First published in: 10th International Conference Computational Structures Technology, Proceedings., Valencia, September 2010

Karlsruhe Institute of Technology

Institute of Mechanics

Software Supported Implementation of Efficient Solid-Shell Finite Elements

September 2010

Steffen Mattern, Karl Schweizerhof

Institute of Mechanics
Kaiserstr. 12
D-76131 Karlsruhe
Tel.: +49 (0) 721/ 608-2071
Fax: +49 (0) 721/608-7990
E-Mail: info@ifm.kit.edu
www.ifm.uni-karlsruhe.de

Software Supported Implementation of Efficient Solid-Shell Finite Elements

Steffen Mattern Karl Schweizerhof

Abstract

The operations necessary for the computation of the internal nodal force vector are in general the most time-consuming parts of an explicit FE-analysis. The paper presents an implementation concept for element routines for volumetric shell – the so-called Solid-Shell – elements based on the application of the symbolic programming tool ACEGEN, a plug-in for the computer algebra software MATHEMATICA. This symbolic implementation means that vector and matrix operations and differentiations do not have to be computed in advance in order to realize a conversion into a programming language. Consequently, programming errors can be avoided almost completely and less time is required for the implementation. Program code in FORTRAN is generated and simultaneously optimized automatically, which leads to very efficient routines compared to manually implemented code.

1 Introduction

In order to realize a continuum-like modeling of a shell structure and being able to to capture 3D effects as in laminated shells, the so-called Solid-Shell element class, presented e.g. in [18, 11], with linear interpolation of geometry and displacements in thickness as well as in shell surface direction is a suitable alternative to purely 3D analysis. As the formulation allows independent interpolation for the in-plane and the out-of-plane direction, a separate higher order interpolation only in the shell-surface plane as often needed for arbitrary curved shell geometries is possible.

The most widely-used explicit time integration method is the central difference scheme, often also called VERLET algorithm [24]. For lumped mass matrices the costly solution of coupled linear equations on global level is not necessary; further the usage of diagonalized mass matrices solely requires vector operations, which leads to low computational cost per time step. Unfortunately, due to the so-called COURANT criterion [8], the time step size is limited to a critical value, which makes the central difference method mostly attractive for 'highly dynamic' problems like impact, strong nonlinearities and short duration transient analyses, where small time steps are required anyway.

In this contribution the focus is on an implementation concept for Solid-Shell elements using linear/ quadratic interpolation of geometry and displacements in in-plane direction together with a linear interpolation in thickness direction. Besides the pure displacement formulation with standard (full) numerical integration, which – as is well-known – leads to an overly stiff behavior, the implementation of different approaches in order to reduce these so-called locking phenomena are discussed. Regarding geometric locking effects, the method of '*Assumed Natural Strains*' (ANS) [2, 6, 4, 5], where the strains are evaluated at specific sampling points and

interpolated with the desired order is applied. POISSON locking, which affects simulations near the incompressible limit is cured with different versions (i. e. different numbers of parameters) of the '*Enhanced Assumed Strain*' (EAS) method [23, 22].

The implementation concept is based on the application of the symbolic programming tool ACEGEN, q.v. [15, 13], which allows a combination of symbolic operations together with the automatic generation of highly efficient program code. Both ANS and EAS schemes are also implemented applying symbolic programming. The generated subroutines are implemented into the in-house finite element code FEAP-MeKa [21]. The specifics of the implementation concept are discussed and the efficiency and functionality of the element formulations are presented on numerical examples.

2 Explicit Time Integration

For numerical time integration, the well-known central difference method is used and implemented as proposed e. g. in [3] or originally for molecular dynamics in [24]. Here the governing equations are shown briefly.

For the current time step n, the accelerations are computed as

$$\ddot{\mathbf{d}}^n = \mathbf{M}^{-1} \left(\mathbf{f}^n - \mathbf{C}\, \dot{\mathbf{d}}^{n-1/2} \right), \tag{1}$$

with the diagonalized system mass matrix $\mathbf{M}$, the system load vector $\mathbf{f}^n$ at time n, the system damping matrix $\mathbf{C}$ and the velocities $\dot{\mathbf{d}}^{n-1/2}$ at time step $n - 1/2$. The velocity between two time steps is updated by

$$\dot{\mathbf{d}}^{n+1/2} = \dot{\mathbf{d}}^{n-1/2} + \overline{\Delta t^n}\, \ddot{\mathbf{d}}^n \tag{2}$$

with $\overline{\Delta t^n} = \frac{(\Delta t^n + \Delta t^{n-1})}{2}$, which leads to the displacements

$$\mathbf{d}^{n+1} = \mathbf{d}^n + \Delta t^n\, \dot{\mathbf{d}}^{n+1/2}, \tag{3}$$

with the current time step size $\Delta t^n = t^{n+1} - t^n$. The time step size is limited by the COURANT-criterion by

$$\Delta t \leq \alpha\, \Delta t_{crit} = \alpha\, \frac{2}{\omega_{max}} \approx \alpha \left(\min_e \frac{l_e}{c_e} \right), \tag{4}$$

where ω_{max} is the largest eigenfrequency, l_e represents a characteristic element length and c_e the wave propagation velocity. The COURANT-criterion is based on linear problems, so in order to consider non-linearities, the factor $\alpha < 1$ is introduced. For moderately non-linear application, usually $\alpha = 0.9$ is sufficient, for applications as e. g. high-speed impact problems, α may have to be 0.9 or even less.

The implementation of the central difference method leads to a system of uncoupled linear equations and only vector operations are performed on global level if diagonal mass matrices are used. This leads to very little CPU-time requirements per time step, compared to implicit methods. The limitation of the time step by Equation (4) makes this method especially appropriate for highly dynamic applications such as crash or impact and for problems with strong non-linearities. For long-term dynamic problems, a very large number of time steps is required, which may lead to a long simulation time, however it is a purely time marching scheme. Efficiency depends mainly on the evaluation of the internal forces, the main topic of the following sections.

