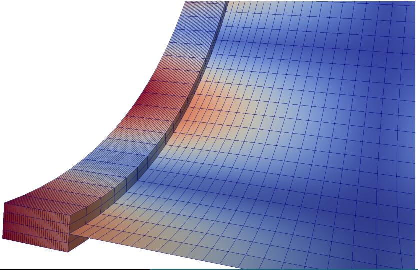
# Solvers for solid mechanics - Recent progress

Mika Malinen, D.Sc. (Tech.)

Aalto University
Department of Mathematics and Systems Analysis
and
CSC – IT Center for Science (on leave)

April 15, 2021



# About contents

The most notable “recent” developments (introduced in Elmer release 8.4., Dec 2018, or later)

- User-defined materials (UMAT) interface for nonlinear elasticity solver, together with documentation
- A nonlinear version of shell solver
- Support for solving strongly coupled FSI problems in frequency domain

Ongoing work

- A tight coupling of 3D elasticity and 2D shell equations

red = the special subjects of this presentation

# I Overview of solvers for solid mechanics

Volumetric discretizations of 2-D/3-D solids

- Linear elasticity (the module `StressSolve`)
    - Basic material laws, with possible anisotropy
    - Modal and stability analysis
    - Harmonic analysis (complex-valued fields)
    - Mesh adaptivity
- Nonlinear elasticity (`ElasticSolve`)
    - Finite deformations
    - Neo-Hookean and St Venant-Kirchhoff materials in-built
    - Note: A St Venant-Kirchhoff material intended for large displacements and small extensional strains
    - A special formulation for an incompressible material
    - Anisotropy for a St Venant-Kirchhoff material
    - User-defined materials (UMAT) interface to handle more general classes of solids (beyond elasticity)

# I Overview of solvers for solid mechanics

Models obtained via dimensional reduction

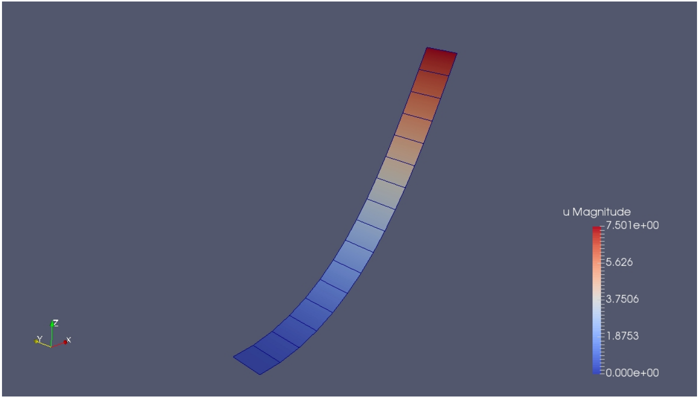
- 1-D beams (`BeamSolver3D`)
    - Shear-deformable (Timoshenko's theory) and allows torsional stiffness
    - A beam can be embedded freely in the 3-dimensional space
    - A linearly elastic material
    - A recent addition (May, 2019)
- 2-D Reissner-Mindlin model for linearly elastic plates (`SMITC`)
- 2-D shell equations (`ShellSolver`)

# I Overview of solvers for solid mechanics

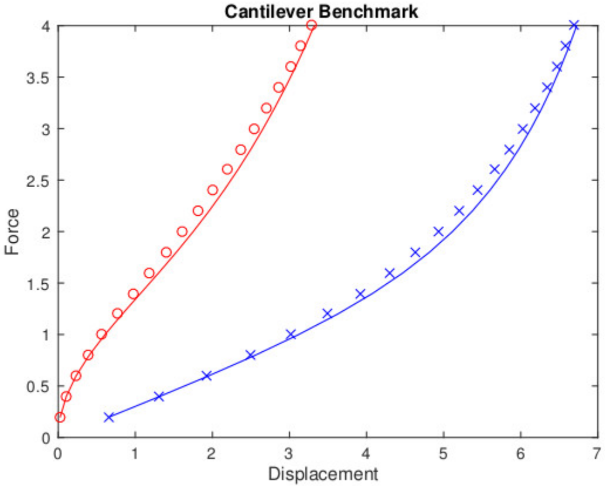
- 2-D shell equations (`ShellSolver`)
  - finite deformations (a linear model as a special case)
  - a St Venant-Kirchhoff material only
  - an extensible director assumed
  - in some aspects a research version (non-standard developments)
  - to replace the (undocumented) facet shell solver (`FacetShellSolver`)

