OpenCMISS-iron examples and tests used by *OpenCMISS* developers at University of Stuttgart, Germany

Christian Bleiler[*], Dr.-Ing. Nehzat Emamy[†], Andreas Hessenthaler[*], Thomas Klotz[*], Aaron Krämer[‡], Benjamin Maier[†], Sergio Morales[*], Mylena Mordhorst[*], Harry Saini[*]

September 12, 2017
17:00

CONTENTS

* Institute of Applied Mechanics (CE), University of Stuttgart, Pfaffenwaldring 7, 70569 Stuttgart, Germany
† Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
‡ Lehrstuhl Mathematische Methoden für komplexe Simulation der Naturwissenschaft und Technik, University of Stuttgart, Allmandring 5b, 70569 Stuttgart, Germany

LIST OF FIGURES

LIST OF TABLES

1 TEST SUMMARY

```
Passed tests: 176 / 214
```

1.1 Details

```
Content of: example-0001/results/failed.tests
No failed tests.

Content of: example-0001-u/results/failed.tests
current_run/l2x1x0_n8x4x0_i8_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.05804
current_run/l2x1x0_n8x4x0_i8_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.05804

Content of: example-0002/results/failed.tests
No failed tests.

Content of: example-0003/results/failed.tests
Failed tests:
current_run/l2x1x0_n2x1x0_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 44.2627
current_run/l2x1x0_n4x2x0_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 37.2770
current_run/l2x1x0_n8x4x0_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 32.2165
current_run/l2x1x0_n2x1x0_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 27.3358
current_run/l2x1x0_n4x2x0_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 22.1869
current_run/l2x1x0_n8x4x0_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 19.7449
current_run/l2x1x0_n2x1x0_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 44.2627
current_run/l2x1x0_n4x2x0_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 37.2770
current_run/l2x1x0_n8x4x0_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 32.2165
current_run/l2x1x0_n2x1x0_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 27.3358
current_run/l2x1x0_n4x2x0_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 22.1869
current_run/l2x1x0_n8x4x0_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 19.7449
current_run/l2x1x1_n2x1x1_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 124.749
current_run/l2x1x1_n4x2x2_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 128.672
current_run/l2x1x1_n8x4x4_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 143.606
current_run/l2x1x1_n2x1x1_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 94.2619
current_run/l2x1x1_n4x2x2_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 98.7606
current_run/l2x1x1_n8x4x4_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 118.047
current_run/l2x1x1_n2x1x1_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 124.749
current_run/l2x1x1_n4x2x2_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 128.672
current_run/l2x1x1_n8x4x4_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 143.606
current_run/l2x1x1_n2x1x1_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 94.2619
current_run/l2x1x1_n4x2x2_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 98.7606
current_run/l2x1x1_n8x4x4_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 118.047

Content of: example-0004/results/failed.tests
No failed tests.

Content of: example-0005/results/failed.tests
No failed tests.

Content of: example-0011/results/failed.tests
No failed tests.

Content of: example-0012/results/failed.tests
```

```
No failed tests.

Content of: example-0302-u/results/failed.tests
All tests failed.

Content of: example-0401/results/failed.tests
No failed tests.

Content of: example-0402/results/failed.tests
No failed tests.
```

2 INTRODUCTION

This document contains information about examples used for testing *OpenCMISS-iron*. Read: How-to[1] and [1].

2.1 Cmgui files for cmgui-2.9

2.2 Variations to consider

- Geometry and topology
  - 1D, 2D, 3D
  - Length, width, height
  - Number of elements
  - Interpolation order
  - Generated or user meshes
  - quad/hex or tri/tet meshes
- Initial conditions
- Load cases
  - Dirichlet BC
  - Neumann BC
  - Volume force
  - Mix of previous items
- Sources, sinks
- Time dependence
  - Static
  - Quasi-static
  - Dynamic
- Material laws
  - Linear
  - Nonlinear (Mooney-Rivlin, Neo-Hookean, Ogden, etc.)
  - Active (Stress, strain)
- Material parameters, anisotropy
- Solver
  - Direct
  - Iterative
- Test cases
  - Numerical reference data
  - Analytical solution
- A mix of previous items

1 https://bitbucket.org/hessenthaler/opencmiss-howto

2.3 Folder structure

TBD..

3 PROGRESS

People working on setting up tests in alphabetical order (surnames) with initials:

- CB : Christian Bleiler
- NE : Dr.-Ing. Nehzat Emamy
- AH : Andreas Hessenthaler
- TK : Thomas Klotz
- AK : Aaron Krämer
- BM : Benjamin Maier
- SM : Sergio Morales
- MM : Mylena Mordhorst
- HS : Harry Saini

3.1 Equations to test

Test single-physics problems before multi-physics problems!

- Diffusion equation (Laplace, Poisson, Generalized Laplace, ALE Diffusion, etc.)
- Linear elasticity equation (compressible and incompressible)
- Finite elasticity equation (compressible and incompressible Mooney-Rivlin, etc.)
- Navier-Stokes equation (ALE, Stokes, etc.)
- Monodomain equation
- CellML models
- Skeletal muscle models
- Fluid-structure interaction
- etc.

3.2 Setting up a new test

Use the following guideline to set up a new test:

1. Check if it is already there
2. Talk to other developers
3. Create a new subfolder examples/example-oxxx
4. Document the setup (computational domain, etc.) in examples/example-oxxx/doc/example.tex
5. Set up example with all parameters as command line arguments, see Section 2.2

6. Set up reference results (CHeart, Abaqus, analytical solution, etc.)
7. Set up script to run all tests in your example directory
8. Set up script to perform comparison between iron results and reference results
9. Set up visualization scripts
10. Compile, run, test, visualize your example
11. Compile, run, test, visualize all examples

For each example, progress is documented in the respective section titles with the following `TAG`:

- `DOCUMENTED`: finish the documentation of the example (spatial domain, number of time steps, boundary conditions, etc.
- `COMPILES`: example compiles (for default parameters)
- `RUNS`: example runs (for default parameters)
- `CONVERGES`: no convergence issues (for default parameters, results not plausible)
- `PLAUSIBLE`: results look sensible (for default parameters)
- `VALIDATED`: for all parameter sets it gives the correct results as compared to CHeart/Abaqus/analytical solution (includes visualization scripts, run scripts, comparison scripts, documentation!, ...)

Move all tags `CONVERGE, PLAUSIBLE` to `VALIDATED`.

Next steps include:

- Everybody runs everything!
- Meeting with Oliver
- Meeting with Auckland

3.3 Long-term goals

- Different testing targets

  `SMALL` : small, fast tests

  `BIG` : same as before; further, bigger and more complex geometries, convergence analysis

  `PARALLEL` : same as before but in parallel
- Add more examples/those which were on the agenda but not started
- Jenkins continuous testing, integration and deployment

  test `SMALL/BIG/PARALLEL` targets

  integrate with GitHub (pull-requests triggers Jenkins, merge on success)

4 DIFFUSION EQUATION

4.1 Equation in general form

The governing equation is,

$$\partial_t u + \nabla \cdot [\boldsymbol{\sigma} \nabla u] = f, \tag{1}$$

with conductivity tensor $\boldsymbol{\sigma}$. The conductivity tensor is,

- defined in material coordinates (fibre direction),
- diagonal,
- defined per element.

4.2 Example-0001 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

4.2.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1], \tag{2}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{3}$$
$$u = 1 \qquad x = 2, y = 1. \tag{4}$$

No material parameters to specify.

4.2.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{5}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{6}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{7}$$

No material parameters to specify.

4.2.3 *Computational model*

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
- Commandline arguments for tests are:
  - 2.0 1.0 0.0 2 1 0 1 0
  - 2.0 1.0 0.0 4 2 0 1 0
  - 2.0 1.0 0.0 8 4 0 1 0
  - 2.0 1.0 0.0 2 1 0 2 0
  - 2.0 1.0 0.0 4 2 0 2 0
  - 2.0 1.0 0.0 8 4 0 2 0
  - 2.0 1.0 0.0 2 1 0 1 1
  - 2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

4.2.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

`Passed tests: 24 / 24`

`No failed tests.`

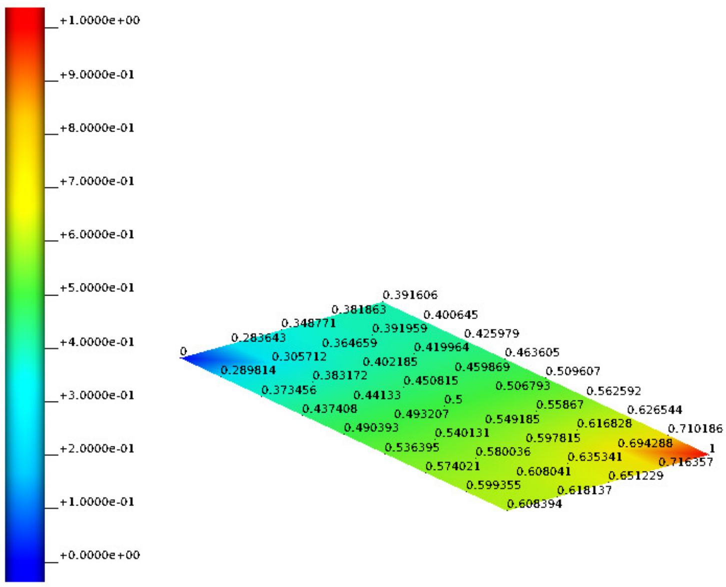

Figure 1: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

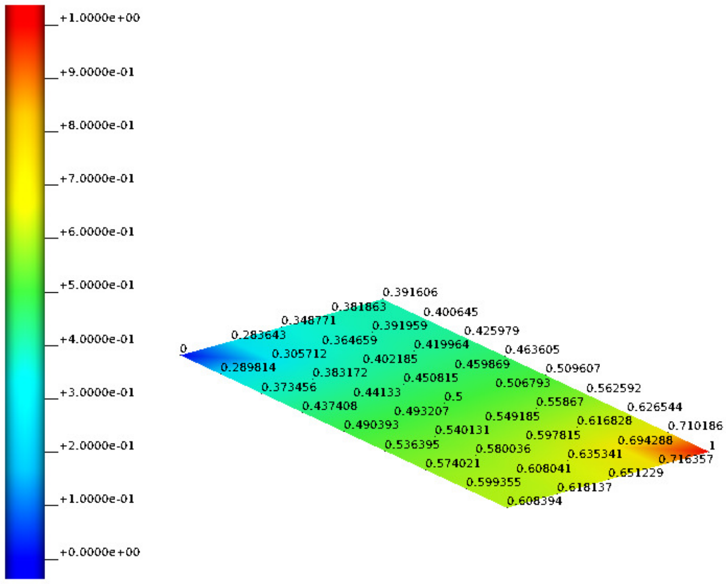

Figure 2: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

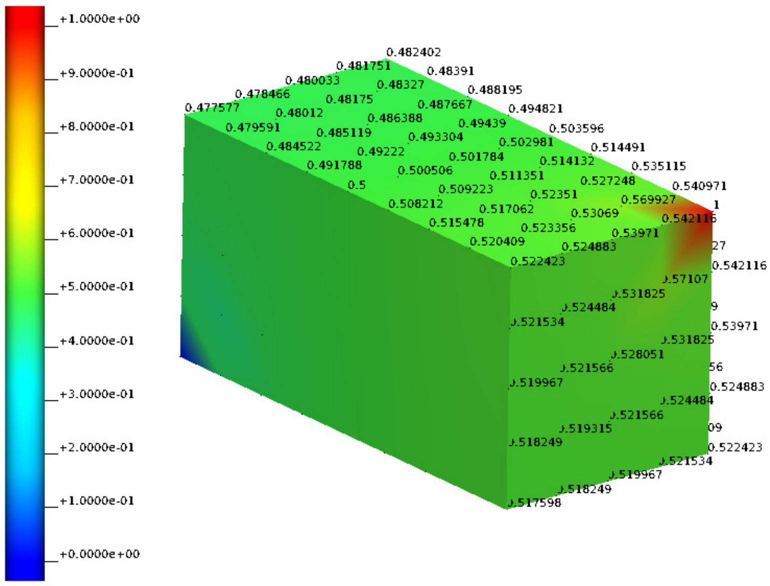

Figure 3: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

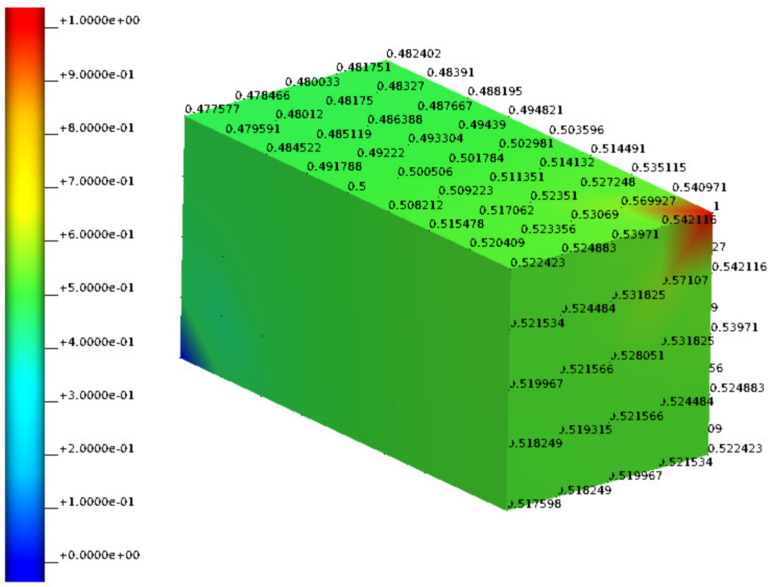

Figure 4: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

4.3 Example-0001-u `[PLAUSIBLE]`

Example uses user-defined regular meshes in CHeart mesh format and solves a static problem, i.e., applies the boundary conditions in one step.

Issues: Interpolation type 8 (quadratic simplex) gives significantly different results compared to CHeart whereas all other simplex mesh results correspond rather well..

4.3.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 2] \times [0, 1], \tag{8}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{9}$$
$$u = 1 \qquad x = 2, y = 1. \tag{10}$$

No material parameters to specify.

4.3.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 2] \times [0, 1] \times [0, 1], \tag{11}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{12}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{13}$$

No material parameters to specify.

4.3.3 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
- Commandline arguments for tests are:
  2.0 1.0 0.0 2 1 0 1 0
  2.0 1.0 0.0 4 2 0 1 0
  2.0 1.0 0.0 8 4 0 1 0
  2.0 1.0 0.0 2 1 0 2 0

2.0 1.0 0.0 4 2 0 2 0
2.0 1.0 0.0 8 4 0 2 0
2.0 1.0 0.0 8 4 0 7 0
2.0 1.0 0.0 8 4 0 8 0
2.0 1.0 0.0 2 1 0 1 1
2.0 1.0 0.0 4 2 0 1 1
2.0 1.0 0.0 8 4 0 1 1
2.0 1.0 0.0 2 1 0 2 1
2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 0.0 8 4 0 7 1
2.0 1.0 0.0 8 4 0 8 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 8 4 4 7 0
2.0 1.0 1.0 8 4 4 8 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1
2.0 1.0 1.0 8 4 4 7 1
2.0 1.0 1.0 8 4 4 8 1

- Note: Binary uses command line arguments to search for the relevant mesh files.

