ln^2(r_0)\right] \tag{67}$$

La dérivée de T s'écrit :

$$T' = -\frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2\left[2\frac{\ln(r/r_0)}{r/r_0}\right] \tag{68}$$

Extremum en r_0 La résolution de l'équation : $T' = 0$ permet de trouver les extremums :

$$\begin{aligned} T' =& 0 \Leftrightarrow \\ -\frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2\left[2\frac{\ln(r/r_0)}{r/r_0}\right] =& 0 \Leftrightarrow \\ \ln(r/r_0) =& 0 \end{aligned} \tag{69}$$

Ce qui donne qu'en r_0, il y a un extremum.

Maximum ou minimum Il va falloir résoudre les inéquations $T' < 0$ et $T' > 0$ pour trancher :

$$\begin{aligned} T' &< 0 \Leftrightarrow \\ -\frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2\left[2\frac{\ln(r/r_0)}{r/r_0}\right] &< 0 \Leftrightarrow \\ \frac{\ln(r/r_0)}{r/r_0} &> 0 \Leftrightarrow \\ \ln(r/r_0) &> 0 \quad car\, r > 0 \\ r &> r_0 \end{aligned} \tag{70}$$

Par le même raisonnement :

$$\begin{aligned} T' &> 0 \\ \ln(r/r_0) &< 0 \\ r &< r_0 \end{aligned} \tag{71}$$

Variations de T Ainsi :

- T est croissante sur $[r_1; r_0]$ car $T' > 0$ pour $r < r_0$
- T est décroissante sur $[r_0; r_1]$ car $T' < 0$ pour $r > r_0$

T a donc une allure de parabole retourné avec un maximum en r_0 : compte tenu du refroidissement sur les deux extrémités du domaines, il est réaliste d'avoir une telle allure. Nous le vérifierons avec la solution analytique et par le calcul numérique.

Cas où $h_1 r_1 = h_2 r_2$ On pose maintenant que :

$$h_2 r_2 = h_1 r_1$$

Ce qui permet de simplifer $\ln(r_0)$ en :

$$\ln r_0 = \frac{1}{2}\ln(r_1 r_2) \tag{72}$$

Ce qui donne :

$$A = \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln(r_1 r_2)$$

En additionnant les deux lignes du système d'équation de la convection forcée, on a :

$$-Ah_1 r_1 \ln(r_1 r_2) - 2Bh_1 r_1 = -\frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2\left[2k\ln(r_2/r_1) + h_1 r_1\left[\ln^2(r_2) + \ln^2(r_1)\right]\right] - 2h_1 r_1 T_1 \tag{73}$$

Après les simplifications, il reste :

$$B = T_1 - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln(r_1)\ln(r_2) + \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 k\left(\frac{\ln(r_2) - \ln(r_1)}{h_1 r_1}\right) \tag{74}$$

Et l'expression de T devient :

$$T = B - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \left(\ln^2(r/r_0) - \ln^2(r_0)\right) \tag{75}$$

On peut le réécrire en :

$$T = T_{max} - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln^2(r/r_0) \tag{76}$$

avec

$$\begin{aligned} T_{max} &= B + \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln^2(r_0) \\ T_{max} &= T_1 - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln(r_1)\ln(r_2) + \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 k\left(\frac{\ln(r_2)-\ln(r_1)}{h_1 r_1}\right) + \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln^2(r_0) \\ T_{max} &= T_1 + \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \left(-\ln(r_1)\ln(r_2) + k\frac{\ln(r_2)-\ln(r_1)}{h_1 r_1} + \ln^2(r_0)\right) \end{aligned} \tag{77}$$

Cas où $h_1 r_1 \neq h_2 r_2$ Dans ce cas, aucune simplification notable est à faire. Il va falloir résoudre le système formé par les 2 équations de la convection forcée (début similaire au calcul de A). Ce qui nous donne donc :

$$\begin{aligned} P * B =& Mk * (h_1 r_1 \ln^2(r_1) + h_2 r_2 \ln^2(r_2)) + 2Mk^2 \ln\frac{r_2}{r_1} + kT_1 * (h_1 r_1 + h_2 r_2) \\ =& - h_1 r_1 h_2 r_2 M \ln(r_1)\ln(r_2)\ln\frac{r_2}{r_1} - 2Mk\ln(r_1)\ln(r_2) * (h_1 r_1 + h_2 r_2) + h_1 r_1 h_2 r_2 \ln\frac{r_2}{r_1} \end{aligned} \tag{78}$$

avec :

$$M = \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2$$

et

$$P = k(h_1 r_1 + h_2 r_2) + h_1 r_1 h_2 r_2 \ln\frac{r_2}{r_1}$$

Du fait qe $T_1 = T_2$, l'expression de A peut être simplifiée en :

$$A = M\ln r_0 = M\frac{1}{2}\ln(r_1 r_2) + M\frac{k}{2}\,\frac{\left(\frac{1}{h_2 r_2} - \frac{1}{h_1 r_1}\right)}{k\left(\frac{1}{h_2 r_2} + \frac{1}{h_1 r_1}\right) + \ln\frac{r_2}{r_1}}\,\ln\frac{r_2}{r_1}. \tag{79}$$

T garde bien entendu la même forme :

$$T = T_{max} - \frac{\sigma}{2\,k}\left(\frac{U}{2\pi}\right)^2 \ln^2(r/r_0)$$

Avec : $T_{max} = B + M\ln^2 r_0$

A.3 Equation de la chaleur instationnaire

L'équation considérée est celle de départ, pour rappel :

$$\rho C_p \frac{\partial T}{\partial t} - \text{div}\ (k\,\mathbf{grad}\,T) = \sigma\left(\frac{U}{2\pi r}\right)^2,$$

Les conditions aux limites sont les mêmes que précédemment, celles de Robin :

$$k\frac{\partial T}{\partial \mathbf{n}} = h(r)\left(T(r) - T_{ambiant}(r)\right), \quad \forall\, t$$

avec $\mathbf{n}$ la normale unitaire extérieur à l'aimant. h_1, h_2 désigneront respectivement $h(r_1)$ et $h(r_2)$. Nous posons T_1, T_2 pour $T_{ambiant}(r_1)$ et $T_{ambiant}(r_2)$.

Les conditions initiales :

$$T(0,r) = T_0 \quad \text{pour } r \in [r_1; r_2]$$

Nous allons voir plusieurs méthodes pour calculer cette solution.

A.3.1 Méthode de la séparation des variables

On commence par la méthode classique de la séparation des variables. Soit T_{hom} la solution, on suppose qu'elle est de la forme : $T_{hom}(r,t) = y(r)e^{\frac{-t}{\tau}}$. En injectant dans l'équation homogène, on a :

$$\begin{aligned}
\rho C_p \frac{\partial T_{hom}}{\partial t} - \text{div}\ (k\,\mathbf{grad}\, T_{hom}) &= 0 \\
\rho C_p y(r)\frac{\partial e^{-\frac{t}{\tau}}}{\partial t} - e^{\frac{-t}{\tau}}\ \text{div}\ (k\,\mathbf{grad}\, y(r)) &= 0 \\
-\frac{\rho C_p}{\tau} y(r) e^{-\frac{t}{\tau}} - e^{\frac{-t}{\tau}} k \frac{\partial^2 y(r)}{\partial r^2} - e^{\frac{-t}{\tau}} k \frac{1}{r}\frac{\partial y(r)}{\partial r} &= 0
\end{aligned} \tag{80}$$

On pose alors : $\tau = \frac{\rho C_p}{k}$ et on obtient cette équation différentielle, qui est une équation de Bessel :

$$\frac{\partial^2 y(r)}{\partial r^2} + \frac{1}{r}\frac{\partial y(r)}{\partial r} + y(r) = 0 \tag{81}$$

D'après les informations que l'on peut trouver dans la littérature, les solutions de cette équation sont de la forme :

$$y(r) = QJ_0(r) + RY_0(r) \tag{82}$$

avec $Q, R \in R$, et J_0 étant la fonction de Bessel d'ordre 0 de première espèce et Y_0 la fonction de Bessel d'ordre 0 de deuxième espèce.

Les conditions initiales nous donnent :

$$T(0,r) = QJ_0(r) + RY_0(r) = T_0$$

Les conditions aux bords permettent d'avoir le système suivant :

$$\begin{cases}
ke^{\frac{-t}{\tau}} y'(r_1) = h_1\left(y(r_1)e^{\frac{-t}{\tau}} - T_1\right) \\
-ke^{\frac{-t}{\tau}} y'(r_2) = h_2\left(y(r_2)e^{\frac{-t}{\tau}} - T_2\right) \quad \forall\, t
\end{cases} \tag{83}$$

En ré-organisant les termes, on a :

$$\begin{cases}
e^{\frac{-t}{\tau}}\left(ky'(r_1) - h_1 y(r_1)\right) = -h_1 T_1 \\
e^{\frac{-t}{\tau}}\left(-ky'(r_2) - h_2 y(r_2)\right) = -h_2 T_2 \quad \forall\, t
\end{cases} \tag{84}$$

Il n'y pas de solutions à ce système.

Conclusion : Il n'y pas de solutions de la forme $T(r,t) = y(r)e^{\frac{-t}{\tau}}$, il faut utiliser une autre méthode que celle de la séparation des variables.

A.3.2 Méthode de la transformée de Laplace

Définition A une fonction $f(t)$ défini sur $[0, \infty[$, on associe sa transformée de Laplace $F(p)$, noté $Ł(f)$, par la relation :

$$F(p) = \int_0^\infty f(t)e^{-pt}dt \quad \forall p \in \mathbb{C} \tag{85}$$

Nous supposerons que les conditions pour la convergence de cette intégrale et donc de l'existence de la transformée de Laplace, sont vérifiées.

Méthode : Nous utiliserons cette transformation pour trouver la solution, pour cela :

1 Il faut appliquer cette transformation à l'équation, ainsi qu'aux conditions initiale et de bords

2 La résoudre dans l'espace symbolique

3 Revenir avec une transformée inverse en utilisant soit les tables de passage ou la formule d'inversion de Mellin

Remarque : Cette méthode s'inspire du livre *Transferts thermiques* [**?**].

Etape 1 On reprend donc l'équation homogène de départ (écrite dans le repère cylindrique):

$$\rho C_p \frac{\partial T}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(rk\frac{\partial T}{\partial r}\right)$$

Grâce au table de passage, on a que si $Ł(f(t)) = F(p)$, alors $Ł(f'(t)) = pF(p) - f(0^+)$; nous avons alors :

$$\rho C_p p\Gamma - T_0 = \frac{1}{r}\frac{d}{dr}(rk\frac{d\Gamma}{dr}) \tag{86}$$

où $\Gamma(r,p) = Ł(T(r,t))$.

Sachant que $Ł(cste) = cste/p$, les conditions de bords s'écrivent :

$$\begin{cases} k\dfrac{d\Gamma}{dr}(r_1,p) = h_1(\Gamma(r_1,p) - \frac{T_1}{p}) \\ -k\dfrac{d\Gamma}{dr}(r_2,p) = h_2(\Gamma(r_2,p) - \frac{T_2}{p}) \end{cases} \tag{87}$$

La condition initiale est directement intégrée dans l'équation transformée.

