FEniCS-Shells:
A modern open-source extensible finite element implementation of linear and nonlinear plate and shell models

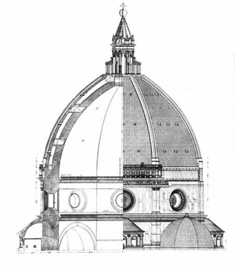
C. Maurini[1], M. Brunetti[1], J. Hale[2], S. Bordas[2]

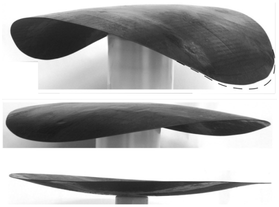
[1] Sorbonne Universités, UPMC Univ. Paris 06, Institut Jean Le Rond d'Alembert

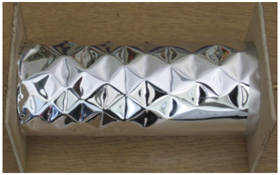
[2] Faculté des Sciences, de la Technologie et de la Communication, Université du Luxembourg

Seoul, 27 July 2016

Plates and shells: deformable surfaces



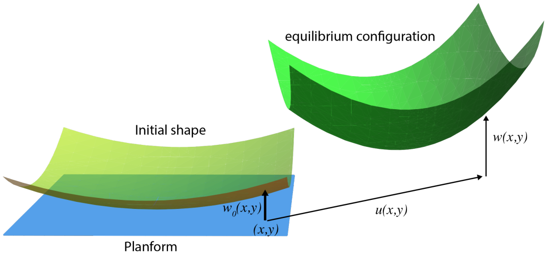




In thin shells the deformations are small and the behaviour is usually linear elastic.

Non-linear complex phenomena are due to geometry and shape.

Shell modelling



Numerical modelling of thin shells is difficult:

- Singular perturbation problem (thickness $t \to 0$)
- H^2 regularity requirement
- Differential geometry: PDE on surfaces
- Competition between membrane and stretching, with possible change of the PDE character with the sign of the Gaussian curvature
- Complex nonlinearities phenomena: buckling, failure, stress-focusing, point and line localisations.

Objectives of FEniCS-shells

- Provide an extensible open-source collection of high-quality numerical methods for thin structures
- Translate complex multiphysical problems on surface to performing scalable numerical implementation through simple formulations in `FEniCS`/ `UFL`
- Use in research (singularities in shells, multistable shells, fracture in shells) and teaching (structural mechanics, advanced numerical methods)

Based on FEniCS

The FEniCS Project is a collection of free software with an extensive list of features for automated, efficient solution of differential equations.

`https://fenicsproject.org`

FEniCS-shells: Implemented models

		Shearable	Bending	Linear	Nonlinear
✓	Reissner-Mindlin plate	●		●	
✓	von Kármán plate	●	●		●
✓	Marguerre shallow shell	●	●		●
✓	Naghdi shell	●			●

- Shearable models implemented through MITC/Duran-Liberman formulation [Dvorkin and Bathe 1986, Duran and Liberman 1992] in order to avoid numerical locking
- Bending models implemented through C/D Galerkin formulation [Engel et al., 2002] in order to avoid H^2-finite elements

Other non considered modern alternatives: subdivision of surfaces, IGA, meshless

The "simple" linear plate theory

Non-trivial numerical problems arise also for flat shells (plates) and linear models

Reissner-Mindlin (RM) plate model

- $w : \Omega \to R$: transverse displacement (scalar)
- $\theta : \Omega \to R^2$ rotation of transverse segments (vector)

$$\mathcal{E}_{\mathrm{RM}}(\theta, w) = \frac{1}{2}\int_\Omega D\nabla^s\boldsymbol{\theta} : \underbrace{\nabla^s\boldsymbol{\theta}}_{\text{curvature}} + \frac{t^{-2}}{2}\int_\Omega F \| \underbrace{\nabla w - \boldsymbol{\theta}}_{\text{shear}} \|^2 - \mathcal{L}_e(w, \theta)$$

Kirchhoff-Love (KL) plate model ($t \to 0$)

$$\mathcal{E}_{\mathrm{KL}}(w) = \frac{1}{2}\int_\Omega D\nabla\nabla w : \nabla\nabla w - \mathcal{L}_e(w)$$

Remark: when $t \to 0$ the RM-model asymptotically converges to the KL-model with $\nabla w = \boldsymbol{\theta}$.

