

The shell finite element explained

Understand the shell finite element mechanical assumptions. And learn step by step how to derive the shell element stiffness matrix.

📅 Posted on April 7, 2020 | ⏰ 19 minutes | 📖 4007 words | 🙃 Ali Baba

The development of an **efficient shell element** has gathered a lot of work since the beginning of structural finite element history. For multiple reasons, **developing an effective shell element is not an easy task**, and we will cover some of these reasons through this article. Multiple shell elements are available in the literature (https://en.wikipedia.org/wiki/Plate_theory), and they are still being improved as time goes on. For this article I have decided to introduce the MITC shell element (Mixed Interpolation of Tensorial Components) because it recently received new improvements making it **one of the most practical and efficient shell element**. After detailing the various assumptions and how they work together, I will explain how to derive the shell element stiffness matrix. Finally, I will conclude this article with an example of an hyperboloid-like model simulated using the SesamX finite element analysis software (/) (implementing the element presented here).

There are 2 main kinds of shell element depending on their underlying geometry: quadrangular and triangular. In this article I will **focus on the MITC4 element** (quadrangular) because it is the most widespread among the MITC shell family.

Relevant papers about the MITC4 shell finite element

This article is mainly inspired from the following papers about the MITC4 element. They are great resources to consult if you are looking for deeper information about this element.

1. Dvorkin EN, Bathe KJ. A continuum mechanics based four-node shell element for general nonlinear analysis (<https://www.emerald.com/insight/content/doi/10.1108/eb023562/full/html>): original paper about the MITC4 element. It introduces the concept of mixed interpolation of tensorial components, and how it improves the transverse shear behavior of the element.
2. Ko Y, Lee PS, Bathe KJ. A new MITC4+ shell element (<https://www.sciencedirect.com/science/article/abs/pii/S0045794916309464>): recent paper exposing new improvements about the MITC4 element membrane behavior.
3. Ko Y, Lee PS, Bathe KJ. A new 4-node MITC element for analysis of two dimensional solids and its application in shell analyses (<https://www.sciencedirect.com/science/article/abs/pii/S0045794917307757>): other improvements about the MITC4 element membrane behavior.
4. Ko Y, Lee Y, Lee PS, Bathe KJ. Performance of the MITC3+ and MITC4+ shell elements in widely-used benchmark problems (<https://www.sciencedirect.com/science/article/abs/pii/S0045794917309550>): comparison of the improved MITC4 element against other well known shell elements (including Abaqus S4 element).

Quick remark on notations

Before starting, let's define some notations that are used through this article:

- vectors are denoted with an underline \underline{u} and tensors with a double underline $\underline{\underline{\epsilon}}$,
- matrices are represented with brackets, as well as vector and tensor components $[]$,
- Einstein summation convention (<https://mathworld.wolfram.com/EinsteinSummation.html>) is used on repeated indices,

- partial derivatives are denoted with the comma notation $u_{,x}$,
- infinitesimal strain and stress tensors are represented in column matrix notation

$$[\varepsilon] = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{12} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \end{bmatrix} \text{ and } [\sigma] = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix}, \text{ where } [\varepsilon] \text{ represents the } \mathbf{\text{engineering strains}}.$$

Introducing the shell element

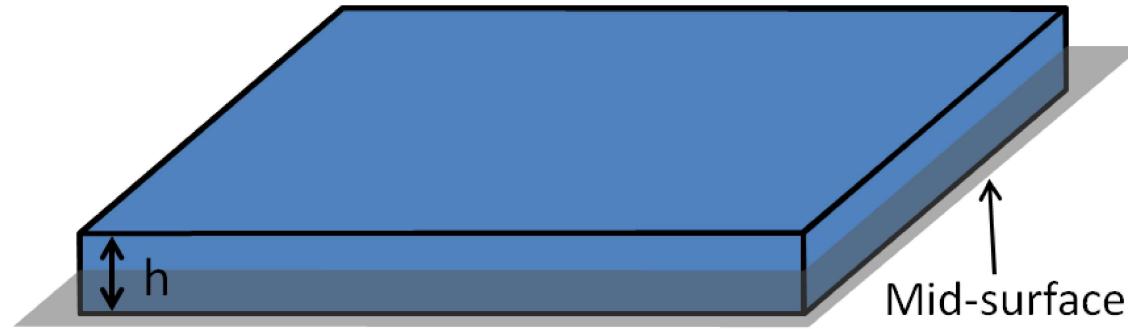
The shell element (https://en.wikipedia.org/wiki/Thin-shell_structure) is intended to analyze **thin structures** undergoing forces and moments in any direction. The structure can be flat or curved but it should have **one dimension smaller than the other 2**. For instance, it is the perfect element to analyze a slab or an aircraft fuselage.

	
Slab	Aircraft fuselage

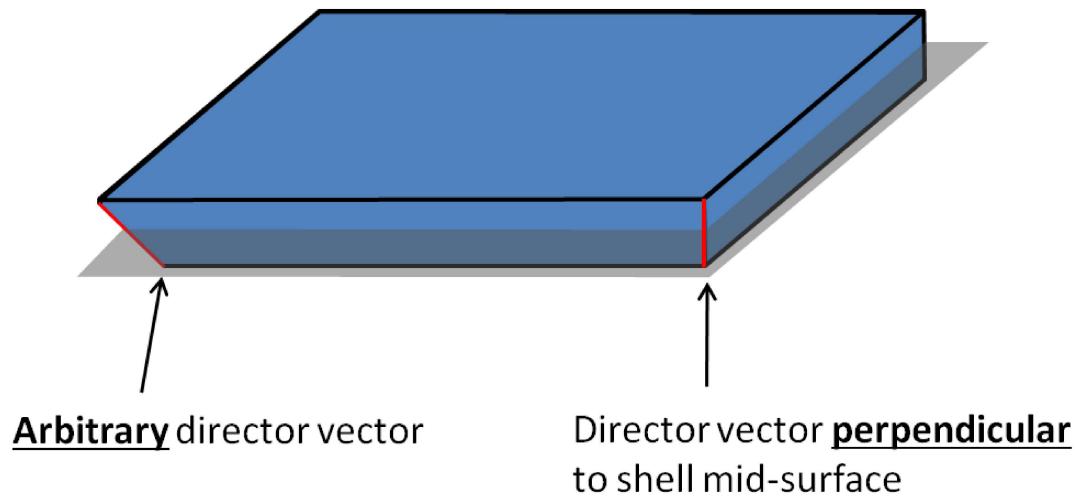
To fully define the shell geometry, the element needs the following 3 characteristics:

- The **mid-surface geometry**,
- The **thickness** at each point of the mid-surface: h ,
- The **director vector** at each point of the mid-surface: \underline{v}

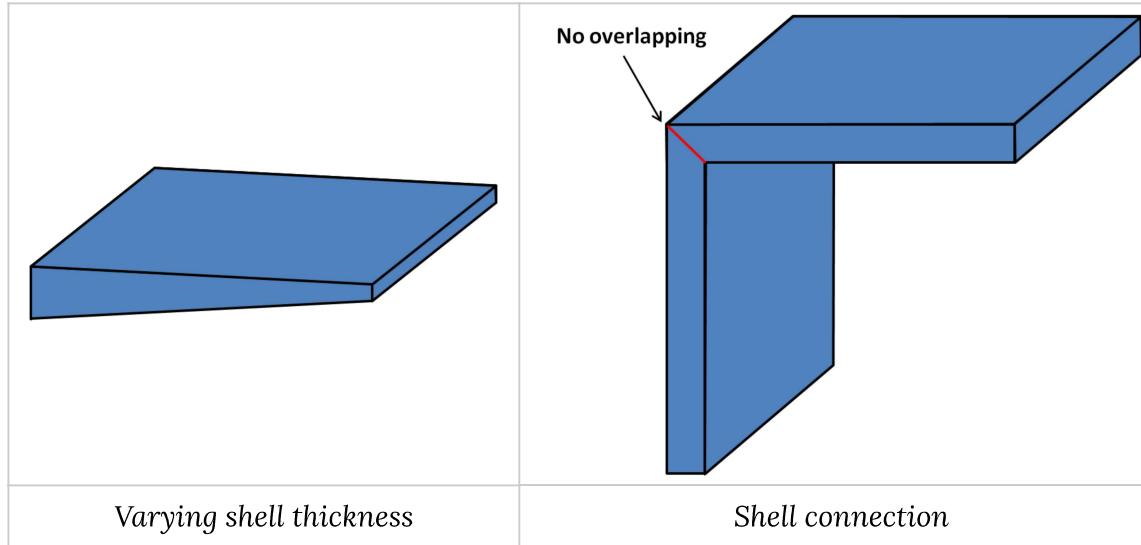
The mid-surface and thickness are what comes naturally in mind when thinking about shells.



The director vector is a specific MITC feature. It allows to control how the “cross section” is tilted relative to the mid-surface.



These features make it possible to modelize shells with varying thickness along the mid-surface, as well as arbitrary shells connections.



Furthermore, this element is not restricted to flat shell and **can also represent curved geometries**. The element is meshed on the mid-surface using 4 nodes. At each node I the following quantities are available:

- the **position** of the node \underline{X}^I ,
- the **thickness** at the node h^I ,
- the **shell director vector** at the node \underline{v}^I ,
- the **displacement vector** at the node \underline{u}^I ,
- and the **rotation vector** at the node $\underline{\theta}^I$.



$$\underline{X}^I, h^I, \underline{v}^I, \underline{u}^I, \underline{\theta}^I$$

MITC4 shell element assumptions

In essence, the shell element is a simplification of a more complex 3D structure. When designing such an element, the aim is to decrease the computational cost, while making relevant assumptions on how the element behaves. The main kinematic assumptions underlying the MITC element are given here with some explanations.

Stress assumption

As the shell must be able to sustain transverse and plane loading, the stress state must allow for transverse shear stress, as well as in plane stress. It is then sufficient to enforce:

$$\sigma_{33} = 0 \quad (1)$$

This is **the classical stress assumption** made for shell elements when the “3” direction is normal to the mid-surface. However, in the case of the MITC element, **the “3” direction is collinear to the director vector at each point of the shell mid-surface** (basis and coordinate systems are detailed further ahead in this article).

Kinematic assumption

Before detailing the kinematic assumption, we need to define a convenient parametric coordinate system on the shell.

Parametric coordinates system and position vector

We rely on the position vector to define the parametric system. Without providing more details, the first 2 coordinates r and s can be any of the mid-surface curvilinear coordinates. The last coordinate t is such that for a given r and s , t evolves when we move along the director vector.

Using these parametric coordinates, we can describe the **position of every point in the space**:

$$\underline{X}(r, s, t) = \underline{X}_0(r, s) + \frac{t}{2} h(r, s) \underline{v}(r, s) \quad (2)$$

Where $\underline{X}_0(r, s)$ **describes the mid-surface position**. To define clearly what r and s are, we need to detail further $\underline{X}_0(r, s)$, $h(r, s)$ and $\underline{v}(r, s)$.