Etape 2 Il s'agit maintenant de trouver une solution dans cet espace transformée de l'équation sans second membre :

$$\frac{d^2\Gamma}{dr^2} + \frac{1}{r}\frac{d\Gamma}{dr} - p\Gamma = 0 \tag{88}$$

On fait un changement de variable en posant :

$$p = q^2, \quad u = rp^{1/2} = rq$$

Ce qui donne finalement une équation de Bessel modifiée :

$$u^2\frac{d^2\Gamma}{du^2} + u\frac{d\Gamma}{du} - u^2\Gamma = 0 \tag{89}$$

La solution générale s'écrit sous la forme suivante, avec I_0 et K_0 les fonctions de Bessel modifiées :

$$\Gamma(r,p) = AI_0(qr) + BK_0(qr) \tag{90}$$

Une solution particulière (évidente) étant :

$$\Gamma_{part} = \frac{1}{p}\frac{T_0}{\rho C_p} \tag{91}$$

on a l'écriture de la solution dans l'espace transformée :

$$\Gamma(r,p) = AI_0(qr) + BK_0(qr) + \frac{1}{p}\frac{T_0}{\rho C_p} \tag{92}$$

L'étude asymptotique de ces fonctions de Bessel affirment que lorsque $r \longrightarrow 0$, $I_0 \longrightarrow 0$ et $K_0 \longrightarrow \infty$. Ce qui pourrait impliquer que $B = 0$, mais du fait de la géométrie cylindrique *creuse*, $r \neq 0$ et K_0 ne divergera pas.

Comme pour la partie stationnaire, nous devons résoudre le système de conditions aux bords :

$$\begin{cases} kAqI_0'(qr_1) + kBqK_0'(qr_1) - h_1AI_0(qr_1) - h_1BK_0(qr_1) - \dfrac{h_1}{p}\dfrac{T_0}{\rho C_p} + \dfrac{h_1}{p}T_1 = 0 \\ -kAqI_0'(qr_2) - kBqK_0'(qr_2) - h_2AI_0(qr_2) - h_2BK_0(qr_2) - \dfrac{h_2}{p}\dfrac{T_0}{\rho C_p} + \dfrac{h_2}{p}T_2 = 0 \end{cases} \tag{93}$$

On donne le système après la réorganisation des termes et nous allégeons les notations:

$$\begin{cases} A(qkI_0' - h_1I_0)_{r_1} + B(qkK_0' - h_1K_0)_{r_1} = \dfrac{h_1}{p}(\dfrac{T_0}{\rho C_p} - T_1) = \dfrac{H_1}{p} \\ -A(qkI_0' + h_2I_0)_{r_2} - B(qkK_0' + h_2K_0)_{r_2} = \dfrac{h_2}{p}(\dfrac{T_0}{\rho C_p} - T_2) = \dfrac{H_2}{p} \end{cases} \tag{94}$$

Après inversion de la matrice du système, nous obtenons les expressions suivantes pour A et B :

$$A = \frac{1}{p}\frac{-H_1(qkK_0' + h_2K_0)_{r_2} - H_2(qkK_0' - h_1K_0)_{r_1}}{(qkK_0' - h_1K_0)_{r_1}(qkI_0' + h_2I_0)_{r_2} - (qkI_0' - h_1I_0)_{r_1}(qkK_0' + h_2K_0)_{r_2}} \tag{95}$$

$$B = \frac{1}{p}\frac{H_1(qkI_0' + h_2I_0)_{r_2} + H_2(qkI_0' - h_1I_0)_{r_1}}{(qkK_0' - h_1K_0)_{r_1}(qkI_0' + h_2I_0)_{r_2} - (qkI_0' - h_1I_0)_{r_1}(qkK_0' + h_2K_0)_{r_2}} \tag{96}$$

Et ne pas oublier que $p = q^2$.

Etape 3 Pour "revenir" dans l'espace de départ par l'inversion de la transformée de Laplace, on peut utiliser le tableau de passage pour le term en $\frac{1}{p}$, mais il faudra utiliser la formule de Mellin pour les autres termes.

Cette formule s'écrit :

$$f(t) = \lim_{X\to\infty} \frac{1}{2\pi i}\int_{\gamma-iX}^{\gamma+iX} e^{pt}F(p)dp \tag{97}$$

Calculer de manière brute cette intégrale sera compliquée; on fera appel au calcul des résidus.

Sur un contour d'intégration comportant des pôles simples ou multiples d'une fonction $f(z)$, on calcule :

- le résidu en un pôle simple, $z = a$ par:

$$Res(a) = \lim_{z\to a}(z-a)f(z)$$

- le résidu en un pôle d'ordre k :

$$Res(a) = \frac{1}{(k-1)!}\lim_{z\to a}\left(\frac{d^{k-1}}{dz^{k-1}}(z-a)^k f(z)\right)$$

Comme nous le verrons plus tard, ici il n'y a que des pôles simples, ce calcul ne sera donc pas fait.

Les pôles sont donc :

- $p = 0,$ pôle simple
- les zéros en p des dénominateurs de A et de B :

$$(qkK_0' - h_1K_0)_{r_1}(qkI_0' + h_2I_0)_{r_2} - (qkI_0' - h_1I_0)_{r_1}(qkK_0' + h_2K_0)_{r_2} = 0 \tag{98}$$

Il semblerait qu'il y ait une infinité de racines et toutes les trouver ne sera évidemment pas possible. Nous pouvons néanmoins essayer d'en trouver quelques-unes et voir si ça peut être suffisant pour approcher la solution exacte. Nous utiliserons donc Maple pour les calculer. Les fonctions de Bessel y sont déjà implémentés; cependant, bémol, la fonction de Bessel J étant :

$$J_n(x) = \sum_{p=0}^{\infty} \frac{(-1)^p}{p!(n+p)!} \left(\frac{x}{2}\right)^{2p+n} \tag{99}$$

Dans Maple, on ne prend que les premiers exposants (!) :

$$J_n(x) = ...$$

Il est en de même pour toutes les autres fonctions de Bessel : Y, K, I. Il est alors nécessaire de recoder ces fonctions avec des procédures, grâce aux formules du libre de G. Goudet [**?**] et du site WolframMathWolrd [**?**].

Par ailleurs avec les extensions de Maple *Student[Calculus1]* et *Student[NumericalAnalysis]*, on peut trouver les racines d'une fonction. Cela se fait avec l'instruction *Roots*.

Si on a réussi à trouver ces zéros... Supposons que l'on ait trouvé ces zéros et qu'on les note q_n, on a donc que :

$$p = q_1^2, q_2^2, q_3^2, ... \tag{100}$$

On obtient alors :

$$AI(r,t) = \lim_{X\to\infty} \frac{1}{2\pi i} \int_{\gamma-iX}^{\gamma+iX} AI_0(qr)e^{pt}dp = Res(0) + \sum_{n=1}^{\infty} Res(q_n^2) \tag{101}$$

et :

$$BK(r,t) = \lim_{X\to\infty} \frac{1}{2\pi i} \int_{\gamma-iX}^{\gamma+iX} BK_0(qr)e^{pt}dp = Res(0) + \sum_{n=1}^{\infty} Res(q_n^2) \tag{102}$$

Le calcul des résidus aux pôles :

- en $p = 0$:

$$Res(0) = 0 \tag{103}$$

- en $p = q_n^2$:

  une formule très utile pour le calcul des résidus est :

$$f(x) = \frac{P(x)}{x^n Q(x)} \quad \text{avec } P(0).Q(0).P(a) \neq 0 \text{ et } a \text{ racine simple de } Q, \text{ alors } Res(a) = \frac{P(a)}{a^n \frac{dQ}{dx}_{|a}} \tag{104}$$

  ici nous avons aussi un quotient, en l'occurence :

$$f(p,r) = \frac{P1(p,r)}{pQ(p,r)} \quad g(p,r) = \frac{P2(p,r)}{pQ(p,r)}$$

  avec

$$P1(p,r) = \left(-H_1(qkK_0' + h_2K_0)_{r_2} - H_2(qkK_0' - h_1K_0)_{r_1}\right) I_0(qr)e^{pt}$$

$$P2(p,r) = \left(H_1(qkI_0' + h_2I_0)_{r_2} + H_2(qkI_0' - h_1I_0)_{r_1}\right) K_0(qr)e^{pt}$$
$$Q(p,r) = (qkK_0' - h_1K_0)_{r_1}(qkI_0' + h_2I_0)_{r_2} - (qkI_0' - h_1I_0)_{r_1}(qkK_0' + h_2K_0)_{r_2}$$

on sait que les pôles sont en q_n, alors :

$$\frac{dQ}{dp} = \frac{dQ}{dq}\frac{dq}{dp} = acalculer \tag{105}$$

Ce qui donne finalement, dans le monde "réel", l'expression de la température en fonction de t et de r :

$$T(t,r) = AI(t,r) + BK(t,r) + \frac{T_0}{\rho C_p} \tag{106}$$

A.3.3 Méthode de la séparation des variables - 2

Trouvé sur le site Wikiversity [**?**] par Ch. Trophime et à defaut d'avoir d'autres méthodes qui aboutissent, on essaie celle-ci : elle repose sur la méthode de séparation des variables, mais change quelque peu par la suite.

Cas général On pose u comme étant :

$$u(r,\theta,t) = R(r)\Theta(\theta)T(t) \tag{107}$$

En l'injectant dans l'équation,

$$\frac{\partial u}{\partial t} = k\Delta u \tag{108}$$

on a (en coordonnées cylindriques):

$$R\Theta T' = k\left(\frac{1}{r}(R'\Theta T + rR"\Theta T) + \frac{1}{r^2}R\Theta"T\right)$$
$$\iff \frac{T'}{kT} = \frac{1}{r}\left(\frac{R'}{R} + r\frac{R"}{R}\right) + \frac{1}{r^2}\frac{\Theta"}{\Theta}$$

On obtient alors 3 eq. diff. pour R, Θ, T :

$$T' + k\lambda^2 T = 0, \quad \lambda^2 = -\frac{1}{r}\left(\frac{R'}{R} + r\frac{R"}{R}\right) + \frac{1}{r^2}\frac{\Theta"}{\Theta}$$
$$\Theta" + \mu^2\Theta = 0, \quad \mu^2 = r\left(\frac{R'}{R} + r\frac{R"}{R}\right) + \lambda^2 r^2$$
$$r^2R" + rR' + (\lambda^2r^2 - \mu^2)R = 0$$

Cas axisymmétrique Dans le cas axisymmétrique, $\Theta" = \Theta' = 0$, ce qui implique que $\mu = 0$. Θ est donc une constante et disparaît alors des equations. Les relations qui restent sont :

$$T' + k\lambda^2 T = 0, \quad \text{d'où } T(t) = Ce^{-k\lambda^2 t}$$
$$r^2R" + rR' + \lambda^2r^2R = 0, \quad \text{d'où } R(r) = AJ_0(\lambda r) + BY_0(\lambda r)$$

Conditions de Robin et conditions initiales Supposons que nous avons dès le début des conditions de Robin homogènes :

$$k\frac{\partial u}{\partial n} = h_1 u(r,t) \quad \text{en } r = r_1 \tag{109}$$

$$k\frac{\partial u}{\partial n} = h_2 u(r,t) \quad \text{en } r = r_2 \tag{110}$$

Les conditions initiales sont telles que :

$$u(r,0) = f(r) \tag{111}$$

et en particulier

$$u(r_1,0) = f(r_1) \quad \text{et } u(r_2,0) = f(r_2)$$

Constantes A, B et C L'écriture de ces conditions en r_1 et r_2 donne :

$$\begin{cases} kR'(r_1)T(t) - h_1R(r_1)T(t) = 0 \\ -kR'(r_2)T(t) - h_1R(r_2)T(t) = 0 \end{cases} \tag{112}$$