4.3.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 30 / 32

current_run/l2x1x0_n8x4x0_i8_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.05804
current_run/l2x1x0_n8x4x0_i8_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.05804
```

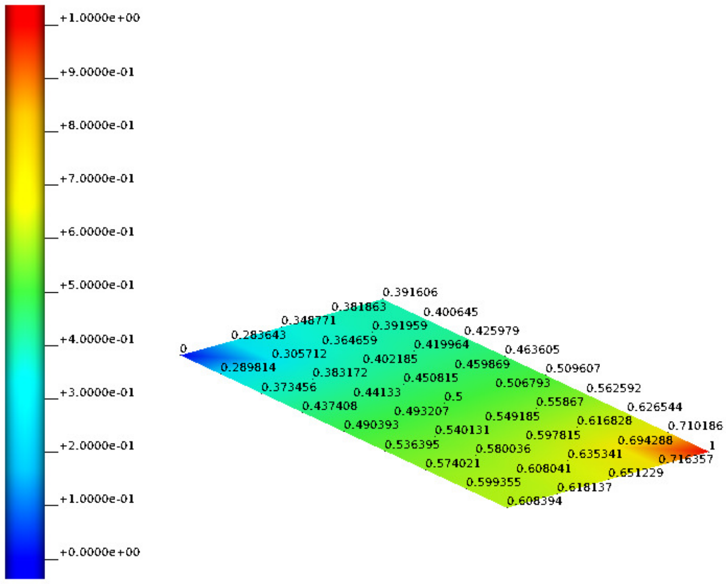

Figure 5: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

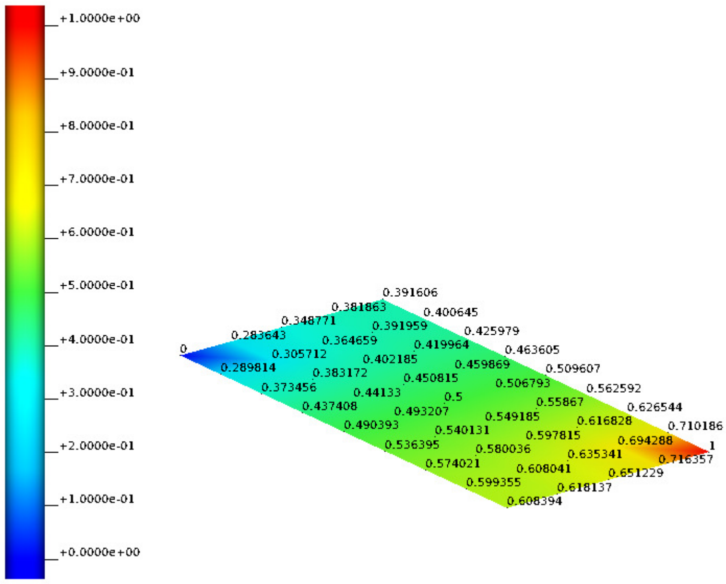

Figure 6: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

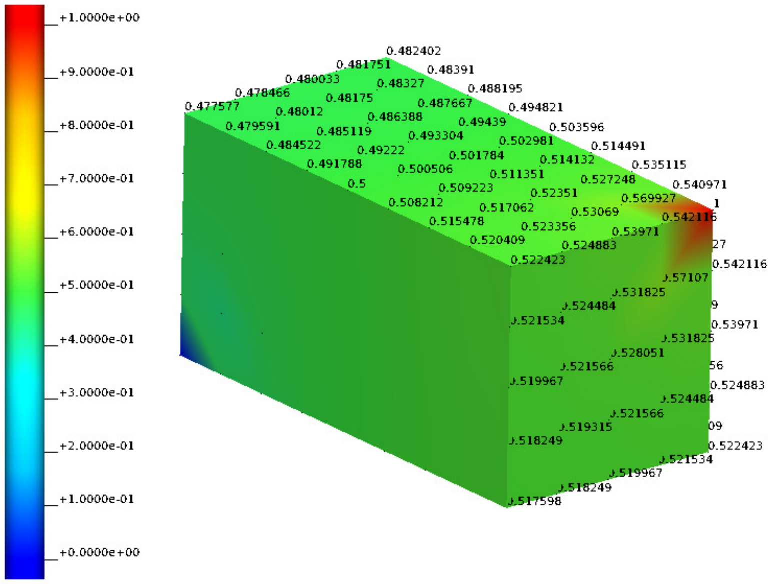

Figure 7: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

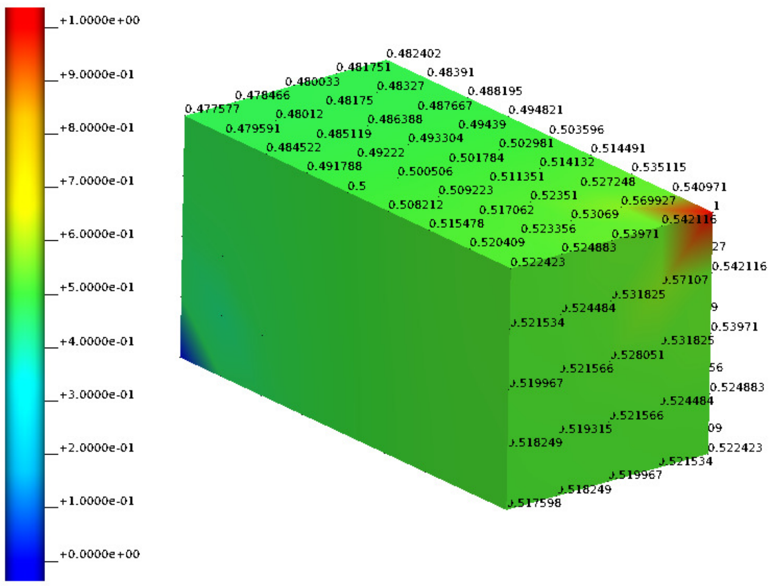

Figure 8: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

4.4 Example-0002 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

4.4.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1], \tag{14}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{15}$$
$$u = 25 - 18y \qquad x = 2. \tag{16}$$

No material parameters to specify.

4.4.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{17}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{18}$$
$$u = 25 - 18y \qquad x = 2. \tag{19}$$

No material parameters to specify.

4.4.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction

  float: length along y-direction

  float: length along z-direction (set to zero for 2D)

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0

  2.0 1.0 0.0 4 2 0 1 0

  2.0 1.0 0.0 8 4 0 1 0

  2.0 1.0 0.0 2 1 0 2 0

  2.0 1.0 0.0 4 2 0 2 0

  2.0 1.0 0.0 8 4 0 2 0

  2.0 1.0 0.0 2 1 0 1 1

  2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

4.4.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 24 / 24
```

```
No failed tests.
```

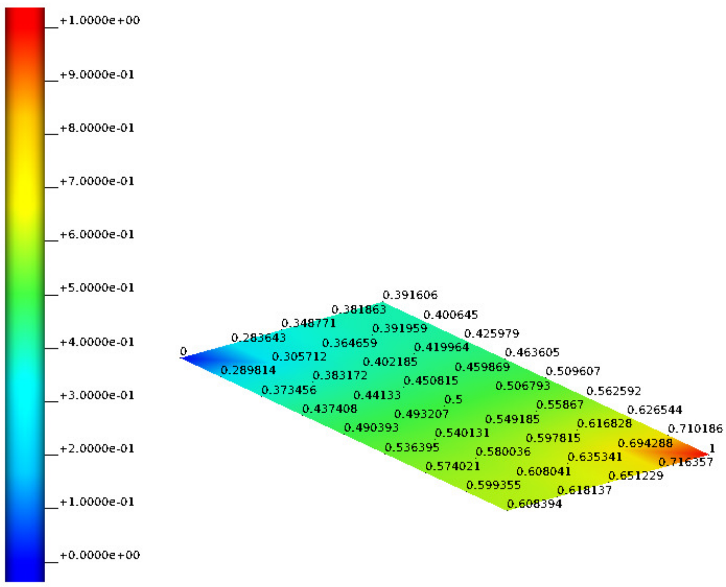

Figure 9: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

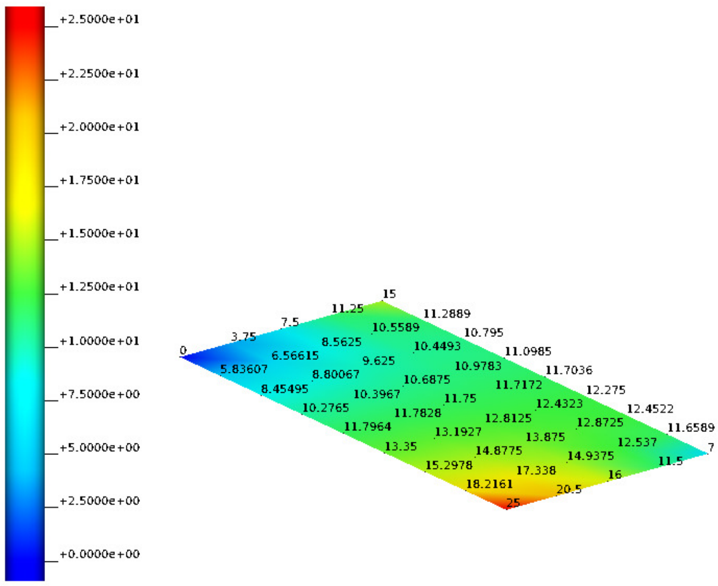

Figure 10: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

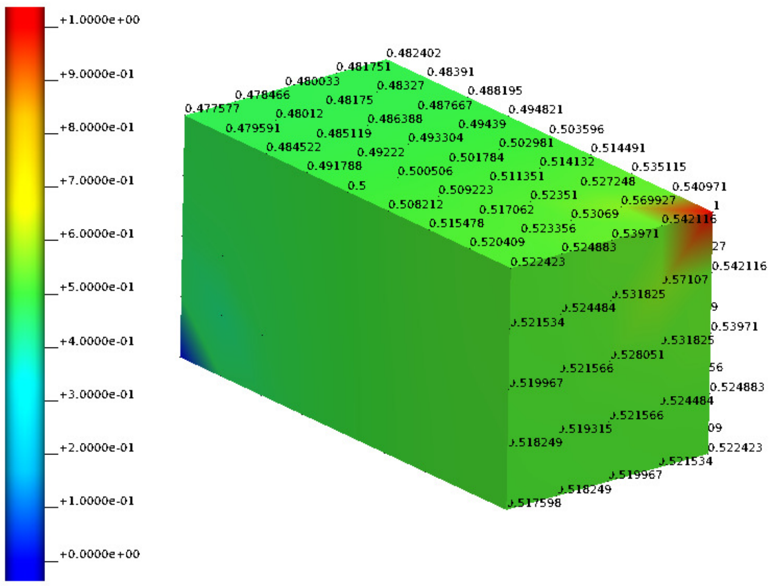

Figure 11: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

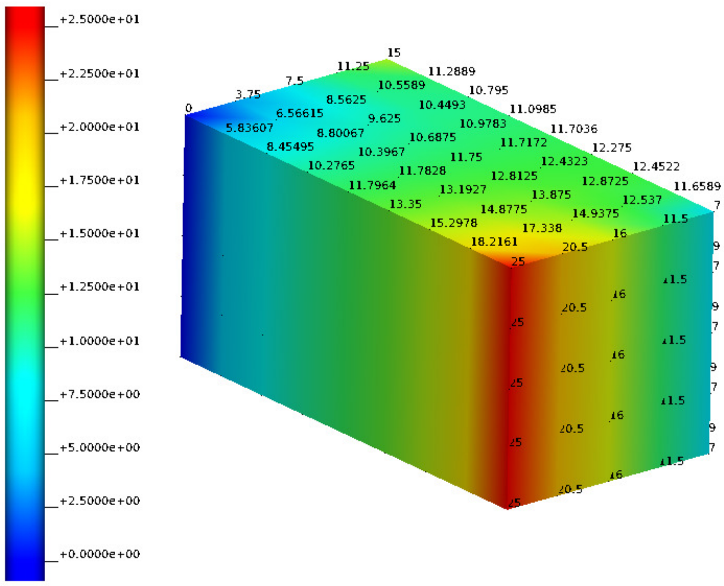

Figure 12: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

4.5 Example-0003 `[COMPILES]`

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

Issues: Not clear how to prescribe Neumann BC. Results seem weird. Reference results are set up.

4.5.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1], \tag{20}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{21}$$

$$\partial_n u = 25 - 18y \qquad x = 2. \tag{22}$$

No material parameters to specify.

4.5.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{23}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{24}$$

$$\partial_n u = 25 - 18y \qquad x = 2. \tag{25}$$

No material parameters to specify.

4.5.3 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
- Commandline arguments for tests are:
  2.0 1.0 0.0 2 1 0 1 0
  2.0 1.0 0.0 4 2 0 1 0
  2.0 1.0 0.0 8 4 0 1 0
  2.0 1.0 0.0 2 1 0 2 0
  2.0 1.0 0.0 4 2 0 2 0

2.0 1.0 0.0 8 4 0 2 0

2.0 1.0 0.0 2 1 0 1 1

2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

4.5.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 0 / 24

Failed tests:
current_run/l2x1x0_n2x1x0_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 44.2627
current_run/l2x1x0_n4x2x0_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 37.2770
current_run/l2x1x0_n8x4x0_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 32.2165
current_run/l2x1x0_n2x1x0_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 27.3358
current_run/l2x1x0_n4x2x0_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 22.1869
current_run/l2x1x0_n8x4x0_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 19.7449
current_run/l2x1x0_n2x1x0_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 44.2627
current_run/l2x1x0_n4x2x0_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 37.2770
current_run/l2x1x0_n8x4x0_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 32.2165
current_run/l2x1x0_n2x1x0_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 27.3358
current_run/l2x1x0_n4x2x0_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 22.1869
current_run/l2x1x0_n8x4x0_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 19.7449
current_run/l2x1x1_n2x1x1_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 124.749
current_run/l2x1x1_n4x2x2_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 128.672
current_run/l2x1x1_n8x4x4_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 143.606
current_run/l2x1x1_n2x1x1_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 94.2619
current_run/l2x1x1_n4x2x2_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 98.7606
current_run/l2x1x1_n8x4x4_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 118.047
current_run/l2x1x1_n2x1x1_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 124.749
current_run/l2x1x1_n4x2x2_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 128.672
current_run/l2x1x1_n8x4x4_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 143.606
current_run/l2x1x1_n2x1x1_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 94.2619
```

```
current_run/l2x1x1_n4x2x2_i2_s1/Example.part0.exnode        | CHeart    - Iron |_2 = 98.7606
current_run/l2x1x1_n8x4x4_i2_s1/Example.part0.exnode        | CHeart    - Iron |_2 = 118.047
```

Figure 13: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

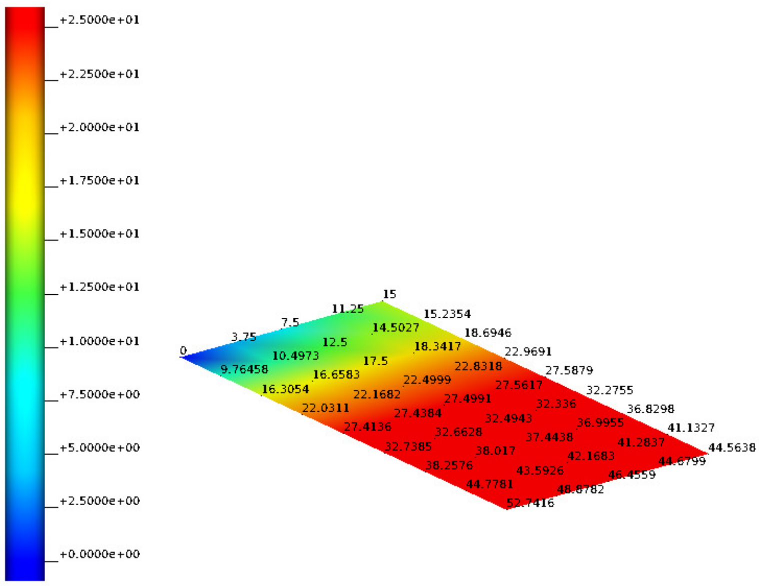

Figure 14: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 15: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

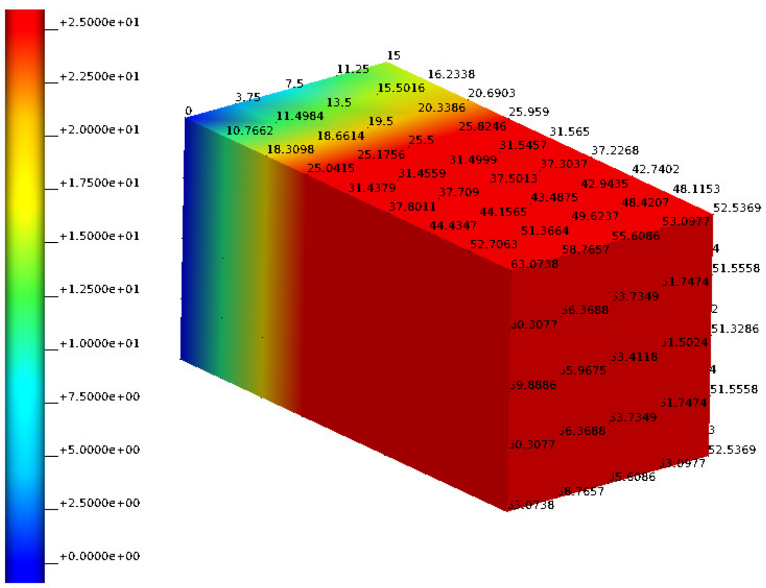

Figure 16: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

4.6 Example-0004 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

4.6.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 2] \times [0, 1], \tag{26}$$

with boundary conditions

$$u = 2.0e^x \cdot \cos(y) \qquad \text{on } \partial\Omega. \tag{27}$$

No material parameters to specify.

4.6.2 *Computational model*

- Commandline arguments are:

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  4 2 0 1 0

  8 4 0 1 0

  2 1 0 2 0

  4 2 0 2 0

  8 4 0 2 0

  4 2 0 1 1

  8 4 0 1 1

  2 1 0 2 1

  4 2 0 2 1

  8 4 0 2 1

  100 50 0 1 0 (not tested yet..)

  100 50 0 2 0 (not tested yet..)

  100 50 0 1 1 (not tested yet..)

  100 50 0 2 1 (not tested yet..)

4.6.3 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 10 / 10
```

```
No failed tests.
```

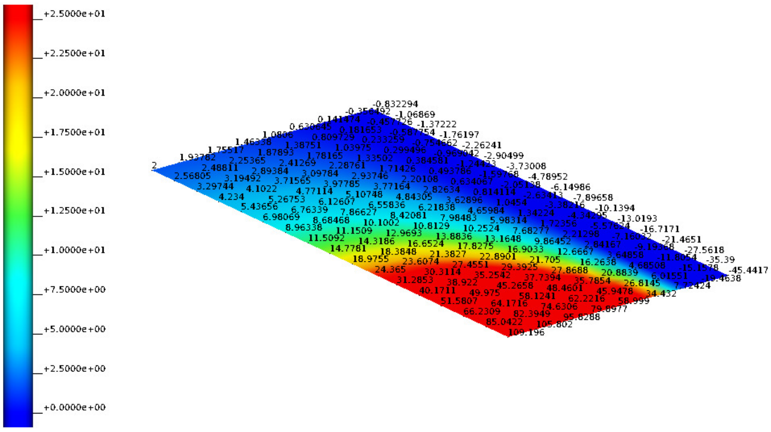

Figure 17: 2D results, iron reference w/ command line arguments [8 4 0 2 0].

