Ecole Polytechnique Fédérale de Lausanne

CSE semester project

Defining models with differential operators in Akantu

Student : Schiesser Thibault

Supervisor : Pr. Anciaux Guillaume

Computational Solid Mechanics Laboratory LSMS

07.06.2024

Contents

1 Introduction

1.1 Objectives

The aim of the project is to extend Akantu's functionalities so that the program can solve equations written in their weak form, i.e. using differential operators. The idea is to expand Akantu's ability to solve a wide range of physics problems, such as fluid mechanics, electromagnetism and heat equations. In addition to providing a wide range of applications, this functionality offers intuitive resolution in the sense that the user inputs the mathematical equation into the program almost as it is.
To achieve these, the differential operators in the weak form of the equations have to be implemented. Specifically, this project focuses on solving a case of the Navier equation, which constitutes an introduction, then on the heat equation through the implementation of a gradient operator. A rotational operator which serves as a demonstrator is also implemented.
With object-oriented programming, the code is designed to provide the necessary structure for easy extension of the program by other differential operators not considered in this project.

1.2 Mathematical preliminaries

This section presents an example of the transformation from the strong formulation of an equation to its weak formulation, i.e. the equation the user inputs into the program. Finally, the weak form will be identified with the finite element discretization, which is the equation solved by the program, even though it remains hidden from the user.
Let's take the example of the 2D heat equation for a homogeneous, isotropic material in an orthogonal system. The strong formulation is [1]:

$$\frac{\partial T}{\partial t} = \alpha \operatorname{div}(\nabla T) + \frac{1}{\rho\, c_P} q = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + \frac{1}{\rho\, c_P} q \tag{1}$$

- $T(x, y, t)$: the temperature,
- α : the thermal diffusivity,
- ρ : density of the material,
- c_P : specific heat capacity,
- t : the time,
- q : volumetric heat source.

q is considered to be 0.

In order to find its integral form, we begin by multiplying the equation (1) by an admissible virtual temperature field T^* such that $T^* = 0$ on $\delta\Omega$. Then the equation is integrated over the domain Ω.

$$\int_\Omega T^* \frac{\partial T}{\partial t}\, d\Omega = \alpha \int_\Omega T^* \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) d\Omega. \tag{2}$$

The right-hand side term can be rewritten by integrating it by parts as follows:

$$\int_\Omega T^* \frac{\partial^2 T}{\partial x^2}\, d\Omega = \left[T^* \frac{\partial T}{\partial x} \right]_{\partial\Omega} - \int_\Omega \frac{\partial T^*}{\partial x} \frac{\partial T}{\partial x}\, d\Omega. \tag{3}$$

The equation (3) is simplified using the fact that T^* is zero at the border of the domain, then it is injected into (2).

$$\int_\Omega T^* \frac{\partial T}{\partial t}\, d\Omega + \alpha \int_\Omega \left(\frac{\partial T^*}{\partial x} \frac{\partial T}{\partial x} + \frac{\partial T^*}{\partial y} \frac{\partial T}{\partial y} \right) d\Omega = 0 \tag{4}$$

Finally, considering no time variation, and using the gradient operator, one finds :

$$\alpha \int_{\Omega} (\nabla T^*)^T \nabla T \, d\Omega = 0. \tag{5}$$

The equation form (5) is intended to be solved by Akantu in a single line of code by writing this expression.
Actually, the finite element program solves a discrete form.
The Ω domain is meshed, and for each element (e.g. for triangle elements) the shape functions N_i are identified, such that

$$T(x, y) \approx N_1(x, y)T_1 + N_2(x, y)T_2 + N_3(x, y)T_3, \tag{6}$$

where T_i are nodal values.
The gradient $\nabla T = \left(\frac{\partial T}{\partial x} \frac{\partial T}{\partial y}\right)$ can thus be approximated as follows:

$$\begin{aligned} \frac{\partial T}{\partial x} &\approx \frac{\partial}{\partial x}(N_1T_1 + N_2T_2 + N_3T_3) \\ &= \frac{\partial N_1}{\partial x}T_1 + \frac{\partial N_2}{\partial x}T_2 + \frac{\partial N_3}{\partial x}T_3, \\ \frac{\partial T}{\partial y} &\approx \frac{\partial}{\partial y}(N_1T_1 + N_2T_2 + N_3T_3) \\ &= \frac{\partial N_1}{\partial y}T_1 + \frac{\partial N_2}{\partial y}T_2 + \frac{\partial N_3}{\partial y}T_3. \end{aligned} \tag{7}$$

or in matrix form :

$$\begin{aligned} \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{pmatrix} &\approx \begin{pmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} \\ &= BV_T \end{aligned} \tag{8}$$

