COMPDYN 2011
III ECCOMAS Thematic Conference on
Computational Methods in Structural Dynamics and Earthquake Engineering
M. Papadrakakis, M. Fragiadakis, V. Plevris (eds.)
Corfu, Greece, 25–28 May 2011

ERROR ESTIMATION AND IMPROVEMENT OF CONVERGENCE OF A TIME STEPPING ALGORITHM BASED ON A FINITE ELEMENT APPROACH IN TIME

Dirk Ostermann[1]

[1] Klosterhofstraße 28, 69469 Weinheim, Germany
www.dirk-ostermann.de
e-mail: kontakt@dirk-ostermann.de

Keywords: Finite Element Method, Betsch-Method, Time Stepping Algorithm, Convergence.

Abstract. *In seismic engineering either a modal analysis (based on the response spectra method) or a time history analysis may be performed. The response spectra method usually over-estimates the dynamic reactions of the structure. Therefore, the time history analysis is more appropriate if realistic results shall be obtained. In time history analysis the convergence of the times stepping algorithm that is used is of key importance.*

In the current paper an error estimation is performed for a time stepping algorithm based on a Finite Element approach in time (the so-called Betsch method, see [1] and [2]). The convergence rate of the time stepping algorithm is derived analytically and proofed numerically for different polynomial degrees of the shape functions. In order to increase the convergence rate a 3-step algorithm according to Tarnow [6] is implemented into the Betsch method. This allows high accurate results even when large time steps and only linear shape functions are used.

1 INTRODUCTION

The Finite Element Method is an approximation method. Hereby the accuracy of the results depends on the discretization (i.e. the number of elements used) and the quality of the elements (i.e. the polynomial degree of the shape functions). When the accuracy of the results is insufficient, the user has two methods to improve the accuracy: He can either increase the number of elements (h-method) or he can use elements with shape functions of higher polynomial degree (p-method).

The Finite Element method is not limited to a spacial discretization. It is also possible to derive a time stepping algorithm that is based on a Finite Element Approach in time, see [2]. Then the h-method means smaller time steps. The p-method is often not possible, because in most commercial Finite Element programs only very few time stepping algorithms are implemented and time stepping algorithms based on approaches of high polynomial degree are rarely available.

In the current contribution a third possibility to increase the accuracy of the solution is introduced: A 3-step algorithm according to Tarnow [6] is described. Hereby the step size is chosen in a way, that the errors of first and second order are eliminated and the method has the same order of convergence as if elements with quadratic shape functions were used.

2 ABREVIATIONS AND NOMENCLATURE

e	Error
h_n	Time step size
$\mathbf{p}$	Generalized momentum
$\mathbf{q}$	Generalized displacement
t	Time
$\mathbf{F}$	Force
H	Hamiltonian function of total energy
M_I	Shape function
$\mathbf{M}$	Mass matrix
α	Local coordinate $0 \leq \alpha \leq 1$

3 TIME STEPPING ALGORITHM

The time stepping algorithm that is presented in this section goes back to an idea of Betsch and is described in detail in [1] and [2]. In this section a brief summary of the Betsch method is given.

The basis for the time stepping algorithm is formed by Hamilton's equations of motion:

$$\begin{aligned} \frac{\partial H}{\partial \mathbf{p}} - \dot{\mathbf{q}} &= \mathbf{0} \\ \frac{\partial H}{\partial \mathbf{q}} + \dot{\mathbf{p}} &= \mathbf{0} \end{aligned} \tag{1}$$

These equations can be transferred into the variational form either by multiplication with test functions and integration over the time domain or by formal variation of the Lagrangian function. Both methods lead to

$$\int_0^T \left(\frac{\partial H}{\partial \mathbf{p}} - \dot{\mathbf{q}} \right) \cdot \delta\mathbf{p} - \left(\frac{\partial H}{\partial \mathbf{q}} + \dot{\mathbf{p}} \right) \cdot \delta\mathbf{q} \quad dt = 0 \tag{2}$$

Now the time domain is subdivided into time steps of size $h_n=t_{n+1} - t_n$ and the abbreviation $\alpha=(t-t_n)/h_n$ is introduced. The local coordinate α is 0 at the beginning of each time step and 1 at the end of each time step. The generalized coordinate **q** and the momentum **p** are interpolated with the functions M_I of polynomial degree k, which are continuous at the element borders:

$$\begin{aligned} \mathbf{q}^h(\alpha) &= \sum_{I=1}^{k+1} M_I(\alpha)\, \mathbf{q}_I \\ \mathbf{p}^h(\alpha) &= \sum_{I=1}^{k+1} M_I(\alpha)\, \mathbf{p}_I \end{aligned} \tag{3}$$

