Comput Mech (2007) 40:383–398
DOI 10.1007/s00466-006-0117-y

ORIGINAL PAPER

# Finite element analysis of viscoelastic structures using Rosenbrock-type methods

**Stefan Hartmann · Jörg Wensch**

Received: 9 September 2005 / Accepted: 3 August 2006 / Published online: 17 October 2006

**Abstract** The consistent application of the space-time discretisation in the case of quasi-static structural problems based on constitutive equations of evolutionary type yields after the spatial discretisation by means of the finite element method a system of differential-algebraic equations. In this case the resulting system of differential-algebraic equations with the unknown nodal displacements and the evolution equations at all spatial quadrature points of the finite element discretisation are solved by means of a time-adaptive Rosenbrock-type methods leading to an iteration-less solution scheme in non-linear finite element analysis. The applicability of the method will be studied by means of a simple example of a viscoelastic structure.

**Keywords** Finite elements · Integration schemes · Differential-algebraic equations · Viscoelasticity · Rosenbrock

## 1 Introduction

Frequently, the implicit version of the finite element method, which is applied to quasi-static processes and constitutive equations of evolutionary type, has been considered as a local problem, namely the integration of the evolution equations by an implicit integration step, usually called stress algorithm, and a global problem, the fulfilment of the equilibrium conditions. In the work of [29] a further notion, namely the consistent tangent operator or the consistent linearisation, has been introduced in order to achieve a "quadratic" convergent global iteration scheme.

In a series of works the classical approach of a global and local problem has been related to algorithms developed in the field of Numerical Mathematics, because it is now known that the space-discretisation using finite elements leads to a system of differential-algebraic equations (DAE-system), see [4,5,9,10,21,39]. In this respect, we apply consistently the line-method, where in a first step the spatial and in a second step the temporal discretisation are carried out. The spatial discretisation is associated to the finite element specific application of introducing elements, particular shape functions as well as the transformation to local coordinates defined within a reference element and the numerical (spatial) integration by, for example, the Gauss quadrature. This step leads to a system of non-linear algebraic equations (algebraic part of the DAE-system) depending on the unknown nodal displacements and all internal variables at all spatial integration points (Gauss-points). Additionally, we have evolution equations, i.e. ordinary differential equations of first order, defining the evolution of the internal variables, which represent the material behaviour under consideration (differential part of the DAE-system). Both the non-linear algebraic equations and the system of ordinary differential equations denote a semi-explicit system of differential-algebraic equations of first order.

This holds in the case of rate-dependent constitutive model types like viscoelasticity and viscoplasticity formulated with ordinary differential equations. In

S. Hartmann (✉)
Institute of Mechanics, University of Kassel,
Mönchebergstrasse 7, 34109 Kassel, Germany
e-mail: hart@ifm.maschinenbau.uni-kassel.de

J. Wensch
Institute of Mathematics, University of Potsdam,
14415 Potsdam, Germany

the case of rate-independent elastoplasticity based on a yield function, different cases (elasticity or plasticity, which depends on the loading condition) define evolution equations as well (if the consistency condition is exploited). Which one is currently active depends on the solution itself. Frequently, a DAE-system is formulated in the case of plastic loading, which defines the evolution of the internal variables, where the algebraic constraint is given by the yield condition and an additional unknown is introduced , namely the plastic multiplier, guaranteeing the rate-independence.

In [4] and [10] diagonally implicit Runge–Kutta methods (DIRK methods) have been applied in order to carry out the time discretisation. This leads to the consecutive computation of $s$ coupled systems of non-linear equations in each integration step, where $s$ is the number of stages within a time-step. The unknowns of the coupled system of non-linear equations are the nodal displacements and all internal variables at all Gauss-points. The application of the Multilevel-Newton algorithm (MLNA) of [25] leads to the solution of a system of non-linear equations on Gauss-point level (for calculating the internal variables for given nodal displacements) and a linear system of equations (for computing the increment of the nodal displacements) on global level within a Newton–Raphson step. This method represents exactly the procedure of local and global iterations (commonly applied in implicit finite elements) and includes in passing the consistent tangent operator introduced in a hidden manner in the work of [29]. The difference of the MLNA to the application of the Newton–Raphson method is explained in [12] and a theoretical background in the case of displacement controlled processes is given in [11]. It has to be emphasised that the Backward–Euler method, which is frequently applied, is embedded in the DIRK/MLNA procedure as a particular case.

With the interpretation of the problems under considerations as DAE-systems, other algorithms than the DIRK-methods are applicable, [3,28,36–38]. The applicability of further time-adaptive methods, see, for example, [20], are worth being studied in the future. In this context, it has to be emphasised that the DAE-approach incorporates an essential by-product, namely time-adaptivity, which leads both to more accurate results (control of local integration error), and a stable procedure as well.

In this presentation, we investigate the behaviour of Rosenbrock-type methods, which offer the possibility of an iteration-less implicit procedure (see [8, Chap. IV.7], [31, Chap. 6.5.2] and the literature cited therein). Instead of solving non-linear systems of equations, one only has to compute a few systems of linear equations with the same coefficient matrix in each integration step from time $t_n$ to time $t_{n+1}$. In order to explain the effects on the problem under consideration, the article has the following structure: first of all, the simplest constitutive model of non-linear viscoelasticity is introduced in order to study the application of Rosenbrock methods (the extension to models in elastoplasticity and to a finite strain theory is, in a certain manner, straightforward see [10]. However, it has to be studied very carefully, for each new constitutive model) in the future. Then, the basic DAE-system is briefly derived, and, afterwards, the Rosenbrock method is recapped. The article concludes with an example.

The notation applied uses geometrical vectors, second order tensors $\mathbf{A}$ with bold-faced roman letters and calligraphic letters for fourth-order tensors $\mathcal{A}$. Furthermore, we have two types of matrices, namely global quantities, in the sense of the finite element method, using italic, bold-faced sans-serif letters $\boldsymbol{\mathsf{A}}$, and local element matrices using bold-faced sans-serif letters $\mathsf{\mathbf{A}}$.

## 2 Basic problem

In the following, the basic equations are briefly summarised. For more details, we refer to [4] and [12]. Here, a model of small strain viscoelasticity is applied because it is the simplest constitutive equation of evolutionary-type representing the structure of a large number of constitutive models. Afterwards the discretisation of the initial boundary-value problem is discussed applying the finite element method.

### 2.1 Constitutive equations

The generalisation of the three-parameter model of linear viscoelasticity to a non-linear test example reads as follows (cf. in view of viscoelasticity, [1] or [13], and the literature cited therein). First of all, the linearised Green strain tensor $\mathbf{E}$ is decomposed into an elastic part $\mathbf{E}_\mathrm{e}$ and a viscous part $\mathbf{E}_\mathrm{v}$. Furthermore, the stress state $\mathbf{T}$ is assumed to be additively decomposable into an equilibrium part $\mathbf{T}_\mathrm{eq}$ and an overstresses part $\mathbf{T}_\mathrm{ov}$. Both are defined by elasticity relations, where the overstresses depend on elastic strains and the equilibrium stress on the total strains. For the viscous strains an evolution equation exists, which depends on the process-dependent viscosity $\eta(\mathbf{T}_\mathrm{ov})$ leading to a model of non-linear viscoelasticity:

Strain decomposition: $$\mathbf{E} = \mathbf{E}_\mathrm{e} + \mathbf{E}_\mathrm{v}, \quad (1)$$

Stress decomposition: $$\mathbf{T} = \mathbf{T}_\mathrm{eq} + \mathbf{T}_\mathrm{ov}, \quad (2)$$

