On convergence of linearized stresses in an infinite pipe computed using the finite element method

Fino benchmark problem series

Number SP-FI-18-BM-B2BD

Rev. A

Date Feb/20/2019

Hash 371a8c4

Author Jeremy Theler jeremy@seamplex.com

Abstract

This technical report studies the linearized stresses according to ASME (i.e. membrane and membrane plus bending stresses) in an infinite-long cylindrical pipe subject to an uniform internal pressure. In particular, the linearized stresses calculated using the finite-element method (as computed by the free and open source tool Fino) depend on the number of element through the pipe thickness. These numerical results are compared to the analytical solution of the infinite-pipe problem. Differences between first and second-order grids, rate of convergence and errors as a function of the computational effort are addressed. The main conclusions are that it is mandatory to use at least second-order elements to accurately compute the membrane plus bending stresses and that two elements along the pipe thickness are enough to reasonably compute the linearized stresses.

Keywords

ASME, stress linearization, infinite pipe, finite elements

Contents

1 Introduction

The main objective of this report is to analyze the convergence of the ASME linearized stresses (membrane and membrane plus bending) in a pressurized infinitely-long circular pipe. This problem was chosen because

1. it is a typical problem tackled analytically in undergraduate courses in mechanical engineering (at least the displacements and stresses, not the ASME linearization),
2. it can be easily solved using standard finite-element tools (fig. 1) so it serves as a benchmark problem, and
3. there is a practical interest to compare stress analysis results from general piping systems to the canonical case of the infinite pipe.

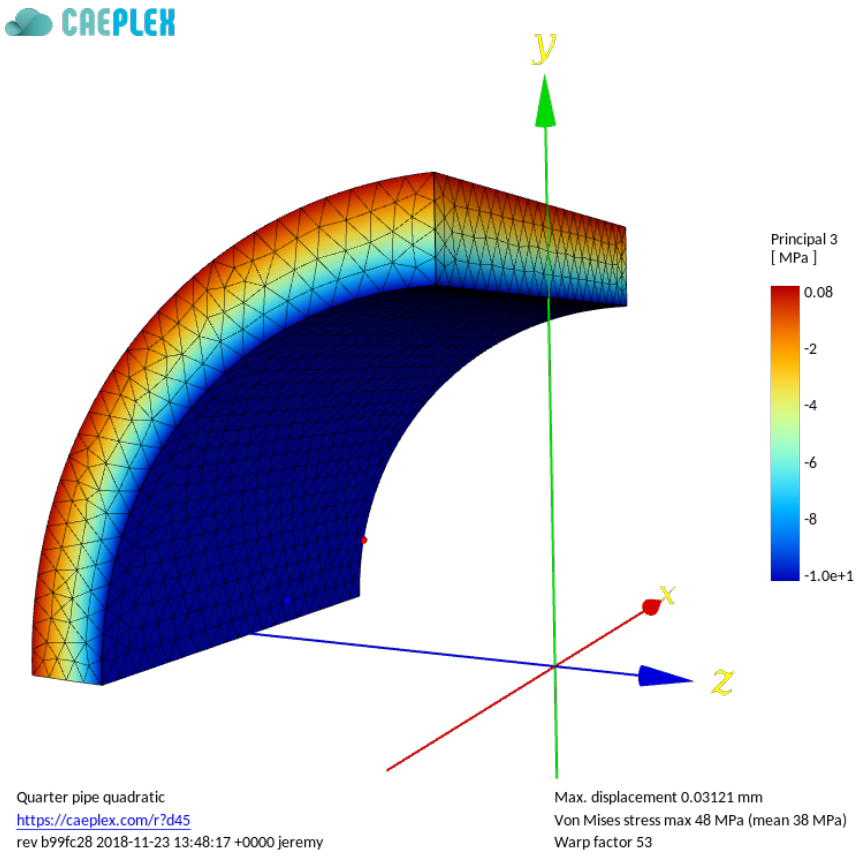


Figure 1: A quarter of an infinite pipe solved with the cloud-based tool CAEplex. The project can be seen (and solved) online with just a regular web browser (even a mobile one) at `https://caeplex.com/r?d45`

In this work, the term convergence is taken as the study of how the linearized stresses computed with a finite-element method change with respect to the number of elements through the pipe thickness. Briefly, this technical report

- analytically obtains the ASME-linearized (membrane and membrane plus bending) stresses for an infinite pipe subject to an uniform internal pressure,
- serves as a benchmarking test, comparing the results obtained by the free and open-source finite-element tool Fino and the analytical solution,

- shows how multi-freedom boundary conditions can be used to simplify the restriction of the rigid-body degrees of freedom in the case of null axial displacement and radial-only displacements,
- illustrates how parametric runs can be performed in Fino,
- analyzes the convergence of the ASME-linearized stresses with respect to the mesh coarseness,
- explores the differences between first and second-order elements regarding errors and computational efforts,
- compares the errors (with respect to the analytical solutions) obtained by using linear and quadratic elements as a function of the computational effort,
- quantitatively and qualitatively address the differences obtained in using both first and second-order elements.

For the finite-element computations, only tetrahedral unstructured grids are used to avoid biasing the results by having elements with edges parallel to the coordinate axis. Even more, the stress classification lines do no need to pass through any actual node of the grid. The software used to solve the finite-element formulation (sec. 1.1) allows the initial and end point of the stress classification lines to be arbitrary points of the domain. A parametric run is performed and the results are both qualitatively and quantitatively analyzed. A number of interesting conclusions are summarized in sec. 5. The appendices that follow dive into further advanced and optional topics.

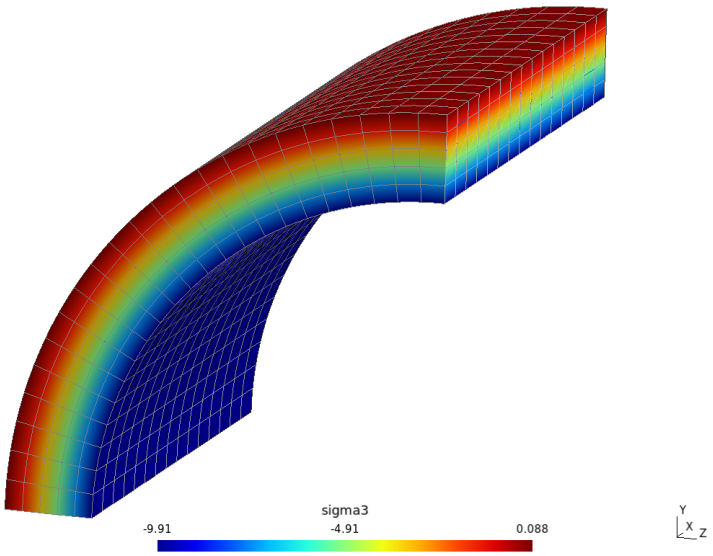


Figure 2: The infinite pipe could also be solved using an quarter-symmetry structured grid, but to avoid having preferred directions only unstructured tetrahedra around the whole circumference were used in this report.

It should be noted that the objective of this report is to study a very simple problem as thoroughly as possible in order to have a sound understanding of the underlying mechanical phenomena. There is absolutely no need to employ finite-element analysis to solve an infinite pressurized pipe. However, finite-elements ought to be employed to analyze any real piping network such as the one depicted in fig. 3.

1.1 About Fino

Fino is a free and open source tool that uses the finite-element method to solve

- steady-state thermo-mechanical problems, or

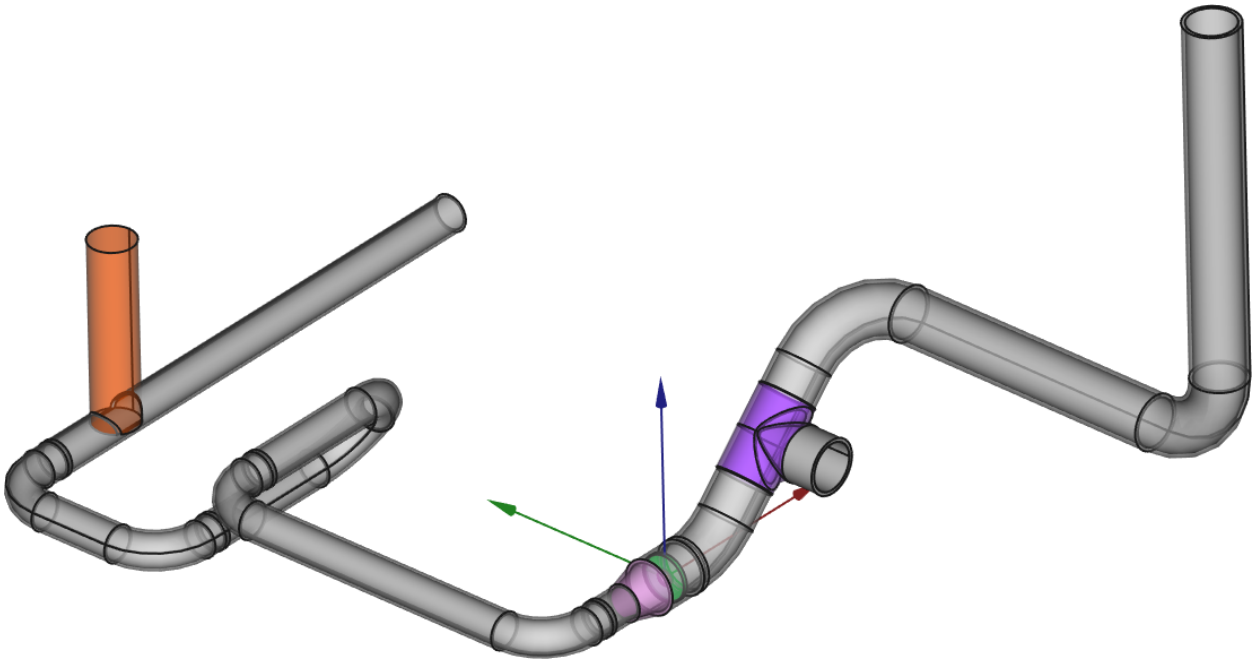


Figure 3: A real-life piping system, with reductions, orifice plates, valves, tees, etc.

- steady or transient heat conduction problems, or
- modal analysis problems.

It is particularly designed to handle complex dependence of material properties (i.e. temperature-dependent properties). It can also perform parametric or optimization runs. The domain over which the problem is solved should be a grid generated by Gmsh. The material properties and boundary conditions may involve arbitrary dependence of space associated to physical entities defined in the mesh.

Fino follows, amongst others, the UNIX philosophy. Fino is a back-end aimed at advanced users. For an easy-to-use web-based front-end with Fino running in the cloud, see CAEplex.

Fino is a free ("Free" both as in "free speech" and in "free beer.") and open source finite-element analysis tool Fino developed from scratch by Seamplex. From our point of view, only open source software[1] should be employed in solving engineering problems. A cognizant engineer cannot rely on privative software and be certain that the results are accurate, exact or even valid enough to draw an educated conclusion out of them. Any reasonable engineering system cannot depend on black boxes.

Figure 4: Fino is freely available from `https://www.seamplex.com/fino`

[1] And if it is free software, even better.

2 Problem

Let us consider an infinite pipe (i.e. a cylinder) of internal radius a and external radius b with uniform mechanical properties—Young modulus E and Poisson's ratio ν—subject to an internal uniform pressure p. We would like to compute the linearized membrane and membrane plus bending stresses according to ASME VIII Section 5 ("ASME Boiler and Pressure Vessel Code, Section VIII, Rules for Construction of Pressure Vessels: Division 2—Alternative Rules" 2010, annex 5-A). To obtain a numerical solution with the finite-element method, let us fix the numerical values of the parameters as follows

$$\begin{aligned} b &= 161.9 \text{ mm} \\ a &= 140.4 \text{ mm} \\ E &= 200 \text{ GPa} \\ \nu &= 0.30 \end{aligned}$$

which correspond to a 12"-inch schedule 100 carbon steel pipe according to ASME B36-10M ("Welded and Seamless Wrought Steel Pipe" 2004). Again, for the sake of numerical results, we fix the internal pressure to

$$p = 10 \text{ MPa}$$

that is a typical pressure found in nuclear power plants (so ASME III applies).

3 Reference analytical solution

Given the cylindrical symmetry of the problem, there is only one independent variable, namely the radial coordinate r. Moreover, there are only two displacement fields that need to be considered: the axial $u_a(r)$ and the radial $u_r(r)$. The former is identically zero due to the fact that the cylinder is infinite (plane strain condition). The equilibrium of momentum along the radial direction r is

$$\frac{d\sigma_r}{dr} + \frac{\sigma_r(r) - \sigma_\theta(r)}{r} = 0 \tag{1}$$

Defining the strains as

$$\begin{aligned} \epsilon_r &= \frac{du_r}{dr} \\ \epsilon_\theta &= \frac{u}{r} \end{aligned}$$

and the stresses as

$$\begin{aligned} \sigma_r &= \frac{E}{(1-\nu)(1-2\nu)} \cdot \Big[(1-\nu)\cdot\epsilon_r + \nu\cdot\epsilon_\theta\Big] \\ \sigma_\theta &= \frac{E}{(1-\nu)(1-2\nu)} \cdot \Big[\nu\cdot\epsilon_r + (1-\nu)\cdot\epsilon_r\Big] \end{aligned}$$

then the differential equation 1 can be casted in terms of the axial displacement $u_r(r)$ as

$$\frac{d^2u}{dr^2} + \frac{1}{r} \cdot \frac{du}{dr} - \frac{u}{r^2} = 0$$

that has the general solution

$$u(r) = c_1 \cdot r + \frac{c_2}{r}$$

For the boundary conditions of the particular problem that the radial stress should be equal to the negative of the internal pressure[2] at $r = a$ and null at $r = b$, namely

$$\begin{cases} \sigma_r(a) & = -p \\ \sigma_r(b) & = 0 \end{cases}$$

the axial displacement has the particular solution

$$u_r(r) = p \cdot \frac{1+\nu}{E} \cdot \frac{a^2}{b^2 - a^2} \cdot \left[1 - 2\nu + \frac{b^2}{r^2}\right] \cdot r$$

and the stresses the radial, tangential (hoop) and axial (longitudinal) stresses are

$$\sigma_r(r) = \frac{p \cdot a^2}{b^2 - a^2} \cdot \left(1 - \frac{b^2}{r^2}\right) \tag{2}$$

$$\sigma_\theta(r) = \frac{p \cdot a^2}{b^2 - a^2} \cdot \left(1 + \frac{b^2}{r^2}\right) \tag{3}$$

$$\sigma_l(r) = 2\nu \cdot \frac{p \cdot a^2}{b^2 - a^2} \tag{4}$$

3.1 Stress linearization

According to the ASME VIII code ("ASME Boiler and Pressure Vessel Code, Section VIII, Rules for Construction of Pressure Vessels: Division 2—Alternative Rules" 2010), a *stress classification line* should be chosen to compute the linearized stresses. Theses SCLs should be normal to iso-stress lines, which in this case may be any line radial from $r = a$ to $r = b$. The ASME code defines that each element of the *membrane* stress tensor M is

$$\mathrm{M}_{ij} = \frac{1}{b-a} \cdot \int_a^b \sigma_{ij}(r)\, dr$$

for $i = x, y, z$ and $j = x, y, z$. Having computed the nine values (of which only six are different) of such tensor, different kind of scalar membrane stresses can be obtained depending on how a single scalar stress "intensity" is computed out of the nine tensor elements. In particular we take into account the Tresca,[3] Von Mises and the three principal membrane stresses. As all the shear stresses are zero, σ_l, σ_r and σ_θ happen to also be the principal stresses with

[2] Exercise for the reader: why the negative of the pressure and not the pressure itself?