D'où après réorganisation :

$$\begin{cases} A\left(k\lambda J_0'(\lambda r_1) - h_1J_0(\lambda r_1)\right) + B\left(k\lambda Y_0'(\lambda r_1) - h_1Y_0(\lambda r_1)\right) = 0 \\ A\left(k\lambda J_0'(\lambda r_2) + h_2J_0(\lambda r_2)\right) + B\left(k\lambda Y_0'(\lambda r_2) + h_2Y_0(\lambda r_2)\right) = 0 \end{cases} \tag{113}$$

Ce système et ses solutions dépendent de la valeur de λ; appelons le $S(\lambda)$, alors :

$\det(S(\lambda)) \neq 0$ il y a une unique solution : $(A, B) = (0, 0)$

$\det(S(\lambda)) = 0$ il y a une infinité de solution : $A = cste.B$

Le déterminant du système s'écrit :

$$\det(S(\lambda)) = (k\lambda J_0'(\lambda r_1) - h_1J_0(\lambda r_1)) * (k\lambda Y_0'(\lambda r_2) + h_2Y_0(\lambda r_2)) - (k\lambda Y_0'(\lambda r_1) - h_1Y_0(\lambda r_1)) * (k\lambda J_0'(\lambda r_2) + h_2J_0(\lambda r_2)) \tag{114}$$

Graphiquement, les variations de $\det(S(\lambda))$ sont :

images/determinant.png

Ainsi pour chaque λ_m tel que $\det(S(\lambda_m)) = 0$, on a un couple A_m, B_m associé; on a donc une famille de solution pour l'équation de R. De la même façon, il y a une famille de solutions pour l'équation de T :

$$T' + k\lambda_m^2 T = 0, \quad \text{d'où } T(t) = C_m e^{-k\lambda_m^2 t}$$

On a alors que :

$$u(r,t) = \sum_{m=0}^{\infty} R_m(r)T_m(t) \tag{115}$$

Pause Vu les variations du déterminant (voir graphe), on peut dire qu'il y a une infinité de λ pour lesquels le système est non-inversible. La question de comment tous les trouver est importante.

D'autre part, pour chaque λ_m, on aura : $A_m = g(\lambda_m)B_m$; et il faudra une équation pour expliciter B_m. On écrit alors :

$$\begin{aligned} u(r_1, 0) = f(r_1) &= \sum_{m=0}^{\infty} T_m(0)R_m(r_1) \\ &= \sum_{m=0}^{\infty} C_m\left(A_mJ_0(\lambda_m r_1) + B_mY_0(\lambda_m r_1)\right) \\ &= \sum_{m=0}^{\infty} C_m\left(B_m g(\lambda_m)J_0(\lambda_m r_1) + B_mY_0(\lambda_m r_1)\right) \\ &= \sum_{m=0}^{\infty} C_mB_m\left(g(\lambda_m)J_0(\lambda_m r_1) + Y_0(\lambda_m r_1)\right) \\ &...? \end{aligned} \tag{116}$$

Nous nous retrouvons de nouveau bloqué à ce fameux calcul infini de racines...

A.3.4 Conclusion sur l'instationnaire

Nous avons tenté, par diverses méthodes, de trouver la solution analytique de l'équation de la chaleur; nous n'avons pas réussi mais sommes au moins allés le plus loin possible. Ces méthodes de résolution peuvent aussi servir à trouver la solution d'autres équations.

B Appendix B : Etude de convergence du schéma BDF

B.1 Formules

La formule utilisée est celle de V. Chabannes (p. 115 de sa thèse), la voici :

$$E_{rr} = \left(\Delta t \sum_{t=t_i}^{t_f} \|u - u^n\|^2_{L^2(\Omega^{t_n})} \right)^{\frac{1}{2}} \tag{117}$$

B.2 Implémentation

B.2.1 du BDF

Rapide description de la façon dont BDF a été implémenté :

```
M_bdf->initialize(T);
for(M_bdf->start(); M_bdf->isFinished()==false; M_bdf->next() )
{

  //calcul de T
    //terme stationnaire
    form1...
    form2...

    termes instationnaire
    auto bdf_poly = M_bdf->polyDeriv();
    if( !steady )
    { form1(Th,FT) += integrate(_range=elements(mesh),
           _expr=cst(rcp)*Px()*idv(bdf_poly)*id(phi_T));
      form2(Th,Th,MatrixT) += integrate(_range=elements(mesh),
           _expr=cst(M_bdf->polyDerivCoefficient(0))*cst(rcp)*Px()*idt(T)*id(phi_T))
               ;
    }

  //calcul erreur spatial
  //ajout a l'erreur en temps

  M_bdf->shiftRight( T );
}

//finalisation calcul d'erreur en temps
```

B.2.2 du calcul d'erreur

Cette erreur en temps se base sur l'erreur spatiale (L^2) qui est déjà implémenté, et dont la convergence a été vérifiée en ordre polynomiale 1,2 et 3.

On déclare une variable *double*, initialiée à 0.0. Comme l'erreur spatiale est calculée à chaque itération en temps, il ne reste qu'à faire :

```
double Time_error = 0.0;

for( time loop )
{
  //calcul...

  //calcul erreur L2 spatial
```

```
  Time_error = Time_error + L2_spatial*L2_spatial;
}

Time_error = math::sqrt(timeStep*Time_error);
```

Y a-t-il quelque chose qui pourrait clocher ?

B.3 Résultats

B.3.1 Résultats généraux

Avec une méthode BDF d'ordre 1, on devra obtenir une pente d'erreur égale à 1; respectivement de même pour l'ordre 2, 3 et 4. Ici :

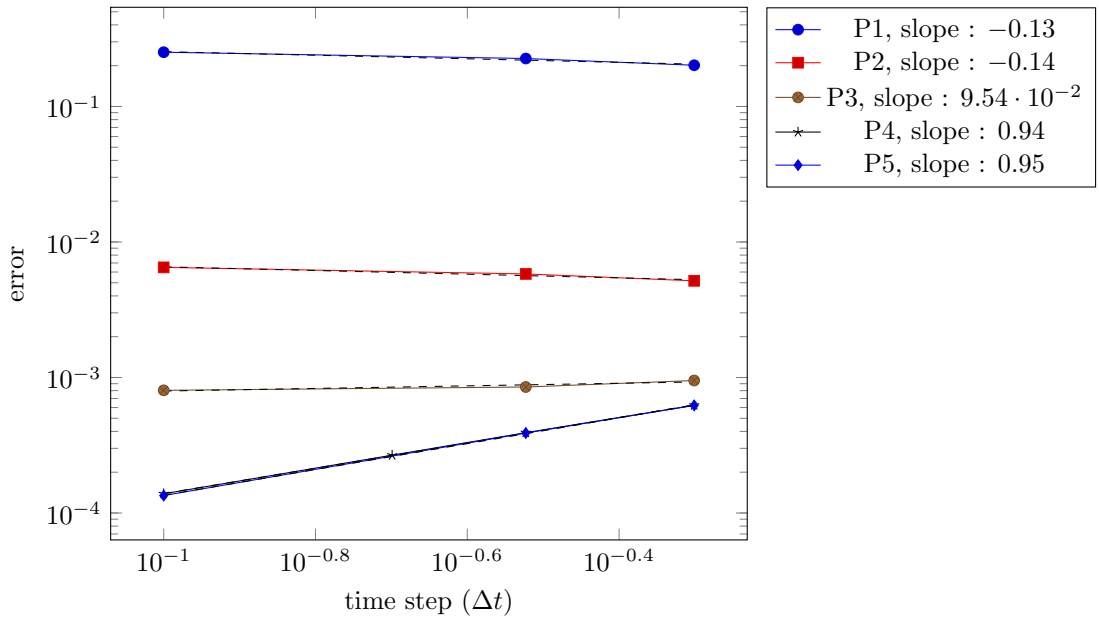
- temps initial 0
- temps final 3
- pas de temps : 0.1, 0.2, 0.3 et 0.5
- pas en espace 0.03
- géométrie : test_heataxi_cv_study
- ordre polynomial et ordre BDF précisé plus loin
- solution exacte $20\sin(\Pi(x-1))\sin(\Pi\frac{y-0.05}{0.1})e^{-t}$

BDF 1 Voici les valeurs brutes :

timeStep	P1 Error	P2 Error	P3 Error	P4 Error	P5 Error
0.1	0.25146	0.00649869	0.000803814	0.00013859	0.000134528
0.2	-			0.000267089	-
0.3	0.225859	0.00580994	0.000850639	0.000391459	0.000389102
0.5	0.201454	0.00516786	0.000949825	0.000623876	0.000622006

Visualisation graphique :

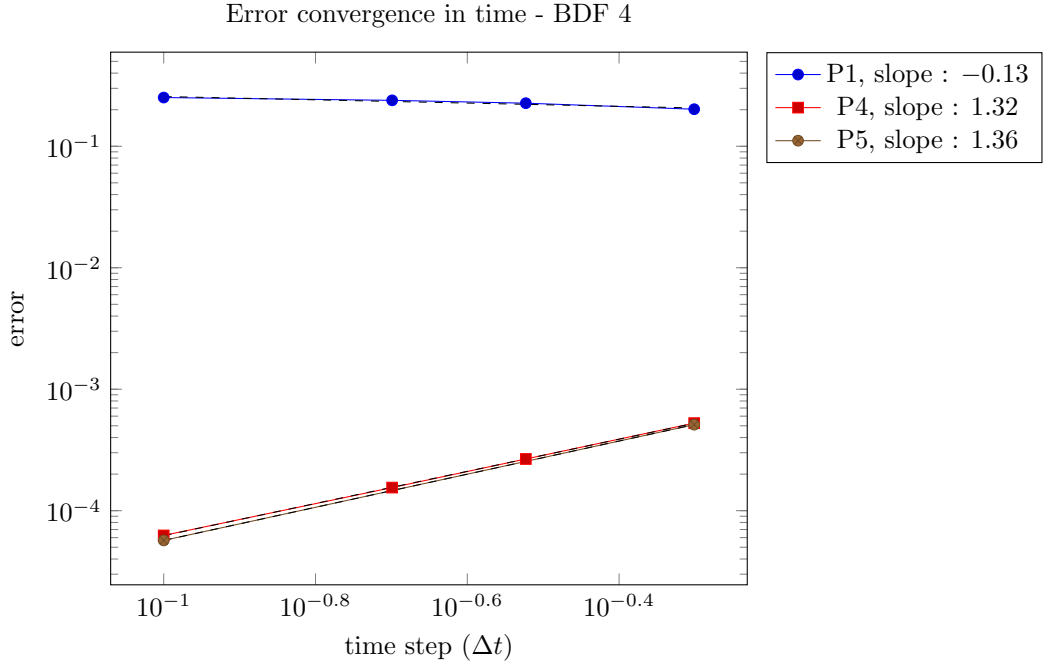
Error convergence in time - BDF 1



BDF 4 Voici les valeurs brutes des erreurs :

timeStep	P1 Error	P2 Error	P3 Error	P4 Error	P5 Error
0.1	0.251836	-	-	6.25506e-05	5.69126e-05
0.2	0.238819	-	-	0.000154966	
0.3	0.226295	-	-	0.000266456	
0.5	0.201896	-	-	0.000525685	0.000510724