By putting (8) into (5), then taking the vector of nodal values out of the integral, the expression that is solved by the finite element program is obtained:

$$\alpha \left(\int_{\Omega} B^T B \, d\Omega \right) V_T = 0 \tag{9}$$

And is written as a linear system :

$$\alpha \, K V_T = 0 \tag{10}$$

Through the weak formulation, the program computes K. Then finally by applying boundary conditions, the heat equation can be solved.
Writing the weak formulation using finite element discretization underlines that the resolution of (5), is possible by the finite element method in the case that tensors contain mesh information, the shape functions and its derivatives. This aspect is developed in section 2.1.

2 Implementation

The Python programming language is used. The full implementation is available at Github.

2.1 Main classes

This section describes only the main classes. For a more detailed description of the implementation, the complete class diagram can be found in the Appendix 12 or in Github. In addition, to get a detailed description of the parameters and returns for each part of the code, please refer directly to the related python files, which have been commented out by Docstrings.

- **Support**: Through Akantu's FEEngine object, this class contains the mesh, the shape functions and its derivatives (6)(7). Each tensor field in the program contains a Support. This lets user write the weak form with differential operators (5), while hiding the discretization used for finite element method.

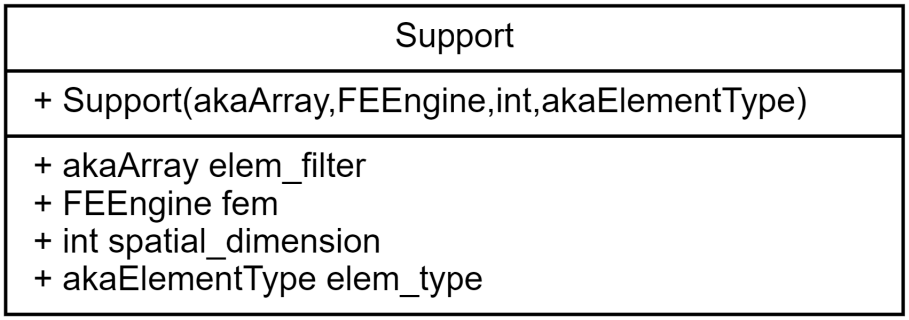

Figure 1: Support Class

- **TensorField** : This class is the parent class of all the objects that are intended to be integrated. Several operators are overloaded to provide operations between TensorField objects. The methods getFieldDimension() and evalOnQuadraturePoints() are required for integration: getFieldDimension() is used to obtain the correct array dimensions for integration, while evalOnQuadraturePoints() is used to evaluate the TensorField at the locations required for numerical Gaussian integration. These coordinates are set when the mesh is created and the element is chosen, i.e. the information is contained in the Support member.

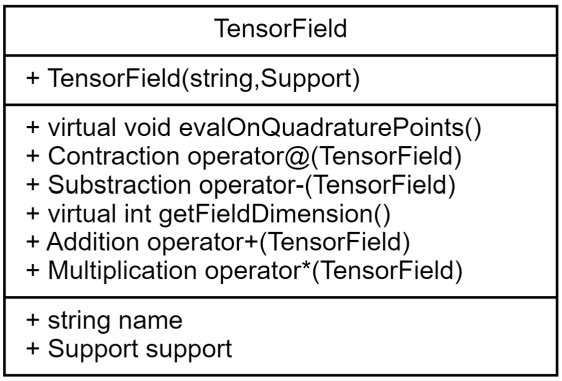

Figure 2: TensorField Class

- **Operator** : This class is derived from the TensorField class. The Operator class is the parent class of the objects returned by the overload operations for TensorField object. It is also the parent of the gradient and curl operator classes. The constructor of this class takes as parameters an arbitrary number of TensorFields.