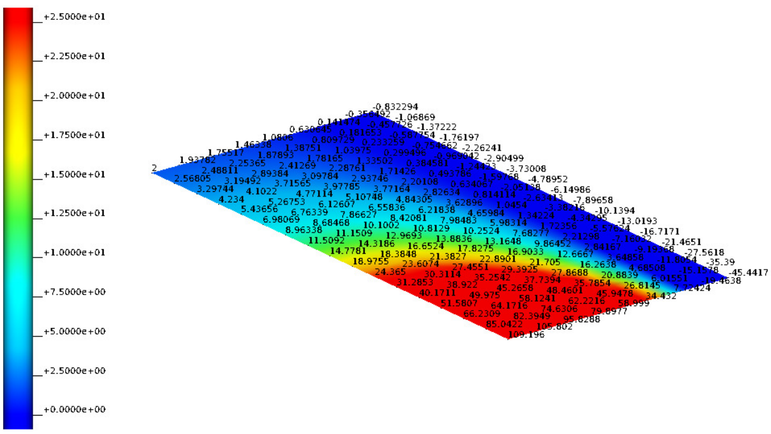

Figure 18: 2D results, current run w/ command line arguments [8 4 0 2 0].

4.7 Example-0005 [VALIDATED]

Example uses two user-defined (2d and 3d) unregular meshes and solves a patch test in form of a static problem, i.e., applies the boundary conditions in one step. Here, a Laplace problem with two Dirichlet boundary conditions is solved. The analytical solution for this setting is known and results in a linear distribution of the scalar Laplace parameter u. The numerical solution has to recover this the linear distribution in order to pass the Patch test. 2d and 3d meshes are according to MacNeil and Harder (1985).

4.7.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 0.24] \times [0, 0.12], \tag{28}$$

with boundary conditions

$$u = 0 \qquad x = 0, \tag{29}$$
$$u = 1 \qquad x = 0.24. \tag{30}$$

No material parameters to specify.

4.7.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 1] \times [0, 1] \times [0, 1], \tag{31}$$

with boundary conditions

$$u = 0 \qquad x = 0, \tag{32}$$
$$u = 1 \qquad x = 1. \tag{33}$$

No material parameters to specify.

4.7.3 *Computational model*

- Commandline arguments are:

  integer: dimension (2: 2d; 3: 3d)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2 1 0

  2 2 0

  2 1 1

  2 2 1

  3 1 0

  3 2 0

  3 1 1

  3 2 1

4.7.4 *Result summary*

Since the analytical result is known and the numerical results have to recover the linear distribution of u, the comparisons are done with the analytical solution.

```
Passed tests: 8 / 8

No failed tests.
```

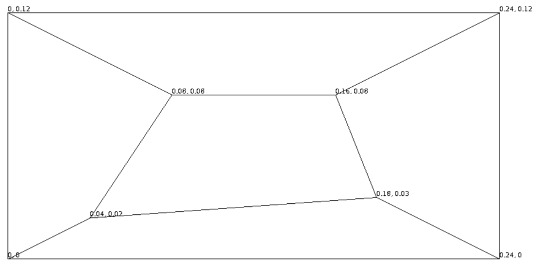

Figure 19: 2D geometry and mesh.

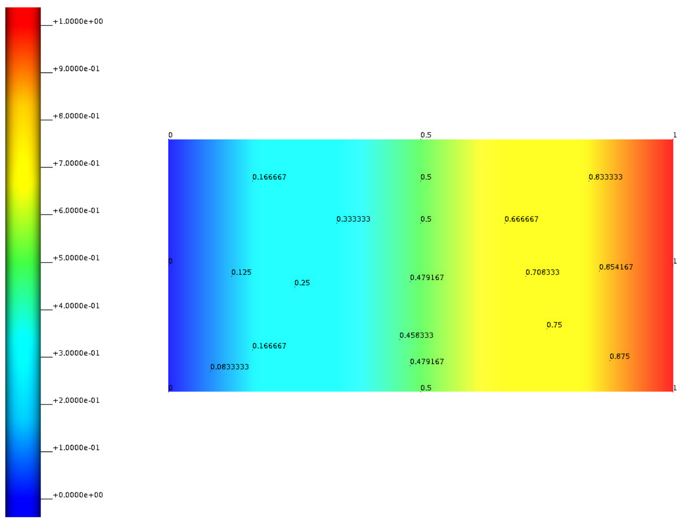

Figure 20: 2D results, iron reference w/ command line arguments [2 2 0].

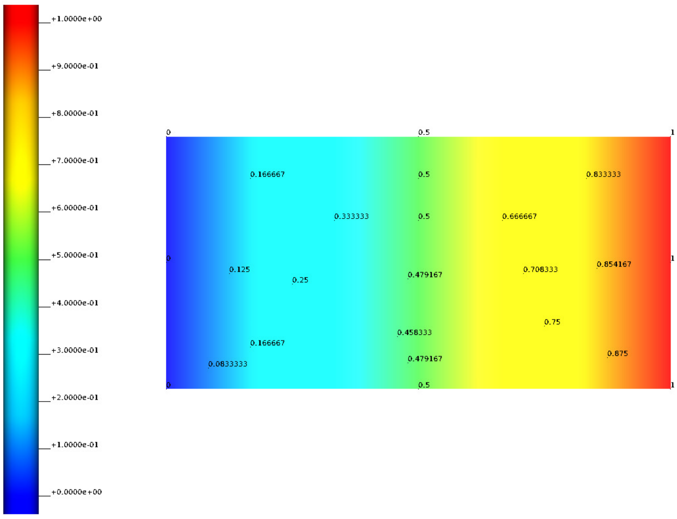

Figure 21: 2D results, current run w/ command line arguments [2 2 0].

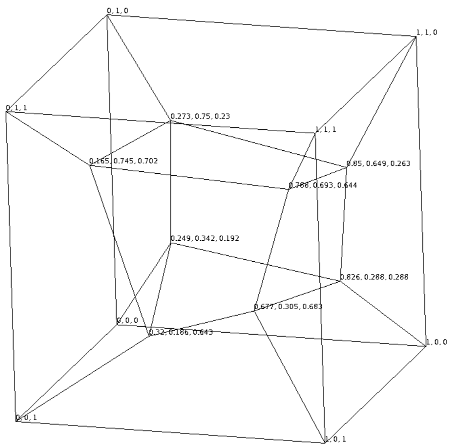

Figure 22: 3D geometry (unit cube) and mesh.

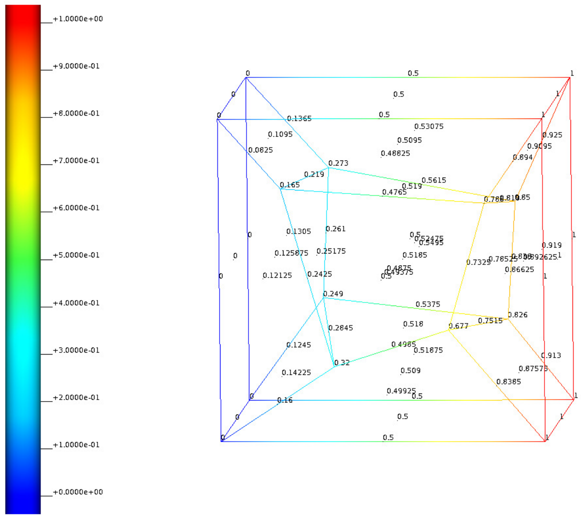

Figure 23: 3D results, iron reference w/ command line arguments [3 2 0].

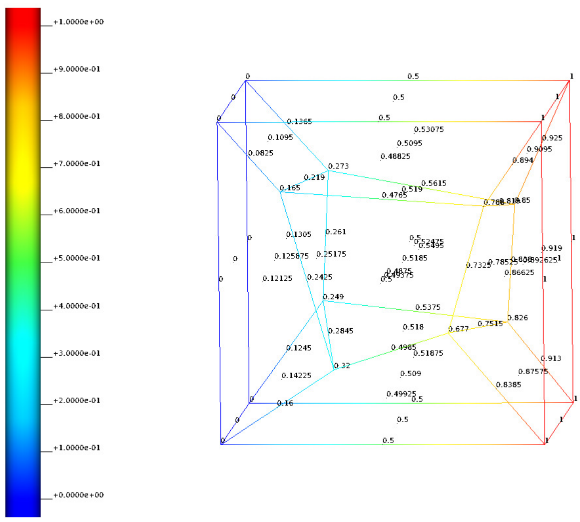

Figure 24: 3D results, current run w/ command line arguments [3 2 0].

4.8 Example-0011 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

4.8.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1], \tag{34}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{35}$$
$$u = 1 \qquad x = 2, y = 1. \tag{36}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \mathbf{I}. \tag{37}$$

4.8.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{38}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{39}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{40}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \mathbf{I}. \tag{41}$$

4.8.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
  float: σ_{11}
  float: σ_{22}
  float: σ_{33} (ignored for 2D)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0 1 1
  2.0 1.0 0.0 4 2 0 1 0 1 1
  2.0 1.0 0.0 8 4 0 1 0 1 1
  2.0 1.0 0.0 2 1 0 2 0 1 1
  2.0 1.0 0.0 4 2 0 2 0 1 1
  2.0 1.0 0.0 8 4 0 2 0 1 1
  2.0 1.0 0.0 2 1 0 1 1 1 1
  2.0 1.0 0.0 4 2 0 1 1 1 1
  2.0 1.0 0.0 8 4 0 1 1 1 1
  2.0 1.0 0.0 2 1 0 2 1 1 1
  2.0 1.0 0.0 4 2 0 2 1 1 1
  2.0 1.0 0.0 8 4 0 2 1 1 1
  2.0 1.0 1.0 2 1 1 1 0 1 1 1
  2.0 1.0 1.0 4 2 2 1 0 1 1 1
  2.0 1.0 1.0 8 4 4 1 0 1 1 1
  2.0 1.0 1.0 2 1 1 2 0 1 1 1
  2.0 1.0 1.0 4 2 2 2 0 1 1 1
  2.0 1.0 1.0 8 4 4 2 0 1 1 1
  2.0 1.0 1.0 2 1 1 1 1 1 1 1
  2.0 1.0 1.0 4 2 2 1 1 1 1 1
  2.0 1.0 1.0 8 4 4 1 1 1 1 1
  2.0 1.0 1.0 2 1 1 2 1 1 1 1
  2.0 1.0 1.0 4 2 2 2 1 1 1 1
  2.0 1.0 1.0 8 4 4 2 1 1 1 1

4.8.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 24 / 24
```

```
No failed tests.
```

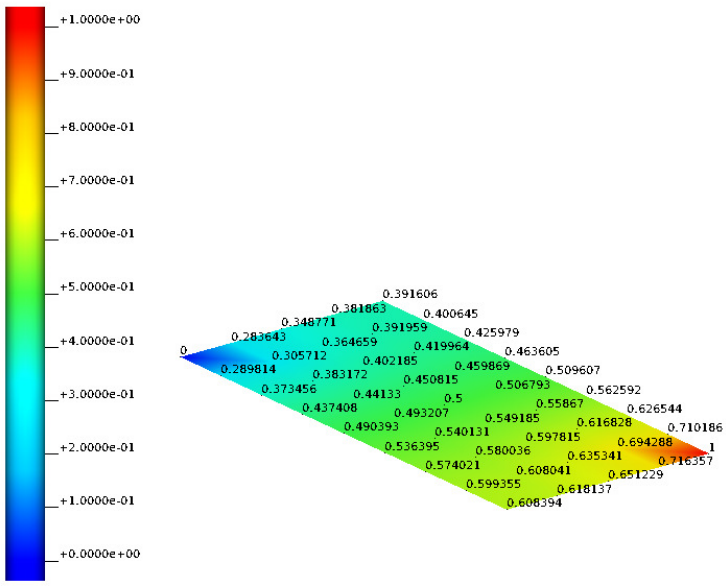

Figure 25: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].

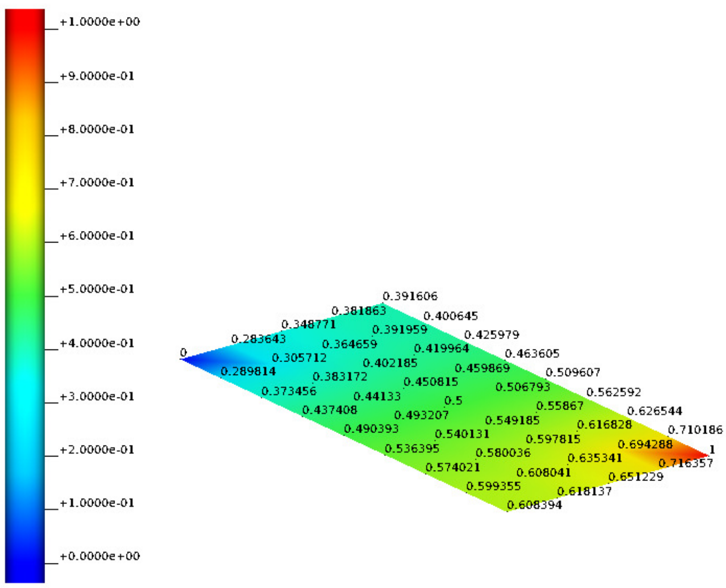

Figure 26: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].

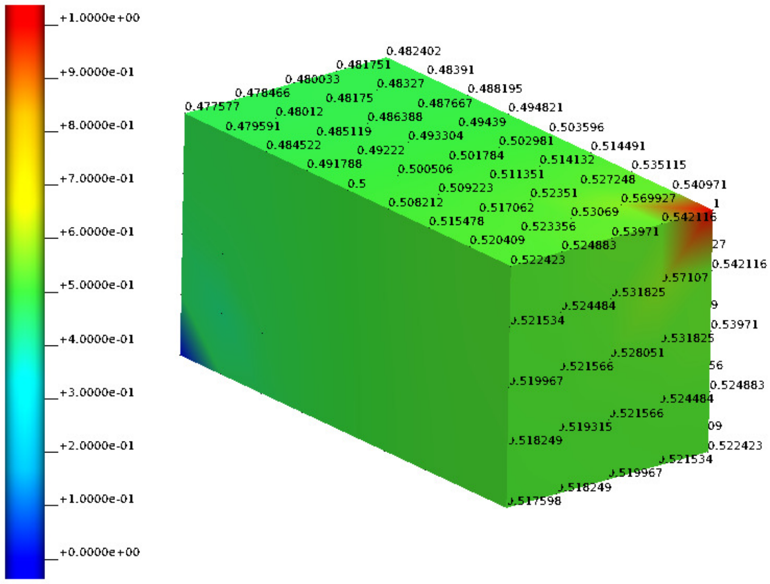

Figure 27: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1].

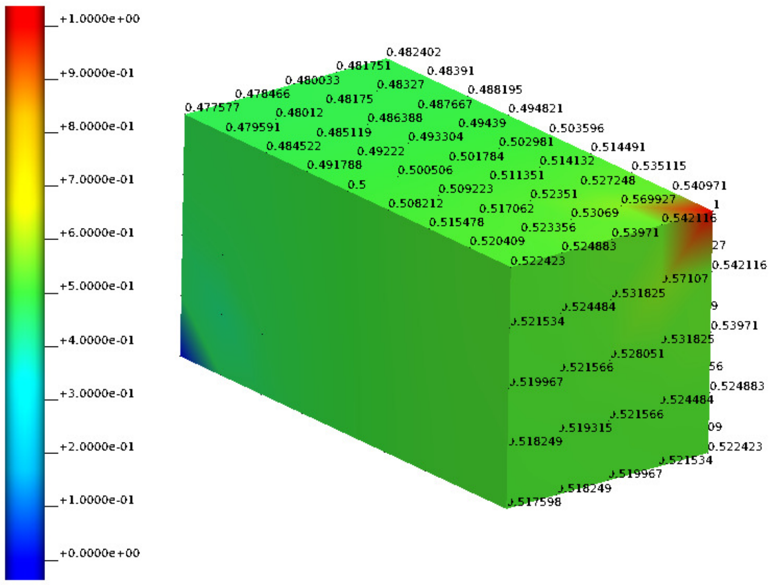

Figure 28: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1].

4.9 Example-0012 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

4.9.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1], \tag{42}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{43}$$

$$u = 1 \qquad x = 2, y = 1. \tag{44}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}. \tag{45}$$

4.9.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{46}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{47}$$

$$u = 1 \qquad x = 2, y = z = 1. \tag{48}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 7 \end{bmatrix}. \tag{49}$$

4.9.3 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
  float: σ_{11}
  float: σ_{22}
  float: σ_{33} (ignored for 2D)
  float: angle 1
  float: angle 2
  float: angle 3

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0 2 3 0 0 0 0
  2.0 1.0 0.0 4 2 0 1 0 2 3 0 0 0 0
  2.0 1.0 0.0 8 4 0 1 0 2 3 0 0 0 0
  2.0 1.0 0.0 2 1 0 2 0 2 3 0 0 0 0
  2.0 1.0 0.0 4 2 0 2 0 2 3 0 0 0 0
  2.0 1.0 0.0 8 4 0 2 0 2 3 0 0 0 0
  2.0 1.0 0.0 2 1 0 1 1 2 3 0 0 0 0
  2.0 1.0 0.0 4 2 0 1 1 2 3 0 0 0 0
  2.0 1.0 0.0 8 4 0 1 1 2 3 0 0 0 0
  2.0 1.0 0.0 2 1 0 2 1 2 3 0 0 0 0
  2.0 1.0 0.0 4 2 0 2 1 2 3 0 0 0 0
  2.0 1.0 0.0 8 4 0 2 1 2 3 0 0 0 0
  2.0 1.0 1.0 2 1 1 1 0 2 3 7 0 0 0
  2.0 1.0 1.0 4 2 2 1 0 2 3 7 0 0 0
  2.0 1.0 1.0 8 4 4 1 0 2 3 7 0 0 0
  2.0 1.0 1.0 2 1 1 2 0 2 3 7 0 0 0
  2.0 1.0 1.0 4 2 2 2 0 2 3 7 0 0 0
  2.0 1.0 1.0 8 4 4 2 0 2 3 7 0 0 0
  2.0 1.0 1.0 2 1 1 1 1 2 3 7 0 0 0
  2.0 1.0 1.0 4 2 2 1 1 2 3 7 0 0 0
  2.0 1.0 1.0 8 4 4 1 1 2 3 7 0 0 0
  2.0 1.0 1.0 2 1 1 2 1 2 3 7 0 0 0
  2.0 1.0 1.0 4 2 2 2 1 2 3 7 0 0 0
  2.0 1.0 1.0 8 4 4 2 1 2 3 7 0 0 0

4.9.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 24 / 24
```

```
No failed tests.
```

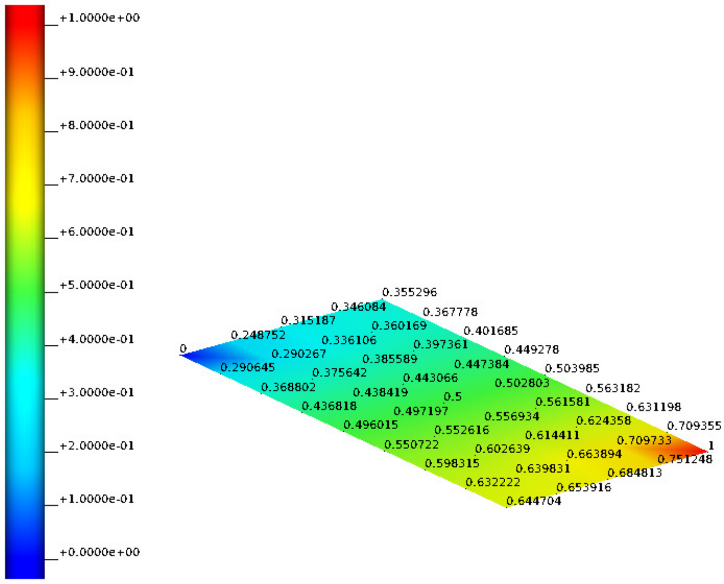

Figure 29: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 2 3 0 0 0 0].