[3] Even though recent revisions of the ASME VIII code dropped the Tresca criterion in favor of Von Mises, the former is still the main way of computing the stress "intensity" in ASME III.

$$\sigma_1 = \sigma_\theta$$
$$\sigma_2 = \sigma_l$$
$$\sigma_3 = \sigma_r$$

So the membrane stresses to be considered are

$$\mathrm{M}_1 = \frac{1}{b-a} \cdot \int_a^b \sigma_\theta(r)\, dr$$
$$\mathrm{M}_2 = \frac{1}{b-a} \cdot \int_a^b \sigma_l(r)\, dr$$
$$\mathrm{M}_3 = \frac{1}{b-a} \cdot \int_a^b \sigma_r(r)\, dr$$
$$\mathrm{M}_{\text{tresca}} = \max\left\{ |\mathrm{M}_1 - \mathrm{M}_2|, |\mathrm{M}_2 - \mathrm{M}_3|, |\mathrm{M}_3 - \mathrm{M}_1| \right\}$$
$$\mathrm{M}_{\text{vonmises}} = \sqrt{\frac{(\mathrm{M}_1 - \mathrm{M}_2)^2 + (\mathrm{M}_2 - \mathrm{M}_3)^2 + (\mathrm{M}_3 - \mathrm{M}_1)^2}{2}}$$

In particular, replacing the stress distribution of the infinite pipe given by eqns. 4, 2, 3 and carrying out the integrals, the principal membrane stresses are

$$\mathrm{M}_1 = \frac{a\, p}{b-a}$$
$$\mathrm{M}_2 = \frac{2\, a^2\, \nu\, p}{b^2 - a^2}$$
$$\mathrm{M}_3 = \frac{a^2 \left(2\, b - \frac{b^2+a^2}{a}\right) p}{(b-a)\ (b^2-a^2)}$$

The other linearized stress, namely the *membrane plus bending* stress tensor MB—again according to ASME VIII Annex 5-A—is

$$\mathrm{MB}_{ij} = \mathrm{M}_{ij} \pm \frac{6}{(b-a)^2} \cdot \int_a^b \sigma_{ij}(r) \cdot \left(\frac{a+b}{2} - r\right) dr \tag{5}$$

where the sign should be taken such that the resulting combined scalar stress (i.e. Tresca, Von Mises, etc.) represents the worst-case scenario. As before, the membrane plus bending scalar stresses are

$$\mathrm{MB}_1 = \mathrm{M}_1 \pm \frac{6}{(b-a)^2} \cdot \int_a^b \sigma_\theta(r) \cdot \left(\frac{a+b}{2} - r\right) dr$$
$$\mathrm{MB}_2 = \mathrm{M}_2 \pm \frac{6}{(b-a)^2} \cdot \int_a^b \sigma_l(r) \cdot \left(\frac{a+b}{2} - r\right) dr$$
$$\mathrm{MB}_3 = \mathrm{M}_3 \pm \frac{6}{(b-a)^2} \cdot \int_a^b \sigma_r(r) \cdot \left(\frac{a+b}{2} - r\right) dr$$
$$\mathrm{MB}_{\text{tresca}} = \max\left\{ |\mathrm{MB}_1 - \mathrm{MB}_2|, |\mathrm{MB}_2 - \mathrm{MB}_3|, |\mathrm{MB}_3 - \mathrm{MB}_1| \right\}$$
$$\mathrm{MB}_{\text{vonmises}} = \sqrt{\frac{(\mathrm{MB}_1 - \mathrm{MB}_2)^2 + (\mathrm{MB}_2 - \mathrm{MB}_3)^2 + (\mathrm{MB}_3 - \mathrm{MB}_1)^2}{2}}$$

For the infinite pipe case, MB_1 is maximum and MB_2 is minimum if we take the positive sign in eq. 5. The integral in the expression for MB_2 above vanishes. The principal membrane plus bending stresses are then

$$\mathrm{MB}_1 = \frac{6\,a^2\,\left(\frac{b^3+(2\,a\,\log a+a)\,b^2-a^2\,b}{2\,a} - \frac{2\,b^2\,\log b+b^2}{2}\right)\,p}{(b-a)^2\,(b^2-a^2)} + \frac{a\,p}{b-a}$$

$$\mathrm{MB}_2 = \frac{2\,a^2\,\nu\,p}{b^2-a^2}$$

$$\mathrm{MB}_3 = \frac{6\,a^2\,\left(\frac{2\,b^2\,\log b+b^2+2\,a\,b}{2} - \frac{b^3+(2\,a\,\log a+a)\,b^2+a^2\,b}{2\,a}\right)\,p}{(b-a)^2\,(b^2-a^2)} + \frac{a^2\,\left(2\,b-\frac{b^2+a^2}{a}\right)\,p}{(b-a)\,(b^2-a^2)}$$

For the particular parameters fixed in sec. 2, the numerical values for the theoretical linearized membrane and membrane plus bending scalar stresses are

$$\begin{aligned}
\mathrm{M}_{\mathrm{tresca}} &= 69.9467\ \mathrm{MPa} \\
\mathrm{M}_{\mathrm{vonmises}} &= 61.7785\ \mathrm{MPa} \\
\mathrm{M}_1 &= 65.3023\ \mathrm{MPa} \\
\mathrm{M}_2 &= 18.1974\ \mathrm{MPa} \\
\mathrm{M}_3 &= -4.6444\ \mathrm{MPa}
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{MB}_{\mathrm{tresca}} &= 79.9062\ \mathrm{MPa} \\
\mathrm{MB}_{\mathrm{vonmises}} &= 70.2562\ \mathrm{MPa} \\
\mathrm{MB}_1 &= 70.2821\ \mathrm{MPa} \\
\mathrm{MB}_2 &= 18.1974\ \mathrm{MPa} \\
\mathrm{MB}_3 &= -9.6241\ \mathrm{MPa}
\end{aligned}$$

respectively. See sec. B.1 and sec. B.2 for the input files used to compute both the integrals and the numerical values, either using Maxima (symbolic integration) or wasora (Theler 2016a) (numerical integration).

4 Parametric finite-element analysis solution

In principle, the problem can be solved as a bi-dimensional plane-strain case. Even more, it is actually a one-dimensional problem with radial dependence only. Nevertheless, we solve a full three-dimensional pipe because we are interested in using the same set of methods and tools we would use in a real-case piping system. We do not even use any half or quarter symmetry but the full 360° annuli.

In order to study the convergence of the linearized stresses computed using the stresses obtained in a finite-element solution, the free and open source tool Fino—which gives the three-dimensional displacement field $[u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})]$ and the corresponding stress tensor distribution—is used. The model geometry is first modeled and then meshed using the free and open source tool Gmsh (Geuzaine and Remacle

2009). The former consists of a cylinder with radii $a < b$ and a length $\ell = 2(b - a)$ (as illustrated in fig. 5) which was directly built using the constructive solid modeling primitives provided by Gmsh through the OpenCASCADE library (fig. 5). The latter is generated parametrically over $n = 1, \ldots, 6$, which indicates the number of expected elements through the pipe wall thickness $b - a$.

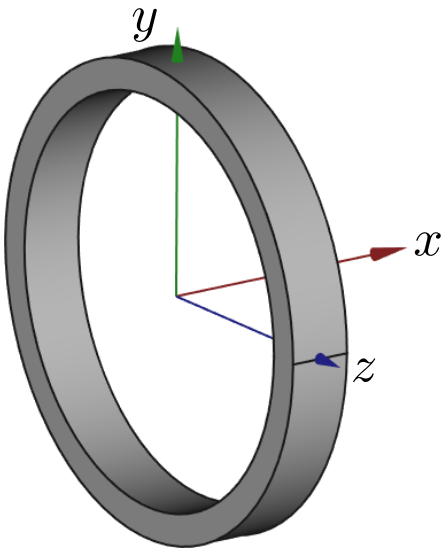


Figure 5: Geometry used for the finite-element analysis. The axial length is twice the wall width $\ell = 2(b - a)$.

Even though the cylinder is a rather simple and symmetric geometry, the grids are fully unstructured in order to avoid biased results that might correspond to a particular choice of the grid structure. It is also because of this reason that a *full* cylinder is modeled instead of just half or a quarter cylinder so there are no element edges parallel to the coordinate axes. In particular, both first and second-order tetrahedra are used for each value of n, giving rise thus to the analysis of twelve different FEM cases. The computational implementation uses the GNU M4 processor called from Fino to expand macros in a Gmsh geometry file template with the particular values of a, b, n and the mesh order (sec. B.3).

The origin $(0, 0, 0)$ of the coordinate system is located at the cylinder's barycenter, while the main axis corresponds to the cartesian x axis, as illustrated in fig. 5. The internal cylindrical face is subject to a uniform pressure p. Due to the fact that we are modeling an infinite cylinder, the external annuli which are parallel to the y-z plane should have a null axial displacement ($u = 0$) to recover the plane-strain conditions. Moreover, the displacements can only be radial so these faces are subject to the condition, remembering that the cylinder's axis is located at $y = z = 0$, that

$$v(x, y, z) \cdot z - w(x, y, z) \cdot y = 0 \tag{6}$$

See sec. A for a detailed derivation of this expression. These three boundary conditions are applied over two different faces, namely

- compression pressure p in the internal cylindrical face, and
- null axial displacement $u = 0$ and only radial displacement $vz - wy = 0$ in the two external axial plane surfaces.

can be easily implemented in Fino as shown in sec. B.4. Note that even though the two conditions in the second bullet correspond to Dirichlet (or essential) boundary conditions, only the first one $u = 0$ is

exactly enforced (up to the error of the iterative linear solver). The second one corresponds to a multi-freedom Dirichlet boundary condition and is implemented using a penalty method and it is thus only approximately imposed.[4]

4.1 Parametric grids

By expanding the macros of the Gmsh input template shown in sec. B.3 for the parameter $n = 1, \ldots, 6$ that represents the number of elements along the pipe wall thickness, the grids shown in figures 6a–8b are obtained. It should be noted that the number and distribution of elements is *exactly* the same in the first and second-order cases. However, not only is the number of nodes different but also the new second-order nodes (i.e. the nodes located along each of the the six edges the tetrahedra have) are placed on the curved surfaces of the cylinder. Thus, the original first-order tetrahedra are *slightly* "distorted" to better represent the continuous geometry in the second-order case. In any case, this distortion is so slight that each case cannot be graphically distinguished even for the coarser case. Nevertheless, this effect is present and is illustrated in the extremely coarse grids presented in fig. 9. The extra cost of this better geometrical representation is a greater number of nodes, which in turn leads to an increase in the computational effort needed to solve the associated finite-element problem as discussed in sec. 4.5. Even though quadratic iso-parametric (i.e. standard) elements can *better* represent circles, they are still approximations. One need to switch to iso-geometric elements for that end.

4.2 Displacements

The finite-element solver gives a vector field of displacements $[u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})]$ in each of the three Cartesian directions x, y and z, respectively. Due to the cylindrical symmetry, the only non-zero displacements will be $v(\mathbf{x})$ and $w(\mathbf{x})$ which, furthermore, will be related in such a way that only radial net displacements occur. Given the radial nature of the three-dimensional geometry, the radial displacement $u_r(r)$ should be compared with

$$\sqrt{[v(x, r\cos\theta, r\sin\theta)]^2 + [w(x, r\cos\theta, r\sin\theta)]^2}$$

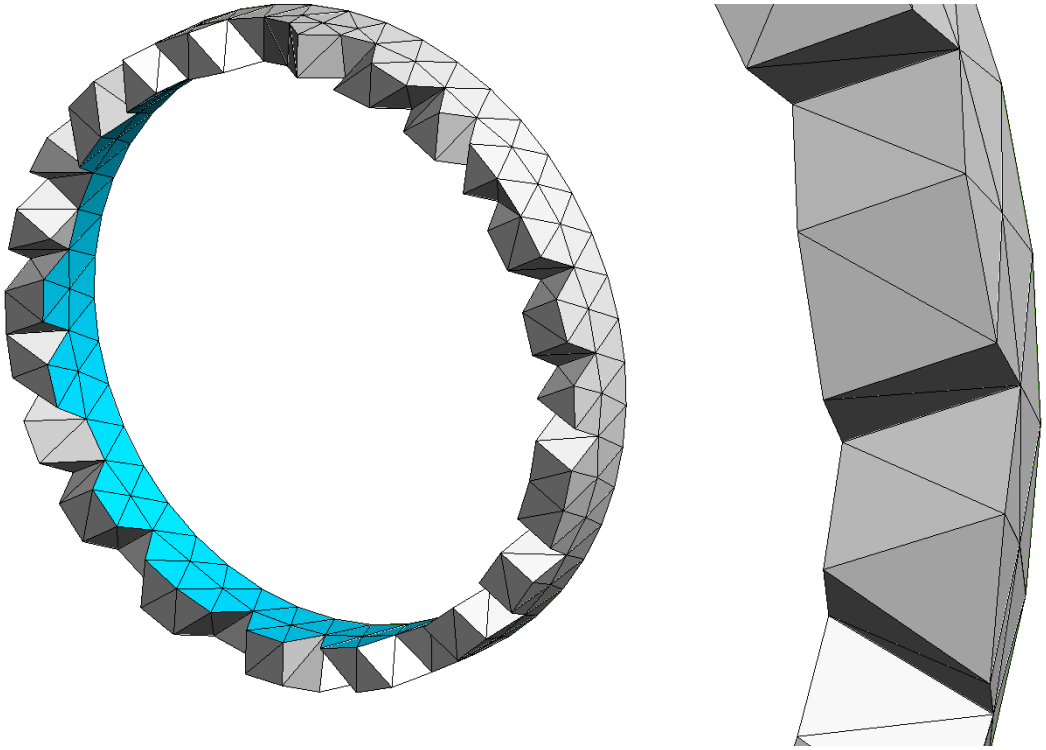
for any $-\ell/2 \leq x \leq +\ell/2$, $a \leq r \leq b$ and azimuthal angle θ. It should be noted that the grid is *fully* unstructured, so there is no particular choice of x and θ than can give different results, say because there are element edges aligned with the coordinate axis. To simplify the extraction of the numerical data, we can take either $v(0, r, 0)$ or $w(0, 0, r)$ as a representative measure of the radial displacement $u_r(r)$.

Fino allows the displacements $[u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})]$ (and the stresses) to be evaluated at any arbitrary point $\mathbf{x}$, i.e. they are real continuous functions of space and not just scattered values (Theler 2016b). The figures in this and the following sections show these full continuous functions of the finite-element solutions as interpolated by the shape functions. Fig. 10 compares the analytical solution of the radial displacement $u_r(r)$ with the numerical solutions for different combinations of mesh order and n as obtained by Fino. It can be seen that for the second-order grid, a low value of n seems to be more accurate than the finer first-order case.

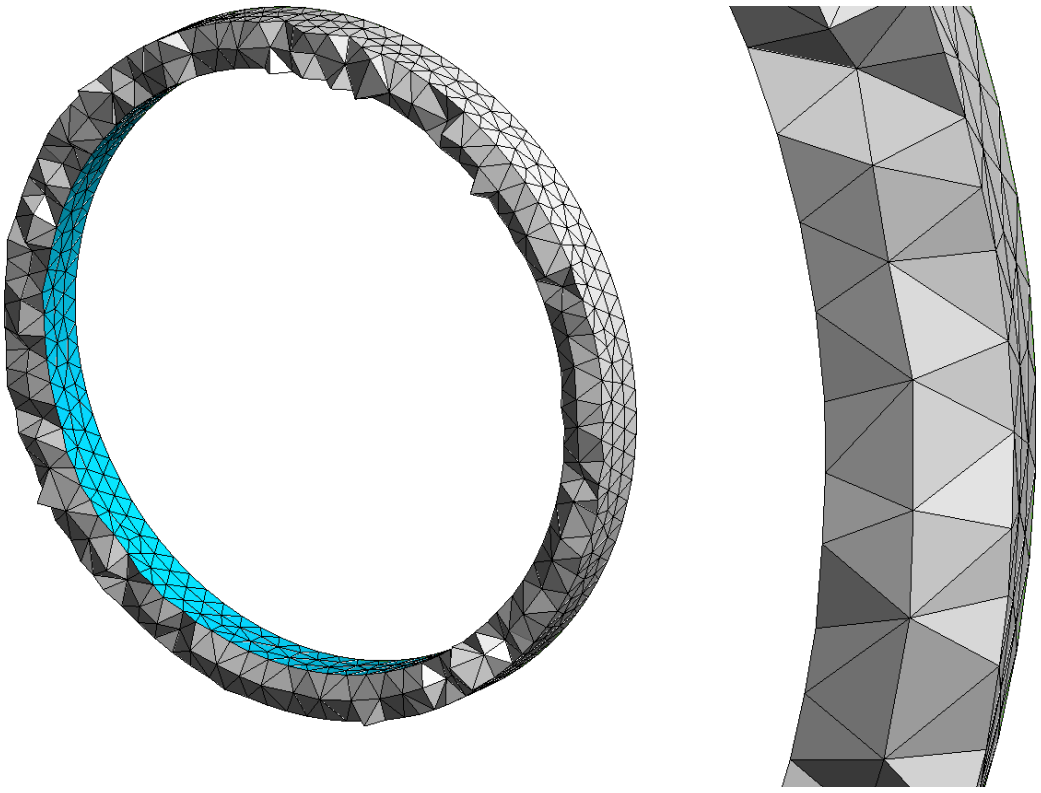
4.3 Stresses

Let us now switch to analyze the stresses, where the difference between first and second-order grids is even more evident. Depending on how the finite-element code averages nodal values from the gradient of

[4] The null axial displacement condition $u = 0$ is a traditional Dirichlet boundary condition because the outward normal $\hat{\mathbf{n}}$ coincides with one of the three coordinate axes. In the general case, it should also be set as a multi-freedom Dirichlet boundary condition.