Shear locking in the discrete problem in the RM model

Minimisation problem for the equilibrium of a plate of thickness t:

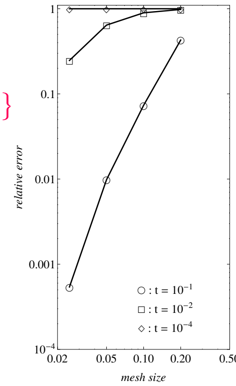
$$\min_{\theta,w\in V}\left\{\frac{1}{2}a(\theta,\theta)+\frac{t^{-2}}{2}\int_{\Omega}\|\nabla w-\theta\|^2-\mathcal{L}_e(w)\right\},\ V=H^1(\Omega,R^2)\times H^1(\Omega,R)$$

For $t\rightarrow 0$ the limit problem is

$$\min_{\theta,w\in K}\left\{\frac{1}{2}a(\theta,\theta)-\mathcal{L}_e\right\},K=\{(\theta,w)\in V:\nabla w=\theta\}$$

while the corresponding discrete problem has limit problem

$$\min_{\theta_h,w_h\in K_h}\left\{\frac{1}{2}a(\theta_h,\theta_h)-\mathcal{L}_e\right\},K_h=K\cap(\boldsymbol{\Theta}_h\times W_h)$$



If $(\boldsymbol{\Theta}_h\times W_h)$ is not large enough the basis functions cannot properly represent the Kirchhoff's constraint $\nabla w=\theta$ and we can have $K_h\equiv\emptyset$.

A well-established locking cure: MITC formulation

The MITC formulation recast the mixed formulation into a displacement form

$$\min_{\theta_h, w_h \in U_h} \left\{ \frac{1}{2} a(\theta_h, \theta_h) + \frac{t^{-2}}{2} \int_\Omega \| R_h(\underbrace{\nabla w_h - \theta_h}_{\gamma_h}) \|^2 - L_e \right\}$$

where the shear term is assembled using a reduction operator.

The element MITC3 (Bathe et al.) adopts an element-wise interpolation:

$$R_h \gamma_h = \left[\begin{array}{c} a - b\,y \\ a + c\,x \end{array} \right]$$

where the constants (a, b, c) are determined by imposing

$$R_h \gamma_h (x_i, y_i) = \gamma_h (x_i, y_i), \quad i = 1, 2, 3$$

at three tying points on the middle of the element edges (x_i, y_i)

Revisiting MITC as a mixed formulation

The locking issue can be solved also with the following mixed formulation

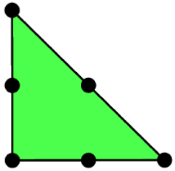
$$\mathrm{stat}\{\mathcal{E}, \quad (\theta, u, \gamma, \lambda) \in \Theta_h \times W_h \times \Gamma_h \times \Gamma_h\}$$

with

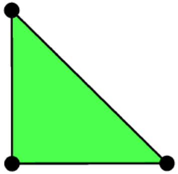
$$\mathcal{E}(\theta, u, \gamma, \lambda) = \left\{ \frac{1}{2} a(\theta, \theta) + \frac{t^{-2}}{2} \int_\Omega \|\gamma\|^2 - \mathcal{L}_e(w) + \int_\Omega \lambda \cdot ((\nabla w - \theta) - \gamma) \right\}$$

Duran-Liberman [Duran, Liberman, 1992] propose the following stable and uniformly convergent choice

$\Theta_h \equiv \mathtt{P}_2^2 \in H_1^2$

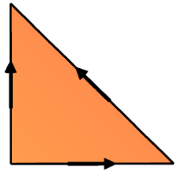


$W_h \equiv \mathtt{P}_1 \in H_1$



$(\nabla w - \theta) \in H(\mathrm{curl})$

$\Gamma_h \equiv \mathtt{NED}_1^{\mathtt{loc}} \in H(\mathrm{curl})$



Linear algebra and reduced formulation

Full space linear algebra formulation

$$\begin{bmatrix} A_{pp} & A_{pq} \\ A_{pq}^T & A_{qq} \end{bmatrix} \begin{bmatrix} U_p \\ U_q \end{bmatrix} = \begin{bmatrix} F_p \\ F_q \end{bmatrix}, \qquad \begin{cases} U_p = (\theta_h, w_h) \in {\mathtt{P}_2}^2 \times \mathtt{P}_1 \\ U_q = (\gamma_h, \lambda_h) \in \mathtt{NED}_1^{\mathtt{loc}} \times \mathtt{NED}_1^{\mathtt{loc}} \end{cases}$$