Equilibrium stress relation: $\mathbf{T}_{\mathrm{eq}} = K(\mathrm{tr}\,\mathbf{E})\mathbf{I} + 2G\mathbf{E}^D, \quad (3)$

Overstress relation: $\mathbf{T}_{\mathrm{ov}} = \mathbf{T}_{\mathrm{ov}}^D = 2\hat{G}(\mathbf{E}-\mathbf{E}_{\mathrm{v}})^D, \quad (4)$

Flow rule: $\dot{\mathbf{E}}_{\mathrm{v}} = \frac{2\hat{G}}{\eta}(\mathbf{E}-\mathbf{E}_{\mathrm{v}})^D, \quad (5)$

Non-linear viscosity: $\eta = \eta_0 \mathrm{e}^{-s_0\|\mathbf{T}_{\mathrm{ov}}^D\|}, \quad \eta > 0. \quad (6)$

In view of the process-dependent viscosity, see also [14] and [10]. $K$ and $G$ define the bulk and shear modulus of the equilibrium stress state and $\hat{G}$ the shear modulus of the overstresses. $\mathbf{I}$ symbolises the second order identity tensor, $\mathrm{tr}\,\mathbf{A} = a^i{}_i$ stands for the trace operator and $D$ denotes the deviator operator, $\mathbf{A}^D = \mathbf{A} - (1/3)(\mathrm{tr}\,\mathbf{A})\mathbf{I}$. $\eta_0$ and $s_0$ are further material parameters influencing the non-linear rate-dependence and the relaxation behaviour.

The applied test example of a constitutive model represents the structure of more general constitutive equations of the type

$$\mathbf{T} = \mathbf{h}(\mathbf{E},\mathbf{q}), \quad (7)$$

$$\dot{\mathbf{q}} = \mathbf{r}(\mathbf{E},\mathbf{q}), \quad \mathbf{q}(t_0) = \mathbf{q}_0, \quad (8)$$

where $\mathbf{q} \in \mathbb{R}^{n_{\mathrm{q}}}$ defines a set of internal variables describing the material behaviour and $\mathbf{h}(\mathbf{E},\mathbf{q})$ denotes an elasticity relation. Here, $\mathbf{q} = \{\mathbf{E}_{\mathrm{v}}\}$ with $n_{\mathrm{q}} = 6$ holds in the three-dimensional case.

## 2.2 Defining the DAE-system

As mentioned in Ellsiepen and Hartmann [4] and Hartmann [12], the equilibrium conditions in the small strain case are defined by

$$\boldsymbol{g}(t,\boldsymbol{u}(t),\boldsymbol{q}(t)) := \sum_{e=1}^{n_{\mathrm{e}}} \boldsymbol{Z}^{e\mathrm{T}} \Bigg\{ \sum_{j=1}^{n_\xi}\sum_{k=1}^{n_\eta}\sum_{l=1}^{n_\zeta} w_{jkl} \times \mathbf{B}^{e(jkl)\mathrm{T}}\,\mathbf{h}\left(\mathbf{E}^{e(jkl)}(t),\mathbf{q}^{e(jkl)}(t)\right) \times \det \mathbf{J}^{e(jkl)} \Bigg\} - \overline{\boldsymbol{p}}(t) = \boldsymbol{0} \quad (9)$$

which is the result of the spatial discretisation using standard finite elements. Here, we have the assemblage matrix $\boldsymbol{Z}^e \in \mathbb{R}^{n_{\mathrm{eu}}\times n_{\mathrm{u}}}$, where $n_{\mathrm{eu}}$ and $n_{\mathrm{u}}$ define the number of element nodal displacements and the number of unknown nodal displacements of the structure, respectively. This matrix, which is usually not programmed (it contains only zeros and a few ones), is used to assign local to global degrees of freedom. $w_{jkl}$ symbolises the product of the weighting factors which occur in the Gauss quadrature in a reference element, $w_{jkl} = w_j \times w_k \times w_l$. The indices $j = 1,\ldots,n_\xi$, $k = 1,\ldots,n_\eta$, and $l = 1,\ldots,n_\zeta$, denote the number of the Gauss-point, where $n_\xi$, $n_\eta$, and $n_\zeta$ are the maximum number of Gauss-points in the direction concerned.

$$\mathbf{E}^{e(jkl)}(t) = \mathbf{B}^{e(jkl)}\mathbf{u}^e = \mathbf{B}^{e(jkl)}\left\{\boldsymbol{Z}^e\boldsymbol{u}(t) + \overline{\boldsymbol{Z}}^e\overline{\boldsymbol{u}}(t)\right\} \quad (10)$$

defines the strain vector in element $e$ at Gauss-point $(jkl)$. $\boldsymbol{u}(t) \in \mathbb{R}^{n_{\mathrm{u}}}$ denotes the vector of unknown nodal displacements and $\overline{\boldsymbol{u}}(t) \in \mathbb{R}^{n_{\mathrm{p}}}$ the prescribed and known nodal displacements. $n_{\mathrm{p}}$ is the number of all prescribed displacement degrees of freedom. Accordingly, $n_{\mathrm{dof}} = n_{\mathrm{u}} + n_{\mathrm{p}}$ holds and $\overline{\boldsymbol{Z}}^e \in \mathbb{R}^{n_{\mathrm{eu}}\times n_{\mathrm{p}}}$ relates the known nodal displacements to the element degrees of freedom. $\mathbf{u}^e \in \mathbb{R}^{n_{\mathrm{eu}}}$ contains the element nodal displacements and $\mathbf{B}^{e(jkl)} := \mathbf{B}^e(\boldsymbol{\xi}_{jkl})$ symbolises the strain-displacement matrix at Gauss-point $\boldsymbol{\xi}_{jkl}$. The matrix $\mathbf{J}^{e(jkl)} := \mathbf{J}^e(\boldsymbol{\xi}_{jkl}) = \partial\boldsymbol{\chi}^e/\partial\boldsymbol{\xi}|_{\boldsymbol{\xi}=\boldsymbol{\xi}_{jkl}}$ represents the Jacobian of the coordinate transformation between local and global coordinates $\mathbf{x} = \chi^e(\boldsymbol{\xi})$. $\overline{\boldsymbol{p}}(t) \in \mathbb{R}^{n_{\mathrm{u}}}$ defines the sum of the given equivalent and concentrated nodal force vector. $\mathbf{T}^{e(jkl)} = \mathbf{h}\left(\mathbf{E}^{e(jkl)}(t),\mathbf{q}^{e(jkl)}(t)\right)$ denotes the stress state in element $e$ at Gauss-point $\boldsymbol{\xi}_{jkl}$, which is defined by the elasticity relation (7). The stress state depends also on the internal variables at the integration point under consideration, which is described in the following.

