Thermo-elastic expansion of finite cylinders

Fino benchmark problem series

Number: SP-FI-17-BM-5460
Rev.: A
Date: 12-Apr-2017
Hash: `c17ba70`
Document type: Benchmark problem
Pages: 15

Author: Jeremy Theler `jeremy@seamplex.com`

Abstract

The problem of the thermal expansion of a cylinder under a know temperature distribution (given as an algebraic expression of x, y and z) is solved with the free and open source tool fino using a finite-element formulation and the results are compared to those reported by Veeder in 1967 using a power series approach. The original problem was posed to study deformations and stresses on the uranium pellets located inside nuclear reactor fuel elements under heavy temperature gradients, which continues to be an interesting problem even nowadays. Not only does this report solve again the problem and compare the results with the original ones, but it also illustrates how the fino back-end works and what its distinctive features are.

fino
a free finite-element solver

Keywords

Fino, benchmark, Veeder, thermal expansion, thermo-elastic problem

Revision history

Rev.	Date	Author	
A	12-Apr-2017	gtheler	First issue

Contents

1 Introduction

Fifty years ago, J. Veeder from AECL (now CANDU Inc.) tackled a problem that is still important even nowadays: that of the thermal expansion of uranium pellets within nuclear reactor fuel element bundles [1]. Leaving aside the neutronic and thermalhydraulic-related issues, the original problem proposed by Veeder poses an interesting case worth of study in order to analyze how numerical computer codes cope with thermo-elastic expansion. In particular, this report shows how this problem can be solved with the free ("Free" both as in "free speech" and in "free beer.") and open source finite-element analysis tool fino—developed by Seamplex—with a Gmsh-generated unstructured grid. More information about the programs, including documentation and downloads can be found at

(a) `https://www.seamplex.com/fino` (b) `http://gmsh.info/`

In particular, this report

- describes again the original problem, including the boundary conditions,
- shows how a proper geometry and grid can be built using Gmsh,
- solves the problem for a particular set of input parameters (geometry & grid coarseness) using fino,
- performs a parametric analysis to show the convergence with respect to the mesh size for both first and second-order elements, and
- discusses both the results and the methodology.

Our main objective is closer to illustrating the features that the finite-element back-end fino can provide—for example to web-based front-ends like CAEplex[1]—than to merely benchmark a numerical solution against a fifty-year-old second-order polynomial of two variables. It should be noted that, following Seamplex' principles,[2] only free and open source software was used in preparing this report.

2 Problem

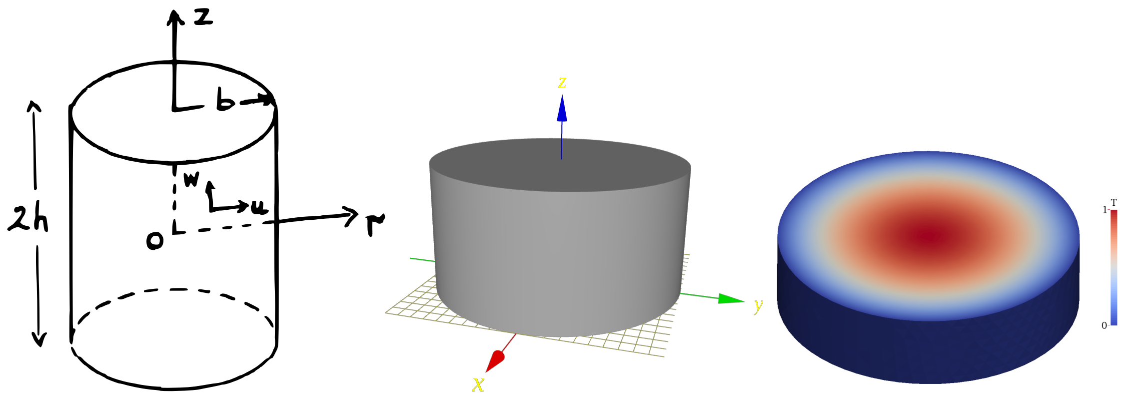
We can see the geometry of the problem to be solved in figure 1, which is the actual original drawing published by Veeder. It consists of finding the displacement fields n the x, y and z directions—namely $u(x,y,z)$, $v(x,y,z)$ and $w(x,y,z)$—in a cylinder of radius b and height $2h$ centered at the origin of an x-y-z system with the cylinder axis along the z direction but otherwise free to expand in any direction, subject to a non-uniform temperature distribution. The original problem is stated in cylindrical coordinates. Without loosing generality, we take our positive x axis as coincident with the original r direction. Moreover, given that the problem is symmetric with respect to the x-y plane we focus only on the $z > 0$ half-space (figure 1b).

The cylinder is subject to a temperature distribution that varies radially on space as

$$
\begin{aligned}
T(r,z) &= T_0 \cdot \left[1 - \left(\frac{r}{b} \right)^2 \right] \\
T(x,y,z) &= T_0 \cdot \left[1 - \left(\frac{x^2+y^2}{b^2} \right) \right]
\end{aligned}
$$

[1] `https://www.caeplex.com`

[2] `https://www.seamplex.com/principles.html`



(a) Original figure from reference [1].

(b) Symmetry over the x-y plane

(c) Temperature distribution

Figure 1: The problem to be solved: original geometry, x-y-z coordinates with symmetry and temperature distributions.

which is depicted in figure 1c, and all the surfaces are free to expand. The fact that this temperature distribution is symmetric with respect to the x-y plane and that the cylinder is centered at the origin but otherwise free to expand in any direction implies that the Dirichlet boundary conditions are

$$u(0,0,0) = 0 \tag{1}$$
$$v(0,0,0) = 0 \tag{2}$$
$$w(0,0,0) = 0 \tag{3}$$
$$w(x,y,0) = 0 \tag{4}$$

i.e., the origin should be fixed and the base surface in the x-y plane should not have any displacement in the z direction. The external faces should be subject to homogeneous Neumann boundary conditions.

It should be noted that these displacement boundary conditions are not enough to restrict all the rigid body motions because rotations around the z axis can still occur. If displacement in the three directions was zero for the base surface—effectively removing rotations—the obtained results would not be comparable to the original ones. It is a feature of fino (actually of the PETSc library [2, 3] it is linked against) that it can obtain a solution even if the problem is not well-defined in the classical way. Should we want to effectively avoid the cylinder from rotating around the z axis, we may add the extra condition

$$v(0,y,0) = 0 \tag{5}$$

The Young modulus E is not needed, as the problem is homogeneous and depends linearly on this parameter. The problem also asks just for the displacements only and not for the stresses, so any value of E that preserves the stability of the numerical formulation may be used. The expansion coefficient α and the temperature increment T_0 are used to nondimensionalize the reported results, so again any arbitrary values may be used. In the same sense, the individual values of h and b are not important individually but as the ratio h/b. The Poisson ration ν, however, appears as a non-linear parameter so its value is also needed.