Linear algebra formulation after elimination

$$A_R U_P = F_R, \quad \begin{cases} A_R = A_{pp} - A_{pq} A_{qq}^{-1} A_{pq}^T \\ F_R = F_p - A_{pq} A_{qq}^{-} 1 F_q \end{cases}$$

- The reduction can be done as a static condensation at the element level.
- This kind of variational formulation is perfect for `FEniCS`. However static condensation is not possible because automatic code generation and assembling implementation.

FEniCS implementation

After the new/overloaded functions and classes of FEniCS-shells

- Define a FunctionSpace with the partition in primal and local variable

```
element = MixedElement([
       VectorElement("Lagrange", triangle, 2), FiniteElement("Lagrange", triangle, 1),
       FiniteElement("N1curl", triangle, 1), FiniteElement("N1curl", triangle, 1)])
U = ProjectedFunctionSpace(mesh, element, num_projected_subspaces=2)
U_F = U.full_space
U_P = U.projected_space
```

- Variational formulation including the constraint in the full space

```
kappa_ = grad(theta_); gamma_ = grad(w_) - theta_
bending_energy = (t**3*Et*((1.0 - nu)*tr(kappa_*kappa_) + nu*(tr(kappa_))**2))*dx
shear_energy = (t*Et*Constant(5.0/6.0)*inner(gamma_,gamma_)*dx
external_work = inner(Constant(1.0)*t**3, w_t)*dx
pontential_energy  =  bending_energy + shear_energy  - external_work
constraint = inner_e(gamma_ - R_gamma_, p_)
lagrangian = pontential_energy + constraint
d1_lagrangian = derivative(lagrangian, u_, u_t)
L = - replace(d1_lagrangian,{u_:u}) # linear form
a = derivative(d1_lagrangian, u_, u) # quadratic form
```

- Assemble with static condensation and solve

```
A, b = assemble(U_P, a, L)
solver.solve(A, u_p_.vector(), b)
reconstruct_full_space(u_, u_p_, a, L)
```

Implementing the static condensation in FEniCS

```
U = ProjectedFunctionSpace(mesh, element, num_projected_subspaces=2)
U_F = U.full_space
U_P = U.projected_space
A, b = assemble(U_P, a, L)
```

For e in elements:

1. Assemble the element matrices $\mathtt{A}^{\mathtt{e}}_{\mathtt{pp}}, \mathtt{A}^{\mathtt{e}}_{\mathtt{pq}}, \mathtt{A}^{\mathtt{e}}_{\mathtt{qq}}, \mathtt{F}^{\mathtt{e}}_{\mathtt{p}}, \mathtt{F}^{\mathtt{e}}_{\mathtt{q}}$ in the full space
2. Calculate through `Eigen` (dense linear algebra) the local static condensation

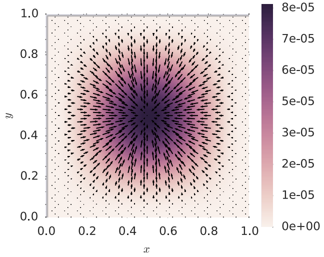
$$\begin{aligned}\mathtt{A}^{\mathtt{e}}_{\mathtt{R}} &\leftarrow \mathtt{A}^{\mathtt{e}}_{\mathtt{pp}} - \mathtt{A}^{\mathtt{e}}_{\mathtt{pq}}\mathtt{A}^{\mathtt{e}-1}_{\mathtt{qq}}\mathtt{A}^{\mathtt{eT}}_{\mathtt{pq}}\\ \mathtt{F}^{\mathtt{e}}_{\mathtt{R}} &\leftarrow \mathtt{F}^{\mathtt{e}}_{\mathtt{p}} - \mathtt{A}^{\mathtt{e}}_{\mathtt{pq}}\mathtt{A}^{\mathtt{e}-1}_{\mathtt{qq}}\mathtt{F}^{\mathtt{e}}_{\mathtt{q}}\end{aligned}$$