The internal variables are locally defined by the evolution equations (8),

$$\dot{\mathbf{q}}^{e(jkl)}(t) - \mathbf{r}(\mathbf{E}^{e(jkl)}(t),\mathbf{q}^{e(jkl)}(t)) = \mathbf{0}. \quad (11)$$

Formally, all these sets of internal variables and the evolution equations concerned could be assembled into a large vector $\boldsymbol{q} \in \mathbb{R}^{n_{\mathrm{Q}}}$,

$$\boldsymbol{q}(t) = \sum_{e,j,k,l} \boldsymbol{Z}_{\mathrm{q}}^{e(jkl)\mathrm{T}}\mathbf{q}^{e(jkl)}, \quad \boldsymbol{r}(t,\boldsymbol{u}(t),\boldsymbol{q}(t)) = \sum_{e,j,k,l} \boldsymbol{Z}_{\mathrm{q}}^{e(jkl)\mathrm{T}} \times \mathbf{r}(\mathbf{E}^{e(jkl)}(t),\mathbf{q}^{e(jkl)}(t)), \quad (12)$$

where the coincidence matrices $\boldsymbol{Z}_{\mathrm{q}}^{e(jkl)} \in \mathbb{R}^{n_{\mathrm{q}}\times n_{\mathrm{Q}}}$ are introduced. $n_{\mathrm{q}}$ is the number of internal variables at one integration point and $n_{\mathrm{Q}}$ all internal variables in the complete structure. The coincidence matrices have the property

$$\boldsymbol{Z}_{\mathrm{q}}^{e(jkl)}\boldsymbol{Z}_{\mathrm{q}}^{\hat{e}(\hat{i}\hat{j}\hat{k})\mathrm{T}} = \begin{cases} \mathbf{I}_{n_{\mathrm{q}}} & \text{if } e = \hat{e} \text{ and } i = \hat{\imath} \\ & \text{and } j = \hat{\jmath} \text{ and } k = \hat{k} \\ \mathbf{0} & \text{else} \end{cases} \quad (13)$$

yielding

$$\mathbf{q}^{e(jkl)}(t) = \boldsymbol{Z}_{\mathrm{q}}^{e(jkl)}\boldsymbol{q}(t), \quad \mathbf{q}^{e(jkl)} \in \mathbb{R}^{n_{\mathrm{q}}}. \quad (14)$$

Accordingly, we arrive at the system of ordinary differential equations

$$\dot{\boldsymbol{q}}(t) - \boldsymbol{r}(t, \boldsymbol{u}(t), \boldsymbol{q}(t)) = \boldsymbol{0}, \quad \boldsymbol{q}(t_0) = \boldsymbol{q}_0. \tag{15}$$

Both, Eqs. (9) and (15) represent a semi-explicit system of differential-algebraic equations with

$$\boldsymbol{F}(t, \boldsymbol{y}(t), \dot{\boldsymbol{y}}(t)) := \begin{Bmatrix} \boldsymbol{g}(t, \boldsymbol{u}(t), \boldsymbol{q}(t)) \\ \dot{\boldsymbol{q}}(t) - \boldsymbol{r}(t, \boldsymbol{u}(t), \boldsymbol{q}(t)) \end{Bmatrix} = \boldsymbol{0}, \quad \boldsymbol{F} \in \mathbb{R}^{n_\mathrm{u}+n_\mathrm{Q}}, \tag{16}$$

with

$$\boldsymbol{y}(t) := \begin{Bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{q}(t) \end{Bmatrix}, \quad \boldsymbol{u} \in \mathbb{R}^{n_\mathrm{u}}, \ \boldsymbol{q} \in \mathbb{R}^{n_\mathrm{Q}}, \tag{17}$$

and the initial conditions

$$\boldsymbol{y}(t_0) := \begin{Bmatrix} \boldsymbol{u}(t_0) \\ \boldsymbol{q}(t_0) \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{q}_0 \end{Bmatrix} =: \boldsymbol{y}_0. \tag{18}$$

Due to the fact that the displacements $\overline{\boldsymbol{u}}(t)$ and the external forces $\overline{\boldsymbol{p}}(t)$ are known functions of the time $t$, the DAE-system is non-autonomous.

### 2.3 Time discretisation using Rosenbrock methods

In the section above as well as in Ellsiepen and Hartmann [4] and Hartmann [10] it has been shown that the line method yields a system of differential-algebraic equations after the spatial discretisation. In these references, the system is solved by means of stiffly accurate diagonally implicit Runge–Kutta methods in combination with a Multilevel-Newton method, which yields the classical expressions of global and local (element) level. This procedure, which embraces the classical Backward–Euler scheme as a special case, is expensive because we have to carry out in each stage of a time step, i.e. load step from time $t_n$ to $t_{n+1}$, the computation of a coupled system of non-linear equations (Multilevel Newton method in each stage). Thus, we apply Rosenbrock-type methods which need merely the solution of one system of linear equations (with constant coefficient matrix during the time step) in each stage. Since these methods are rarely considered in the finite element literature, we recall some fundamental ideas so that the text becomes self-explanatory. To this end, we follow the representations of Hairer and Wanner [8, pp. 102 ff.] and Strehmel and Weiner [31, pp. 278 ff.].

Normally, one starts from an autonomous system of ordinary differential equations $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$ applying the integration step

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \sum_{i=1}^{s} b_i \boldsymbol{f}(\boldsymbol{Y}_{ni}) \tag{19}$$

in order to advance the approximate solution $\boldsymbol{y}_n$ at time $t_n$ to the next approximation $\boldsymbol{y}_{n+1}$ at time $t_{n+1} = t_n + \Delta t_n$. The stage quantities $\boldsymbol{Y}_{ni}$ ($i = 1, \ldots, s$) are given by

$$\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i} a_{ij} \boldsymbol{f}(\boldsymbol{Y}_{nj}), \quad i = 1, \ldots, s. \tag{20}$$

Rosenbrock methods work with the quantities (stage derivatives)

$$\dot{\boldsymbol{Y}}_{ni} = \boldsymbol{f}(\boldsymbol{Y}_{ni}), \quad i = 1, \ldots, s. \tag{21}$$

If the stage quantities (20) are inserted into the definition of the stage derivative (21), one arrives at a system of non-linear equations in each stage

$$\boldsymbol{F}(\dot{\boldsymbol{Y}}_{ni}) = \boldsymbol{0}, \quad i = 1, \ldots, s, \tag{22}$$

with

$$\boldsymbol{F}(\dot{\boldsymbol{Y}}_{ni}) := \dot{\boldsymbol{Y}}_{ni} - \boldsymbol{f}\left( \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} \dot{\boldsymbol{Y}}_{nj} + \Delta t_n \, a_{ii} \dot{\boldsymbol{Y}}_{ni} \right). \tag{23}$$

The fundamental idea of the Rosenbrock-type methods is the application of one Newton iteration step on Eq. (22) using as an initial choice a linear combination

$$\dot{\boldsymbol{Y}}_{ni}^{(0)} = -\frac{1}{a_{ii}} \sum_{j=1}^{i-1} \gamma_{ij} \dot{\boldsymbol{Y}}_{nj}. \tag{24}$$

for the stage derivative. Here, we have $\gamma_{ij} = a_{ij} - \alpha_{ij}$ and $\gamma_{ii} = a_{ii}$, i.e. new weighting factors must be introduced. The Newton iteration step of function (23) reads

$$\boldsymbol{F}\left(\dot{\boldsymbol{Y}}_{ni}^{(1)}\right) = \boldsymbol{F}\left(\dot{\boldsymbol{Y}}_{ni}^{(0)}\right) + \left.\frac{\mathrm{d}\boldsymbol{F}}{\mathrm{d}\dot{\boldsymbol{Y}}_{ni}}\right|_{\dot{\boldsymbol{Y}}_{ni}^{(0)}} \left\{ \dot{\boldsymbol{Y}}_{ni}^{(1)} - \dot{\boldsymbol{Y}}_{ni}^{(0)} \right\} = \boldsymbol{0}, \tag{25}$$

which leads to the more detailed representation

$$\left[ \boldsymbol{I} - \Delta t_n \, a_{ii} \left.\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{Y}_{ni}}\right|_{\boldsymbol{Y}_{ni}^{(0)}} \right] \left\{ \dot{\boldsymbol{Y}}_{ni}^{(1)} - \dot{\boldsymbol{Y}}_{ni}^{(0)} \right\} = -\dot{\boldsymbol{Y}}_{ni}^{(0)} + \boldsymbol{f}\left( \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} \dot{\boldsymbol{Y}}_{nj} + \Delta t_n \, a_{ii} \dot{\boldsymbol{Y}}_{ni}^{(0)} \right), \tag{26}$$

i.e. we arrive at the system of linear equations by inserting the initial choice (24),

$$\left[ \boldsymbol{I} - \Delta t_n \, a_{ii} \left.\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{Y}_{ni}}\right|_{\boldsymbol{Y}_{ni}^{(0)}} \right] \dot{\boldsymbol{Y}}_{ni}^{(1)} = \Delta t_n \left.\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{Y}_{ni}}\right|_{\boldsymbol{Y}_{ni}^{(0)}} \times \left\{ \sum_{j=1}^{i-1} \gamma_{ij} \dot{\boldsymbol{Y}}_{nj} \right\} + \boldsymbol{f}\left( \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} \alpha_{ij} \dot{\boldsymbol{Y}}_{nj} \right). \tag{27}$$