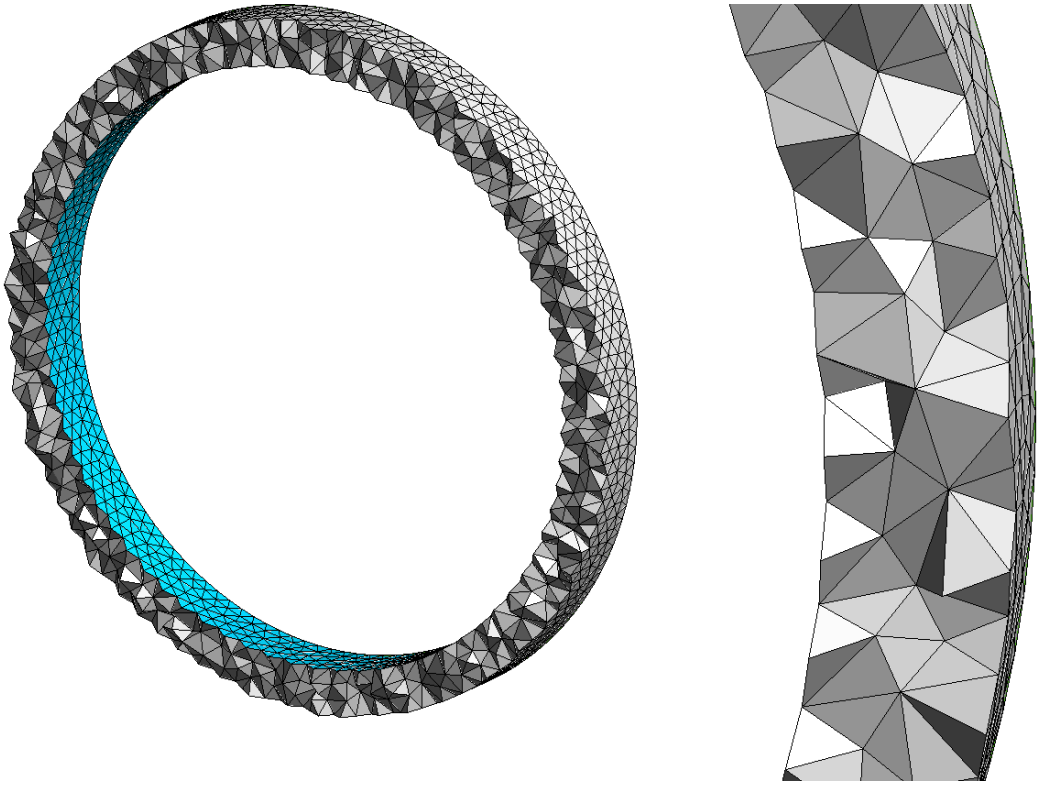


(a) $n = 1$

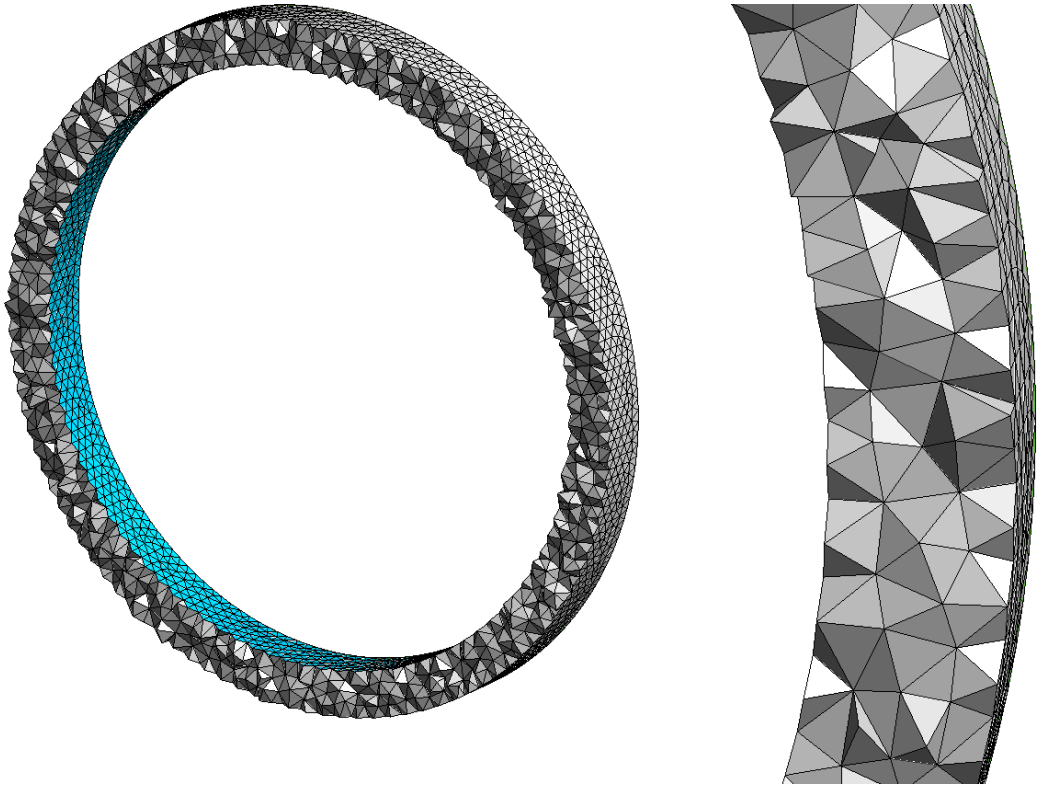


(b) $n = 2$

Figure 6: Parametric grids with n elements through the pipe thickness.

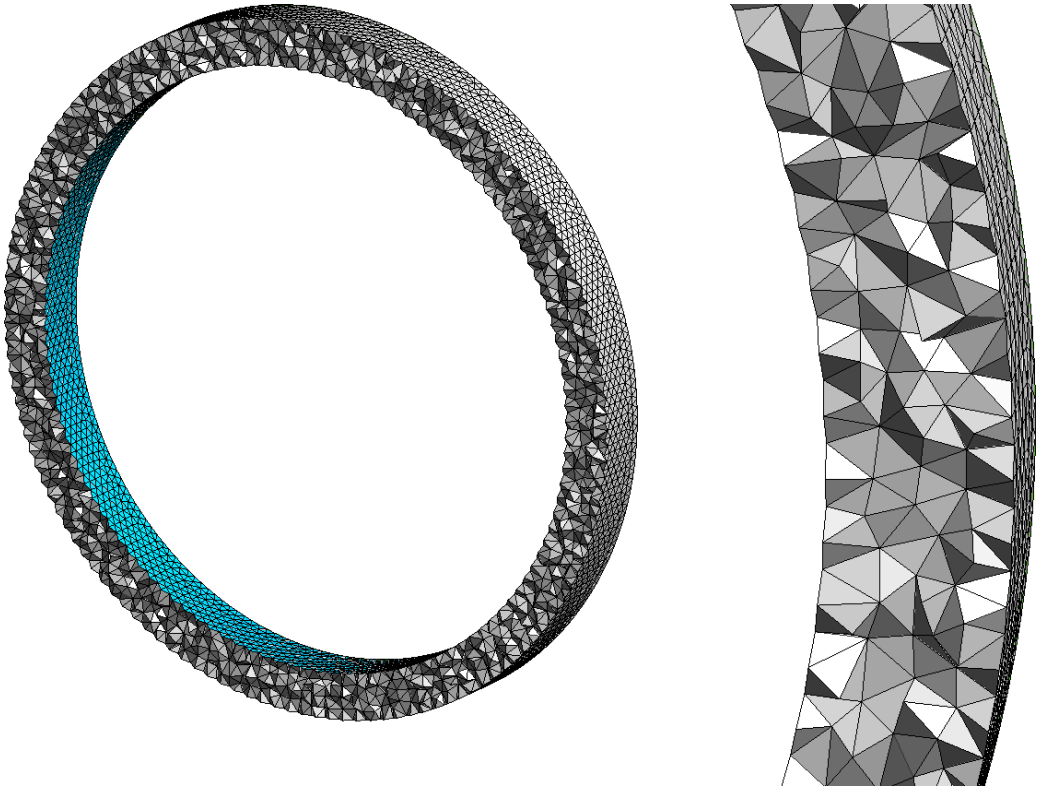


(a) $n = 1$

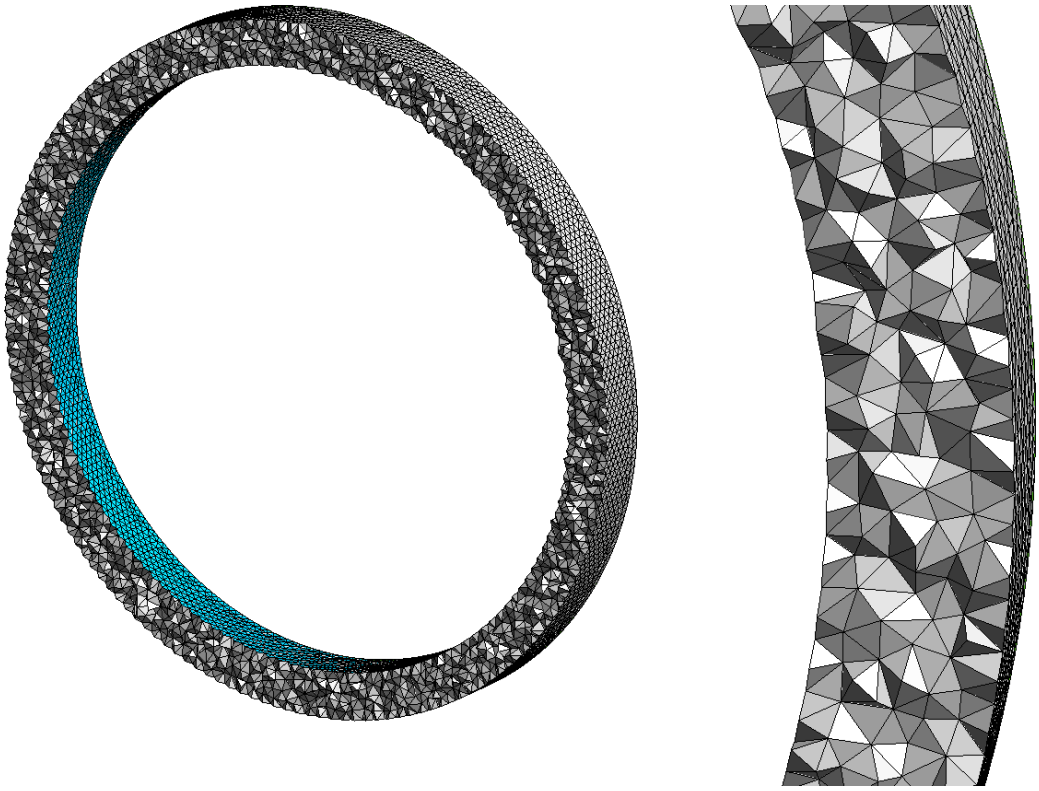


(b) $n = 2$

Figure 7: Parametric grids with n elements through the pipe thickness.



(a) $n = 1$



(b) $n = 2$

Figure 8: Parametric grids with n elements through the pipe thickness.

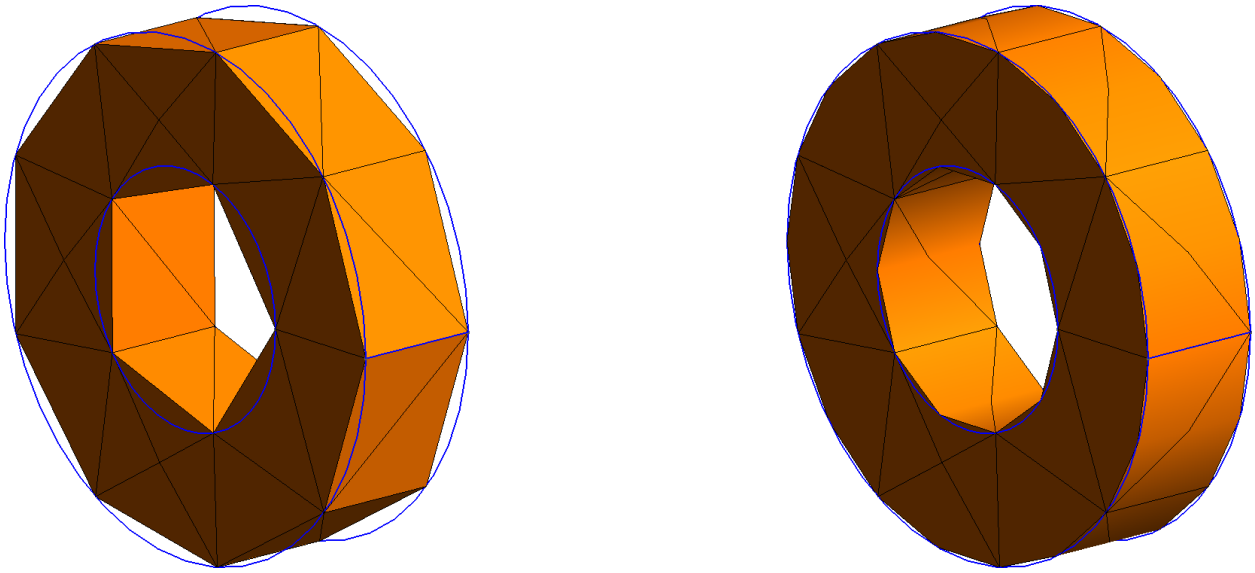


Figure 9: Comparison between first (left) and second-order (right) elements. Even though both of them have the same number of elements, the latter can better represent curved surfaces at the cost of having many more nodes.

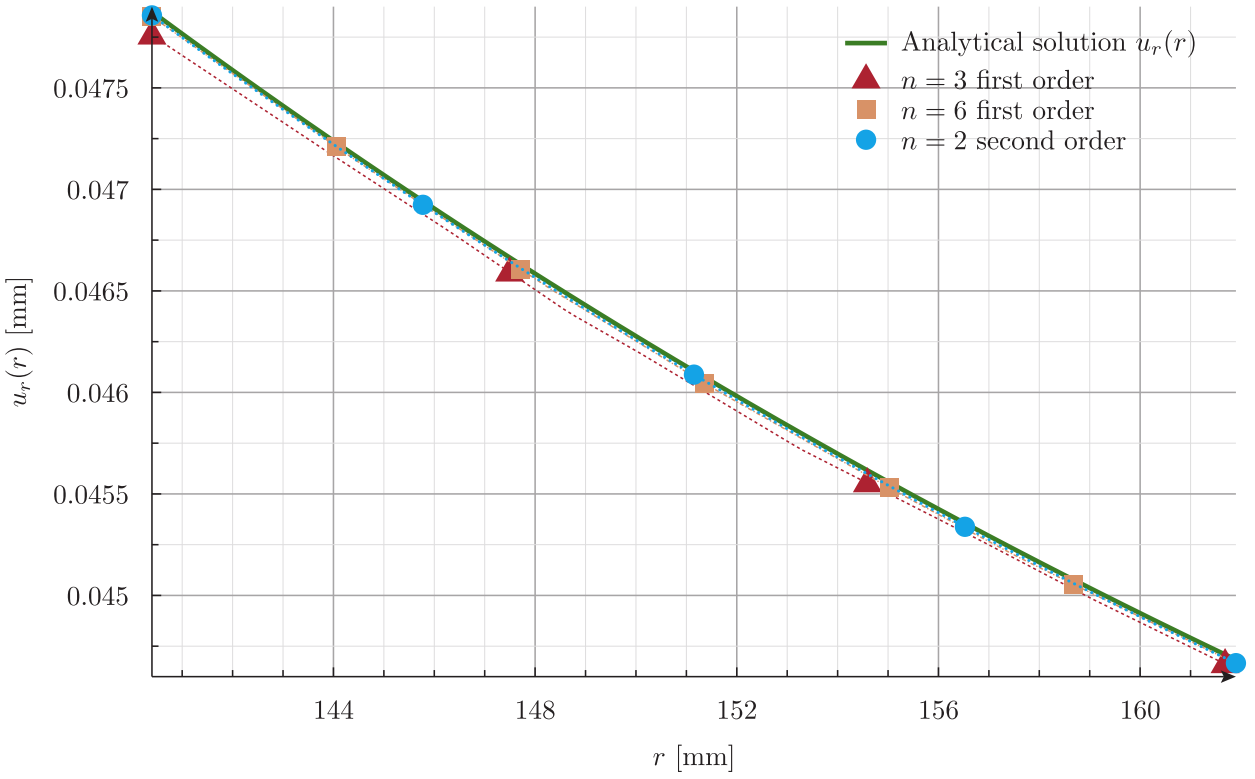


Figure 10: Radial displacements. Note that the FEM solutions are continuous as interpolated by the shape functions.

the displacements—which are discontinuous across element interfaces—there might be slight differences in the actual results. But in essence, we can see in figs. 11, 12a, 12b that again, quadratic elements beat linear ones even with for coarser grids. This is especially true for σ_r that when evaluated at $r = a$ should be $\sigma_r(a) = -p = -10$ MPa according to eq. 2. We need to recall that the finite-element formulation does not strictly fulfill natural (i.e. Neuman or load) boundary conditions. It only can approximate them, with the approximation being better with refined meshes.