Using the position of the nodes and the shape functions we have:

- $\underline{X}_0(r, s) = N^I(r, s)\underline{X}^I$
- $h(r, s)\underline{v}(r, s) = N^I(r, s)h^I\underline{v}^I$

Where the shape functions N^I are:

$$\begin{cases} N^1(r, s) = \frac{1}{4}(1-r)(1-s) \\ N^2(r, s) = \frac{1}{4}(1+r)(1-s) \\ N^3(r, s) = \frac{1}{4}(1+r)(1+s) \\ N^4(r, s) = \frac{1}{4}(1-r)(1+s) \end{cases}$$

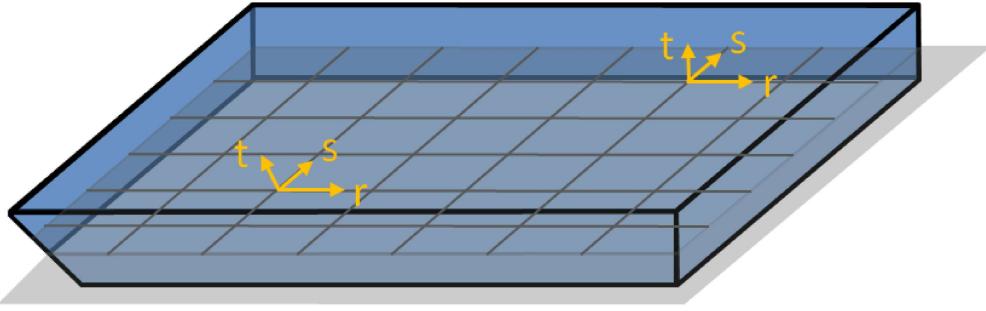
Which yields the full expression of the **position from the parametric coordinates**:

$$\underline{X}(r, s, t) = N^I(r, s)\underline{X}^I + \frac{t}{2}N^I(r, s)h^I\underline{v}^I$$

(3)

Please note the following remarks:

- for $t = 0$ the domain $r, s \in [-1, 1]$ describes the shell mid-surface,
- t moving from -1 to 1 corresponds to a point moving from the inferior face to the superior face of the shell,
- the full product $h(r, s)\underline{v}(r, s)$ is linearly interpolated from its values at the nodes (we want to stress that $h(r, s)$ and $\underline{v}(r, s)$ are not interpolated on their own but through their product),
- even though expression (3) is useful to define the parametric space, we use extensively expression (2) in the following for its conciseness.



Displacement vector

Next we express the **displacement vector at each point inside the shell**, relative to its parametric coordinates. We have:

$$\underline{u}(r, s, t) = \underline{u}_0(r, s) + \frac{t}{2}h(r, s)(\underline{\theta}(r, s) \times \underline{v}(r, s))$$

Where:

- $\underline{u}_0(r, s)$ is the displacement vector on the shell mid-surface,
- $\underline{\theta}(r, s)$ is the rotation vector on the shell mid-surface.

We see from this formula that, for a given point on the mid-surface, the rotation vector $\underline{\theta}(r, s)$ describes simply the rotation of the director vector. Hence

$\frac{t}{2}h(r, s)(\underline{\theta}(r, s) \times \underline{v}(r, s))$ is simply the displacement resulting from the director vector rotation.

However, this comes at a price. In general, if we seek to compute the rotation of a vector \underline{v} under a given rotation vector $\underline{\theta}$, we have to apply Rodrigues' formula (https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula). **This formula is non linear on $\underline{\theta}$** , but can be simplified to $\underline{\theta} \times \underline{v}$ if we assume that $\underline{\theta}$ is small (which is the case here).

As for the position vector, we relate the **displacement inside the shell from the displacements and rotations at the nodes**:

$$\underline{u}(r, s, t) = N^I(r, s)\underline{u}^I + \frac{t}{2}N^I(r, s)h^I(\underline{\theta}^I \times \underline{v}^I)$$

And we note again that the full term $h(r, s)(\underline{\theta}(r, s) \times \underline{v}(r, s))$ is interpolated at once from the values at the nodes. To make things easier, we can introduce the vector $\underline{\phi} = \underline{\theta} \times \underline{v}$ to finally write:

$$\boxed{\underline{u}(r, s, t) = \underline{u}_0(r, s) + \frac{t}{2}h(r, s)\underline{\phi}(r, s)} \quad (4)$$

And after interpolation from the values at the nodes:

$$\boxed{\underline{u}(r, s, t) = N^I(r, s)\underline{u}^I + \frac{t}{2}N^I(r, s)h^I\underline{\phi}^I} \quad (5)$$

Basis and coordinate systems

Before going further, let's define and summarize the different basis and coordinate systems that are used while deriving the stiffness matrix.

Covariant basis

We have defined previously the parametric coordinate system (r, s, t) . It is time now to give **the associated covariant basis**

(https://en.wikipedia.org/wiki/Covariance_and_contravariance_of_vectors).

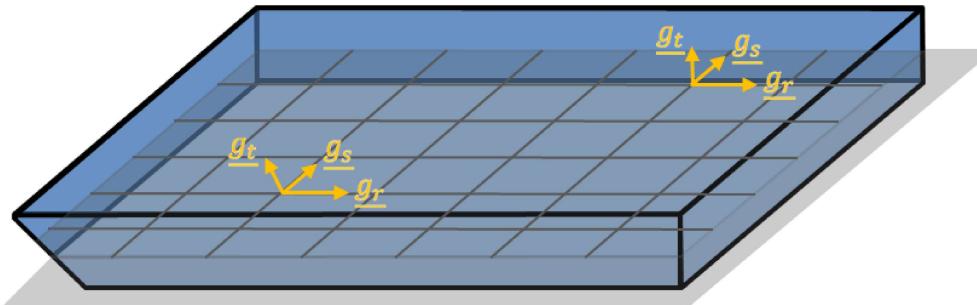
Using (2), this basis is defined as:

$$\begin{cases} \underline{g}_r(r, s, t) = \underline{X}_{,r} = \underline{X}_0_{,r} + \frac{t}{2}(h\underline{v})_{,r} \\ \underline{g}_s(r, s, t) = \underline{X}_{,s} = \underline{X}_0_{,s} + \frac{t}{2}(h\underline{v})_{,s} \\ \underline{g}_t(r, s) = \underline{X}_{,t} = \frac{1}{2}h\underline{v} \end{cases} \quad (6)$$