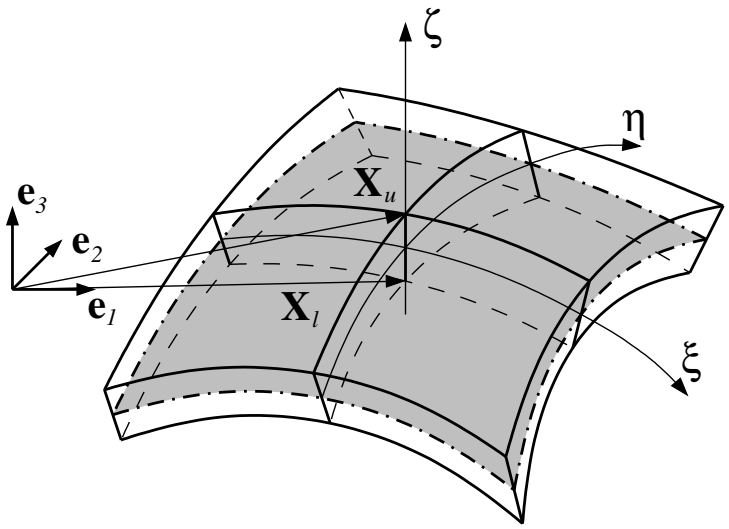


Figure 1: geometry of a solid shell

3 The Solid-Shell Concept

In this section, a very short introduction into the Solid-Shell concept is given. For more detailed information, it is referred to the comprehensive literature, e.g. [18, 11, 17]. More recent developments regarding especially the treatment of artificial stiffness effects by reduced integration techniques are given e. g. in [1, 19, 7, 20] in the context of implicit finite element applications.

3.1 Kinematics

The Solid-Shell concept provides a shell formulation with displacement degrees of freedom only. Under the assumption of the degenerated shell concept that the normals to the mid-surface remain straight, following the notation given in Figure 1, the initial geometry is given by

$$\mathbf{X}(\xi,\eta,\zeta) = \frac{1}{2}\left((1+\zeta)\,\mathbf{X}_u(\xi,\eta) + (1-\zeta)\,\mathbf{X}_l(\xi,\eta)\right), \tag{5}$$

Linear interpolation of the displacements of the upper and the lower surface leads to

$$\mathbf{u}(\xi,\eta,\zeta) = \frac{1}{2}\left((1+\zeta)\,\mathbf{u}_u(\xi,\eta) + (1-\zeta)\,\mathbf{u}_l(\xi,\eta)\right). \tag{6}$$

In this contribution, linear and quadratic, isoparametric Solid-Shell elements are used with bi-linear/ bi-quadratic interpolation in membrane and linear interpolation in thickness direction. For the discretization of the initial geometry, this leads to

$$\mathbf{X}^{el}(\xi,\eta,\zeta) = \sum_{i=1}^{nip}\left(\frac{1}{2}\,N_i(\xi,\eta)\,\boldsymbol{\Theta}(\zeta)\,\mathbf{X}_i\right), \tag{7}$$

where nip is the number of in-plane nodes. The upper and lower nodal locations are described by the vector $\mathbf{X}_i = \left[\begin{array}{cc}\mathbf{X}_{iu} & \mathbf{X}_{il}\end{array}\right]^T$, the interpolation is performed linearly in thickness direction with the interpolation matrix $\boldsymbol{\Theta}(\zeta)$. The in-plane interpolation is achieved in the present case with linear ($nip = 4$) or quadratic ($nip = 9$) Lagrangian shape functions. According to the isoparametric concept, the displacements are interpolated with the same shape functions. The application of higher order Lagrangian or Serendipity type shapes is also possible, quadratic Serendipity functions e. g. lead to a formulation with 16 element nodes ($nip = 8$).

3.2 Residual Force Vector

The system force vector in Equation (1) is composed of the globally defined external nodal forces $\mathbf{f}^{ext}$ and the internal nodal forces $\mathbf{f}^{int}$ which are computed on element level. The internal forces can be written as the derivative of the internal energy Π^{int} with respect to the vector of nodal degrees of freedom $\mathbf{d}$. The implemented element formulations are formulated with different hyper-elastic material models as e. g. *Neo-Hooke* material law, defined by the strain energy function

$$W(\mathbb{C}) = \frac{\mu}{2}\left(\mathrm{I}_{\mathbb{C}} - 3\right) - \mu \, \ln J + \frac{\lambda}{2}\left(\ln J\right)^2 \quad \text{with} \quad J^2 = \mathrm{III}_{\mathbb{C}} \,, \tag{8}$$

with the first and third invariant $\mathrm{I}_{\mathbb{C}}$ and $\mathrm{III}_{\mathbb{C}}$ of the *right Cauchy-Green deformation tensor*. The internal energy and hence the internal nodal force vector is then obtained by

$$\Pi^{int} = \int_V W(\mathbb{C}) \, dV \quad , \quad \mathbf{f}^{int} = \Pi^{int}_{,\mathbf{d}} = \frac{\partial}{\partial \, \mathbf{d}} \int_V W(\mathbb{C}) \, dV \,. \tag{9}$$

3.3 Mass Matrices

As mentioned in Section 2, an efficient usage of the central difference method implies diagonalized mass matrices. The entries of the consistent element mass matrix $\mathbf{M}^{el}$

$$\mathbf{M}^{el} = \int_V \rho \, \mathbf{N} \cdot \mathbf{N}^T \, dV \tag{10}$$

with the shape functions assembled in the matrix $\mathbf{N}$ are therefore e. g. summed up row by row, in order to achieve the diagonalized form

$$M^{el,d}_{ij} = \begin{cases} \sum_k M^{el}_{ik} & i = j \\ 0 & i \neq j \end{cases} \tag{11}$$

For element formulations using the Lagrangian shape functions, this method leads to identical or very similar mass matrices as other methods as e. g. described in [12] such as '*nodal integration*' or '*scaled diagonals*' which are also implemented in our code. When using Serendipity type shape functions, the '*row-sum*'-technique in Equation (11) is not preferable, as it might lead to negative entries in the diagonalized mass matrix; other schemes are also not perfect. For this reason Serendipity elements are not used here, though they work very well for implicit methods.