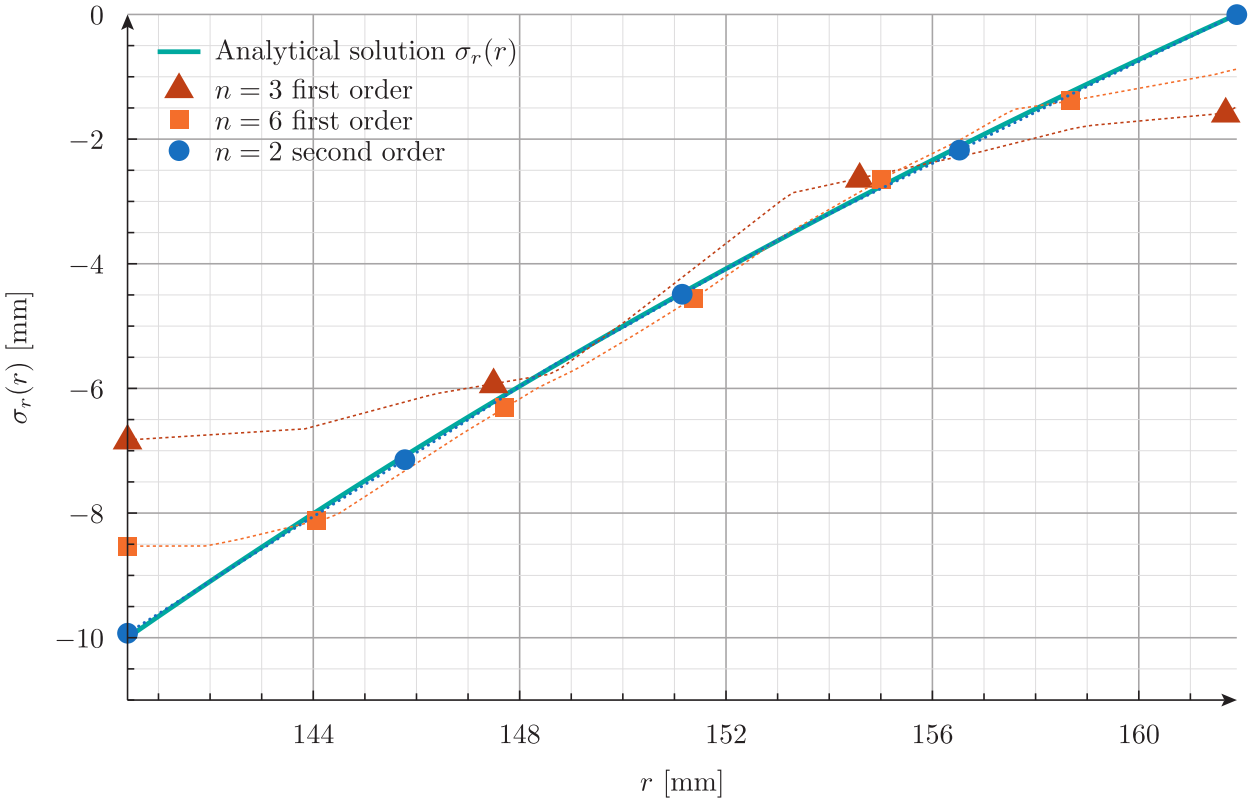
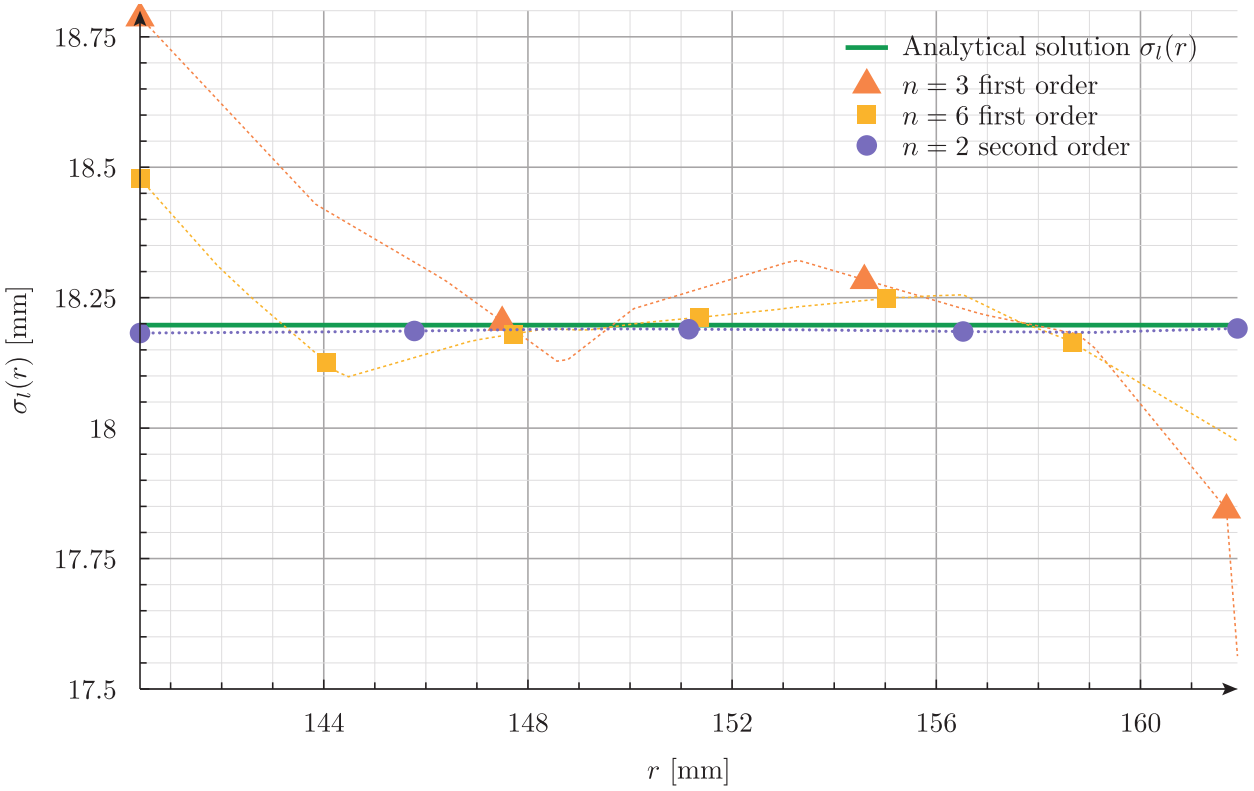


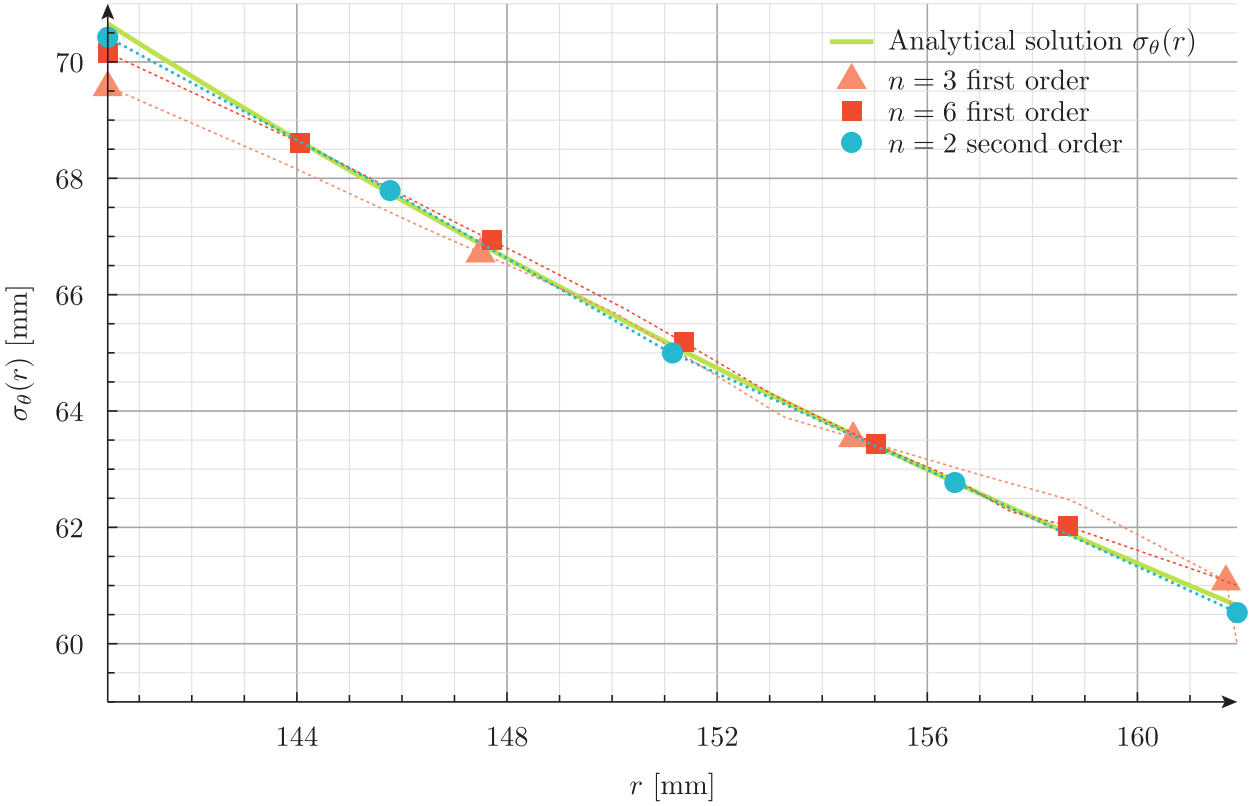
Figure 11: Radial normal stress $\sigma_r(r) = \sigma_3(r)$

The reason of this behavior is that in first-order tetrahedra, the derivatives of the shape functions are uniform in space. Therefore, the stresses (that depend on the gradient of the displacements) are uniform throughout each element. The values of these uniform stresses are somehow an average of the contribution of each of the four nodes of the tetrahedron. For those elements that are in the bulk of the domain, the processes of computing the stresses at the nodes—which depends on the finite-element code implementation but in general involves averaging back the contributions of each of the elements that share a node—gives reasonable results because some of the element would contribute by defect and some by excess to the final nodal average. But those nodes that are located in the internal (external) faces of the pipe will have elements contributing only with values by excess (defect). Therefore, first-order elements will mostly give incorrect stresses at $r = a$ and $r = b$, overestimating them in the former and underestimating them in the latter. This is exactly the behavior of the first-order stresses obtained in figs. 11, 12a, 12b. Second-order elements still have non-uniform spatial derivatives and they can recover (almost[5]) the real values for the stresses at the nodes located in the boundary of the domain.

[5]The rate of convergence of the stresses in the nodes located in the boundary of the domain is linear, while it is quadratic in the bulk.



(a) Longitudinal normal stress $\sigma_l(r) = \sigma_2(r)$



(b) Tangential normal stress $\sigma_\theta(r) = \sigma_1(r)$

Figure 12: Longitudinal and tangential normal stresses

4.4 Linearized stresses

We can now analyze how the results of the linearization of the stresses according to ASME VIII ("ASME Boiler and Pressure Vessel Code, Section VIII, Rules for Construction of Pressure Vessels: Division 2—Alternative Rules" 2010, annex 5-A). Note that Fino allows the location of the SCLs to be arbitrary, in the sense that the end points do not need to be actual nodes of the grids. See sec. B.4 for details of the input files used to compute the linearized stresses, where the definition of the SCLs are given as continuous coordinates and not as references to any particular grid node.

Figures 13a–17b show the membrane M and membrane plus bending MB stresses for different compositions of the stress tensor (Tresca, Von Mises and the three principal stresses) for $n = 1, \ldots, 6$ compared to the analytical solutions obtained in sec. 3. The vertical axes span a $\pm 5\%$ range around the value given by analytical solution. It can be seen that M converges faster than MB.

This last statement can be explained using the uniform-derivative reasoning explained in sec. 4.3. As the membrane stress is essentially an average along the stress classification line, the excess and defect errors committed by the first-order grids in $r = a$ and $r = b$ almost cancel out. Therefore, the error for computing M with linear elements is not that relevant. However, the membrane plus bending stress is a first-order momentum of the stress distribution. This time, the deviations do not cancel out and they both contribute with the same sign to the value of linearized stress. It is this effect that gives such a slow convergence for MB.

4.5 Computational resources

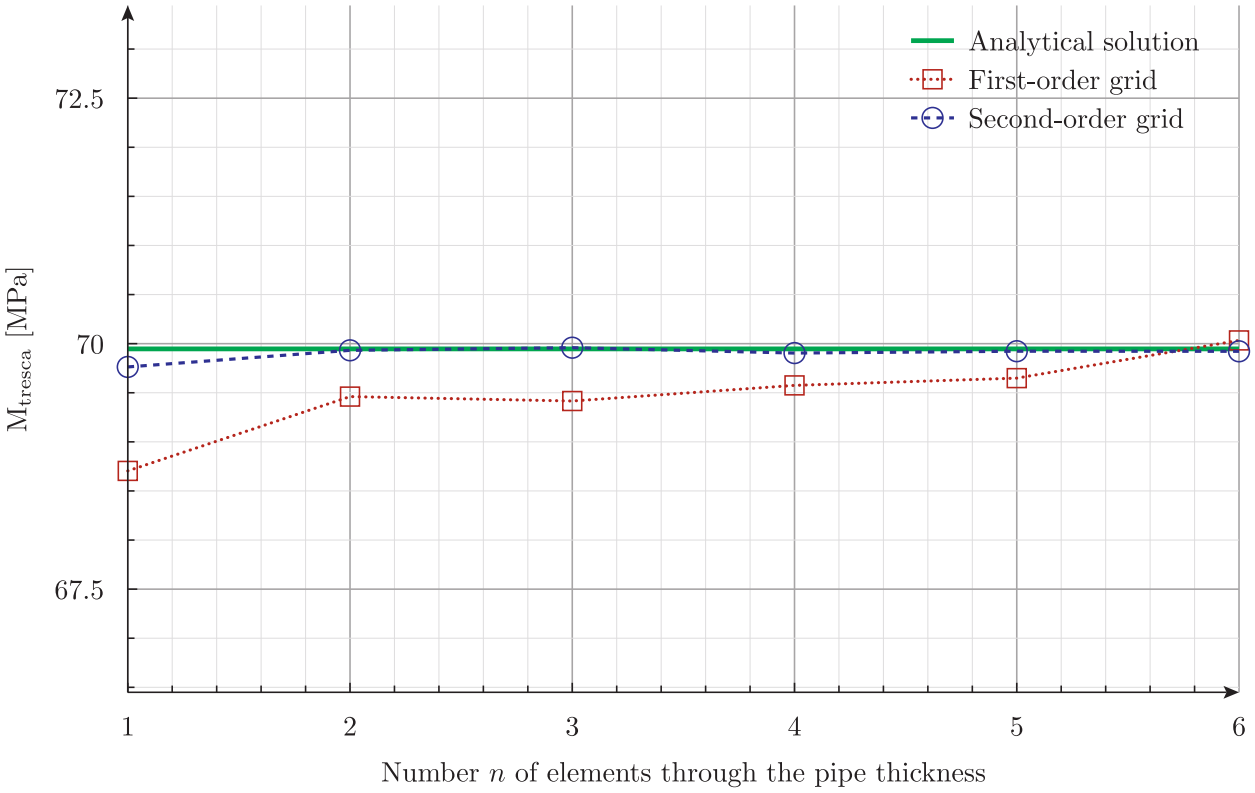
As evident as it it that second-order elements give far superior results than linear ones, any fair analysis needs to compare solutions that involve similar amounts of computational efforts both in random-access memory and number of floating-point operations—otherwise we would end up comparing apples and oranges. That is to say, we might be comparing results from a run that takes one week with a one-second execution and then conclude that the first scheme gives better results than the second. Yes, it does, but it is not a fair comparison.

Roughly speaking, the CPU and memory consumption depend non-linearly on the size of the discretized problem, which is three times the number of *nodes* and not (at least directly) on the number of elements of the mesh. Remember that the latter are the same for a fixed value of n both in first and second-order grids (see again the exaggerated fig. 9). Furthermore, the stiffness matrix of a second-order finite-element formulation is less sparse than the one corresponding to a similar first-order problem (though if now the number of nodes was kept constant to have similar matrix sizes, the former would have less elements so the effort needed to build it would be surprisingly lower for the second-order matrix) rendering the solution of the linear system of equations even more difficult.

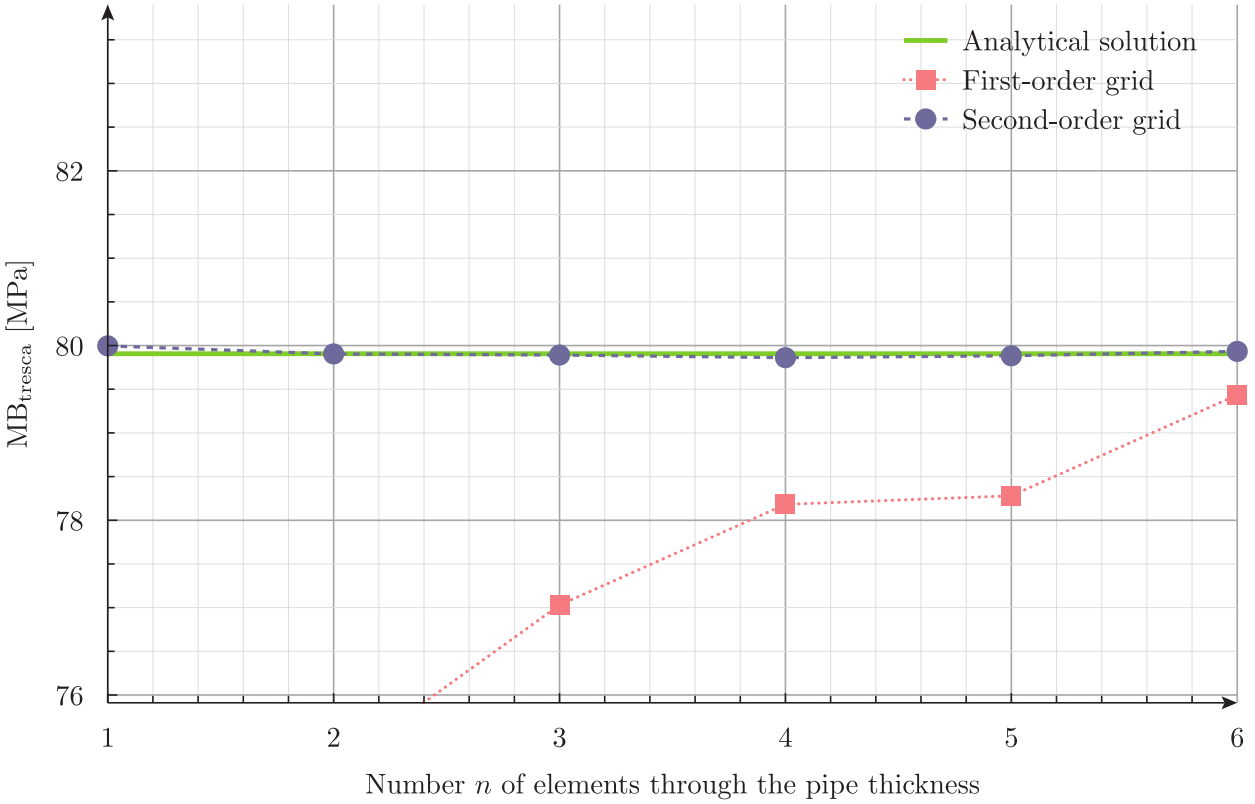
In effect, the number of elements as a function of n is the same for both grids, as it can seen in fig. 18a. But the number of nodes is far higher in the second-order grid for the same n (fig. 18b). This leads to more CPU and memory, as shown in figs. 19a, 19b.