Remarks:

- \underline{g}_r and \underline{g}_s depend on r , s and t while \underline{g}_t depends only on r and s ,
- if the element has the same director vectors and thickness at the nodes, then hv is constant, and now \underline{g}_r and \underline{g}_s depend only on r and s ,

In a sense, **the covariant basis “follows” the geometry of the element**. Therefore, it is used to compute specific strains inside the element.



Contravariant basis

We will also use the **contravariant basis** associated with the previous covariant one. It is defined as follow:

$$\begin{cases} \underline{g}^r(r, s, t) = \frac{\underline{g}_s \times \underline{g}_t}{\underline{g}_r \cdot (\underline{g}_s \times \underline{g}_t)} \\ \underline{g}^s(r, s, t) = \frac{\underline{g}_t \times \underline{g}_r}{\underline{g}_r \cdot (\underline{g}_s \times \underline{g}_t)} \\ \underline{g}^t(r, s, t) = \frac{\underline{g}_r \times \underline{g}_s}{\underline{g}_r \cdot (\underline{g}_s \times \underline{g}_t)} \end{cases} \quad (7)$$

This basis is used conjointly with the covariant basis to manipulate the strain tensor. In fact, it has the following properties: $\underline{g}^i \cdot \underline{g}_j = 0 \forall i, j i \neq j$ and $\underline{g}^i \cdot \underline{g}_i = 1$

Local basis and local Cartesian coordinate system

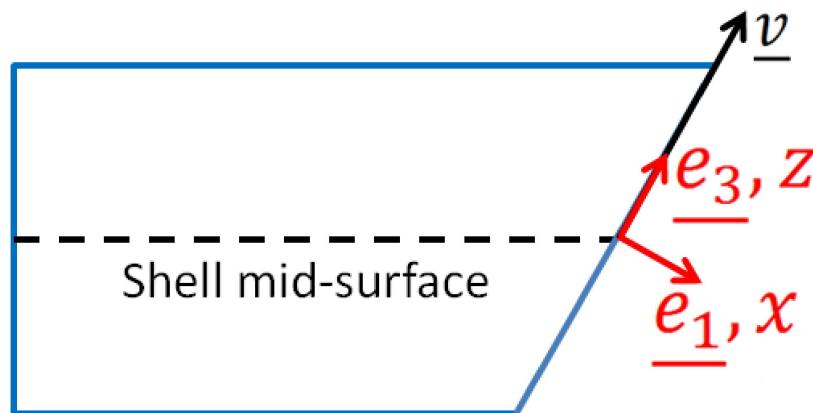
From the covariant basis, we also define a **Cartesian local basis** ($\underline{e}_1, \underline{e}_2, \underline{e}_3$) as follow:

$$\begin{cases} \underline{e}_3(r, s) = \frac{\underline{g}_t}{\|\underline{g}_t\|} \\ \underline{e}_1(r, s, t) = \frac{\underline{g}_s \times \underline{e}_3}{\|\underline{g}_s \times \underline{e}_3\|} \\ \underline{e}_2(r, s, t) = \underline{e}_3 \times \underline{e}_1 \end{cases} \quad (8)$$

And from this local basis, we define the **local Cartesian coordinate system** (x, y, z). We note that \underline{e}_3 is not perpendicular to the shell mid-surface but is aligned with \underline{v} .

The local basis is used to express the stress strain relation for the shell. The stress assumption given in (1) applies in this basis. While the local Cartesian coordinate system is used to integrate the virtual work over the element volume.

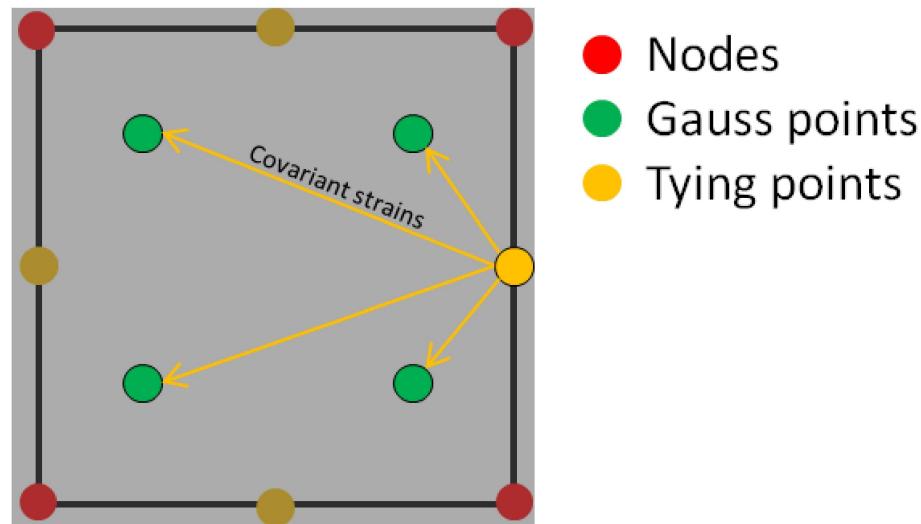
Note that when \underline{v} is not perpendicular to the mid-surface, the plane described by \underline{e}_1 and \underline{e}_2 is not on the mid-surface.