First of all, we redefine the stage derivative $\dot{\boldsymbol{Y}}_{ni} := \dot{\boldsymbol{Y}}_{ni}^{(1)}$ and exploit only the functional matrix at time $t_n$ in order to minimise the computational cost,

$$\boldsymbol{J}_n := \left.\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{Y}_{ni}}\right|_{\boldsymbol{y}_n}, \tag{28}$$

then the linear system reads

$$\left[\boldsymbol{I} - \Delta t_n\, a_{ii} \boldsymbol{J}_n\right] \dot{\boldsymbol{Y}}_{ni} = \Delta t_n\, \boldsymbol{J}_n \left\{\sum_{j=1}^{i-1} \gamma_{ij} \dot{\boldsymbol{Y}}_{nj}\right\} + \boldsymbol{f}\left(\boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} \alpha_{ij} \dot{\boldsymbol{Y}}_{nj}\right). \tag{29}$$

Note, that a Rosenbrock method is given by the stage coefficients $\alpha_{ij}, (1 \le j < i \le s)$ and $\gamma_{ij}, (1 \le j \le i \le s)$ and the weights $b_i, 1 \le i \le s$.

The extension to non-autonomous systems $\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y})$ is, in a certain way, straightforward. In this respect one defines an autonomous system

$$\dot{\tilde{\boldsymbol{y}}} = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{y}}) \quad \text{with} \quad \tilde{\boldsymbol{y}} = \begin{Bmatrix} t \\ \boldsymbol{y} \end{Bmatrix} \quad \text{and}$$
$$\dot{\tilde{\boldsymbol{y}}} = \begin{Bmatrix} \dot{t} \\ \dot{\boldsymbol{y}} \end{Bmatrix} = \begin{Bmatrix} 1 \\ \boldsymbol{f}(t, \boldsymbol{y}) \end{Bmatrix}. \tag{30}$$

Then, Eq. (29) reads after evaluating the second equation of (30)

$$\left[\boldsymbol{I} - \Delta t_n\, \gamma \boldsymbol{J}_n\right] \dot{\boldsymbol{Y}}_{ni} = \Delta t_n\, d_i \left.\frac{\partial \boldsymbol{f}}{\partial t}\right|_{(t_n, \boldsymbol{y}_n)} + \Delta t_n\, \boldsymbol{J}_n \left\{\sum_{j=1}^{i-1} \gamma_{ij} \dot{\boldsymbol{Y}}_{nj}\right\} + \boldsymbol{f}\left(T_{ni}, \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} \alpha_{ij} \dot{\boldsymbol{Y}}_{nj}\right) \tag{31}$$

with $d_i = \gamma + \sum_{j=1}^{i-1} \gamma_{ij}$, $T_{ni} = t_n + c_i \Delta t_n$ and $c_i = \sum_{j=1}^{i-1} \alpha_{ij}$. Usually, all coefficients $a_{ii}$, $i = 1, \ldots, s$, are chosen to have the same value, $a_{ii} = \gamma$. In order to circumvent the matrix-vector multiplication on the right-hand side, we define a new variable

$$\boldsymbol{B}_{ni} = \dot{\boldsymbol{Y}}_{ni} + \left\{\sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma} \dot{\boldsymbol{Y}}_{nj}\right\}, \tag{32}$$

so that, finally, (31) reads

$$\left[\boldsymbol{I} - \Delta t_n\, \gamma \boldsymbol{J}_n\right] \boldsymbol{B}_{ni} = \Delta t_n\, d_i \left.\frac{\partial \boldsymbol{f}}{\partial t}\right|_{(t_n, \boldsymbol{y}_n)} + \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma} \dot{\boldsymbol{Y}}_{nj} + \boldsymbol{f}\left(T_{ni}, \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} \alpha_{ij} \dot{\boldsymbol{Y}}_{nj}\right). \tag{33}$$

Note, that the system matrix $\boldsymbol{I} - \Delta t_n\, \gamma \boldsymbol{J}_n$ is the same for all stages.

The treatment of non-autonomous differential-algebraic equations, which results from the spatial discretisation in the finite element application, see Eq. (16),

$$\boldsymbol{0} = \boldsymbol{g}(t, \boldsymbol{u}, \boldsymbol{q}), \tag{34}$$
$$\dot{\boldsymbol{q}} = \boldsymbol{r}(t, \boldsymbol{u}, \boldsymbol{q}), \tag{35}$$

is carried out in two steps using the $\varepsilon$-embedding method. $\boldsymbol{u}$ are the unknown nodal displacements and $\boldsymbol{q}$ all internal variables at all spatial integration points. In the *first step*, Eq. (34) is modified by a perturbation

$$\varepsilon \dot{\boldsymbol{u}} = \boldsymbol{g}(t, \boldsymbol{u}, \boldsymbol{q}) \quad \text{or} \quad \dot{\boldsymbol{u}} = \frac{1}{\varepsilon} \boldsymbol{g}(t, \boldsymbol{u}, \boldsymbol{q}), \tag{36}$$

$0 < \varepsilon \lll 1$. Then, we define

$$\boldsymbol{y} := \begin{Bmatrix} \boldsymbol{u} \\ \boldsymbol{q} \end{Bmatrix} \quad \text{and} \quad \dot{\boldsymbol{y}} := \begin{Bmatrix} \dot{\boldsymbol{u}} \\ \dot{\boldsymbol{q}} \end{Bmatrix} = \begin{Bmatrix} \frac{1}{\varepsilon} \boldsymbol{g}(t, \boldsymbol{u}, \boldsymbol{q}) \\ \boldsymbol{r}(t, \boldsymbol{u}, \boldsymbol{q}) \end{Bmatrix} \tag{37}$$

and apply the integration formula (31) to Eqs. $(36)_2$ and (35) leading to a coupled system of linear equations

$$\begin{bmatrix} \boldsymbol{I} - \Delta t_n\, \gamma \frac{1}{\varepsilon} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} & -\Delta t_n\, \gamma \frac{1}{\varepsilon} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}} \\ -\Delta t_n\, \gamma \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} & \boldsymbol{I} - \Delta t_n\, \gamma \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \end{bmatrix}_{\boldsymbol{z}_n} \begin{Bmatrix} \dot{\boldsymbol{U}}_{ni} \\ \dot{\boldsymbol{Q}}_{ni} \end{Bmatrix}$$
$$= \Delta t_n\, d_i \begin{Bmatrix} \frac{1}{\varepsilon} \frac{\partial \boldsymbol{g}}{\partial t} \\ \frac{\partial \boldsymbol{r}}{\partial t} \end{Bmatrix}_{\boldsymbol{z}_n} + \Delta t_n \begin{bmatrix} \frac{1}{\varepsilon} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} & \frac{1}{\varepsilon} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}} \\ \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \end{bmatrix}_{\boldsymbol{z}_n}$$
$$\times \begin{Bmatrix} \sum_{j=1}^{i-1} \gamma_{ij} \dot{\boldsymbol{U}}_{nj} \\ \sum_{j=1}^{i-1} \gamma_{ij} \dot{\boldsymbol{Q}}_{nj} \end{Bmatrix} + \begin{Bmatrix} \frac{1}{\varepsilon} \boldsymbol{g}(T_{ni}, \boldsymbol{S}_{ni}^u, \boldsymbol{S}_{ni}^q) \\ \boldsymbol{r}(T_{ni}, \boldsymbol{S}_{ni}^u, \boldsymbol{S}_{ni}^q) \end{Bmatrix}, \tag{38}$$