4.6 Error as a function of CPU time

So, on the one hand, for a fixed n, second-order elements give far better results than linear elements. On the other hand, the former consume far more computational resources than the latter. Therefore, in order to compare apples and apples, in principle the comparison of results should be performed by fixing the number of *nodes* (and, again, not the number of *elements*) as discussed above. But given that the problem we are solving does have analytical solutions, we can also analyze the errors associated to each of the finite-element formulations as a function of the computational effort.

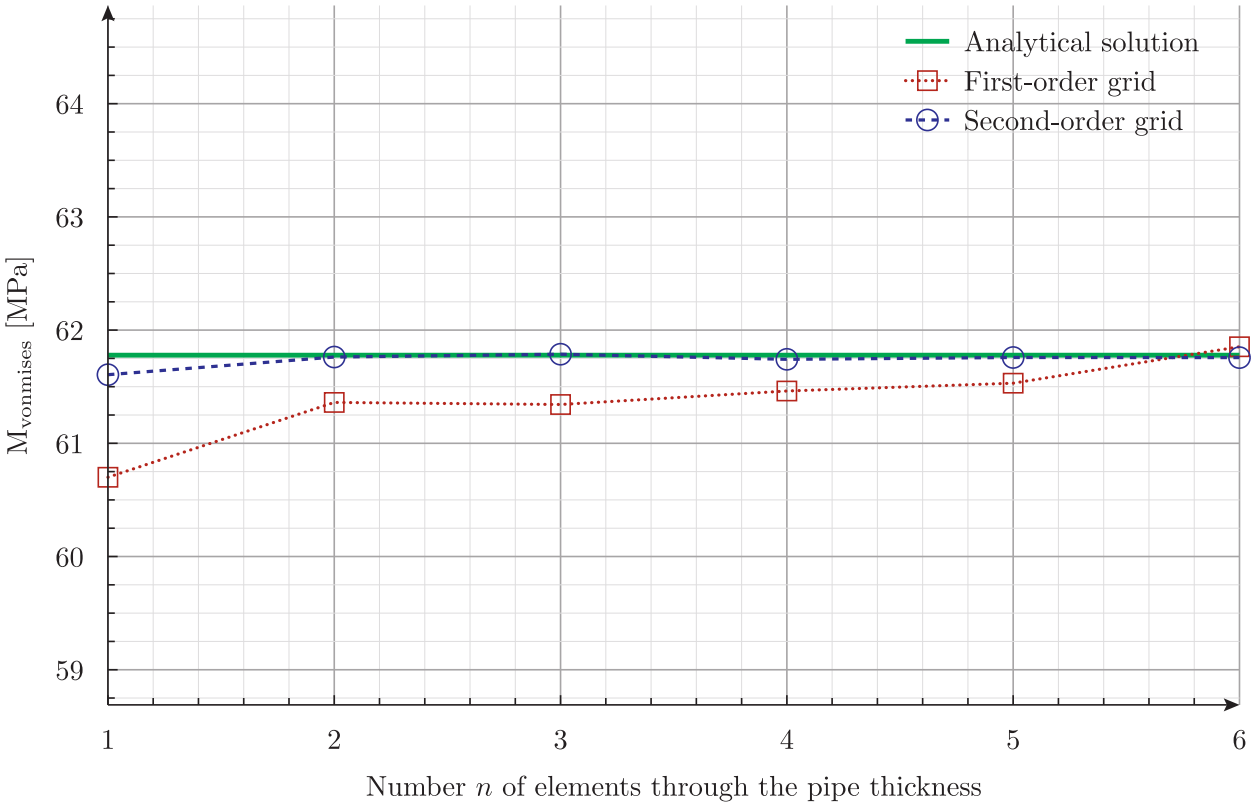


(a) Membrane Tresca stresses

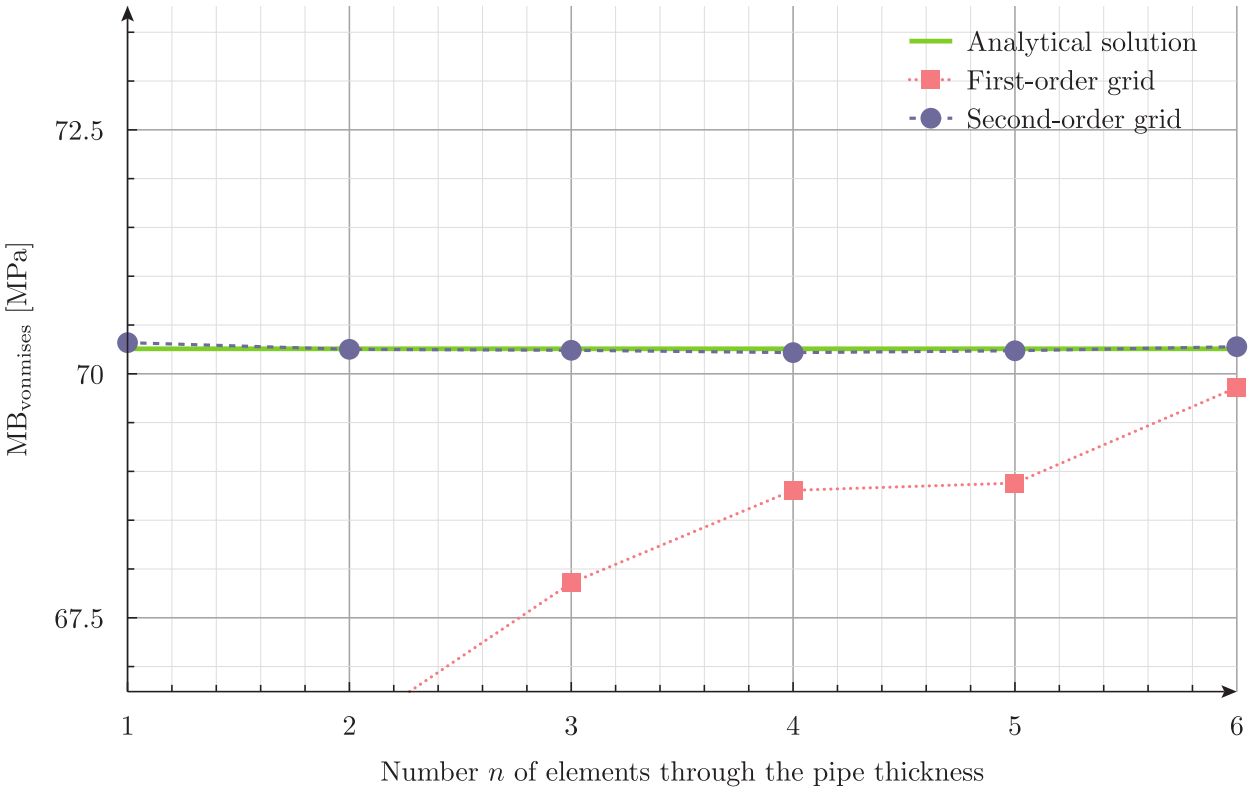


(b) Membrane plus bending Tresca stresses.

Figure 13: Linearized Tresca stresses

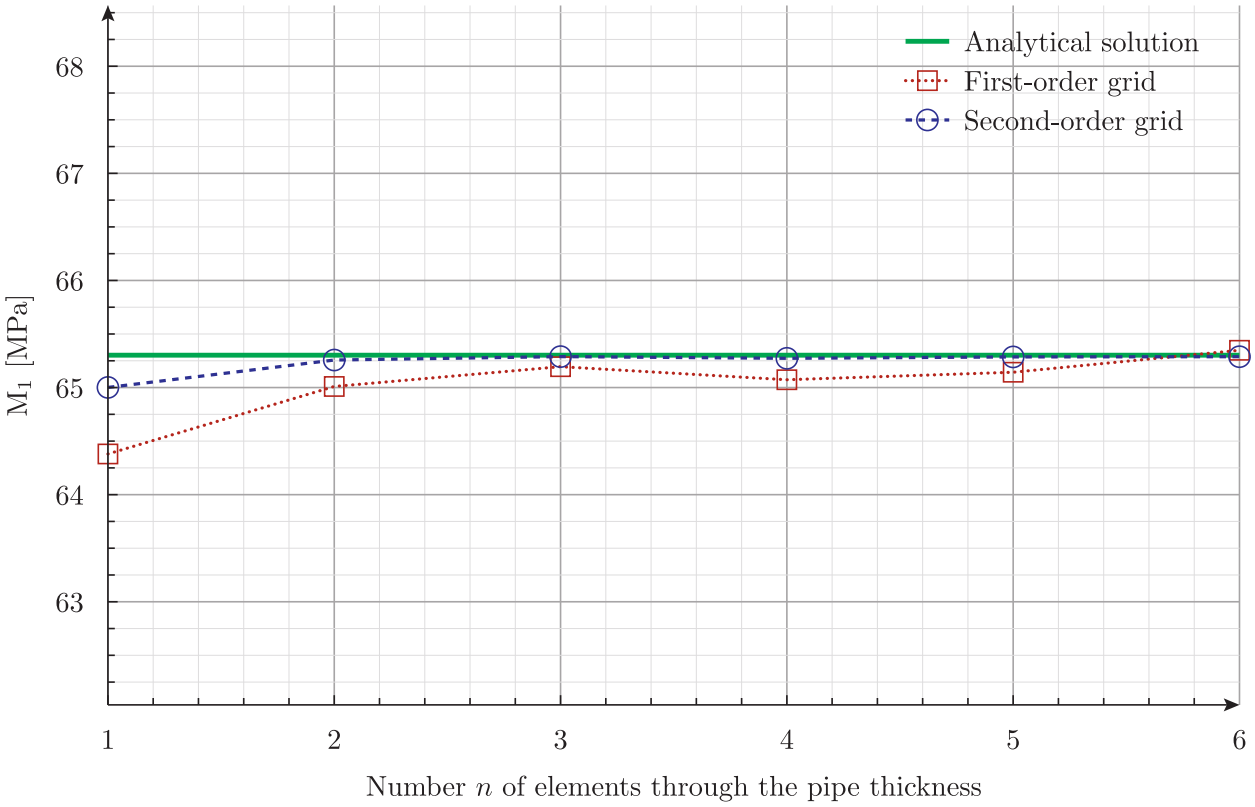


(a) Membrane Von Mises stresses.

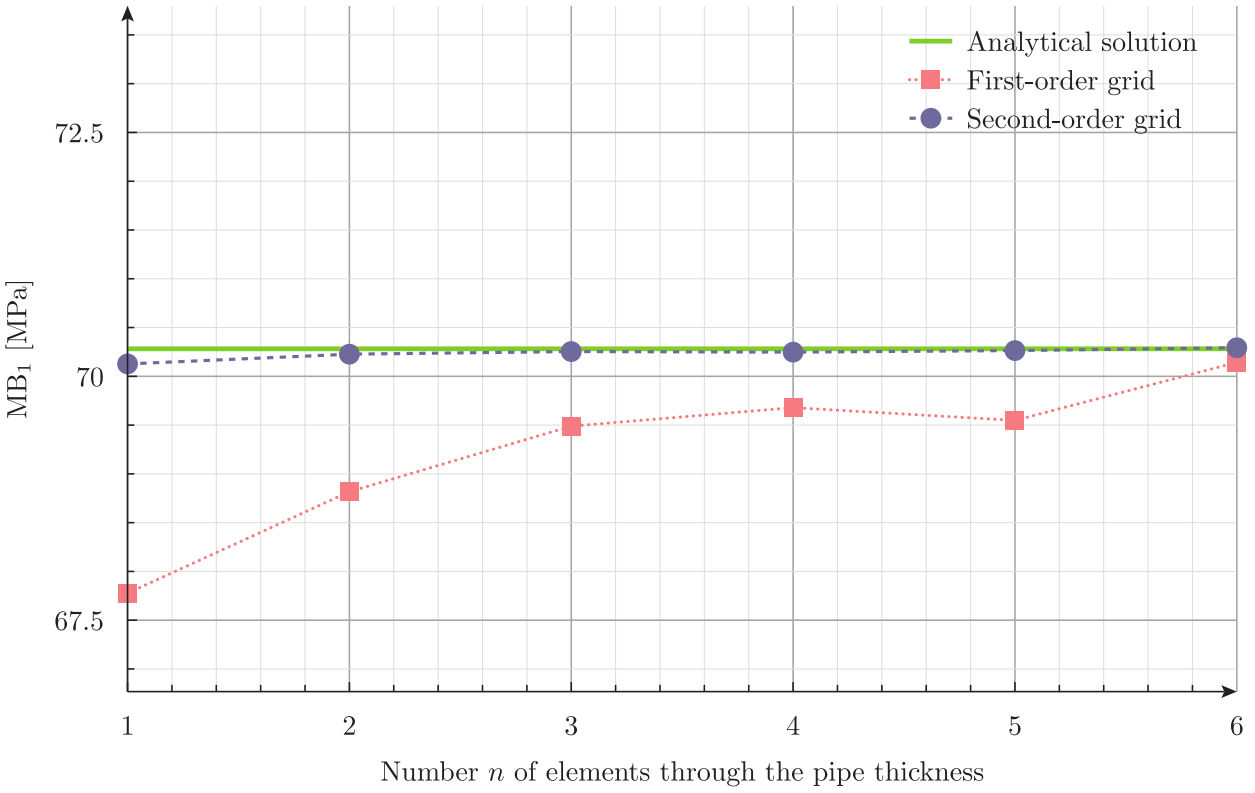


(b) Membrane plus bending Von Mises stresses.

Figure 14: Linearized Von Mises stresses

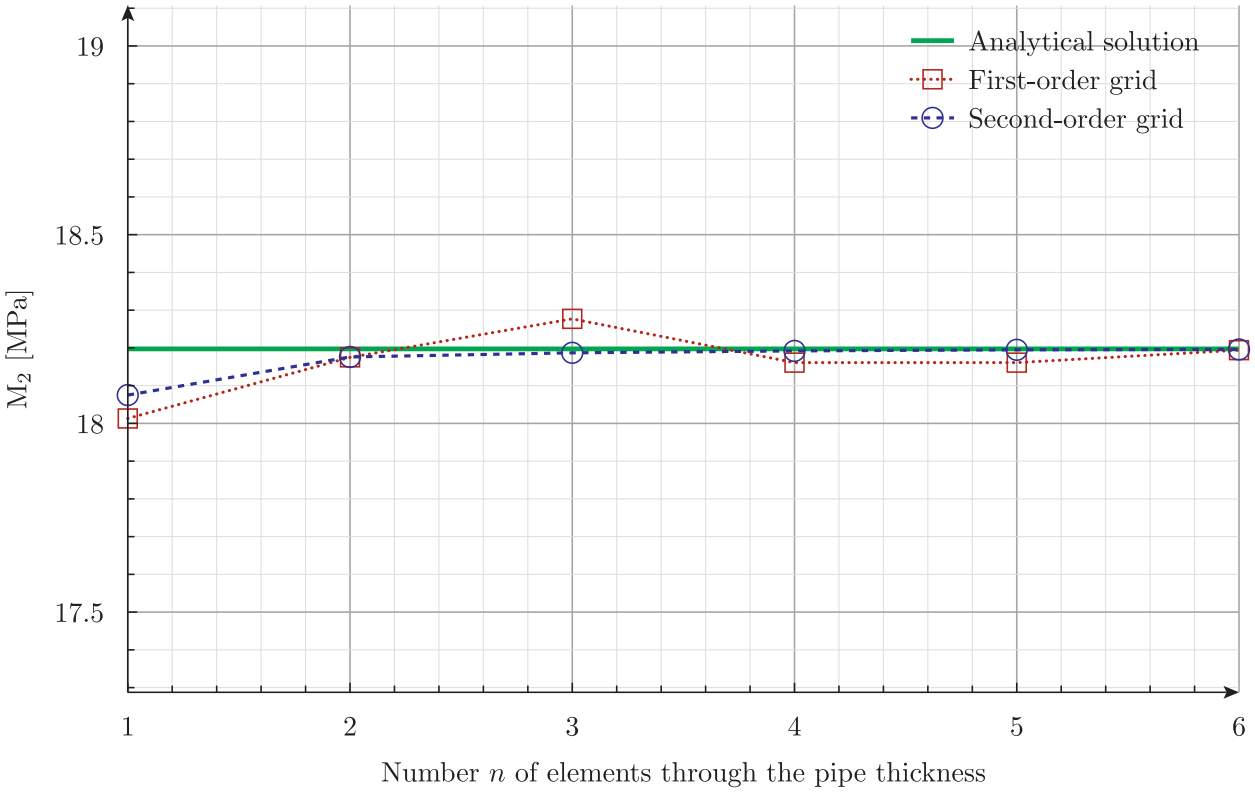


(a) Membrane principal stresses 1.

