Symbolic software tools in the development of finite elements

Anders Eriksson*, Costin Pacoste

Structural Mechanics Group, Department of Structural Engineering, Royal Institute of Technology, SE-100 44, Stockholm, Sweden

Received 16 October 1998

Abstract

Symbolic software has been used in a number of projects concerned with the development of finite element procedures, primarily aiming at complex, i.e. interacting and higher order instabilities, where high accuracy in formulations is required. The symbolic tools improve the efficiency and documentation of the developed procedures, in order to facilitate comparisons between different element assumptions. Beam formulations for plane and space models were developed, in total displacement and co-rotational contexts, respectively. Symbolic derivation allowed analytical verification of equivalence between certain formulations within these two contexts. Treatment of finite space rotations, based on the rotational vector makes the history-less treatment of rotations easier, which is needed in the evaluation of critical equilibrium subsets in higher-dimensional parameter space. A co-rotational viewpoint, where local element displacements can be obtained from global variables in a systematic manner, allowed different element expressions in a common framework. Different simple, linear elements have been tested with respect to computational efficiency. A field consistence approach was used to develop highly accurate beam and plane stress elements. The common element formulations, based on the matrix multiplication $B^TDB$, is often inefficient, due to the large number of operations needed in the matrix product. Other formulations, based on an analytical integration and differentiation of the strain energy, producing explicit expressions for the stiffness terms, were considerably more efficient for certain elements.

Keywords: finite elements; non-linear mechanics; instability; symbolic software

1. Introduction

Previous papers by the authors have discussed the treatment of complex instability phenomena [1,2]. The term ‘complex’ here refers to higher order phenomena, e.g. butterfly situations, but also to interacting instability phenomena, where several bifurcation or limit states affect one another. Several situations of these types have been shown for real structures [3].

One conclusion from the previous work has been that a high degree of accuracy is needed in the treatment of the phenomena. Simplified models tend to hide the important aspects of behaviour, leading to over-simplified solutions, and neglecting some critical phenomenon types. This demand for accuracy primarily relates to the formulations of the finite elements used for discretization of the structures [1]. In short, the conclusion is that the element must be based on
carefully chosen kinematic assumptions, and that the mathematical derivation of the element must be highly accurate.

Another conclusion is drawn from the work in [4]. As the complex phenomena often only occur for special instances of a generic structure, it is important to have numerical methods available, which can accurately find these specific instances. Methods for this, following the response in a generic structure as a function of some additional parameters, are discussed in the reference. The methods are based on augmented equilibrium formulations, where the relevant subset of equilibrium states are found in a higher-dimensional space.

1.1. Accuracy demands

The need for accuracy in the element formulation can be seen from an energy viewpoint. The discussion is then based on the treatment of the total potential of a structural model under load. In general, this can be stated as

\[ \pi(u, \lambda) = \phi(u) - \lambda \hat{p}u \]

where the strain energy \( \phi \) is seen as a function of the displacements in \( u \), assuming that no history is needed in the formulation. The potential energy of the applied loading is a linear function in a load factor \( \lambda \) and the displacements, assuming a conservative, one-parameter, proportional loading situation, its distribution described by \( \hat{p} \).

Based on the total potential function \( \pi \), the equilibrium of the model demands stationarity, i.e. a vanishing gradient of the energy with respect to the displacement components. Introducing this in a Taylor expansion of the total potential around an equilibrium state, gives a local energy description in terms of the expansion of the total potential around an equilibrium displacement components. Introducing this in a Taylor gradient of the energy with respect to the equilibrium of the model demands stationarity, i.e. a vanishing.

\[ \pi^\prime(u, \lambda) = \frac{1}{2!} \phi^\prime(u) \delta u^2 + \frac{1}{3!} \phi^\prime(u) \delta u^3 + \frac{1}{4!} \phi^\prime(u) \delta u^4 + \cdots \]

where

\[ \delta u = u - u^* \]

and \( u^* \) contains the displacements at equilibrium for the load factor \( \lambda^* \); \( \phi^\prime \) is expanded around this point. A critical situation occurs when also the second differential of the total potential vanishes. This situation involves a critical mode, and can be expressed as:

\[ \phi^\prime(u) \dot{v} \delta u = 0, \ \forall \delta u \]

where \( \dot{v} \) is the critical mode, corresponding to an eigenmode of the tangential stiffness of the model. Only under the assumption of an identically linear pre-critical path should this mode be referred to as a ‘buckling mode’, in the traditional meaning [3].

Having identified the critical mode, the variation of the total potential \( \pi \) can be expressed as a scalar function of a path length along the buckling mode:

\[ \pi_1^\prime(\xi) = \frac{1}{3!} A_1 \xi^3 + \frac{1}{4!} A_4 \xi^4 + \frac{1}{5!} A_5 \xi^5 + \cdots \]

where \( \xi \) is the length of a projection of the displacement increment from the critical state on the critical mode, and index \( b \) refers to the critical point. Evaluating the differentials, this leads to expressions such as:

\[ A_3 = \phi^\prime(u) \dot{v}^3 \]

Noting the assumed linearity of the load term in the total potential, it is seen that the behaviour of the model around the critical equilibrium state is governed by the strain energy differentials of the model. For treatment of complex instability phenomena, rather high order differentials of this strain energy are needed. For instance, a correct representation of the main behaviour of a butterfly instability needs expressions for the sixth differential of the strain energy.