Visualisation graphique :



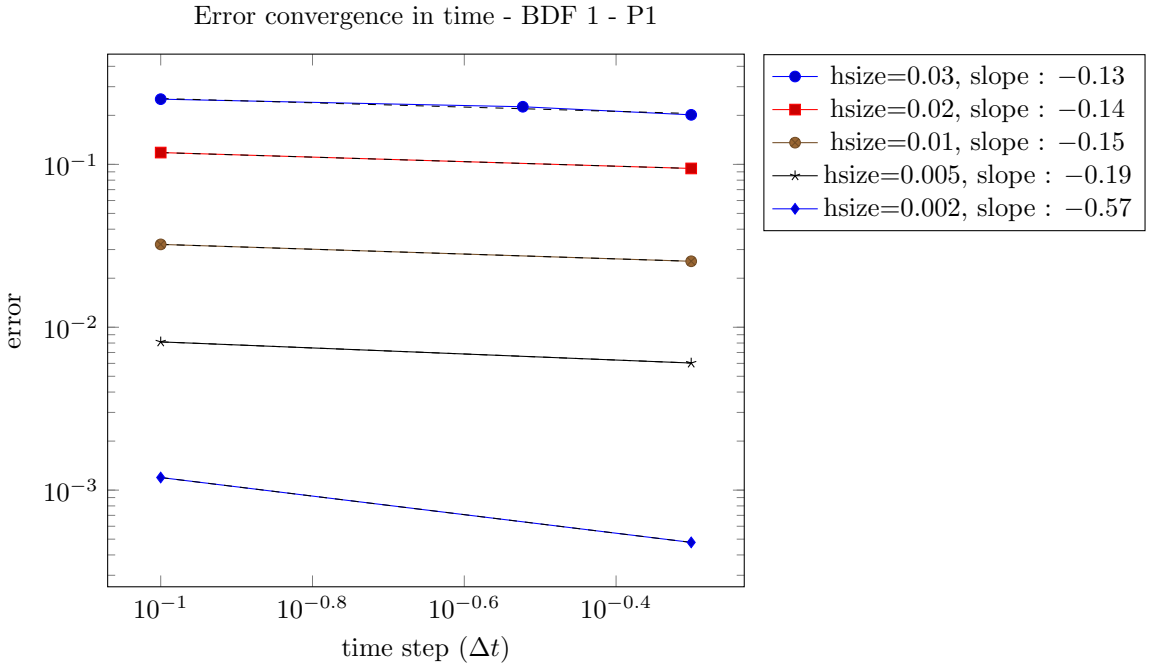
Conclusion En augmentant l'ordre polynomial, l'erreur en temps, même si elle ne respecte pas encore la théorie, a une allure plus logique : elle diminue quand Δt diminue. Il faut donc minimiser davantage l'erreur spatiale et avoir une convergence de l'erreur temporelle également en P1 : pour le cas du BDF1, en P4 et en P5 les résultats sont corrects (pente de 0.94), mais pas en P1, P2 et P3. Il faut donc trouver le pas d'espace adapté pour la convergence en P1 et en BDF1.

B.3.2 Etude en fonction du hsize - 1

On garde la même configuration mais on change le pas en espace. Il s'agit de trouver le pas en espace qui, en P1 et avec BDF1, permet d'avoir une pente d'erreur égale à 1.

Voici les valeurs brutes (P1 et BDF1) :

timeStep	h 0.03	h 0.02	h 0.01	h 0.005	h 0.002
0.1	0.25146	0.118191	0.0322695	0.0081306	0.00119499
0.2	-	-	-	-	-
0.3	-	-	-	-	-
0.5	0.201454	0.0944229	0.0254161	0.00603001	0.000477614



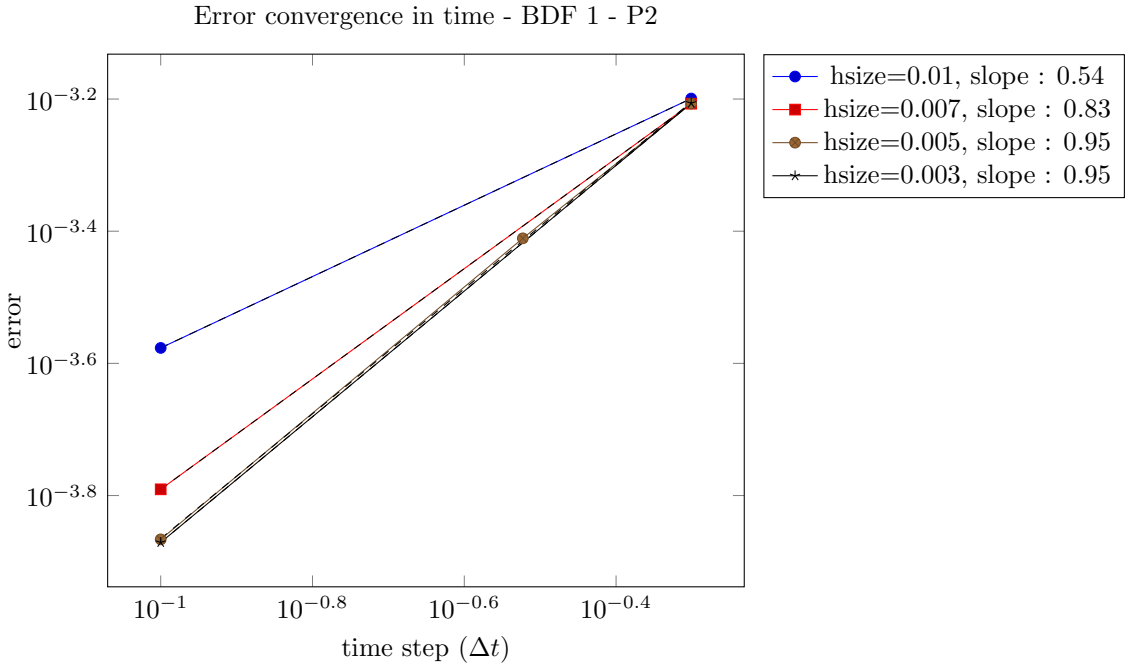
Conclusion Le hsize reste bien caché...

B.3.3 Etude en fonction du hsize - 2

On essaie pour P2.

Voici les valeurs brutes (P2 et BDF1) :

timeStep	h 0.01	h 0.007	h 0.005	h 0.003
0.1	0.000265112	0.000161998	0.000136074	0.000134628
0.2	-	-	-	-
0.3	-	-	0.000388494	-
0.5	0.000632114	0.000620869	0.000621119	-



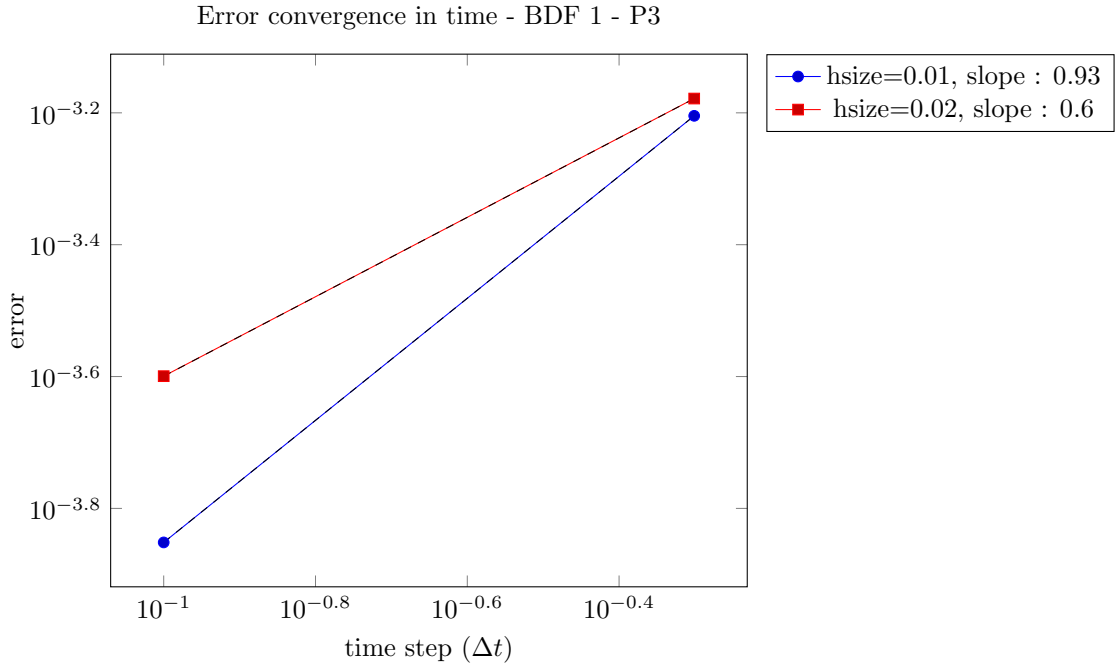
Conclusion Pour P2 et BDF1 et hsize fixé à 0.005, nous avons une convergence de l'erreur en temps. Naturellement, nous pouvons supposer que cette convergence se fait aussi en P3 et au-delà (vérifié plus bas).

B.3.4 Etude en fonction du hsize - 3

On vérifie la convergence en P3 et en BDF1. Sachant que pour un pas de maille de 0.005 on a la convergence en P2-BDF1, on peut faire les tests pour P3-BDF1 en l'augmentant à 0.01 :

Voici les valeurs brutes (P3 et BDF1) :

timeStep	h 0.01	h 0.02
0.1	0.00014076	-
0.2	-	-
0.3	-	-
0.5	0.000624473	-



Conclusion On a vu juste, la convergence est bel et bien atteinte pour un pas de maille de 0.01.

B.4 Equation parabolique

B.4.1 Convergence temporelle en P1

Comme on n'obtient toujours pas de convergence en P1 (BDF1 et au-delà), on procède à la validation de l'implémentation du BDF et du calcul de l'erreur temporelle par un autre moyen, par une autre équation, l'équation parabolique :

$$\frac{\partial u}{\partial t} - \Delta u = f \tag{118}$$

On part directement de l'exemple du laplacian du doc/manual/laplacian/laplacian.cpp, on implémente le BDF et le calcul d'erreur en temps, et quelques modifications pour prendre en compte le paramètre t (comme les modifs de CT pour HeatAxi). Attention, on n'est plus en coordonnées cylindriques, mais purement et simplement cartésiennes !

Remarque : le calcul d'erreur est similaire à celui fait par V. Chabannes. L'implémentation du BDF a été comparée avec HeatSink et NSProj du doc/manual.

