The NAFEMS Benchmark Challenge #1

Fino test case 060-nafems-challenge-problem1

Title	**The NAFEMS Benchmark Challenge #1**
Tags	elasticity tension
Runnng time	1 sec
See also	006-cylinder-pure-compression
CAEplex case	`https://caeplex.com/p/26d55`
Available in	HTML PDF ePub

1 Problem description

This problem is a very nice exercise proposed by Dr. Ramsay as the first case of the The NAFEMS Benchmark Challenge Volume 1 book [1]. The original formulation of the problem is the following:

> **Stress at the Centre of a Square Plate with Linear Boundary Tractions**
>
> A unit square homogeneous and isotropic steel plate is centred at the origin of the x-y plane with edges parallel with the coordinate axes and is loaded with linearly distributed normal and tangential boundary tractions as shown in fig. 1. The plate can be assumed to be thin so that a plane-stress constitutive relationship is appropriate and for convenience a unit thickness may be used.

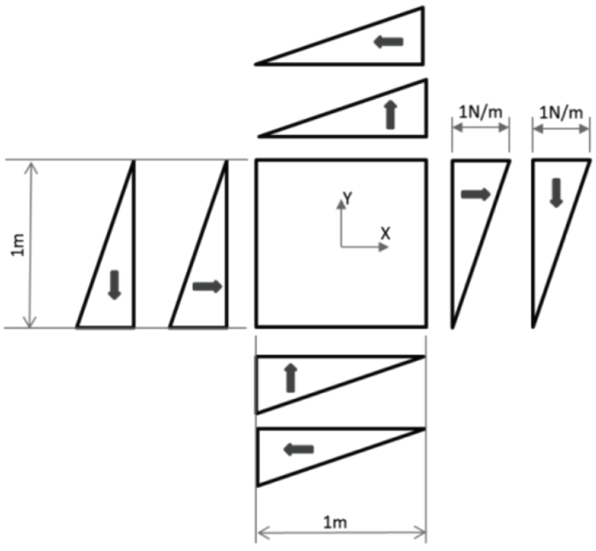


Figure 1: NAFEMS Challenge #1—problem statement

> **The Challenge**
>
> The challenge is to produce two models of this problem in your finite element software and then answer some questions. The first model should use a single four-noded element and the

second a single eight-noded element. As engineers interested in the integrity of the plate we might wish to see the distribution of von Mises stress over the plate. We would like you to provide:

a. Numerical values for the von Mises stress at the centre of the plate for both models,
b. A statement as to which of these values is correct,
c. Contour plots of von Mises Stress for both models,
d. A brief commentary on how you modelled the problem and what, if anything, of interest you note about this problem—please include details of the software that you used.

2 Geometry and mesh

Let's solve the exercise with Fino the UNIX way. The problem does not ask for any explicit Dirichlet boundary conditions. Even though Fino (actually PETSc) can handle singular stiffness matrices and return any of the infinitely many possible solutions, their derivatives will not be accurate. Yet it would be interesting to test the degree of pollution introduced by not removing all rigid-body modes of displacement. Hence, instead of only two models we are going to produce four models. For each element type, we solve

- "A" without any displacement boundary condition
- "B" with a set of displacement boundary conditions (both of Dirichlet and multi-freedom types) in order to remove rigid-body movement

Hence, the four basic models are named

- quad 4 "A"
- quad 4 "B"
- quad 8 "A"
- quad 8 "B"

First, the geometry is the same for all four models, namely a non-dimensional 1×1 square centered at the origin. As the four edges carry loads, one physical group per edge is needed. Also, to allow for setting displacement boundary conditions, we also create one physical group for each of the four corner nodes. Thus we start with a common `quad.geo` and ask Gmsh to create one for us in the range $[-\frac{1}{2}, -\frac{1}{2}] \times [+\frac{1}{2}, +\frac{1}{2}]$:

```
// create the continuous rectangle
SetFactory("OpenCASCADE");
Rectangle(1) = {-1/2, -1/2, 0, 1, 1};

// name the edges to set Neumann BCs
Physical Curve("bottom") = {1};
Physical Curve("top") = {3};
Physical Curve("left") = {4};
Physical Curve("right") = {2};

// name the corner points to set Dirichlet BCs
Physical Point("one") = {1};
Physical Point("two") = {2};
Physical Point("three") = {3};
Physical Point("four") = {4};

// one physical surface for the bulk of the material
Physical Surface("bulk", 3)  = {1};

Mesh.RecombineAll = 1;       // ask for quads instead of triangs
```