It is concluded that, although the expressions above refer to the total potential for the whole structural model, the terms are in reality assembled from element terms in an element based model, thereby transferring the need for high order energy differentials to the element expressions.

1.2. Subsets of equilibrium states

In order to treat parameter dependence in structural response, generalized path-following methods are discussed in [4]. It is in the reference shown how many aspects of instability for a discretized structural model can be revealed from an augmented set of equilibrium equations, of the general form:

\[ G(z) = \begin{pmatrix} F(d, A) \\ g(d, A) \end{pmatrix} = 0 \]

where the vector \( z \) contains \( n \) displacement components \( d \), and \( p \) control variables \( A \), the latter represents a main load factor \( \lambda_1 \), and possibly some other parameters representing structural geometry or disturbing load cases. The augmented set of equations contains the \( n \) equilibrium equations corresponding to the displacement components, but also \( r \) added relations, defining a certain subset of equilibrium states in the higher-dimensional space, e.g. the critical states. Interest is focused on the development of numerical algorithms for the treatment of resulting one-dimensional solution sets [4], i.e. the case \( r = p - 1 \), but
algorithms for solution of higher dimensional solution spaces are also possible [5]. These are then in general based on element formulations where additional parameters can be included, thereby showing the inherent sensitivity properties.

The treatment of such multi-parametric problems also need history-less descriptions of the structural problems, i.e. formulations where internal forces are uniquely dependent on the current displacements. This puts specific demands on the material model, but also, as discussed below, on the treatment of finite rotations.

1.3. The applicational objective

As an example for the methods of the previous sections, an analysis has been performed on the commonly discussed T-shaped beam, cf. e.g. [6]. For a division of the structure into 1536 flat facet elements [7], the two first critical loads for a structure of length $L=450$, flange width $w=38$, web height $h=65$, thickness $t=1$, and with material properties $E=70960$, $\nu=0.321$, were found as $P_{cr,1}=2849$, and $P_{cr,2}=3423$. The critical modes at these two bifurcation states are represented by the models in Fig. 1; as the fundamental behaviour is slightly non-linear, these modes are not orthogonal.

As two examples of complex results for this model, Fig. 2 shows a ‘fold line’ for the critical loads as functions of the flange width $w$ of the model, and the post-critical behaviour of the structure at the first critical load, for three different lengths of the model; the length creating the butterfly instability was found from a special path [4].

1.4. Objectives of projects

The example shown, together with others, verifies that complex instability phenomena do occur for real structures, and not just for specially designed mathematical examples. In order to treat these phenomena, three requirements on the structural models must be
fulfilled:

- Elements must be carefully formulated, and consistently treated.
- The derivation of the elements must be efficient, and carefully documented.
- The resulting code formulations of the elements should be efficient.

This paper will discuss how these aspects can be satisfied by a systematic usage of symbolic software in the development of element formulations. The discussed topics will then be seen in the context of usage of the standard versions of the symbolic softwares, without development of special auxiliary packages, as in [8].

2. Beam formulation

A number of plane beam element formulations are given in [1]. The starting point is taken in the geometric description of Fig. 3. Geometric considerations are based on the description of the position vector to the middle line of the beam segment:

\[ r(x) = (x + u(x))i + w(x)j \]

with \( x \) as a coordinate along the initial straight configuration, and \( i, j \) unit vectors in the initial coordinate directions. Strain measures in the beam segment are defined from the differential of the position vector w.r.t. the undeformed state, as discussed in [9]:

\[ \epsilon = (1 + u_x) \cos \theta + w_x \sin \theta - 1, \]
\[ \gamma = w_x \cos \theta - (1 + u_x) \sin \theta \]

With the assumption that ordinary elastic constants can be used for these generalized strains, the strain energy for a beam element of length \( l \) can be written as:

\[ \phi(u) = \frac{1}{2} \int_0^l [EA \epsilon^2 + GA \gamma^2 + EL \kappa^2] \, dx \]

Noting the trigonometric terms in the strain components, it is obvious that the strain energy can be described with different degrees of accuracy. An extensive discussion of the possibilities for representing these trigonometric expressions is given in [1]. Besides the exact formulation, different Taylor expansions of the trigonometric functions can be developed, based on different expansion points and different degrees of the terms; these choices lead to a selective order of accuracy, but also to different computational demands.

The different choices are handled in [2] by symbolic software, where different assumptions are easily handled and well documented. An example is given in Fig. 4, where linear expansions of the sine and cosine terms around a rigid rotation \( \theta = \theta_0 \), and the Bernoulli hypothesis gives the strains, the strain energy and the first and second differentials of the strain energy, i.e. internal force vector \( F \), and tangent stiffness \( k_t \), respectively. The results are here primarily expressed in the four natural displacement components:

\[ f_1 = \arctan \left( \frac{w_j - w_i}{l + u_i - u_j} \right); \quad f_2 = \frac{\theta_j - \theta_i}{2} \]
\[ f_3 = \frac{\theta_i + \theta_j}{2} - f_1; \quad f_4 = \frac{u_j - u_i}{l} \]

where \( \mathbf{q} = [u_i, w_i, \theta_i, u_j, w_j, \theta_j]^T \) contains the nodal displacements in the initial local reference frame.