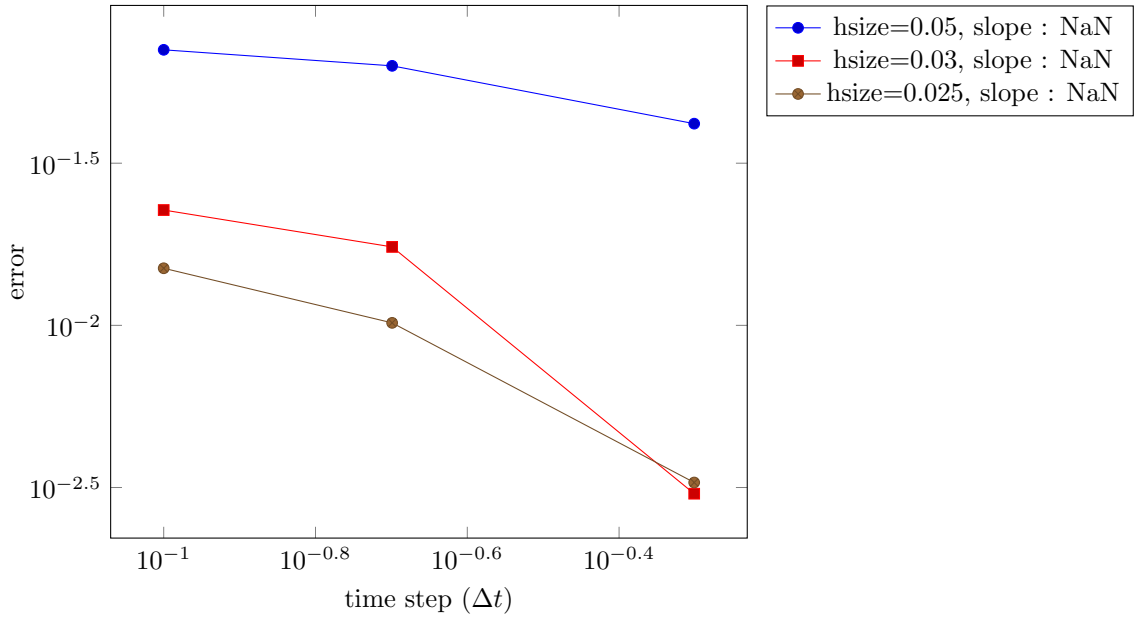
Options On prend la solution exacte : $\sin(\pi(x-1))\sin(\pi\frac{y-0.05}{0.1})e^{-t}$. Le rhs est calculé par le programme.

Voici alors les valeurs brutes pour P1-BDF1:

D'abord pour les pas de mailles pour lesquels on obtient pas de convergence :

timeStep	TimeError0.05	TimeError0.03	TimeError0.025
0.5	0.0418714	0.00302629	0.0032765
0.2	0.0631308	0.0174553	0.0101781
0.1	0.0707328	0.0226771	0.0149995

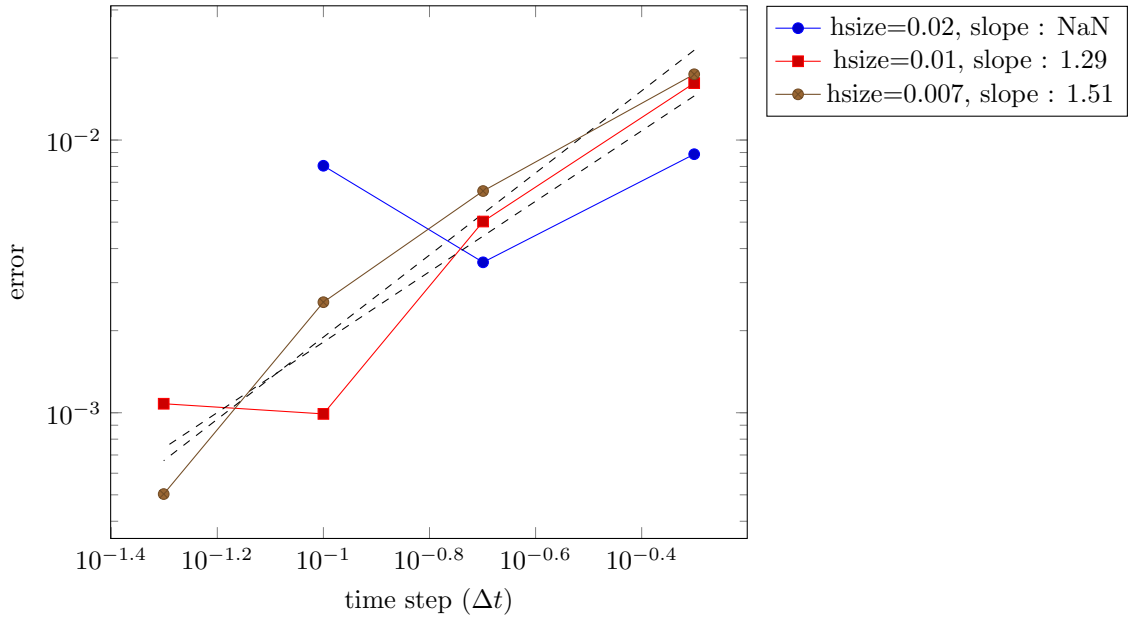
Parabolic equation - Error convergence in time - BDF 1 - P1



Et maintenant pour les pas de mailles où on obtient une pente positive :

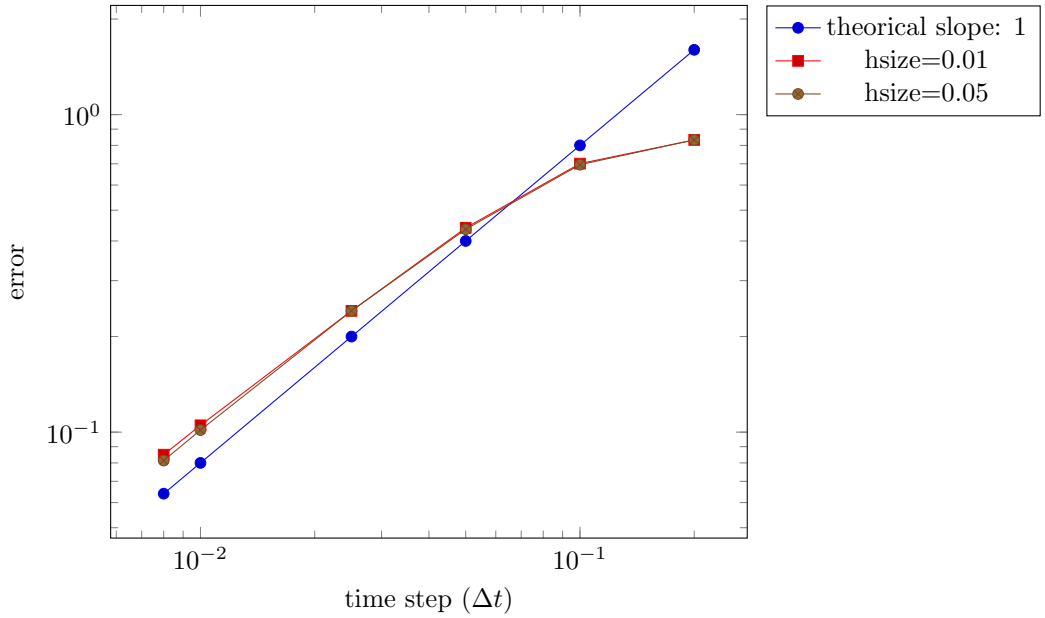
timeStep	TimeError0.02	TimeError0.01	TimeError0.007
0.5	0.00887368	0.0161717	0.0174287
0.2	0.00356165	0.00502612	0.00650241
0.1	0.00804849	0.000990303	0.00254321

Parabolic equation - Error convergence in time - BDF 1 - P1



Cela paraît anarchique et on ne comprend pas ce qu'il se passe : tantôt l'erreur baisse, tantôt elle augmente et on ne sait pas très bien pourquoi. On continue alors les tests on diminuant au fur et à mesure le pas de temps, et voici ce qu'on obtient :

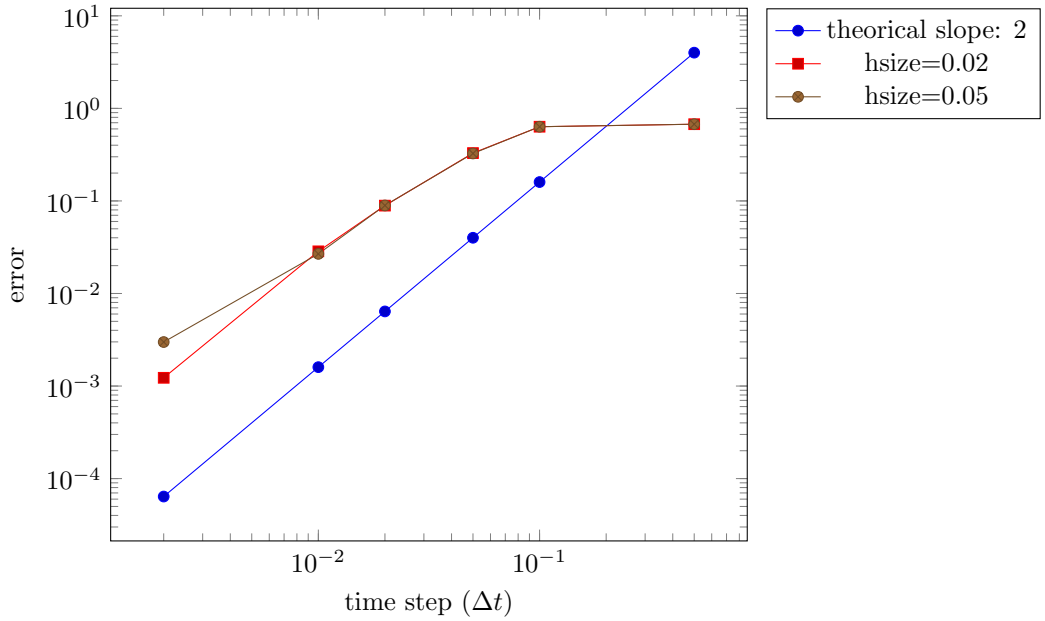
Parabolic equation - Error convergence in time - BDF 1 - P1



Conclusion Si on ne prend pas en compte l'erreur au pas de temps 0.1 et 0.2, on a une pente de 0.9, qui est tout à fait convenable. Dans les graphes précédents, l'erreur temporelle ne pouvait pas converger, bien que le pas d'espace fût très petit (0.007), car le pas de temps était trop élevé. En en prenant des plus petits, on a pu avoir cette convergence.

Test pour BDF2 On teste alors les pentes en BDF2 :

Parabolic equation - Error convergence in time - BDF 2 - P1

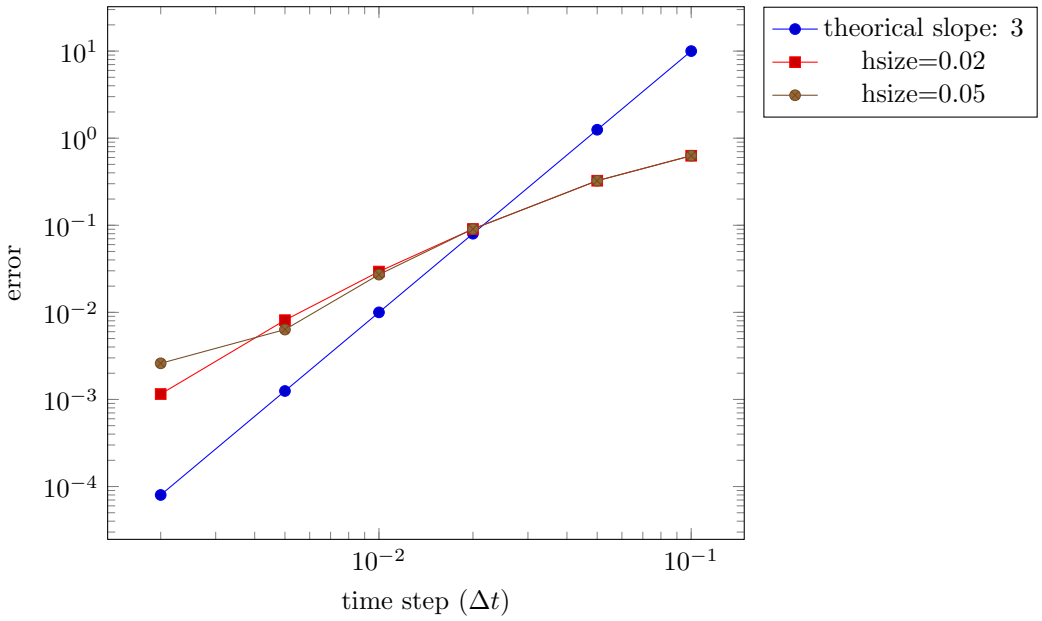


Conclusion Comme précédemment, pour des pas de temps pas assez petits, l'erreur ne diminue comme elle le devrait; en le diminuant, elle se rapproche de plus en plus de la pente théorique. Au dernier pas de temps pris (dt = 0.002), il y a une nette différence entre celle obtenue avec un h=0.05

et un h=0.02 : pour le premier cas, la courbe de convergence s'écarte de l'allure qu'elle avait prise, alors que pour le deuxième cas, elle continue d'évoluer comme la pente théorique. Donc avec un tel dt, un h=0.05 ne convient pas car cela induit une erreur spatiale prépondérante.

