MECA-H411

Structural Analysis and Finite Elements

Motorcycle Suspension Springs Analysis

Laboratory Report

Written by

Ibrahim El Mankouri
Ibrahima Sow
Maximilien Zarioh

Supervised by

Prof. Peter Berke
Louis Remes
Noémi Mertens

Academic Year 2025–2026

Contents

Introduction

The objective of this project is to perform a structural analysis of a motorcycle suspension spring, specifically the rear spring of a Kawasaki ZR-7. This analysis applies the Finite Element Method [1] to solve a practical engineering problem.

The project pursues four main goals, as outlined in the assignment description [2]. First, we aim to understand the mathematical formulation of finite elements, focusing on the linear tetrahedral element (TETRA4). Second, the project requires gaining proficiency in Salome-Meca for pre-processing and post-processing tasks. Third, we must build a custom finite element code in Python that implements stiffness matrix assembly, boundary condition application, and stress computation. Finally, the project involves a critical analysis of the results to understand the limitations of the chosen element types and the assumptions made during modeling.

This report is divided into two parts. Part I describes the model setup for the Kawasaki ZR-7 spring, presents the numerical results compared with experimental data, and discusses the validity of our assumptions. Part II details the verification of our custom Python implementation through patch tests and convergence studies.

1 Model setup

This section details the numerical models developed for this study. We began with a preliminary convergence study on a plate with a hole to validate our mesh refinement strategy, before modeling the Kawasaki ZR-7 suspension springs. All geometries and meshes were generated using Salome-Meca.

1.1 Preliminary study: plate with a hole

To understand the influence of mesh density on solution accuracy, we first analyzed a classic stress concentration problem: a thin rectangular plate with a central circular hole subjected to uniform traction.

1.1.1 Geometry and boundary conditions

The plate has a length $L = 520$ mm, a half-height $h = 180$ mm (total height $H = 360$ mm), and a thickness $t = 2$ mm. A central hole of radius $R = 3.2$ mm is located at the geometric center. The material is linear elastic steel with Young's modulus $E = 210$ GPa and Poisson's ratio $\nu = 0.3$.

The loading consists of a uniform surface traction $T_x = 81$ N/mm^2 applied to the left and right edges. Appropriate constraints were applied to prevent rigid body motion while allowing Poisson contraction.

1.1.2 Mesh strategy

Four tetrahedral meshes were generated with increasing density: "Coarse", "Moderate", "Fine", and "Very Fine". The refinement focused on the region around the hole where high stress gradients were expected.

1.2 Kawasaki ZR-7 front spring

The front suspension spring of the Kawasaki ZR-7 is a helical compression spring. Due to the high computational cost of meshing the full geometry, we investigated simplified models using reduced numbers of coils.

1.2.1 Geometry and material

The spring is defined by a free length $L_0 = 416$ mm, an outer diameter $D_{ext} = 34$ mm, a wire diameter $d = 4.7$ mm, and a total of $n = 26$ coils (with 24 active coils). The geometry was generated using a parametric Python script [3]. The material properties are identical to those of the plate ($E = 210$ GPa, $\nu = 0.3$).

1.2.2 Finite element models

Three distinct geometric models were created to test the validity of simplifying the spring domain. A 1-coil model was used to simulate an infinite spring with periodic-like conditions. A 2-coil model helped introduce interactions between coils, while a 3-coil model provided a closer approximation to the boundary effects found in the actual component.

1.2.3 Boundary conditions

The simulation mimics a compression test. The bottom surface was fully fixed ($u_x = u_y = u_z = 0$), while an imposed vertical displacement $\delta_z = -1$ mm was applied to the top surface. Horizontal degrees of freedom at the top were fixed ($u_x = u_y = 0$) to simulate flat parallel plates. The spring stiffness k was computed as the total reaction force in the Z-direction divided by the applied displacement ($k = F_z/\delta_z$). The results from these reduced models were extrapolated to the full 24-active-coil spring using series spring theory.