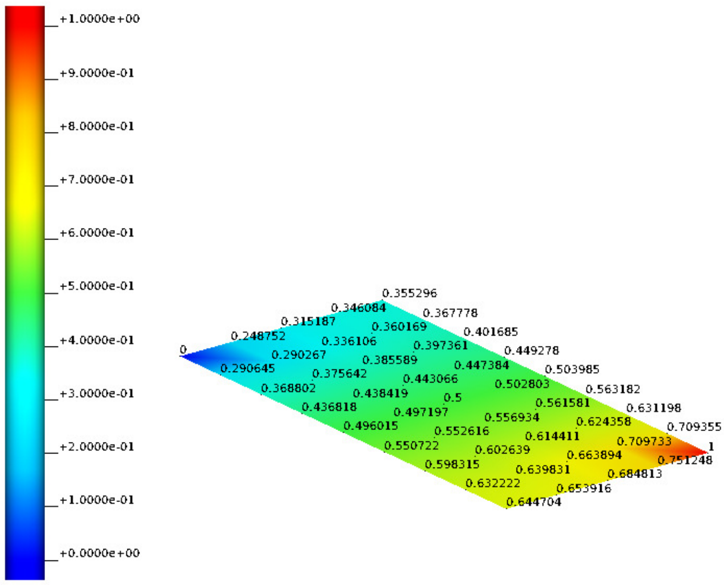

Figure 30: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 2 3 0 0 0 0].

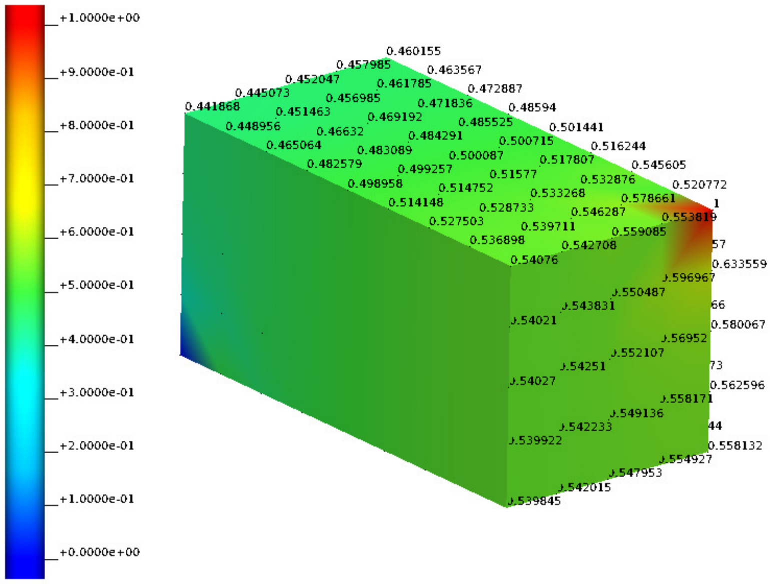

Figure 31: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 2 3 7 0 0 0].

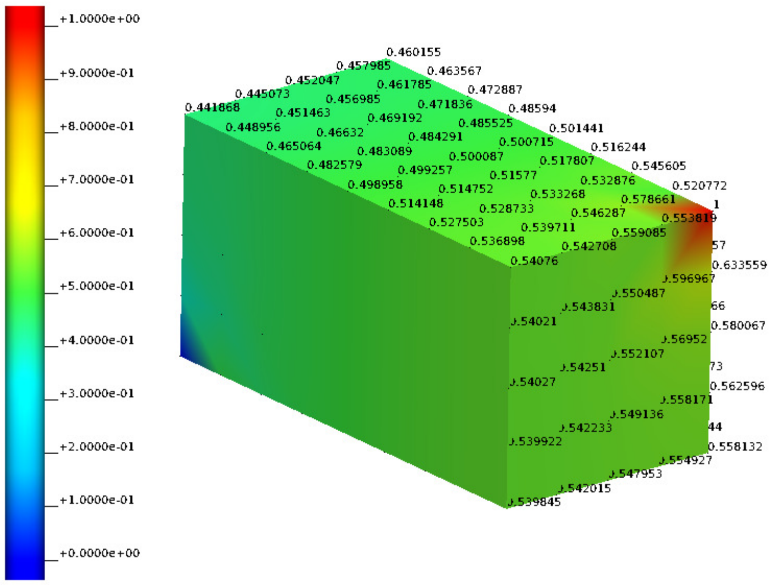

Figure 32: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 2 3 7 0 0 0].

4.10 Example-0013 [PLAUSIBLE]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.
ISSUE: 3D results are off. Probably due to fibre rotation bug!

4.10.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1], \tag{50}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{51}$$
$$u = 1 \qquad x = 2, y = 1. \tag{52}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}. \tag{53}$$

Rotation of fibres by 30 degrees (first angle).

4.10.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{54}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{55}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{56}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 7 \end{bmatrix}. \tag{57}$$

Rotation of fibres by (30, 40, 10) degrees.

4.10.3 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
  float: σ_{11}

float: σ_{22}

float: σ_{33} (ignored for 2D)

float: angle 1

float: angle 2

float: angle 3

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 4 2 0 1 0 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 8 4 0 1 0 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 2 1 0 2 0 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 4 2 0 2 0 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 8 4 0 2 0 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 2 1 0 1 1 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 4 2 0 1 1 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 8 4 0 1 1 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 2 1 0 2 1 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 4 2 0 2 1 2 3 0 0.523598775598299 0 0

  2.0 1.0 0.0 8 4 0 2 1 2 3 0 0.523598775598299 0 0

  2.0 1.0 1.0 2 1 1 1 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 4 2 2 1 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 8 4 4 1 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 2 1 1 2 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 4 2 2 2 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 8 4 4 2 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 2 1 1 1 1 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 4 2 2 1 1 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 8 4 4 1 1 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 2 1 1 2 1 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 4 2 2 2 1 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