Test pour BDF3 On valide ensuite l'ordre BDF suivant :

Parabolic equation - Error convergence in time - BDF 3 - P1



Conclusion Similairement au BDF d'ordre 2, il y a une nette différence entre la courbe de h=0.05 et h=0.02. Il semble que descendre à 0.02 ne suffise pas, il faudrait descendre à 0.01 pour diminuer davantage l'erreur spatiale (qui, actuellement, empêche de voir l'erreur de la discrétisation différences finies).

B.4.2 Conclusion sur l'étude de l'eq. parabolique

Ce test effectué avec l'équation parabolique a permis de valider l'implémentation du BDF ainsi que du calcul de l'erreur en temps et de voir pourquoi on n'avait pas la convergence de HeatAxi (du moins en P1) : "simplement" parce que les Δt fixés étaient trop grands; on peut alors refaire les tests sans hésiter à diminuer radicalement h et Δt.

B.5 Résultats - 2

Ayant eu la confirmation que BDF est bien implémenté ainsi que le calcul de l'erreur en temps, on refait l'étude du BDF avec la géométrie test (test_heataxi_cv_study), avec la solution exacte :

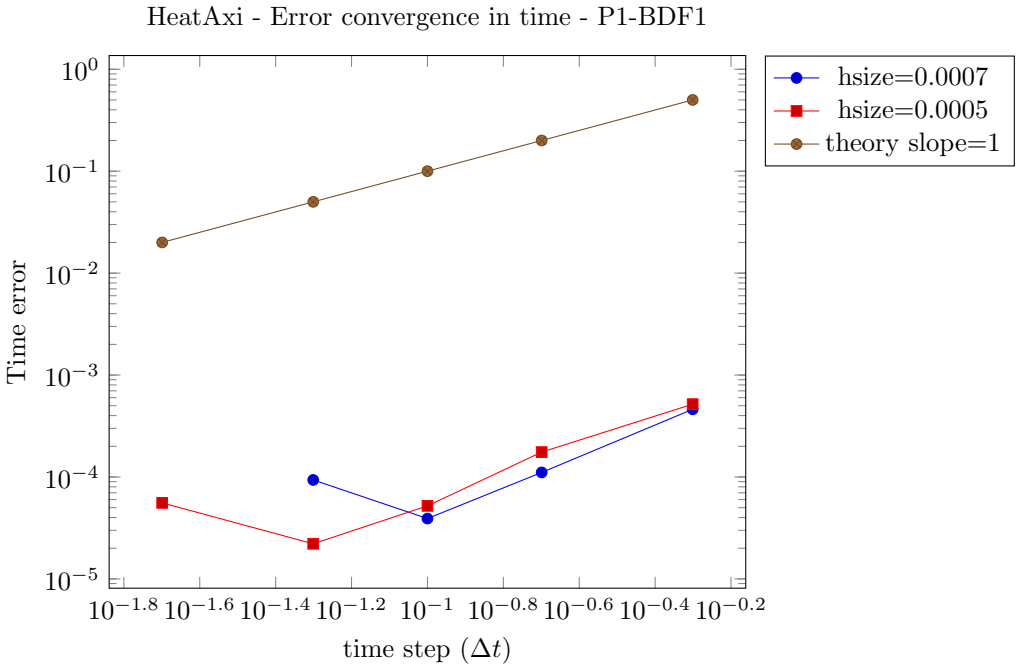
$$20\sin(\pi(x-1))\sin(\pi(y-0.05)/0.1)e^{-t}$$

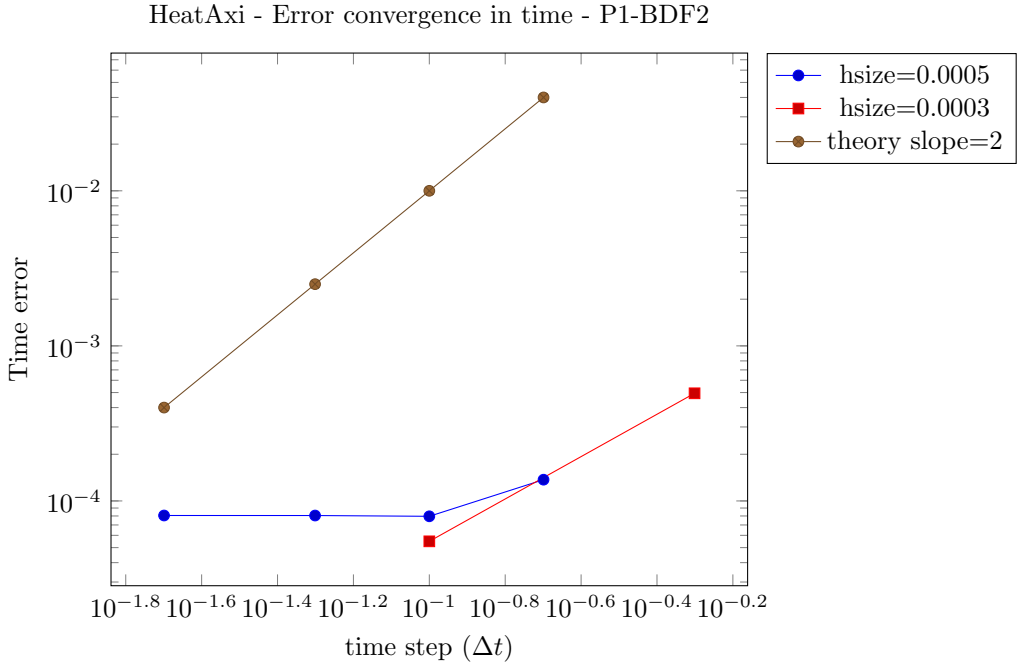
Pour rappel, le calcul de l'erreur en temps se fait avec la formule :

$$E_{rr} = \left(\Delta t \sum_{t=t_i}^{t_f} \|u - u^n\|^2_{L^2(\Omega^{t_n})}\right)^{\frac{1}{2}}$$

Voici les graphes :

B.5.1 Tests en P1



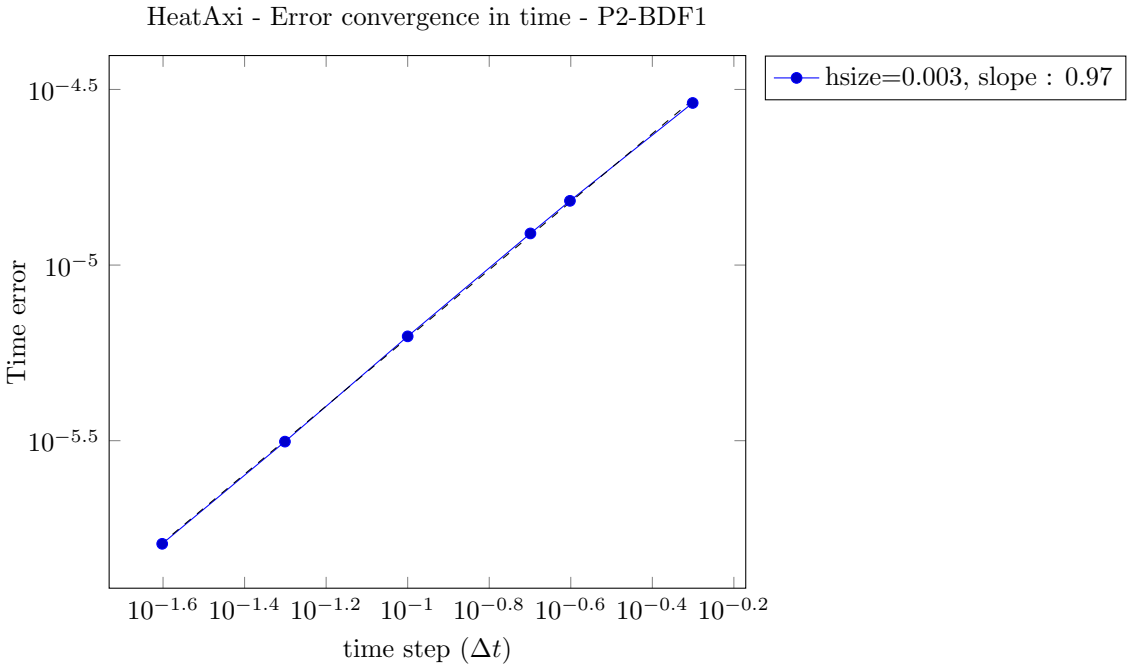


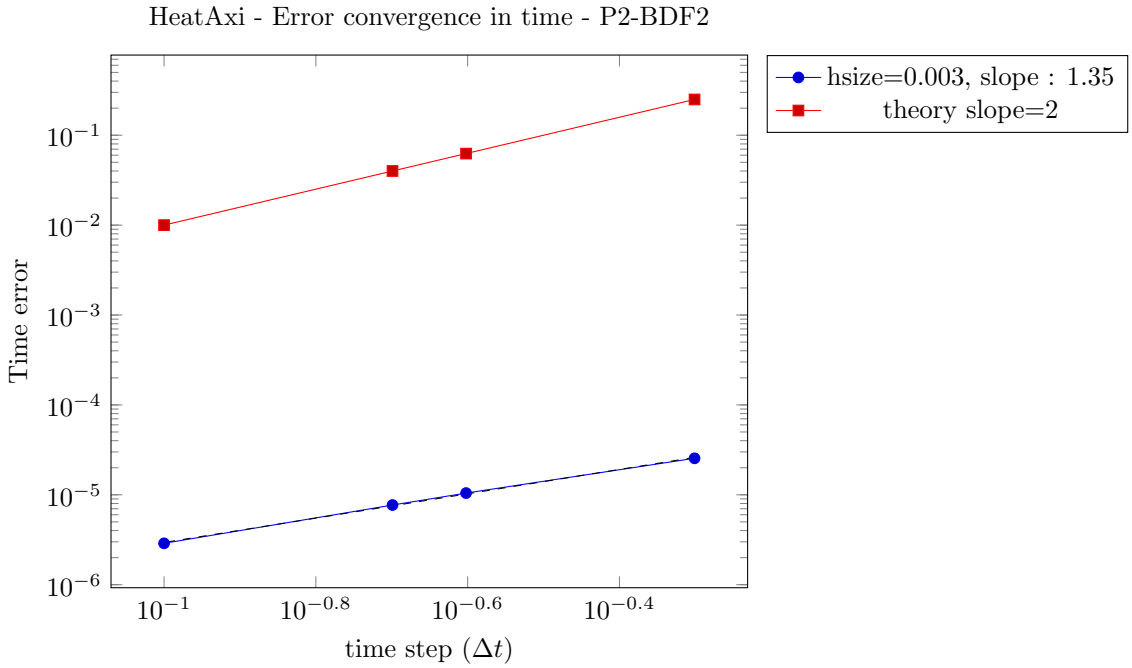
Conclusion : Excepté pour le dernier Δt en P1-BDF1, les erreurs en temps diminuent quand le pas de temps diminue. Les pentes ne sont pas optimales et ce pour la *simple* raison qu'en espace, le maillage (même avec un hsize de 0.0005 et près d'un million de degrés de liberté) fausse le calcul de cette erreur en temps.