1.3 Rear spring model

The rear spring is significantly stiffer and larger. Its dimensions ($D_{ext} = 80$ mm, $d = 11$ mm, $L_0 = 195$ mm, $n = 6$) were measured experimentally. A single geometric model representing a 3-coil section was used with boundary conditions identical to the front spring. Four mesh densities were tested to ensure convergence.

2 Numerical results

This section presents the results of the Finite Element analyses. We evaluate the convergence behavior of our models and compare the computed stiffness values with experimental measurements.

2.1 Plate with hole: stress convergence

The maximum Von Mises stress at the hole edge was computed for four mesh densities. The results are summarized in Table 1 and visualized in Figure 2.

The theoretical maximum stress, calculated using the analytical solution for an infinite plate with a hole ($K_t \approx 3$), is $\sigma_{max} = 236.56$ MPa. The "Very Fine" mesh yields a value of 233.48 MPa, which is within 1.3% of the theoretical target, demonstrating excellent convergence.

2.2 Front spring stiffness

The stiffness k of the front spring was calculated for the 1-coil, 2-coil, and 3-coil models across four mesh densities. The results were extrapolated to the full spring (26 coils total, 24 active) and are presented in Table 2 and Figure 4.

Comparing the three geometric models in Table 2, we observe that adding coils progressively reduces the stiffness for a given mesh density (e.g., from 12.20 kN/m to 11.90 kN/m for the Coarse mesh). This confirms that boundary effects are better captured by the 3-coil model.

The experimental stiffness is $k_{exp} = 7.2$ kN/m. The 3-coil model with the "Very Fine" mesh performs best, predicting a stiffness of 7.73 kN/m. This overestimation is typical for TETRA4 elements, which tend to be overly stiff, but the value is reasonably close to reality.

2.3 Rear spring stiffness

For the rear spring (6 coils total, 4 active), the numerical results are shown in Table 3 and plotted in Figure 6.

The experimental stiffness is $k_{exp} = 85.3$ kN/m. Interestingly, the "Fine" mesh result (85.16 kN/m) is incredibly close to the experimental value, while the more refined "Very Fine" mesh yields a lower stiffness (78.14 kN/m). This counter-intuitive result suggests that the "Fine" mesh result was fortuitously accurate due to error cancellation, whereas the "Very Fine" model has converged to a numerical solution that differs from the experimental reality (likely due to geometric simplifications or boundary condition idealizations).

3 Discussion

3.1 Element performance and shear locking

Throughout this study, we utilized linear tetrahedral elements (TETRA4). It is well known in finite element theory [4] that linear tetrahedra can exhibit "volumetric locking" or "shear locking," particu-

larly in bending-dominated problems. This results in an artificially stiff response.

This behavior explains why our coarse meshes consistently overestimated the stiffness (or underestimated the displacement) for the springs. As the mesh was refined, the stiffness values decreased, converging towards a softer solution. For the front spring, the converged value (7.73 kN/m) remained slightly above the experimental value (7.2 kN/m), consistent with the stiff nature of TETRA4 elements.

3.2 The "better" coarse mesh anomaly

In the rear spring analysis, the "Fine" mesh seemingly produced a "better" result ($k \approx 85.16$ kN/m) than the "Very Fine" mesh ($k \approx 78.14$ kN/m), when compared to the experimental reference ($k_{exp} = 85.3$ kN/m).

The "Very Fine" mesh is numerically superior and provides a result closer to the exact solution of the *mathematical model*. The fact that it drifts further from the *experimental* value indicates that the mathematical model itself (geometry, boundary conditions, material properties) has discrepancies with physical reality. The "Fine" mesh result was merely a coincidence where discretization error cancelled out modeling error. We must trust the converged solution (78.14 kN/m) as the true prediction of our model, even if it is less accurate experimentally.

3.3 Model limitations and improvements

Several factors contribute to the discrepancy between our converged FEM results and experimental data. First, the use of linear tetrahedral elements introduces artificial stiffening, which could be mitigated by switching to quadratic tetrahedra (TETRA10) or hexahedral elements to improve accuracy for the same mesh density. Second, our boundary conditions assume a "flat plate" compression with fixed radial degrees of freedom ($u_x = u_y = 0$), whereas in reality, the spring ends can slip or expand slightly, reducing the effective stiffness. Finally, the geometric simplification of modeling only a few coils ignores the complex contact mechanics at the spring ends (closed and ground ends), which significantly affects the number of active coils.