The same is done for the virtual generalized coordinate $\delta\mathbf{q}$ and the virtual generalized momentum $\delta\mathbf{p}$. Hereby shape functions of polynomial degree (k-1) are used, which are discontinuous at the element borders:

$$\begin{aligned} \delta\mathbf{q}^h(\alpha) &= \sum_{I=1}^{k} \tilde{M}_I(\alpha)\, \delta\mathbf{q}_I \\ \delta\mathbf{p}^h(\alpha) &= \sum_{I=1}^{k} \tilde{M}_I(\alpha)\, \delta\mathbf{p}_I \end{aligned} \tag{4}$$

The time derivatives of **q** and **p** can be expressed by the derivatives of the shape functions M_I:

$$\begin{aligned} \dot{\mathbf{q}}(\alpha) &= \frac{1}{h_n} \sum_{I=1}^{k} \frac{dM_I(\alpha)}{d\alpha}\, \mathbf{q}_I = \sum_{I=1}^{k} \tilde{M}_I(\alpha) \cdot \tilde{\mathbf{q}}_I \\ \dot{\mathbf{p}}(\alpha) &= \frac{1}{h_n} \sum_{I=1}^{k} \frac{dM_I(\alpha)}{d\alpha}\, \mathbf{p}_I = \sum_{I=1}^{k} \tilde{M}_I(\alpha) \cdot \tilde{\mathbf{p}}_I \end{aligned} \tag{5}$$

As the polynomial degree decreases by derivation for the time derivatives of **q** and **p** the virtual functions $\tilde{M}$ of polynomial degree (k-1) can be used. Inserting this into the variational form of Hamilton's equations leads to

$$\begin{aligned}
&\sum_{I=1}^{k} \delta\mathbf{p}_I{}^T \left(\sum_{J=1}^{k} \int_0^1 \tilde{M}_J \tilde{M}_I d\alpha \; \tilde{\mathbf{q}}_J - h_n \sum_{J=1}^{k+1} \mathbf{M}^{-1} \int_0^1 M_J \tilde{M}_I d\alpha \; \mathbf{p}_J \right) = 0 \\
&\sum_{I=1}^{k} \delta\mathbf{q}_I{}^T \left(\sum_{J=1}^{k} \int_0^1 \tilde{M}_J \tilde{M}_I d\alpha \; \tilde{\mathbf{p}}_J + h_n \int_0^1 \left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) \tilde{M}_I \; d\alpha \right) = 0
\end{aligned} \tag{6}$$

For linear shape functions M_I one obtains a residual equation that can be solved for the unknown vector $\mathbf{q}_{n+1}$:

$$\mathbf{R} \;=\; \frac{2}{h_n}\mathbf{M}\;\left(\mathbf{q}_{n+1} - \mathbf{q}_n\right) \;-\; 2\mathbf{p}_n \;+\; h_n \int_0^1 \left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) d\alpha \;=\; \mathbf{0} \tag{7}$$

For quadratic shape functions M_I one obtains

$$\mathbf{R} = \begin{bmatrix} \dfrac{1}{h_n}\mathbf{M}\left(-\mathbf{q}_n + \mathbf{q}_{n+1}\right) - \;\mathbf{p}_n + \; h_n \displaystyle\int_0^1 \left(1-\alpha\right)\left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) d\alpha \\ \dfrac{1}{h_n}\mathbf{M}\left(5\mathbf{q}_n - 8\mathbf{q}_{n+\frac{1}{2}} + 3\mathbf{q}_{n+1}\right) + \;\mathbf{p}_n + \; h_n \displaystyle\int_0^1 \alpha\left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) \; d\alpha \end{bmatrix} = \mathbf{0} \tag{8}$$