4 Treatment of 'Locking' Phenomena

A very important issue concerning the implementation of lower order shell finite elements is the activation of artificial stresses for different loading situations, the so-called '*locking*' phenomena. Though not as distinctive as in elements with linear displacement interpolation, locking also appears with quadratic shape functions and can be reduced – or even completely removed – by several corrections within the element formulation. Proposals for locking-free Solid-Shell elements, using reduced integration rules together with stabilization techniques against artificial kinematics can be found in [1, 19, 7, 20]. In the current contribution, fully integrated element formulations are presented, where different locking phenomena are treated with the well-known methods of '*Assumed Natural Strains* (ANS)' [2, 6] and '*Enhanced Assumed Strains* (EAS)' [23, 22], which have already been applied to Solid-Shell elements for non-linear implicit analyses [11, 9, 10].

4.1 Assumed Natural Strains

The so-called *geometric* locking effects, e. g. (transverse) shear locking, curvature thickness locking and – when using quadratic interpolation of geometry – also membrane locking are treated with the method of '*Assumed Natural Strains* (ANS)'. Specific loading scenarios lead to artificial stiffnesses, caused by an insufficient interpolation of the strains. This behavior can be cured by evaluating the strain function at sampling points and interpolate these values with a specific order. The method was presented for four node elements by BATHE and DVORKIN in [2] for transverse shear strains and extended by BETSCH and STEIN in [4] and independently by BISCHOFF and RAMM in [5] to normal strains in thickness direction. An application of the method to elements with quadratic shape functions was presented by BUCALEM and BATHE in [6].

CARDOSO ET.AL suggested for linear Solid-Shell elements in [7] an evaluation at two sampling points over the thickness ($\zeta = -1/\zeta = 1$), together with a linear interpolation in thickness direction. The implemented linear and quadratic Solid-Shell formulations contain interpolations of the transverse shear strains and – in order to cure membrane locking – the membrane strains of the quadratic elements are also interpolated. As an example, the interpolation rule for the transverse shear strain $E_{\eta\zeta}$ is given for the linear

$$
\begin{aligned}
{}^{ANS}E_{\eta\zeta} = \frac{1}{2}(1-\zeta)\left(\sum_{i=1}^{2}\frac{1}{2}\left(1+\xi_i\,\xi\right)E_{\eta\zeta}(\xi_i,\eta_i,-1)\right) + \\
\frac{1}{2}(1+\zeta)\left(\sum_{i=1}^{2}\frac{1}{2}\left(1+\xi_i\,\xi\right)E_{\eta\zeta}(\xi_i,\eta_i,1)\right)
\end{aligned}
$$

with

$$
\xi_i = \begin{pmatrix} -1 & 1 \end{pmatrix} \quad , \quad \eta_i = \begin{pmatrix} 0 & 0 \end{pmatrix} \tag{12}
$$

and for the quadratic Solid-Shell formulation

$$
\begin{aligned}
{}^{ANS}E_{\eta\zeta} = \frac{1}{2}(1-\zeta)\left(\sum_{i=1}^{3}\sum_{j=1}^{2} q_i\,\ell_j\,E_{\eta\zeta}(\xi_i,\eta_j,-1)\right) + \\
\frac{1}{2}(1+\zeta)\left(\sum_{i=1}^{3}\sum_{j=1}^{2} q_i\,\ell_j\,E_{\eta\zeta}(\xi_i,\eta_j,1)\right)
\end{aligned}
$$

with

$$
\begin{aligned}
q_i &= \begin{cases} {}^1\!/_2\sqrt{{}^5\!/_3}\left(\sqrt{{}^5\!/_3}\,\xi - 1\right) \\ 1 - {}^5\!/_3\,\xi^2 \\ {}^1\!/_2\sqrt{{}^5\!/_3}\left(\sqrt{{}^5\!/_3}\,z + 1\right) \end{cases} \quad , \quad \ell_j = \begin{cases} {}^1\!/_2\,(1-\sqrt{3}\,\eta) \\ {}^1\!/_2\,(1+\sqrt{3}\,\eta) \end{cases} \quad \text{and} \\
\xi_i &= \begin{pmatrix} -\sqrt{{}^5\!/_3} & 0 & \sqrt{{}^5\!/_3} \end{pmatrix} \quad , \quad \eta_j = \begin{pmatrix} -\sqrt{{}^1\!/_3} & \sqrt{{}^1\!/_3} \end{pmatrix}
\end{aligned} \tag{13}
$$

The different strain components are evaluated at different sampling points and interpolated in a similar way, in order to eliminate the artificial stiffness effects.

An interpolation of the normal strains in thickness direction in order to cure the so-called 'Curvature Thickness Locking' is proposed by BETSCH and STEIN in [4] and BISCHOFF and RAMM in [5] for 4-node shell elements. The strains are evaluated at the element nodes and

interpolated with the Lagrangian shape functions. For the Solid-Shell elements, the evaluation of the strains is performed at the nodes of the mid-surface, which leads to

$$^{ANS}E_{\zeta\zeta} = \sum_{i=1}^{4} N_i^{lin}(\xi,\eta)\, E_{\zeta\zeta}(\xi_i,\eta_i)\,. \tag{14}$$

for the linear and

$$^{ANS}E_{\zeta\zeta} = \sum_{i=1}^{9} N_i^{quad}(\xi,\eta)\, E_{\zeta\zeta}(\xi_i,\eta_i)\,. \tag{15}$$

for the quadratic formulations.

