Proceedings of the
Annual Stability Conference
Structural Stability Research Council
Atlanta, Georgia, April 21-24, 2026

An open-source triangular shell finite element software implementation for the study of thin-walled structures

Cristopher D. Moen[1], Sándor Ádány[2], Amoke Shabhari[3]

Abstract
A triangular shell finite element formulation is implemented in the `Julia` scientific computing language for use with the popular open-source finite element software framework `Ferrite.jl`. Both elastic and geometric stiffness matrices are available from `TriShellFiniteElement.jl`, allowing for the calculation of elastic deformations, stresses, and buckling in thin-walled structures. The shell element formulation utilizes linear shape functions to interpolate for membrane deformations, and considers additional quadratic shape functions to predict bending deformation while avoiding shear locking. The triangular shell element is shown to perform accurately when compared to Abaqus shell element and analytical solutions in a series of thin and thick plate examples which consider elastic deformation and elastic buckling. The elastic deformation of a steel column base plate is modeled. Upcoming work is discussed which includes element stiffness matrix and stress transformations for 3D analysis and the consideration of plasticity.

1. Introduction

Shell finite element analysis is a powerful tool for studying steel structures. There are suprisingly few modern shell finite element implementations available for public consumption however, and it is hard to know what the classical, bread and butter commericial software programs have implemented under the hood. These gaps are motivating a multi-year effort to develop a trusted open-source shell finite element software capability. The vision for this new capability, outlined in Moen(2024) , is to provide a means of exploring, implementing, validating, and applying new shell element formulations in a multi-threaded high performance computing environment.

Our recent efforts have confirmed that it is possible, and not too much work, to write a new shell element formulation as an open-source software library and connect it to existing open-source finite element software framework. In our case we have focused on `Ferrite.jl`, a popular, highly performant finite element package written in the Julia scientific computing language (Bezanson et al. 2012). It is exciting that the open-source shell finite element formulation described in Moen and Ádány (2025) performs as well as the Abaqus S4R element (Simulia 2024) when studying elastic deformations and buckling of thin-walled structures.

[1]President and CEO, RunToSolve LLC, <cris.moen@runtosolve.com>
[2]Associate Research Scientist, Johns Hopkins University, <asandor2@jhu.edu>
[3]Senior Engineer, RunToSolve LLC, <amoke.shabhari@runtosolve.com>

The work on this project continues, and in this paper, advances are presented that focus on the implementation, validation, and application of a Mindlin triangular shell element. The triangular element is especially useful for meshing more complicated geometries, like around holes. The triangular shell element theory is introduced first, followed by a description of its implementation in `Ferrite.jl`. The paper concludes with examples that exercise the shell element with membrane dominated and flexural dominated deformations, as well as elastic buckling, and a steel column base plate. Comparisons are made to Abaqus and theory.

2. Triangular shell element mechanics

A three-noded quadrilateral Mindlin element is explained in detail. The material is assumed to be isotropic and linearly elastic. The involved displacement functions are as follows: $u(x, y)$, $v(x, y)$ and $w(x, y)$ are translations along x, y and z, and $\psi_x(x, y)$ and $\psi_y(x, y)$ are cross-section rotations along x and y directions, respectively (i.e., about y and x axis, respectively), see Figure 1.

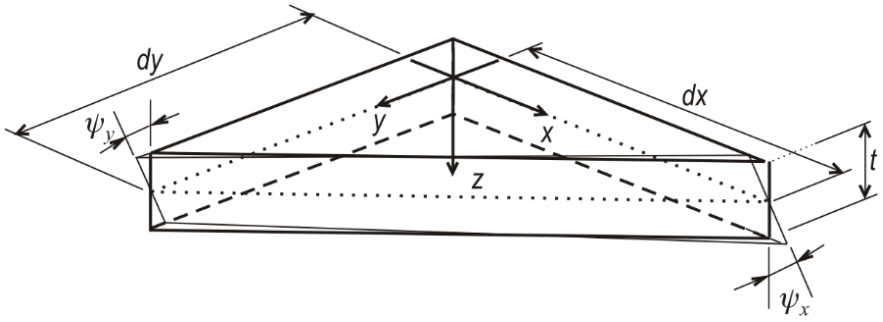

Figure 1: Elementary triangle of a surface

The element mechanics consider the superposition of membrane and bending stress/displacement fields, and w is assumed constant along the thickness, hence: $w(x, y, z) = w(x, y)$. Regarding u and v, they are linearly varying across the thickness, and thus, can be expressed as the sum of membrane strains (at $z = 0$) and bending strains (at $z \neq 0$):

$$\varepsilon_x = \varepsilon_{x,m} + \frac{d\psi_x}{dx} z \tag{1}$$

$$\varepsilon_y = \varepsilon_{y,m} + \frac{d\psi_y}{dy} z \tag{2}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + \left(\frac{\partial \psi_x}{\partial x} + \frac{\partial \psi_y}{\partial y} \right) z \tag{3}$$

$$\gamma_{xz} = \psi_x + \frac{dw}{dx} \tag{4}$$

$$\gamma_{yz} = \psi_y - \frac{dw}{dy} \tag{5}$$

where ψ_x and ψ_y are cross-section rotations along x and y directions, respectively (i.e., rotations about y and x axis, respectively). It is noted that $x-y-z$ is a global cartesian coordinate system, and $x-y$ is aligned to the plane of the element. The membrane displacement functions are $u(x,y)$ and $v(x,y)$ interpreted at $z=0$. The linear membrane strains are obtained by the derivatives of $u(x,y)$ and $v(x,y)$:

The stresses are:

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix} = \begin{bmatrix}\frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0\\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0\\ 0 & 0 & G\end{bmatrix}\begin{bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\end{bmatrix} \tag{6}$$

$$\begin{bmatrix}\tau_{xz}\\ \tau_{yz}\end{bmatrix} = \begin{bmatrix}\frac{5}{6}G & 0\\ 0 & \frac{5}{6}G\end{bmatrix}\begin{bmatrix}\gamma_{xz}\\ \gamma_{yz}\end{bmatrix} \tag{7}$$

where E and G are the Young's and shear modulus, respectively, and ν is the Poisson's ratio. The 5/6 factor is to take into consideration the difference between the model and real through-thickness shear strain/stress distributions (constant in the model, but approx. quadratic in reality).

Nonlinear strains are calculated from membrane strains only, approximated by the Green-Lagrange tensor. Nonlinear strains are calculated at $z=0$ only, i.e., assuming constant nonlinear strains across the thickness. From $u(x,y)$, $v(x,y)$, and $w(x,y)$, they are expressed as follows:

$${\varepsilon_{x,m}}^{NL} = \frac{1}{2}\left[\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2 + \left(\frac{dw}{dx}\right)^2\right] \tag{8}$$

$${\varepsilon_{y,m}}^{NL} = \frac{1}{2}\left[\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2 + \left(\frac{dw}{dy}\right)^2\right] \tag{9}$$

$${\gamma_{xy,m}}^{NL} = \left[\frac{du}{dx}\frac{du}{dy} + \frac{dv}{dx}\frac{dv}{dy} + \frac{dw}{dx}\frac{dw}{dy}\right] \tag{10}$$

3. Finite element formulation

3.1. Elastic stiffness matrix

The membrane u and v displacements are interpolated using linear shape functions, expressed as follows.

$$N_1 = 1-\xi-\eta \tag{11}$$

$$N_2 = \xi \tag{12}$$

$$N_3 = \eta \tag{13}$$

Each node, therefore, has a u and v translational DOF, leading to 6 membrane DOF for the element. Following the usual steps of finite element derivations, the $k_{\mathrm{e,m}}$ membrane partition of the elastic stiffness matrix for the element can be obtained; it is 6×6.

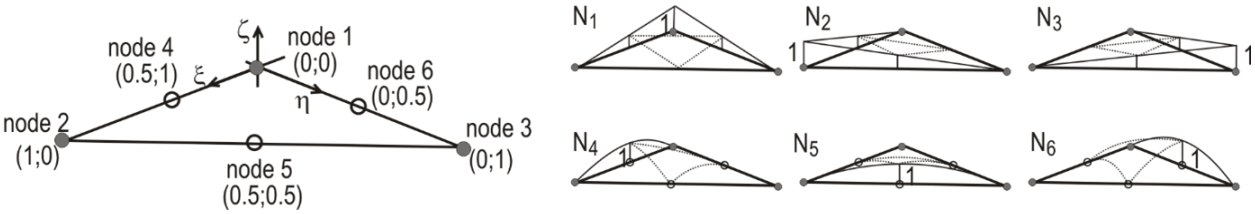

Figure 2: Local coordinates, shape functions

The bending behavior is described by three functions: $w(x, y)$, $\psi_x(x, y)$ and $\psi_y(x, y)$. Theoretically, it would be possible to perform the displacement approximation using the same 3 shape functions as in Eq. 11 to Eq. 13, however, this interpolation scheme is subjected to poor convergence and shear locking. A straightforward solution is proposed by Tessler and Hughes(1985) and Liu(2002); the essential idea is to use higher-degree interpolation for $w(x, y)$, in which case the linear interpolation for $\psi_x(x, y)$ and $\psi_y(x, y)$ can be kept. This approach is followed here.

Accordingly, $\psi_x(x, y)$ and $\psi_y(x, y)$ are approximated by 3 shape functions, see Equation 11 to Equation 13, which requires two nodal DOF per node. At the same time, $w(x, y)$ is interpolated using 6 shape functions. The first 3 shape functions are identical to those used above, while the other 3 shape functions are as follows:

$$N_4 = 4\xi(1 - \xi - \eta) \tag{14}$$

$$N_5 = 4\xi\eta \tag{15}$$

$$N_6 = 4\eta(1 - \xi - \eta) \tag{16}$$

The shape functions for $w(x, y)$ are illustrated in Figure 2. The introduction of the 3 additional shape functions can readily be interpreted as introducing 3 more nodes with an out-of-plane translational DOF at each node, resulting in 6 DOF for approximating $w(x, y)$.

Following the usual steps of finite element derivations, the $k_{\mathrm{e,b}}$ bending (i.e., plate) partition of the elastic stiffness matrix for the element can be expressed; it is 12×12. However, it is impractical to have nodes with different degrees of freedom, therefore, it is convenient to eliminate the extra 3 DOFs at the mid-nodes (i.e., nodes 4, 5 and 6). Unlike in Tessler and Hughes(1985), here, static condensation is employed to eliminate the extra DOF, which finally leads to 3 bending DOF at each corner node, resulting in 9×9 size $k_{\mathrm{e,b}}$ and $k_{\mathrm{g,b}}$ partitions.