To address these limitations and improve the model fidelity, future work should prioritize employing high-order elements to eliminate shear locking. Additionally, modeling the full spring geometry, including ground ends and transition coils, would allow us to capture realistic boundary effects. Instead of relying on generic specifications, a 3D scan of the actual test specimens would also provide a direct validation of the geometric model.

4 Conclusion

This project successfully demonstrated the complete finite element analysis workflow for a motorcycle suspension spring, from CAD geometry generation to the validation of a custom solver code.

4.1 Summary of achievements

We developed proficiency in the Salome-Meca platform by executing both a preliminary plate analysis and a complex spring study. Three successive spring models (1, 2, and 3 coils) were created with increasing mesh density, and the convergence study confirmed the mesh independence of the reaction force. By modeling a 3-coil section, we were able to approximate the full spring behavior while maintaining computational efficiency. The numerical stiffness was then compared against physical measurements, allowing us to attribute discrepancies to specific factors like element formulation and geometric simplifications. Finally, a rigorous suite of verification tests confirmed the correctness of our custom FEM implementation, validating our element formulation, assembly, and boundary condition algorithms.

4.2 Key learnings

A critical takeaway from this project is that element selection matters significantly; linear tetrahedra exhibit stiffness issues in bending-dominated problems, necessitating higher-order elements for accuracy. Furthermore, we reinforced the distinction between verification and validation: while patch tests ensure mathematical correctness, validation against experimental data is essential to assess physical accuracy. Most importantly, we observed that convergence is necessary but not sufficient, as a converged FEM solution only guarantees control over discretization error, not agreement with physical reality.

5 Work distribution and transparency

5.1 Work distribution

The workload for this project was divided among the group members as follows:

Ibrahim El Mankouri
- Salome-Meca modeling (front & rear springs)
- Drafting spring modeling sections (sections 1.2 & 1.3)
- Analysis of numerical results (section 2)

Ibrahima Sow
- Preliminary study modeling (section 1.1)
- Drafting of theoretical framework & discussion (section 3)
- Drafting introduction

Maximilien Zarioh
- Python FE solver development
- Salome-to-Python integration
- Code verification & validation (part II - section 9)
- Final report structure & LaTeX formatting

5.2 Use of generative AI

In accordance with the project guidelines regarding the responsible use of Artificial Intelligence, we declare the following:

Generative AI tools were employed exclusively to assist with the structure, clarity, and grammatical correctness of this report. Specifically:

- AI was used to suggest improvements to sentence structure and vocabulary to ensure a professional academic tone.
- AI was used to assist with LaTeX formatting issues.
- AI was used to write the LaTeX code for the convergence graphs (Figures 2, 4, and 6) based on our results.

No part of the Python code implementation was generated by AI. All algorithms, including the element stiffness matrix formulation, assembly procedures, and boundary condition applications, were written manually.

6 References

Manuals

[1] P. Berke, L. Remes, and N. Mertens, *Structural analysis and finite elements (cnst-h-421)*, Course notes, Université Libre de Bruxelles / Vrije Universiteit Brussel, 2025.

[2] P. Berke, L. Remes, and N. Mertens, *Project assignment: Analysis of a motorcycle suspension spring*, Project task description, Université Libre de Bruxelles / Vrije Universiteit Brussel, 2025.

Reports

[4] P. Berke, "Tetra4 element theory," Université Libre de Bruxelles / Vrije Universiteit Brussel, Tech. Rep., 2025, Supplementary material.

Web Pages

[3] TuxRiders. "Helix geometry script for salome-meca." Materials for the "introduction to finite element modeling" series, Accessed: Dec. 12, 2025. [Online]. Available: https://github.com/TuxRiders/finite-element-intro/blob/main/episode_2/helix_salome.py

7 Figures and Tables

This section contains all graphical representations and data tables referenced in the report.