4.2 Enhanced Assumed Strains

Unlike the geometrical locking effects, which are controlled by the interpolation of geometry and displacements, the so-called material locking – also referred as volumetric locking or POISSON locking – is controlled by a material parameter, the POISSON ratio ν. An important example for Solid-Shell elements is the so-called POISSON thickness locking, which goes back to the fact that for a pure bending scenario, the condition of disappearing normal thickness stresses can not be satisfied in general. This leads to an overly stiff behavior, especially near the incompressible limit ($\nu \to 0.5$). At $\nu = 0.0$, the locking effect does not appear at all, as the stress components are uncoupled.

As a general concept to avoid artificial stresses, the '*Enhanced Assumed Strain*' method (EAS) was introduced by SIMO and RIFAI in [23] and generalized by SIMO and ARMERO in [22]. The method is based on the idea of enhancing the compatible strain field by introducing additional degrees of freedom, which can be condensed out on element level.

For the current contribution, different EAS formulations have been implemented in order to cure POISSON thickness locking for Solid-Shell elements. Therefore, only the normal strains in thickness direction have to be enhanced using

$$\tilde{E}_{33} = \frac{\det \mathbf{J}_0}{\det \mathbf{J}}\, t_{33}\, \mathbf{M}^i\, \boldsymbol{\alpha}\,, \tag{16}$$

where $\mathbf{J}_0$ describes the Jacobian $\mathbf{J}$, evaluated at the element center and t_{33} is required for transformation to the local element co-ordinates. The matrices $\mathbf{M}^i$ contain the interpolation functions for example for linear elements

$$\mathbf{M}^1 = \left[\ \zeta\ \right]\,, \quad \mathbf{M}^3 = \left[\ \zeta \quad \xi\zeta \quad \eta\zeta\ \right] \text{ and } \quad \mathbf{M}^4 = \left[\ \zeta \quad \xi\zeta \quad \eta\zeta \quad \xi\eta\zeta\ \right]\,. \tag{17}$$

The superscript at the matrices $\mathbf{M}$ indicates the number of additional degrees of freedom and hence the dimension of the vector $\boldsymbol{\alpha}$. In order to reach a consistent enhancement for the quadratic Solid-Shell formulation, 8 parameters are necessary with the interpolation matrix

$$\mathbf{M}^8 = \left[\ \zeta \quad \xi\zeta \quad \eta\zeta \quad \xi\eta\zeta \quad \xi^2\zeta \quad \eta^2\zeta \quad \xi^2\eta\zeta \quad \xi\eta^2\zeta\ \right]\,. \tag{18}$$

On element level, the internal energy can now be computed with the compatible and the enhanced strains as in Equation (9). An additional condition has to be satisfied locally, leading to the increment of the additional degrees of freedom

$$\Delta\boldsymbol{\alpha} = -\mathbf{D}^{-1}\,\mathbf{P}\,, \tag{19}$$

with the matrices

$$\mathbf{P} = \Pi^{int}_{,\boldsymbol{\alpha}} \qquad \text{and} \qquad \mathbf{D} = \Pi^{int}_{,\boldsymbol{\alpha}\boldsymbol{\alpha}} \tag{20}$$

as derivatives of the internal energy with respect to the additional degrees of freedom.

As can be seen in Equation (19), the matrix $\mathbf{D}$ from Equation (20) has to be inverted on element level. The dimension of this square matrix is equal to the number of EAS parameters, thus the numerical effort of the EAS elements is strongly affected by the number of enhancements.

5 Implementation Concept

Compared to implicit time integration algorithms, which are dominated by the solution of equations, the central difference scheme as an explicit time integration method requires far less operations on global level. As discussed in Section 2, only vector operations have to be performed on global level. This and the already mentioned very small time steps lead to the fact that – within an explicit time integration scheme – most of the time, necessary for an entire structural analysis, is spent on element level. In different examples, computed with the in-house finite element code FEAP-MEKA [21], based on FEAP by R.L. TAYLOR, up to – or even more than – $90\,\%$ of the overall CPU-time had been spent on element level. For commercial codes, this fracture may be smaller, due to computationally expensive procedures on global level which are not yet implemented in the explicit version of the used academic code, e. g. contact search algorithms. Nevertheless it is obvious, that the element processing requires a dominant part of the overall simulation time, which motivates an efficient implementation of element code.

5.1 AceGen

In order to achieve an efficient and comfortable implementation of the subroutines on element level, the improved – so-called '*automatic*' – code generation and optimization tool ACEGEN, a plug-in for the computer algebra software MATHEMATICA is used. The program is developed by the group of KORELC, see [13, 16, 14]. The plug-in uses the symbolic capabilities of MATHEMATICA in order to create automatically optimized program code in FORTRAN. It is possible to enter formulas as written, without bothering about programming issues. Hence matrix operations, summations, differentiation – also with respect to vectors and matrices – can be implemented straight forward and programming errors can be reduced significantly.

For the numerical integration of Equation (9) at the current time step, the operation

$$\mathbf{f}^{int} = \frac{\partial \Pi_{int}}{\partial \mathbf{d}} = \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{n} \frac{\partial W(\xi_i, \eta_j, \zeta_k)}{\partial \mathbf{d}} \det \mathbf{J}(\xi_i, \eta_j, \zeta_k)\, w_i\, w_j\, w_k \tag{21}$$

has to be evaluated for each element using an integration rule with m x m quadrature points in-plane and n points in thickness direction. ACEGEN allows the usage of MATHEMATICAs symbolic capabilities, so implemented functions can be used in order to perform matrix operations or differentiations. This increases the convenience of programming and decreases the number of programming errors considerably. Also the computational speed is increased by using automatic code generation, which is shown in the following section. For implementation, the following steps – itemized in Figure 2 – have to be carried out:

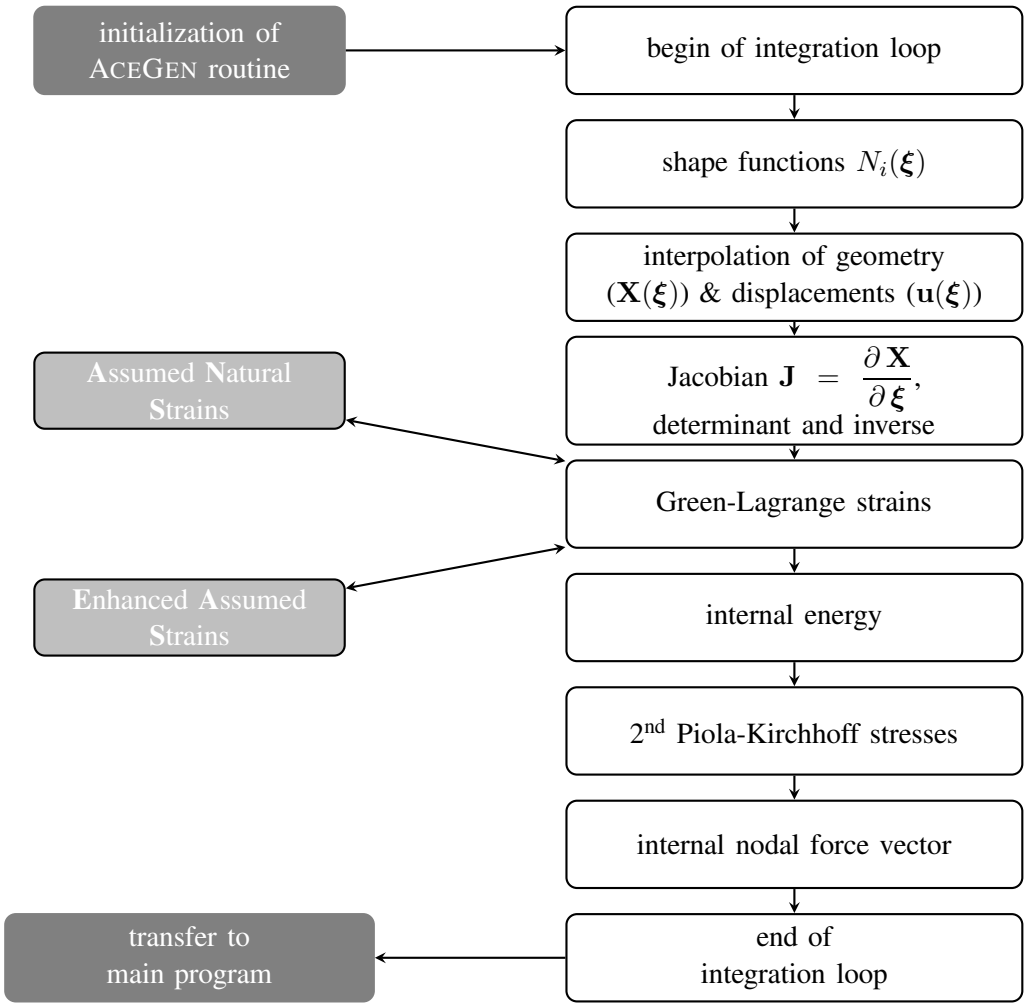


Figure 2: implementation flow-chart for element subroutine with ACEGEN

Initialization: The subroutine as well as the input (geometry, current displacements, coordinates and weights of the quadrature points and material parameters) and output variables (internal force vector) are defined. The used ACEGEN commands are `SMSInitialize` and `SMSModule`.

Element matrices: After the import of the element data, the necessary matrices and vectors – Jacobian, convective base vectors, Green-Lagrange strain tensor, etc. – can be evaluated using MATHEMATICA's symbolic capabilities. Differentiation with respect to variables or tensors is also possible (`SMSD`), which is used e. g. to evaluate $\Pi^{int}_{,\mathbf{d}}$.

Export internal force vector: The internal force vector at the current integration point has to be exported, to be available outside the symbolic subroutine. The command `SMSExport` is used with the option `"AddIn"=True` in order to automatically sum the results for all quadrature points to the global memory field.

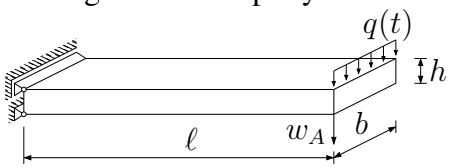
Code generation: In the last step, FORTRAN-code is generated and automatically optimized, using the command `SMSWrite`. The language (FORTRAN, C, etc.) and the level of optimization depend on options, defined in `SMSInitialize`. Figure 3 shows a portion of this automatically generated FORTRAN-code.

The efficiency and performance of the automatically generated and simultaneously optimized subroutines is coupled to some conditions and rules, one has to consider when using the programming tool. The main challenge is to reach a consequent application of the symbolic capabilities. It is important to identify the numerically 'expensive' operations within an algorithm and concentrate them into as few as possible routines. The subdivision of a procedure into many small routines as it is usually done in manual programming is not optimal here, as the symbolically used variables have to be initialized every time. In the present case, all operations necessary for the computation of the internal forces are performed in a single routine as shown symbolically in Figure 2. Consequently, ACEGEN is able to optimize the generated code with regard to the performed operations and produce highly efficient code. Compared to manual programming, the initialization and introduction of variables as well as the transfer of data into and out of the subroutine is the main difficulty when using ACEGEN. Another problem is the debugging, as the generated code is not longer readable – as can be seen in Figure 3. Further any change in the program requires a complete re-generation of the subroutine with ACEGEN. Errors in the initialization of variables can lead to wrong results when performing e. g. symbolic differentiations. This errors can only be found and corrected using MATHEMATICA, as manual debugging is not possible.