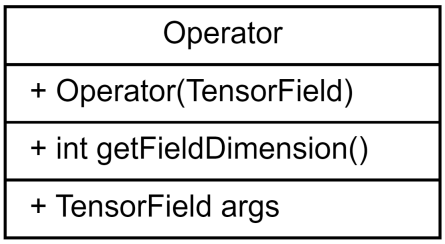

Figure 3: Operator Class

- **Integrate** and **Assembly** : The integrate method is used to integrate any TensorField object. First, it calls the TensorField's "evalOnQuadraturePoint()" method. This highlights a polymorphism. Next, it performs numerical integration using the Gauss method, element by element. Next, it is necessary to assemble the result returned by the integrator, which is structured element by element. The methods of the Assembly class provide this assembling functionality, depending on the nature of the integrated object.

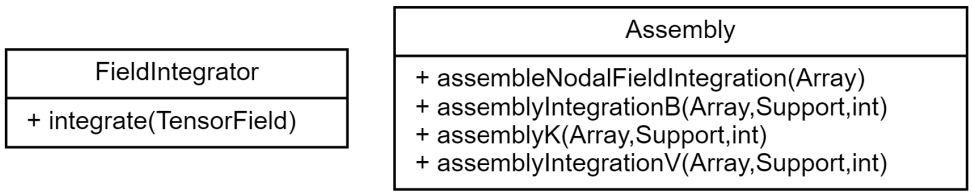

Figure 4: Integrate and Assembly Class

2.2 Algorithm Description

These are the successive steps involved in the problem resolution :

1. Mesh : Discretizing the domain and selecting the element type.
2. Support : Instantiation of the object that contains the mesh, the shape functions and its derivatives.
3. Writing the equation using differential operators.
4. Integration.
5. Assembly.
6. Problem resolution: applying the boundary conditions to solve the problem.

For each of the practical tests detailed in the next section, the code is commented using the step numbering mentioned above.

3 Practical Tests

The tests carried out for each of the equations presented below are reproducible with the files present on Github. Respectively :

- Navier : patchtest_navier.py.
- Heat equation : test_heat1D_seg2.py, test_heat1D_seg3.py, test_heat2D_t3.py.
- Curl operator : test_CurlOperator.py

3.1 Navier

3.1.1 Weak formulation

The first problem solved is the Navier problem in static, with the strong formulation [2]:

$$\begin{aligned} \Delta\sigma + b = 0, &\quad \text{on } \Omega \\ \sigma n = t, &\quad \text{on } \partial\Omega_t \end{aligned} \tag{11}$$

where,

- σ : Stress tensor,
- t : Stress at $\partial\Omega$,
- b : Volumetric force.

Considering no volumetric force, and the constitutive law for an istropic linear elastic case ($\sigma = D\varepsilon$) its weak formulation is :

$$\int_\Omega \delta\varepsilon^T D\varepsilon \, dV = 0 \tag{12}$$

where, $\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$.

With finite element discretization, this equation becomes,

$$\left(\int_\Omega B^T DB \, dV\right) u = 0 \tag{13}$$

- u : nodal displacement vector.
- B: matrix containing the derivatives of shape functions N such that $\varepsilon \approx Bu$.

Solving this equation is the first step in this project to introduce notation with differential operators. We have taken the liberty of rewriting the equation with a gradient operator as follows: $\nabla N := B$. The aim is to solve the Navier equation by writing the following expression in the program:

$$\int_\Omega (\nabla N)^T D(\nabla N) \, dV \tag{14}$$

3.1.2 Test

In order to test this first use of a gradient operator, a comparison is made between the result obtained by the gradient operator approach and the approach already implemented in Akantu. The case tested is a plate in plane strain, with unit thickness. Traction is applied in the y direction to the upper side of the plate ($y = 1$) . The displacements on the lower side of the plate ($y = 0$) has been locked in both directions.