Ceci est notamment remarquable dans le deuxième graphe (P1-BDF2) : pour h=0.0005 l'erreur stagne, alors que pour h=0.0003 elle continue de descendre.

Remarque : Il est difficile de diminuer davantage la taille de la maille; Gmsh refuse parfois de créer le fichier .msh car pour lui : *no elements in the mesh.*

B.5.2 Tests en P2

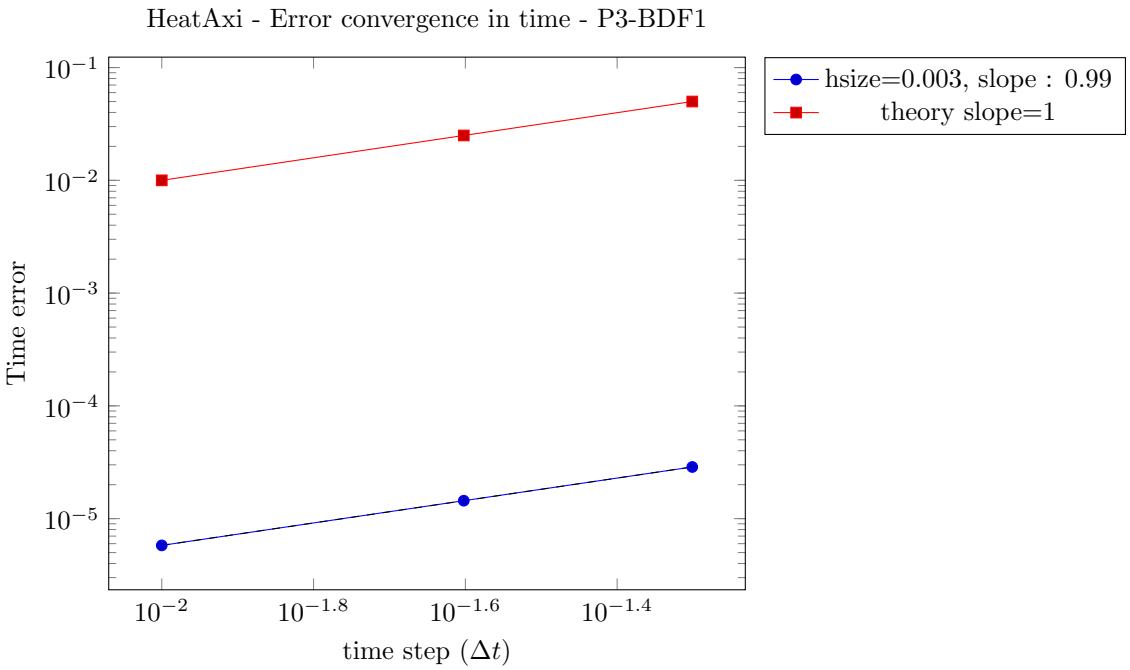


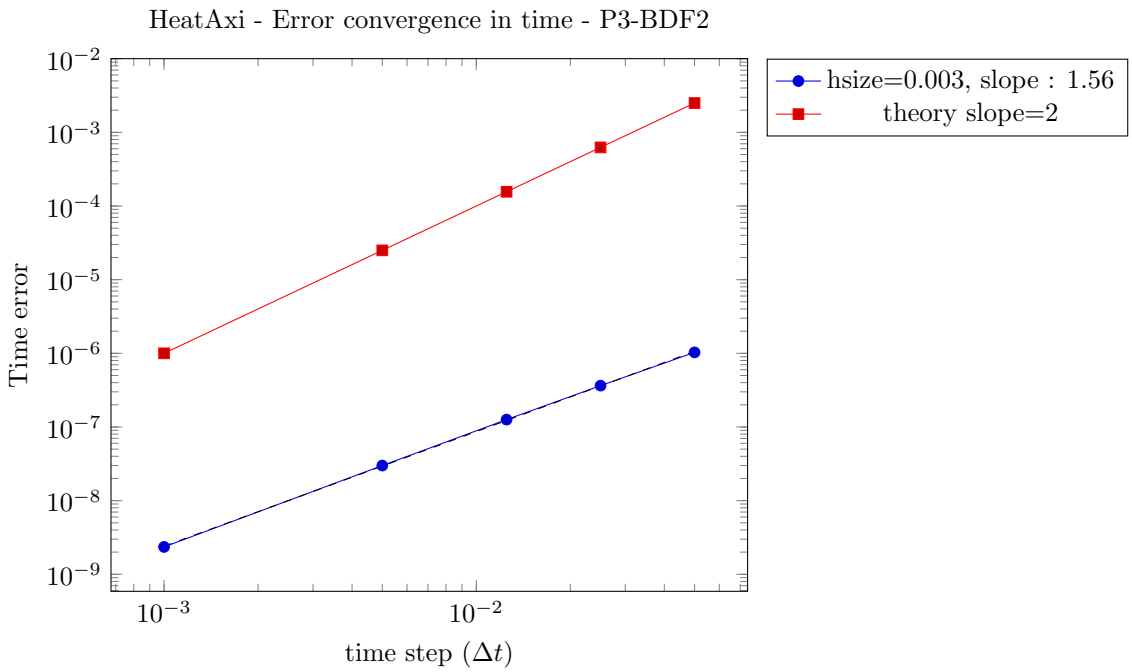


Conclusion : La convergence P2-BDF1 était déjà établi (voir §3.3); pour P2-BDF2, on peut se réjouir d'une pente (1.35) plus grande que pour P2-BDF1, mais qui n'est pas optimale pour autant.

B.5.3 Tests en P3

Attention : ici on essaie avec un Tf=0.1. En effet, on remarque lors des tests en instationnaires, que l'erreur au premier pas de temps (à $t = \Delta t$) est la plus grande, car elle de 10 à 100 fois plus grandes que l'erreur aux autres temps. On aimerait alors réduire sa contribution à l'erreur finale.

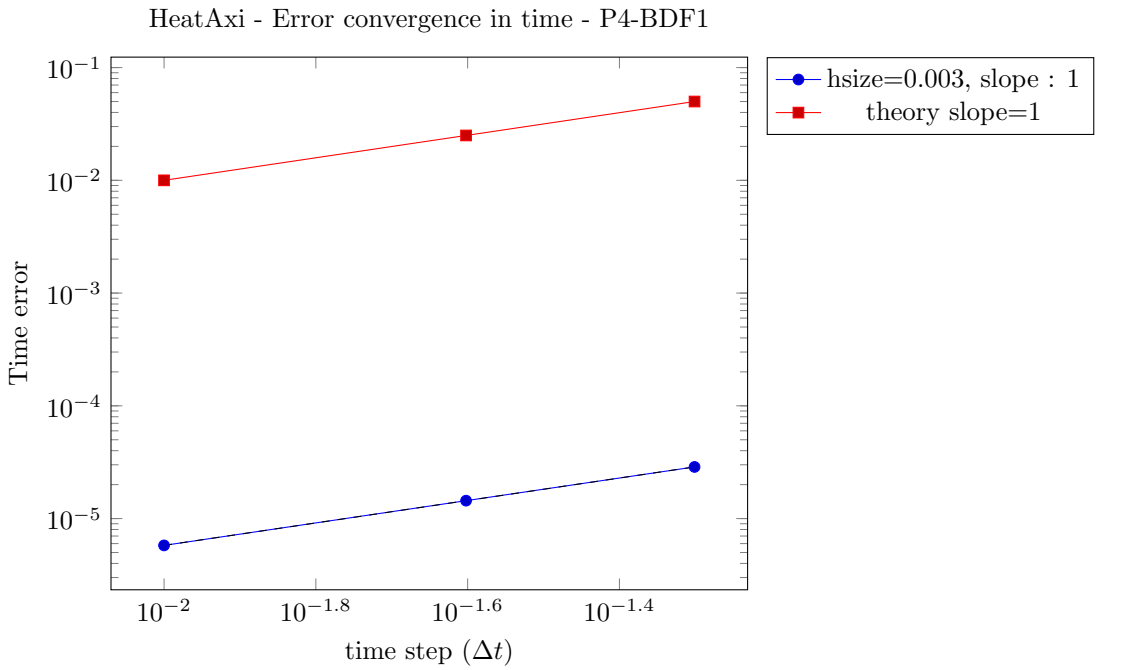


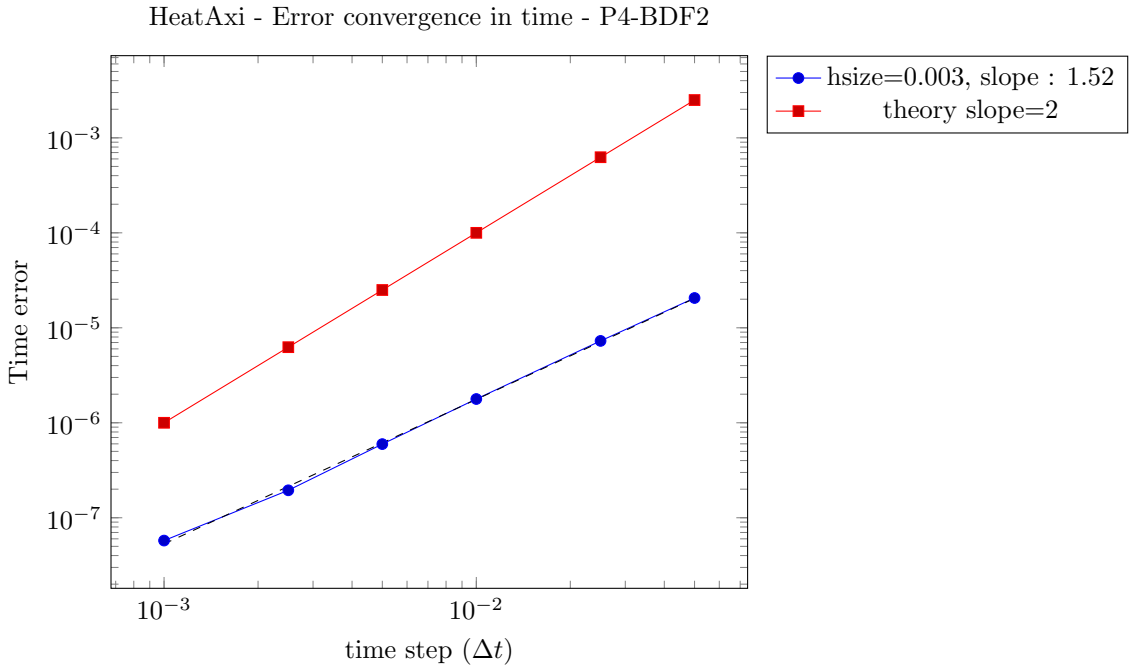


Conclusion : Entre P2-BDF2 et P3-BDF2, il y a une nette amélioration de la pente : elle s'accentue en passant de 1.35 à 1.56. On peut supposer (les tests suivants doivent pouvoir le montrer) que la pente se rapprochera de 2 pour les test pour P4-BDF2 et P5-BDF2.