MITC4 shell element derivation

The biggest challenge regarding shell element design is to **avoid shear and membrane locking**, namely the tendency of the element to represent too rigidly shear and membrane motions. Multiple techniques have been employed to alleviate this issue, each having its pros and cons. Regarding the MITC, this issue is tackled without adding extra degree of freedom or instability to the element. Which makes it **an efficient and conceptually simple shell element**.

In fact, it relies on assuming specific transverse shear and membrane strains interpolation inside the element from strain values at tying points. And these points are specifically chosen to avoid locking. **The fundamental assumption underlying the MITC element is the use of the covariant strain components** to perform this interpolation. Roughly speaking, covariant strain components at the tying points are manipulated and inserted as covariant strain components at any point (for instance Gauss points). Hence the covariant / contravariant basis act as a hub connecting the strains from multiple points.



Applying the classical finite element method (https://www.sesamx.io/blog/deep_finite_element_method_introduction/) we can derive the shell element stiffness matrix.

Strain tensor

The strain tensor can be expressed in whatever basis we choose. To go on with the MITC procedure, we need to express the strains in the contravariant basis to get the covariant strain components. Then the tying procedure is applied on some of these terms to improve the element behavior.

Covariant strain tensor components

From definition, the **covariant strains components** are given by:

$$\underline{\underline{\varepsilon}} = \widetilde{\varepsilon}_{ij} \underline{g}^i \otimes \underline{g}^j \quad \forall i, j = r, s, t$$

And it can be shown that:

$$\widetilde{\varepsilon}_{ij} = \frac{1}{2}(\underline{u}_{,i} \cdot \underline{g}_j + \underline{u}_{,j} \cdot \underline{g}_i)$$

Using (4) we can simplify further this expression:

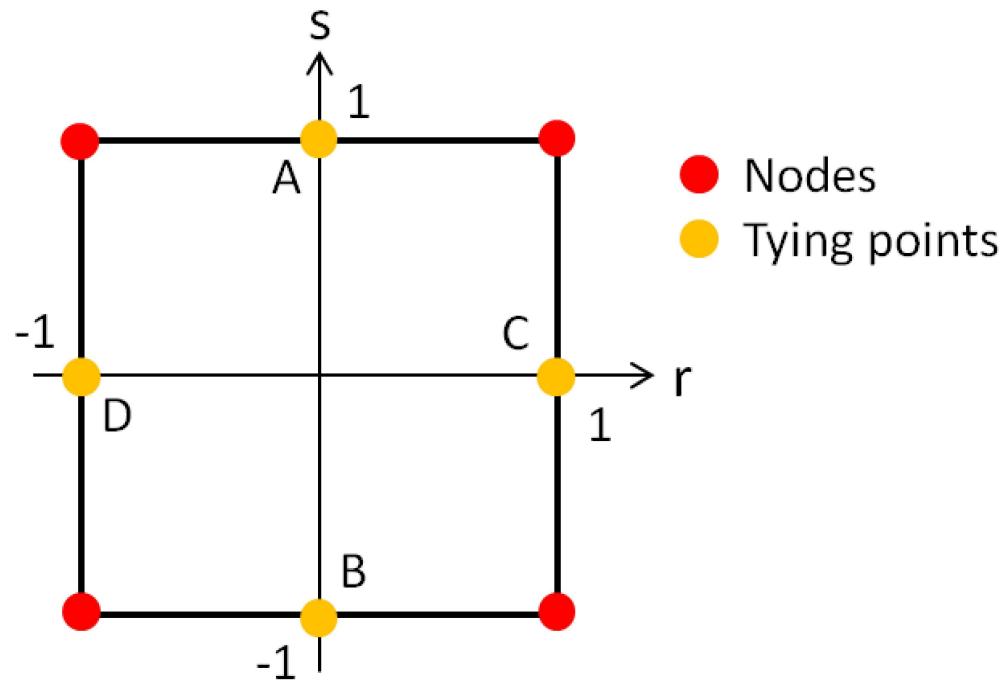
$$\left\{ \begin{array}{l} \widetilde{\varepsilon}_{rr} = \underline{u}_{0,r} \cdot \underline{g}_r + \frac{t}{2}(h\underline{\phi})_{,r} \cdot \underline{g}_r \\ \widetilde{\varepsilon}_{ss} = \underline{u}_{0,s} \cdot \underline{g}_s + \frac{t}{2}(h\underline{\phi})_{,s} \cdot \underline{g}_s \\ \widetilde{\varepsilon}_{tt} = \frac{1}{2}h\underline{\phi} \cdot \underline{g}_t = 0 \\ \widetilde{\varepsilon}_{rs} = \frac{1}{2}(\underline{u}_{0,r} \cdot \underline{g}_s + \underline{u}_{0,s} \cdot \underline{g}_r) + \frac{t}{4}((h\underline{\phi})_{,r} \cdot \underline{g}_s + (h\underline{\phi})_{,s} \cdot \underline{g}_r) \\ \widetilde{\varepsilon}_{st} = \frac{1}{2}(\underline{u}_{0,s} \cdot \underline{g}_t) + \frac{1}{4}(h\underline{\phi} \cdot \underline{g}_s) + \frac{t}{4}((h\underline{\phi})_{,s} \cdot \underline{g}_t) \\ \widetilde{\varepsilon}_{rt} = \frac{1}{2}(\underline{u}_{0,r} \cdot \underline{g}_t) + \frac{1}{4}(h\underline{\phi} \cdot \underline{g}_r) + \frac{t}{4}((h\underline{\phi})_{,r} \cdot \underline{g}_t) \end{array} \right. \quad (9)$$

Where the displacement and rotation vectors (through $\underline{\phi}$) appear more clearly. We note that $\widetilde{\varepsilon}_{tt} = 0$ because by definition $\underline{\phi}$ is perpendicular to \underline{g}_t . The relations given here are valid for an arbitrary point inside the element. Next step is about modifying

these terms from their values at the tying points.