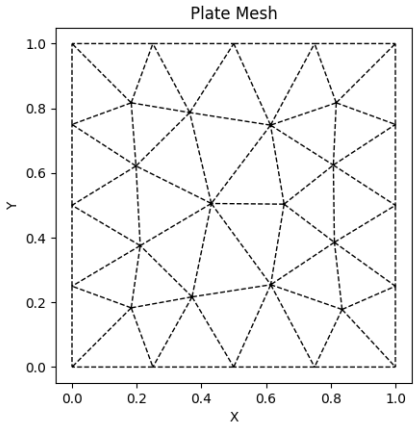

Figure 5: Plate mesh

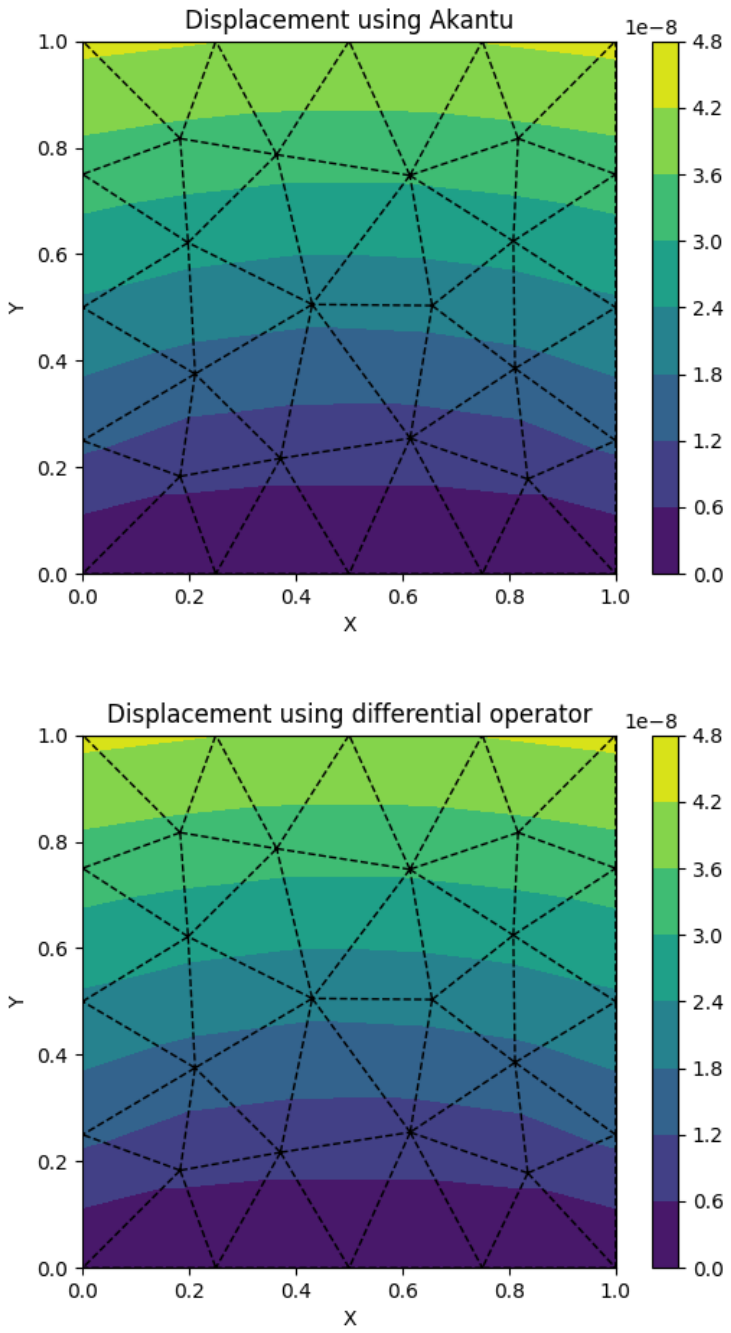

Figure 6: Comparing results for both approaches

The results obtained with both approaches are consistent, confirming the correct implementation of the differential operator considered.

3.2 Heat equation

3.2.1 Weak formulation

The mathematical development to get the weak form of this case is described in section 1.2.

Reminder of the strong formulation (2D) :

$$\alpha \operatorname{div}(\nabla T) = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = 0 \tag{15}$$

- $T(x, y)$: temperature,
- α : thermal diffusivity.

Reminder of the weak formulation :

$$\alpha \int_{\Omega} (\nabla T^*)^T \nabla T \, d\Omega = 0 \tag{16}$$

3.2.2 Test

In order to validate the implementation of the gradient operator, the heat equation is solved for a 1D case with a mesh composed of segments with 2 nodes, then segments with 3 nodes, and for a 2D case with a mesh composed of triangular elements with 3 nodes.