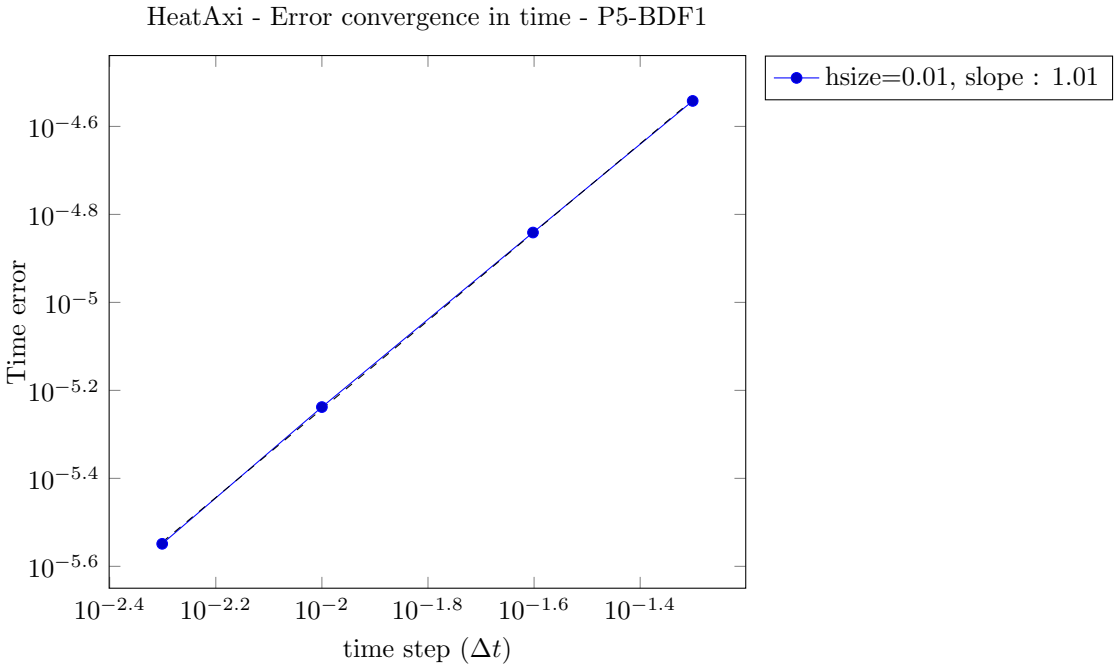
B.5.4 Tests en P4

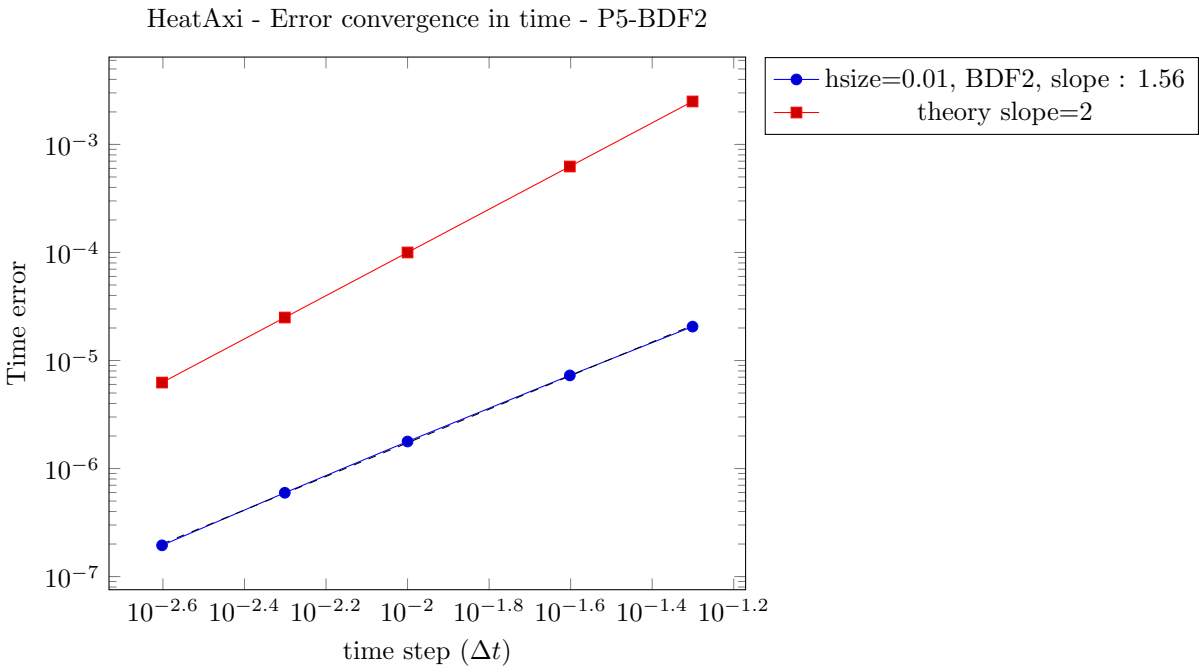
On continue les tests en prenant un $T_f = 0.1$.


HeatAxi - Error convergence in time - P4-BDF1
Time error
time step (Δt)
hsize=0.003, slope : 1
theory slope=1


HeatAxi - Error convergence in time - P4-BDF2
Time error
time step (Δt)
hsize=0.003, slope : 1.52
theory slope=2

B.5.5 Tests en P5



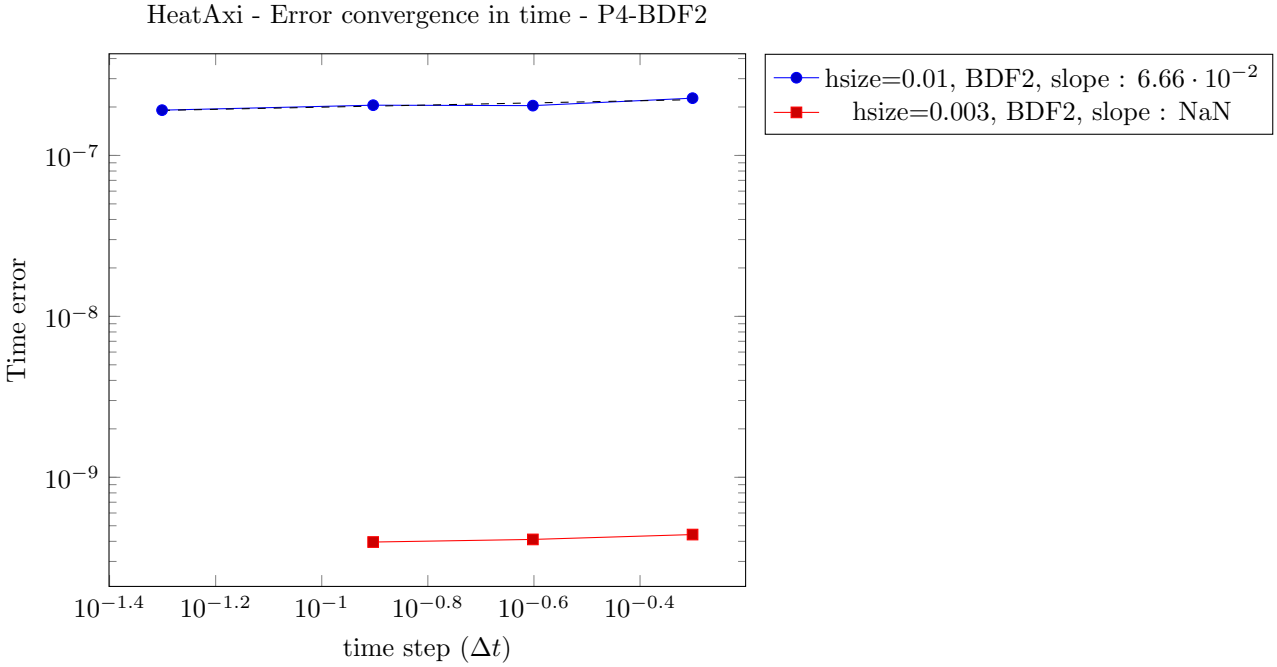


Conclusion Avoir pris $T_f = 0.1$ n'a pas servi à avoir une meilleure convergence. D'autres tests ont été fait en prenant une expression en t différente : $\exp(-3t)$, $\exp(-0.1t)$, $\sin(t)$, et en essayant de prendre un T_f plus petit, comme 0.01 ou 0.001.

Ces tests ont été fait en P4 et P5, car on voulait réduire au maximum l'erreur en espace et montrer la convergence du BDF avec ces ordres polynomiaux au moins : mais que la taille de maille soit 0.01 ou 0.003 (en P4 cela passe de 20 000 degrés de liberté à 210 000), l'erreur en temps ne change pas. Cela voudrait dire que cette erreur est exclusivement dûe au schéma différences finies et non à la MEF. Qu'elle en est la raison ? Cela reste à établir.

Solution ? Après en avoir discuté avec CP, il a proposé de prendre une expression en t qui varierai linéairement : car on obtiendrait normalement une solution exacte.

On prend alors l'expression en t suivante : $t + 1$, avec un $T_f = 1$, et $h = 0.01$ et de $h = 0.003$.



L'erreur en temps n'est pas grandement affectée par les différents Δt; pour $h = 0.003$ les erreurs en temps sont plus petites, mais constantes. On peut alors conclure que ce qu'on observe n'est que l'erreur en espace, et qu'en temps, le polynôme de degré 1 est parfaitement approximé par le schéma des différences finies.

B.6 Conclusion de l'étude BDF

Cette étude de convergence en temps a été longue et fastidieuse, mais très formatteur. Elle a permis de comprendre les limites des calculs sur plusieurs plans :

Sur le plan numérique quand il y a beaucoup de calculs à faire, la somme de toutes les erreurs numériques peut rendre bancal une étude, qui en théorie devrait marcher. La précision machine, qui est aux alentours de 10^{-15}, se ressent donc bien avant : on le voit sur les courbes de l'étude stationnaire (en P4 et P5), elles stagnent à 10^{-10}. Ainsi, que les ordinateurs ne font que des calculs approchés est un fait parfaitement connu, mais il faut savoir déterminer à partir de quel *moment* on peut lui reprocher quelque chose.

Sur le plan modélisation en effet, certaines limites ne sont pas dûes aux machines, mais bien à l'utilisateur qui fixe les paramètres. Les pentes théoriques de convergence, que ce soit en stationnaire ou non, sont des pentes optimales : on ne peut pas espérer faire mieux. Les atteindre n'est pas chose aisée pour des raisons de coûts : si on veut une grande précision, il faut être prêt à en payer le prix par une plus grande consommation de ressources et de temps.

Après ces étapes de vérification et de validation, on peut parfaitement déclarer que l'implémentation des méthodes élements finis et différences finis est juste ainsi que le code sur les calculs des erreurs (exactes et estimées).