Now, to get a single four-node quadrangle we create another file `quad4.geo` that merges the previous one and asks for only one first-order quadrangle:

```
Merge "quad.geo";
Transfinite Line {1:4} = 1;      // make sure we have only one element
Transfinite Surface "*";
Mesh.ElementOrder = 1;           // ask for first–order elements
```

To obtain an eight-node quadrangle, we do the same in `quad8.geo`. Note that eight-node quadrangles are considered "incomplete" in Gmsh:

```
Merge "quad.geo";
Transfinite Line {1:4} = 1;      // make sure we have only one element
Transfinite Surface "*";
Mesh.ElementOrder = 2;           // ask for second–order...
Mesh.SecondOrderIncomplete = 1;  // ...incomplete elements (i.e. quad8)
```

3 Input file

From fig. 1, the loads at the edges are the same for all the four cases when stating them as continuous expressions of x and y. Indeed, this is how Neumann boundary conditions are expected by Fino to be provided by the user:

$$\sigma_{tx} = -\left(\frac{1}{2} - x\right) \quad \text{if } y = -\frac{1}{2}$$
$$\sigma_{ty} = +\left(\frac{1}{2} - x\right) \quad \text{if } y = -\frac{1}{2}$$

$$\sigma_{tx} = -\left(\frac{1}{2} + x\right) \quad \text{if } y = +\frac{1}{2}$$
$$\sigma_{ty} = +\left(\frac{1}{2} + x\right) \quad \text{if } y = +\frac{1}{2}$$

$$\sigma_{tx} = +\left(\frac{1}{2} - y\right) \quad \text{if } x = -\frac{1}{2}$$
$$\sigma_{ty} = -\left(\frac{1}{2} - y\right) \quad \text{if } x = -\frac{1}{2}$$

$$\sigma_{tx} = +\left(\frac{1}{2} + y\right) \quad \text{if } x = +\frac{1}{2}$$
$$\sigma_{ty} = -\left(\frac{1}{2} + y\right) \quad \text{if } x = +\frac{1}{2}$$

In the "A" cases, no further boundary conditions are needed. For the "B" cases, the following conditions are used in order to remove the rigid-body modes of displacement:

1. fixed lower left corner, point “one” (Dirichlet)
2. horizontal displacement equal to vertical displacement in upper right corner, point “three” (penalty method)
3. same displacements in the remaining two corners, points “two” and “four” (penalty method)

In effect, this single file `challenge.fin` defines and solves either of the four cases depending on two extra arguments to Fino, which are expanded to `$1` and `$2` respectively. The first argument should be equal to either `4` or to `8` and the second one either `A` or `B`.

```
MESH FILE_PATH quad$1.msh    # read input mesh file, either quad4 or quad8
FINO_PROBLEM PLANE_STRESS    # use plane-stress constitutive equations
FINO_SOLVER GRADIENT nodes

E = 1   # unitary young modulus
nu = 0  # null poisson's ratio

# neumann boundary conditions
PHYSICAL_GROUP bottom BC tx=-(1/2-x) ty=+(1/2-x)
PHYSICAL_GROUP top    BC tx=-(1/2+x) ty=+(1/2+x)
PHYSICAL_GROUP left   BC tx=+(1/2-y) ty=-(1/2-y)
PHYSICAL_GROUP right  BC tx=+(1/2+y) ty=-(1/2+y)

# set or not dirichlet BCs depending on $2
INCLUDE dirichlet-$2.fin

FINO_STEP   # solve!

# von mises computed explicitly from strains to force usage of the derivatives of the
# shape functions in the interpolation and not to use the shape functions themselves
vonmises(x,y) := sqrt(dudx(x,y)^2 - dudx(x,y)*dvdy(x,y) + dvdy(x,y)^2 + 3*(0.5*(dudy(x,y) + dvdx(x,y)))^2)
```

Note that being this a stress-defined problem, the Young Modulus E only changes the displacements—which are not asked for in the problem statement. So any value of E would (should) give the same von Mises stresses. Regarding the Poisson’s ratio ν, it is set to zero to simplify the analysis.