1D case : Consider a bar of 1 metre in length. We impose $T(x = 0) = 20°$, and $T(x = 1) = 10°$. Given that $\frac{\partial^2 T}{\partial x^2} = 0$ (15), we can express T analytically as follows:

$$T(x) = ax + b \tag{17}$$

Then, considering the boundary conditions, one finds :

$$T(x) = -10x + 20 \tag{18}$$

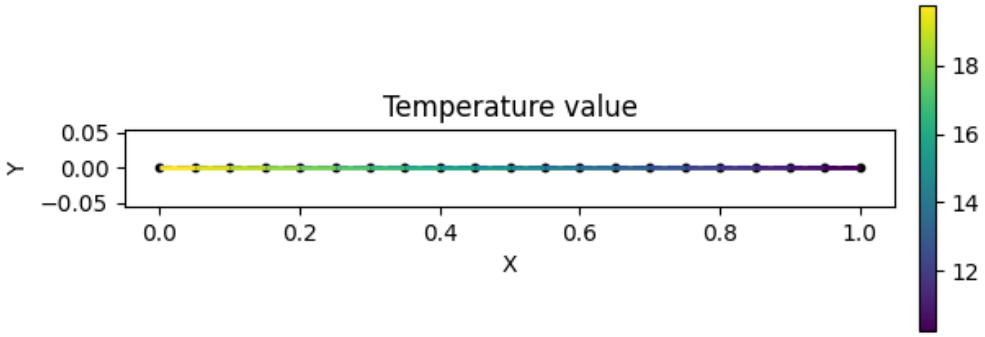

Figure 7: Segments with 2 nodes case

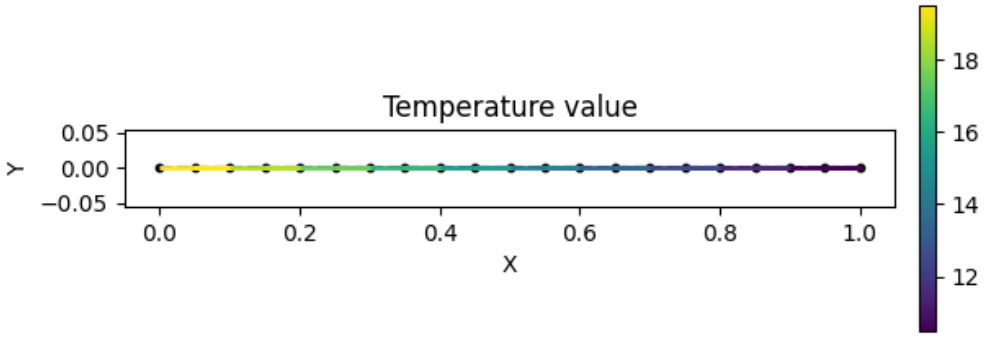

Figure 8: Segments with 3 nodes case

The expected analytical solution is found.

2D Case : Consider a plate measuring 1 m by 1 m, where $T(x = 0, y) = 20°$ and $T(x = 1, y) = 10°$. Since no variation on y is imposed, the solution should be independent of the y component, similar to the 1D case, i.e. :

$$T(x, y) = -10x + 20 \tag{19}$$

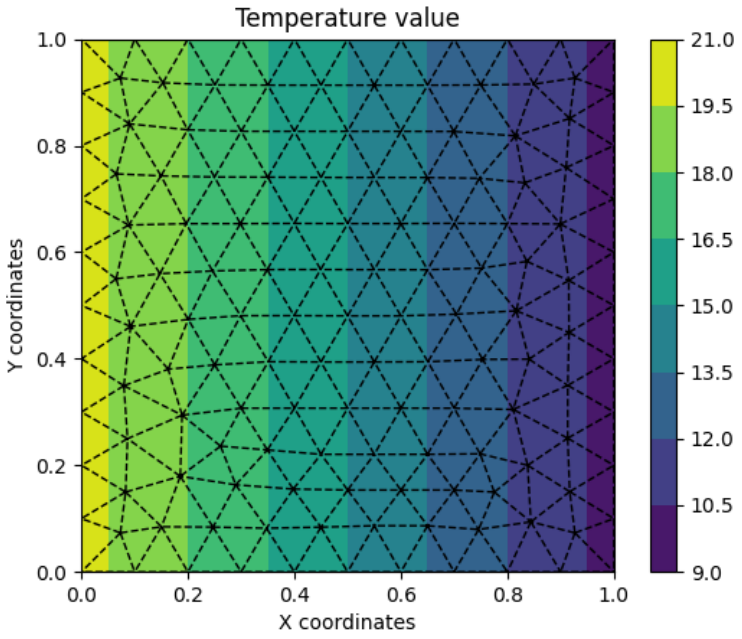

Figure 9: Results for the mesh with 3-node triangles

The expected analytical solution is found.