Further, the axial and transversal components of the displacement field are linearly interpolated, according to:

\[ u_x = f_4; \quad w_x = (1 + f_4) \tan f_1 \]

The rotation of the cross section \( \theta \) is expressed as a sum of two components:

\[ \theta = \theta_0 + \tilde{\theta} \]

where \( \theta_0 \equiv f_1 \) is the rigid rotation component. The
non-rigid rotation component \( \tilde{\theta} \) is interpolated, in a traditional manner, according to:

\[
\tilde{\theta} = \left(1 - 2 \frac{x}{l_0} \right) f_2 + \left(1 - 6 \frac{x}{l_0} + 6 \frac{x^2}{l_0^2} \right) f_3
\]  

(4)

The transformation between these and the six basic displacements is described by a transformation matrix \( A_E \).

In Fig. 4, \( u_i, v_i, t_i, u_j, v_j, t_j \) represent the six nodal quantities—four translations and two rotations.

2.1. Co-rotational form

It is concluded in previous papers that there is no inherent difference between elements based on a total displacement formulation, as in the example in Fig. 4, and elements based on a co-rotational formulation, where the deformational quantities in the element are obtained by eliminating the rigid body motion from the total displacements of the element, cf. e.g. [11,12].

The co-rotational viewpoint is interpreted here, noting that the definition of this term is not fully established and consistent, as the introduction of a 'ghost-reference' state, being a rigidly transported version of the initial state. With respect to this transformed state, deformational displacements are calculated, and internal forces evaluated. In the local reference system, expressions simpler than the ones used in a total displacement formulation can be used without loss of accuracy.

The key part of the co-rotational viewpoint is an extraction of 'local' element displacements from the total displacements, measured in the global system. Also these transformations can be well documented by a symbolic derivation, as shown for the plane beam element in Fig. 5. The result from this derivation are the needed matrices for transformations between the local and global displacement measures, but also the matrices related to the change of directions of the current forces, due to a changed ghost-reference state at increased displacements. It is noted that these expressions are independent of the strain assumptions in the local element frame.

The obtained natural displacement modes for the element are here, as a secant reference system is introduced through the current nodal positions, an elongation and two rotational components:

\[
\tilde{\vec{u}} = l_n - l, \quad \tilde{\vec{\theta}}_i = \theta_i - \theta, \quad \tilde{\vec{\theta}}_j = \theta_j - \theta
\]  

(5)

In the above equations, \( l \) and \( l_n \) denote the initial and current length of the element:

\[
l = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}
\]

(6)

\[
l_n = \sqrt{(x_j + u_j - x_i - u_i)^2 + (y_j + w_j - y_i - w_i)^2}
\]
and \( \theta_r \) denotes the rigid rotation, which can be computed from the global nodal displacements, in principle, as:

\[
\theta_r = \arctan \left( \frac{(x_j - x_i)(w_j - w_i) - (y_j - y_i)(u_j - u_i)}{(x_j - x_i)(x_j - x_i + u_j - u_i) + (y_j - y_i)(y_j - y_i + w_j - w_i)} \right)
\]  

(7)

In the above equations, \((x_i, y_i)\) denote the coordinates, and \((u_i, w_i)\) the displacements of point \(i\) in the global reference system.

Based on these local quantities, force and stiffness expressions of any accuracy can be developed and introduced.

The transformation to the global system of these quantities is obtained from the matrices:

\[
A_e = \left[ \frac{\partial \hat{a}_e^i}{\partial q_j} \right] \quad A_{el} = \left[ \frac{\partial^2 \hat{a}_e^i}{\partial q_j \partial q_k} \right]
\]  

(8)

The symbolic code needed to generate these matrices is shown in Fig. 5. Here, \(A_e\) is the transformation matrix to the global system, whereas \(A_{el}\) is the initial force stiffness matrix, related to the acting normal force.

It is obvious that the symbolic derivation of the element formulations gives a very clear documentation of the assumptions being made. The main assumptions tested in [1] are the effects of the Bernoulli hypothesis, and the different simplifications of the trigonometric terms in the exact strain expression. As there is always a trade-off between accuracy and computational efficiency, it is clearly shown by numerical examples that different degrees of assumptions give different capabilities of the element to handle complex situations. The symbolic software tools also make it possible to compare the expressions obtained from the different assumptions. Variations of these assumptions can be developed and tested very quickly, and with a minimized risk of errors in the algorithmic coding of the used procedures. This feature, which allows the analyst to concentrate on the basic hypothesis and definition equation, instead of programming related issues, is one of the main arguments for the use of symbolic software.

As one example, the origin of locking can be clearly shown by the derivations. A similar viewpoint has also been utilized when developing a ‘field consistence’ approach to locking-free beam elements [13,14].

2.2. Total versus co-rotational formulations

It is analytically verified, by symbolic software, that an equivalence exists in the final expressions, when matching kinematic assumptions are introduced in the two formulations. As an example, a formulation in a total displacement context based on a first-order Taylor expansion of the trigonometric functions around the rigid rotation, is equivalent to a locally linear formulation in a co-rotational context. The context, in which the equivalence is shown, is indicated by Fig. 6.