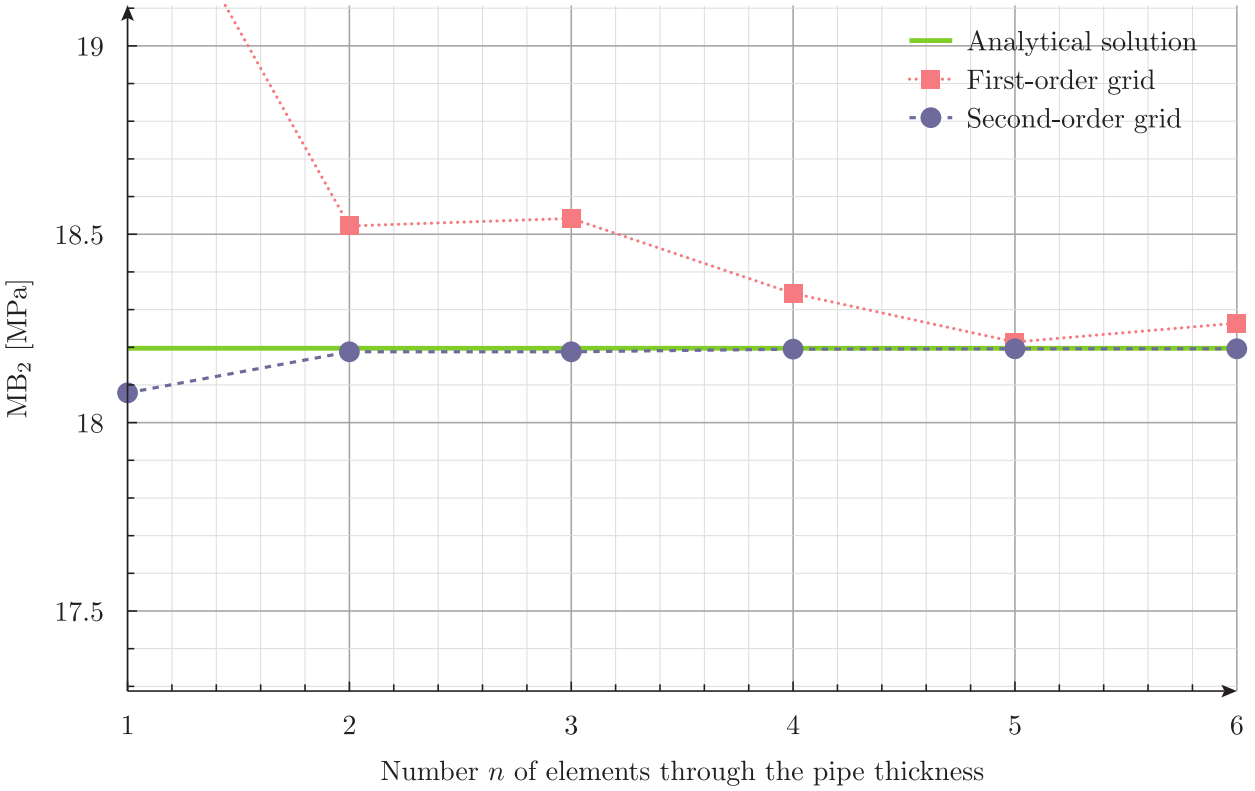


(b) Membrane plus bending principal stresses 1.

Figure 15: Linearized principal stresses 1

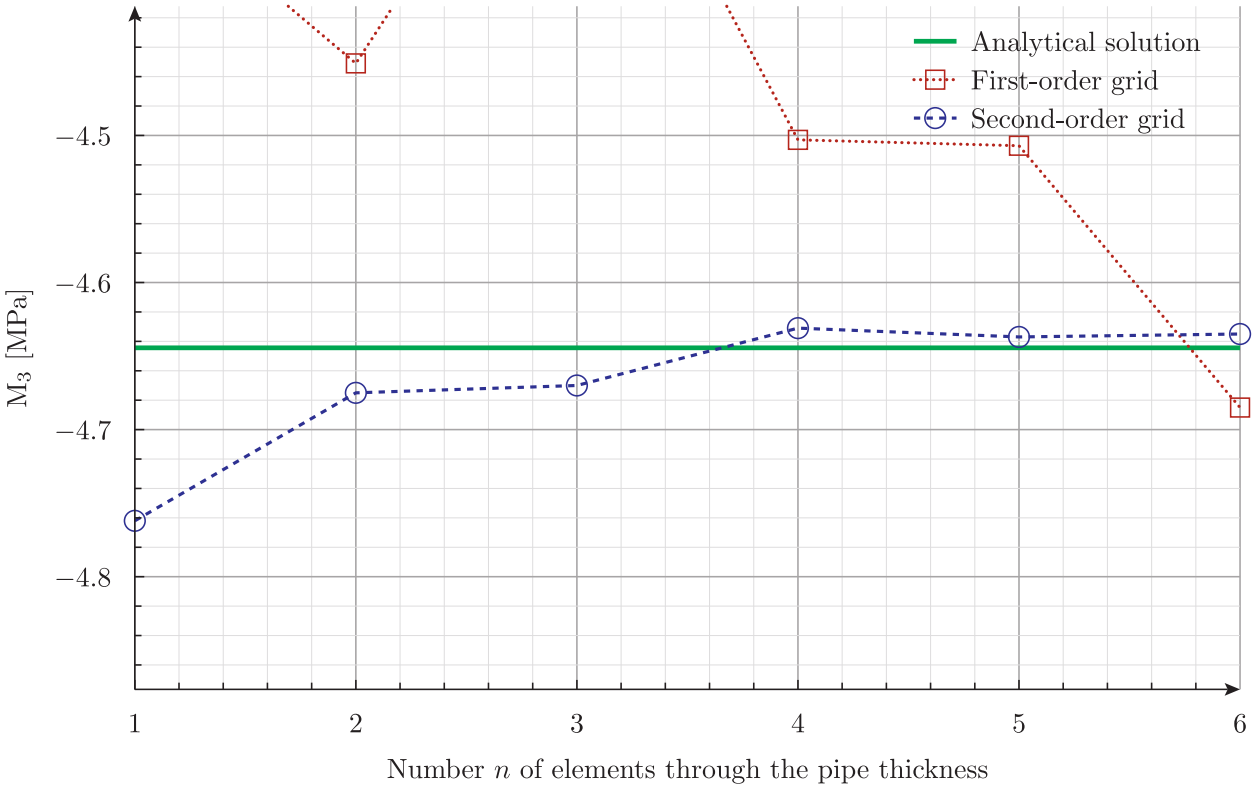


(a) Membrane principal stresses 2.

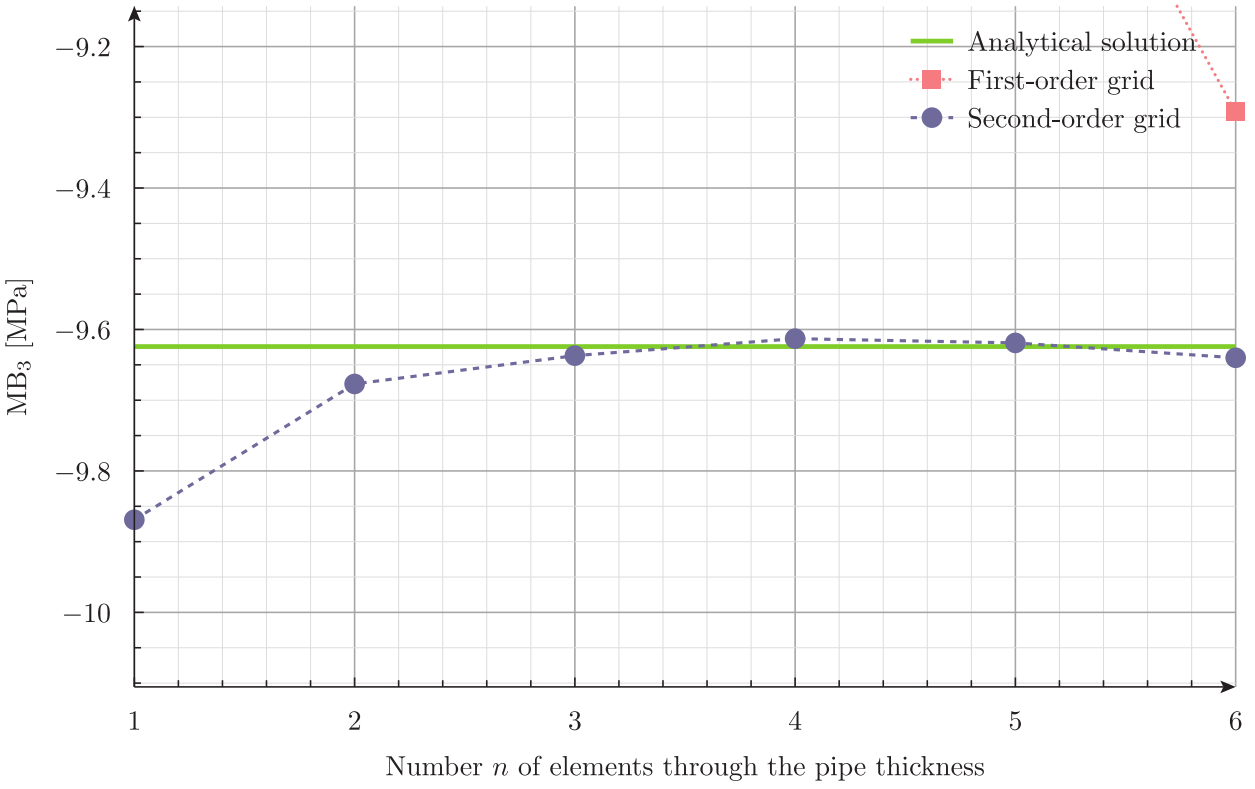


(b) Membrane plus bending principal stresses 2.

Figure 16: Linearized principal stresses 2

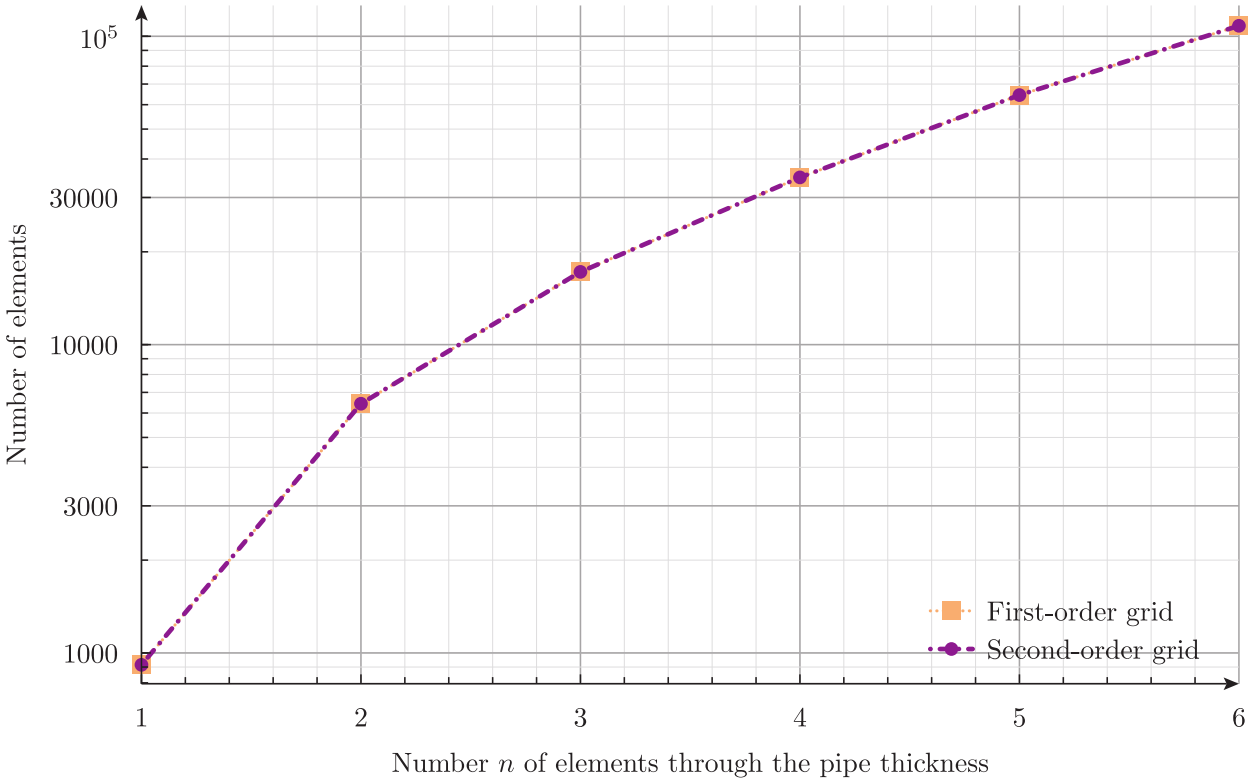


(a) Membrane principal stresses 3.

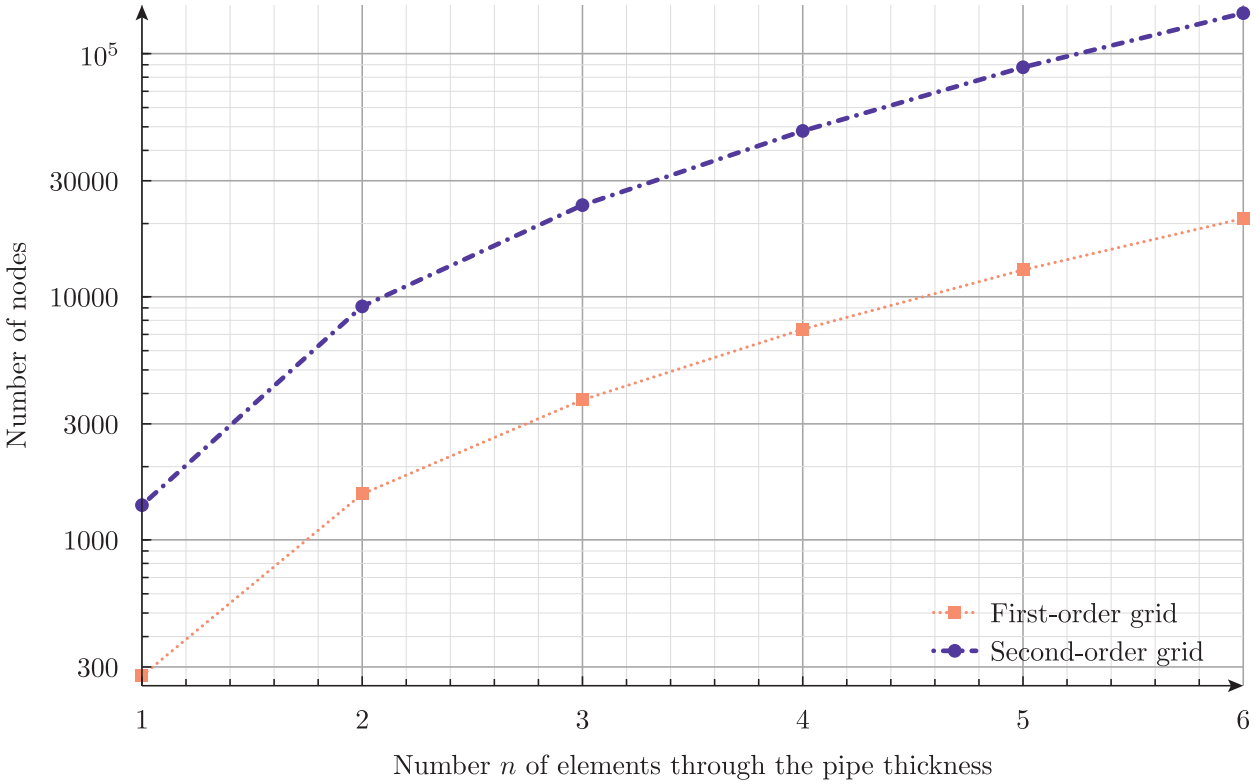


(b) Membrane plus bending principal stresses 3.

Figure 17: Linearized principal stresses 3

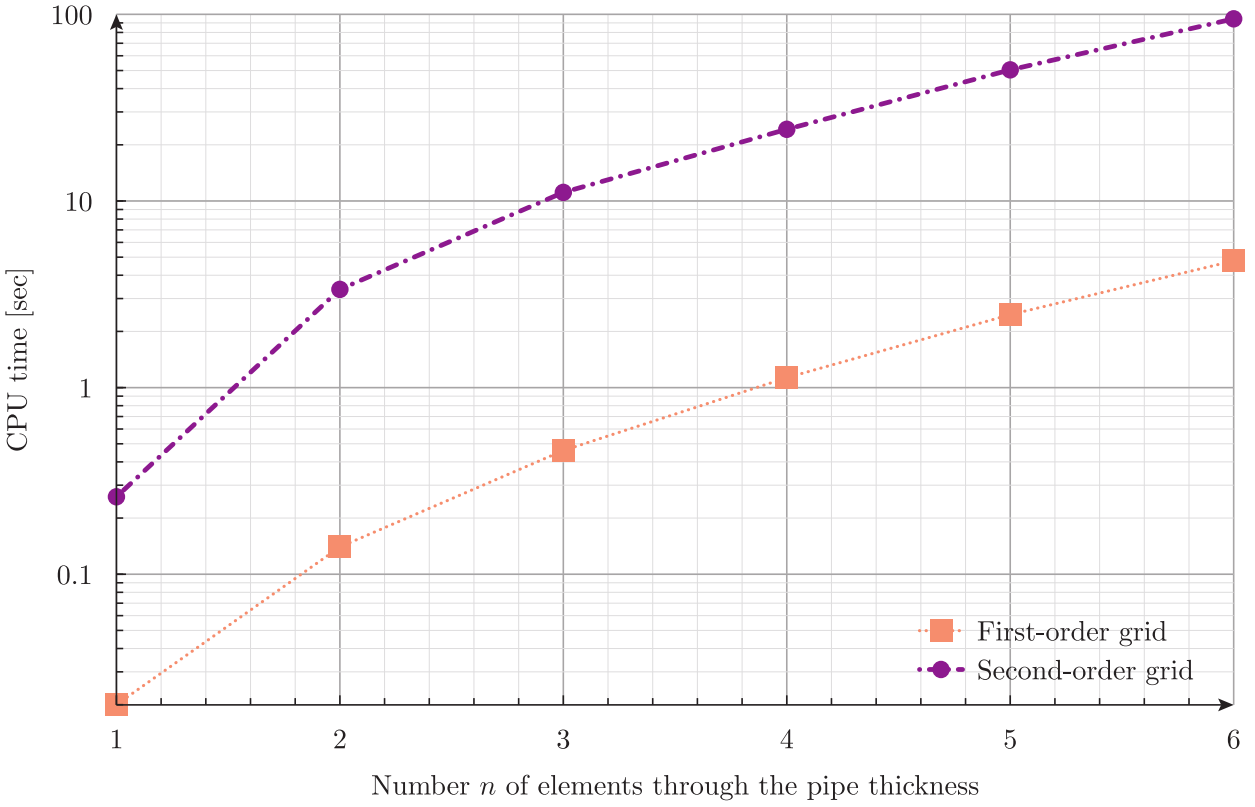


(a) The number of elements vs. n is the same for both grids.