For the transformation between the local and global coordinate systems, the first 3 shape functions are employed. Accordingly, e.g., the global x coordinates are expressed as:

$$x = \sum_{i=1}^{3} x_i N_i(\xi, \eta) \tag{17}$$

where x_i are the nodal coordinates of the nodes.

It is noted that, due to the simplicity of the applied shape functions, both the 6×6 $k_{\mathrm{e,m}}$ and the 12×12 $k_{\mathrm{e,b}}$ can be obtained by analytical integration, even with the coordinate transformation included. Regarding the static condensation (i.e., reducing the 12×12 bending partition to a 9×9 matrix), our experience is that using today's symbolic mathematical tools, the static condensation is possible to complete analytically, but the resulting expressions (i.e., elements of the stiffness matrix) are extremely long. Therefore, it is computationally more efficient to perform the static condensation numerically.

Finally, the elastic stiffness matrix for the element is composed of the 6×6 $k_{\mathrm{e,m}}$ and the 9×9 $k_{\mathrm{e,b}}$, resulting in a 15×15 k_{e}. That is, there are 5 DOF at each node. It is noted that, optionally, we can add the drilling DOF to the nodes, which becomes necessary whenever the structure to be analyzed is composed of plates laying in different planes.

3.2. Geometric stiffness matrix

As typical in shell finite elements, the geometric stiffness matrix is derived using the mid-surface membrane stresses (or forces). In accordance with the stress and strain definitions, mid-surface (i.e., at $z = 0$) stresses are induced by u and w translations. Since linear shape functions are employed for the u and v, the strains and stresses in a finite element are constant. While this can be regarded as a negative feature which makes it necessary to have a fine mesh to accurately approximate the stresses, it has the benefit that the whole membrane stress field of the element can be described by three scalars: σ_x, σ_y and τ_{xy}.

Moreover, in the derivation of the geometric stiffness matrix, the $u(x, y)$, $v(x, y)$, and $w(x, y)$, translational displacements are needed (at the mid-surface) to have the nonlinear (second-order) strains, as given by the Green-Lagrange formulae, see Eq. 8 to Eq. 10. For the sake of convenience, all the 3 translations are approximated by 3 shape functions, i.e., N_1, N_2 and N_3, as defined above.

Due to the simplicity of the shape functions and due to the constant membrane stresses, the geometric stiffness matrix can be expressed analytically.

3.3. Integration

To calculate the stiffness matrix, the final step is an integration. To obtain a specific element of the stiffness matrix, somewhat simplified and symbolically, the following operation is to be completed:

$$k = \iint f(x, y)\, dx\, dy \tag{18}$$

(Note, theoretically there is integration along the thickness, too, but it is always straightforward to complete if the material is elastic, that is why the real challenge is the integral over the surface.) If the element is flat and rectangular, the above integral can be performed analytically, and the stiffness matrix entries can be expressed in closed format. However, in more general cases the integration cannot be completed analytically, hence, numerical integration must be applied.

$$k = \sum_{i=1}^{n_G} W_i f(x_i, y_i) \tag{19}$$

where W_i are the weights, and n_G is the number of integration points. It is common to use the Gauss quadrature, this is what is applied in our implementation, too. Since the element is triangular, one integration point is applied for all stiffness matrices except for $k_{\mathrm{e,b}}$ which we found requires a minimum of 3 integration points to match the analytical element solution because of the nonlinear N_4, N_5, and N_6 shape functions employed. As we know, for triangular elements, the accuracy of Gaussian quadrature depends on the polynomial order of the integrand. The fundamental one-point rule is accurate only for linear shape functions, whereas quadratic terms arising from nonlinear shape functions require higher-order integration. Three-point Gaussian quadrature rules represent the lowest possible order that accurately integrate the quadratic polynomials over a triangular domain (Dunavant 1985). For the triangular shell element with the vertices (0,0), (0,1), (1,0), the Guassian quadrature points and weights for the 3-point rule are as shown in Figure 3.

```
DofHandler{2, Grid{2, Triangle, Float64}} …
QuadratureRule{RefTriangle, Vector{Float64}, Vector{Vec{2, Float64}}}
  weights: Array{Float64}((3,)) [0.166666666666665, 0.166666666666665, 0.166666666666665]
  points: Array{Vec{2, Float64}}((3,))
    1: Vec{2, Float64}
      data: Tuple{Float64, Float64}
        1: Float64 0.16666666666667
        2: Float64 0.16666666666667
    2: Vec{2, Float64}
      data: Tuple{Float64, Float64}
        1: Float64 0.16666666666667
        2: Float64 0.66666666666667
    3: Vec{2, Float64}
      data: Tuple{Float64, Float64}
        1: Float64 0.66666666666667
        2: Float64 0.16666666666667
```

Figure 3: Weights and location coordinates of the 3-point Gaussian quadrature for triangular elements

4. Triangular shell element software implementation

The `Julia` implementation of this triangular shell element follows the element mechanics discussed herein. The code and accompanying documentation are available at `TriShellFiniteElement.jl` which includes examples of how to use the shell element with `Ferrite.jl`. (A quadrilateral shell element described in Moen and Ádány (2025) is also available, see `QuadShellFiniteElement.jl`.)