The discussion in this part verifies that there is no inherent difference between the two basic viewpoints in describing element deformations. The analytical verification of this clearly shows that the two alternatives can be seen as two different ways to group the displacement terms. Although this has only so far been analytically verified for the plane beam formulation, it is numerically verified also for space beam formulations; when the finite space rotation terms are included in the element formulation, the analytical verification has not succeeded, due to the high computational demands. It is, however, assumed that the basic equivalence between the two viewpoints holds also for other element types. This assumption is the basis for developing shell elements in [7].

Finally, it is noted that the possibility of easily performing such comparisons, at least in certain cases, is another strong argument for the use of symbolic software tools.

3. Space beam elements

Space beam elements have been discussed in [2], with derivations based on Fig. 7, cf. [15]. The situation here is considerably more complicated, due to the finite space rotations. Thus, the set \(C\) of all configurations of the beam is given by

\[
C = \{ (r,R) \mid r:(0,l) \to \mathbb{R}^3; \quad R:(0,l) \to SO(3) \}
\]

where \(r\) is a position vector and \(R\) is an orthogonal matrix specifying the orientations of the cross section. Defined as above, the configuration space is not a linear vector space but a non-linear differentiable manifold, due to the presence of the special orthogonal (Lie) group \(SO(3)\).
Similarly to the plane beam elements, the derivation of the element expressions can be done in total displacement or co-rotational form; these are numerically proven to be identical, with matching kinematic element assumptions. Symbolic derivation of element expressions has been used to derive several variations, based on linear or non-linear strain assumptions, and introducing different simplifications, such as averaged strains, partly aiming at avoiding locking problems.

3.1. Finite rotations

One of the central issues in the development of non-linear three-dimensional beam elements is the parameterization of the orthogonal matrix \( R \) introduced in the previous subsection, cf. [16]. The approach adopted in this paper is based on the use of the rotational vector

\[
\Psi = [\Psi_1, \Psi_2, \Psi_3]; \quad R = \exp(\tilde{\Psi})
\]

where the tilde denotes the skew-symmetric matrix obtained with the components of the corresponding vector. Admissible variations \( \delta R \) can then be computed according to:

\[
\delta R = \frac{d}{dt} [R]_{t=0} = \frac{d}{dt}(\exp(\delta \tilde{\Theta}))_{t=0} = \delta \tilde{\Theta} R
\]

\[
= \frac{d}{dt}(R \exp(\delta \tilde{\Theta}))_{t=0} = R \delta \tilde{\Theta}
\]

using either spatial \( \delta \Theta \) or material \( \delta \Theta \) angular variations. Note that both spatial and material components of the angular variations are independent of the technique used to parameterize \( R \). In order to obtain a consistent parameterization, both \( \delta \Theta \) and \( \delta \Theta \) are projected onto the chosen parameter space:

\[
\delta \Theta = T_s(\Psi) \delta \Psi, \quad \delta \tilde{\Theta} = T_m(\Psi) \delta \Psi
\]

Explicit expressions for the linear operators \( T_s \) and \( T_m \) can be found in [2]. These expressions were obtained and their relations verified by symbolic manipulations. Following this operation, the variation of the adopted strain measures will be computed in terms of the variables of the rotational vector \( \delta \Psi \).

3.2. Versions of space beam elements

Symbolic code for the derivation of a space beam element is given in Fig. 8. The element is here based on a total displacement formulation, where 12 degrees of freedom are considered, \( u_{kl} \) and \( \theta_{kl} \) where \( k = 1, 2, 3 \) refers to the three axes, and \( l = i, j \) are the two element nodes. All the quantities are linearly interpolated between the nodal values. The symbolic quantities \( a_1 \), \( a_2 \), \( a_3 \) are related to the interpolation of the rotational quantities. All strain components are evaluated as element averages at the midpoint of the element, \( x_1 = l/2 \). The internal force and tangential stiffness expressions for the element are obtained by differentiating the strain energy expression, \( F \tilde{A} \), with respect to the twelve degrees of freedom, giving a vector and a matrix. Note that the straightforward procedure leading to the formulation of the element is enabled by the adopted parameterization of the finite rotations, which operates entirely in the linear spaces of vectors \( \mathbf{r} \) and \( \Psi \).

With very similar symbolic procedures, four different elements are derived in [2]. These are based on total displacement and co-rotational viewpoints, and included different degrees of strain assumptions, according to:

- `b3dt`—total displacement form, Timoshenko strains;
- `b3ds`—total displacement form, simplified version of Timoshenko strains;
- `b3cl`—co-rotational form, linear strains;
- `b3cs`—co-rotational form, shallow arch strains.

An example used for numerical verification of the differences between the formulations is shown in Fig. 9, giving the results from analyses of the problem by 16-element models.

It is seen from the figure that the results agree very well between the element types, up to deformations where the singularity effects in the rotation representation start influencing the results. A small difference between the ‘t’ and ‘c’ elements is due to a difference in modelling of the shear behaviour.