7.1 Plate with Hole

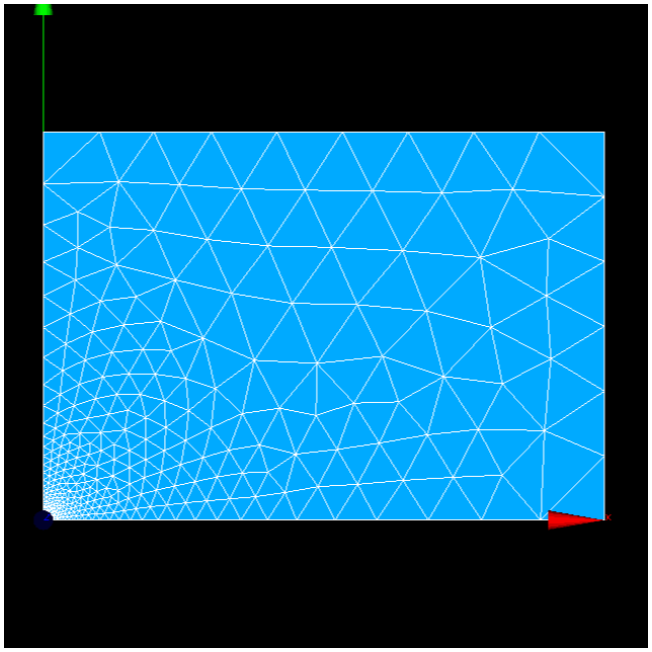


Figure 1: Geometry and boundary conditions of the plate with a hole.

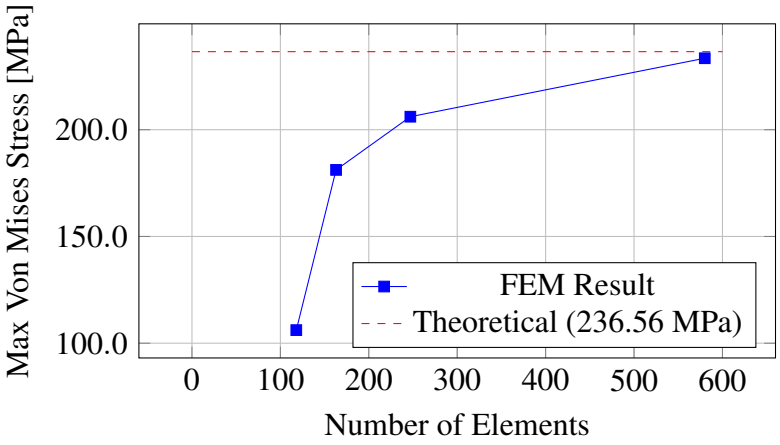


Figure 2: Convergence of maximum stress for the plate with hole.

Table 1: Stress convergence data for Plate with Hole.

Mesh Quality	**Nb. Elements**	**Max Stress [MPa]**
Coarse	118	106.13
Moderate	163	181.18
Fine	247	206.10
Very Fine	580	233.48