5. Examples

All examples presented here are for a 100 mm x 1000 mm plate with E = 200000 MPa and ν = 0.30. The plate boundary conditions are shown in Figure 4. The mesh density is varied as follows:

([2, 2], [2, 8], [2, 32], [10, 92], [30, 322], [98, 980], [312, 3120]) where [number of quadratic divisions in the 100 mm direction, number of quadratic divisions in the 1000 mm direction]. Each quadratic division is further subdivided into two triangular elements. The total number of degrees of freedom in each model are then [(2+1) x (2+1) x 5 = 45, 135, 495, 5115, 50065, 485595, 4884365]. Boundary conditions are shown in the figure below. Results from models using the quadrilateral shell element in Moen and Ádány (2025) are also included for comparison.

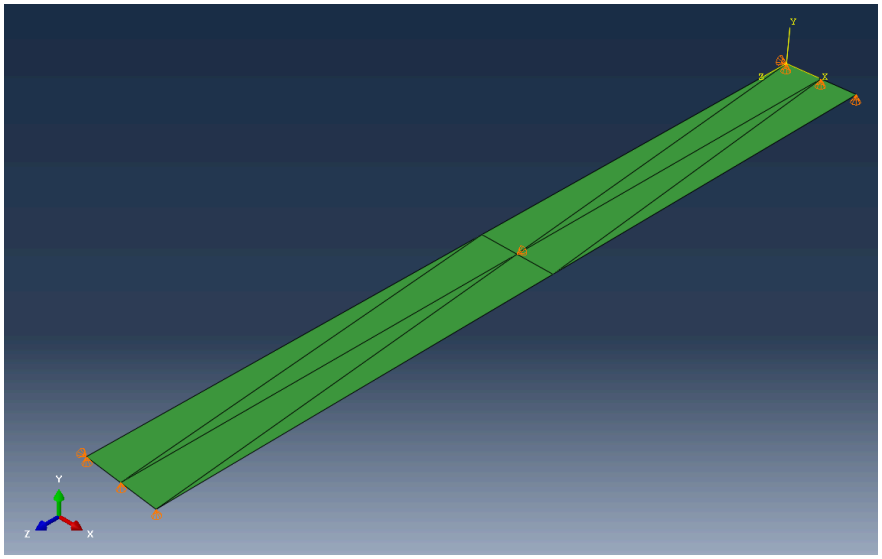

Figure 4: Plate boundary conditions: fixed at ends in Y, middle in Z, two corner nodes in X

5.1. Out-of-plane bending with midspan distributed load

This example is essentially a beam, simply supported at the two ends, subjected to a concentrated force at the middle. However, the problem is solved as a plate problem, the load acting perpendicularly to the plane of the shell elements. Therefore, the 'concentrated' force is applied as a line load, as shown, the intensity being 1000 N/mm and 1 N/mm for the thick and thin plate, respectively. Due to the loading and supports, only the plate bending degrees of freedom are activated. The maximum (Y-directional) displacements at the middle cross-section are summarized in Table 1.

Since the problem solved is as a plate problem, the displacements along the middle transverse line are not perfectly constant, that is why both the maximum displacement (which occurs at the edge) and the average displacement (i.e., average of the nodal displacements along the line of the middle cross-section) are presented. Due to the simplicity of the problem, the deflection of the plate can be solved analytically, and in this case the solution is 1.289 mm and 156.252 mm for the thick and thin plate, respectively. (Note, the analytical solution without considering the shear deformations, i.e., using classic Euler-Bernoulli beam theory are 1.25 and 156.25 mm, respectively; thus, the effect of shear deformations is negligible if the plate is thin, while observable - though small - if the plate is thick.)

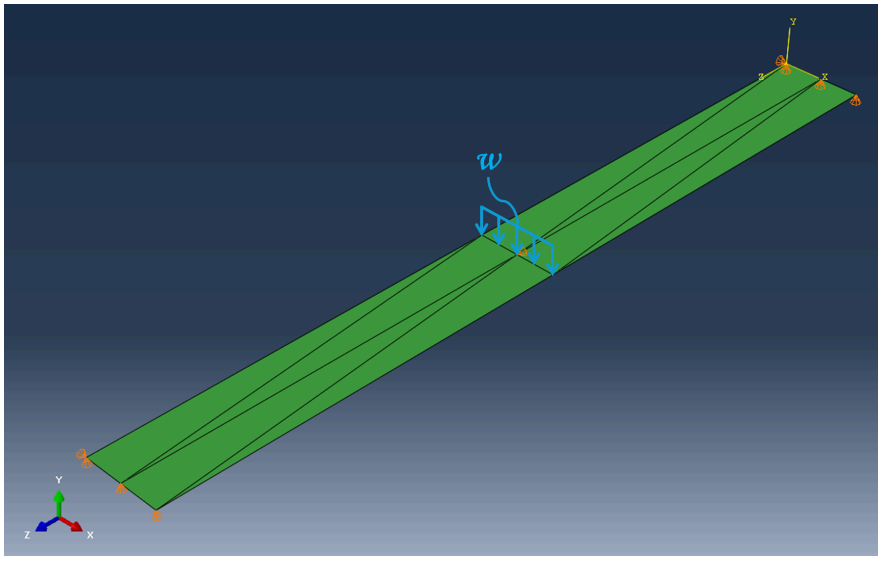

Figure 5: Out-of-plane midspan distributed load w

Table 1: Maximum out-of-plane displacements for out-of-plane bending with midspan distributed load