Covariant transverse shear tying procedure

The procedure detailed here can be found in details in [1] (<https://www.emerald.com/insight/content/doi/10.1108/eb023562/full/html>). We are interested in reworking the terms $\widetilde{\varepsilon}_{st}$ and $\widetilde{\varepsilon}_{rt}$ that are responsible for **transverse shear locking**. The idea is to compute the term $\widetilde{\varepsilon}_{st}$ at the tying points C and D, and the term $\widetilde{\varepsilon}_{rt}$ at the tying points A and B (see picture below). Such that we have, $\widetilde{\varepsilon}_{rt}^A$, $\widetilde{\varepsilon}_{rt}^B$, $\widetilde{\varepsilon}_{st}^C$ and $\widetilde{\varepsilon}_{st}^D$.



Note that these **tying points are located on the mid-surface**. Eventually, elsewhere in the element, $\widetilde{\varepsilon}_{rt}$ and $\widetilde{\varepsilon}_{st}$ are interpolated from the values at the tying points. Hence, for an arbitrary point located at (r, s) we have:

$$\begin{cases} \widetilde{\varepsilon_{rt}} = \frac{1}{2}(1+s)\widetilde{\varepsilon_{rt}^A} + \frac{1}{2}(1-s)\widetilde{\varepsilon_{rt}^B} \\ \widetilde{\varepsilon_{st}} = \frac{1}{2}(1+r)\widetilde{\varepsilon_{st}^C} + \frac{1}{2}(1-r)\widetilde{\varepsilon_{st}^D} \end{cases}$$

Covariant membrane strain tying procedure

The full procedure can be found in [2]

(<https://www.sciencedirect.com/science/article/abs/pii/S0045794916309464>) and [3] (<https://www.sciencedirect.com/science/article/abs/pii/S0045794917307757>). It is more involved than the transverse shear procedure but quite similar. The idea is to rework the terms $\widetilde{\varepsilon_{rr}}$, $\widetilde{\varepsilon_{ss}}$ and $\widetilde{\varepsilon_{rs}}$ **such that membrane locking is suppressed** by using specific tying points inside the element. As for the shear terms, it yields new relations for the terms $\widetilde{\varepsilon_{rr}}$, $\widetilde{\varepsilon_{ss}}$ and $\widetilde{\varepsilon_{rs}}$ that linearly depend on strains at the tying points.

Summary

Once the tying procedure is completed, we can rewrite (9) using the covariant strains at the tying points. We then insert the displacement and rotation vectors interpolated from the nodes values (5). This yield the full expression of the covariant strain components that linearly depend on the degrees of freedom at the nodes.

The last step is then to **project the covariant strain components into the local basis** (8). Expressing the strain tensor in the contravariant basis on one hand and in the local basis on the other hand we have:

$$\underline{\underline{\varepsilon}} = \widetilde{\varepsilon_{ij}} \underline{\underline{g}}^i \otimes \underline{\underline{g}}^j = \varepsilon_{kl} \underline{\underline{e}}_k \otimes \underline{\underline{e}}_l \quad i, j = r, s, t \quad \text{and} \quad k, l = 1, 2, 3$$

Which relates the covariant strain components to the local strain components:

$$\varepsilon_{kl} = \widetilde{\varepsilon_{ij}} (\underline{g}^i \cdot \underline{e}_k) (\underline{g}^j \cdot \underline{e}_l)$$

Using the fact that $\widetilde{\varepsilon_{tt}} = 0$ and that \underline{e}_3 is collinear to \underline{g}_t (hence $\underline{e}_3 \cdot \underline{g}^r = \underline{e}_3 \cdot \underline{g}^s = 0$) we can write the local strains in matrix representation (using engineering strains):

$$[\varepsilon] = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{12} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \end{bmatrix} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 0 \\ 2\varepsilon_{12} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \end{bmatrix} \quad (10)$$

Stress tensor

To relate the stresses to the strains we need to apply Hooke's law
https://en.wikipedia.org/wiki/Hooke%27s_law for linear elastic material.

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{12} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2+2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2+2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2+2\nu \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix}$$

Note that this material law is applied on **strains and stresses expressed in the local basis** (where e_3 is not necessarily perpendicular to the mid-surface). Using assumption (1) on the right hand side, as well as the result we got in (10) on the left hand side, leads to a peculiar relation:

$$\underbrace{\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 0 \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{13} \end{bmatrix}}_{\text{kinematic assumption}} = \frac{1}{E} \underbrace{\begin{bmatrix} \sigma_{11} - \nu\sigma_{22} \\ \sigma_{22} - \nu\sigma_{11} \\ -\nu\sigma_{11} - \nu\sigma_{22} \\ (2+2\nu)\sigma_{12} \\ (2+2\nu)\sigma_{23} \\ (2+2\nu)\sigma_{13} \end{bmatrix}}_{\text{stress assumption}}$$

As for the truss element

https://www.sesamx.io/blog/truss_finite_element/#stress-tensor and beam element (https://www.sesamx.io/blog/beam_finite_element/#stress-tensor), this

relation cannot hold. However using a similar explanation, we can find a way out. When making the kinematic assumption we were interested in the macroscopic behavior of the shell. Whereas the stress assumption relates more to a microscopic behavior. It turns out that even if ε_{33} is not 0 (**microscopic scale**) its effect on the displacement (**macroscopic scale**) is negligible compared to what comes from the other terms. Of course, this explanation becomes questionable as the thickness of the shell increases.