3 Solution

The original reference [1] solves the problem for different values of the ratio h/b and the Poisson ratio ν. It uses a power expansion of both $\mathring{u}$ and $\mathring{v}$ (the accent $\circ$ means "original") in terms of the nondimensional radial and axial positions ρ and ξ respectively. Given the number of equations involved (i.e. the boundary conditions and a functional minimizing total strain energy) only the first six terms are retained:

$$\mathring{u}(\rho,\xi) = b \cdot \rho \cdot \left[a_{00} + a_{01} \cdot R(\rho) + a_{10} \cdot Z(\xi) + a_{02} \cdot R(\rho)^2 + a_{11} \cdot R(\rho) \cdot Z(\xi) + a_{20} \cdot Z(\xi)^2\right]$$
$$\mathring{w}(\rho,\xi) = b \cdot \xi \cdot \left[b_{00} + b_{01} \cdot R(\rho) + b_{10} \cdot Z(\xi) + b_{02} \cdot R(\rho)^2 + b_{11} \cdot R(\rho) \cdot Z(\xi) + b_{20} \cdot Z(\xi)^2\right]$$

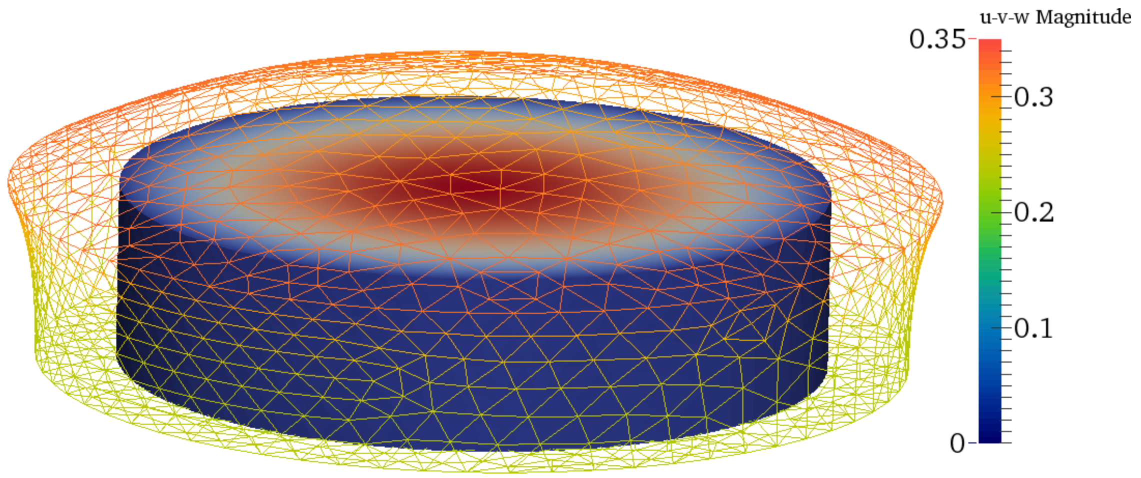


Figure 2: Illustration of the solution for $h/b = 0.5$ and $\nu = 0.333$ obtained by fino by postprocessing the VTK file it generates with Paraview.

where $R(\rho) = \rho^2 - 1$ and $Z(\xi) = \xi^2 - 1$.

In particular, for $h/b = 1/2$ and $\nu = 1/3$, the reported coefficients are

$$
\begin{aligned}
a_{00} &= +0.66056 & b_{00} &= -0.01773\\
a_{01} &= -0.44037 & b_{01} &= -0.46713\\
a_{10} &= +0.23356 & b_{10} &= -0.04618\\
a_{02} &= -0.06945 & b_{02} &= +0.10417\\
a_{11} &= -0.10417 & b_{11} &= -0.01152\\
a_{20} &= +0.00288 & b_{20} &= -0.00086
\end{aligned}
$$

Figure 2 illustrates the solution obtained by fino using a finite-element approach over an unstructured grid, which we describe in the following sections.

3.1 Geometry and mesh

As the problem geometry is a simple cylinder, we can both generate and mesh it with Gmsh. If the problem had had a more complex geometry, a CAD tool would have been neded. In particular, we use the OpenCASCADE interface Gmsh provides to create a cylinder of radius b and height h centered at the origin with its base on the x-y plane. We have to add an explicit point at the origin so we can set boundary conditions (1), (2) and (3). If we wanted to avoid rotations around the z axis we would add also a line from $(0, 0, 0)$ to $(0, b, 0)$ and associate boundary condition (5) to it.

In order to be able to choose the cylinder radius and height on the one hand and the characteristic mesh length ℓ_c on the other hand, instead of directly construction a Gmsh geometry script file (the usual `.geo` extension) we first create an M4-macro template that fino can operate on and replace variables h, b and ℓ_c with algebraic expressions so the complete set of problem parameters can be defined in the fino input file—potentially being read from the command line arguments or be varied parametrically in a pre-defined way using the facilities provided by the wasora framework [4] over which the fino tool is built. This M4 template is:

```
SetFactory("OpenCASCADE");        // Gmsh >= 2.16.0 is needed

Cylinder(1) = {0,0,0, 0,0,h, b};  // create a cylinder of radius b and height h
Point(10) = {0, 0, 0, lc};        // add a point at the origin
Point{10} In Surface {3};          // and embed it on the base

// physical entities (these will be linked to boundary conditions in veeder.fin)
Physical Point("origin") = {10};
Physical Surface("base") = {3};

Physical Volume("bulk") = {1};

// these lines are needed to avoid the cylinder from rotating but Fino can handle this situation nevertheless
Line(4) = {10, 2};
Line{4} In Surface {3};
```

```
Physical Line("extra_fix") = {4};

// mesh size settings
Mesh.CharacteristicLengthMin = 0.8*lc;
Mesh.CharacteristicLengthMax = 1.2*lc;
Mesh.CharacteristicLengthExtendFromBoundary = 0;

// mesh optimization settings
//Mesh.Lloyd = 1;
Mesh.HighOrderOptimize = 1;
Mesh.Optimize = 1;
Mesh.OptimizeNetgen = 1;
```