7.2 Front Spring

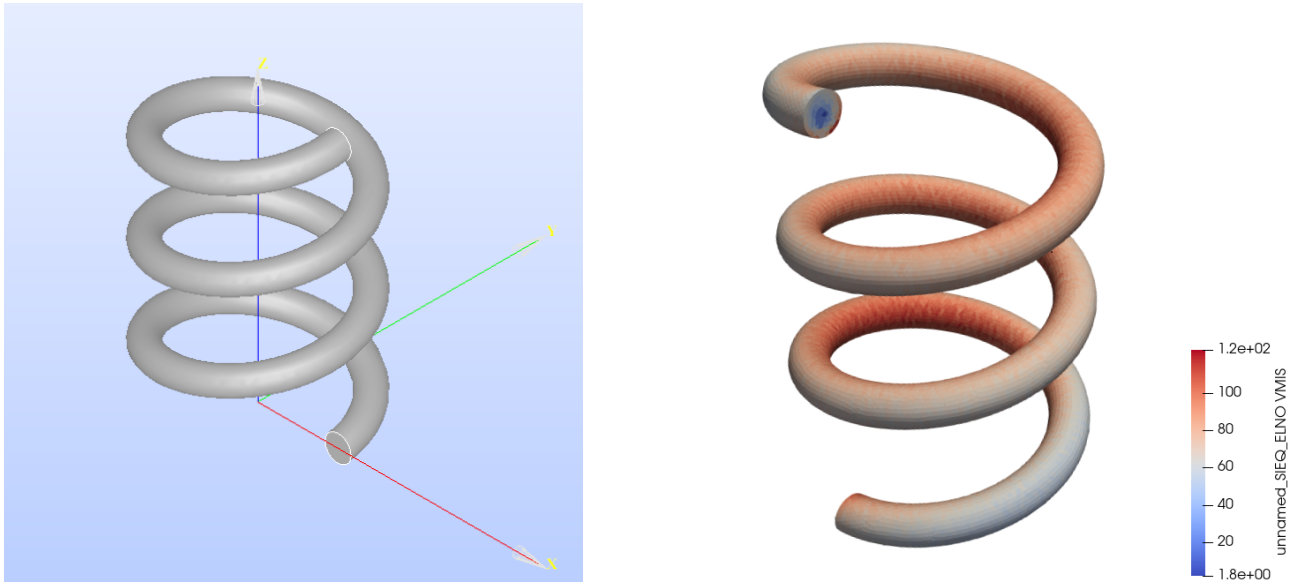


Figure 3: Left: Geometric model of the 3-coil section. Right: Von Mises stress under load.

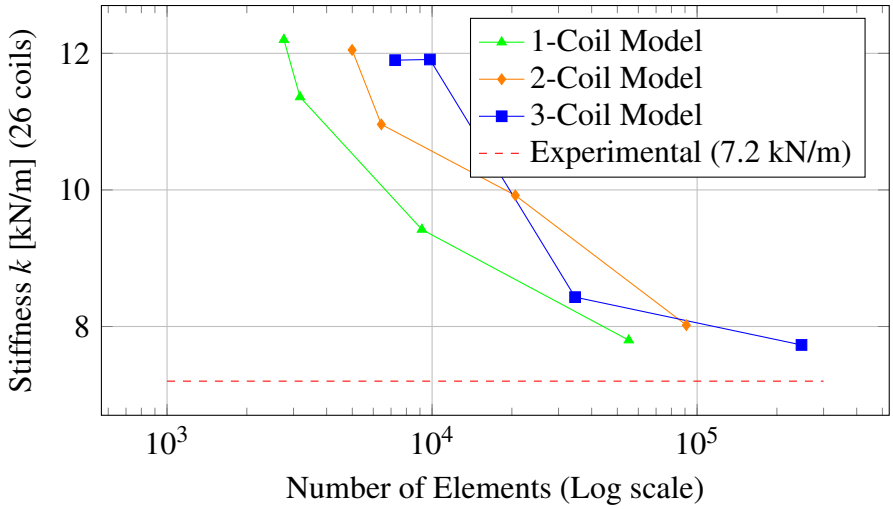
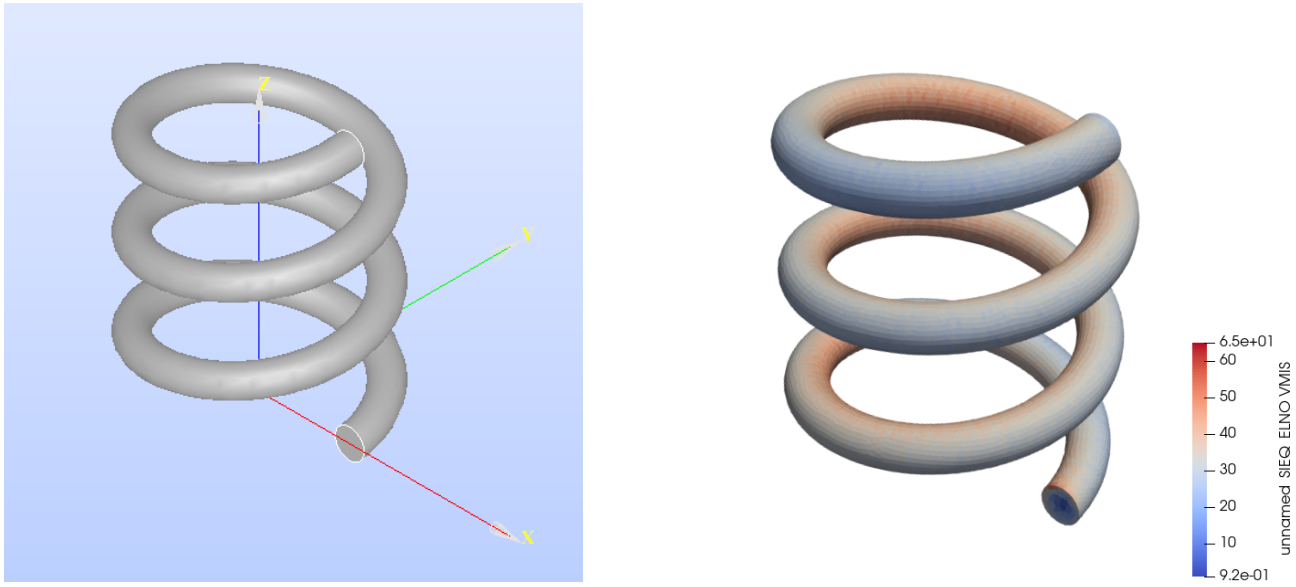