3.3 Curl Operator

3.3.1 Weak formulation

In the case of the curl operator, no known physical problems have been tested. However, the implementation of this operator is intended as a demonstrator of the future possibilities that can

be added to Akantu.
In order to still ensure the correct implementation of the rotational, the following integral form has been solved:

$$\int_{\Omega} (\nabla \times A) \cdot (\nabla \times \delta A)\, dV = 0 \tag{20}$$

3.3.2 Test

The case tested is a 3-D bar of $L_x = 1\ m$, $L_y = 1\ m$, $L_z = 10\ m$. The mesh is composed of thetrahedral elements.

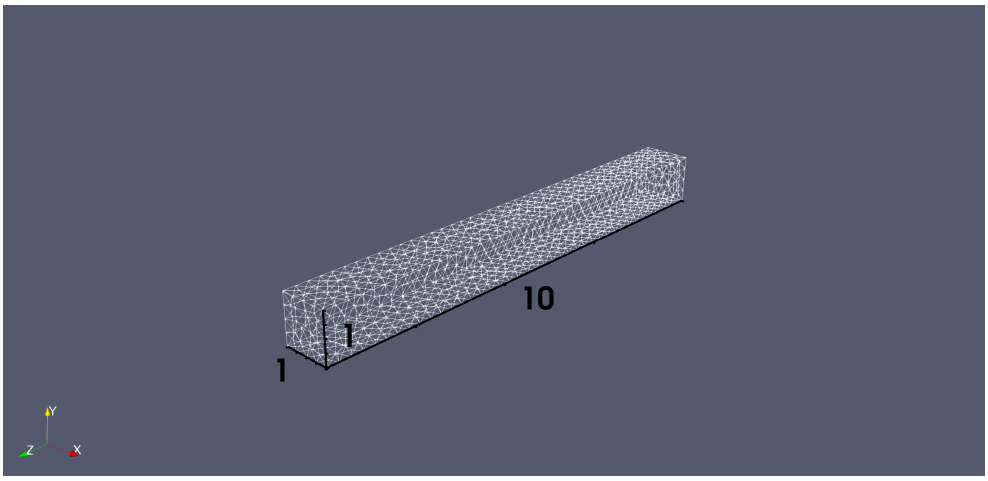

Figure 10: Bar mesh with thetahedral element

The boundary conditions applied are $A = (20, 20, 20)$ on the $z = 0$ plane.

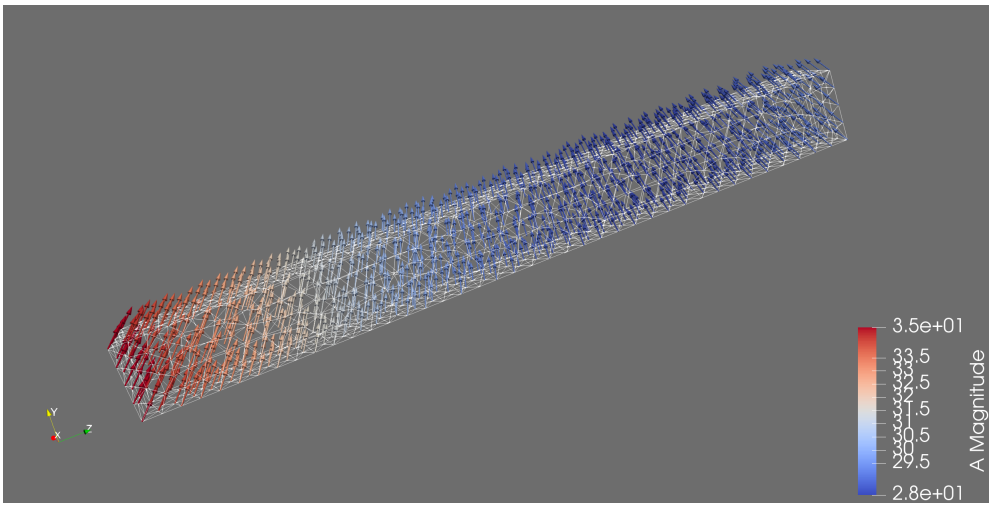

Figure 11: A vectors

The figure above shows the obtained values of A vectors for the tested case. The magnitude and orientation of the vectors vary without discontinuity. This suggests that the implementation of the rotational operator is correct. To be absolutely certain, however, it would be necessary to solve an equation with a known analytical solution.