Listing 1: veeder.geo.m4

3.2 Thermo-elastic problem

Note that the file above cannot be directly passed to Gmsh as neither `h` nor `b` nor `lc` are defined. We use fino's (actually wasora's) keyword `M4` to define macros with these names containing the evaluated algebraic expressions according to the variables with the same name:

```
# solves the benchmark problem by J. Veeder "Thermo-elastic expansion of finite cylinders", AECL-2660, 1967

# geometric parameters
b = 1          # cylinder radius
h = 0.5        # cylinder half-height
lc = h/3       # characteristic length of the mesh

# material properties (given as variables means that they are uniform over space)
E = 1          # young modulus (does not matter for the displacement)
nu = 0.333     # poisson ratio
alpha = 1/2    # temperature expansion coefficient

# temperature distribution (given as algebraically-defined function of x, y and z)
T0 = 1
T(x,y,z) := T0*(1-(x^2+y^2)/(b^2))

# mesh
M4 { # create the geo file from a template using the value of h, b & lc from above
 INPUT_FILE_PATH veeder.geo.m4
 OUTPUT_FILE_PATH veeder.geo
 MACRO h  h
 MACRO b  b
 MACRO lc lc
}
SHELL "gmsh␣-3␣-v␣0␣-order␣2␣veeder.geo␣>␣/dev/null"  # call gmsh (2nd order)
MESH FILE_PATH veeder.msh DIMENSIONS 3                # use the resulting mesh

# boundary conditions
PHYSICAL_ENTITY NAME base       BC w=0      # no displacement in z
PHYSICAL_ENTITY NAME origin     BC u=0 v=0  # fixed (because w is already 0)
# PHYSICAL_ENTITY NAME extra_fix  BC v=0      # no displacement in y (without this the cylinder rotates around z)

# solve!
FINO_STEP

# non-dimensional displacement profiles
u~(xi)  := u(b, 0, xi*h) / (b*alpha*T0)
w~(rho) := w(rho*b, 0, h) / (b*alpha*T0)

INCLUDE original.was  # original solution as algebraic functions

# write the detailed profiles and the absolute error to a file
PRINT_FUNCTION FILE_PATH profile.dat {
      u~      u~o  u~(xi)-u~o(xi)
      w~      w~o  w~(xi)-w~o(xi)
} MIN 0 MAX 1 NSTEPS 100 HEADER

# write vtk output
MESH_POST FILE_PATH veeder.vtk T  VECTOR u v w

# screen output (only 10 values of displacements, text commented out so data can be plotted with gnuplot)
PRINT "\#␣==================================================================="
PRINT "\#␣h/b␣␣␣␣␣␣␣␣=␣" %.2f h/b
PRINT "\#␣nu␣␣␣␣␣␣␣␣␣=␣" %.2f nu
PRINT "\#␣elements␣␣␣=␣" %g elements
PRINT "\#␣nodes␣␣␣␣␣␣=␣" %g nodes
PRINT "\#␣-------------------------------------------------------------------"
PRINT "\#␣cpu␣time␣[sec]␣=␣"  %.2f time_cpu_build "(build)␣"  %.2f time_cpu_solve "(solve)"  SEP "␣"
PRINT "\#␣memory␣[Gb]␣␣␣␣=␣"  %.2f memory_usage_global/1e9 TEXT "/" available_memory/1e9  SEP "␣"
PRINT_FUNCTION FORMAT "%␣.3f" {
      u~   u~(xi)-u~o(xi)
      w~   w~(xi)-w~o(xi)
} MIN 0 MAX 1 NSTEPS 10 HEADER
```

Listing 2: veeder.fin

The original solution is computed in a separate file for clarity, which is a pure-wasora input file [4] and is included form the main fino file:

```
a00 = 0.66056
a01 = -0.44037
a10 = 0.23356
a02 = -0.06945
a11 = -0.10417
a20 = 0.00288
b00 = -0.01773
b01 = -0.46713
b10 = -0.04618
b02 = 0.10417
b11 = -0.01152
b20 = -0.00086

R(rho) := rho^2 - 1
Z(xi) := xi^2 - 1

uo(rho,xi) := rho * (a00 + a01*R(rho) + a10*Z(xi) + a02* R(rho)^2 + a11 * R(rho)*Z(xi) + a20 * Z(xi)^2)
wo(rho,xi) := xi  * (b00 + b01*R(rho) + b10*Z(xi) + b02* R(rho)^2 + b11 * R(rho)*Z(xi) + b20 * Z(xi)^2)

u~o(xi) := uo(1,xi)
w~o(rho) := wo(rho,1)
```

Listing 3: original.was

The fino input file `veeder.fin` thus first generates the geometry and the mesh by filling out the M4 template file `veeder.geo.m4` generating `veeder.geo`, which is the input file script that Gmsh uses to then generate the mesh file `veeder.msh` finally read back by fino with the `MESH` keyword. Once the problem is solved, the non-dimensional displacements in x and in z are non-dimensionalized and evaluated along the axial and radial directions respectively

$$
\begin{aligned}
\tilde{u}(\xi) &= \frac{u(b, 0, \xi \cdot h)}{b \cdot \alpha T_0} \\
\tilde{w}(\rho) &= \frac{w(\rho \cdot b, 0, h)}{b \cdot \alpha T_0}
\end{aligned}
$$

and then compared to the original six-term power-series results evaluated in the same directions

$$
\begin{aligned}
\mathring{\tilde{u}}(\xi) &= \mathring{u}(1, \xi) \\
\mathring{\tilde{w}}(\rho) &= \mathring{w}(\rho, 1)
\end{aligned}
$$

computed algebraically in the file `original.was`.

Running fino v0.5.48-gfc1ad7b with `veeder.fin` as the main input file computes the desired results:

```
$ fino veeder.fin
# ======================================================================
# h/b        =  0.50
# nu         =  0.33
# elements   =  1710
# nodes      =  2687
# ----------------------------------------------------------------------
# cpu time [sec] =  0.09 (build)  1.16 (solve)
# memory [Gb]    =  0.11 / 16.78
# xi    u~      u~(xi)-u~o(xi)  w~      w~(xi)-w~o(xi)
 0.000   0.449   0.019   0.574   0.021
 0.100   0.450   0.018   0.566   0.019
 0.200   0.454   0.015   0.544   0.017
 0.300   0.460   0.010   0.510   0.016
 0.400   0.470   0.003   0.462   0.014
 0.500   0.483  -0.004   0.401   0.010
 0.600   0.500  -0.012   0.327   0.003
 0.700   0.522  -0.021   0.244  -0.004
 0.800   0.549  -0.028   0.155  -0.009
 0.900   0.582  -0.034   0.063  -0.012
 1.000   0.623  -0.038  -0.027  -0.009
$
```