The Dirichlet boundary conditions files are either `dirichlet-A.fin` or `dirichlet-B.fin`. The former is empty and the latter is

```
# dirichlet boundary conditions
PHYSICAL_GROUP one    BC fixed
PHYSICAL_GROUP three  BC 0=u-v
PHYSICAL_GROUP four   # needed for the BC below
PHYSICAL_GROUP two    BC mimic(u_four) mimic(v_four)
```

Item a. asks for the von Mises stress at the origin. In Fino, any space-dependent result such as the displacements u and v or the stresses σ_x, σ_y and τ_{xy} are given as spatial functions which can be referred to in the input file as `u(x,y)`, `v(x,y)`, `sigmax(x,y)`, `sigmay(x,y)` and `tauxy(x,y)` respectively. The von Mises stress $\sigma(x,y)$ is given as `sigma(x,y)`. All these functions can be evaluated at any arbitrary point of the domain. If the argument $\mathbf{x}$ coincides with the location of a node, the nodal value is returned. For the case of stresses, they are computed as discussed in Stresses in a 10-node tetrahedron with prescribed displacements. But if $\mathbf{x}$ does not coincide with a node, the value is interpolated using the elemental shape functions. For large problems with many nodes and elements, this interpolation procedure usually gives acceptable results for stresses. However, in the present case, directly computing the answer to the first item using `sigma(0,0)`

is wrong as the stresses ought to be interpolated using the derivatives of the shape functions instead of the shape functions themselves. This issue can be solved by explicitly computing the von Mises stress out of the gradients of the displacements—which are indeed interpolated using the derivatives of the shape functions—in a custom function. Therefore a `vonmises(x,y)` function is defined using the partial derivatives of the displacements (i.e. strains) `dudx(x,y)`, `dudy(x,y)`, `dvdx(x,y)` and `dvdy(x,y)` for the plane-stress formulation as

$$\text{vonmises}(x,y) = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 - \frac{\partial u}{\partial x}\cdot\frac{\partial v}{\partial y} + \left(\frac{\partial v}{\partial y}\right)^2 + 3\left[\frac{1}{2}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]^2}$$

This file `challenge.fin` only solves the problem and does not give any result. Very much like in the previous section, we use this input as a common file which is included when answering each of the four questions of the problem.

To answer item a., we print evaluate `sigma(0,0)` and `vonmises(0,0)`. The first value will be only approximated and would converge to the real stress only if the mesh is refined. The second value is the answer to the problem.

```
PRINT "Case quad$1$2"
INCLUDE challenge.fin
PRINT "Von Mises at origin interpolated with shape functions:" %.3f sigma(0,0)
PRINT "Von Mises at origin interpolated with derivatives:    " %.3f vonmises(0,0)
```

The second question asks which of the two values is correct. One way to answer that is to solve the problem with a refined mesh, expecting the approximate finite-element solution to converge to the actual continuous solution as the number of unknowns increases. This can be easily be done by creating two new geometry files, say `quad4-ref.geo`

```
Merge "quad4.geo";
Transfinite Line {1:4} = 100;    // refine
```

and `quad8-ref.geo`

```
Merge "quad8.geo";
Transfinite Line {1:4} = 50;     // refine
```

and pass `4-ref` and `8-ref` as the first extra arguments to Fino.