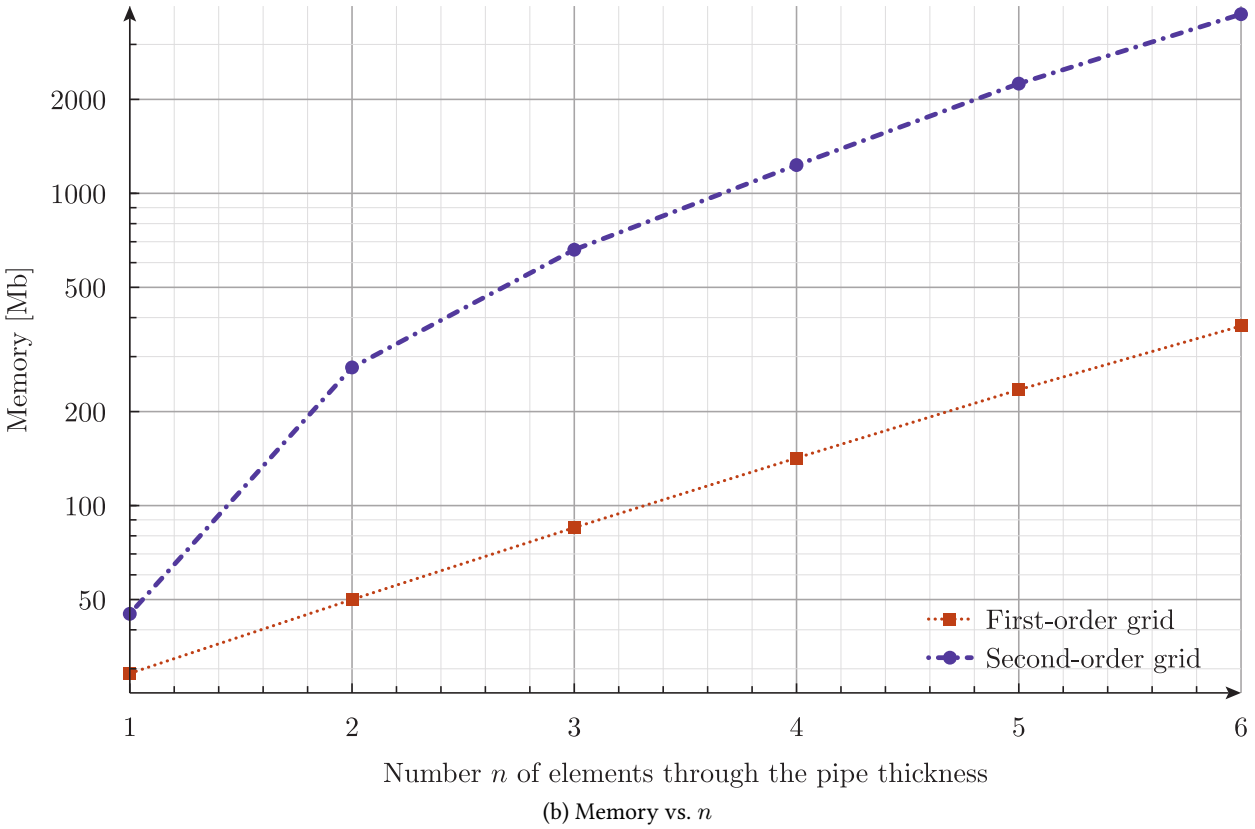


(b) The number of nodes vs. n is higher for the second-order case.

Figure 18: Elements and nodes vs. n



(a) CPU time vs. n



(b) Memory vs. n

Figure 19: CPU time and memory vs. n

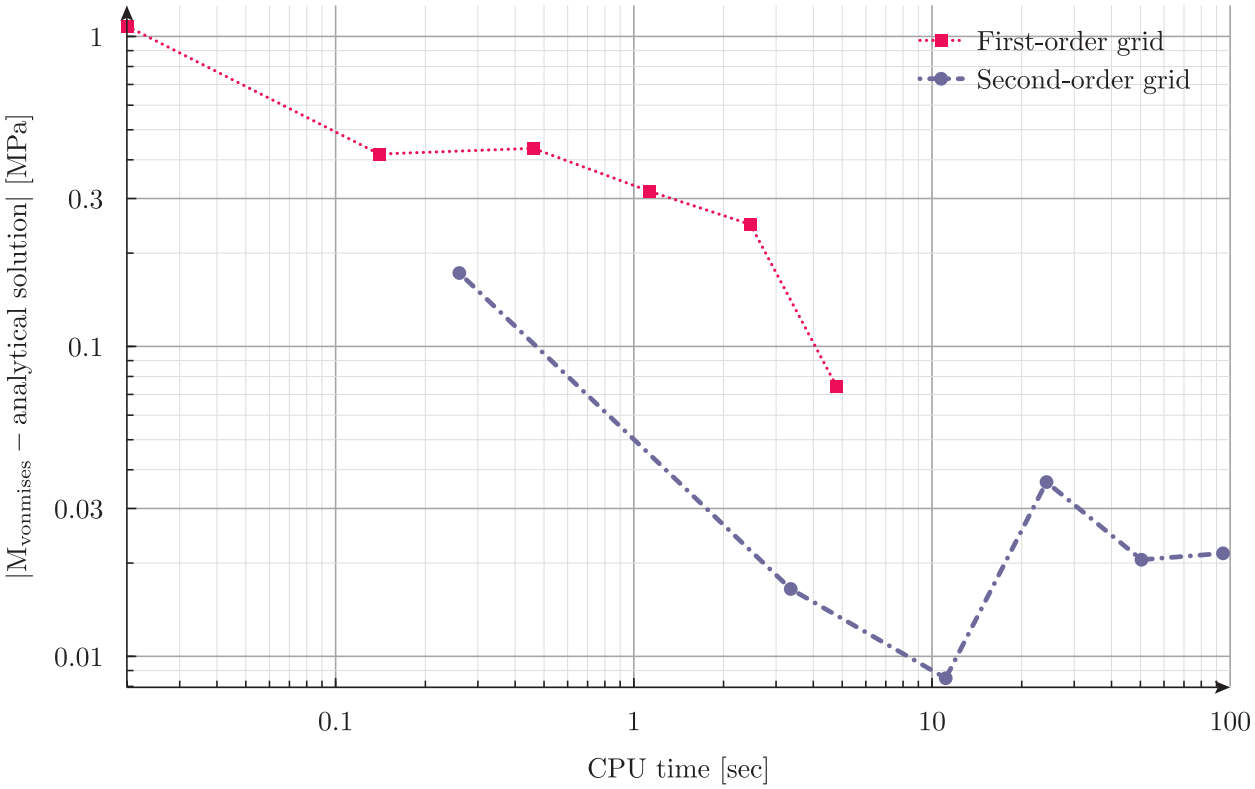
Figures 20a and 20b show a log-log plot of the absolute value of the difference between the linearized stresses computed using Fino and the analytical solution of sec. 3 as a function of the CPU time needed to solve the finite-element formulation. It can be seen that the membrane stress M can be reasonably solved using first-order elements, as the first-order error is comparable to the second-order error for similar amounts of CPU time. However, for the evaluation of the bending stresses MB, linear elements give very poor results when compared to quadratic elements for the same computational effort.

These two figures quantitatively illustrate the effect discussed in sec. 4.4 that the membrane and the membrane plus bending stress are zero and first-order moments respectively. In the former case, the excess and defect errors of the extreme nodes cancel out whereas in the latter case they do not.

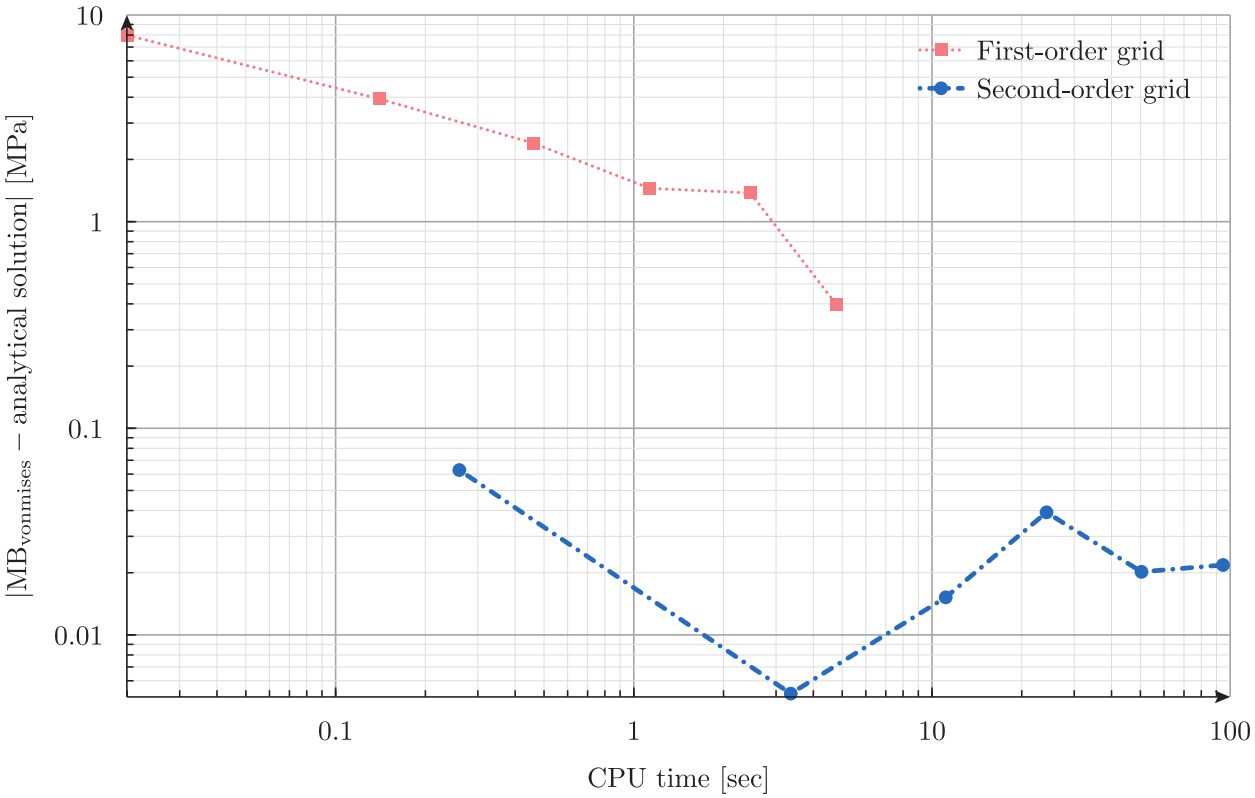
5 Conclusions

There are a few conclusion we can draw out of the results obtained in this report:

- Engineering problems ought not to be solved using black-boxes (i.e. privative software whose source code is not freely available).
- The pressurized infinite pipe has only one independent variable (the radius r) and one primary dependent variable (the radial displacement u_r).
- The problem has analytical solution for the radial displacement and the radial, tangential and axial stresses.
- There are no shear stresses, so these three stresses are also the principal stresses.
- Analytical expressions for the membrane and membrane plus bending stresses along any radial SCL can be obtained.
- The spatial domain can be discretized using linear or higher-order elements. In particular first and second-order elements have been used in this report.
- For the same number of elements, second-order grids need more nodes than linear ones, although they can better represent curved geometries.
- The discretized problem size depends on the number of nodes and not on the number of elements.
- The finite-element results for the displacements are similar to the analytical solution, with second-order grids giving better results for the same number of elements (this is expected as they involved far more nodes).
- The three stress distributions computed with the finite-element give far more reasonable results for the second-order case than for the first-order grid. This is qualitatively explained by the fact that first-order tetrahedra have uniform derivatives and such the elements located in both the external and external faces represent the stresses not at the actual faces but at the barycenter of the elements.
- Membrane stresses converge well for both the first and second-order cases because they represent a zeroth-order moment of the stress distribution and such the excess and defect errors committed by the internal and external elements approximately cancel out.
- Membrane plus bending stresses converge very poorly in first-order grids because the excess and defect errors do not cancel out because it is a first-order moment of the stress distribution.
- The computational effort to solve a given problem, namely the CPU time and the needed RAM depend non-linearly on various factors, but the most important one is the problem size which is three times the number of nodes in the grid.
- The error with respect to the analytical solutions as a function of the CPU time needed to compute the membrane stress is similar for both first and second-order grids. But for the computation of the membrane plus bending stress, first-order grids give very poor results compared to second-order grids for the same CPU time.



(a) Error in the computation of membrane stress.



(b) Error in the computation of membrane plus bending stress.

Figure 20: Errors vs. n

All in all, these are the two main conclusions:

1. Using second-order elements for the computation of ASME linearized stresses is (almost) mandatory.
2. Using just two elements along the pipe thickness is enough to attain convergence of the linearized stresses.

A Multi-freedom boundary condition for radial-only displacements

To set the condition that in the external surfaces all displacement ought to be radially-symmetrical around the barycenter of the surface, a multi-freedom Dirichlet boundary condition needs to be set. That is to say, it is not the value of the actual displacements that is fixed but a certain relationship between the three displacements u, v and w and the spatial coordinates x, y and z.

In effect, let $\mathbf{x_0} = [\pm\ell/2, 0, 0]$ be the location of the barycenter of the face where the condition is to be applied. To force all displacements to be only radial, we require the vectors $\mathbf{u} = [u, v, w]$ and $(\mathbf{x} - \mathbf{x_0})$ to have the same direction. Using the scalar product notation:

$$\mathbf{u} \cdot (\mathbf{x} - \mathbf{x}_0) = \|\mathbf{u}\| \cdot \| (\mathbf{x} - \mathbf{x}_0) \|$$
$$u \cdot (x - x_0) + v \cdot (y - y_0) + w \cdot (z - z_0) = \sqrt{u^2 + v^2 + w^2} \cdot \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$$

and taking into account that

- $u = 0$ because of the null axial displacement condition,
- $\mathbf{x_0} = [\pm\ell/2, 0, 0]$, and
- $x = \pm\ell/2$,

then by squaring both members of the equation, we can simplify this expression to

$$(vy + wz)^2 = (v^2 + w^2) \cdot (y^2 + z^2)$$

Further algebraic manipulation leads to

$$(vy)^2 + 2 \cdot vy \cdot wz + (wz)^2 = (vy)^2 + (vz)^2 + (wy)^2 + (wz)^2$$
$$2 \cdot vy \cdot wz = (vz)^2 + (wy)^2$$
$$0 = (vz - wy)^2$$

from which eq. 6 follows.