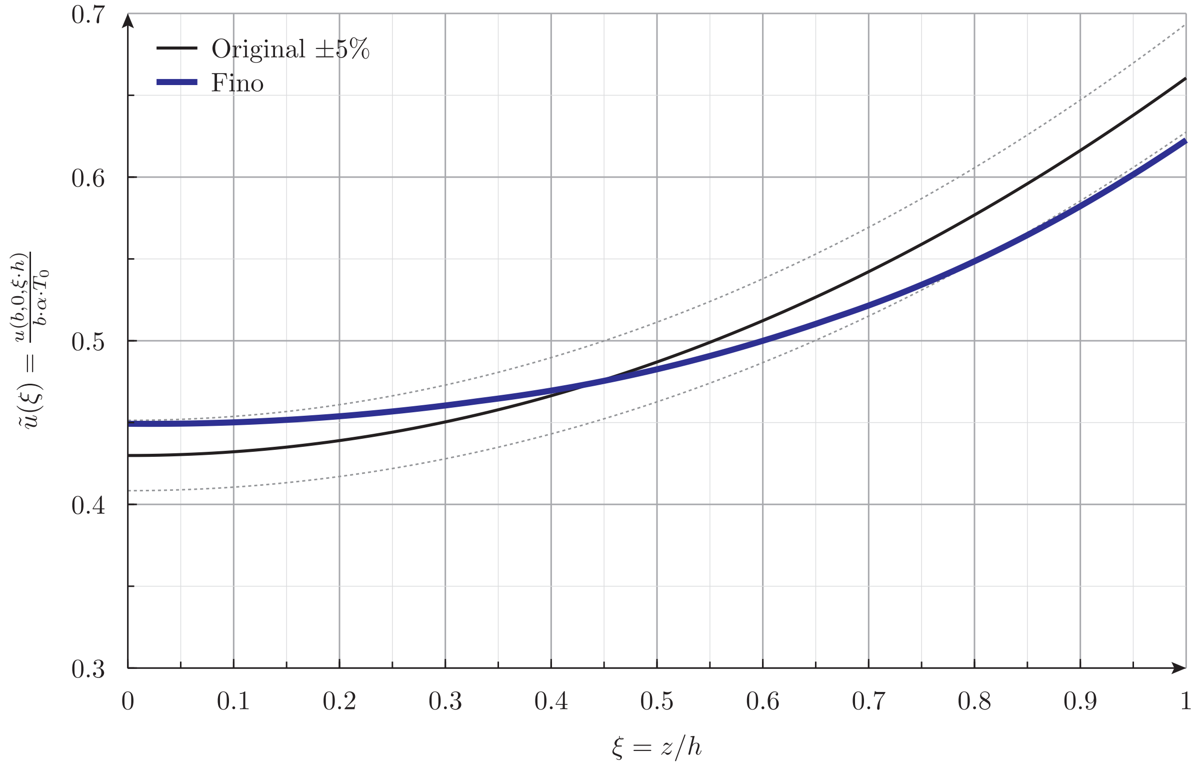


Figure 3: Radial non-dimensional displacement at the lateral surface of the cylinder ($\rho = 1$).

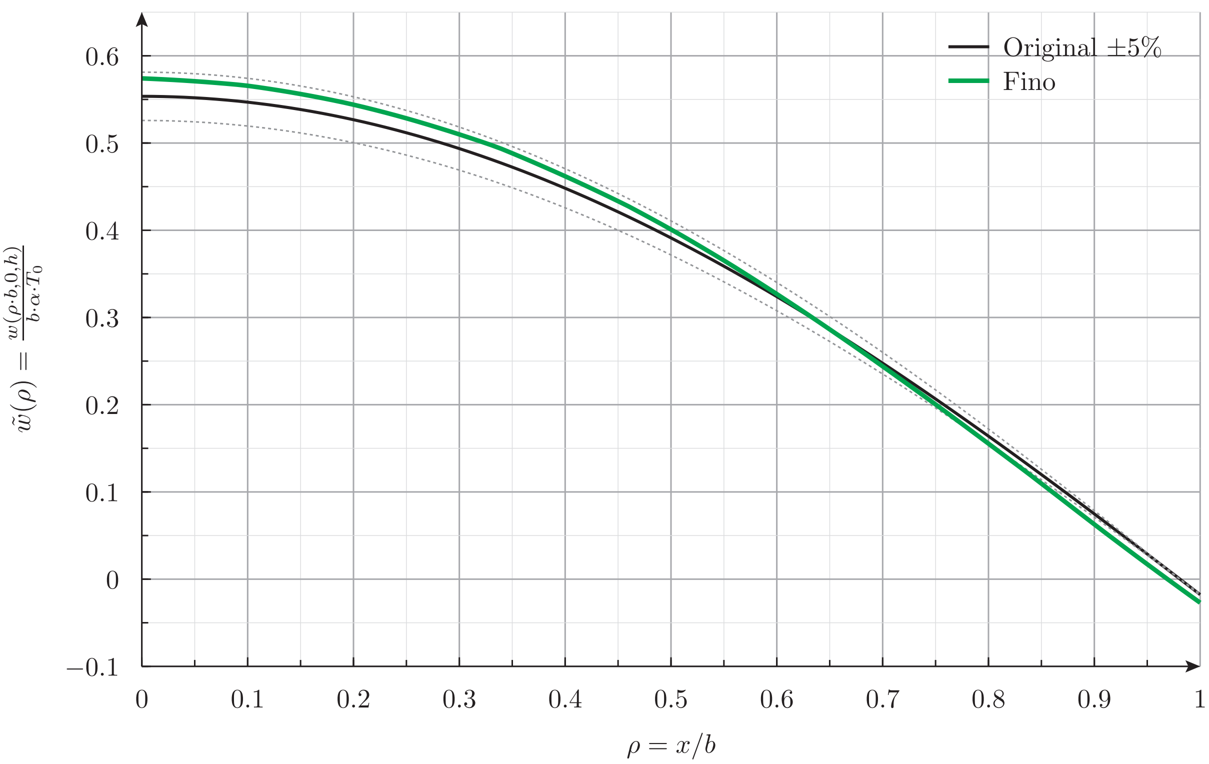


Figure 4: Axial non-dimensional displacement at the upper surface of the cylinder ($\xi = 1$).

Figures 3 and 4 show the one-dimensional profiles that the original reference [1] reports both numerically and graphically, namely the radial non-dimensional displacement evaluated along the cylinder lateral surface $\tilde{u}(\xi)$ and the axial displacement along the cylinder upper surface $\tilde{w}(\rho)$. It should be noted that the original solution was computed using a six-term polynomial expansion subject to a strain energy minimization. That is to say, even though it is a better approach to finite cylinder problems that assuming a circular geometry under either plane stress o plane strain, we expected fino's solution to be closer to the actual real solution than the one reported by Veeder fifty years ago, that is more an illustrative reference rather than a solution benchmark.

3.3 Parametric study over grid size (and element order)

We can exploit the wasora's framework design basis [5] to easily perform a parametric study with respect to the grid characteristic length ℓ_c. We can use the PARAMETRIC keyword to vary one variable, say c in a specified range in a pre-defined way, for example in linear steps or by following a quasi-random number sequence. Because the geometry script file is generated from a M4 template, we can set Gmsh's ℓ_c as a function of fino's coarseness factor c. Moreover, we can use the construction $1 to pass arguments in the fino command line as Gmsh's arguments so we can have the user to choose the order of the elements at runtime.

```
b = 1
h = 0.5

E = 1
nu = 0.333
alpha = 1/2

T0 = 1
T(x,y,z) := T0*(1-(x^2+y^2)/(b^2))

PARAMETRIC c MIN 2 MAX 24 STEP 1

OUTPUT_FILE geo  veeder-$1-%.2f.geo c
M4 INPUT_FILE_PATH veeder.geo.m4  OUTPUT_FILE geo  MACRO h  h  MACRO b  b MACRO lc h*($1)/c
t0 = clock()
SHELL "gmsh␣-3␣-v␣0␣-order␣$1␣veeder-$1-%.2f.geo␣>␣/dev/null" c
t1 = clock()
INPUT_FILE mesh veeder-$1-%.2f.msh c
MESH FILE mesh DIMENSIONS 3

PHYSICAL_ENTITY NAME base       BC w=0
PHYSICAL_ENTITY NAME origin     BC u=0 v=0
PHYSICAL_ENTITY NAME extra_fix  BC v=0

FINO_STEP

PRINT %.0f nodes elements %.2f c %.8f u(b,0,0)/(b*alpha*T0) w(0,0,h)/(b*alpha*T0) %.3f t1-t0 time_wall_build time_wall_solve %.2e ↩
      memory_usage_global
```