Next we are asked to provide contour plots of the von Mises stresses in both models. Not only do we have four models instead of two because of the Dirichlet conditions, but we also have both `sigma(x,y)` and `vonmises(x,y)` which we would like to compare. Even more, we might use the single-element case or the refined mesh. This leads to having potentially sixteen contour plots to compare. We ask Fino to write $\sigma(x,y)$ in the range $[-\frac{1}{2},-\frac{1}{2}]\times[+\frac{1}{2},+\frac{1}{2}]$ with 20 sampling points in each direction:

```
INCLUDE challenge.fin
PRINT_FUNCTION FILE_PATH contour$1$2.dat sigma vonmises MIN -1/2 -1/2 MAX +1/2 +1/2 NSTEPS 20 20
```

The last question asks for comments on how the problem is modeled, which are discussed in sec. 6. This case has a few caveats and particularities which make it extremely interesting. Some insight of what is going on with the mathematics behind the discretized problem is to take a peek at the algebraic objects created by the FEM solver. We create yet another input file which asks Fino to dump the stiffness matrix K, the right-hand side vector $\mathbf{b}$ and the unknown displacements vector $\mathbf{u}$ as GNU Octave ASCII representations.

```
PRINT "Case quad$1$2"
INCLUDE challenge.fin
FINO_DEBUG FILE_PATH quad$1$2 MATRICES_PETSC_OCTAVE
```

4 Execution

To answer question a. we execute `a.fin` with both `4` and `8` extra parameters:

```
$ for i in 4 8; do for j in A B; do fino a.fin $i $j; done; done
Case quad4A
Von Mises at origin interpolated with shape functions:  0.000
Von Mises at origin interpolated with derivatives:      0.000
Case quad4B
Von Mises at origin interpolated with shape functions:  0.000
Von Mises at origin interpolated with derivatives:      0.000
Case quad8A
Von Mises at origin interpolated with shape functions:  1.000
Von Mises at origin interpolated with derivatives:      0.000
Case quad8B
Von Mises at origin interpolated with shape functions:  1.000
Von Mises at origin interpolated with derivatives:      0.000
$
```

To answer question b. we execute again `a.fin` but using `4-ref` and `8-ref` parameters to use the refined meshes:

```
$ fino a.fin 4-ref A
Case quad4-refA
Von Mises at origin interpolated with shape functions:  0.010
Von Mises at origin interpolated with derivatives:      0.000
$ fino a.fin 4-ref B
Case quad4-refB
Von Mises at origin interpolated with shape functions:  0.010
Von Mises at origin interpolated with derivatives:      0.000
$ fino a.fin 8-ref A
Case quad8-refA
Von Mises at origin interpolated with shape functions:  0.020
Von Mises at origin interpolated with derivatives:      0.000
$ fino a.fin 8-ref B
Case quad8-refB
Von Mises at origin interpolated with shape functions:  0.020
Von Mises at origin interpolated with derivatives:      0.000
$
```

The data needed to build contour plots is created by executing `c.fin` with all the single-element and refined meshes:

```
$ for i in 4 8 4-ref 8-ref; do for j in A B; do fino c.fin $i $j; done; done
$
```

Finally, we can check the determinant of the stiffness matrices in the single-element cases. In order to do that we dump the algebraic objects in `.m` files and then call Octave:

```
$ for i in 4 8; do for j in A B; do fino matrix.fin ${i} ${j}; octave --eval "source quad${i}${j}-K.m; ↩
    det(K)"; done; done
Case quad4A
ans =   -4.5938e-52
Case quad4B
ans =    1.7347e-18
Case quad8A
ans =   -8.7122e-49
Case quad8B
ans =    8.3935e-18
$
```

5 Results

The output of `a.fin` would indicate that the answer from `quad4` is right and `sigma(0,0)` from `quad8` is wrong but `vonmises(0,0)` is right—which we already suspected. Also, cases "B" seem to give more accurate values for the derivatvies than cases "A." The actual analytical solution is

$$\sigma_{\mathrm{vM}} = 2\sqrt{(x+y)^2} = 2\,|x+y|$$

which for reference is shown in fig. 2 as the contour plots of the refined cases.