It should be noted here that in the definition of the ‘b3ds’ element, the simplification relates to some higher order derivatives \( T_m \) and \( T_s \). If the symbolic
Fig. 8. Symbolic evaluation of space beam element in Maple, [10]. The five subsections evaluate expressions for displacements, their differentials, rotation parameters, strain components and energy terms.

\[
\begin{align*}
Ni &= 1-x/1; \quad Nj := x/1; \\
u1x &= Ni*u1i + Nj*u1j; \quad u2x := Ni*u2i + Nj*u2j; \quad u3x := Ni*u3i + Nj*u3j; \\
t1x &= Ni*t1i + Nj*t1j; \quad t2x := Ni*t2i + Nj*t2j; \quad t3x := Ni*t3i + Nj*t3j; \\
# du1 := diff(u1x, x); \quad du2 := diff(u2x, x); \quad du3 := diff(u3x, x); \\
dt1 := diff(t1x, x); \quad dt2 := diff(t2x, x); \quad dt3 := diff(t3x, x); \\
# al := (t1x^2 + t2x^2 + t3x^2)^1/2; \\
a := \sin(al)/al; \quad a1 := (1-a)/a \cdot 2; \\
a2 := 1/2*(\sin(al/2)/(a1/2))^2; \\
# x1 := 1/2*1; \\
ex0 := -(t1x^2 + t2x^2 + t3x^2)*(1 + du1) + (a1*t3x + a2*t3x)*du2 + (-a1*t2x + a2*t2x)*du3 - 1; \\
gy0 := -(a1*t3x + a2*t1x + t2x)*(1 + du1) + (1 - a2*(t1x^2 + t3x^2))*du2 + (a1*t1x + a2*t1x + t3x)*du3; \\
gz0 := (a1*t2x + a2*t1x + t3x)*(1 + du1) + (1 - a2*(t1x^2 + t2x^2))*du2 + (a1*t1x + a2*t2x + t3x)*du3; \\
\text{x} := x1; \quad \text{ex} := ex0^2; \quad \text{gy} := gy0^2; \quad \text{gz} := gz0^2; \\
kx := kx0^2; \quad ky := ky0^2; \quad kz := kz0^2; \\
\text{Fi} := 1/2*(EA*ex + GAY*gy + GAz*gz + EIz*ky + EIz*kz); \\
\text{T} := \text{grad}(\text{Fi}, [u1i, u2i, u3i, t1i, t2i, t3i, u1j, u2j, u3j, t1j, t2j, t3j]); \\
\text{Kt} := \text{hessian}(\text{Fi}, [u1i, u2i, u3i, t1i, t2i, t3i, u1j, u2j, u3j, t1j, t2j, t3j]);
\end{align*}
\]

Fig. 9. Test example for finite rotation space beam elements.
software is not used, these simplifications are necessary, due to the tedious algebra involved in the derivation of the tangent stiffness.

However, the use of a symbolic software tool makes such simplifications unnecessary, as in the case of the 'b3dtt' element, defined in Fig. 4.

4. Co-rotational framework and linear elements

The discussion an experiments presented in the previous sections have been the basis for the development of a library of finite elements, with controlled properties with respect to high accuracy non-linear behaviour. The basis for these elements is the co-rotational viewpoint, together with the description for finite space rotations. As much of this basis is independent of the specific element expressions being introduced, the basic routines are efficiently derived by symbolic manipulations. It is particularly noted that the co-rotational framework is identical for all element formulations with the same basic geometric description, leading to common expressions for the extraction of local element displacements from the global degrees of freedom.

So far, this method has been adopted for plane and space beam elements [1,2], and for triangular, faceted shell elements [7,17], the latter obtained from combinations of some basic in-plane elements with some plate bending elements.

As the computational treatment of the complex nonlinear and instability problems is very demanding, it is desirable that the local element formulations are as simple as possible. Therefore a study has been performed in order to find the most efficient final algorithmic expressions for some classes of simple, linear elements with controlled properties. Essentially two different methods for the formulation of element expressions can be used. These and an intermediate form were studied.

The used symbolic method is here based on extensive mathematical and numerical treatment, as opposed to the physical–mechanical reasoning leading to e.g. the development in [18]. It is thereby applicable to all element types, but can be assumed to be less phenomenological. This aspect is, however, minimized by the careful documentation and automatic transfer methods in the symbolic software.

The general context is stated in the following way: the symbolic software is used to define element assumptions, and to define useful expressions for element attributes, e.g. the stiffness matrix. The derivation is brought to certain stages, where the results are transferred as numerical algorithms to a FEM program, here represented by Matlab [19]. It is in particular noted that only basic derivation, and no problem solving, is done in the symbolic software.

4.1. Field consistence approach

Refs. [13,20] describe how a ‘field consistence’ approach can be used to develop highly accurate finite elements. In particular, the method is used to explain and remedy the origin of some locking phenomena.

The main idea in the discussed method is to ensure that all deformation modes of the element fulfill the governing field equations. For beam and plane stress elements, these are obtained from the Euler–Lagrange equations corresponding to any of the three commonly used variational principles. Solutions to these are matched to the chosen nodal quantities, leading to interpolation functions with consistent approximations to the relevant displacements. Using symbolic software, this is a rather easy task for beam elements, but it is considerably more complex for plane stress and plate elements.