Listing 4: convergence.fin

```
$ fino convergence.fin 1 | tee convergence1.dat
165     593     2.00    0.46526524      0.48190672      0.157   0.007   0.024   2.17e+07
438     1710    3.00    0.46008672      0.54209584      0.313   0.021   0.049   3.03e+07
750     3112    4.00    0.45434478      0.55620820      0.357   0.040   0.093   3.64e+07
1277    5489    5.00    0.45857662      0.55848495      0.593   0.072   0.174   4.86e+07
2026    8909    6.00    0.45303474      0.56445990      0.799   0.131   0.305   6.55e+07
2888    13212   7.00    0.45543201      0.56919136      1.268   0.203   0.545   8.58e+07
4081    19004   8.00    0.45463409      0.56866651      1.750   0.298   0.894   1.14e+08
5420    25825   9.00    0.45542952      0.56909475      2.402   0.413   1.137   1.45e+08
7210    35043   10.00   0.45252516      0.57000994      3.259   0.578   1.550   1.87e+08
9115    44739   11.00   0.45255898      0.56967317      3.770   0.683   1.802   2.32e+08
11620   57828   12.00   0.45189046      0.57139978      5.191   0.929   2.407   2.92e+08
14466   72536   13.00   0.45238953      0.57108274      7.003   1.223   3.348   3.59e+08
17663   89572   14.00   0.45240719      0.57207873      9.136   1.396   3.967   4.36e+08
21296   108902  15.00   0.45281872      0.57219991      10.552  1.844   5.583   5.24e+08
25314   130385  16.00   0.45112099      0.57189450      14.084  2.113   6.017   6.21e+08
29834   155143  17.00   0.45115732      0.57229426      16.957  2.730   7.114   7.29e+08
34756   182139  18.00   0.45083251      0.57163400      18.797  3.550   9.990   8.52e+08
40310   212731  19.00   0.45086085      0.57198814      22.565  3.736   10.621  9.85e+08
46307   245787  20.00   0.45177690      0.57232963      26.004  4.002   12.904  1.13e+09
53342   284241  21.00   0.45117474      0.57196447      32.263  6.413   13.952  1.31e+09
60883   325769  22.00   0.45124344      0.57206281      36.818  5.710   17.391  1.49e+09
69082   370944  23.00   0.45153789      0.57212709      42.394  6.342   20.941  1.70e+09
77736   418803  24.00   0.45068535      0.57221631      47.336  7.670   22.676  1.91e+09
$ fino convergence.fin 2 | tee convergence2.dat
319     188     2.00    0.45568723      0.58041702      0.114   0.010   0.037   2.69e+07
882     546     3.00    0.45187752      0.58034789      0.164   0.029   0.090   4.69e+07
```

```
959     593     4.00    0.45198779      0.57154752      0.166   0.031   0.094   5.11e+07
1709    1084    5.00    0.45409443      0.57267366      0.206   0.060   0.205   7.67e+07
2687    1710    6.00    0.45177463      0.57399829      0.271   0.096   0.372   1.10e+08
3522    2278    7.00    0.45235025      0.57247349      0.315   0.127   0.592   1.39e+08
4812    3142    8.00    0.45167304      0.57302810      0.404   0.190   0.870   1.84e+08
6370    4195    9.00    0.45176117      0.57269689      0.527   0.239   1.239   2.38e+08
8263    5488    10.00   0.45094817      0.57263617      0.662   0.323   1.591   3.04e+08
10330   6854    11.00   0.45212083      0.57278623      0.729   0.402   1.982   3.76e+08
13363   8909    12.00   0.45195177      0.57276954      0.897   0.549   2.870   4.81e+08
16458   11005   13.00   0.44973480      0.57266300      1.138   0.663   3.673   5.89e+08
19459   13212   14.00   0.45080242      0.57282446      1.290   0.791   4.279   6.96e+08
23544   15994   15.00   0.45012205      0.57268124      1.595   0.963   5.546   8.40e+08
27815   19004   16.00   0.45005605      0.57265601      1.808   1.218   6.813   9.91e+08
32548   22355   17.00   0.45038178      0.57264827      2.074   1.416   8.182   1.16e+09
37400   25800   18.00   0.45043822      0.57265860      2.578   1.582   9.768   1.33e+09
43387   30056   19.00   0.45100654      0.57264936      2.862   1.850   11.264  1.53e+09
50408   35043   20.00   0.45076934      0.57265230      3.353   2.280   14.063  1.78e+09
57147   39780   21.00   0.44946261      0.57265608      4.049   2.536   15.353  2.02e+09
64216   44780   22.00   0.44923081      0.57265049      4.225   2.779   17.766  2.26e+09
73053   51062   23.00   0.44937308      0.57265277      4.829   3.178   20.220  2.57e+09
82510   57828   24.00   0.44952588      0.57265069      5.629   3.804   24.085  2.90e+09
$
```

Figure 5 shows how non-dimensional axial and radial displacements evaluated at $(b, 0, 0)$ and $(0, 0, h)$ respectively change with the number of nodes in the grid, for both first and second-order elements. We take number of nodes as the abscissa because the problem size is proportional to the number of nodes—actually it is three times the number of nodes—not the number of elements. In effect, figure 6 plots the number of elements (including both tetrahedra and triangles) versus the number of nodes for first and second-order grids. Of course, for the same number of nodes, first-order grids contain far more elements—between seven and eight times more.