And for cubic shape functions one obtains

$$\mathbf{R} = \begin{bmatrix} \dfrac{1}{h_n}\mathbf{M}\left(-\dfrac{5}{2}\mathbf{q}_n + \dfrac{3}{2}\mathbf{q}_{n+\frac{1}{3}} + \dfrac{3}{2}\mathbf{q}_{n+\frac{2}{3}} - \dfrac{1}{2}\mathbf{q}_{n+1}\right) - \mathbf{p}_n & + h_n \displaystyle\int_0^1 \left(2\alpha^2 - 3\alpha + 1\right) \left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) d\alpha \\ \dfrac{1}{h_n}\mathbf{M}\left(3\mathbf{q}_n - 3\mathbf{q}_{n+\frac{1}{3}} - 3\mathbf{q}_{n+\frac{2}{3}} + 3\mathbf{q}_{n+1}\right) & + h_n \displaystyle\int_0^1 \left(-4\alpha^2 + 4\alpha\right) \left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) \; d\alpha \\ \dfrac{1}{h_n}\mathbf{M}\left(-7\mathbf{q}_n + 15\mathbf{q}_{n+\frac{1}{3}} - 12\mathbf{q}_{n+\frac{2}{3}} + 4\mathbf{q}_{n+1}\right) - \mathbf{p}_n & + h_n \displaystyle\int_0^1 \left(2\alpha^2 - \alpha\right) \left(\mathbf{F}_{\text{int}} - \mathbf{F}_{ext}\right) \; d\alpha \end{bmatrix} = \mathbf{0} \tag{9}$$

The procedure for shape functions of higher polynomial degree is described in [5].

4 THEORETICAL DERIVATION OF THE CONVERGENCY RATE

The Betsch algorithm described in section 3 can only be an approximation of an exact solution **q***(t). The exact solution **q***(t) can be developed into a Fourier series:

$$\begin{aligned}\mathbf{q}^*(t) &= \mathbf{q}^*(t_n) + \frac{\mathbf{q}^{*\prime}(t_n)}{1!}(t-t_n) + \frac{\mathbf{q}^{*\prime\prime}(t_0)}{2!}(t-t_n)^2 + \ldots \\ &= \sum_{i=0}^{\infty} \mathbf{a}_i (t-t_n)^i \end{aligned} \tag{10}$$

$$\text{with} \quad \mathbf{a}_i = \frac{\mathbf{q}^{*(i)}(t_n)}{i!}$$

The solution $\mathbf{q}^h(t)$ is approximated by the shape functions M_I of polynomial degree k as described in section 3.

$$\mathbf{q}^h(t) = \sum_{I=1}^{k+1} M_I \ \mathbf{q}_I \tag{11}$$

When both expressions are compared it is found that the exact solution **q***(t) contains all terms with polynomial degree i ≥ 0. However, the approximation $\mathbf{q}^h(t)$ does only contain terms with polynomial degree $0 \le I \le k$.

Therefore the difference between the exact solution **q***(t) and the approximation $\mathbf{q}^h(t)$ is of order $(t-t_n)^{k+1}$:

$$\left\| \mathbf{q}^*(t) - \mathbf{q}^h(t) \right\| \ \sim \ (t-t_0)^{k+1} \tag{12}$$

Hereby the term $(t-t_0)$ is limited by the time step size h_n.

The error e can also be written as a product of an (unknown) coefficient C and the time step size hn with exponent (k+1):

$$e \ = \ \left\| \mathbf{q}^*(t) - \mathbf{q}^h(t) \right\| \ = \ C \cdot h_n^{k+1} \tag{13}$$

Normally the error is displayed in a diagram with double-logarithmic axes. Then the equation describes a straight line with slope (k+1):

$$\log e = \log C + (k+1) \cdot \log h_n \tag{14}$$

However, when the error is evaluated only in the nodes (i.e. at the end of each time step and not within each time step), the slope of the convergence line is (2k) instead of (k+1). A detailed proof can be found in [4].

5 NUMERICAL DERIVATION OF THE CONVERGENCY RATE

The convergence rate can also be calculated numerically. Therefore the "accurate" solution is calculated with the algorithm explained in section 3 with a very small size of time steps. Then the error is negligibly small, so that this displacement-time process may be used as a reference solution $\mathbf{q}^*(t)$.

The diagram in Figure 1 shows the displacement-time process for the reference solution $\mathbf{q}^*$ (continuous line) and for an approximation $\mathbf{q}^h$ with time step size h_n=0.5s (dashed line).