  2.0 1.0 1.0 8 4 4 2 1 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433

4.10.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 12 / 24

Failed tests:
current_run/l2x1x1_n2x1x1_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.29720
current_run/l2x1x1_n4x2x2_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.25938
current_run/l2x1x1_n8x4x4_i1_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.27804
current_run/l2x1x1_n2x1x1_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.20837
current_run/l2x1x1_n4x2x2_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.21308
current_run/l2x1x1_n8x4x4_i2_s0/Example.part0.exnode       | CHeart   - Iron |_2 = 0.25631
current_run/l2x1x1_n2x1x1_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.29720
current_run/l2x1x1_n4x2x2_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.25938
current_run/l2x1x1_n8x4x4_i1_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.27804
current_run/l2x1x1_n2x1x1_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.20837
current_run/l2x1x1_n4x2x2_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.21308
current_run/l2x1x1_n8x4x4_i2_s1/Example.part0.exnode       | CHeart   - Iron |_2 = 0.25631
```

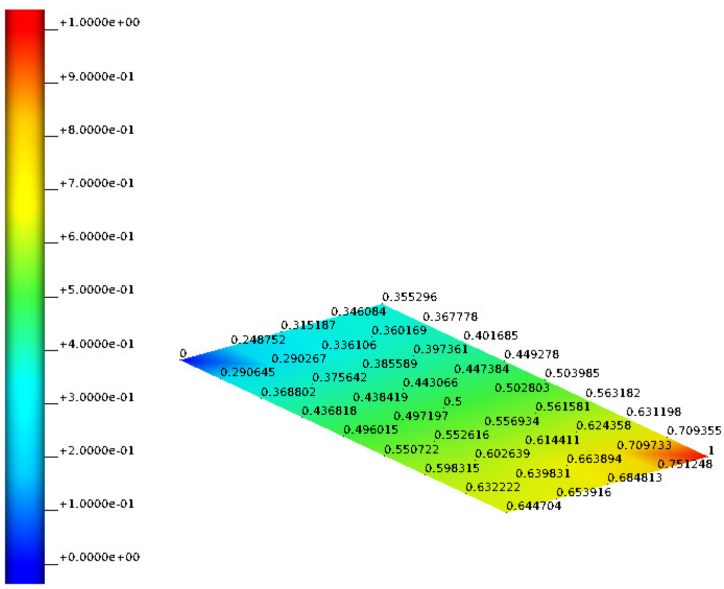

Figure 33: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 2 3 0 0.523598775598299 0 0].

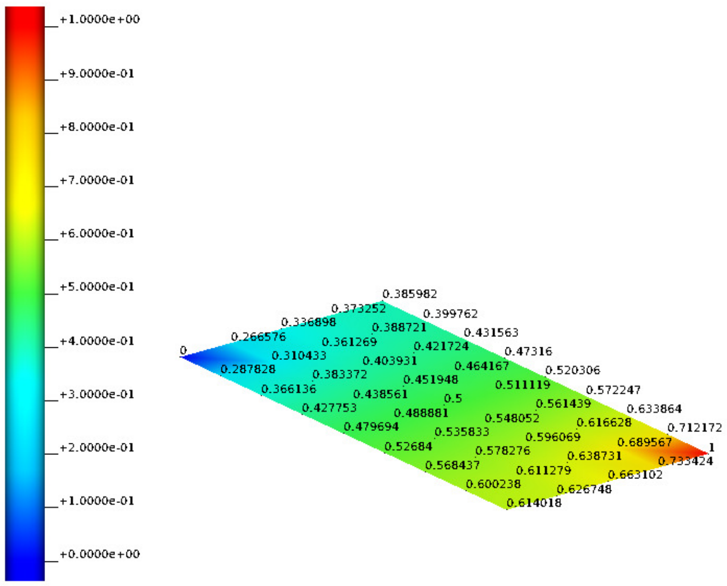

Figure 34: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 2 3 0 0.5235987755598299 0 0].

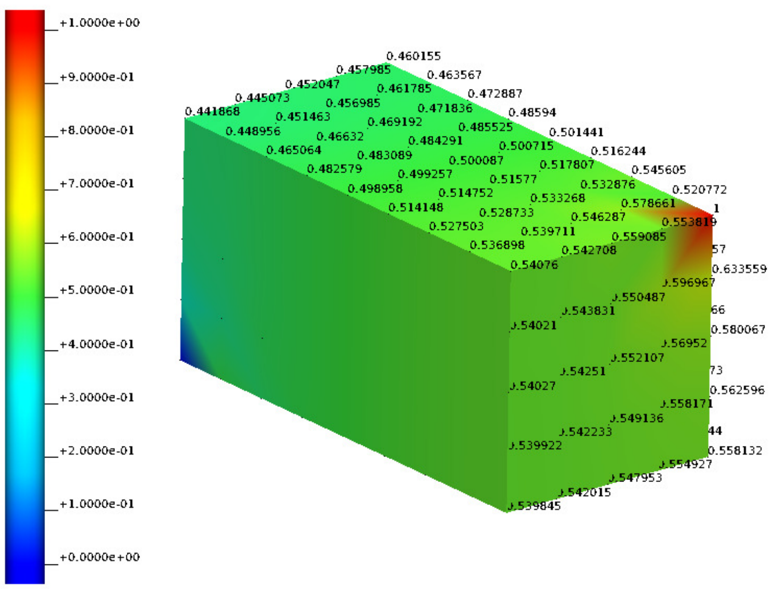

Figure 35: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 2 3 7 0.5235987755598299 0.6981317007977332 0.174532925199433].

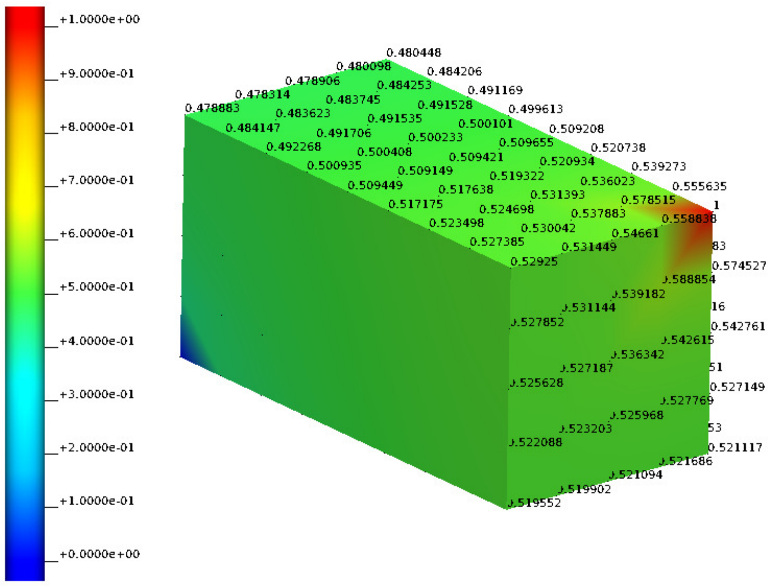

Figure 36: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 2 3 7 0.523598775598299 0.698131700797732 0.174532925199433].

5 LINEAR ELASTICITY

5.1 Equation in general form

$$\partial_{tt}\mathbf{u} + \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \tag{58}$$

5.2 Example-0101 [PLAUSIBLE]

5.2.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{0} \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \tag{59}$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad x = y = 0, \tag{60}$$

$$u_x = 16 \qquad x = 160, \tag{61}$$

and in 3D by

$$u_x = u_y = u_z = 0 \qquad x = y = z = 0, \tag{62}$$

$$u_x = 16 \qquad x = 160. \tag{63}$$

The material parameters are

$$E = 10000 \texttt{MPa}, \tag{64}$$

$$\nu = 0.3, \tag{65}$$

$$\rho = 5 \times 10^{-9} \texttt{tonne.mm}^3. \tag{66}$$

5.2.2 *Computational model*

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
  - float: elastic modulus
  - float: Poisson ratio
  - float: displacement percentage load
- Command line arguments for tests are:
  - 160 120 0 8 6 0 1 0 10000 0.3 0.05
  - 160 120 0 16 12 0 1 0 10000 0.3 0.05
  - 160 120 0 32 24 0 1 0 10000 0.3 0.05
  - 160 120 120 8 6 6 1 0 10000 0.3 0.05
  - 160 120 120 16 12 12 1 0 10000 0.3 0.05
  - 160 120 120 32 24 24 1 0 10000 0.3 0.05
  - 160 120 0 8 6 0 2 0 10000 0.3 0.05
  - 160 120 0 16 12 0 2 0 10000 0.3 0.05

160 120 0 32 24 0 2 0 10000 0.3 0.05
160 120 120 8 6 6 2 0 10000 0.3 0.05
160 120 120 16 12 12 2 0 10000 0.3 0.05
160 120 120 32 24 24 2 0 10000 0.3 0.05

5.2.3 *Results*

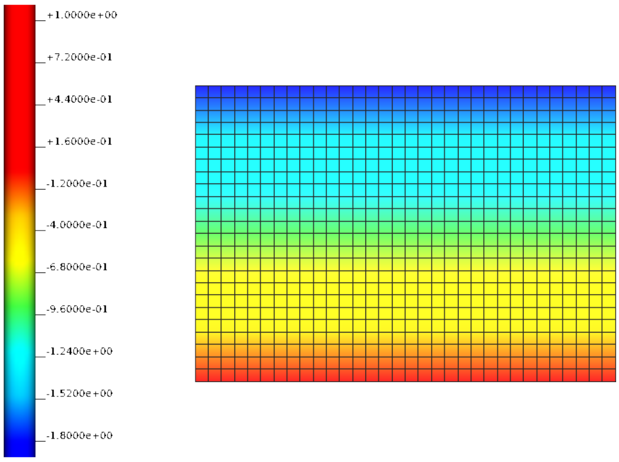

Figure 37: Results, iron 2D fine mesh.

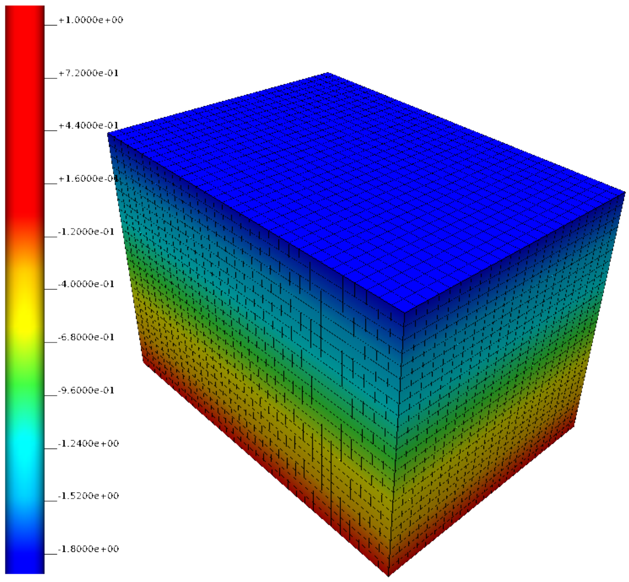

Figure 38: Results, iron 3D fine mesh.

5.2.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_y along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accoding to

$$\mathtt{L_2\text{-}norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\mathtt{abaqus}} - u^i_{y,\mathtt{iron}}\right)^2}, \tag{67}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.

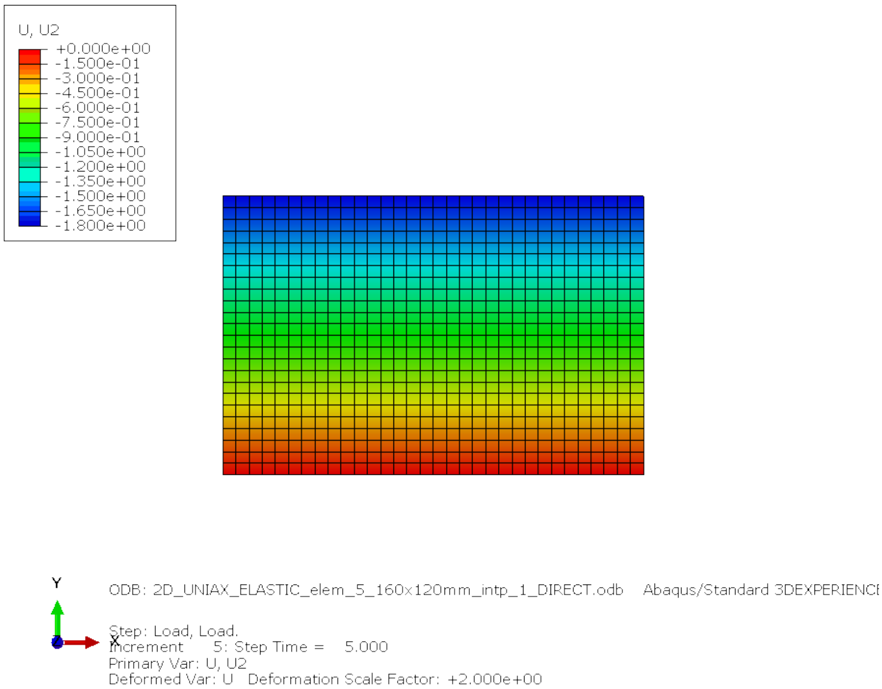

Figure 39: Results, Abaqus 2D fine mesh.

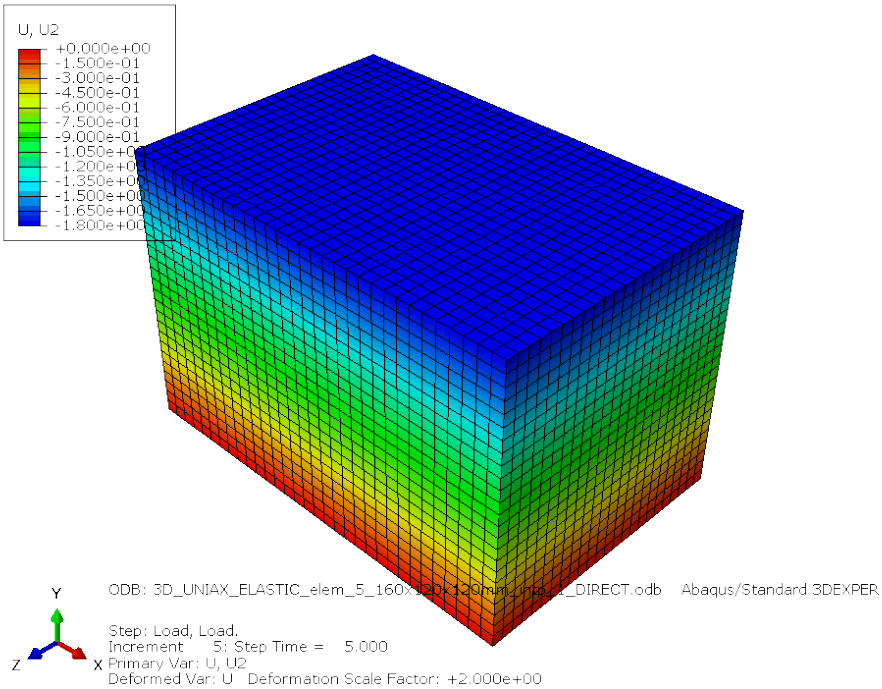

Figure 40: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	5.322×10^{-16}	Linear
2D	Medium	1.559×10^{-15}	Linear
2D	Fine	2.900×10^{-15}	Linear
3D	Coarse	3.071×10^{-17}	Linear
3D	Medium	2.125×10^{-17}	Linear
3D	Fine	2.924×10^{-17}	Linear
2D	Coarse	9.728×10^{-16}	Quadratic
2D	Medium	2.039×10^{-15}	Quadratic
2D	Fine	2.159×10^{-15}	Quadratic
3D	Coarse	6.687×10^{-16}	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 1: Quantiative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extenions

5.3 Example-0102 [PLAUSIBLE]

5.3.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = 0 \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \tag{68}$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad y = 0, \tag{69}$$

$$u_y = 8 \qquad x = 160, \tag{70}$$

and in 3D by

$$u_x = u_z = 0 \qquad x = 0, \tag{71}$$

$$u_y = 0 \qquad y = 0, \tag{72}$$

$$u_x = 160 \qquad x = 160, \tag{73}$$

$$u_y = 8 \qquad x = 160. \tag{74}$$

The material parameters are

$$E = 10000\texttt{MPa}, \tag{75}$$

$$\nu = 0.3, \tag{76}$$

$$\rho = 5 \times 10^{-9}\texttt{tonne.mm}^3. \tag{77}$$

5.3.2 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
  float: elastic modulus
  float: Poisson ratio
  float: displacement percentage load
- Command line arguments for tests are:
  160 120 0 8 6 0 1 0 10000 0.3 0.05
  160 120 0 16 12 0 1 0 10000 0.3 0.05
  160 120 0 32 24 0 1 0 10000 0.3 0.05
  160 120 120 8 6 6 1 0 10000 0.3 0.05
  160 120 120 16 12 12 1 0 10000 0.3 0.05
  160 120 120 32 24 24 1 0 10000 0.3 0.05

160 120 0 8 6 0 2 0 10000 0.3 0.05

160 120 0 16 12 0 2 0 10000 0.3 0.05

160 120 0 32 24 0 2 0 10000 0.3 0.05

160 120 120 8 6 6 2 0 10000 0.3 0.05

160 120 120 16 12 12 2 0 10000 0.3 0.05

160 120 120 32 24 24 2 0 10000 0.3 0.05

5.3.3 *Results*

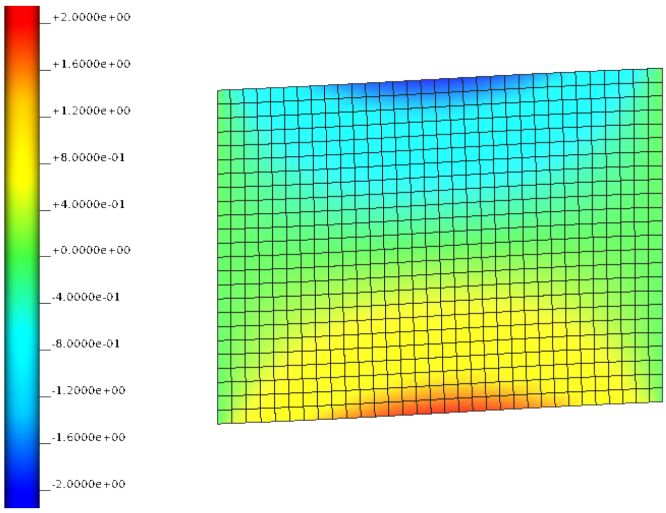

Figure 41: Results, iron 2D fine mesh.

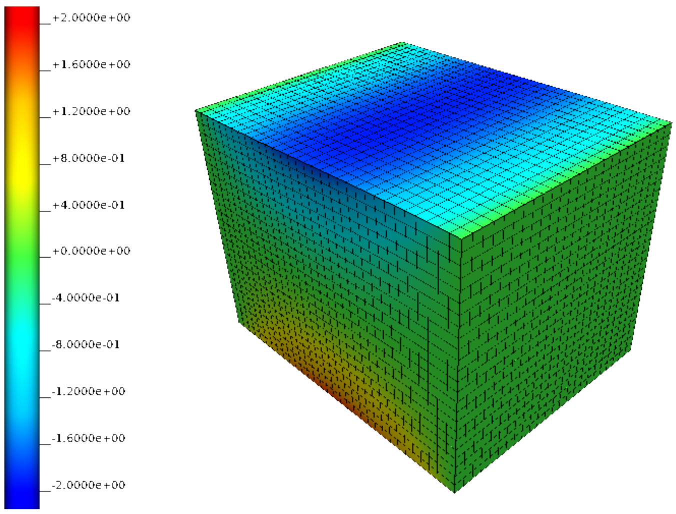

Figure 42: Results, iron 3D fine mesh.

5.3.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_x along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accoding to