We can therefore simplify this relation and write:

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 0 \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{13} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} \sigma_{11} - \nu\sigma_{22} \\ \sigma_{22} - \nu\sigma_{11} \\ 0 \\ (2+2\nu)\sigma_{12} \\ (2+2\nu)\sigma_{23} \\ (2+2\nu)\sigma_{13} \end{bmatrix}$$

And finally we invert it and omit ε_{33} and σ_{33} to get the stresses from the strains:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0 & 0 & 0 \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{E}{2(1+\nu)} & 0 & 0 \\ 0 & 0 & 0 & \frac{kE}{2(1+\nu)} & 0 \\ 0 & 0 & 0 & 0 & \frac{kE}{2(1+\nu)} \end{bmatrix}}_{[D]} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{13} \end{bmatrix}$$

Where the classical transverse shear correction factor $k = \frac{5}{6}$ was introduced.

MITC4 shell element stiffness matrix

The element stiffness matrix is obtained through the expression of the virtual work (https://www.sesamx.io/blog/deep_finite_element_method_introduction/#expression-of-the-weak-form-in-tensor-components). Denoting the virtual strains as $[\bar{\varepsilon}]$ we have for the virtual work over the element volume:

$$\overline{W} = \iiint_V [\bar{\varepsilon}]^T [\sigma] dV \quad (11)$$

Where $dV = dx dy dz$ refers to the volume element in the local Cartesian coordinate system. However, the formula previously derived for the strains depends on the parametric coordinate system (r, s, t) . We can relate this volume element to the parametric coordinates using the **determinant of the Jacobian matrix** (https://www.sesamx.io/blog/deep_finite_element_method_introduction/#element-integrals-evaluation):

$$dV = |\det([J])| dr ds dt$$

It can be shown that the Jacobian matrix is directly given by the **covariant vectors components expressed in the local basis**:

$$[J] = [[g_r], [g_s], [g_t]]$$

Where $[g_i]$ represents the components of the vector \underline{g}_i in the local basis.

Besides, it can be shown that as long as the $[g_i]$ are components of vectors in an orthonormal basis, we have:

$$\det([J]) = \underline{g}_t \cdot (\underline{g}_r \times \underline{g}_s)$$

Hence after expanding (11) we get the following expression:

$$\overline{W} = \iiint_V [\bar{\varepsilon}]^T [D][\varepsilon] |\underline{g}_t \cdot (\underline{g}_r \times \underline{g}_s)| dr ds dt$$

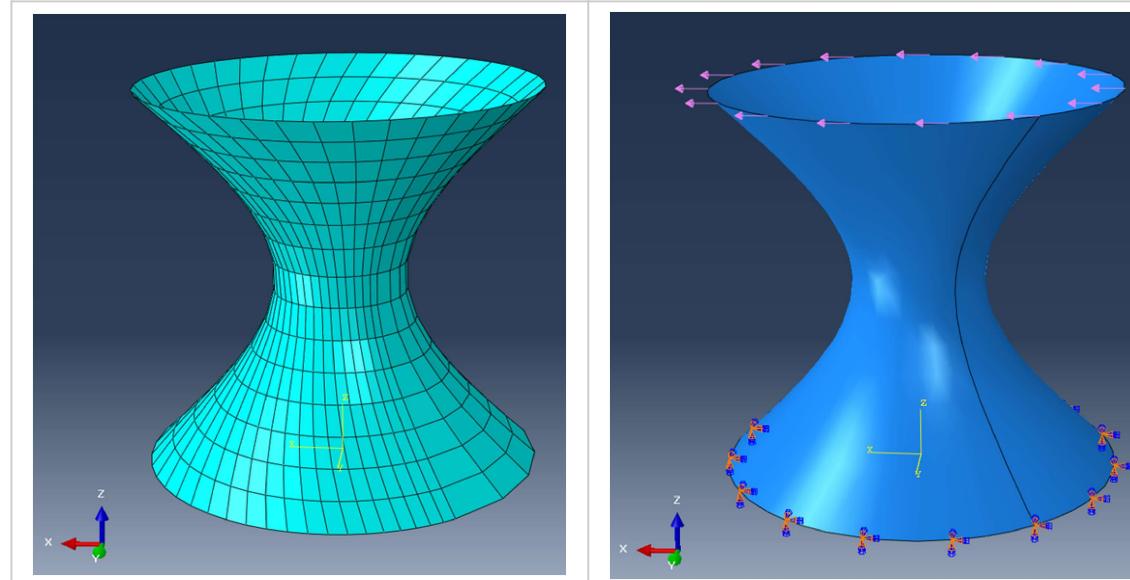
Unlike classical shell or plate element, we cannot in general “pre-integrate” along t this volume integral because the dependence on t is arbitrary. We simply apply a numerical integration over r , s and t . The $2 \times 2 \times 2$ Gauss rule (https://en.wikipedia.org/wiki/Gaussian_quadrature) is enough to integrate exactly along r and s , and has proven to be accurate enough to integrate along t for most applications (even though we cannot ensure an exact integration).

Remark: In the specific case where $h\nu$ is constant over the element, we have seen that g_r and g_s no longer depend on t . This makes it possible to integrate exactly along t without requiring numerical integration.

Eventually, using (10), we can relate each of the terms obtained to the nodes displacements and rotations and recognize the expression of the stiffness matrix.

Example of a simple structure using shell elements

As an example, the following hyperboloid-like model is set up and simulated with the SesamX finite element software (/).