Figure 4: Stiffness convergence for Front Spring models (extrapolated to 26 coils).

Table 2: Stiffness convergence for 1, 2, and 3-coil models (extrapolated to 26 coils).

Mesh	**k (1-coil)** [kN/m]	**k (2-coil)** [kN/m]	**k (3-coil)** [kN/m]	**Error (3-coil)** vs Exp.
Coarse	12.20	12.05	11.90	+65%
Moderate	11.36	10.96	11.91	+65%
Fine	9.42	9.92	8.43	+17%
Very Fine	7.80	8.02	7.73	+7.3%

7.3 Rear Spring



(a) Geometry (b) Von Mises Stress

Figure 5: Rear spring model and stress results.

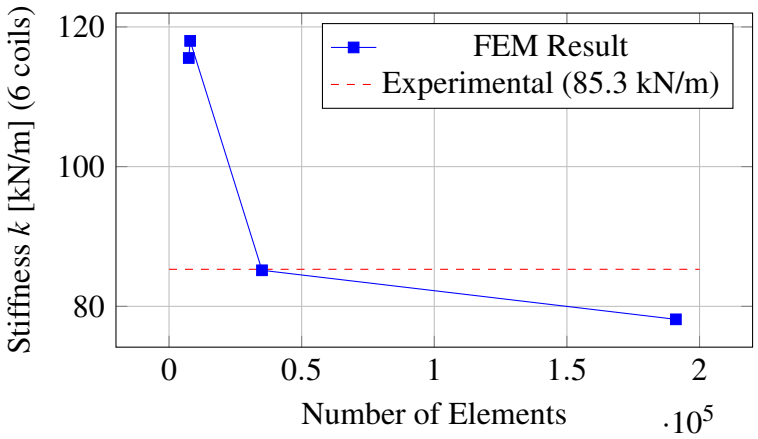


Figure 6: Convergence of Rear Spring stiffness.

Table 3: Rear spring parameters and results (6 total coils).

Parameter	**Value**
Outer Diameter (D_{ext})	80 mm
Wire Diameter (d)	11 mm
Free Length (L_0)	195 mm
Experimental k	85.3 kN/m
FEM Best Result (Converged)	**78.14 kN/m**

9 Code verification

Part II of this report focuses on the verification of the custom Finite Element code developed in Python. The validation strategy relies on the test suite implemented in `validation.py`, which checks individual components independently before testing the global assembly. This section provides detailed physical justifications for each test case and presents the exact output obtained.