$$\mathtt{L_2\text{-}norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\mathtt{abaqus}} - u^i_{y,\mathtt{iron}}\right)^2}, \tag{78}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.

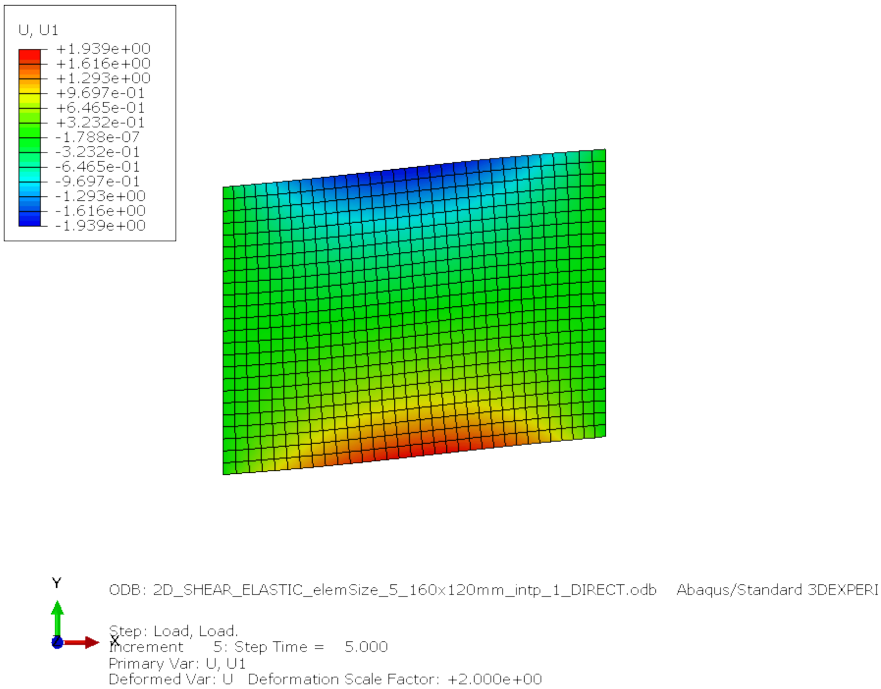

Figure 43: Results, Abaqus 2D fine mesh.

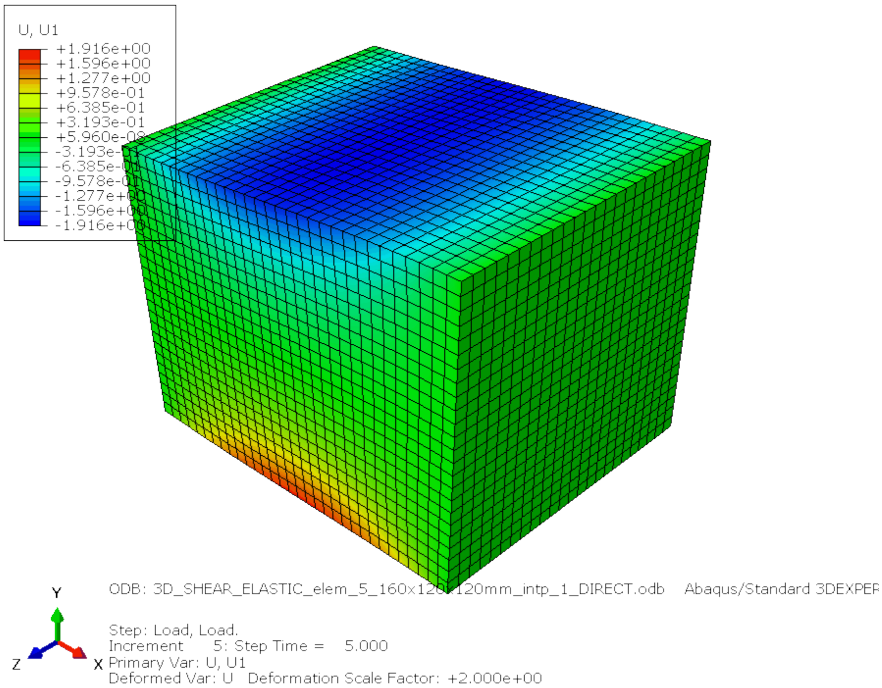

Figure 44: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-`norm`	Interpolation
2D	Coarse	6.696×10^{-3}	Linear
2D	Medium	1.273×10^{-3}	Linear
2D	Fine	2.489×10^{-4}	Linear
3D	Coarse	4.234×10^{-4}	Linear
3D	Medium	4.184×10^{-5}	Linear
3D	Fine	3.781×10^{-6}	Linear
2D	Coarse	3.036×10^{-4}	Quadratic
2D	Medium	6.099×10^{-5}	Quadratic
2D	Fine	1.089×10^{-5}	Quadratic
3D	Coarse	...	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 2: Quantiative error between Abaqus 2017 and iron simulations for linear elastic shear

5.4 Example-0111 [PLAUSIBLE]

5.4.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = f(\mathbf{u}, t) \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \tag{79}$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad x = y = 0, \tag{80}$$

$$f(u_x) = 6.0 \times 10^4 \qquad x = 160, \tag{81}$$

and in 3D by

$$u_x = u_y = u_z = 0 \qquad x = y = z = 0, \tag{82}$$

$$f(u_x) = 7.2 \times 10^6 \qquad x = 160. \tag{83}$$

The material parameters are

$$E = 10000\texttt{MPa}, \tag{84}$$

$$\nu = 0.3, \tag{85}$$

$$\rho = 5 \times 10^{-9}\texttt{tonne.mm}^3. \tag{86}$$

5.4.2 *Computational model*

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
  - float: elastic modulus
  - float: Poisson ratio
  - float: XXX
- Command line arguments for tests are:
  - 160 120 0 8 6 0 1 0 10000 0.3 XXX
  - 160 120 0 16 12 0 1 0 10000 0.3 XXX
  - 160 120 0 32 24 0 1 0 10000 0.3 XXX
  - 160 120 120 8 6 6 1 0 10000 0.3 XXX
  - 160 120 120 16 12 12 1 0 10000 0.3 XXX
  - 160 120 120 32 24 24 1 0 10000 0.3 XXX
  - 160 120 0 8 6 0 2 0 10000 0.3 XXX
  - 160 120 0 16 12 0 2 0 10000 0.3 XXX

160 120 0 32 24 0 2 0 10000 0.3 XXX
160 120 120 8 6 6 2 0 10000 0.3 XXX
160 120 120 16 12 12 2 0 10000 0.3 XXX
160 120 120 32 24 24 2 0 10000 0.3 XXX

5.4.3 *Results*

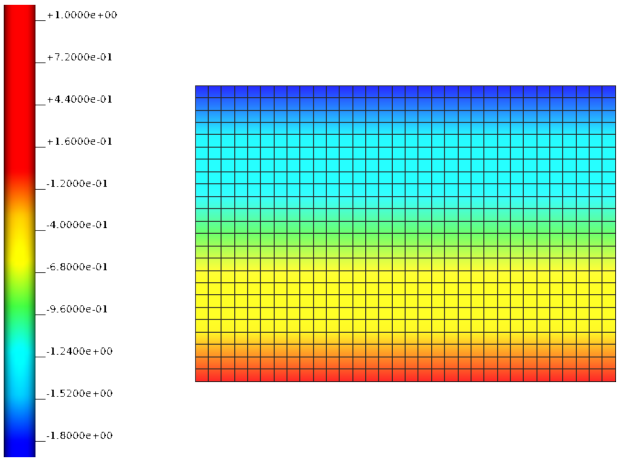

Figure 45: Results, iron 2D fine mesh.

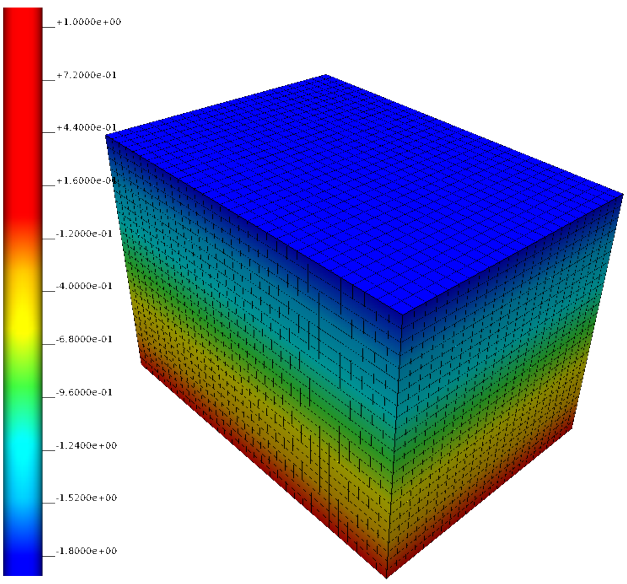

Figure 46: Results, iron 3D fine mesh.

5.4.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_y along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accoding to

$$\mathtt{L_2\text{-}norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\mathtt{abaqus}} - u^i_{y,\mathtt{iron}}\right)^2}, \tag{87}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.

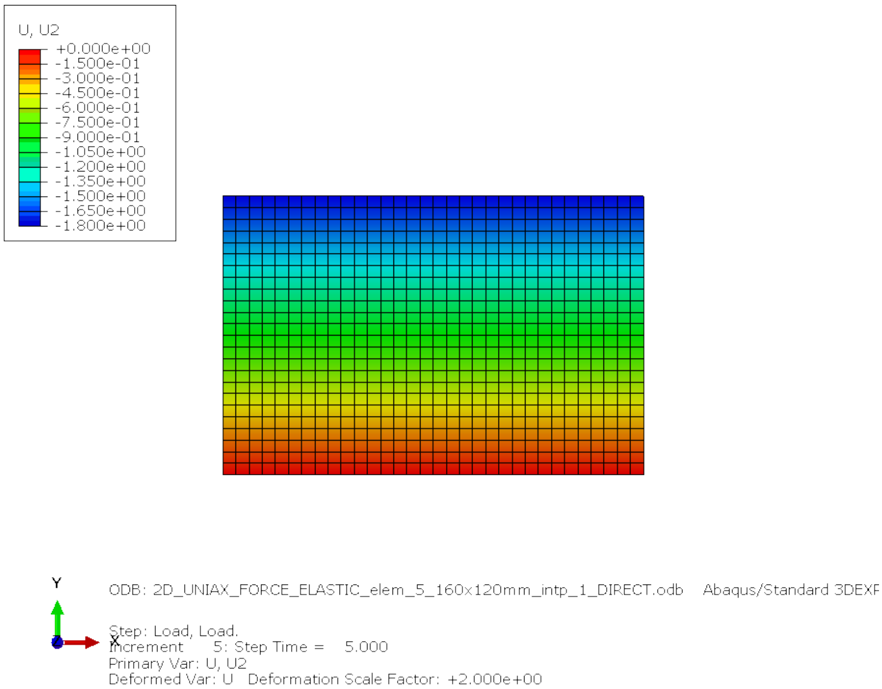

Figure 47: Results, Abaqus 2D fine mesh.

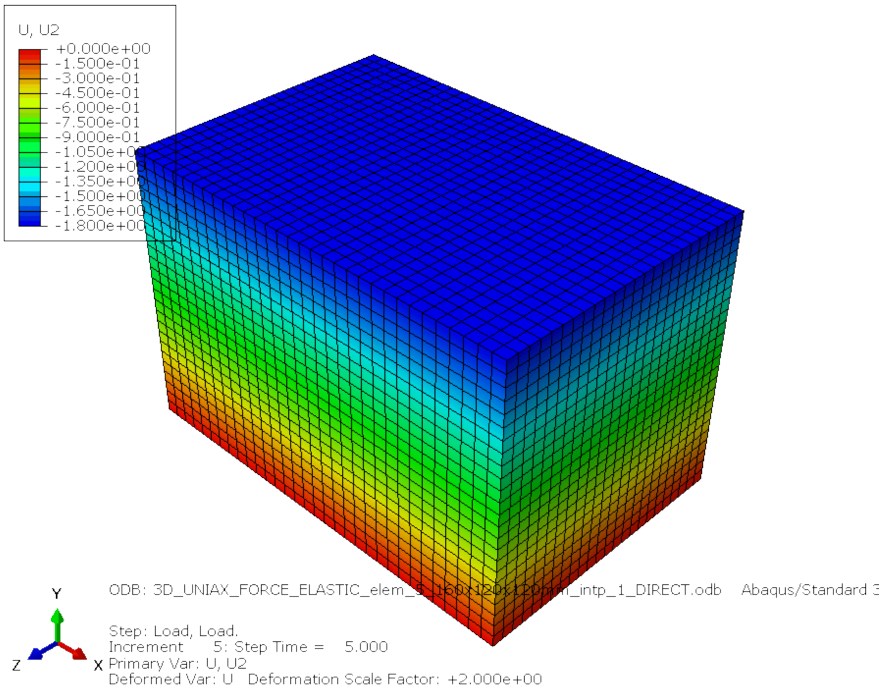

Figure 48: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	...	Linear
2D	Medium	...	Linear
2D	Fine	...	Linear
3D	Coarse	...	Linear
3D	Medium	...	Linear
3D	Fine	...	Linear
2D	Coarse	...	Quadratic
2D	Medium	...	Quadratic
2D	Fine	...	Quadratic
3D	Coarse	...	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 3: Quantiative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extenions

5.5 Example-0112 `[PLAUSIBLE]`

5.5.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \tag{88}$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad y = 0, \tag{89}$$

$$\mathbf{f}(u_y) = 6.0 \times 10^4 \qquad x = 160, \tag{90}$$

and in 3D by

$$u_x = u_z = 0 \qquad x = 0, \tag{91}$$

$$u_y = 0 \qquad y = 0, \tag{92}$$

$$u_x = 160 \qquad x = 160, \tag{93}$$

$$\mathbf{f}(u_y) = 7.2 \times 10^6 \qquad x = 160. \tag{94}$$

The material parameters are

$$E = 10000\texttt{MPa}, \tag{95}$$

$$\nu = 0.3, \tag{96}$$

$$\rho = 5 \times 10^{-9}\texttt{tonne.mm}^3. \tag{97}$$

5.5.2 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
  float: elastic modulus
  float: Poisson ratio
  float: XXX
- Command line arguments for tests are:
  160 120 0 8 6 0 1 0 10000 0.3 XXX
  160 120 0 16 12 0 1 0 10000 0.3 XXX
  160 120 0 32 24 0 1 0 10000 0.3 XXX
  160 120 120 8 6 6 1 0 10000 0.3 XXX
  160 120 120 16 12 12 1 0 10000 0.3 XXX
  160 120 120 32 24 24 1 0 10000 0.3 XXX

160 120 0 8 6 0 2 0 10000 0.3 XXX

160 120 0 16 12 0 2 0 10000 0.3 XXX

160 120 0 32 24 0 2 0 10000 0.3 XXX

160 120 120 8 6 6 2 0 10000 0.3 XXX

160 120 120 16 12 12 2 0 10000 0.3 XXX

160 120 120 32 24 24 2 0 10000 0.3 XXX

5.5.3 *Results*

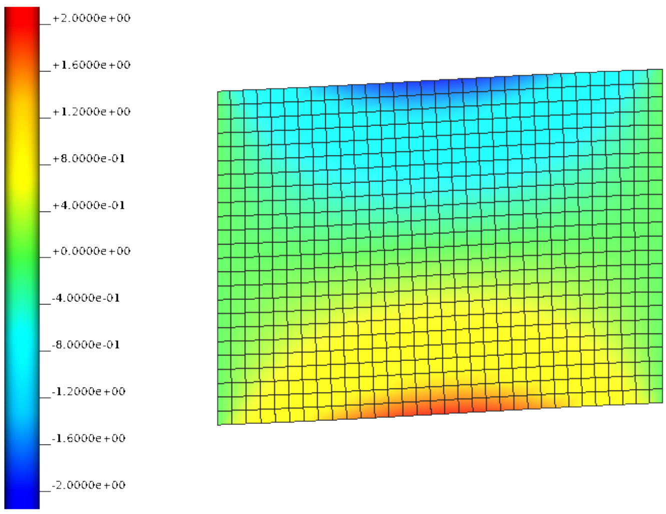

Figure 49: Results, iron 2D fine mesh.

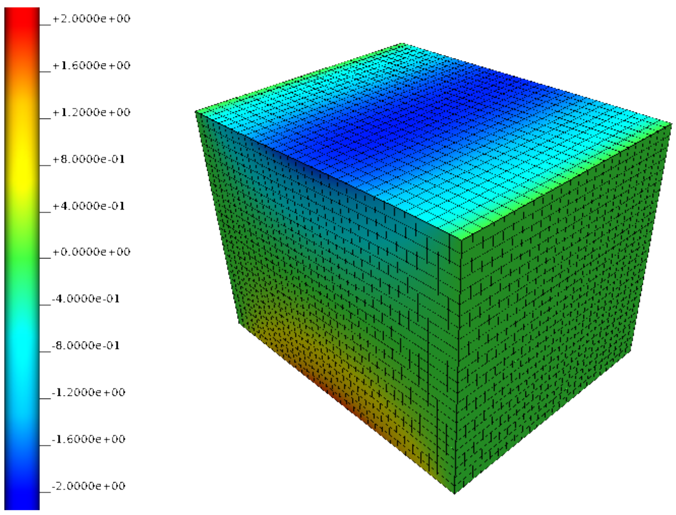

Figure 50: Results, iron 3D fine mesh.

5.5.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_x along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accoding to

$$L_2\text{-}\mathtt{norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\mathtt{abaqus}} - u^i_{y,\mathtt{iron}}\right)^2}, \tag{98}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.

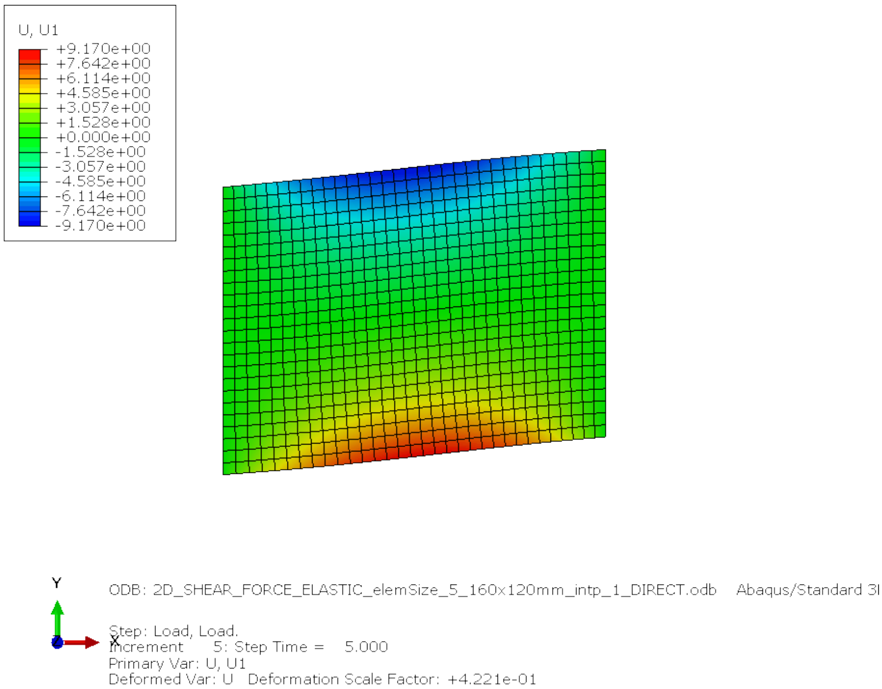

Figure 51: Results, Abaqus 2D fine mesh.

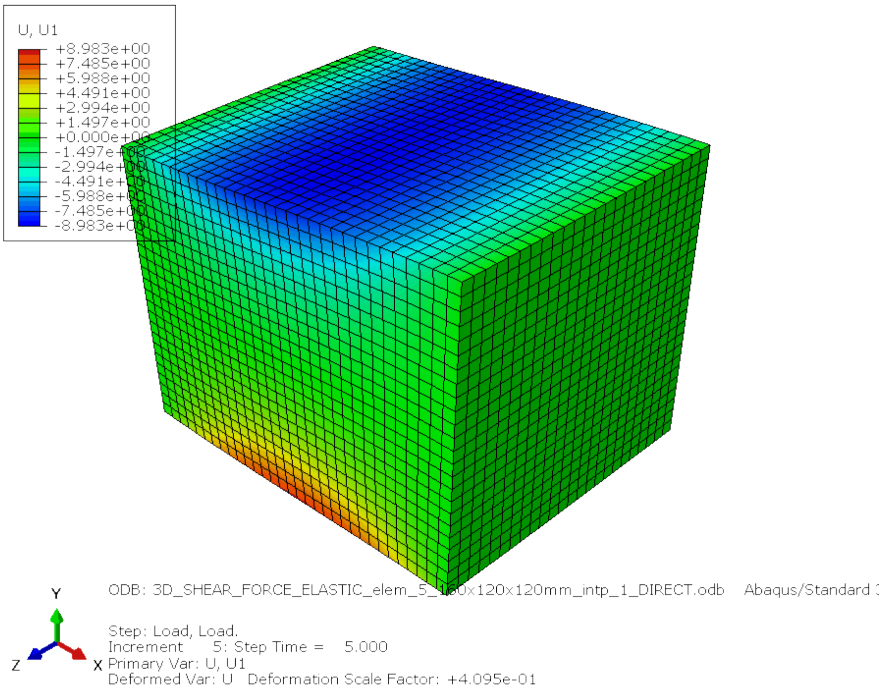

Figure 52: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	...	Linear
2D	Medium	...	Linear
2D	Fine	...	Linear
3D	Coarse	...	Linear
3D	Medium	...	Linear
3D	Fine	...	Linear
2D	Coarse	...	Quadratic
2D	Medium	...	Quadratic
2D	Fine	...	Quadratic
3D	Coarse	...	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 4: Quantiative error between Abaqus 2017 and iron simulations for linear elastic shear

6 FINITE ELASTICITY

7 NAVIER-STOKES FLOW

7.1 Equation in general form

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathbf{I}) = \rho \mathbf{f} \tag{99}$$

7.2 Example-0302-u [COMPILES]

Example uses user-defined simplex meshes in CHeart mesh format with quadratic/linear interpolation for velocity/pressure and solves a dynamic problem.

Setup is the well-known lid-driven cavity problem on the unit square or unit cube in two and three dimensions.

Current issue: does not converge after 30 some time iterations (2D and 3D).

Visualization issue: In exelem-file, replace