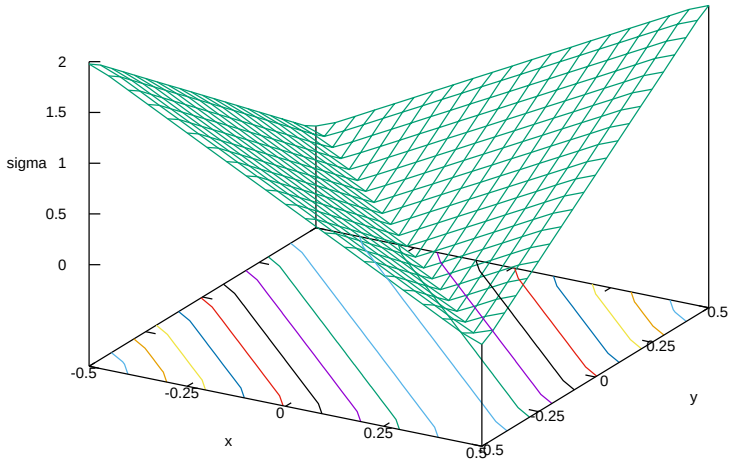
In effect, the contour plots from fig. 2 help us say that the `quad4` case completely misses the problem and `quad8` actually solves it exactly.

6 Aftermath

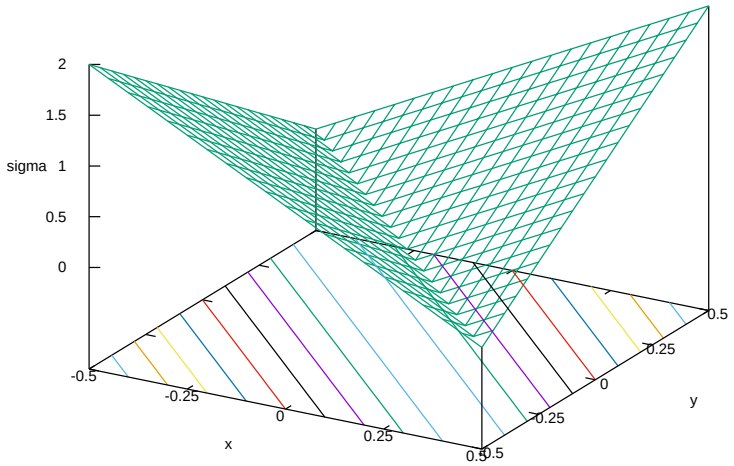
The `quad4` case is not able to recover any behavior of the continuous problem, as the boundary conditions are such that the integration process that translate a continuous boundary condition such as

$$\sigma_t = \begin{bmatrix} -\left(\frac{1}{2} - x\right) \\ +\left(\frac{1}{2} - x\right) \end{bmatrix}$$

into lumped nodal values result in all the four conditions from each edge canceling each other and giving a null load vector. This integration process is performed internally by Fino so the user does not need to worry about the details—but if she does then the full source code is freely available. The official NAFEMS