The correct application of ACEGEN – which required some learning time and gaining of experience – allows the direct implementation of symbolic algorithms and the fast and error-free generation of program code. Also modification in the element formulation can be implemented without programming errors. The generated code is highly efficient, as can be seen in the following numerical examples. As the contrast an improvement of manually implemented routines in order to achieve a significant speed-up is exhausting and error-prone and hence time-consuming.

```
[...]
v(1061)=v(1624)*v(616)
v(1062)=v(1625)*v(616)
v(1063)=v(1626)*v(616)
v(1064)=v(1624)*v(572)+v(1621)*v(599)
v(1065)=v(1625)*v(572)+v(1622)*v(599)
v(1066)=v(1626)*v(572)+v(1623)*v(599)
[...]
```

Figure 3: exemplary section of automatically generated and optimized FORTRAN code



$$
\begin{array}{lcllcl}
E & = & 2.10 \cdot 10^5 & \ell & = & 10 \\
\nu & = & 0.0/0.3/0.499 & b & = & 1 \\
\rho & = & 7.85 \cdot 10^{-9} & h & = & 0.1 \\
q(t) & = & \begin{cases} 100\, q_0 & 0.00 \le t \le 0.01 \\ q_0 & 0.01 < t \le 0.02 \end{cases} & q_0 & = & 1
\end{array}
$$

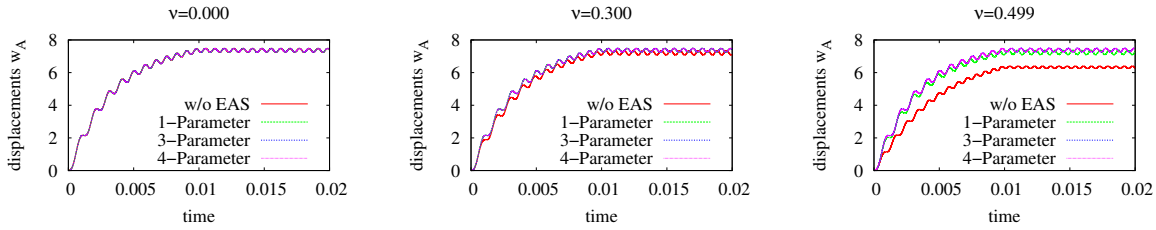


Figure 4: Test for volumetric locking; Cantilever with tip load, varying POISSON ratio

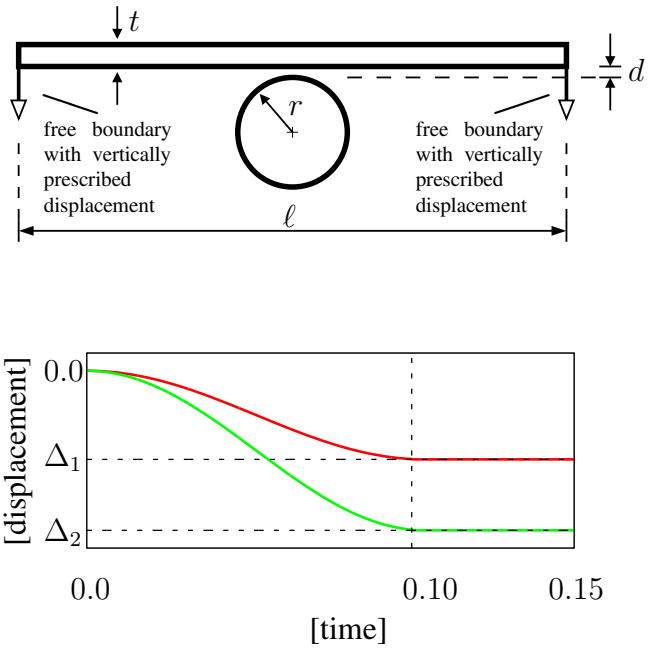
6 Numerical Examples

6.1 Bending of Cantilever Beam

As a first, very simple example, a benchmark for volumetric locking, well-known from static analyses – the clamped cantilever beam with tip load – is discretized with regular mesh of 10 linear Solid-Shell elements. As the boundaries are chosen statically determinate and geometrical locking effects are cured, the tip displacement w_A must be invariant against the POISSON ratio ν. As shown in Figure 4 and known from statics, three EAS-parameters are sufficient for correct results in this example. For $\nu = 0.30$, even one single EAS-parameter is enough to cancel the locking effect, which is especially important for explicit analyses regarding the numerical effort. The CPU-times measured in the present examples for $\nu = 0.00$ are

- without EAS: 82 sec
- 1 EAS parameter: 104 sec
- 3 EAS parameter: 162 sec
- 4 EAS parameter: 223 sec .

It must be noted, that a time history analysis is performed for this quasi-static case with 1.15 mill. time steps in order to demonstrate the functionality of the implemented approach in the context of an explicit time integration scheme . A simulation of the same example with a fully integrated manually programmed element formulation required 1696 sec, which shows the efficiency of the ACEGEN generated routines. A manual improvement or the routines by hand is possible, but would be time consuming and error-prone.



$$
\begin{array}{lcl}
E & = & 210 \cdot 10^9 \\
\nu & = & 0.3 \\
\rho & = & 7.85 \cdot 10^{-9} \\
\ell & = & 2 \\
t & = & 0.02 \\
r & = & 0.25 \\
d & = & 0.01 \\
 & & \\
\Delta_1 & = & 0.5 \\
\Delta_2 & = & 0.9
\end{array}
$$

Figure 5: Thin elastic plate with impacting rigid sphere: (a) geometry and boundary conditions; (b) loading diagram; (c) material and geometrical data

6.2 Thin Elastic Plate with Contact

In a second, more sophisticated example, the impact of a thin elastic plate on a rigid sphere is simulated. The contact between the elastic structure and the rigid surface is realized by a 'Mortar'-type penalty contact formulation (i. e. *Gaußpoint*-wise penetration check) with an analytical description of the contact surface. The operations of the contact routines, which also have to be performed for every time step have also been implemented using AceGen, but will not be discussed here in detail. Geometry and material properties, as well as the loading scenario in the form of a displacement boundary condition applied to the plate's edges can be obtained from Figure 5.