where use is made of the abbreviations $\boldsymbol{z}_n = \{t_n, \boldsymbol{u}_n, \boldsymbol{q}_n\}$ as well as

$$\boldsymbol{S}_{ni}^u = \boldsymbol{u}_n + \Delta t_n \sum_{j=1}^{i-1} \alpha_{ij} \dot{\boldsymbol{U}}_{nj} \quad \text{and}$$
$$\boldsymbol{S}_{ni}^q = \boldsymbol{q}_n + \Delta t_n \sum_{j=1}^{i-1} \alpha_{ij} \dot{\boldsymbol{Q}}_{nj}. \tag{39}$$

In the *second step* one multiplies the first equation of Eq. (38) with $\varepsilon$ and sends $\varepsilon$ towards zero, which leads to the final expression

$$\begin{bmatrix} -\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} & -\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}} \\ -\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} & \boldsymbol{I}-\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \end{bmatrix}_{\boldsymbol{z}_n} \begin{Bmatrix} \dot{\boldsymbol{U}}_{ni} \\ \dot{\boldsymbol{Q}}_{ni} \end{Bmatrix}$$
$$= \Delta t_n\, d_i \begin{Bmatrix} \dfrac{\partial \boldsymbol{g}}{\partial t} \\ \dfrac{\partial \boldsymbol{r}}{\partial t} \end{Bmatrix}_{\boldsymbol{z}_n} + \Delta t_n \begin{bmatrix} \dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} & \dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}} \\ \dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} & \dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \end{bmatrix}_{\boldsymbol{z}_n}$$
$$\times \begin{Bmatrix} \sum_{j=1}^{i-1} \gamma_{ij}\dot{\boldsymbol{U}}_{nj} \\ \sum_{j=1}^{i-1} \gamma_{ij}\dot{\boldsymbol{Q}}_{nj} \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{g}(T_{ni}, \boldsymbol{S}^u_{ni}, \boldsymbol{S}^q_{ni}) \\ \boldsymbol{r}(T_{ni}, \boldsymbol{S}^u_{ni}, \boldsymbol{S}^q_{ni}) \end{Bmatrix}. \quad (40)$$

In order to minimise the matrix operations, the tangent operator of the right-hand side is moved to the left which leads in view of Eq. (33) to

$$\begin{bmatrix} -\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} & -\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}} \\ -\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} & \boldsymbol{I}-\Delta t_n\,\gamma\dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \end{bmatrix}_{\boldsymbol{z}_n} \begin{Bmatrix} \boldsymbol{B}^u_{ni} \\ \boldsymbol{B}^q_{ni} \end{Bmatrix}$$
$$= \Delta t_n\, d_i \begin{Bmatrix} \dfrac{\partial \boldsymbol{g}}{\partial t} \\ \dfrac{\partial \boldsymbol{r}}{\partial t} \end{Bmatrix}_{\boldsymbol{z}_n} + \begin{Bmatrix} \boldsymbol{g}(T_{ni}, \boldsymbol{S}^u_{ni}, \boldsymbol{S}^q_{ni}) \\ \boldsymbol{r}(T_{ni}, \boldsymbol{S}^u_{ni}, \boldsymbol{S}^q_{ni}) \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{0} \\ \sum_{j=1}^{i-1} \dfrac{\gamma_{ij}}{\gamma}\dot{\boldsymbol{Q}}_{nj} \end{Bmatrix}. \quad (41)$$

The solution of this system of linear equations, see definition (32),

$$\boldsymbol{B}^u_{ni} = \dot{\boldsymbol{U}}_{ni} + \left\{ \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma}\dot{\boldsymbol{U}}_{nj} \right\},$$
$$\boldsymbol{B}^q_{ni} = \dot{\boldsymbol{Q}}_{ni} + \left\{ \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma}\dot{\boldsymbol{Q}}_{nj} \right\}, \quad (42)$$

defines the stage derivatives

$$\dot{\boldsymbol{U}}_{ni} = \boldsymbol{B}^u_{ni} - \left\{ \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma}\dot{\boldsymbol{U}}_{nj} \right\},$$
$$\dot{\boldsymbol{Q}}_{ni} = \boldsymbol{B}^q_{ni} - \left\{ \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma}\dot{\boldsymbol{Q}}_{nj} \right\}, \quad (43)$$

which have to be stored in each stage. At the end of the time step the displacements and internal variables are computable using vector addition

$$\begin{Bmatrix} \boldsymbol{u}_{n+1} \\ \boldsymbol{q}_{n+1} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{u}_n \\ \boldsymbol{q}_n \end{Bmatrix} + \sum_{i=1}^{s} b_i \begin{Bmatrix} \dot{\boldsymbol{U}}_{ni} \\ \dot{\boldsymbol{Q}}_{ni} \end{Bmatrix}, \quad (44)$$

see Eq. (19).

It has to be emphasised that the constraint (34) is not exactly satisfied any longer. However, this "weak" equilibrium condition is also in the classical approach not precisely satisfied, since the internal variables $\boldsymbol{q}$ result from an integration step. Therefore, the approximation of Eq. (34) is assumed to be justified. Furthermore, one can see the stiffness matrix in Eq. (41), $\boldsymbol{K}_n = -\Delta t_n\,\gamma\,\partial\boldsymbol{g}/\partial\boldsymbol{u}|_{\boldsymbol{z}_n}$, and the right-hand side $\boldsymbol{g}(T_{ni}, \boldsymbol{S}^u_{ni}, \boldsymbol{S}^q_{ni})$, which are known in current finite element implementations. However, one additional term arises, namely $\Delta t_n\, d_i \partial\boldsymbol{g}/\partial t|_{\boldsymbol{z}_n}$. Accordingly, one needs not only the external loads at time $t_{n+1}$, but its time derivatives as well.

It must be remarked that the system (41) is completely solved on "global" level in the investigations which follow. However, it is possible to make use of the element-wise decoupling (see the discussions in [12]), so that one would arrive on "local" (element) level at a linear system of equations. We omit this discussion since the main attention is focused on the applicability of the method.

At this point a few remarks on Rosenbrock-type methods must be made. Formulation (29), where the exact Jacobian is evaluated for $\boldsymbol{y}_n$ and an additional freedom by the choice of the coefficients $\gamma_{ij}$ is introduced, has been published in [18,24]. Methods of higher order have been analysed in [19]. Methods where an inexact Jacobian is used, which satisfy $\boldsymbol{J} = \boldsymbol{f}_{,\boldsymbol{y}}(\boldsymbol{y}_n) + \mathcal{O}(\Delta t)$, are investigated in [17]. An additional step leads to W-methods, where no assumption on the choice of $\boldsymbol{J}$ is made, see [30]. A compromise between these strategies constitutes partitioned methods where sub-blocks of the Jacobian are set to zero and only those blocks being crucial for stability remain, see [32,33,35].