As an example, a two-node Timoshenko beam element is developed in [13]. For this particular case, the virtual work (VW) principle, the Hellinger–Reissner (HR) principle, and the Hu–Washizu (HW) principle are studied. It is shown in [20] how the three principles, in general, will lead to identical results, when the assumed fields fulfill the stated field consistence requirements. Below, the VW case will be discussed, for fields fulfilling also the HR principle.

The considered deformation quantities in the beam are curvature and shear angle

\[ \kappa = \frac{\partial \psi}{\partial x}, \quad \gamma = \frac{\partial w}{\partial x} - \psi \]

where a beam along the x-axis is considered, and the displacements are represented by the transversal translation \( w \) and the rotation of the section normal \( \psi \).

Introducing sectional properties, the relevant internal forces can be written

\[ M = EI \kappa, \quad Q = kGA \gamma \]

from which the internal virtual work for a beam segment of length \( l \) can be stated as

\[ \delta V = \int_0^l (M \delta \kappa + Q \delta \gamma) \, dx \]

Introducing relevant expressions for the generalized strains, and creating the Euler–Lagrange equations, gives two coupled differential equations between the two displacement fields

\[ EI \frac{\partial^2 \psi}{\partial x^2} + kGA \left( \frac{\partial w}{\partial x} - \psi \right) = 0 \]
\( kGA \left( \frac{\partial^2 w}{\partial x^2} - \frac{\partial \psi}{\partial x} \right) = 0 \)

For this case, it is easy to find simultaneous solutions to the equations. Ref. [13] develops solutions according to
\[
w(x) = c_2 + c_3 x + c_4 x^2 + \frac{1}{6} \frac{kGA}{EI} (c_1 - c_3) x^3
\]
\[
\psi(x) = c_1 + 2 c_4 x + \frac{1}{2} \frac{kGA}{EI} (c_1 - c_3) x^2
\]
where the coupling coefficients are noted. By adopting these expressions to the nodal values at \( x=0 \) and \( x=l \),
\[
w(0) = w_i, \quad \psi(0) = \psi_i, \quad w(l) = w_j, \quad \psi(l) = \psi_j
\]
the \( c \) coefficients can be obtained, as expressions of the geometric properties of the element. This means that the interpolating functions in the element are dependent on the ratio
\[
\lambda = \frac{EI}{kGAf^2}
\]
and thus not equal for shear dominated and for bending dominated beam elements. From the obtained interpolation functions, the stiffness matrix can be obtained by rather simple algebraic treatment.

As shown in [13], the obtained element is free of locking, and also converges much more rapidly than the common two-node two-dimensional Timoshenko beam element; one element of this type gives more accurate results for a point-loaded cantilever than any division into less than 50 ordinary elements.

The symbolic software in this case made verification of the consistence of the assumed fields easier and more reliable. For more complicated element types, the symbolic tools are necessary in order to find correct assumptions, cf. [20,21].

4.2. Linear elements

This subsection describes some general aspects of the formulation of common linear finite elements and the symbolic treatment of those. The description is also valid for the elements formulated from the field consistence approach in the previous subsection, after introduction of the relevant interpolation functions.

The derivation of the element expressions has in all cases started from the nodal coordinates, and the chosen set of nodal displacement components. The interpolated displacements within the element are obtained as functions of the nodal displacements. Applying the relevant strain operator gives the strains as functions of the position within the element and the nodal unknowns. This, of course, assumes some regularity of the behaviour.

From the definition of the strains, the methods used diverge. The common ‘numerical’ method utilizes the virtual work relation, and produces a \( \mathbf{B} \) matrix, which corresponds to the strain operator and expresses the dependence of the strains on the unknowns. With a linear strain–stress relation, described by a matrix \( \mathbf{D} \), and assuming linear geometry, the virtual work expression gives the element stiffness matrix as the integral \( \int_{vol} \mathbf{B}^T \mathbf{DB} \), taken over the element volume. As, for all but the simplest of elements, the \( \mathbf{B} \) matrix is dependent on the position, the integral must be evaluated by a numerical integration of the matrix product. The evaluation of the matrix product must also be considered, as the matrices are of some size. Symbolic tools are here used to find expressions for the \( \mathbf{B} \) matrix components of chosen generality.

The considered ‘analytical’ method uses the strain expressions to give the corresponding stress expressions. Thereby, an expression for the strain energy density \( dU \) can be found, which can be integrated over the whole element volume, keeping the dependence on the geometry and the nodal unknowns. The linear equilibrium equation, represented by the element stiffness matrix, can be obtained from the second variation of this energy, with respect to the unknown displacements. In the used context, the result will be closed form expressions for the element stiffness matrix components, as functions of the geometry of the element, and the assumptions made. The expression for the integrated element strain energy is practically impossible to handle manually. Even with symbolic mathematical software, such as Maple [10] or Mathematica [22], the treatment is very demanding, and is not always possible for elements of some complexity.