6.2.1 Performance Study

The simulation with the first load curve with a maximum displacement of $\Delta_1 = 0.5$ lead to a bending dominated deformation of the structure. In order to show the efficiency of the implemented element and contact algorithms, the simulation has been performed with a pure displacement formulation without any strain modifications, as for this formulation, the routines are available both manually programmed and AceGen generated. Table 1 shows the CPU-times of the simulations on a 20×20 element mesh with linear Solid-Shells, which required 53.120 time steps. As can clearly be seen, the AceGen routines lead to a significant decrease of the computational effort – in this example, the CPU-time can be reduced by a factor of 15. The influence of the contact routines is smaller, as the fraction of contact processing itself – compared to the element operations – is smaller in this example. The high efficiency of the implemented routines can be reached without any manual improvement of the program code, as AceGen simultaneously optimizes the code regarding the operations. This leads to a faster program as well as to a faster and less error-prone programming.

Solid-Shell subroutine	contact subroutine	CPU-time [s]	relative
manually	manually	3202	100.00 %
manually	ACEGEN	3103	96.91 %
ACEGEN	manually	309	9.65 %
ACEGEN	ACEGEN	207	6.46 %

Table 1: CPU-times of simulations with manually and ACEGEN programmed routines

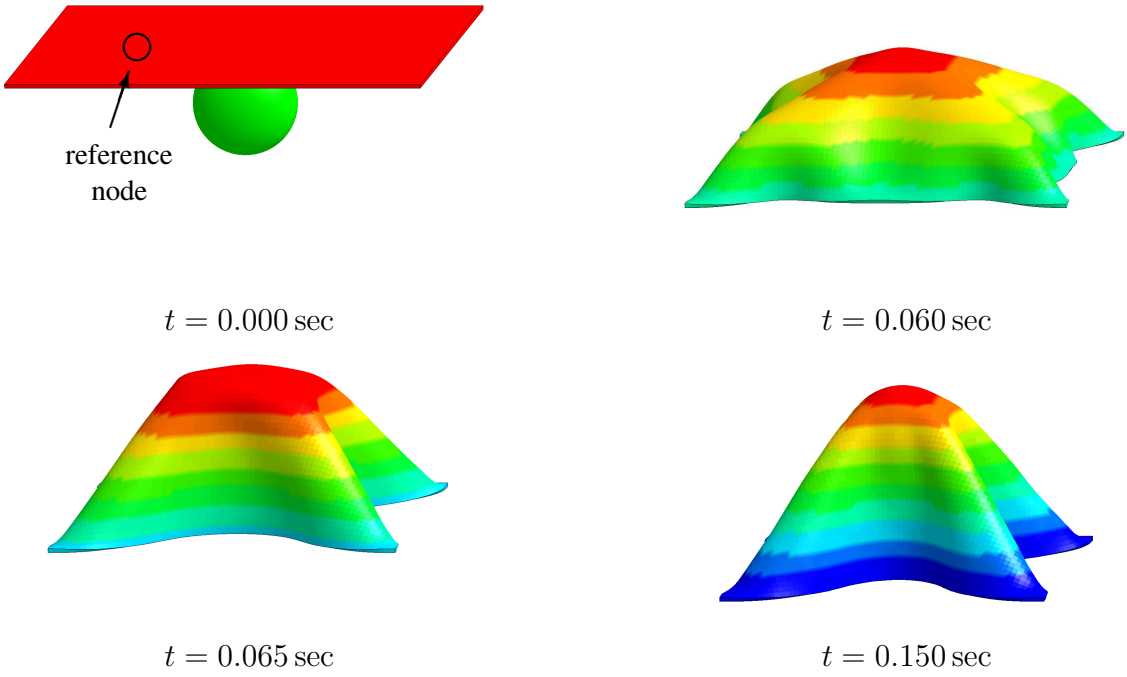
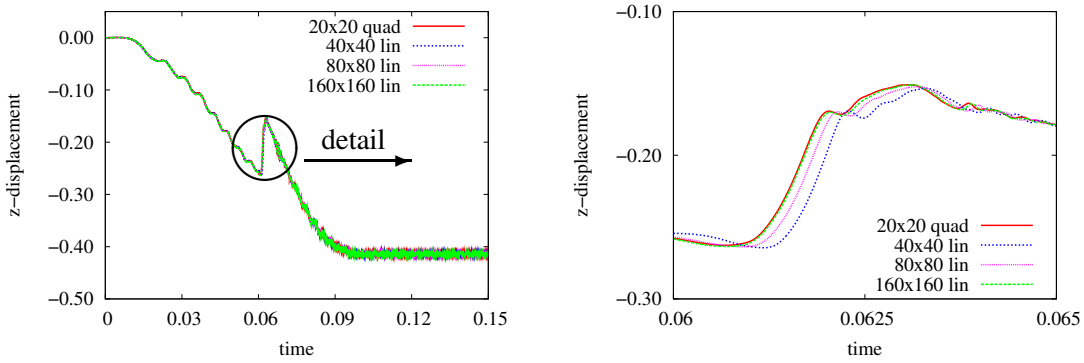


Figure 6: 'snap-through' effect of thin elastic plate

6.2.2 'Snap-Through' Effect

Regarding the second loading displacement curve given in Figure 5 (b) with a maximum displacement of $\Delta_2 = 0.9$, the structure shows a 'snap-through' effect at $\Delta \approx 0.60$. The application of explicit time integration, which is not affected by singularities of the system matrix as in implicit static analysis allows the simulation of structural stability problems as a dynamic 'snap-through' process.

The structure is simulated with different meshes with linear and quadratic Solid-Shell elements. In Figure 7 the vertical displacements of the reference node, depicted in Figure 5 are given. The left diagram shows the results of the complete simulations, which correlate very well for the different discretizations. As can be seen in the detail on the right side, far more linear elements are necessary in order to reproduce the results reached with the quadratic shape functions. This clearly shows the improvement of the geometry and displacement interpolation when using higher order elements, especially at curved geometries. This conclusion would be even more articulate when investigating initially curved geometries with high curvatures.