3. Add to global stiffness on the reduced space using the `dofmap` of $\mathtt{U}_{\mathtt{P}}$

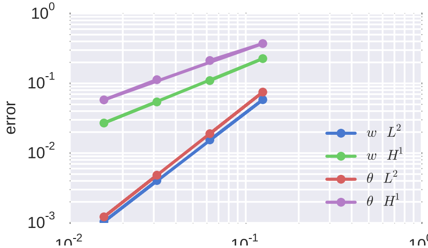
Note: As an option, $\mathtt{A}, \mathtt{b} = \mathtt{assemble}(\mathtt{U}_{\mathtt{F}}, \mathtt{a}, \mathtt{L})$ assemble the matrix on the full space without static condensation.

RM plate: convergence

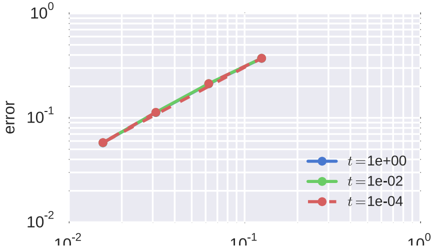
Clamped plate test (analytical solution by [Lovadina, 1995])



Convergence for thin plate ($t = 0.001$)

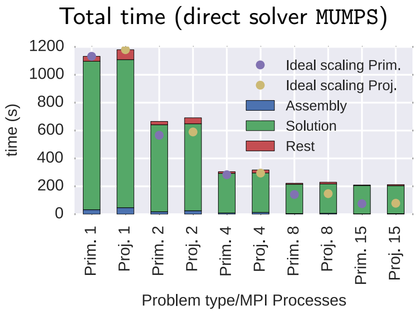


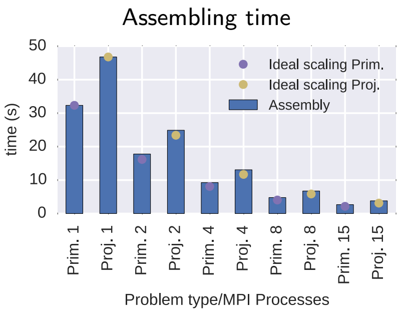
Convergence for several thichnesses



Assembling performance

Strong scaling with $\sim 2e^6$ elements, $\sim 10e^6$ dofs





Comments

- Assembling time is negligible
- Assembling scales almost perfectly
- Iterative solvers/preconditioners are needed (difficult issue)

Extension to nonlinear shells

Enjoy FEniCS/UFL syntax to write the variational formulation

```
# Naghdi strain measures
# Director vector and
d = lambda theta: as_vector([sin(theta[1])*cos(theta[0]), -sin(theta[0]), cos(theta[1])*cos(the
# Deformation gradient and strain measures
F = lambda u: as_tensor([[1.0, 0.0],[0.0, 1.0],[Constant(0), Constant(0)]]) + grad(u)
G = lambda F0: 0.5*(F0.T*F0 - Identity(2)) # Stretching tensor (1st Naghdi strain measure)
K = lambda F0, d: 0.5*(F0.T*grad(d) + grad(d).T*F0) # Curvature tensor (2nd Naghdi strain meas
g = lambda F0, d: F0.T*d # Shear strain vector (3rd Naghdi strain measure)
# Dual stress tensor (constitutive law)
S = lambda X: 2.0*mu*X + ((2.0*mu*lb)/(2.0*mu + lb))*tr(X)*Identity(2)
# Energies
psi_G = (2.0*eps)*inner(S(G(F(z_))), G(F(z_))) # Membrane energy density
psi_K = ((2.0*eps)**3/12.0)*inner(S(K(F(z_), d(theta_))), K(F(z_), d(theta_))) # Bending energ
psi_g = eps*2.0*mu*inner(Rgamma_, Rgamma_) # Shear energy density
psi = 0.5*(psi_G + psi_K + psi_g)*dx # Stored strain energy density
# constraint
L_R = inner_e(g(F(z_), d(theta_)) - Rgamma_, p_)
```