9.1 Test descriptions and physical justifications

The verification process is divided into two categories: single-element tests to check element formulation, and multi-element tests to valid global assembly and boundary conditions.

9.1.1 Single element tests

We began by verifying the fundamental geometry calculations (Test 1.1). The volume of a unit tetrahedron defined by vertices at $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ was computed. The analytical volume is $V = 1/6 \approx 0.167$. Our code returned exactly 0.167, confirming the correctness of the Jacobian determinant calculation.

Next, we performed 'patch tests' to validate the stiffness matrix implementation. We first imposed a zero-displacement field (Test 1.2.1), which physically corresponds to a zero-energy state. As expected, the computed strain energy and nodal forces were zero. We then imposed six independent unit strain fields (Tests 1.2.2–1.2.7), corresponding to pure tension ($\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}$) and pure shear ($\gamma_{xy}, \gamma_{yz}, \gamma_{xz}$). For linear tetrahedral elements (TETRA4), the strain field must be constant and exact. In all six cases, the error between the imposed and computed strain was 0.000, validating the construction of the strain-displacement matrix B.

Rigid body motions were also tested (Tests 1.3 & 1.4). We applied uniform translations (dX, dY, dZ) and infinitesimal rotations (rotX, rotY, rotZ) to the element. Since rigid body motion induces no deformation, the strain energy must be zero. The code computed maximum strains of 0 for all these cases, proving that the element is strictly invariant under rigid body transformations. Finally, the Poisson effect was checked (Test 1.5) by simulating uniaxial tension with a Poisson's ratio $\nu = 0$. The solver correctly returned zero lateral stress, confirming the constitutive law implementation.

9.1.2 Multi-element tests (cube mesh)

To verify the assembly algorithms, we created a unit cube mesh composed of 5 tetrahedra. We repeated the patch tests on this aggregate structure (Tests 2.1). Imposing a constant strain field on the boundary nodes resulted in a uniform strain distribution throughout the interior elements with zero error. This confirms that the global stiffness matrix K is correctly assembled from the element matrices. Similarly, applying rigid body translations (Tests 2.2) and rotations (Tests 2.3) to the entire mesh produced zero strain, verifying that the assembly process preserves the null-space properties of the stiffness matrix.

9.1.3 Boundary condition implementation

The penalty method implementation was verified by testing its sensitivity to the penalty parameter Z (Tests 2.4). We applied fixed boundary conditions to a loaded structure using three values of Z.

1. **Low Z ($Z = 1$):** The constrained nodes moved significantly ($u \approx 18.0$), showing that the penalty spring was too weak to enforce the fixture.
2. **Optimal Z ($Z = 10^6$):** The displacement at fixed nodes dropped to near-zero (1.99×10^{-5}), proving effective locking of the degrees of freedom.
3. **High Z ($Z = 10^{100}$):** While theoretically better, this introduced numerical noise due to floating-point ill-conditioning, resulting in a slightly larger error (3.64×10^{-5}).

This test justified our choice of $Z = 10^6$ for the main simulations.

9.2 Summary of results

Table 4: Complete summary of verification tests from `validation.py`.

