Towards effective shell modelling with the FEniCS project

J. S. Hale*, P. M. Baiz
Department of Aeronautics

19th March 2013
Outline

- Introduction
- Shells:
  - chart
  - shear-membrane-bending and membrane-bending models
  - example forms
- Two proposals for discussion:
  - geometry: chart object for describing shell geometry
  - discretisation: projection/reduction operators for implementation of generalised displacement methods
- Summary
So far...

- dolfin manifold support already underway, merged into trunk [Marie Rognes, David Ham, Colin Cotter]
- I have already implemented locking-free (uncurved) beams and plate structures using dolfin manifold
- Next step: curved surfaces, generalised displacement methods (?)
- Aim of my talk is to start discussion on the best path
Why shells?

- **The mathematics:** Shells are three-dimensional elastic bodies which occupy a ‘thin’ region around a two-dimensional manifold situated in three-dimensional space.

- **The practical advantages:** Shell structures can hold huge applied loads over large areas using a relatively small amount of material. Therefore they are used abundantly in almost all areas of mechanical, civil and aeronautical engineering.

- **The computational advantages:** A three-dimensional problem is reduced to a two-dimensional problem. Quantities of engineering relevance are computed directly.
Figure: British Museum Great Court. Source: Wikimedia Commons.
Figure: Specialized OSBB bottom bracket. Source: bikeradar.com
A huge field

There are *many* different ways of:

- obtaining shell models
- representing the geometry of surfaces on computers
- discretising shell models successfully

And therefore we need suitable *abstractions* to ensure generality and extensibility of any shell modelling capabilities in FEniCS.
Two methodologies

Mathematical Model approach:
1. Derive a mathematical shell model.
2. Discretise that model using appropriate numerical method for description of geometry and fields.

Degenerated Solid approach:
1. Begin with a general 3D variational formulation for the shell body.
2. Degenerate a solid 3D element by inferring appropriate FE interpolation at a number of discrete points.
3. No explicit mathematical shell model, one may be implied.
Mathematical model

\[
t = \frac{\varphi_1 \times \varphi_2}{\| \varphi_1 \times \varphi_2 \|}
\]

\[
\varphi(\zeta^1, \zeta^2) : \Omega \rightarrow \mathbb{R}^3
\]

\[
(\zeta^1, \zeta^2) \in \Omega \subset \mathbb{R}^2
\]
shear-membrane-bending (smb) model

Find $U \in \mathcal{V}_{smb}$:

$$h^3 A_b(U, V) + h A_s(U, V) + h A_m(U, V) = F(V) \quad \forall V \in \mathcal{V}_{smb} \quad (1)$$

membrane-bending (mb) model

Find $U \in \mathcal{V}_{mb}$:

$$h^3 A_b(U, V) + h A_m(U, V) = F(V) \quad \forall V \in \mathcal{V}_{mb} \quad (2)$$
smb models vs mb models

- smb model takes into account the effects of shear; ‘closer’ to the 3D solution for thick shells, matches the mb model for thin shells.
- Boundary conditions are better represented in smb model; hard and soft supports, boundary layers.
- smb $U \in H^1(\Omega)$ vs mb $U \in H^2(\Omega)$
Mathematical model

Let’s just take a look at the bending bilinear form $A_b$ for the mb model:

$$A_b(U, V) = \int_{\Omega} \rho_{\alpha\beta} H_b^{\alpha\beta\gamma\delta} \rho_{\gamma\delta} \, dA$$  \hspace{1cm} (3a)

$$\rho_{\alpha\beta} := \varphi_{,\alpha\beta} \cdot t \frac{1}{j} (u,1 \cdot (\varphi,2 \times t) - u,2 \cdot (\varphi,1 \times t))$$  \hspace{1cm} (3b)

$$+ \frac{1}{j} (u,1 \cdot (\varphi_{,\alpha\beta} \times \varphi,2 - u,2 \cdot (\varphi_{,\alpha\beta} \times \varphi,1))$$

$$- u_{,\alpha\beta} \cdot t$$

$$H_b^{\alpha\beta\gamma\delta} := \frac{Eh^3}{12(1 - \nu^2)} \left( \nu (\varphi^{\alpha} \cdot \varphi^{,\beta}) + \ldots \right)$$  \hspace{1cm} (3c)
**Continuous model**

Terms describing the differential geometry of the shell mid-surface. The mid-surface is defined by the chart function.