The considered ‘semi-analytical’ method uses the strain expressions to give the corresponding stress expressions. The strain energy density \( dU \) is then numerically obtained as a function of the considered point. The second variation of this numerically integrated energy contribution, with respect to the unknown displacements, gives expressions for the contributions to the element stiffness matrix components, essentially corresponding to the product \( \mathbf{B}^T \mathbf{DB} \). The improvement, compared to the analytical method, is that simpler expressions are obtained at the intermediate stages of the derivation. The gain in efficiency is related to the removed matrix multiplications in the numerical algorithm.

As mentioned, the expressions for the element stiffness matrix components are algebraically complex. This can to some extent be remedied by different forms of simplifications, and combinations of polynomial terms. Several different methods for this are available.
in the symbolic softwares. These different simplifications will lead to different final evaluation procedures. A conclusion from the present study is that it is impossible to see in advance which specific form will lead to the simplest possible procedure. In order to improve these possibilities, sophisticated packages can be developed [8]. It is also possible to use symbolic software to develop auxiliary expressions, being important components of the matrices [23,24].

Not only the symbolic treatment, but also the formulation itself can be stated in different ways, with different results as a consequence. In particular, the area weight, which is the determinant of the Jacobian transformation matrix, can be included in several ways. This observation is dependent on the fact that the Jacobian is a multiplier in the area integration of the energy, but is a common denominator in all strain components. Considerable improvements in code efficiency can be gained by utilizing this observation.

As a conclusion, three basic methods for derivation of element stiffness matrices have been considered in the reported work. In short terms, these three basic methods can be described, according to:

A: analytical stresses, strain energy density and strain energy; analytical differential;
N: analytical B matrix; numerical matrix products and integration;
S: analytical stresses, and strain energy density; analytical differential, numerical integration;

where the analytical steps are performed in the symbolic derivation, and the numerical parts in the resulting FEM procedures.

The results from the three forms are transferred to the FEM program in the form of Fortran or C++ code. In the ‘A’ and ‘S’ forms, this means that computer language forms of each matrix component are transferred, whereas in the ‘N’ form similar expressions for the components of the B and D matrices are created, possibly as functions of the local coordinates. In the latter case, the produced expressions are included in a repetition bracket over the numerical integration points.

4.3. Derivation of the CST triangle

As a simple example, the derivation of the simple ‘CST’ plane stress triangle is given in Fig. 10. The derivation is in the most general form, where no assumptions are made concerning the placement of the coordinate system in the element, i.e. that the coordinates \((x_i, y_i)\) and the displacements \((u_i, v_i)\), are arbitrary for \(i = 1, 2, 3\). In the figure, the lines with three dots indicate obvious, slightly modified, repetitions of the previous lines. The result from the symbolic procedure is in the figure printed as Fortran statements for the vector and matrix components.

4.4. Resulting efficiency

Some results from the numerical tests of different resulting element algorithms are given in this section. The comparison here focuses on the different ways to describe the element in the numerical algorithm, and is not inherently related to the symbolic software. For all but the very simple CST element, the symbolic software is, however, needed to deliver alternatives to the traditional ‘N’ form.

As a first example, the efficiencies of the ‘A’ and ‘N’ forms of the element stiffness matrices were compared for the three simple elements in Fig. 11. The CST element is an iso-parametric three-node plane-stress element with constant strains, the ‘LST’ element a sub-parametric triangular plane-stress element with side nodes, and the ‘LMT’ element a three-node plate-bending element, cf. [25]. All the three element types were derived, assuming an arbitrary placement and orientation in the \((x, y)\) plane, as indicated by the figure. By this, no final coordinate transformations are needed in the resulting procedures.

The resulting computational demands for one evaluation of the stiffness matrices are given in Table 1. In the comparisons of efficiency, the computational work is measured in Matlab flops [19], corresponding to one mathematical operation, disregarding the possible difference in complexity between additions and multiplications.

The ‘A’ forms are more efficient for all these elements. The primary explanation to this fact is given by the right hand column, where the number of flops for the evaluation of matrix products \(B^TDB\) is given. Although this can be slightly reduced by utilizing the symmetry of the final expression, it is obvious that these multiplications will lead to inefficient formulations for these simple elements. In particular, element types with simple interpolations but rather many degrees of freedom, such as a four-node tetrahedron solid element, will suffer from this, as each integration point in the element will demand \(2n_d n_s (n_d + n_s)\) operations for the matrix product, with \(n_d\) and \(n_s\) the numbers of degrees of freedom and stress components, respectively. No consideration is here given to the possibilities to improve the performance by utilizing modern computer architectures in the evaluation.

4.5. Quadrilateral plane-stress element

Another test example studied the common four-node iso-parametric plane stress element. This element is rather easily formulated in a numerical ‘N’ form, where the B matrix, in coordinate form, can be for-
mally simply described. The main computational work in the evaluation of the stiffness matrix is therefore the matrix multiplications at each integration point.