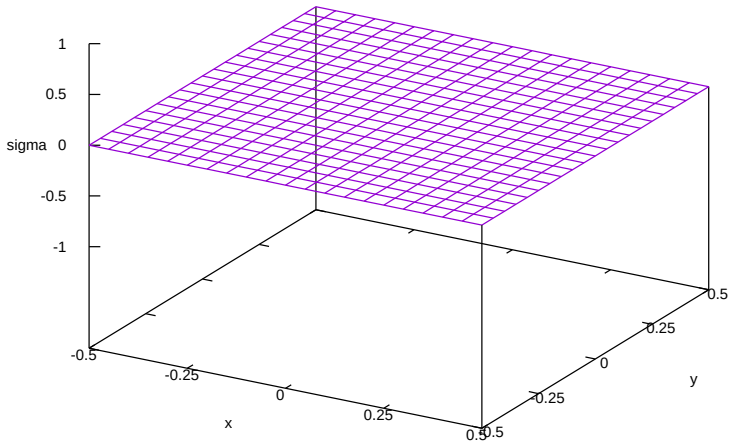


(a) Refined mesh made of 100×100 `quad4`

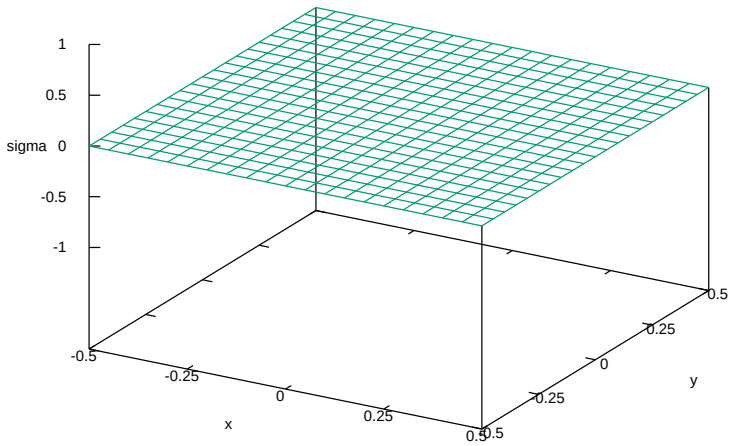


(b) Refined mesh made of 50×50 `quad8`

Figure 2: Contour plots of von Mises stresses of the refined cases “B”

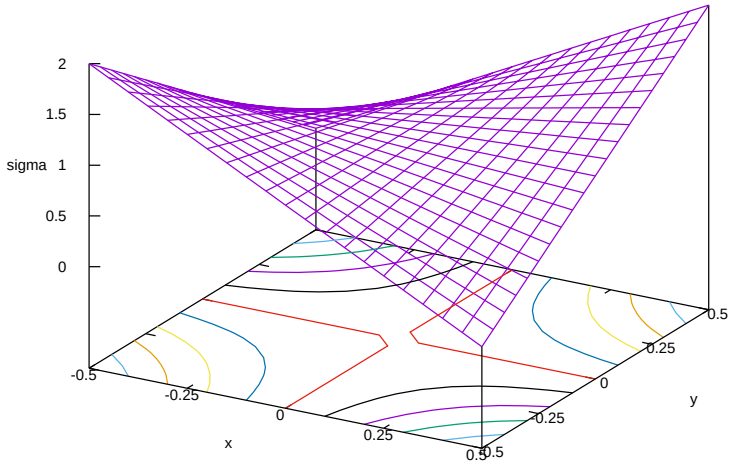


(a) `sigma`

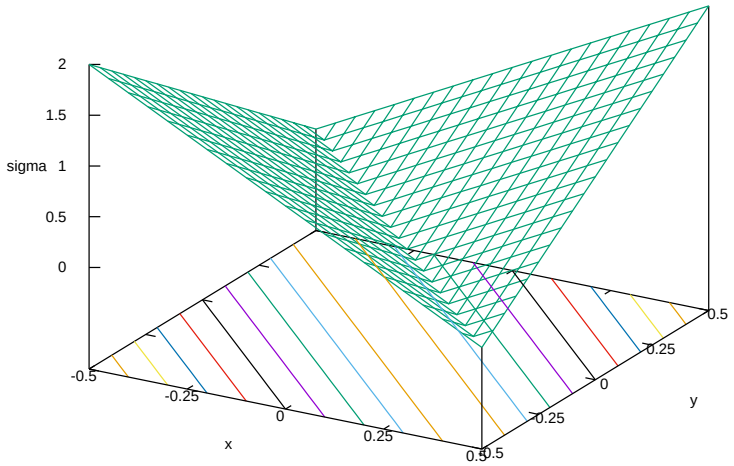


(b) `vonmises`

Figure 3: Contour plots of von Mises stresses in a single four-node quadrangle “B” case