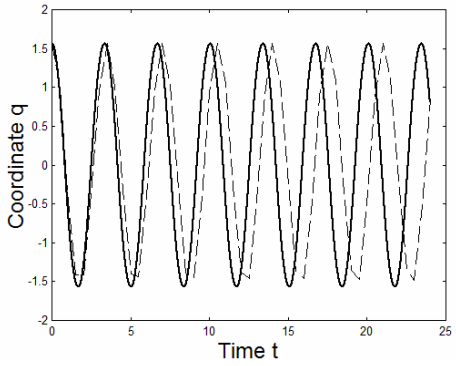

Figure 1: Displacement time process for reference solution $\mathbf{q}^*$ (continuous line) and for the approximation $\mathbf{q}^h$ (dashed line)

During the first seconds both solutions are almost identical. But with the time both solutions are drifting apart. After about 24s the reference solution $\mathbf{q}^*$ and the approximation $\mathbf{q}^h$ are out of phase.

The error can be calculated as the sum of the differences between the reference solution $\mathbf{q}^*$ and the approximation $\mathbf{q}^h$ over the investigated time domain:

$$\| e \|_{L2} = \left(\sum_i \left(q_i^h - q_i^* \right)^2 \cdot \Delta t_i \right)^{\frac{1}{2}} \tag{15}$$

Hereby the time interval Δt can either be equal to the time step size h_n (when the error is evaluated only in the nodes, i.e. at the end of each time step) or the time interval Δt can be less than the time step size h_n (when the error is evaluated also within each time step).

The couple of time step size h_n and error e gives one particular point of the convergence line in Figure 2. When the displacement-time process is calculated with other time step sizes and the error is evaluated for each of these calculations more points of the convergence line

are obtained and a trend of the convergence line can be found. This has already been proofed in [1].

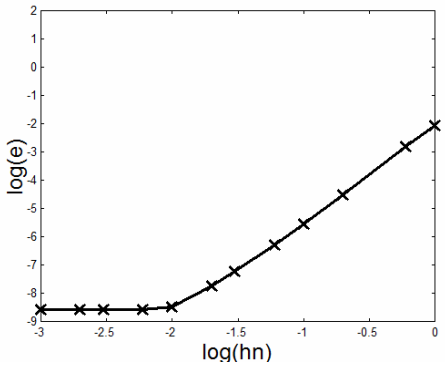

Figure 2: Convergence line

For very small time step sizes (in this particular case $h_n < 0.01s$) the convergence line tends to the horizontal. Here the accuracy of the computer is reached because the calculations are performed only with a fixed number of digits. A further reduction of the time step size does not result in a more accurate solution.

5.1 Linear Shape Functions

For linear shape functions one obtains the convergence lines displayed in Figure 3. The left diagram is obtained when the error is evaluated only in the nodes (i.e. at the end of each time step). The slope of the convergence line is (2k)=2.

The right diagram is obtained when the error is evaluated also within the time steps. Hereby the displacement within the time steps is approximated with the shape functions M_I. Then the slope of the convergence line is (k+1)=2. This has already been proofed in [1]. Both findings confirm the theoretical considerations from section 4.

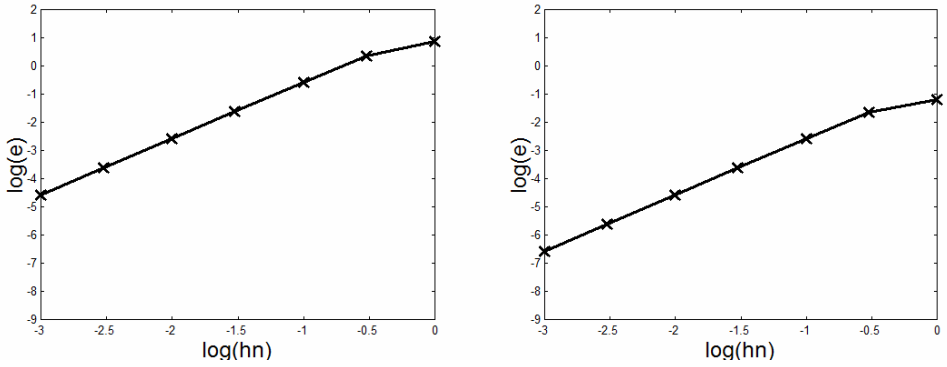

Figure 3: Convergence for linear shape functions
(left diagram: error evaluation only in the nodes, right diagram: error evaluation also between the nodes)

5.2 Quadratic Shape Functions

For quadratic shape functions one obtains the convergence lines displayed in Figure 4. Again, the left diagram shows the convergence line when the error is evaluated only in the nodes. This gives a slope of the convergence line of (2k)=4.