	Num. of DOF	No. of nodes	ABAQUS (S4R)		ABAQUS (S3R)		TriShellFiniteElement.jl	
			(Average)	(max)	(Average)	(max)	Average	Max
t = 2 mm	45	9	116.936	117.162	111.139	111.262	110.609	110.695
	135	27	153.685	153.904	153.308	153.507	152.246	152.423
	495	99	155.994	156.214	155.962	156.176	154.372	154.547
	5115	1023	156.026	156.414	156.033	156.422	156.071	156.459
	50065	10013	156.018	156.435	156.016	156.433	156.076	156.491
	485595	97119	156.050	156.478	156.048	156.476	156.064	156.497
	4884365	976873	156.043	156.473	156.041	156.472	156.065	156.499
t = 100 mm	45	9	0.976	0.978	0.975	0.976	0.999	1.000
	135	27	1.270	1.271	1.270	1.271	1.296	1.298
	495	99	1.288	1.290	1.288	1.289	1.350	1.352
	5115	1023	1.289	1.292	1.289	1.292	1.335	1.338
	50065	10013	1.288	1.292	1.288	1.292	1.340	1.343
	485595	97119	1.288	1.292	1.288	1.292	1.337	1.341
	4884365	976873	1.288	1.292	1.288	1.292	1.337	1.341

The triangular element performs well here, although the triangular element solution is slightly more flexible than the analytical solution.

5.2. Out-of-plane bending with uniform pressure

This example is similar to the previous one, the only difference being the load, which is now a uniformly distributed load over the whole surface. (The beam equivalent of the problem would be a simply supported beam with a uniformly distributed line load over its full length.) The load intensities are 1 N/mm^2 and 0.001 N/mm^2 for the thick and thin case, respectively.

The results are summarized in Table 2. The analytical solutions are 0.80075 and 97.657 mm for the thick and thin case, respectively. The triangular shell finite element solution `TriShellFiniteElement.jl` is again slightly more flexible than the analytical solution

and the solution predicted using Abaqus S4R and S3R elements. Also, the (lang:“julia”, “TriShellFiniteElement.jl”) requires a finer mesh before converging on the solution.

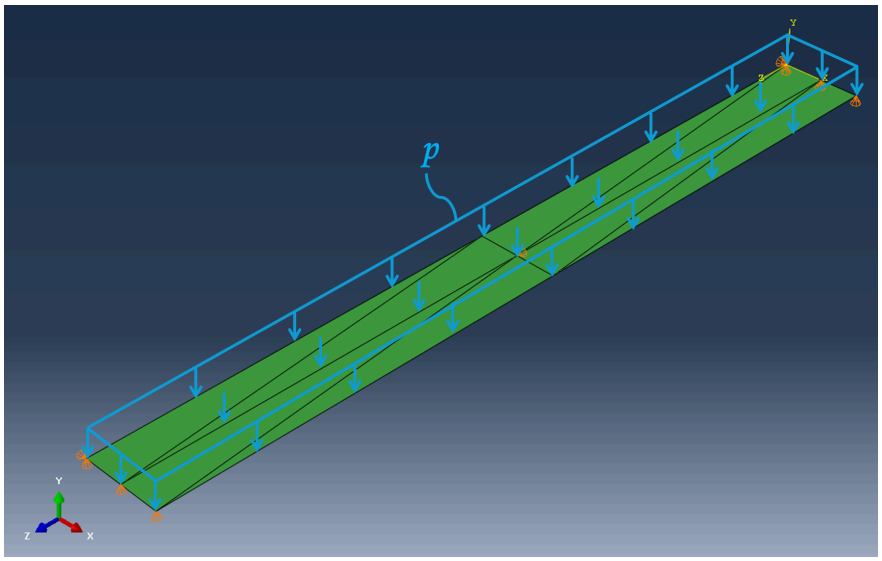

Figure 6: Out-of-plane uniform pressure p

Table 2: Maximum out-of-plane displacements for out-of-plane bending with uniform pressure

	Num. of DOF	No. of nodes	ABAQUS (S4R)		ABAQUS (S3R)		TriShellFiniteElement.jl	
			(Average)	(max)	(Average)	(max)	Average	Max
t = 2 mm	45	9	58.468	58.581	55.569	55.631	36.884	36.913
	135	27	95.156	95.272	94.954	95.067	83.900	83.993
	495	99	97.446	97.563	97.433	97.549	93.658	93.758
	5115	1023	97.518	97.726	97.522	97.730	96.524	96.730
	50065	10013	97.980	98.205	97.979	98.204	97.268	97.490
	485595	97119	93.663	93.883	93.662	93.882	97.460	97.691
	4884365	976873	95.238	95.463	95.238	95.463	97.527	97.759
t = 100 mm	45	9	0.488	0.490	0.488	0.488	0.500	0.500
	135	27	0.781	0.782	0.781	0.782	0.795	0.796
	495	99	0.800	0.801	0.799	0.800	0.831	0.832
	5115	1023	0.800	0.802	0.800	0.802	0.824	0.826
	50065	10013	0.800	0.802	0.800	0.802	0.826	0.828
	485595	97119	0.800	0.802	0.800	0.802	0.825	0.827
	4884365	976873	0.800	0.802	0.800	0.802	0.825	0.827

5.3. In-plane bending with midspan distributed load

This example is a beam, similar to the one presented in Section 5.1, however, now the load is acting in the plane of the shell elements. Accordingly, now the membrane degrees of freedoms (and only those) are activated. The load intensities are 1000 N/mm for both thickness values. The analytical solutions are 1.289 mm and 64.45 mm for the thick and thin cases, respectively.