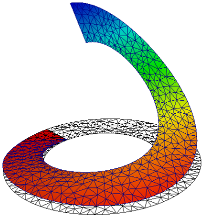
So

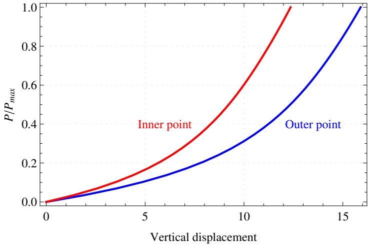
in the projected space a customised NonlinearProblem using the condensed assembler

```
problem = ProjectedNonlinearProblem(U_P, F, u_, u_p_, bcs=bcs, J=J)
solver = NewtonSolver()
solver.solve(problem, u_p_.vector())
```

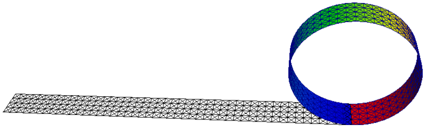
Large transformations of nonlinear shells

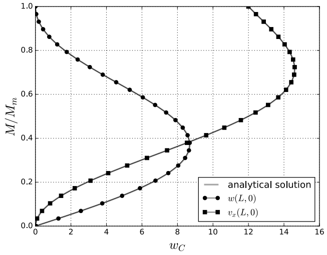
Pulling of a slit annular plate





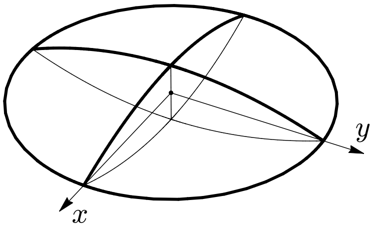
Rolling of a strip





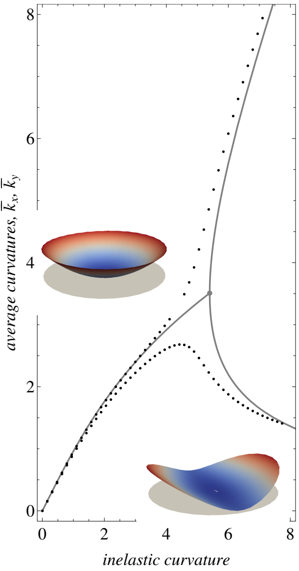
Bistable disc with spherical target curvature

Circular plate with lenticular cross section subjected to a temperature gradient through its thickness



Average curvatures at bifurcation [Mansfield, 1962]

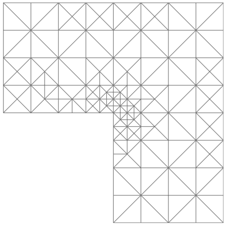
$$\kappa_T = \frac{8}{(1+\nu)^{3/2}}$$

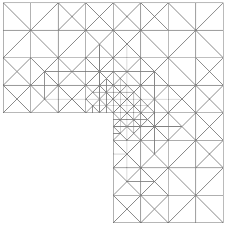


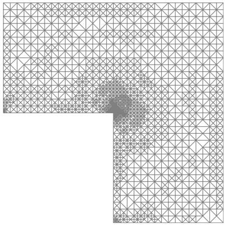
Adaptive remeshing

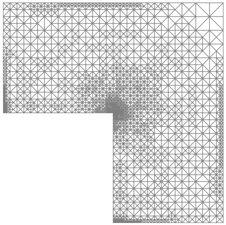
Thick plate

Thin plate









Conclusions an perspectives

- HPC open-source extensible tool for shell computations
- Projected and full formulations have similar performances
- Projected formulation is better for memory.
- Projected formulation is practical for stability analysis
- Full formulation is probably better for preconditioners (todo)
- Higher order `MITC7` are possible (`MITC7`), the advantage of static condensation would be more evident
- Working on curvilinear coordinate mapping
- Aim at applications on growth, morphing structures, fracture with phase-field models, shape control, multiphysical coupling, analysis of singular structures in shells (d-cone, streching ridge)

Codes available at

`https://bitbucket.org/unilucompmech/fenics-shells`

We provide functional `docker` containers to run the application!

Mail to `corrado.maurini@upmc.fr` if interested in use/collaboration