```
 1.   constant(2)*constant, no modify, grid based.
   #xi1=0, #xi2=0
 2.   constant(2)*constant, no modify, grid based.
```

with

```
 1.   constant*constant, no modify, grid based.
   #xi1=0, #xi2=0
 2.   constant*constant, no modify, grid based.
```

and likewise for 3D, replace

```
 1.   constant(2;3)*constant*constant, no modify, grid based.
   #xi1=0, #xi2=0, #xi3=0
 2.   constant(2;3)*constant*constant, no modify, grid based.
```

with

```
 1.   constant*constant*constant, no modify, grid based.
   #xi1=0, #xi2=0, #xi3=0
 2.   constant*constant*constant, no modify, grid based.
```

7.2.1 *Mathematical model - 2D*

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho\boldsymbol{v}) + \nabla\cdot(\rho\boldsymbol{v}\otimes\boldsymbol{v}) - \nabla\cdot(\mu\nabla\boldsymbol{v} - p\mathbf{I}) = \rho\mathbf{f} \qquad \Omega = [0,1]\times[0,1], \tag{100}$$

$$\nabla\cdot\boldsymbol{v} = 0, \tag{101}$$

with boundary conditions

$$\boldsymbol{v} = 0 \qquad x = 0, \tag{102}$$

$$\boldsymbol{v} = 0 \qquad x = 1, \tag{103}$$

$$\boldsymbol{v} = 0 \qquad y = 0, \tag{104}$$

$$\boldsymbol{v} = [1,0]^T \qquad y = 1. \tag{105}$$

Viscosity $\mu = 0.0025$, density $\rho = 1$. Thus, Reynolds number $Re = 400$.

7.2.2 *Mathematical model - 3D*

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v}) - \nabla \cdot (\mu \nabla \boldsymbol{v} - p\mathbf{I}) = \rho \mathbf{f} \quad \Omega = [0,1] \times [0,1] \times [0,1], \tag{106}$$

$$\nabla \cdot \boldsymbol{v} = 0, \tag{107}$$

with boundary conditions

$$\boldsymbol{v} = 0 \qquad x = 0, \tag{108}$$

$$\boldsymbol{v} = 0 \qquad x = 1, \tag{109}$$

$$\boldsymbol{v} = 0 \qquad y = 0, \tag{110}$$

$$\boldsymbol{v} = [1,0]^T \qquad y = 1, \tag{111}$$

$$\boldsymbol{v} = 0 \qquad z = 0, \tag{112}$$

$$\boldsymbol{v} = 0 \qquad z = 1. \tag{113}$$

Viscosity $\mu = 0.01$, density $\rho = 1$. Thus, Reynolds number $Re = 100$.

7.2.3 *Computational model*

- Commandline arguments are:

  integer: number of dimensions (2: 2D, 3: 3D
  integer: mesh refinement level (1, 2, 3, ...)
  float: start time
  float: stop time
  float: time step size
  float: density
  float: viscosity
  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2 1 0.0 1.0 0.001 0.0025 1.0 0
  2 2 0.0 1.0 0.001 0.0025 1.0 0
  2 3 0.0 1.0 0.001 0.0025 1.0 0
  2 1 0.0 1.0 0.001 0.0025 1.0 1
  2 2 0.0 1.0 0.001 0.0025 1.0 1
  2 3 0.0 1.0 0.001 0.0025 1.0 1
  3 1 0.0 1.0 0.001 0.01 1.0 0
  3 2 0.0 1.0 0.001 0.01 1.0 0
  3 3 0.0 1.0 0.001 0.01 1.0 0
  3 1 0.0 1.0 0.001 0.01 1.0 1
  3 2 0.0 1.0 0.001 0.01 1.0 1
  3 3 0.0 1.0 0.001 0.01 1.0 1

- Note: Binary uses command line arguments to search for the relevant mesh files.

7.2.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

8 MONODOMAIN

8.1 Example-0401 [PLAUSIBLE]

8.1.1 *Mathematical model*

We solve the Monodomain Equation

$$\sigma\Delta V_m(t) = A_m\Big(C_m\frac{\partial V_m}{\partial t} + I_{ionic}(V_m)\Big) \quad \Omega = [0,1]\times[0,1], \quad t\in[0,3.0] \tag{114}$$

where $V_m(t)$ is given by the Hodgkin-Huxley system of ODEs [2]
with boundary conditions

$$V_m = 0 \qquad x = y = 0, \tag{115}$$

$$V_m = 0 \qquad x = y = 1. \tag{116}$$

and initial values

$$V_m(t=0) = -75$$

Additionally a stimulation current I_{stim} is applied for $t_{stim} = [0, 0.1]$ at the center node of the domain (i.e. at $(x,y) = (\frac{1}{2}, \frac{1}{2},))$.
Material parameters:

$$\sigma = 3.828$$

$$A_m = 500$$

$C_m = 0.58$ for the slow-twitch case, $C_m = 1.0$ for the fast-twitch case

$I_{Stim} = 1200$ for the slow-twitch case, $I_{Stim} = 2000.0$ for the fast-twitch case

8.1.2 *Computational model*

- This example uses generated meshes
- Commandline arguments are:
  - number elements X
  - number elements Y
  - interpolation order (1: linear; 2: quadratic)
  - solver type (0: direct; 1: iterative)
  - PDE step size
  - stop time
  - output frequency
  - CellML Model URL
  - slow-twitch
  - ODE time-step
- Commands for tests are:

```
./folder/src/example 24 24 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml F 0.0001
./folder/src/example 24 24 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml F 0.005
./folder/src/example 10 10 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml F 0.0001
mpirun -n 2 ./folder/src/example 24 24 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml
```

```
mpirun -n 8 ./folder/src/example 24 24 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml
./folder/src/example 2 2 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml F 0.0001
mpirun -n 2 ./folder/src/example 2 2 1 0 0.005 3.0 1 hodgkin_huxley_1952.cellml F
```

- This is a dynamic problem.

8.1.3 *Results*

```
Passed tests: 20 / 20

No failed tests.
```

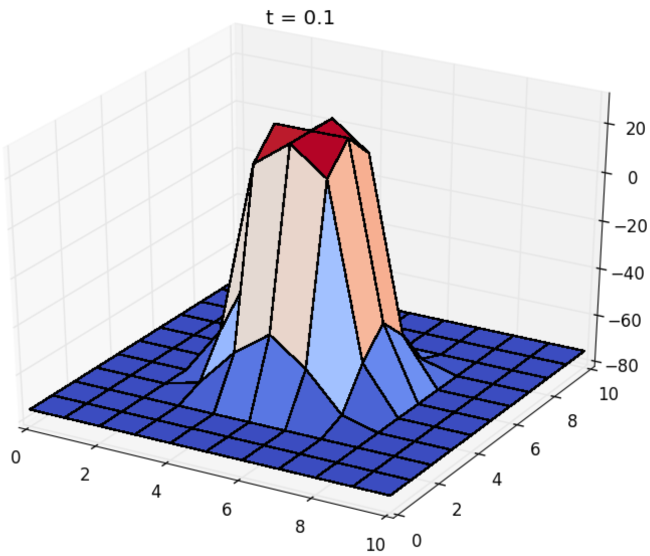

Figure 53: Result of scenario with 10×10 elements, $t = 20$, direct solver, $p = 1$ process

With the 'big' target there will be animations created. You get a better understanding of the solutions by looking at them in `iron-tests/examples/example-0401/doc/figures`.

8.1.4 *Validation*

We compare with a Matlab implementation as well as with reference iron files.

The matlab scripts use finite difference discretization instead of finite elements. The results are qualitatively the same but exactly. The compare script also tests for matlab reference data which is only included for the 2 examples with 24×24 elements. There is a big L_2-error. The tolerance is set to a high value to allow for the tests to succeed. With this the comparing-mechanism is tested. Maybe in the future someone succeeds to generate suitable matlab data that then can just be exchanged without having to rewrite the compare script.

The iron files to compare with are the output of the simulation as of Aug. 2017. In that way we can check if the simulation brakes with respect to the current state. In order to keep file sizes minimal the comparision is only conducted for time steps $t = 0.01, 0.1, 0.2, 1, 2, 3$ for the 'big' target and $t = 0.1, 0.2, 1$ for the 'fast' target.

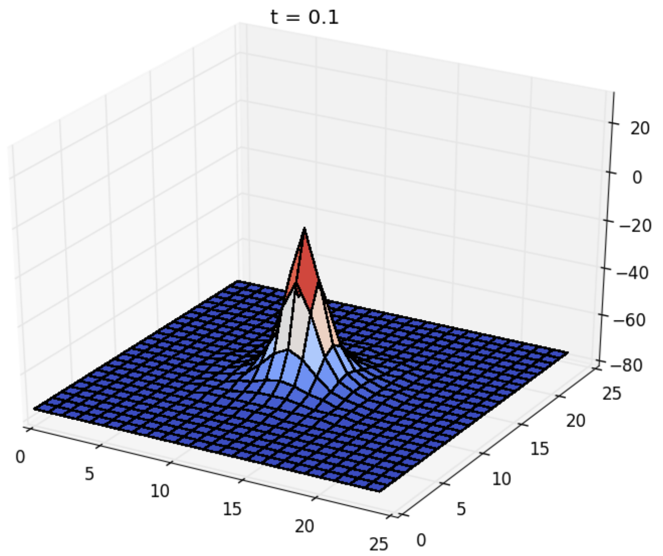

Figure 54: Result of scenario with 24×24 elements, $t = 20$, direct solver, $p = 1$ process

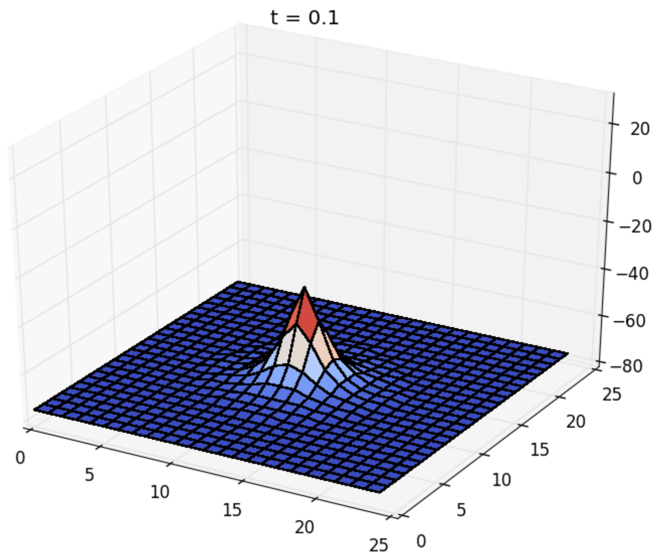

Figure 55: Result of scenario with 24×24 elements, $t = 20$, iterative solver, $p = 1$ process

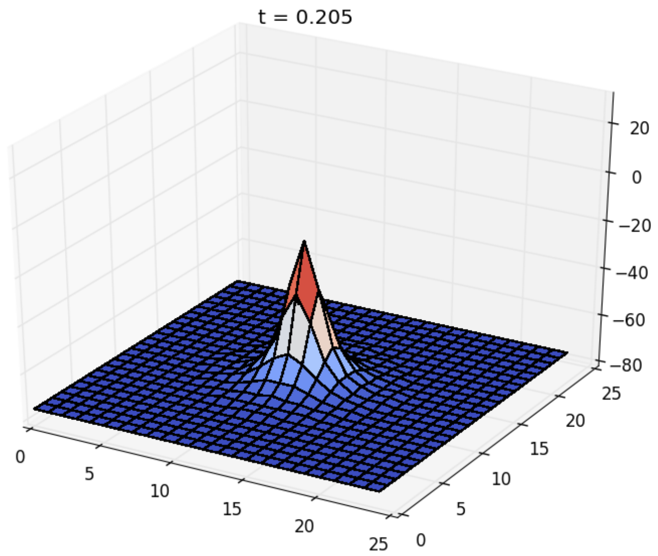

Figure 56: Result of scenario with 24×24 elements, $t = 20$, iterative solver, $p = 2$ processes

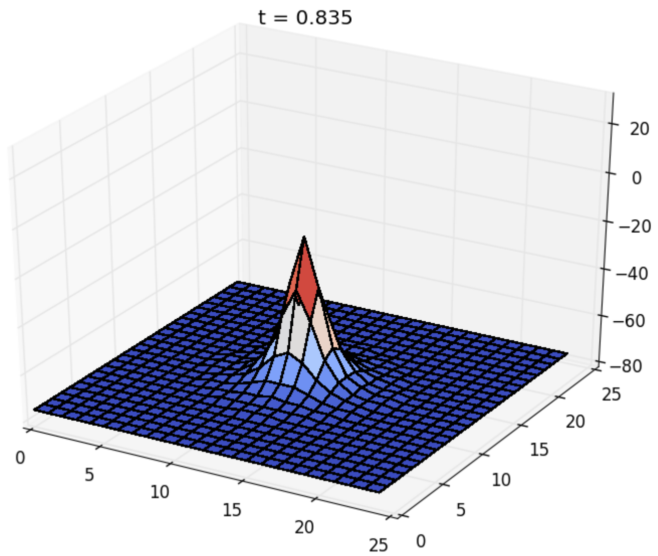

Figure 57: Result of scenario with 24×24 elements, $t = 20$, iterative solver, $p = 8$ processes

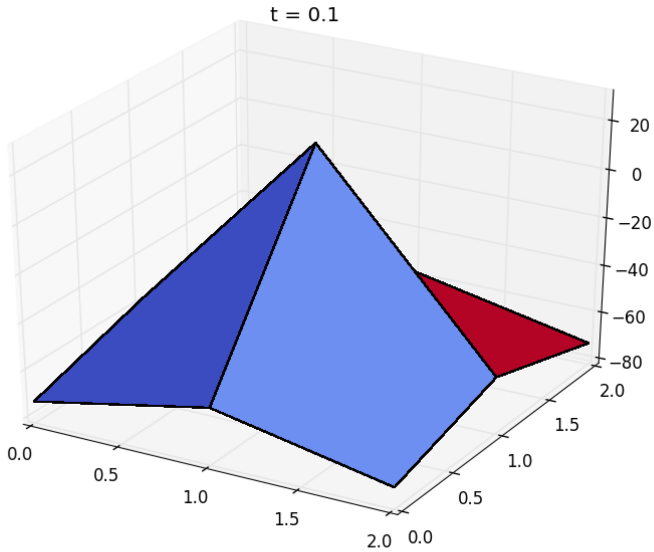

Figure 58: Result of scenario with 2×2 elements, $t = 20$, direct solver, $p = 1$ process

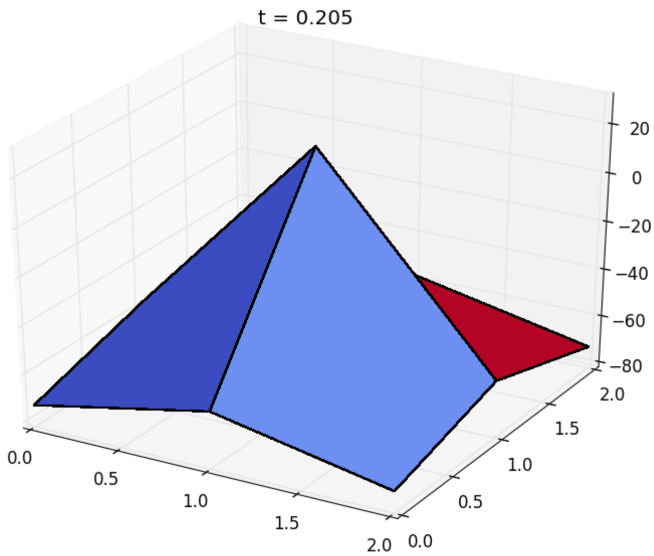

Figure 59: Result of scenario with 2×2 elements, $t = 20$, direct solver, $p = 2$ processes

8.2 Example-0402 `[PLAUSIBLE]`

8.2.1 *Mathematical model*

We solve the Monodomain Equation

$$\sigma \Delta V_m(t) = A_m \left(C_m \frac{\partial V_m}{\partial t} + I_{ionic}(V_m) \right) \quad \Omega = [0,1] \times [0,1], \quad t \in [0, 3.0] \tag{117}$$

where $V_m(t)$ is given by the CellML description of Noble's 1998 improved guinea-pig ventricular cell model system of ODEs [3]

with boundary conditions

$$V_m = 0 \qquad x = y = 0, \tag{118}$$

$$V_m = 0 \qquad x = y = 1. \tag{119}$$

and initial values

$$V_m(t = 0) = -75$$

Additionally a stimulation current I_{stim} is applied for $t_{stim} = [0, 0.1]$ at the center node of the domain (i.e. at $(x, y) = (\frac{1}{2}, \frac{1}{2},))$.

Material parameters:

$\sigma = 3.828$

$A_m = 500$

$C_m = 0.58$ for the slow-twitch case, $C_m = 1.0$ for the fast-twitch case

$I_{Stim} = 1200$ for the slow-twitch case, $I_{Stim} = 2000.0$ for the fast-twitch case

8.2.2 *Computational model*

- This example uses generated meshes
- Commandline arguments are:

  number elements X

  number elements Y

  interpolation order (1: linear; 2: quadratic)

  solver type (0: direct; 1: iterative)

  PDE step size

  stop time

  output frequency

  CellML Model URL

  slow-twitch

  ODE time-step
- Commands for tests are:

```
./folder/src/example 24 24 1 0 0.005 3.0 1 n98.xml F 0.0001
./folder/src/example 24 24 1 0 0.005 3.0 1 n98.xml F 0.005
./folder/src/example 10 10 1 0 0.005 3.0 1 n98.xml F 0.0001
```

```
mpirun -n 2 ./folder/src/example 24 24 1 0 0.005 3.0 1 n98.xml F 0.0001
mpirun -n 8 ./folder/src/example 24 24 1 0 0.005 3.0 1 n98.xml F 0.0001
./folder/src/example 2 2 1 0 0.005 3.0 1 n98.xml F 0.0001
mpirun -n 2 ./folder/src/example 2 2 1 0 0.005 3.0 1 n98.xml F 0.0001
```

- This is a dynamic problem.

8.2.3 *Results*