B Host, codes, scripts and input files

The complete set of scripts and input files needed to run the cases discussed in this report can be found at
https://bitbucket.org/seamplex/pipe-linearize

The results shown in this report correspond to revision

```
commit 7bf9b6d66d74136bb0a84292ccee0ff9812919d7
Author: gtheler <jeremy@seamplex.com>
Date:   Mon Feb 18 07:18:06 2019 -0300

    plot
```

and were obtained using the free-as-in-free-speech[6] tools Gmsh and Fino using the following versions:

```
$ fino --info
Fino v0.5.235-ga66de30
a free finite-element thermo-mechanical solver

Last commit date   : Sun Feb 17 09:24:33 2019 -0300
Build date         : 2019-02-17 09:25:27
Build architecture : linux-gnu x86_64
Builder            : root@tom
Compiler           : gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
Compiler flags     : -O2
PETSc version      : Petsc Release Version 3.7.5, Jan, 01, 2017
PETSc arch         : x86_64-linux-gnu-real
SLEPc version      : SLEPc Release Version 3.7.3, Sep 29, 2016

---------         --------        --------      ---   ---
wasora v0.5.291-gffe3d44'
wasoras an advanced suite for optimization & reactor analysis

Last commit date   : Wed Feb 13 08:51:54 2019 -0300
Build date         : 2019-02-17 09:25:13
Build architecture : linux-gnu x86_64
Builder            : gtheler@tom
GSL version        : 2.3
SUNDIALs version   : 2.5.0
Readline version   : 7.0
$
$ gmsh -info
Version          : 4.1.0
License          : GNU General Public License
Build OS         : Linux64-sdk
Build date       : 20190113
Build host       : gmsh.info
FLTK version     : 1.4.0
PETSc version    : 3.9.3 (complex arithmtic)
OCC version      : 7.3.1
MED version      : 3.3.1
Packaged by      : gitlab-runner
Web site         : http://gmsh.info
Issue tracker    : https://gitlab.onelab.info/gmsh/gmsh/issues
```

These codes were executed on a host with the following features:

[6] And not just free as in "free beer"

```
Linux tom 4.9.0-8-amd64 #1 SMP Debian 4.9.144-3 (2019-02-02) x86_64 GNU/Linux
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 94
Model name:            Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Stepping:              3
CPU MHz:               2824.340
CPU max MHz:           4000.0000
CPU min MHz:           800.0000
BogoMIPS:              6816.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              8192K
NUMA node0 CPU(s):     0-7
```

The complete case can be run by executing the following script:

```
#!/bin/sh
for i in `cat deps-run`; do
  if [ -z "`which␣$i`" ]; then
    echo "error:␣$i␣not␣installed␣(or␣not␣found)"
    exit 1
  fi
done

./versions.sh > versions.txt
./cpu.sh > cpu.txt

# get problem parameters for maxima from definition in wasora
grep "␣=␣" problem.was | awk '{print $1"␣:␣"$3";"}' | grep -v l > problem.max

# compute analytical results with maxima
maxima -b analytical.max > analytical.txt

# compute theoretical results with wasora
wasora analytical.was | tee analytical.ppl

# run fem parametric cases
fino pipe.fin 1 | tee convergence-1.dat
fino pipe.fin 2 | tee convergence-2.dat
```

Listing 1: run.sh

B.1 Analytical solutions with Maxima

The reference solutions presented in sec. 3 where computed from the theoretical stresses by symbolic integration using Maxima.

```
b : 323.8/2;
a : b-21.5;
n : 4;
E : 200e3;
nu : 0.3;
p : 10;
```

Listing 2: problem.max

```
/* analytically (and symbolically) solve for the linearized stresses in an infinite pipe */
assume(a > 0, b > 0, b > a);

sigmal(r)     := 2*nu*p*a^2/(b^2-a^2);
sigmar(r)     := p*a^2/(b^2-a^2) * (1 - b^2/r^2);
sigmatheta(r) := p*a^2/(b^2-a^2) * (1 + b^2/r^2);

sigma1(r) := sigmatheta(r);
sigma2(r) := sigmal(r);
sigma3(r) := sigmar(r);

M_1 : 1/(b-a)*integrate(sigma1(r), r, a, b);
M_2 : 1/(b-a)*integrate(sigma2(r), r, a, b);
M_3 : 1/(b-a)*integrate(sigma3(r), r, a, b);

M_tresca : max(abs(M_1-M_2), abs(M_2-M_3), abs(M_3-M_1))$
M_vonmises : sqrt(((M_1-M_2)^2 + (M_2-M_3)^2 + (M_3-M_1)^2)/2)$

MB_1 : M_1 + 6/(b-a)^2*integrate(sigma1(r)*((a+b)/2-r), r, a, b);
MB_2 : M_2 + 6/(b-a)^2*integrate(sigma2(r)*((a+b)/2-r), r, a, b);
MB_3 : M_3 + 6/(b-a)^2*integrate(sigma3(r)*((a+b)/2-r), r, a, b);

MB_tresca : max(abs(MB_1-MB_2), abs(MB_2-MB_3), abs(MB_3-MB_1))$
MB_vonmises : sqrt(((MB_1-MB_2)^2 + (MB_2-MB_3)^2 + (MB_3-MB_1)^2)/2)$

batch("problem.max");

float(ev(M_tresca));
float(ev(M_vonmises));
float(ev(M_1));
float(ev(M_2));
float(ev(M_3));

float(ev(MB_tresca));
float(ev(MB_vonmises));
float(ev(MB_1));
float(ev(MB_2));
float(ev(MB_3));
```

Listing 3: analytical.max

The results are illustrated in the following terminal mimic:

```
$ maxima -b analytical.max

Maxima 5.38.1 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.12
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1) batch("analytical.max")

read and interpret file: #p/home/gtheler/seamplex/doc/pipe-linearized/pipe-linearize/analytical.max
```

```
(%i2) assume(a > 0,b > 0,b > a)
(%o2)                         [a > 0, b > 0, b > a]
(%i3) sigmal(r):=(2*nu*p*a^2)/(b^2-a^2)
                                                2
                                           2 nu p a
(%o3)                      sigmal(r) := ---------
                                             2    2
                                            b  - a
(%i4) sigmar(r):=((p*a^2)/(b^2-a^2))*(1-b^2/r^2)
                                            2          2
                                         p a          b
(%o4)                   sigmar(r) := (-------) (1 - --)
                                         2    2        2
                                        b  - a        r
(%i5) sigmatheta(r):=((p*a^2)/(b^2-a^2))*(1+b^2/r^2)
                                              2          2
                                           p a          b
(%o5)                 sigmatheta(r) := (-------) (1 + --)
                                           2    2        2
                                          b  - a        r
(%i6) sigma1(r):=sigmatheta(r)
(%o6)                      sigma1(r) := sigmatheta(r)
(%i7) sigma2(r):=sigmal(r)
(%o7)                       sigma2(r) := sigmal(r)
(%i8) sigma3(r):=sigmar(r)
(%o8)                       sigma3(r) := sigmar(r)
(%i9) M_1:(1/(b-a))*integrate(sigma1(r),r,a,b)
                                       a p
(%o9)                                 -----
                                      b - a
(%i10) M_2:(1/(b-a))*integrate(sigma2(r),r,a,b)
                                       2
                                    2 a  nu p
(%o10)                              ---------
                                      2    2
                                     b  - a
(%i11) M_3:(1/(b-a))*integrate(sigma3(r),r,a,b)
                                         2    2
                               2        b  + a
                              a  (2 b - -------) p
                                           a
(%o11)                        ---------------------
                                         2    2
                               (b - a) (b  - a )
(%i12) M_tresca:max(abs(M_1-M_2),abs(M_2-M_3),abs(M_3-M_1))
(%i13) M_vonmises:sqrt(((M_1-M_2)^2+(M_2-M_3)^2+(M_3-M_1)^2)/2)
(%i14) MB_1:M_1+(6/(b-a)^2)*integrate(sigma1(r)*((a+b)/2-r),r,a,b)
             3                    2    2       2          2
          2  b  + (2 a log(a) + a) b  - a  b   2 b  log(b) + b
       6 a  (------------------------------ - ----------------) p
                            2 a                       2              a p
(%o14) ------------------------------------------------------------ + -----
                                  2   2    2                        b - a
                           (b - a)  (b  - a )
(%i15) MB_2:M_2+(6/(b-a)^2)*integrate(sigma2(r)*((a+b)/2-r),r,a,b)
                                       2
                                    2 a  nu p
(%o15)                              ---------
```

```
                                    2    2
                                   b  - a
(%i16) MB_3:M_3+(6/(b-a)^2)*integrate(sigma3(r)*((a+b)/2-r),r,a,b)
               2          2           3                    2    2
         2  2 b  log(b) + b  + 2 a b   b  + (2 a log(a) + a) b  + a  b
       6 a  (------------------------ - ------------------------------) p
                        2                            2 a
(%o16) ------------------------------------------------------------------
                                  2   2    2
                             (b - a)  (b  - a )
                                                                 2    2
                                                    2           b  + a
                                                   a  (2 b - -------) p
                                                                 a
                                                  + --------------------
                                                                 2    2
                                                    (b - a) (b  - a )
(%i17) MB_tresca:max(abs(MB_1-MB_2),abs(MB_2-MB_3),abs(MB_3-MB_1))
(%i18) MB_vonmises:sqrt(((MB_1-MB_2)^2+(MB_2-MB_3)^2+(MB_3-MB_1)^2)/2)
(%i19) batch("problem.max")

read and interpret file: #p/home/gtheler/seamplex/doc/pipe-linearized/pipe-linearize/problem.max
(%i20) b:323.8/2
(%o20)                               161.9
(%i21) a:b-21.5
(%o21)                               140.4
(%i22) n:4
(%o22)                                 4
(%i23) E:200000.0
(%o23)                            200000.0
(%i24) nu:0.3
(%o24)                               0.3
(%i25) p:10
(%o25)                               10
(%o25)                          problem.max
(%i26) float(ev(M_tresca))
(%o26)                        69.94671856849428
(%i27) float(ev(M_vonmises))
(%o27)                        61.77849925613015
(%i28) float(ev(M_1))
(%o28)                        65.30232558139535
(%i29) float(ev(M_2))
(%o29)                        18.19737977828893
(%i30) float(ev(M_3))
(%o30)                      - 4.644392987098919
(%i31) float(ev(MB_tresca))
(%o31)                        79.90622324679049
(%i32) float(ev(MB_vonmises))
(%o32)                        70.25616539947844
(%i33) float(ev(MB_1))
(%o33)                        70.28207792053773
(%i34) float(ev(MB_2))
(%o34)                        18.19737977828893
(%i35) float(ev(MB_3))
(%o35)                      - 9.624145326252755
(%o35)                        analytical.max
$
```

B.2 Theoretical solutions with wasora

Instead of symbolic integration, adaptive numerical quadrature can be performed easily using wasora (Theler 2016a). First, we put all the problem parameters in a file `problem.was`:

```
# problem parameters for
# 12"—inch schedule 100
b = 323.8/2     # external radius [ mm ]
a = b-21.5      # internal radius [ mm ]
l = 2*(b-a)     # axial length [ mm ]
n = 4           # number of elements through thickness

E = 200e3       # Young modulus [ MPa ]
nu = 0.3        # Poisson's ratio [ non—dimensional ]
p = 10          # internal pressure [ MPa ]


# definition of analytical solutions for comparison from
# <http://eprints.whiterose.ac.uk/110536/1/art%253A10.1007%252Fs00707—016—1762—7.pdf>
ur(x,y,z) := (p*a^2*sqrt(y^2+z^2))/(E*(b^2-a^2)) * ((1-2*nu)*(1+nu) + (1+nu)*b^2/(y^2+z^2))
sigmal(x,y,z)     := 2*nu*p*a^2/(b^2-a^2)
sigmar(x,y,z)     := p*a^2/(b^2-a^2) * (1 - b^2/(y^2+z^2))
sigmatheta(x,y,z) := p*a^2/(b^2-a^2) * (1 + b^2/(y^2+z^2))
```

Listing 4: problem.was

And then include this file from the main one `analytical.was`:

```
# analytical stress linearization of a infinite pipe
# of internal radius a, external radius b and internal pressure p

# problem definition (in a separate file shared by the FEM solution)
INCLUDE problem.was

# principal stresses along the radial coordinate (may be y or z)
sigma1(r) := sigmatheta(0,0,r)
sigma2(r) := sigmal(0,0,r)
sigma3(r) := sigmar(0,0,r)

# computation of main membrane stresses
M_1 = 1/(b-a)*integral(sigma1(r), r, a, b)
M_2 = 1/(b-a)*integral(sigma2(r), r, a, b)
M_3 = 1/(b-a)*integral(sigma3(r), r, a, b)

# von mises and tresca membrane stresses
M_tresca = max(abs(M_1-M_2), abs(M_2-M_3), abs(M_3-M_1))
M_vonmises = sqrt(((M_1-M_2)^2 + (M_2-M_3)^2 + (M_3-M_1)^2)/2)

# computation of membrane plus bending stresses
MB_1 = M_1 + 6/(b-a)^2*integral(sigma1(r)*((a+b)/2-r), r, a, b)
MB_2 = M_2 + 6/(b-a)^2*integral(sigma2(r)*((a+b)/2-r), r, a, b)
MB_3 = M_3 + 6/(b-a)^2*integral(sigma3(r)*((a+b)/2-r), r, a, b)

# von mises and tresca
MB_tresca = max(abs(MB_1-MB_2), abs(MB_2-MB_3), abs(MB_3-MB_1))
MB_vonmises = sqrt(((MB_1-MB_2)^2 + (MB_2-MB_3)^2 + (MB_3-MB_1)^2)/2)

# print results
PRINT "\# analytical results in [ MPa ]"
PRINT "M_tresca     = " M_tresca
PRINT "M_vonmises   = " M_vonmises
PRINT "M_1          = " M_1
PRINT "M_2          = " M_2
PRINT "M_3          = " M_3
PRINT
```

```
PRINT "MB_tresca␣␣␣=␣" MB_tresca
PRINT "MB_vonmises␣=␣" MB_vonmises
PRINT "MB_1␣␣␣␣␣␣␣␣=␣" MB_1
PRINT "MB_2␣␣␣␣␣␣␣␣=␣" MB_2
PRINT "MB_3␣␣␣␣␣␣␣␣=␣" MB_3
```

Listing 5: analytical.was

```
$ wasora analytical.was
# analytical results in [ MPa ]
M_tresca    =   69.9467
M_vonmises  =   61.7785
M_1         =   65.3023
M_2         =   18.1974
M_3         =   -4.64439

MB_tresca   =   79.9062
MB_vonmises =   70.2562
MB_1        =   70.2821
MB_2        =   18.1974
MB_3        =   -9.62415
$
```

B.3 Geometry and mesh generation with Gmsh

```
SetFactory("OpenCASCADE");

l = 2*(b-a);


Cylinder(1) = {-l/2, 0, 0,  +l, 0, 0,  b};
Cylinder(2) = {-l/2, 0, 0,  +l, 0, 0,  a};
BooleanDifference(3) = {Volume{1};Delete;}{Volume{2};Delete;};

Physical Volume("pipe") = {3};
s() = Boundary{ Volume{3}; };
Physical Surface("symmetry") = {s(1), s(2)};
Physical Surface("pressure") = {s(3)};

Mesh.CharacteristicLengthMax = (b-a)/n;

Mesh.Algorithm = 6;
Mesh.ElementOrder = order;
```

Listing 6: pipe.geo.m4

B.4 Parametric finite-element analysis with Fino

```
# convergence study of linearized stresses in an infinite pipe
# with respect to the number of elements in the pipe thickness

# read problem parameters
INCLUDE problem.was

# swipe the number of elements through the pipe thickness from 1 to 6
PARAMETRIC n MIN 1 MAX 6 STEP 1

# prepare gmsh .geo file from m4 template
```

```
OUTPUT_FILE geo pipe-$1-%g.geo n DO_NOT_OPEN
M4 INPUT_FILE_PATH pipe.geo.m4  {
   OUTPUT_FILE geo
   MACRO a a
   MACRO b b
   MACRO n n
   MACRO order $1
}

# call gmsh
SHELL "gmsh␣-v␣0␣-3␣pipe-$1-%g.geo" n

# read the just-generated mesh
INPUT_FILE msh pipe-$1-%g.msh n DO_NOT_OPEN
MESH FILE msh DIMENSIONS 3

# set boundary conditions:
#  * internal pressure (easy)
PHYSICAL_ENTITY pressure  BC P=p
#  * null axial displacement (u=0) and only radial in the zx plane (multifreedom)
PHYSICAL_ENTITY symmetry  BC u=0    0=v*(z-0)-w*(y-0)

# solve problem
FINO_STEP

# write distribution of results in gmsh format (optional)
OUTPUT_FILE out pipe-out-$1-%g.msh n
MESH_POST FILE out VECTOR u v w sigma1 sigma2 sigma3 sigmax sigmay sigmaz tauxy tauyz tauzx

OUTPUT_FILE dist pipe-dist-$1-%g.dat n
PRINT_FUNCTION v sigma1 sigma2 sigma3 FILE dist  MIN 0 a 0 MAX 0 b 0 NSTEPS 1 100 1

# compute linearized stresses with different combinations
FINO_LINEARIZE START_POINT 0 a 0 END_POINT 0 b 0 TOTAL tresca     M M_tresca   MB MB_tresca
FINO_LINEARIZE START_POINT 0 a 0 END_POINT 0 b 0 TOTAL vonmises   M M_vonmises MB MB_vonmises
FINO_LINEARIZE START_POINT 0 a 0 END_POINT 0 b 0 TOTAL principal1 M M_1        MB MB_1
FINO_LINEARIZE START_POINT 0 a 0 END_POINT 0 b 0 TOTAL principal2 M M_2        MB MB_2
FINO_LINEARIZE START_POINT 0 a 0 END_POINT 0 b 0 TOTAL principal3 M M_3        MB MB_3

# write convergence in standard output
IF in_outer_initial
  PRINT "\#␣FEM␣with␣order␣$1␣elements"
ENDIF
PRINT {
   %g n
 %.3f M_tresca  M_vonmises  M_1  M_2  M_3
      MB_tresca MB_vonmises MB_1 MB_2 MB_3
   %g nodes
      elements
 %.2f time_cpu_total
 %.0f  memory/(1024^2)
}

# draw a picture of the mesh showing how many elements are there
SHELL "m4␣-Dn=%g␣-Dorder=$1␣pipe-draw-mesh.geo.m4␣>␣pipe-draw-mesh-$1-%g.geo" n n
```

Listing 7: pipe.fin

References

"ASME Boiler and Pressure Vessel Code, Section VIII, Rules for Construction of Pressure Vessels: Division 2—Alternative Rules." 2010. American Society of Mechanical Engineers.

Geuzaine, C., and J. F. Remacle. 2009. "Gmsh: A Three-Dimensional Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities." *International Journal for Numerical Methods in Engineering* 11 (79): 1309–31.

Theler, J. 2016a. "Description of the Computational Tool Wasora." Technical Description SP-WA-15-TD-9E3D. Seamplex.

———. 2016b. "On the Design Basis of a New Core-Level Neutronic Code Written from Scratch." Technical Report SP-MI-14-AR-5B44. Seamplex.

"Welded and Seamless Wrought Steel Pipe." 2004. ASME B36.10M. American Society of Mechanical Engineers.