**Discrete model**

We do not (usually) have an explicit representation of the chart. It must be constructed implicitly from the mesh and/or data from a CAD model. There are many different ways of doing this.
Geometry

Proposal 1

A base class `Chart` object which exposes various new symbols describing the geometry of the shell surface. Specific subclasses of `Chart` will implement a particular computational geometry procedure. The user can then express their mathematical shell model independently from the underlying geometrical procedure using the provided high-level symbols.
shell_mesh = mesh("shell.xml")
normals = MeshFunction(...)
C = FunctionSpace(shell_mesh, "CG", 2)

chart = Chart(shell_mesh, C, 
               method="patch_averaged")
chart = Chart(shell_mesh, C, 
               method="CAD_normals", normals=normals)
...
b_cnt = chart.contravariant_basis()
b_cov = chart.covariant_basis()
dA = chart.measure()
a = chart.first_fundamental_form()
...
A_b = ...
Current discretisation options

mb model:
- $H^2(\Omega)$ conforming finite elements
- DG methods

smb model:
- straight $H^1(\Omega)$ conforming finite elements
- mixed finite elements (CG, DG)
- generalised displacement methods
A quick note

For simplicity I will just talk about the smb model reduced to plates, the chart function is the identity matrix; considerably simpler asymptotic behaviour but concepts apply to shells also.
Locking

Inability of the basis functions to represent the limiting Kirchhoff mode.
Move to a mixed formulation

Treat the shear stress as an independent variational quantity:

\[
\gamma_h = \lambda \bar{t}^{-2}(\nabla z_3h - \theta_h) \in S_h
\]

Discrete Mixed Weak Form

Find \((z_3h, \theta_h, \gamma_h) \in (V_3h, R_h, S_h)\) such that for all \((y_3h, \eta, \psi) \in (V_3h, R_h, S_h)\):

\[
a_b(\theta_h; \eta) + (\gamma_h; \nabla y_3 - \eta)_{L^2} = f(y_3)
\]

\[
(\nabla z_3h - \theta_h; \psi)_{L^2} - \frac{\bar{t}^2}{\lambda}(\gamma_h; \psi)_{L^2} = 0
\]
Move back to a displacement formulation

Linear algebra level: Eliminate the shear stress unknowns \textit{a priori} to solution

\[
\begin{bmatrix}
A & B \\
B^T & C
\end{bmatrix}
\begin{bmatrix}
u \\
\gamma
\end{bmatrix} = \begin{bmatrix}
f \\
0
\end{bmatrix}
\] (5)

To do this we can rearrange the second equation and then if and only if \( C \) is diagonal/block-diagonal we can invert cheaply giving a problem in original displacement unknowns:

\[
(A + BC^{-1}B^T)u = f
\] (6)
Currently, this can be done with CBC.Block TRIA0220, TRIA1B20 [Arnold and Brezzi][Boffi and Lovadina]
https://answers.launchpad.net/dolfin/+question/143195 David Ham, Kent Andre-Mardal, Anders Logg, Joachim Haga and myself

\[
A, B, BT, C = [\text{assemble}(a), \text{assemble}(b), \\
              \text{assemble}(bt), \text{assemble}(c)]
\]

\[
K = \text{collapse}(A - B * \text{LumpedInvDiag}(C) * BT)
\]
Move back to a displacement formulation

Variational form level:

\[ \gamma_h = \lambda \bar{t}^{-2} \Pi_h (\nabla z_{3h} - \theta_h) \]  \hspace{1cm} (7)

\[ \Pi_h : \mathcal{V}_{3h} \times \mathcal{R}_h \rightarrow \mathcal{S}_h \]  \hspace{1cm} (8)

giving:

\[ a_b(\theta_h; \eta) + (\Pi_h (\nabla z_3 - \theta); \nabla y_3 - \eta)_{L^2} = f(y_3) \]  \hspace{1cm} (9)
Proposal 2

A new class `Projection` in UFL that signals to FFC that a projection between `FunctionSpace` objects is required. This requires additions in DOLFIN, UFL, FFC and FIAT.
MITC7

\[ \mathcal{V}_{3h} \]
\[ \mathcal{R}_h \]
\[ \mathcal{S}_h \]
\begin{verbatim}
... 
V_3 = FunctionSpace(mesh, "CG", 2) 
R = VectorFunctionSpace(mesh, "CG", 2, dim=2) + 
    VectorFunctionSpace(mesh,"B", 3, dim=2) 
S = FunctionSpace(mesh, "N1curl", order=2) 
Pi_h = Projection(from=R, to=S)
...
U = MixedFunctionSpace([R, V_3]) 
theta, z_3 = TrialFunctions(U) 
eta, y_3 = TestFunctions(U) 
a_s = inner(grad(z_3) - Pi_h(theta), grad(y_3) 
    - Pi_h(eta))*dx
\end{verbatim}
Summary

- A big field with lots of approaches; need appropriate abstractions (inc. other PDEs on surfaces)
- Proposal 1: Expression of geometric terms in shell models using a natural form language which reflects the underlying mathematics
- Proposal 2: Effective discretisation options for the implementation of generalised displacement methods
Thanks for listening.