(a) `sigma`



(b) `vonmises`

Figure 4: Contour plots of von Mises stresses in a single eight-node quadrangle “B” case

answer to the challenge also deepens into the mathematical details of this process. In effect, the right-hand side vector $\mathbf{b}$ for the `quad4` case is zero:

```
$ cat quad4A-b.m
%Vec Object: b 1 MPI processes
%  type: seq
b = [
0.0000000000000000e+00
0.0000000000000000e+00
0.0000000000000000e+00
0.0000000000000000e+00
0.0000000000000000e+00
0.0000000000000000e+00
0.0000000000000000e+00
0.0000000000000000e+00
];
$
```

Hence, the displacements and the stresses vanish identically in this case as seen in fig. 3. On the other hand, for `quad8` the lumped nodal values loads from the boundary conditions are still null for the corner nodes but non-zero for the mid-edge nodes:

```
$ cat quad8A-b.m
%Vec Object: b 1 MPI processes
%  type: seq
b = [
8.3266726846886741e-17
-8.3266726846886741e-17
-1.6711541637787068e-17
1.6711541637787068e-17
8.3266726846886741e-17
-8.3266726846886741e-17
-1.6711541637787068e-17
1.6711541637787068e-17
-3.3333333333333337e-01
3.3333333333333337e-01
3.3333333333333337e-01
-3.3333333333333337e-01
-3.3333333333333337e-01
3.3333333333333337e-01
3.3333333333333337e-01
-3.3333333333333337e-01
];
$
```

This case can recover the correct stress values at the eight nodes and, by computing the spatial derivatives of the displacements using the spatial derivatives of the shape functions, it can also recover the exact analytical solution at all the interior points of the domain. So the answer to question b. is that the `quad4` solution definitely gives wrong results at all points of the domain. The fact that the proper result for $\sigma(0,0)$ is recovered is pure chance. On the other hand, the `quad8` solution gives the right results provided the stresses are computed out of the properly-interpolated strains. This is not the case for the vast majority of

FEA programs—including Fino—as the main way of computing the von Mises stress is through `sigma(x,y)` which interpolates nodal values with the shape functions. Fig. 4 shows the difference between these to approaches, which is nevertheless significant only in this particularly-rare (and as such exquisite) challenge case proposed by Angus Ramsay. Even more, if instead of asking Fino to interpolate either `sigma` or `vonmises` inside the domain at arbitrary sampling points we asked it to export a post-processing view say in VTK format far more critical differences may occur. In effect, here is the author's rationale for including it as a NAFEMS challenge:

> **Raison d'être for the Challenge**
>
> This challenge derives from a philosophical question: can a problem be specified where the finite element response is null? At first sight it seems a little improbable that such a problem can be conceived. However, when one realises that the boundary tractions are applied to the model in the form of consistent nodal forces and that if suitable tractions are chosen such that the consistent nodal forces cancel out then such a problem is easily found. This is the case for the challenge problem when a single four-noded element is used. Further consideration of the problem shows that it possesses a theoretically exact solution which involves linear stress fields that can be captured exactly with a single eight-noded element. The theoretically exact von Mises stress at the centre of the plate is zero and therefore both the single four-noded element and the single eight-noded element predict this value correctly. Disappointingly, however, it will be seen that even though the exact solution is recovered for the single eight-noded element, the available post-processing facilities in many commercial finite element systems will not allow the user to appreciate this fact because they use linear-interpolation of nodal stress values to simplify plotting procedures.

Indeed, most of the time FEM models have hundreds of thousands of nodes and elements and post-processing software cannot afford to interpolate nodal data using the partial derivatives of the shape functions all the time. On the one hand, it might be the case that the displacements are not available so computation of the strains as

$$\frac{\partial u}{\partial x} = \sum_{j=1}^{N} \frac{\partial h_j}{\partial x} \cdot u_j$$

is not possible. On the other hand, as seen in fig. 2, the interpolated values using different schemes are smaller as the number of elements increases. So it is no surprise that post-processing software defaults to the low-order interpolation mentioned in the above paragraph. It is up to the practicing engineer to be aware of what the implications of these default behavior is.