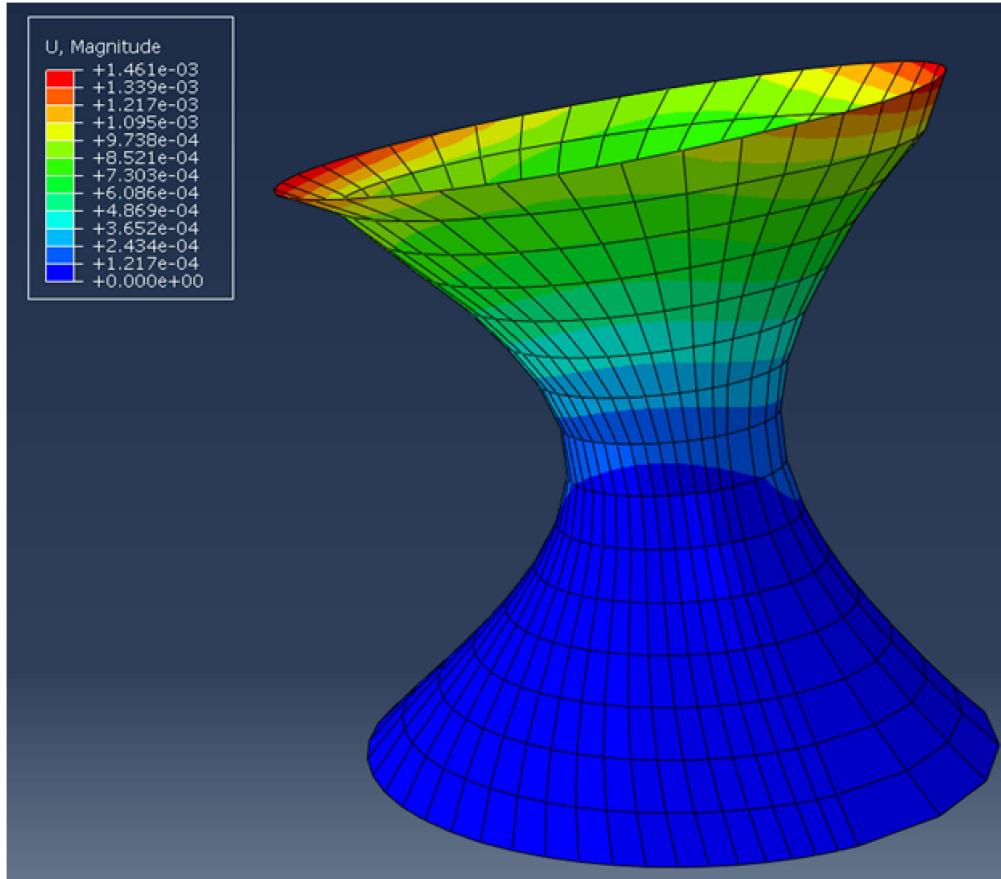
Mesh	Loading and boundary conditions
------	---------------------------------

The mesh is voluntarily distorted and curved (the quads are not planar). This mesh can be qualified as a bad mesh, however it is useful to **showcase the robustness of the MITC4 shell element.**

A linear elastic material is applied with $E = 200GPa$ and $\nu = 0.33$ and a thickness of $80mm$ is applied on all the elements. The circle at $Z = 0$ is clamped while a uniformly distributed load along X is applied at the top circle of the hyperboloid.

This case is then solved with a **linear static resolution**.

And the following figure illustrates the displacement of the model.



To reproduce this simulation on your own, you need:

- at least the free version of the SesamX software ([/pricing/](#)) installed,
- the shell model mesh
([/blog/files/003_Shell_finite_element/400_ABAQUS_QD4_SHELL_HYPERBOLOID_ASSEMBLY.inp](#))
(in Abaqus format),
- and the SesamX input cards
([/blog/files/003_Shell_finite_element/400_SESAMX_QD4_SHELL_HYPERBOLOID_ASSEMBLY.sx](#))
(defining the element properties, the loading and so on).

If you are looking to extend this example, you can get help from the SesamX documentation ([/doc/user_manual/sesamx](#)).

Conclusion

I hope this article has provided you a better understanding of the shell finite element.
If you are looking for more information, do not hesitate to send me an email.

Did you like this content?

Register to our newsletter and get notified of new articles



(//twitter.com/share?
url=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f&text=The%20shell%20finite%20element%20ex



(//plus.google.com/share?
url=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f)



(//www.facebook.com/sharer/sharer.php?
u=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f)



(//reddit.com/submit?
url=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f&title=The%20shell%20finite%20element%20ex



(//www.linkedin.com/shareArticle?
url=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f&title=The%20shell%20finite%20element%20ex



(//www.stumbleupon.com/submit?
url=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f&title=The%20shell%20finite%20element%20ex



(//www.pinterest.com/pin/create/button/?
url=https%3a%2f%2fwww.sesamx.io%2fblog%2fshell_finite_element%2f&description=The%20shell%20finite%20elemen

[← PREVIOUS POST \(HTTPS://WWW.SESAMX.IO/BLOG/BEAMFINITEELEMENT/\)](https://www.sesamx.io/blog/beam_finite_element/)

NEXT POST →
[\(HTTPS://WWW.SESAMX.IO/BLOG/TRIANGULAR_SHELLFINITEELEMENTFORMULATION/\)](https://www.sesamx.io/blog/triangular_shell_finite_element_formulation/)



(mailto:ali.baba@mailbox.org)



(https://www.sesamx.io/index.xml)

Ali Baba (<https://www.sesamx.io>) • © 2021 • SesamX - The engineer friendly finite element software
(<https://www.sesamx.io>)

Hugo v0.55.3 (<http://gohugo.io>) powered • Theme by Beautiful Jekyll (<http://deanattali.com/beautiful-jekyll/>) adapted to Beautiful Hugo (<https://github.com/halogenica/beautifulhugo>)