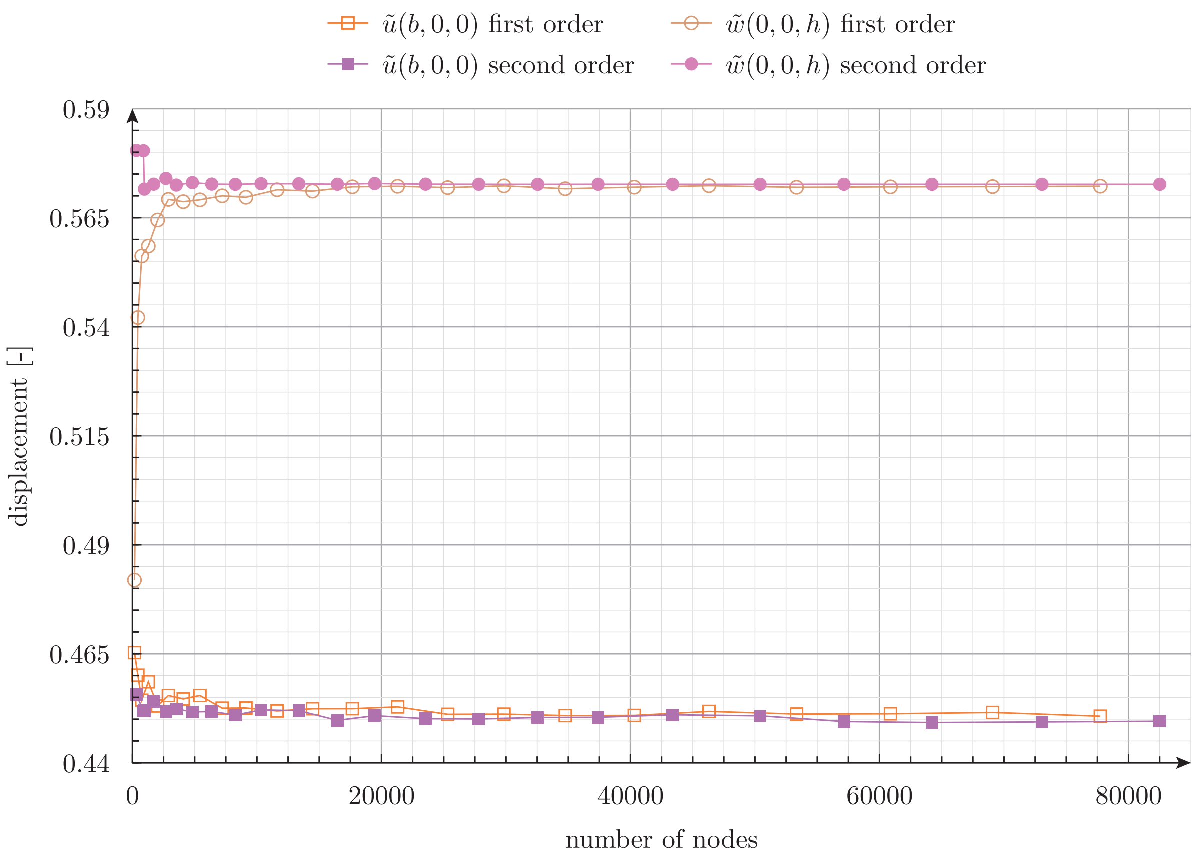


Figure 5: Radial and axial displacements at two locations as functions of the number of elements of the grid.

Whilst the number of nodes defines the problem size (and thus more nodes mean that more computational resources are needed to solve the problem), the number of elements defines the effort needed to build the stiffness

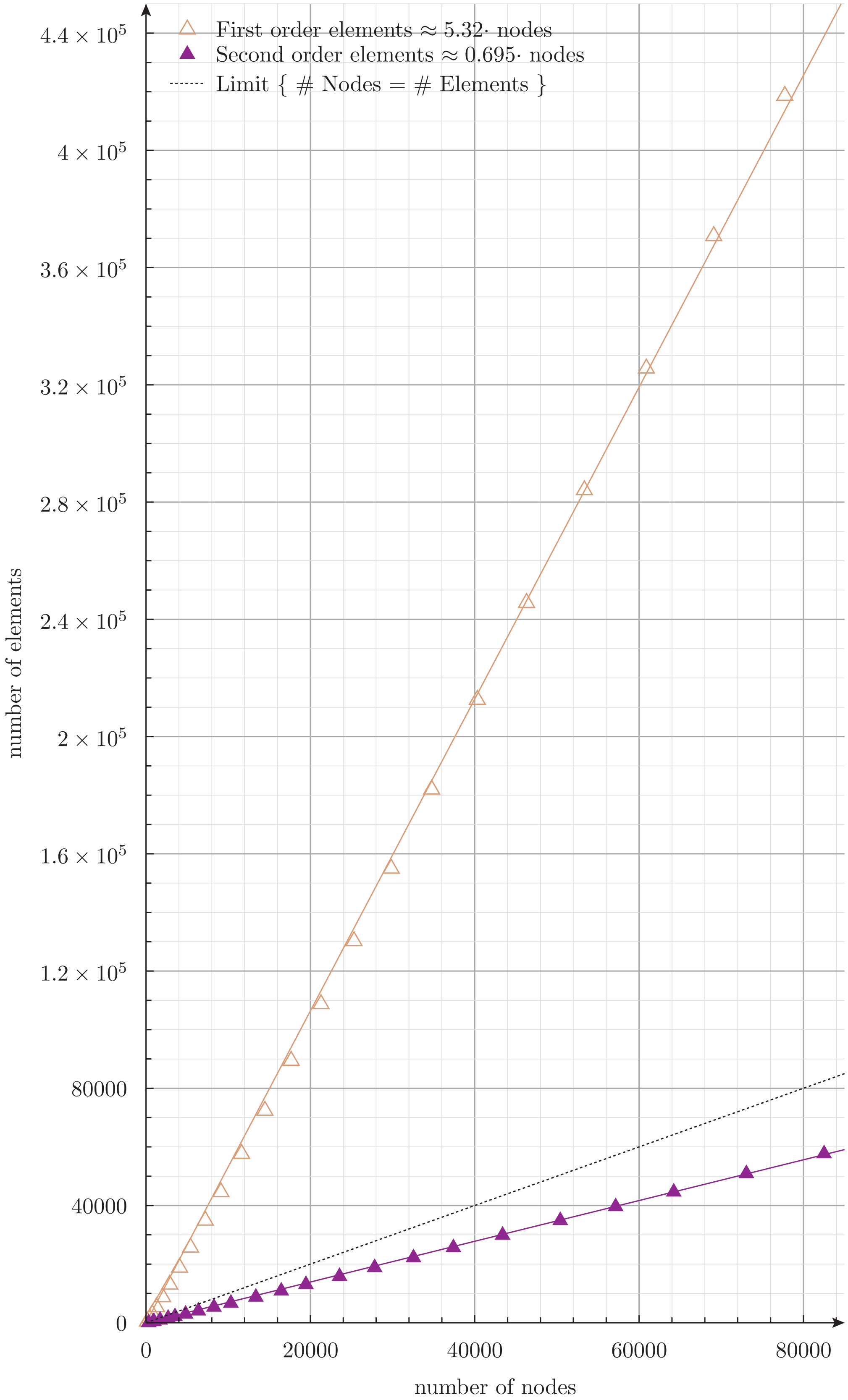


Figure 6: Number of elements vs. number of nodes for first and second-order tetrahedra.