The right diagram shows the convergence line when the error is evaluated also between the nodes. In this case, the slope of the convergence line is (k+1)=3. The lower slope of the convergence line (3 instead of 4) is only due to a different definition of the error evaluation. Of course, the results in the nodes are of the same accuracy in both cases.

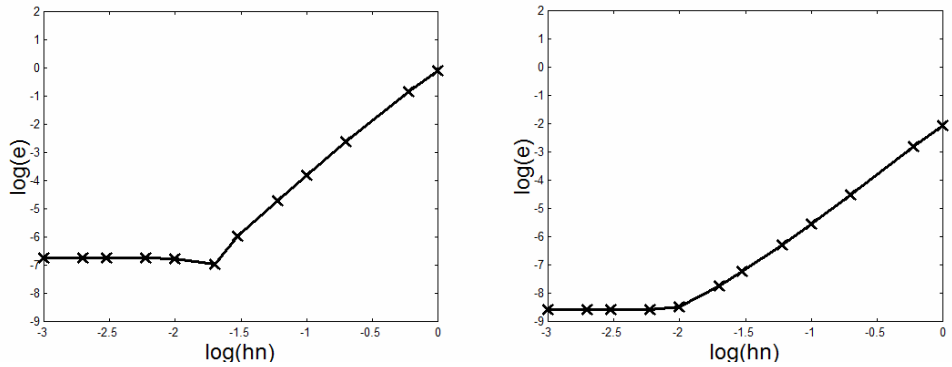

Figure 4: Convergence for quadratic shape functions
(left diagram: error evaluation only in the nodes, right diagram: error evaluation also between the nodes)

5.3 Cubic Shape Functions

The convergence lines for cubic shape functions are displayed in Figure 5. In the left diagram (evaluation of the error only in the nodes) the slope of the convergence line is (2k)=6 while the slope of the convergence line in the right diagram (evaluation of the error also between the nodes) is (k+1)=4.

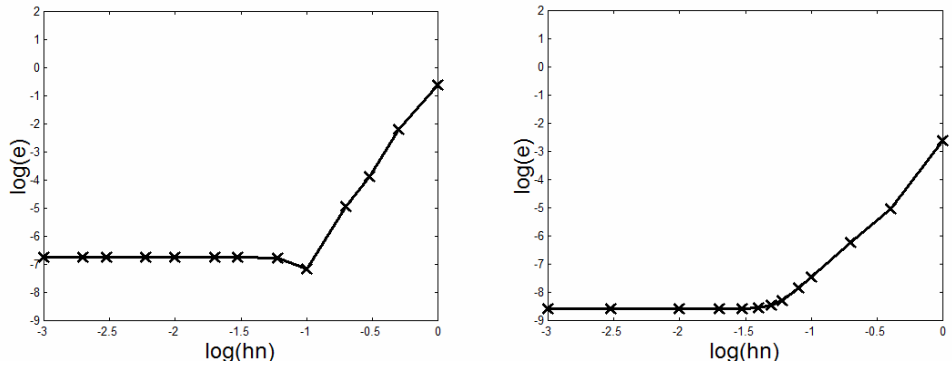

Figure 5: Convergence for cubic shape functions
(left diagram: error evaluation only in the nodes, right diagram: error evaluation also between the nodes)

6 3-STEP-METHOD FOR IMPROVEMENT OF THE CONVERGENCY RATE

As seen in the last section, it is high effort to obtain relatively accurate results. Either the time step size has to be reduced significantly (leading to high computational costs) or the polynomial degree of the shape functions has to be increased (often these high sophisticated algorithms are not available). Therefore a method would be desirable that gives high accurate results with relatively large time steps, although using linear shape functions. This goal can be reached by implementing the Tarnow method (see [6] and [7]) into the above described time stepping algorithm of Betsch. The procedure of the 3-step algorithm according to Tarnow is as follows:

- The values at time t_n are known from the previous time step. The values at time t_{n+1} shall be calculated. The Tarnow method does not calculate these values directly. Instead, the values at time $t_{n+\alpha}$ are calculated in a first step. The values are not accurate, they include an error of second order, third order and higher order because linear shape functions are used.
- In a second step, the values at time $t_{n+1-\alpha}$ are calculated based on the results at time $t_{n+\alpha}$. Again, the results include an error of second order, third order and higher order.
- In a third step, the values at time t_{n+1} are calculated based on the results at time $t_{n+1-\alpha}$. In this step the errors are again of second order, third order and higher order.