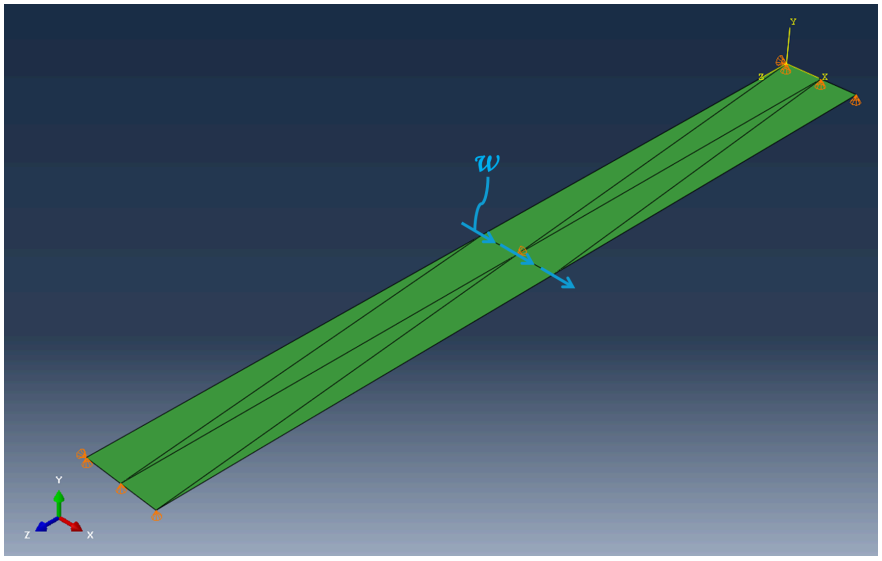

Figure 7: In-plane midspan distributed load w

As the discretization is increased, all the variants seem to tend to a displacement value slightly above the analytical solution.

Table 3: Maximum in-plane displacements for in-plane bending with midspan distributed load

	Num. of DOF	No. of nodes	ABAQUS (S4R)		ABAQUS (S3R)		TriShellFiniteElement.jl	
			(Average)	(max)	(Average)	(max)	Average	Max
t = 2 mm	45	9	3.194	3.197	3.193	3.196	3.194	3.197
	135	27	19.973	20.022	19.956	20.005	19.973	20.022
	495	99	38.807	38.914	38.788	38.895	38.807	38.914
	5115	1023	62.584	62.682	62.537	62.635	62.584	62.682
	50065	10013	64.925	65.012	64.901	64.989	64.924	65.012
	485595	97119	65.442	65.526	65.423	65.507	65.444	65.526
	4884365	976873	65.774	65.856	65.755	65.840	65.775	65.856
t = 100 mm	45	9	0.998	1.204	0.061	0.061	0.064	0.064
	135	27	1.543	1.563	0.382	0.383	0.399	0.400
	495	99	1.701	1.706	0.771	0.774	0.776	0.778
	5115	1023	1.319	1.322	1.250	1.252	1.252	1.254
	50065	10013	1.316	1.318	1.298	1.300	1.298	1.300
	485595	97119	1.321	1.323	1.308	1.310	1.309	1.311
	4884365	976873	1.327	1.329	1.315	1.317	1.316	1.317

5.4. In-plane bending with uniform pressure

This example is similar to the one in Section 5.2, however, now the load is acting in the plane of the shell elements. Accordingly, now the membrane degrees of freedoms (and only those) are activated. The load intensities are 1 N/mm^2 for both thickness values. The analytical solutions are 0.80075 mm and 40.0375 mm for the thick and thin cases, respectively. The various FEM solutions are summarized in Table 4. The observations are very similar to those listed in Section 5.3.

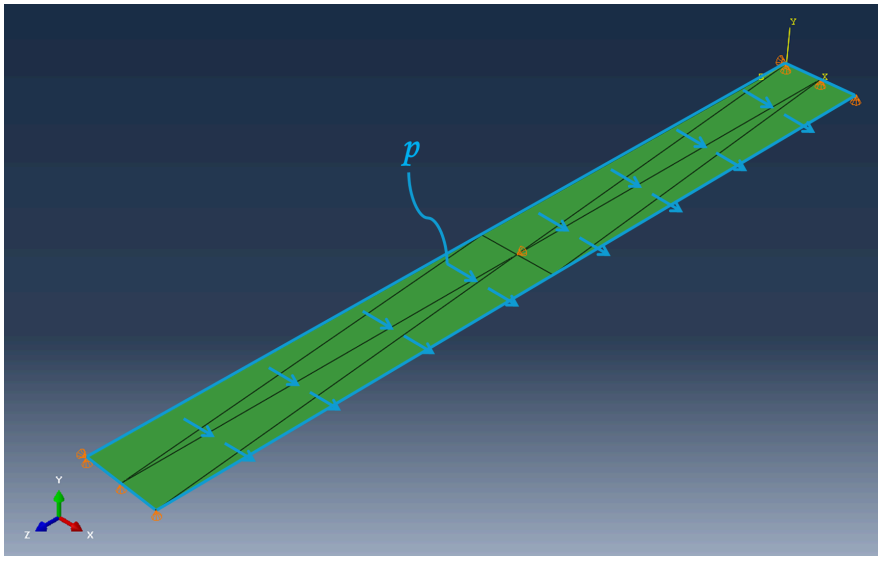

Figure 8: In-plane uniform pressure p

Table 4: Maximum in-plane displacements for in-plane bending with uniform pressure