```
Passed tests: 12 / 12

No failed tests.
```

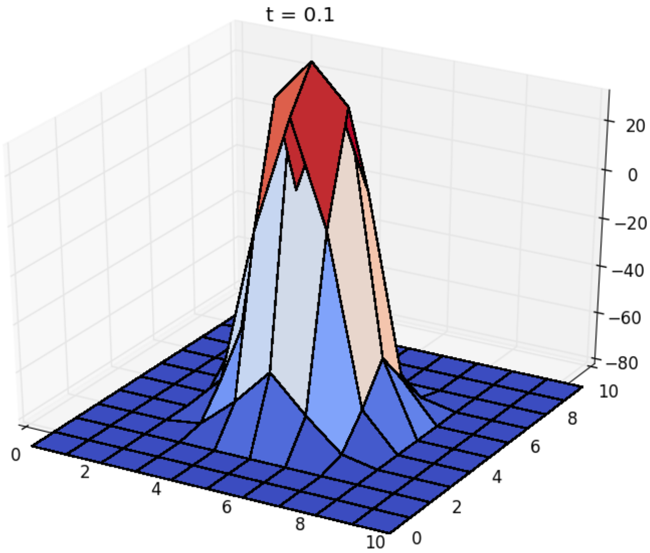

Figure 60: Result of scenario with 10×10 elements, $t = 20$, direct solver, $p = 1$ process

With the 'big' target there will be animations created. You get a better understanding of the solutions by looking at them in `iron-tests/examples/example-0402/doc/figures`.

8.2.4 *Validation*

We compare with reference iron files of Aug. 2017. See also the notes on `example-0401`.

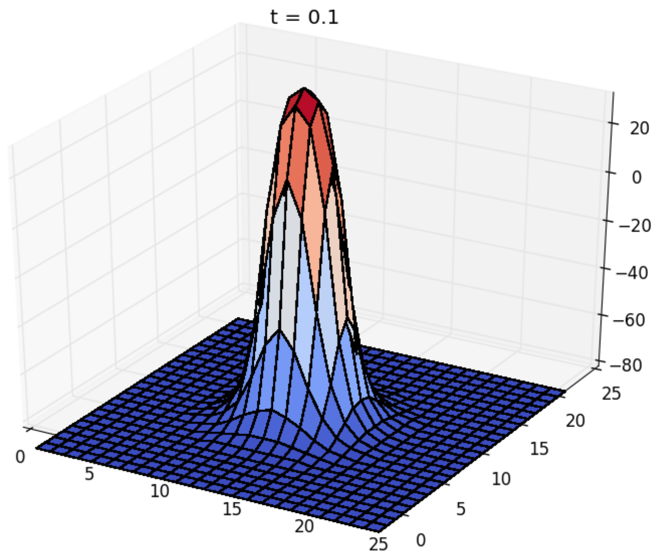

Figure 61: Result of scenario with 24×24 elements, $t = 20$, direct solver, $p = 1$ process

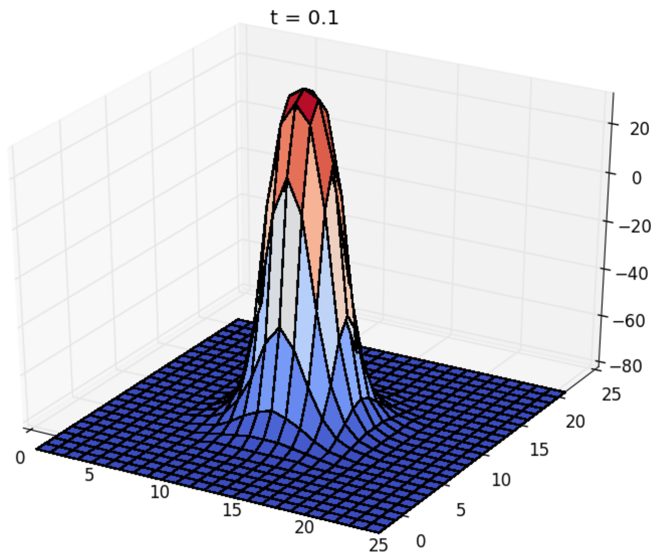

Figure 62: Result of scenario with 24×24 elements, $t = 20$, iterative solver, $p = 1$ process

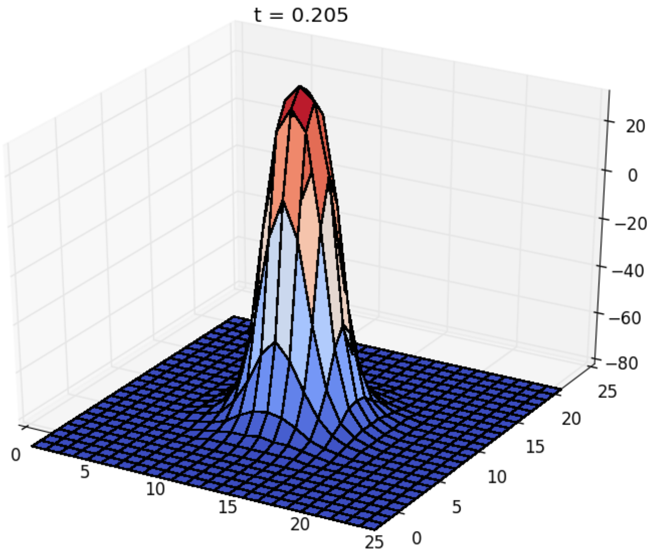

Figure 63: Result of scenario with 24×24 elements, $t = 20$, iterative solver, $p = 2$ processes

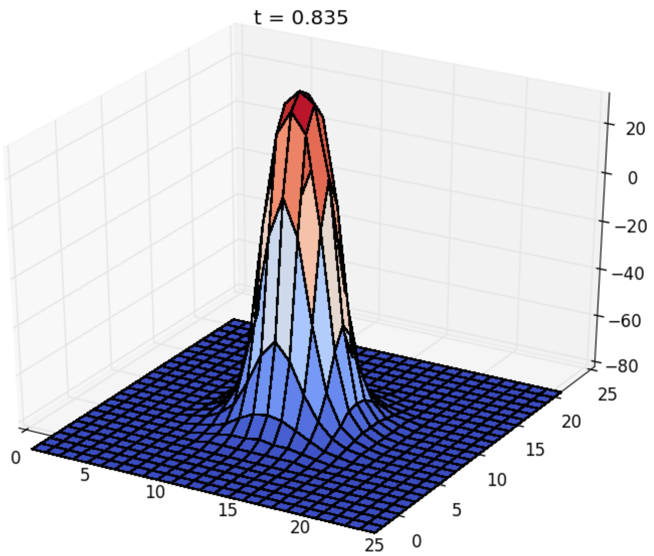

Figure 64: Result of scenario with 24×24 elements, $t = 20$, iterative solver, $p = 8$ processes

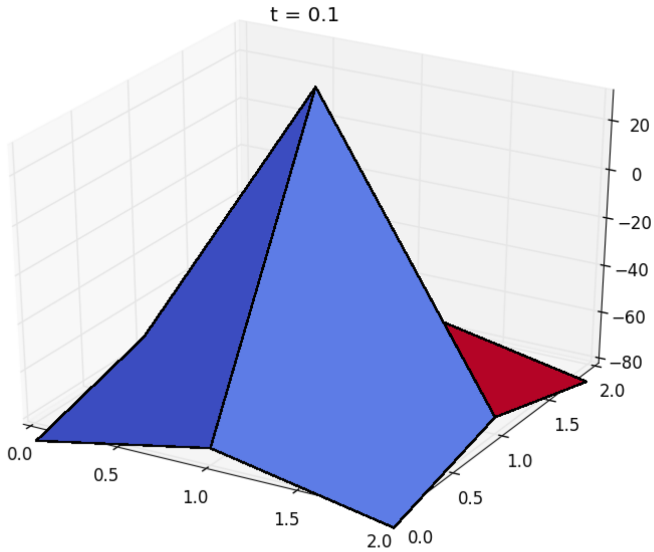

Figure 65: Result of scenario with 2×2 elements, $t = 20$, direct solver, $p = 1$ process

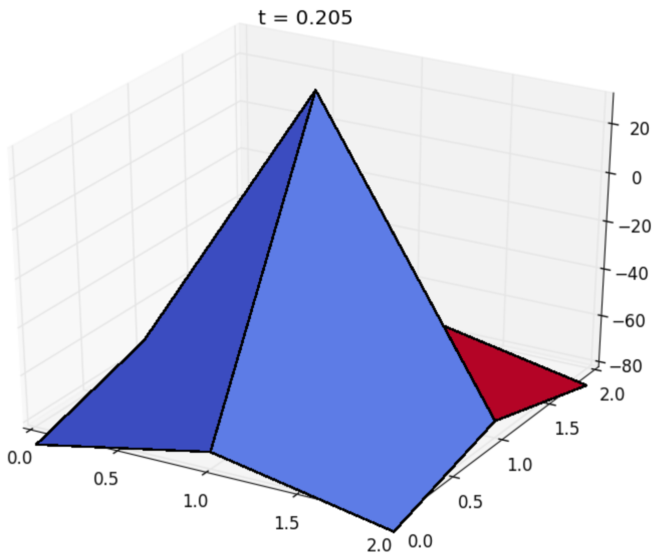

Figure 66: Result of scenario with 2×2 elements, $t = 20$, direct solver, $p = 2$ processes

8.3 Example-0404-c `[PLAUSIBLE]`

8.3.1 *Mathematical model*

We solve the Monodomain Equation

$$\sigma\Delta V_m(t) = A_m\Big(C_m\frac{\partial V_m}{\partial t} + I_{ionic}(V_m)\Big) \quad \Omega = [0,1], \quad t \in [0,10.0] \qquad (120)$$

where $V_m(t)$ is given by the Hodgkin-Huxley system of ODEs [2]
with Neumann boundary conditions

$$\frac{\partial u}{\partial n} = 0 \qquad x = 0, \qquad (121)$$

$$\frac{\partial u}{\partial n} = 0 \qquad x = 1. \qquad (122)$$

and initial values

$$V_m(t=0) = -75$$

Additionally a stimulation current I_{stim} is applied for $t_{stim} = [0, 0.5]$ at the center node of the domain (i.e. at $(x,y) = (\frac{1}{2}, \frac{1}{2},))$.
Material parameters:

$$\sigma = 3.828$$

$$A_m = 500$$

$C_m = 0.58$ for the slow-twitch case, $C_m = 1.0$ for the fast-twitch case

$$I_{Stim} = \begin{cases} 75/10 \cdot (2X) & \text{for } X \geqslant 10 \text{ reference elements} \\ 75 & \text{for } < \ 10 \text{ reference elements} \end{cases} \quad \text{for the slow-twitch case,}$$

$$I_{Stim} = \begin{cases} 75/12 \cdot (2X) & \text{for } X \geqslant 12 \text{ reference elements} \\ 75 & \text{for } < \ 12 \text{ reference elements} \end{cases} \quad \text{for the fast-twitch case}$$

8.3.2 *Computational model*

- This example uses generated meshes
- Commandline arguments are:

  number of elements

  order of interpolation

  solver type (0: direct; 1: iterative)

  time step PDE

  end time

  output file stride

  cellml model file

  if slow-twitch (T: slow-twitch, F: fast-twitch)

  time step ODE
- Commandline arguments for tests are:

```
64 2 0 0.01 10 5 hodgkin_huxley_1952.cellml F 0.01
64 2 0 0.005 10 10 hodgkin_huxley_1952.cellml F 0.005
```

```
64 2 0 0.001 10 50 hodgkin_huxley_1952.cellml F 0.001
64 2 0 0.0005 10 100 hodgkin_huxley_1952.cellml F 0.0005
64 2 0 0.00025 10 200 hodgkin_huxley_1952.cellml F 0.00025
```

- This is a dynamic problem.
- More test cases for 2nd order should be constructed.

8.3.3 *Results*

We run the scenario for different time step widths and examine the experimental order of convergence.

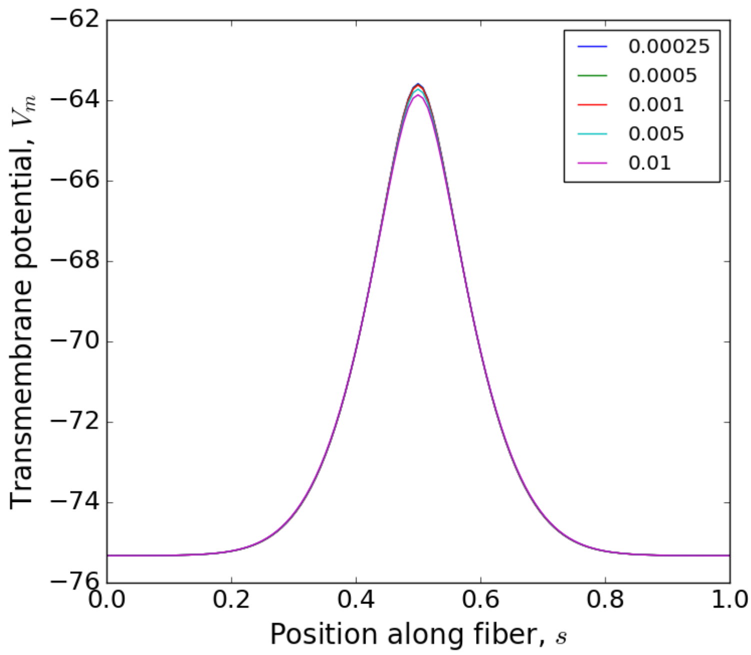

Figure 67: V_m for time $t = 1.0$, different time step widths $dt \in \{0.01, 0.005, 0.001, 0.0005, 0.00025\}$

8.3.4 *Validation*

The purpose of this test case is to see if convergence orders are as expected, no actual validation of the output takes place.

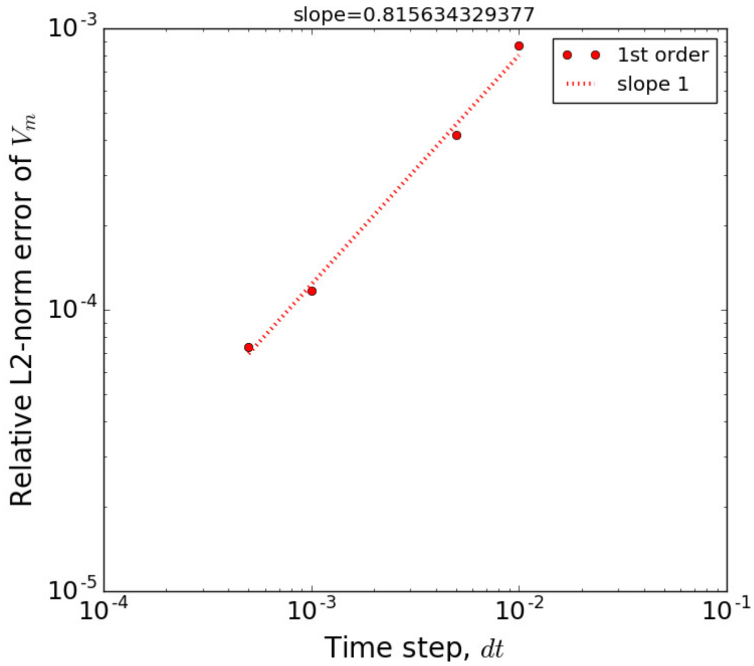

Figure 68: Error at $t = 1.0$ for different time steps widths. The slope (=experimental order of convergence) should be around 1.

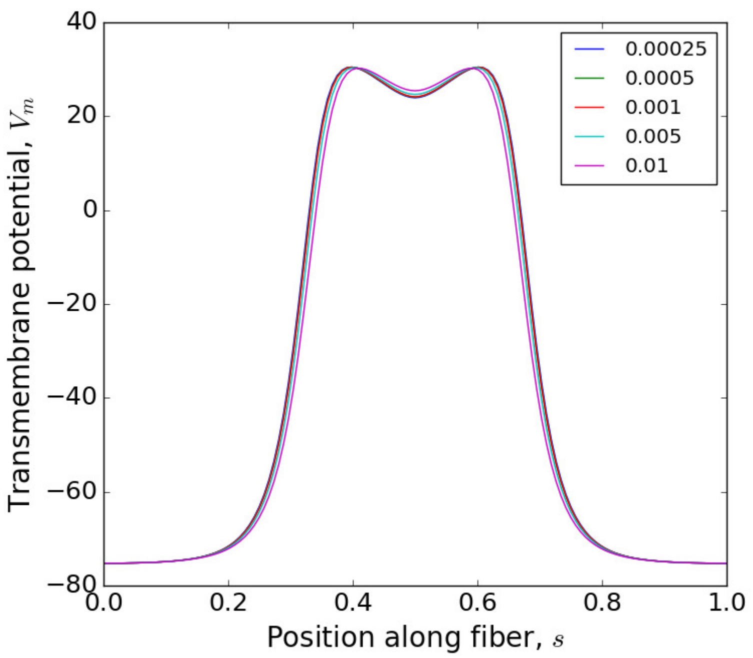

Figure 69: V_m for time $t = 3.0$, different time step widths $ddt \in \{0.01, 0.005, 0.001, 0.0005, 0.00025\}$

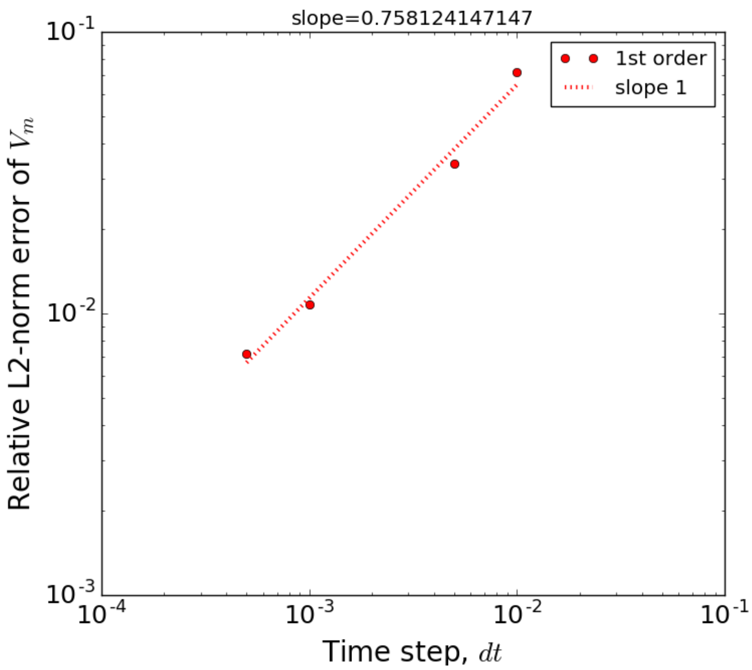

Figure 70: Error at $t = 3.0$ for different time steps widths. The slope (=experimental order of convergence) should be around 1.

9 CELLML MODEL

REFERENCES

[1] Chris Bradley, Andy Bowery, Randall Britten, Vincent Budelmann, Oscar Camara, Richard Christie, Andrew Cookson, Alejandro F Frangi, Thiranja Babarenda Gamage, Thomas Heidlauf, et al. Opencmiss: a multi-physics & multi-scale computational infrastructure for the vph/-physiome project. *Progress in biophysics and molecular biology*, 107(1):32–47, 2011.

[2] Alan L Hodgkin and Andrew F Huxley. Propagation of electrical signals along giant nerve fibres. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 177–183, 1952.

[3] Denis Noble, Anthony Varghese, Peter Kohl, and Penelope Noble. Improved guinea-pig ventricular cell model incorporating a diadic space, ikr and iks, and length-and tension-dependent processes. *Canadian Journal of Cardiology*, 14(1):123–134, 1998.