Rosenbrock-type methods are frequently used for time-discretisation of partial differential equations. In [22] Rothes method is applied to systems of parabolic differential equations where the PDE is discretised in time by Rosenbrock methods before an adaptive spatial discretisation is applied to the resulting elliptic system. A frequently applied technique in numerical weather prediction is to linearise the non-linear advection terms resulting in partitioned Rosenbrock methods, see [2,16,26].

Furthermore, it must be emphasised that, although the extension of Rosenbrock methods to differential-algebraic equations using the direct approach is straightforward, it is not clear how the order of the method under consideration behaves. The investigation of the order splits into two parts: firstly, one has to establish a relation between the local error (error after one step, i.e., one step is executed starting on the exact solution) and the global error (the solution at $t_n = t_0 + n\Delta t_n$ is computed, where $\boldsymbol{y}_0 = \boldsymbol{y}(t_0)$), and, secondly, one has to investigate the local error. This has been done in [27]. To preserve the order in the case of ODEs, we need $|R(\infty)| < 1$ (see [7] for the stability function $R(z)$ of a Rosenbrock method) and additional order conditions.

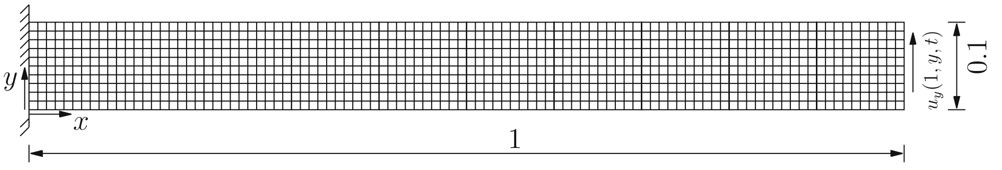


**Fig. 1** Geometric setting of the beam

There are one, four, or rather five additional order conditions to obtain the order of 2, 3 or rather 4, respectively. For methods utilising these conditions, see Sect. 3.1.1.

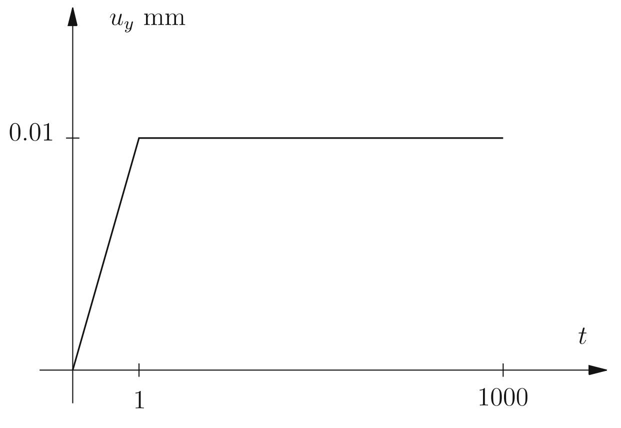
## 3 Example problem of a beam under various temporal loading conditions

We consider a beam in the $(x, y)$-plane under plane strain conditions, i.e. we assume constrained displacements $u_z = 0$ in $z$-direction and that all other variables are independent of $z$. The computational domain is a rectangular area in the $(x, y)$-plane of size $1 \times 0.1$ mm$^2$. The beam is totally fixed at the left-hand side $u_x(0, y, t) = u_y(0, y, t) = 0$ (cantilever beam). The spatial discretisation makes use of bilinear shape functions for the displacements and a spatial one-point integration. At these points, the internal variables have to be evaluated. In this respect, the evolution equations for the viscous stresses are derived by an averaging procedure in each element. Here, a regular mesh of 100 elements in $x$- and 10 elements in $y$-direction is used. Figure 1 shows the geometric setting. Since a fictive material is chosen, the material parameters are defined as follows: $K = 25$ MPa, $G = 10$ MPa, $\eta_0 = 100$ MPa s, $s_0 = 1$ MPa$^{-1}$. The material parameter $\hat{G}$ is chosen to be $\hat{G} = 100$ MPa in the first two examples and later on, in Subsect. 3.3, it is defined by $\hat{G} = 10{,}000$ MPa.

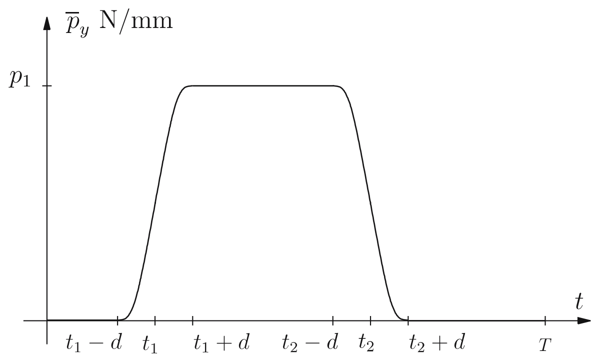
In the following, the behaviour of various time integrators in view of computational cost and accuracy are investigated, see Subsect. 3.1. To this end, the constant stepsize behaviour and time-adaptive procedures are studied using a relaxation process, see Fig. 2a. To increase the non-linear effects of the loading as well as the material non-linearities, two additional studies are added. First of all, the non-linearity of the external load is increased applying an approximated rectangular load function, Fig. 2b, see Subsect. 3.2. This influences the right-hand side of the Rosenbrock-type methods. Afterwards, the material non-linearity is increased by changing the material parameters, see Subsect. 3.3.

### 3.1 Relaxation process

In the first example, the displacement $u_y(1, y, t)$ is increased linearly in $t \in [0, 1\text{s}]$ up to a maximum of $u_y(1, y, 1) = 0.01$ mm. During the period $t \in [1, 1{,}000]$ the displacements are kept constant at, $u_y(1, y, t) = 0.01$ mm, $1 \leq t \leq 1000$ s, see Fig. 2a.



**(a)** Relaxation process



**(b)** Approximated rectangular load

**Fig. 2** Load functions of the examples

Figure 3 shows the evolution of stress $T_{xx}$ in element (50,1), i.e. at the lower boundary in the middle of the beam. In the left picture a logarithmic scale in the time interval $t \in [0, 1000]$ is used, whereas in the right picture the evolution in the time interval $t \in [0, 1]$ is displayed using a linear scale. The stress in time behaviour is moderately curved, which is a fact of a fast increase of the load. However, a non-linear behaviour could be seen during the relaxation process $t \in [1, 1000]$. After this loading process the equilibrium stress state (linear elastic behaviour) is approximately reached. Here, we define $\hat{G} = 100$ MPa.

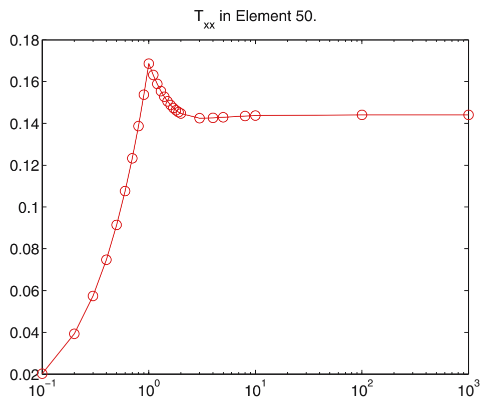


Fig. 3 Stress versus time behaviour, $(T_{xx} - t)$ at $(x, y) = (50, 1)$

### *3.1.1 Constant stepsize*

We start the studies of the Rosenbrock-type methods using a constant stepsize. In order to compare the proposed procedure with a classical approach, use is made of the classical Backward–Euler method (implicit Euler-method), usually applied in implicit finite elements. If $s$ is the number of stages and $p$ the order, the following procedures are considered:

- implicit Euler method ($s = 1, p = 1$)
- linearly implicit Euler method ($s = 1, p = 1$) ($\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n\,(\boldsymbol{I} - \Delta t_n\,\boldsymbol{J}_n)^{-1}\boldsymbol{f}(t_n, \boldsymbol{y}_n)$)
- method ROS2 from [34] ($s = 2, p = 2$)
- method ROS3P from [23] ($s = 3, p = 3$)
- method RODAS3 from [34] ($s = 4, p = 3$)
- method RODAS from [7] ($s = 6, p = 4$)

The Backward–Euler method requires the solution of one non-linear system per time-step, whereas the Rosenbrock-type methods (linearly implicit methods) with $s$ stages require the solution of $s$ linear systems with the same coefficient matrix. In the case of the Backward–Euler method, there are several possible strategies for controlling the accuracy of the solution of the non-linear system. Usually, the tolerance NTOL of the Newton–Raphson method is coupled to the prescribed error tolerance of the time-integrator. In our constant stepsize simulation such a tolerance is not available. To this end, we relate the tolerance NTOL to the stepsize via

$$\mathrm{NTOL} = 0.01\Delta t_n{}^2\,\|\mathbf{f}_0\|_{L_2} \tag{45}$$

where $\boldsymbol{f}_0$ is the first right-hand side evaluation in each step. Here, it has to be pointed out that other simulations with more restrictive tolerances lead to higher computational cost, although the results are not more accurate. Thus, the factor 0.01 seems to be sufficient.

In the first numerical experiment using constant time-steps, we compare the error at $t = 1$ for different stepsizes. In Fig. 4 the $L_2$-norm of the error in the $\boldsymbol{q}$-components (internal variables) and in the displacement components $\boldsymbol{u}$ versus the stepsize are depicted. Figure 5, however, shows the $L_2$-norm of the error in $\boldsymbol{y}^{\mathrm{T}} = (\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{q}^{\mathrm{T}})$ and the error of the stress $T_{xx}(25, 0, t)$. The predicted order of convergence is easily verified for all methods.

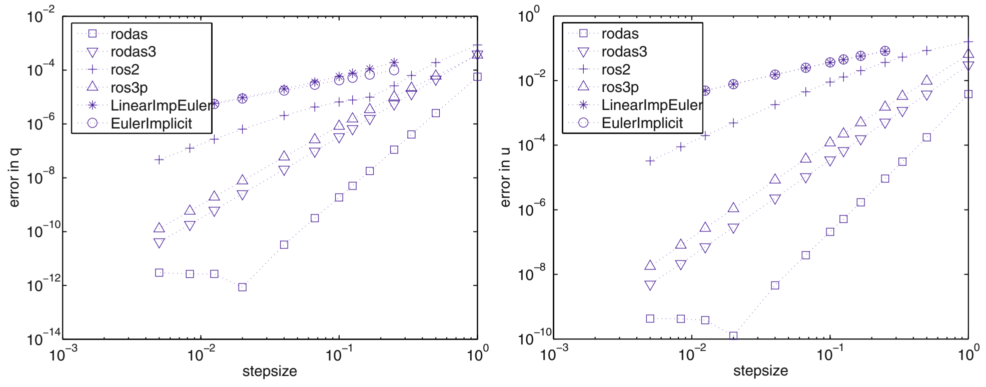


Fig. 4 Error of the components $\boldsymbol{q}$ and $\boldsymbol{u}$ versus the stepsize

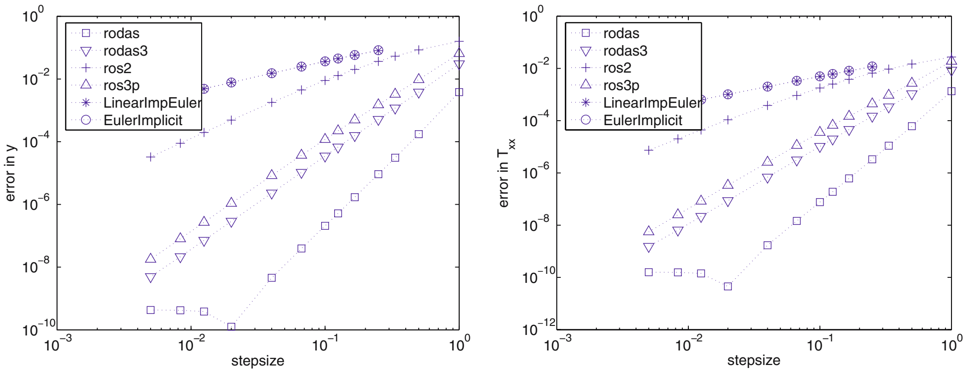


**Fig. 5** Error of the components $\boldsymbol{y}$ and $T_{xx}(25, 0, t)$ versus the stepsize

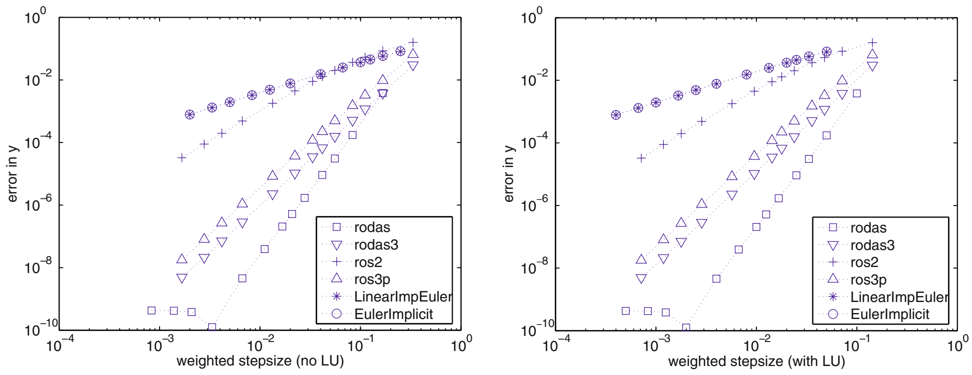


**Fig. 6** Error of the components $\boldsymbol{y}$ of different measures for the computational cost

All four figures display similar results: the Rosenbrock-type methods work very efficiently compared to the implicit Euler method and are equally robust. There is no difference in accuracy between the implicit Euler method and the linearly implicit Euler method. Note that the implicit Euler scheme usually needs no more than two Newton–Raphson iterations (three solutions of linear equations) for solving the nonlinear equations owing to the simplicity of the constitutive model and the geometrical linearity.

If the linear equations (41) are solved by some form of matrix factorisation, the main computational cost lies in the factorisation of the coefficient matrix. On the other hand, if the systems are solved by an iterative procedure, then the count for the computational cost has to be modified. The following three pictures show the $L_2$-norm for all components ($\boldsymbol{y}$) versus different measures for the computational effort. The left picture in Fig. 6 uses the number of steps multiplied by the number of stages $s$ in order to measure the computational cost. This is a fair measure if the main cost lies in the solution of the linear equation or in the evaluation of the right-hand side. The right-hand side picture of Fig. 6 uses the number of steps multiplied by the factor $s + k$, where $k = 4$ defines a measure for the computational cost. This formula is based on the assumption that the linear equation is solved by some kind of sparse LU decomposition, where the cost for the LU decomposition is approximately $k$ times the cost of the backward substitution. Note that $k = 4$ is a very conservative approach. In practice, values of $k = 10$ and more can be expected. Nevertheless, for that case we can use Fig. 5 which gives roughly the same picture.

In all the computations use is made of a sparse LU-solver. The decomposition of a system of 5,222 equations took approximately 100 ms. In Fig. 7 the performance of the methods with respect to the required CPU-time can be evaluated. Note that even if we measure computational cost by the number of evaluated stages (left picture of Fig. 6), then the higher order Rosenbrock-type methods are much faster than the implicit Euler method.