Since the exercise asks for the stress at the origin computed using a single element and we already have seen that second-order elements give correct solutions at the nodes, an obvious next step would be to use a single element with a node at the origin, namely a `quad9`. These kind of elements with nodes at the center of each face are created by Gmsh by asking for "complete" second-order elements:

```
Merge "quad.geo";
Transfinite Line {1:4} = 1;       // make sure we have only one element
Transfinite Surface "*";
Mesh.ElementOrder = 2;            // ask for second-order...
Mesh.SecondOrderIncomplete = 0;   // ...complete elements (i.e. quad9)
```

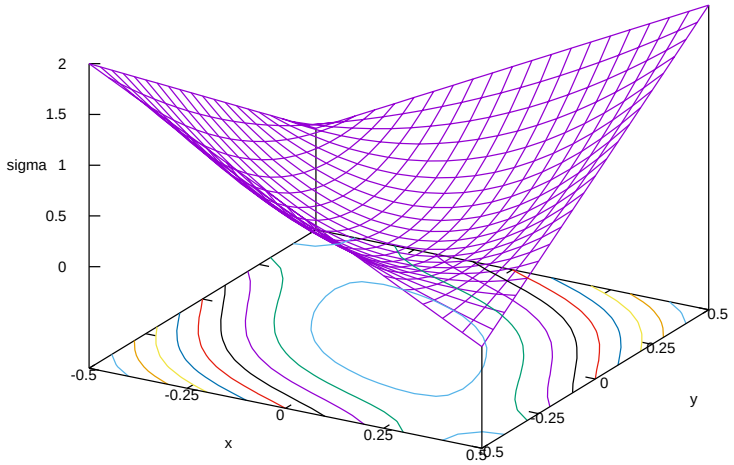
When passing `9` as the first argument to `a.fin` we get the correct stress at the origin even using `sigma(0,0)`:

```
$ fino a.fin 9 A
Case quad9A
Von Mises at origin interpolated with shape functions:  0.000
Von Mises at origin interpolated with derivatives:      0.000
$ fino a.fin 9 B
Case quad9B
Von Mises at origin interpolated with shape functions:  0.000
Von Mises at origin interpolated with derivatives:      0.000
$
```

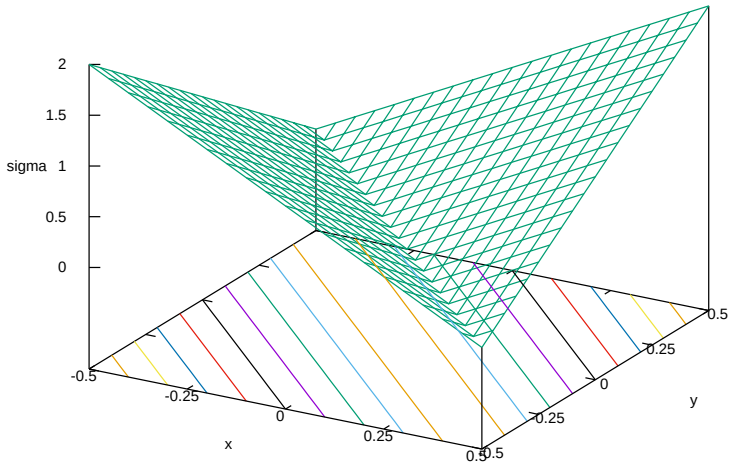
This extra degree of freedom allows the shape-interpolated function `sigma(x,y)` to approximate the physical situation far better than `quad8` fig. 5. Even more, the refined solution converges faster to the real solution since there are more nodes at the diagonal $x + y = 0$.

Lastly, as already checked in sec. 4, the stiffness matrices of the "A" cases are singular whilst those of the "B" cases have very large determinants due to the penalty method used to set the second and third displacement conditions. Yet Fino (actually PETSc) is able to find a solution to the linear system of equations in all cases, even in the refined ones.

[1] A. Ramsay, *The NAFEMS benchmark challenge volume 1.* NAFEMS, 2018.



(a) `sigma`



(b) `vonmises`

Figure 5: Contour plots of von Mises stresses in a single nine-node quadrangle “B” case