	Num. of DOF	No. of nodes	ABAQUS (S4R)		ABAQUS (S3R)		TriShellFiniteElement.jl	
			(Average)	(max)	(Average)	(max)	Average	Max
t = 2 mm	45	9	45.528	59.294	1.602	1.604	1.611	1.612
	135	27	55.581	58.876	12.181	12.208	12.197	12.224
	495	99	54.662	55.276	24.092	24.148	24.106	24.161
	5115	1023	41.581	41.638	39.030	39.084	39.064	39.117
	50065	10013	41.440	41.489	40.624	40.673	40.642	40.691
	485595	97119	41.692	41.740	41.063	41.111	41.080	41.126
	4884365	976873	41.987	42.034	41.377	41.424	41.389	41.449
t = 100 mm	45	9	0.583	0.770	0.031	0.031	0.032	0.032
	135	27	0.957	0.973	0.233	0.234	0.244	0.244
	495	99	1.065	1.068	0.479	0.480	0.482	0.483
	5115	1023	0.828	0.829	0.780	0.781	0.781	0.782
	50065	10013	0.828	0.829	0.812	0.813	0.813	0.814
	485595	97119	0.834	0.835	0.821	0.822	0.822	0.823
	4884365	976873	0.840	0.841	0.828	0.828	0.828	0.829

5.5. Column buckling

All the previous examples involved linear static analysis. In the example presented here linear buckling analysis is performed. The load is a unit compressive stress, acting in the plane of the shell elements, at the short edges of the member, defined as an edge load. In both cases the first mode is a flexural buckling, with one half-wave longitudinally, i.e., the situation is essentially identical to the classic Euler column buckling problem. The first (i.e. lowest) critical stress calculated analytically is 1603.78 and 0.65797 N/mm^2 for the thick and thin member, respectively. Note, in these values the effect of shear deformation is included, that is why the values are somewhat lower compared to the classic Euler-formula prediction.

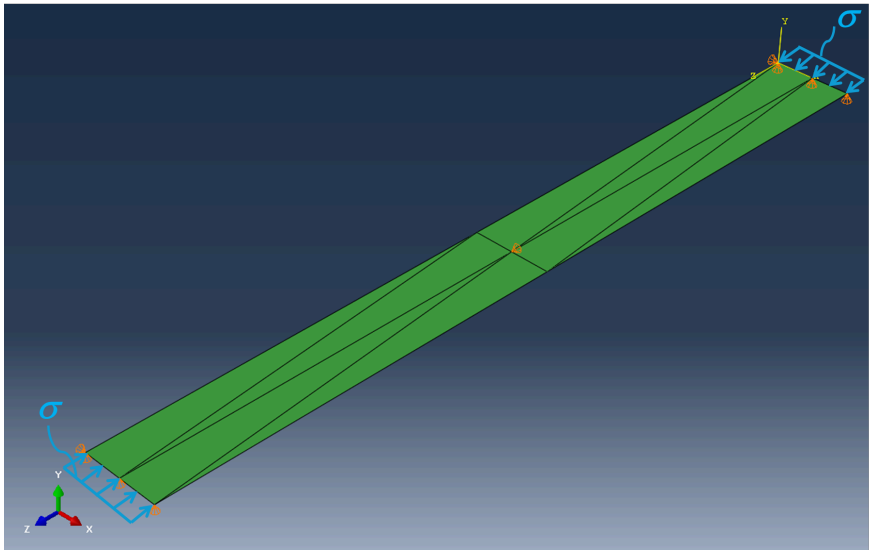

Figure 9: Uniform stress σ at plate ends

Table 5: Critical buckling stresses for column buckling

	Num. of DOF	No. of nodes	ABAQUS (S4R)		TriShellFiniteElement.jl		ABAQUS (S3R)	
			(Mode 1)	(Mode 2)	(Mode 1)	(Mode 2)	(Mode 1)	(Mode 2)
t = 2 mm	45	9	1.070	167.000			1.125	458.080
	135	27	0.676	2.940			0.677	2.969
	495	99	0.660	2.658			0.660	2.660
	5115	1023	0.659	2.645			0.659	2.645
	50065	10013	0.659	2.643			0.659	2.643
	485595	97119	0.659	2.643			0.659	2.643
	4884365	976873					0.659	2.643
t = 100 mm	45	9	2562.600	3356.500			2564.400	55437.000
	135	27	1373.300	1645.200			1644.800	5699.200
	495	99	1232.100	1606.400			1606.800	2733.200
	5115	1023	1603.600	1604.300			1604.200	1679.400
	50065	10013	1604.000	1616.400			1604.000	1624.100
	485595	97119	1604.000	1617.900			1604.000	1618.600
	4884365	976873					1604.000	1618.100

5.6. Analysis of column base plate with holes

A tensile load P is applied on a steel column base plate. The plate is fixed against all translations and rotations at the hole locations. To simulate the structural response, the base plate is modeled with fixed boundary conditions at the bolt hole locations, constraining all translational and rotational degrees of freedom. The discretization and subsequent analysis were conducted using the Julia-based finite element workflow. The geometry was discretized into a mesh of triangular shell elements (Figure 10) with `Gmsh.jl` for mesh generation. The problem was solved for out-of-plane deformations using the triangular shell element formulation in `TriShellFiniteElement.jl`, see Figure 11.