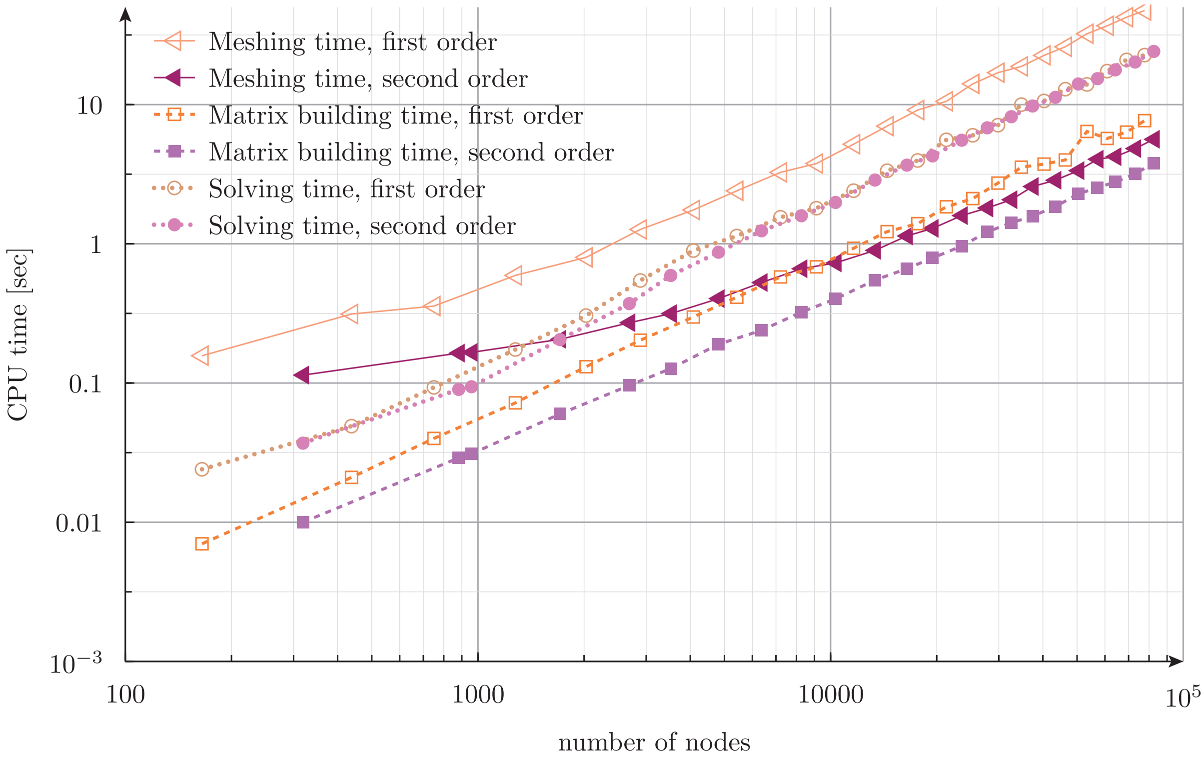


Figure 7: CPU time (serial) needed to build and solve the stiffness matrix of the problem as functions of the number of nodes for first and second-order elements. The time needed to generate the grid is not taken into account.

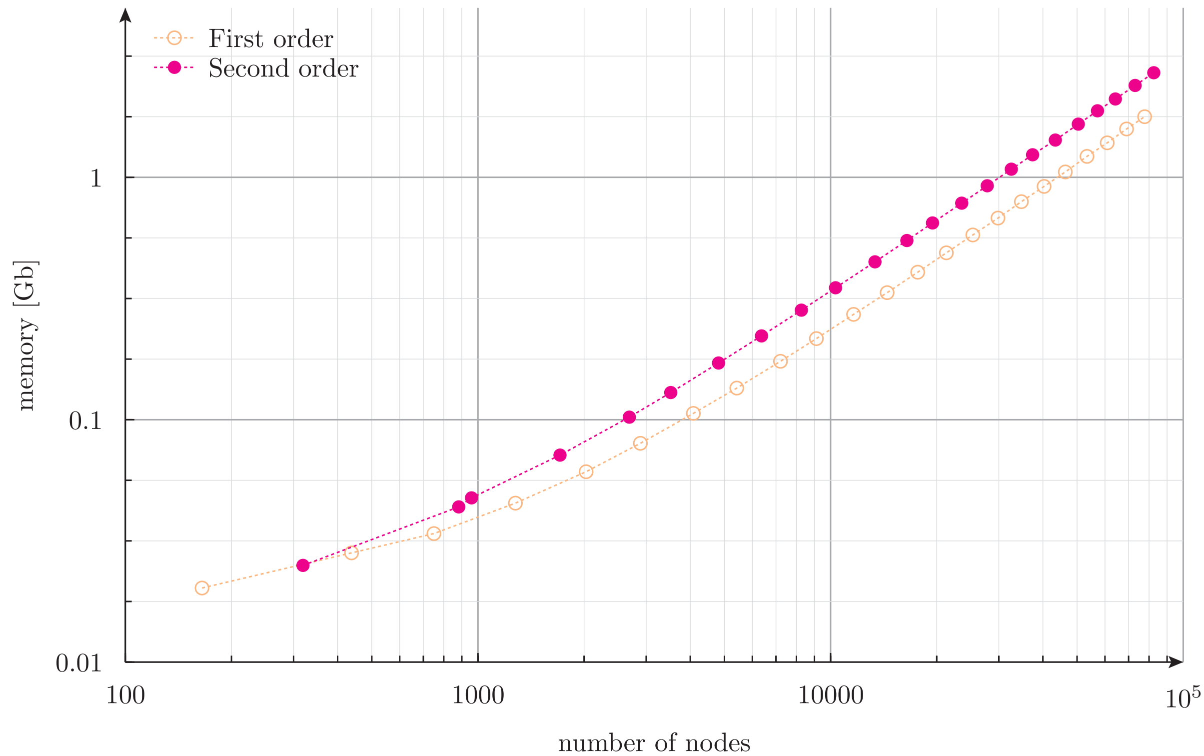
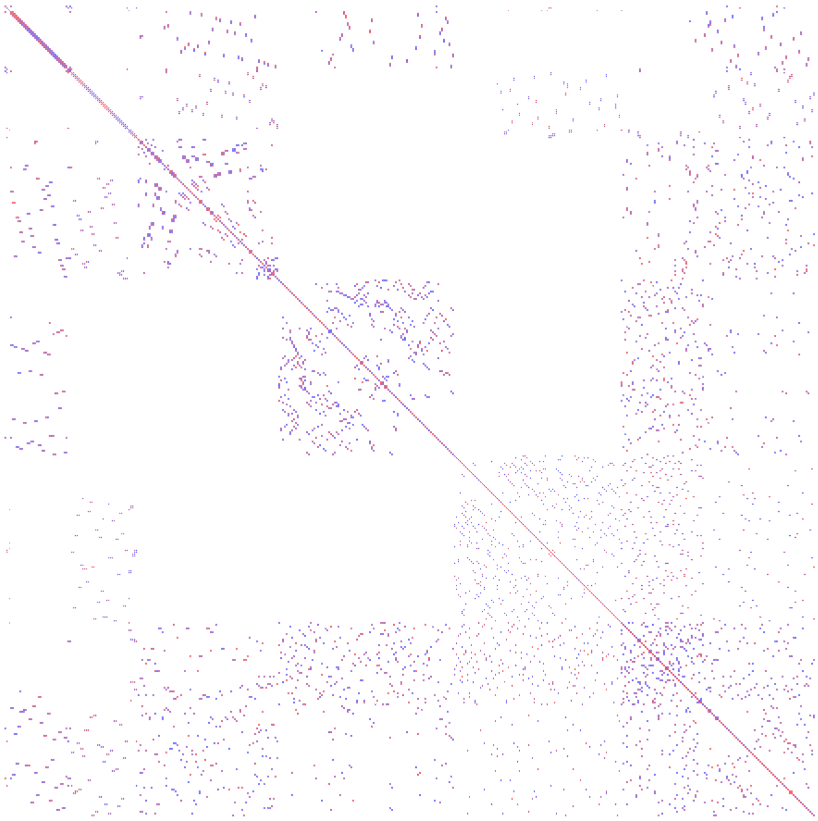
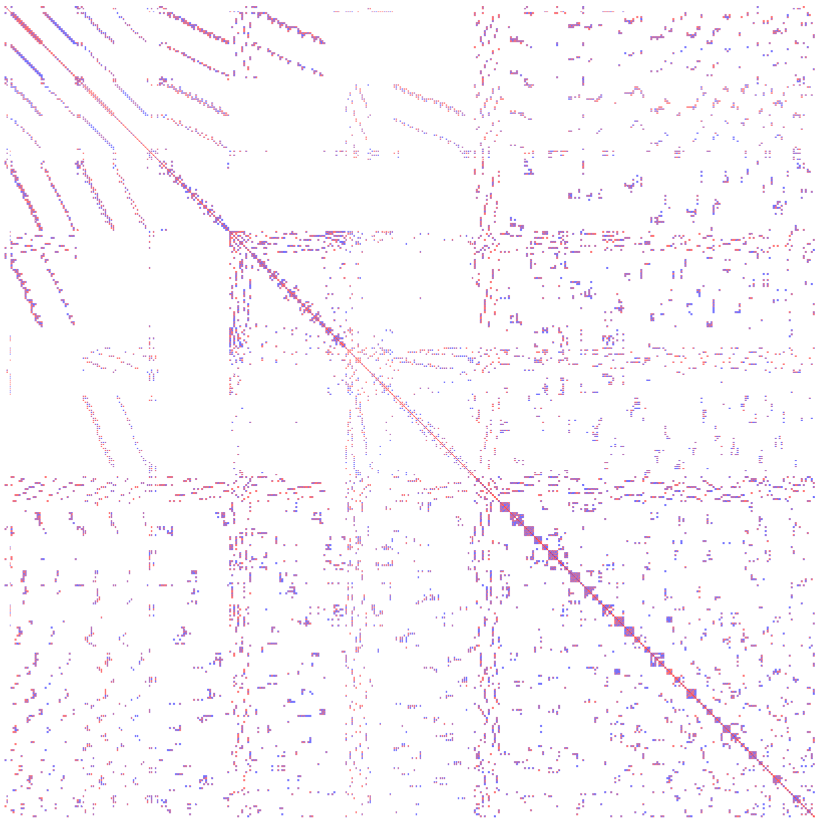