# Nonlinear shell analysis: A cantilever benchmark

- A cantilever is subject to a shear force at an end

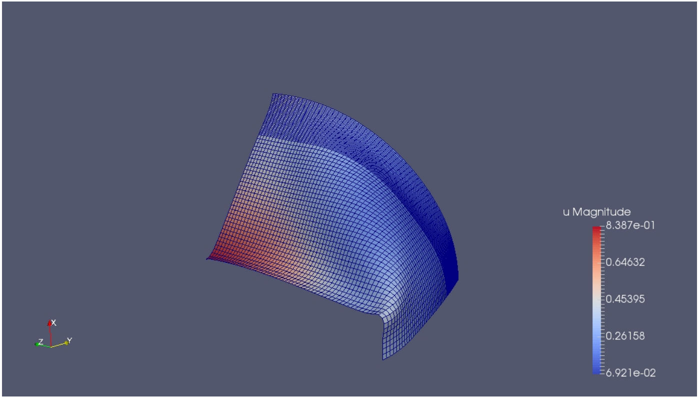


# Nonlinear shell analysis: A cantilever benchmark

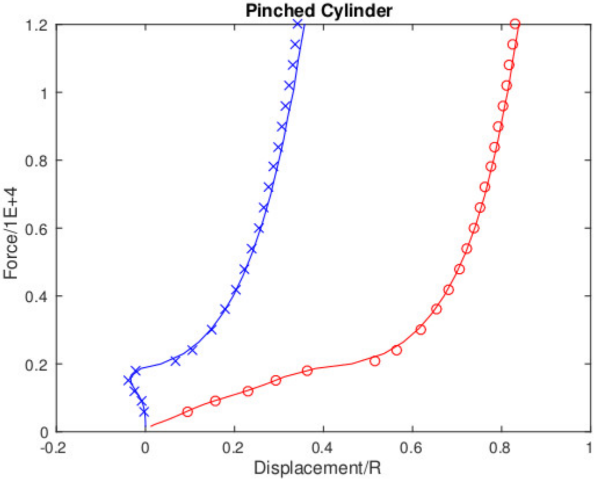


# Nonlinear shell analysis: A pinched cylinder benchmark

- A straight cylindrical shell is subject to a pinching force and has rigid end diaphragms allowing axial slip