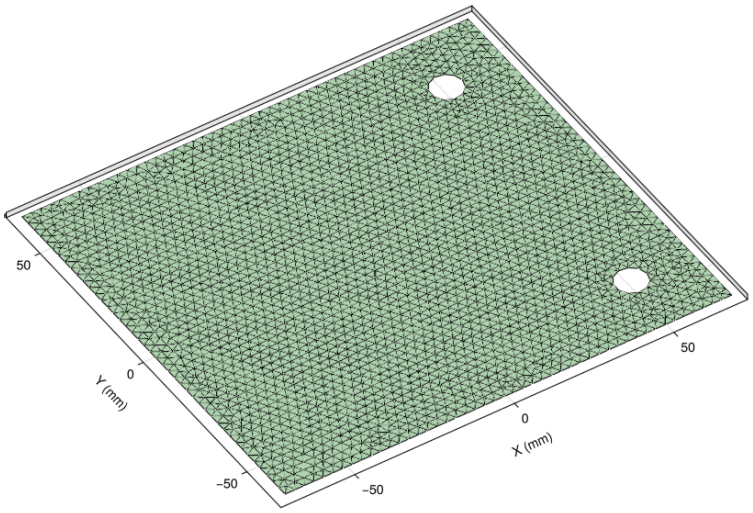

Figure 10: Undeformed base plate, meshed with triangular shell elements

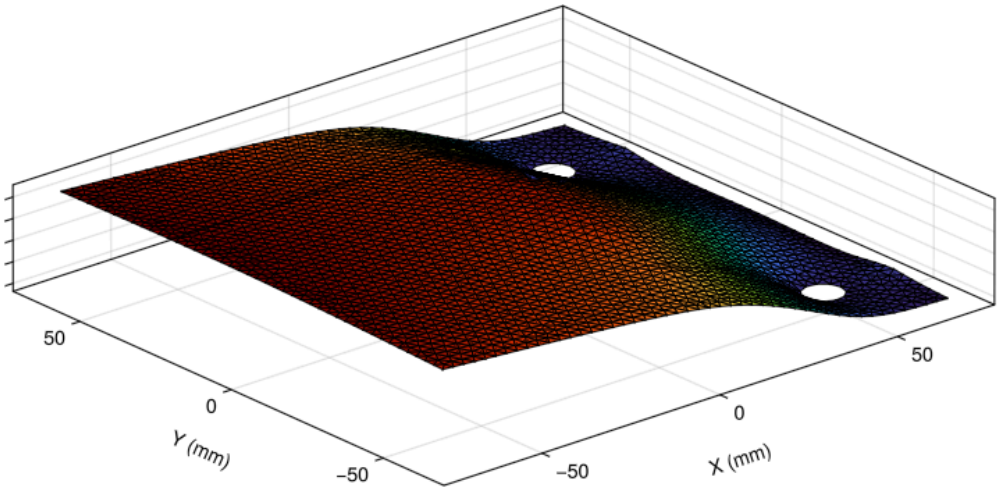

Figure 11: Out-of-plane deformation in the base plate

6. Discussion

It is exciting to now have two shell finite element formulations `TriShellFiniteElement.jl` and `QuadShellFiniteElement.jl` with some validation and documentation supporting them. There is still work to be done though. We need to showcase how much faster these open-source packages on 'big' models compared to commericial finite element programs. General local-to-global transformations for element stiffness, nodal deformations, and element stresses should be developed and validated to move us from 2D to 3D analysis. Nonlinear solvers can be plugged in soon to consider geometric nonlinearity of thin-walled structures. And looming is the subject of plasticity, starting with isotropic hardening, and how to implement that well for these elements. Overall the

`Julia` open-source finite element software community is excited about this work, see a recent presentation, and we are finding more and more collaborators to move things along.

7. Conclusion

A triangular shell finite element formulation is implemented as an open-source software package and used with a popular general open-source finite element software framework. The element formulation, which used linear shape functions to interpolate membrane deformations and quadratic shape functions to interpolate bending deformations, performed well when compared to analytical and Abaqus shell finite element solutions that evaluated membrane, bending, and buckling behavior of thin and thick plates. Upcoming work includes local element stiffness and stress transformations to accommodate 3D analysis of thin-walled structures.

References

Bezanson, J., Karpinski, S., Shah, V. B., and Edelman, A. (2012). "Julia: A fast dynamic language for technical computing". *arXiv Preprint arXiv:1209.5145*.

Dassault Systèmes. (2024). *Abaqus (Version 2024)*. https://www.3ds.com/products/simulia/abaqus

Dunavant, D. A. (1985). "High Degree Efficient Symmetrical Gaussian Quadrature Rules for the Triangle". *International Journal for Numerical Methods in Engineering*, *21*, 1129–1148.

Liu, Y. J., and Riggs, H. R. (2002). *"Development of the MIN-N family of triangular anisoparametric Mindlin plate elements"*. University of Hawaii at Manoa.

Moen, C. D. (2024). "A fast, scalable shell finite element formulation implemented with open-source software". *Annual Stability Conference, Structural Stability Research Council,* San Antonio, TX, March 19–22, 2024.

Moen, C. D., and Ádány, S. (2025). "Thin shell finite element formulations implemented in open-source software". *Annual Stability Conference, Structural Stability Research Council,* Louisville, KY, April 01–04, 2025.

Tessler, A., and Hughes, T. J. R. (1985). "A three-node Mindlin plate element with improved transverse shear". *Computer Methods in Applied Mechanics and Engineering*, *50*(1), 71–101.