Figure 8: Amount of RAM needed to solve the problem vs. number of nodes.



(a) First-order tetrahedra, 1314×1314



(b) Second-order tetrahedra, 1212×1212

Figure 9: Stiffness matrices of a first-order and second-order formulations with a similar number of global degrees of freedom.

matrix. Moreover, the meshing time is less for second-order because relatively bigger elements are needed to fill up the continuous domain. Nevertheless, they have more nodes so the numerical integration in each element needed for the matrix assembly is slightly more expensive. Overall, however, it takes less CPU time to build a second-order stiffness matrix than a first-order one with the same number of nodes, as illustrated in figure 7. It should be noted though, that the resulting second-order stiffness matrix is less sparse and more connected, so more memory is needed to solve the problem (figures 8 and 9).

4 Conclusions

We have solved a fifty-years-old problem that is still both technologically and numerically interesting. We can see the way fino invites to tackle the case—for example using human-friendly algebraic expressions and calling a script-friendly mesher through a macro template—is rather different than other point-and-click software packages. This UNIX approach[6] allows a wide variety of workflows, from a completely automated execution and reporting under a Git-based revision control system up to a web-based front-end running on the cloud.

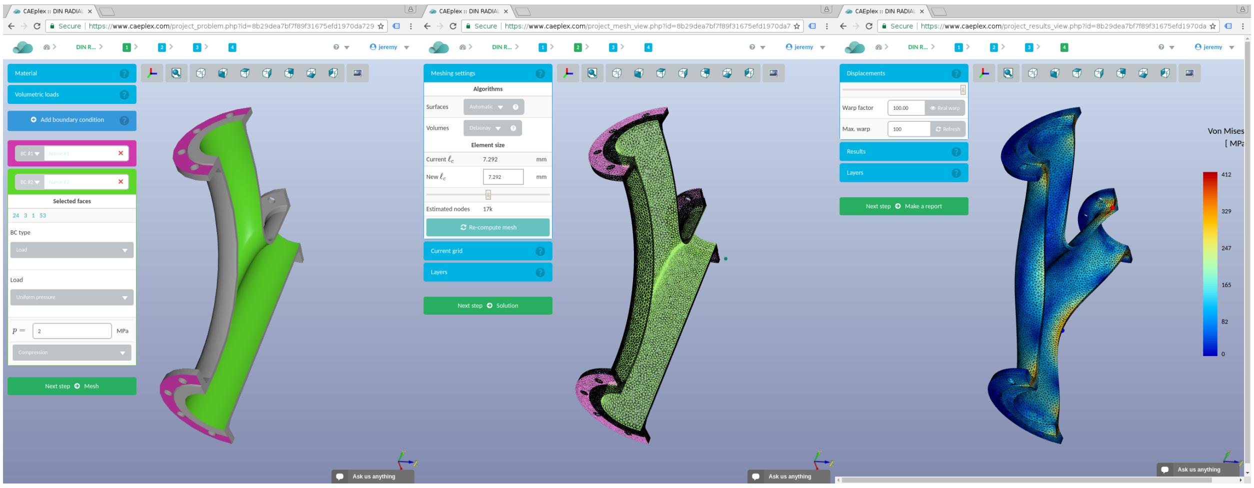


Figure 10: Fino and Gmsh running as a back-ends on the cloud for the web-based front-end CAEplex at `caeplex.com`

We also got to grasp some of the insights of grid convergence and the different aspects of first and second-order elements. We even got to see and compare the resulting stiffness matrices, which further illustrates the openness nature of the free and open source finite-element analysis software package fino developed by Seamplex.

References

[1] J. Veeder. *Thermo-elastic expansion of finite cylinders*. Tech. rep. AECL-2660. Chalk River: Atomic Energy of Canada Limited, 1967.

[2] Satish Balay et al. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries". In: *Modern Software Tools in Scientific Computing*. Ed. by E. Arge, A. M. Bruaset, and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202.

[3] Satish Balay et al. *PETSc Web page*. `http://www.mcs.anl.gov/petsc`. 2016.

[4] G. Theler. *Description of the computational tool wasora*. Technical Description SP-WA-15-TD-9E3D. Version B. Seamplex, 2016.

[5] G. Theler. *On the design basis of a new core-level neutronic code written from scratch*. Tech. rep. SP-MI-14-AR-5B44. Version D. Seamplex, 2016.

[6] Eric S. Raymond. *The Art of UNIX Programming*. Addison-Wesley, 2003.