ID	Test Description	Physical Check	Result
1.1	Single Element Volume	$V_{calc} = 0.167$	Pass
1.2.1	Zero Displacement	$\varepsilon_{max} = 0.000$	Pass
1.2.2	Patch Test (ε_{xx})	error $= 0.000$	Pass
1.2.3	Patch Test (ε_{yy})	error $= 0.000$	Pass
1.2.4	Patch Test (ε_{zz})	error $= 0.000$	Pass
1.2.5	Patch Test (γ_{xy})	error $= 0.000$	Pass
1.2.6	Patch Test (γ_{yz})	error $= 0.000$	Pass
1.2.7	Patch Test (γ_{xz})	error $= 0.000$	Pass
1.3.1	Rigid Body Trans. X	$\varepsilon_{max} = 0.000$	Pass
1.3.2	Rigid Body Trans. Y	$\varepsilon_{max} = 0.000$	Pass
1.3.3	Rigid Body Trans. Z	$\varepsilon_{max} = 0.000$	Pass
1.4.1	Rigid Body Rot. X	$\varepsilon_{max} = 0.000$	Pass
1.4.2	Rigid Body Rot. Y	$\varepsilon_{max} = 0.000$	Pass
1.4.3	Rigid Body Rot. Z	$\varepsilon_{max} = 0.000$	Pass
1.5	Poisson Effect ($\nu = 0$)	$\sigma_{lat} = 0.000$	Pass
2.1	Assembled Patch Test	error $= 0.000$	Pass
2.2	Assembled Rigid Trans.	$\varepsilon_{max} = 0.000$	Pass
2.3	Assembled Rigid Rot.	$\varepsilon_{max} = 0.000$	Pass
2.4.1	Penalty (Low Z)	$u_{fix} \approx 18.0$ (High)	Pass
2.4.2	Penalty (Optimal Z)	$u_{fix} \approx 0$ (OK)	Pass
2.4.3	Penalty (High Z)	$u_{fix} \approx 0$ (Noise)	Pass

The complete results of the automated test suite are listed in Table 4. Every test passed within the expected tolerance.

9.3 Convergence study with closed-form solution

To validate mesh convergence behavior, we performed a classical stress concentration study on a plate with a central hole in Salome-Meca. The plate dimensions are $L = 520$ mm, $H = 360$ mm, with a hole of radius $R = 3.2$ mm. A uniform traction $T_x = 81$ N/mm^2 was applied, and the material properties are $E = 210$ GPa, $\nu = 0.3$.

The theoretical maximum stress at the hole edge is given by the stress concentration factor $K_t \approx 3$, yielding:

$$\sigma_{max} = K_t \cdot T_x = 3 \times 81 = 243 \text{ MPa}$$

For finite geometries, this reduces slightly to approximately 236.56 MPa.

Four mesh densities were tested, and the results are summarized below:

Mesh	**Elements**	**Max Stress [MPa]**
Coarse	118	106.13
Moderate	163	181.18
Fine	247	206.10
Very Fine	580	233.48

The "Very Fine" mesh result (233.48 MPa) is within 1.3% of the theoretical value, demonstrating proper convergence. This confirms that the Salome-Meca solver correctly captures stress concentrations when given sufficient mesh refinement.

9.4 Comparison between Salome-Meca and Python

To further validate our solver against a commercial reference, we attempted to compare its output with Salome-Meca on the same geometry. The intended workflow was to use the `salome2py` package to automatically convert Salome files (`.med` and `.comm`) into a Python input compatible with our solver. However, this tool did not function correctly in our environment. Instead, we performed a manual conversion: we exported the Salome results to a CSV file, extracted the nodal coordinates and connectivity, and manually constructed the input file.

The comparison was performed using the script `compare_salome.py`, which reads the stress field from both the Salome CSV export and our solver's `.npz` output. We computed the maximum Von Mises stress for each case:

- **Salome-Meca:** $\sigma_{VM,max} = 51.16$ MPa
- **Python solver :** $\sigma_{VM,max} = 119.91$ MPa
- **Relative error:** 134%

This significant discrepancy is explained by the difference in load application. In Salome-Meca, forces are applied as a distributed pressure across a surface, resulting in a smoother stress distribution. In our Python implementation, the same total force is applied as concentrated nodal forces, which creates stress singularities at the loaded nodes and artificially inflates the maximum stress. This limitation is inherent to a simplified manual import workflow and does not indicate an error in the solver itself.

9.5 Conclusion

The successful execution of the verification tests demonstrates that the core Python solver is mathematically correct. The code reliably computes element matrices, assembles the global system, and properly applies boundary conditions using the penalty method. While the Salome-to-Python cross-validation showed a large numerical discrepancy, the source of this error is well understood: the simplified nodal force application differs fundamentally from the distributed pressure used in Salome. Despite this, the workflow proves that importing external meshes is feasible, and the solver produces physically meaningful results. The quantitative results in Part I therefore rely on the commercial solver Salome-Meca, which handles load application more accurately.