4 Further

In this section, potential improvements and future prospects are highlighted.

- **Testing curl operator**: Although the implementation of the rotational has shown encouraging results, it is important to validate its implementation with a known analytical solution.
- **Redundancy** : Some functionalities such as matrix B assembly (3.1.1) or assembly methods following the "integrate" method (2.1) have been implemented, even though they were already possible with Akantu.
- **Apply boundary conditions** : Step 6 of the algorithm can be improved 6. Currently, it is necessary to apply boundary conditions explicitly to matrices and vectors in order to solve the linear system. It would be possible, as Akantu already does, to include groups of nodes when generating the mesh and thus facilitate the application of boundary conditions by reducing the number of lines of code required by the user when the linear system needs to be solved.
- **Adding other differential operators** : The current code structure, which uses object-oriented programming, enables easy code extension. This means that new differential operators can easily be added, depending on the functionality desired with Akantu.

5 Conclusion

This project shows that a wide variety of physical equations can be solved in their weak formulation in just a few lines of code. This is demonstrated through 3 examples. Firstly, the Navier equation has been solved by imposing a gradient operator notation. This introduction case has helped to familiarize with the challenges of the implementation and to structure the code for the rest of the work. Then, the gradient operator has been implemented and tested on a heat equation case. Finally, the rotational operator has been implemented. Although this last operator has not been tested on a concrete physical case where the analytical solution is known, the results obtained are encouraging and demonstrate the wide range of possibilities offered by the approach considered in this work.
Finally, the code is designed to be extendable in the future, perhaps to include other differential operators depending on the physical problems that users wish to solve with Akantu.

6 Appendix

Note : The Class diagram is available in Github with a better quality.

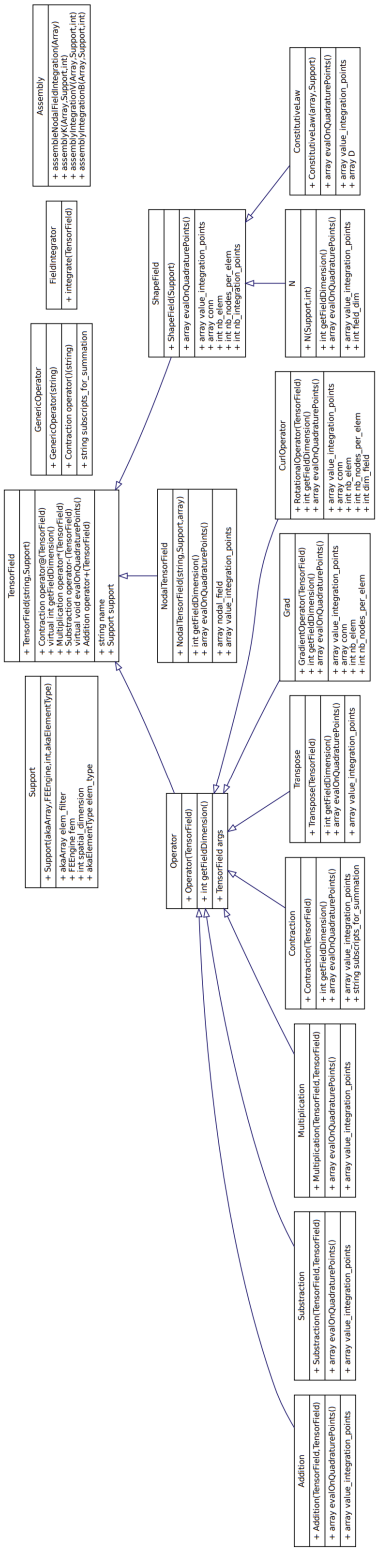

Figure 12: Class Diagram

Bibliography

1. Debard, Y. Méthode des éléments finis : thermique. *université du Mans.* https://iut.univ-lemans.fr/ydlogi/cours/thermique.pdf (2006).
2. Guillaume Anciaux, J.-F. M. CIVIL-321 : Numerical modelling of solids and structures. *EPFL* (2023).