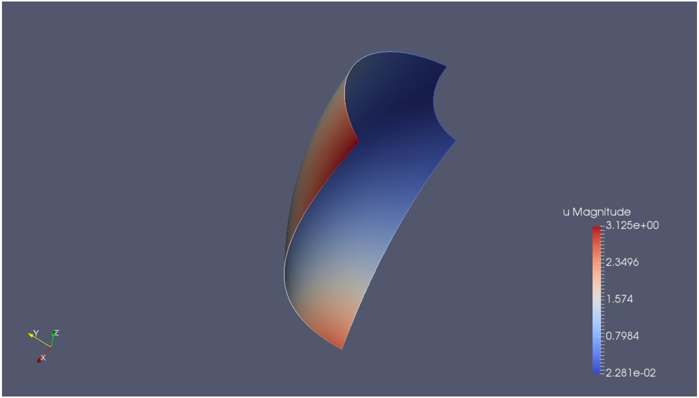


# Nonlinear shell analysis: A pinched cylinder benchmark

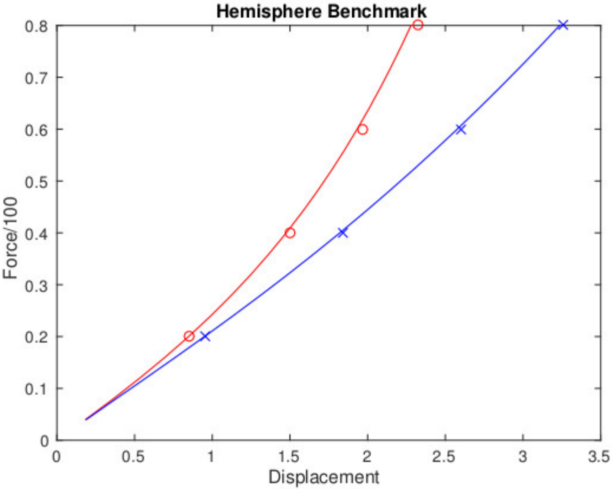


# Nonlinear shell analysis: An open hemisphere benchmark

- An open hemispherical shell is subject to inward and outward pinching forces



# Nonlinear shell analysis: An open hemisphere benchmark



# I Overview of solvers for solid mechanics

Additional utilities

- Pointwise springs and masses (`SpringAssembly`)
  - an additional assembly procedure to add springs and masses
  - allows the assembly although the mesh files do not specify point elements
  - a recent addition (Apr, 2020)

To sum up:

- Basic models available, limitations on available material models and postprocessing
- Higher-order discretizations may not be an option
- Pure solid mechanics has not really been on focus

# II User-defined material models

A typical question

*"I need to apply a special nonlinear material model. Does Elmer support such a simulation?"*

A typical answer

*"In principle yes, but you need to program the material law ..."*

A local enhancement suffices:

- Only a special subroutine has to be written
- The code of the solver need not be touched

# II User-defined material models

Some historical comments:

- Our UMAT development was initiated in a project
- The project goal was to enable interfacing with material models written for ABAQUS
- The interface was published later under open source
- A thesis (M.Sc.) also utilized the UMAT interface: `http://URN.fi/URN:NBN:fi:tty-201810032372`

# II User-defined material models

In practice

- UMAT is a Fortran subroutine with a fixed calling convention
- ABAQUS gives its own documentation

Different software work differently:

- Some adaptation on the Elmer side was needed
- For example, Elmer expresses the equilibrium equations in terms of the first Piola-Kirchhoff stress, while UMAT describes the material response in terms of the Cauchy stress
- Use modern Fortran when working with Elmer

# II User-defined material models

Limitations:

- Not all arguments of the UMAT subroutine are supported
- The implementation shouldn't rely on utility subroutines that are available only within Abaqus
- At the moment just stationary cases, but no technical hindrance to enable transient cases
- An adaptive load incrementation is not supported within Elmer
- That is, some simulation controls doesn't have a meaning within Elmer

# II User-defined material models

How to start

- UMAT subroutine can be compiled independently of the solver of Elmer
- `elmerf90` command coming with the installed Elmer can be used for compilation
- A ready template for writing UMAT is a part of the Elmer source code:

  `../fem/src/modules/UMATLib.F90`
- It also contains some examples of basic material models
- See also example cases given as tests

  `.../fem/tests/UMAT_*`

# II User-defined material models

- The file containing UMAT implementation can be named freely, so one may compile for example

```
elmerf90 MyUMATLib.F90 -o MyUMATLib
```

- Several material models can also be contained in a single file
- Use the keyword `UMAT Subroutine` in a material section to specify the file and to pick the subroutine desired

```
Material X
   UMAT Subroutine = "MyUMATLib" "my_umat"
   ...
```

- You may also want to specify a path, for example

```
UMAT Subroutine = "./MyUMATLib" "my_umat"
```

# II User-defined material models

Special keywords:

- `Number of Material Constants`
- `Material Constants`: Ordering and consistent use are at the responsibility of the user
- `Number of State Variables`
- `Output State Variables`: set `True` in order to obtain stresses (`UmatStress`), energy variables (`UmatEnergy`), and additional state variables (`UmatState`) as fields associated with integration points
- `Initialize State Variables`: an optional extra call to obtain the state variables in the initial state
- `Name`

# II User-defined material models

Some remarks:

- `Calculate Stresses` and `Calculate Strains` create stresses and strains as nodal fields
- `Calculate Strains` produces the standard material strain
- However, UMAT can define the Cauchy stress to be a function of any strain measure which may be computed in terms of the deformation gradient (switches now to an inexact Newton method)
- With UMAT the in-built convergence criterion is always `"residual"`
- An incompressible material is not yet supported (via a mixed formulation with an additional pressure variable)

# II User-defined material models

The best place to find details is the template `UMATLib.F90` and its comment lines

- Specifies a constitutive law

$$\boldsymbol{\sigma}_m(\mathbf{p},t) = \bar{\boldsymbol{\sigma}}(\hat{\boldsymbol{E}}(\boldsymbol{F})(\mathbf{p},t), \mathbf{q}(\mathbf{p},t)).$$

- Here $\hat{\boldsymbol{E}}(\boldsymbol{F})$ is the strain field, $\boldsymbol{F}$ is the deformation gradient, and is $\mathbf{q} = (q_1, \ldots, q_N)$ a $N$-tuple of state variables
- The stress response function is a composition

$$\boldsymbol{F} \mapsto \bar{\boldsymbol{\sigma}}(\cdot, \mathbf{q}) \circ \hat{\boldsymbol{E}}(\boldsymbol{F}),$$

  so we can differentiate as

$$\boldsymbol{U} \mapsto D\bar{\boldsymbol{\sigma}}(\hat{\boldsymbol{E}}(\boldsymbol{F}), \mathbf{q})[D\hat{\boldsymbol{E}}(\boldsymbol{F})[\boldsymbol{U}]]$$

- The user must specify the derivative $D\bar{\boldsymbol{\sigma}}(\hat{\boldsymbol{E}}(\boldsymbol{F}), \mathbf{q})$
- If not possible in a closed form, an approximation may suffice

# II User-defined material models

For some additional details see also *Elmer Models Manual*

# III Coupling procedures

In principle two ways to couple different models:

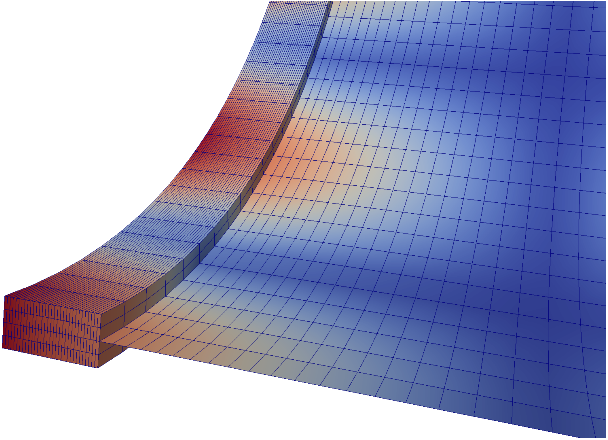
- a loose numerical coupling (the default strategy in Elmer)
- a tight numerical coupling: all unknowns updated/solved simultaneously

An implementation of a tight numerical coupling may not be an easy task:

- However, it may be the only practical way to handle a very strong physical coupling
- Gradual developments to enable tight coupling procedures

An example here: the coupling of a 2-D shell and a 3-D solid

# A graphical abstract: the coupling of solids and shells



# III Coupling procedures

Essential ingredients for enabling a tight coupling:

- An ability to construct a monolithic matrix from constituent blocks, for example to create:

$$\begin{bmatrix} K & D \\ H & A \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} F \\ G \end{bmatrix}$$

  where $K$ and $A$ are the stiffness matrices of 3D solid and shell parts
- Special keyword constructs/procedures so that existing solvers can be utilized to assemble the diagonal blocks
- Special assembly subroutines for creating coupling blocks (here $D$ and $H$)

# III Coupling procedures

Remarks:

- After a monolithic system has been created, its solution can be sought by applying a Krylov method
- The block matrix construct within Elmer is generic ⇒ should work similarly in different cases
- On the other hand, $D$ and $H$ don't exist as matrices when using a loose coupling ⇒ specific code needed
- Block preconditioning to combine the strengths of loose and tight numerical coupling