The parameter α can now be chosen in a way that the errors of second order and third order are compensated in the intermediate steps and only an error of forth and higher order remains. This is the case when the parameter α is chosen to 1.3512. A detailed proof of this method can be found in [6].

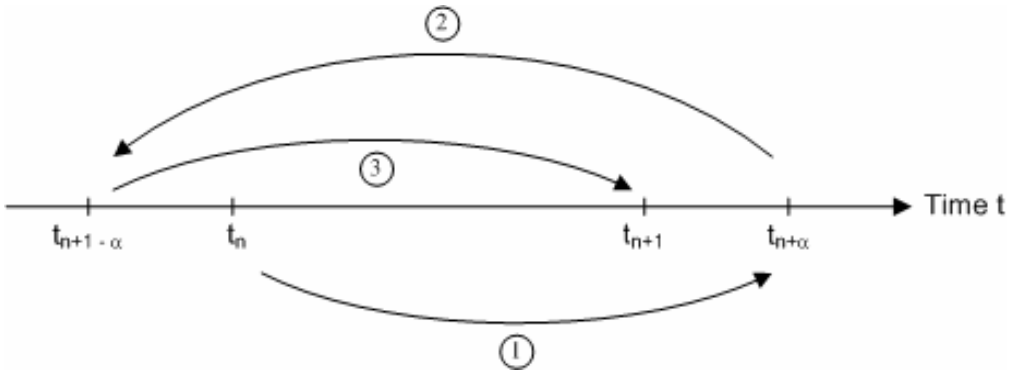

Figure 6: 3-step algorithm according to Tarnow

The above described Tarnow method has now been implemented into the time stepping algorithm of Betsch. Figure 7 shows the results when the Betsch algorithm is used with linear shape functions and the 3-step algorithm. The results are of 4th order accurate, therefore the slope of the convergence line is 4. So the same accuracy is reached as for a 1-step algorithm with quadratic shape functions.

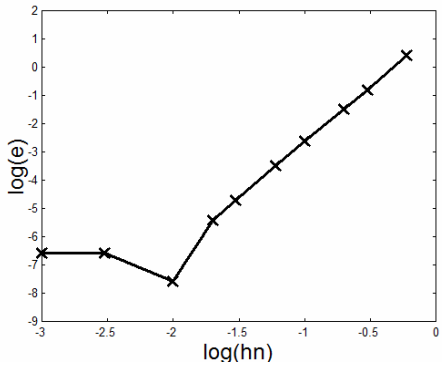

Figure 7: Convergence for linear shape functions with 3-step algorithm

7 SUMMARY

The convergence rate of the Betsch method depends on the polynomial degree of the shape functions. For linear shape functions a convergence rate of 2 was found. When the 3 step algorithm according to Tarnow is implemented into the Betsch method, the convergence rate is increased from 2 to 4. This allows high quality results, even when only linear shape functions and large time steps are used.

8 REFERENCES

[1] Betsch, P., Steinmann, P.: *Inherently Energy Conserving Time Finite Elements for Classical Mechanics*, Journal for Computational Physics, 160, 88-116, 2000

[2] Betsch, P.: *Computational Methods for Flexible Multibody Dynamics*, Habilitation Lehrstuhl für Technische Mechanik, Universität Kaiserslautern, UKL/LTM T02-02, 2002

[3] Henrici, P.: *Discrete Variable Methods in Ordinary Differential Equations*, Wiley, New York, 1962

[4] Hulme, B.L.: *One-Step Piecewise Polynomial Galerkin Methods for Initial Value Problems*, Mathematics of Computation, 26, 415-426, 1972

[5] Ostermann, D.: *Kontinuierliche und diskontinuierliche Galerkin-Methoden in der Elastodynamik und ihre Anwendung auf Probleme der Strukturmechanik*, Dissertation, TU Darmstadt, Shaker, 2005

[6] Tarnow, N.: *Energy and Momentum Conserving Algorithms for Hamiltonian Systems in the Nonlinear Dynamics of Solids*, Dissertation, Department of Mechanical Engineering, Standford University, 1993

[7] Tarnow, N., Simo, J.C.: *How to render second order accurate time-stepping algorithms fourth order accurate while retaining the stability and conservation properties*, Computational Methods in Applied Mechanical Engineering, 115, 233-252, 1994

[8] Wriggers, P.: *Nichtlineare Finite-Element-Methoden*, Springer, 2001