(a) displacement of reference node

(b) detail with 'snap-through'

Figure 7: results of different discretizations with linear and quadratic Solid-Shells

7 Conclusions & Outlook

In the paper Solid-Shell element formulations with linear and quadratic interpolation of the in-plane geometry and displacement with *Assumed Natural Strains* and *Enhanced Assumed Strains* in order to reduce artificial stiffness effects are presented. An implementation concept using the automatic code generation tool ACEGEN based on the computer algebra program MATHEMATICA is shown and the advantages regarding programming and computational efficiency are discussed. Especially in the context of explicit time integration, efficient element routines are very important, as the fracture of element processing on the overall simulation time is extremely high. The numerical examples show the improvement of the ACEGEN generated element routines compared to a manually performed implementation.

In the course of the project, the application of ACEGEN for further element formulations is planned, also structural elements (i. e. shell elements) with linear and quadratic geometry and displacement interpolations are currently implemented, as they are especially important for applications with explicit time integration. As a very positive result we can conclude that the use of symbolic programming seems to be advantageous in many applications in structural mechanics. A very promising application is the implementation of complex material models, as here the advantage of automatic differentiation is particularly interesting.

Acknowledgements

The presented work is part of a project, currently funded by the German Research Foundation (DFG). The support is gratefully acknowledged.

References

[1] R.J. Alves de Sousa, R.P.R. Cardoso, R.A. Fontes Valente, J.W. Yoon, R.M. Natal Jorge, and J.J. Gracio. A new one-point quadrature enhanced assumed strain (EAS) solid-shell element with multiple integration points along thickness: Part I – geometrically linear applications. *Int. J. Num. Meth. Eng.*, 62:952–977, 2005.

[2] K.-J. Bathe and E.N. Dvorkin. A formulation of general shell elements - the use of mixed interpolation of tensorial components. *International Journal For Numerical Methods In Engineering*, 22:697–722, 1986.

[3] T. Belytschko, W.K. Liu, and B. Moran. *Nonlinear finite elements for continua and structures*. Wiley, 2004.

[4] P. Betsch and E. Stein. An assumed strain approach avoiding artifical thickness straining for an non-linear 4-node shell element. *Communications In Numerical Methods In Engineering*, 11:899–909, 1995.

[5] M. Bischoff and E. Ramm. Shear deformable shell elements for large strains and rotations. *International Journal For Numerical Methods In Engineering*, 40:4427–4449, 1997.

[6] M.L. Bucalem and K.-J. Bathe. Higher-order mitc general shell elements. *International Journal For Numerical Methods In Engineering*, 36(21):3729–3754, 1993.

[7] R.P.R. Cardoso, J.W. Yoon, M. Mahardika, S. Choudhry, R.J. Alves de Sousa, and R.A. Fontes Valente. Enhanced assumed strain (eas) and assumed natural strain (ans) methods for one-point quadrature solid-shell elements. *International Journal For Numerical Methods In Engineering*, 75:156–187, 2008.

[8] R. Courant, K.O. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100:32–74, 1928.

[9] M. Harnau, K. Schweizerhof, and R.Hauptmann. On 'solid-shell' elements with linear and quadratic shape functions for small and large deformations. *ECCOMAS Congress*, 11, 2000.

[10] R. Hauptmann, S. Doll, M. Harnau, and K. Schweizerhof. 'solid-shell' elements with linear and quadratic shape functions at large deformations with nearly incompressible materials. *Computers & Structures*, 79(18):1671–1685, 2001.

[11] R. Hauptmann and K. Schweizerhof. A systematic development of 'solid-shell' element formulations for linear and non-linear analyses employing only displacement degrees of freedom. *Int. J. Num. Meth. Eng.*, 42(1):49–69, 1998.

[12] Thomas J. R. Hughes. *The finite element method*. Dover Publ., dover ed., 1. publ. edition, 2000.

[13] J. Korelc. Automatic generation of finite-element code by simultaneous optimization of expressions. *Theoretical Computer Science*, 187(1-2):231–248, 1997.

[14] J. Korelc. Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with Computers*, 18(4):312–327, 2002.

[15] J. Korelc. `http://www.fgg.uni-lj.si/Symech/`, 2010.

[16] J. Korelc and P. Wriggers. Computer algebra and automatic differentiation in derivation of finite element code. *ZAMM*, 79:811–812, 1999.

[17] C. Miehe. A theoretical and computational model for isotropic elastoplastic stress analysis in shells at large strains. *Comput. Meth. Appl. Mech. Eng.*, 155(3-4):193–234, 1998.

[18] H. Parisch. A continuum-based shell theory for non-linear applications. *Int. J. Num. Meth. Eng.*, 38:1855–1883, 1995.

[19] S. Reese. A large deformation solid-shell concept based on reduced integration with hourglass stabilization. *Int. J. Num. Meth. Eng.*, 69(8):1671–1716, 2007.

[20] M. Schwarze and S. Reese. A reduced integration solid-shell finite element based on the eas and the ans concept - geometrically linear problems. *International Journal for Numerical Methods in Engineering*, 80(10):1322–1355, 2009.

[21] K. Schweizerhof and Coworkers. Feap-meka, finite element analysis program. Karlsruher Institut für Technologie, based on Version 1994 of R. Taylor, "FEAP – A Finite Element Analysis Program", University of California, Berkeley.

[22] J.C. Simo and F. Armero. Geometrically non-linear enhanced strain mixed methods and the method of incompatible modes. *International Journal For Numerical Methods In Engineering*, 33:1413–1449, 1992.

[23] J.C. Simo and M.S. Rifai. A class of mixed assumed strain methods and the method of incompatible modes. *Computers & Structures*, 29:1595–1638, 1990.

[24] L. Verlet. 'Experiments' on classical fluids I. Thermomechanical properties of Lennard-Jones molecules. *Physical Review*, 159:98–103, 1967.