# III Coupling procedures

For details on writing a sif file for a tightly coupled model see Chapter 14 of ElmerSolver Manual, *“Block-matrix construct to build tightly coupled solvers”*

- Two solver sections needed as usual
- The first solver section to assemble the (1,1)-block and to control the solution of the fully coupled system
- The second solver section is subsidiary, integrating the (2,2)-block
- The keyword `Structure-Structure Coupling` activates the integration of interaction blocks

# III Coupling procedures

Special keywords:

- `Linear System Block Mode`: a main switch to create the linear system by using block construct
- `Block Solvers(2)`: pointers to solvers which define constituent blocks
- `Pre Solvers(1)`: activates the execution of a subsidiary solver in the assembly
- `Block Monolithic`: to create the coupled system as a single object
- `Shell Solver Index`: to inform that the coupling with the shell solver is wanted

# III Coupling procedures: Verification

- Static and eigenanalysis problems have been considered in verification
- The results have been compared with the results of alternate models of the same problem (for example a pure shell model or solid model)
- Seems to work
- Resources: See test cases `../fem/tests/Shell_with_Solid_*` in the code repository

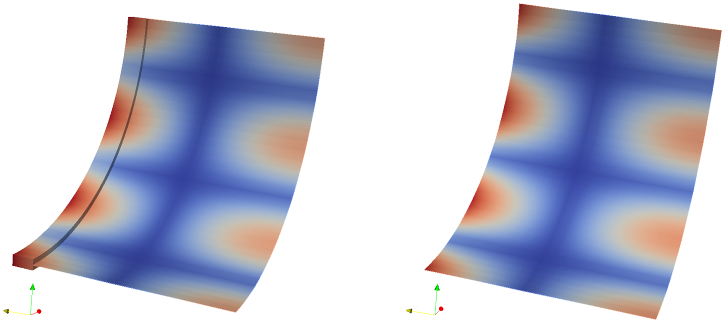
# III Coupling procedures: Verification

A cylindrical shell problem with bending-dominated asymptotic behaviour, see `.../tests/Shell_with_Solid_Eigenanalysis/Readme.txt`

| Shell and Solid | Shell |
|---|---|
| EigenSolve: 1: 4.441527E+04 | EigenSolve: 1: 4.291169E+04 |
| EigenSolve: 2: 1.302856E+06 | EigenSolve: 2: 1.255816E+06 |
| EigenSolve: 3: 4.885759E+06 | EigenSolve: 3: 4.836657E+06 |
| EigenSolve: 4: 7.316037E+06 | EigenSolve: 4: 7.036570E+06 |
| EigenSolve: 5: 1.032933E+07 | EigenSolve: 5: 1.039543E+07 |
| EigenSolve: 6: 1.169052E+07 | EigenSolve: 6: 1.118662E+07 |
| EigenSolve: 7: 2.382371E+07 | EigenSolve: 7: 2.296068E+07 |
| EigenSolve: 8: 2.429051E+07 | EigenSolve: 8: 2.428822E+07 |
| EigenSolve: 9: 2.473271E+07 | EigenSolve: 9: 2.503445E+07 |
| EigenSolve: 10: 2.594538E+07 | EigenSolve: 10: 2.591956E+07 |

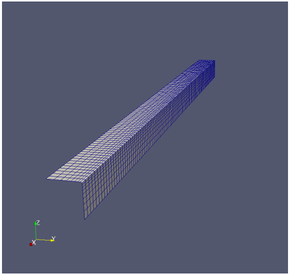
# III Coupling procedures: Verification

The sixth mode as given by the coupled model and the pure shell model (the 2-norm of the displacement vector)



# III Coupling procedures: Future work

- Some geometric constraints on the mesh
- Enabling parallelism
- Non-smooth shell mid-surfaces can in general be troublesome and then switching to a drilling rotation formulation seems to have a relative merit
- The construction of coupling blocks is not yet fully general for the drilling rotation formulation



# IV Concluding remarks

- In future divergence-conforming (and curl-conforming) basis functions could be utilized to create non-standard formulations
- Thanks for your attention!
- Questions or comments?