The analytical form of the element stiffness can be developed from the basic geometry. The expressions will contain the Jacobian transformation of normalized to local coordinates. In the most general case, this area ratio can be written as:

\[ \det(J[L_1,L_2]) = \det(\text{invJ}[L_1,L_2])\]

\[ J[L_1,L_2] = \text{Inverse}[\text{invJ}[L_1,L_2]]\]

\[ d\text{1dx}[L_1,L_2] = J[L_1,L_2][[1,1]]\]

\[ d\text{dx}[L_1,L_2] = D[t[u[L_1,L_2],x,\text{Constants}->\text{vars}]\]

\[ \text{epsx}[L_1,L_2] = \text{Collect}[d\text{dx}[L_1,L_2],[L_1,L_2]]\]

\[ \text{sigx}[L_1,L_2] = \text{Simplify}[\text{epsx}[L_1,L_2]+nu \text{ epsy}[L_1,L_2]]\]

\[ d\text{U}[L_1,L_2] = \text{sigx}[L_1,L_2] \text{ epsx}[L_1,L_2]+\text{sigy}[L_1,L_2] \text{ epsy}[L_1,L_2]+\text{tauxy}[L_1,L_2] \text{ gamxy}[L_1,L_2]\]

\[ c\text{fkt} = (t Y)/(1 - nu^2)\]

\[ U\text{tot} = c\text{fkt} \text{ Integrate}[\text{Ad}[L_1,L_2] d\text{U}[L_1,L_2],\{L_1,0,1}\{L_2,0,1-L_1]\}

\[ \text{Do}[f[[i]] = \text{Simplify}[D[U\text{tot},\text{vars}[[i]]]],\{i,\text{nvars}]\]

\[ \text{Do}[k[[i, j]] = D[f[[i]],\text{vars}[[j]]],\{i,\text{nvars}\},\{j,\text{nvars}]\]

\[ \text{Do}[\text{Write}[	ext{strim}," K(","i","j",")=",k[[i,j]]],[j,\text{nvars}],\{i,\text{nvars}]\]

Fig. 10. Symbolic derivation of analytical form for CST element in Mathematica [22]. The result from the symbolic derivation is transferred as Fortran code to the FEM program.
ent cases are obtained, corresponding to different basic geometries—the simplest case being \( b_1 = b_2 = 0 \) for a rectangular element. Four of these cases correspond to situations where two nodes coincide, degenerating the quadrilateral to a triangle. The symbolic derivation of the element very clearly shows the effects of this degeneration, and also the effects of other special conditions in the element.

The eight cases need separate different code segments in the analytical formulation; these, however, are only concerned with a minor part of the procedure.

A semi-analytical formulation in this case develops expressions for the density of strain energy at an arbitrary point, and differentiates it analytically at chosen integration points, which are thus introduced already at the development of the computational procedure. It is here utilized that the determinant of the Jacobian is a constant at each integration point.

The resulting efficiencies for the developed procedures are shown in Table 2, referring to the common Cook test example [26]. Three different element meshes were considered, as indicated by the figures. For these meshes, and with three different ‘N’ accuracies, a four-point ‘S’ formulation, and an analytical formulation, the resulting deflections are shown. The obtained results show that a \( 2 \times 2 \) numerical integration deviates by about 2% from the analytical formulation, for the irregular mesh, although it is correct for the rectangular elements. It is also noted that the ‘S’ formulation with \( 2 \times 2 \) integration points completely agrees with the ‘N’ formulation with the same number of points.

The computational demands of the different formulations are shown in the last column. In the N forms,
the simplifications due to the symmetry of the stiffness matrix are utilized, thus giving a lower number of operations for the matrix operations than the one stated above. In the ‘A’ case, the number of operations quoted is the number valid for the completely irregular element; slightly fewer operations are needed for a rectangular element.

For both ‘N’ and ‘A’ cases, the resulting code is optimized with respect to expression evaluation. This replaced all expression parts needed more than once, and optimised the evaluation of the symmetric $B^TDB$ products. In the S case, some simplifications are still possible at the operation count given in the table.

5. Conclusions

The paper has, in rather general terms, discussed the improved possibilities to create finite elements when using symbolic software. The main aspects discussed can be summarized as:

- Symbolic derivations can give accurate elements, by allowing consistent formulations, not truncating minor effects. In certain cases, e.g. the three-dimensional beams, simplifying assumptions needed in order to make the expressions tractable, can be avoided.
- A new element (variation) can be derived in a short time.
- The derivation is well documented.
- Reduced/selective integrations can be included already in the derivation of an element.
- Limitations are clearly indicated by the algebraic process.
- The final FEM code for, at least simple, elements can be efficient.
- Analytical expressions can be obtained also for non-linear elements.

Several of these conclusions agree with those in [18], where for efficiency a shell element has been fully analytically derived, leading to very efficient program code [17].

With experience from two different symbolic softwares, Maple [10], and Mathematica [22], it can be concluded that they are functionally equivalent, but rather different with respect to user environment. They are, at least in principle, very good tools for the needed symbolic calculations. They can thus handle and treat needed expressions of often more than 1000 terms, but run into problems, when the problem sizes reach certain limits [8]. It is obvious from experiences that manual guidance, or sophisticated added packages as in [8], improves performance for both programs. The computer language codes produced from the two programs are also different, as Maple can produce more optimized code; the price to pay for this is a code that is not easily readable.

As a final comment, the symbolic software tools give many new possibilities for the development of computational mechanical formulations. As, however, these tools give a large number of possible formulations, and also several ways to perform the derivation of these formulations, much work remains in order to find generally applicable procedures for the derivation and documentation of finite element procedures.

References

[15] Simo JC. A finite strain beam formulation. The three-