The left picture in Fig. 8 displays the error in the constraint, i.e. the equilibrium conditions which are not exactly fulfilled in the case of Rosenbrock-type methods. However, all methods keep the solution of the constraints up to the round-off error level. The right picture in the same experiment illustrates the influence of more stringent tolerances in the Newton–Raphson iteration. We have displayed the error for the linearly implicit Euler method, for the implicit Euler method with internal tolerances as in (45) and for very stringent tolerances

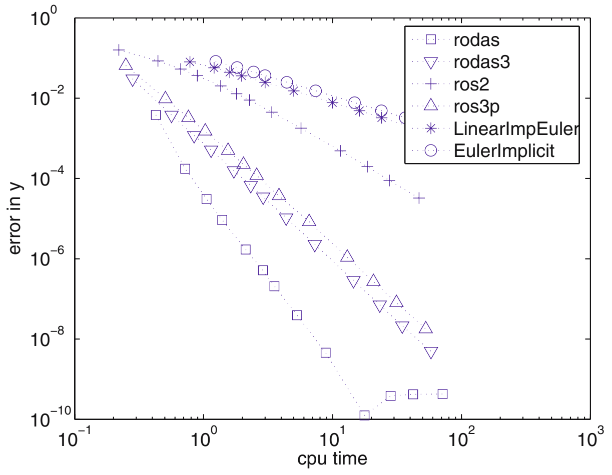


**Fig. 7** Error of $\boldsymbol{y}(1)$ versus CPU-time

where the constant 0.01 is replaced by $10^{-6}$ in (45). There is no improvement in the global error, only additional computational cost is generated if the number of iterations in the Newton–Raphson method is incremented.

### *3.1.2 Computation with stepsize control*

In order to compare all methods, a highly accurate solution with relative tolerance of $10^{-9}$ in the interval [0, 1000] is computed. At the points $t \in \{0.5, 1, 2, 10, 100, 1000\}$, the numerical solutions for different methods with different tolerances are compared.

To control the local integration error we have implemented a standard stepsize control technique based on a $C\Delta t^{q+1}$ error model where $C$ has to be estimated from the last step and $q$ is the known order of the estimated local error.

The stepsize control includes in detail:

- Estimation of local error via embedded Rosenbrock methods ROS3P, RODAS3 and RODAS.
- Estimation of local error via Richardson extrapolation for the Backward–Euler method and the linearly implicit Euler method. The extrapolation step is not carried out because of the weaker stability properties of the extrapolated solution. Of course, the more accurate solution computed by two successive Euler steps is used as update.
- The error term chosen is a combination of relative and absolute error. In the computations we prescribe RTOL = TOL and use ATOL = 0.01 · RTOL. A step is accepted if $\|\mathrm{err}\|_1 \le \mathrm{ATOL} + \mathrm{RTOL}\|\boldsymbol{y}_{n+1}\|_1$. We remark that a component-wise comparison may enhance the reliability, but we did not encounter problems with the applied simple choice.
- If a step with stepsize $\Delta t_n$ give a local error estimation err, then the next stepsize $\Delta t_{\text{new}}$ (either for the next step after acceptance or for a repeated step) is computed via

$$\Delta t_{\text{new}} = 0.85 \max\left(0.2, \min\left(5, \left(\frac{\mathrm{ATOL} + \mathrm{RTOL}\|y_{n+1}\|_1}{\max(\|\mathrm{err}\|_1, 1.E-100)}\right)^{\frac{1}{q+1}}\right)\right)\Delta t_n\,. \tag{46}$$

- The initial stepsize is chosen via $\Delta t_0 = 0.1\sqrt{\mathrm{RTOL}}$.
- The stopping criterion of the Newton iterations in the implicit Euler method is

$$\|\Delta\boldsymbol{y}_{n+1}\| \le 0.1(\mathrm{ATOL} + \mathrm{RTOL}\max(\|\boldsymbol{y}_n\|_1, \|\boldsymbol{y}_{n+1}\|_1)). \tag{47}$$

- No special strategies are applied if repeated rejections occur.

We remark that there exist several strategies to improve the behaviour of the stepsize control, especially,

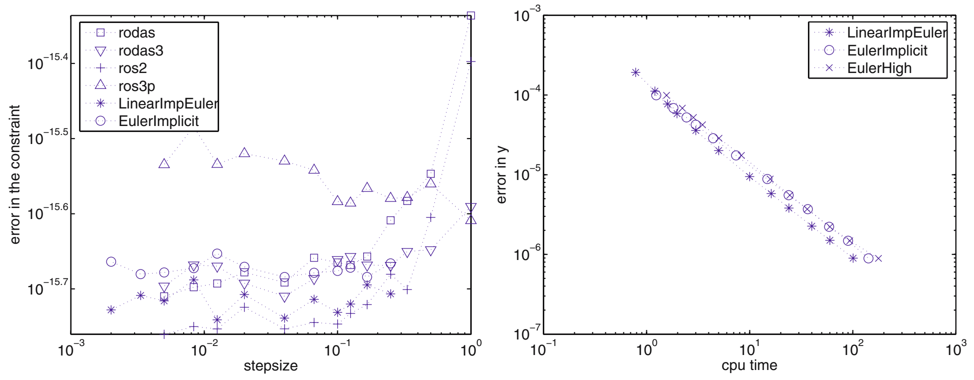


**Fig. 8** Error in constraints $\boldsymbol{g}$ versus the stepsize and the error behaviour versus the CPU-time of the Newton-iteration

if repeated rejections occur. We mention the PI controller from [6] where estimations on $C$ from the last two successful steps are used to predict a new stepsize. This strategy is especially successful when nonstiff codes are applied to (mildly) stiff equations. However, the PI controller does not suggest itself for our application where the right-hand side is non-smooth. We ignore the problems to be expected for the moment.

Figure 9 shows the global error of the solution of the Rosenbrock method RODAS and the implicit Euler method. We have chosen tolerances $10^{-4}, 10^{-6}, 10^{-8}$ for the fourth order Rosenbrock method, whereas less stringent tolerances $10^{-2}$ and $10^{-4}$ for the Euler methods are chosen. Note that the Rosenbrock method keeps the global error below the prescribed tolerance. The reason is that the error estimation by embedding a lower order method overestimates the local error approximately by a power of $\Delta t_n$. This effect cancels out because we execute roughly $|t_{\text{end}} - t_0|\bar{h}^{-1}$ steps (for an average stepsize $\bar{h} \approx \Delta t_n$). For the Euler method this is not the case because Richardson extrapolation "embeds" a method of the same order.

Figure 10 shows the attained accuracy compared with the number of steps versus the number of solved linear systems. The left picture is relevant if the cost for a (sparse) LU decomposition dominates the computational cost. The right picture is relevant if the linear systems are solved iteratively. Clearly, even in this case the higher order methods are much more effective than the Euler methods. Only for low tolerances in the magnitude of 0.01 the Euler method is competitive.

Finally, Fig. 11 shows the stepsizes which the Rosenbrock method RODAS and the Euler method need for computing the solution in the interval $t \in [0, 1000]$. Both methods increase the stepsize dramatically for $t \to 1{,}000$, i.e. the adaptive stepsize selection strategy works well for both methods. For more complicated situations, see the following subsections.

## 3.2 Rectangular loading process

The right-hand side of the Rosenbrock-type methods depends on the partial derivative of the "equilibrium conditions" (9), which results from the prescribed displacement or equivalent load vectors (Dirichlet or Neumann boundary conditions). In order to study the influence of an applied non-linear load, an approximated rectangular shear load distribution is applied to the right boundary of the cantilever beam. The load function is given by $\overline{p}_y(t, 1, y) = l_1(t)$ with

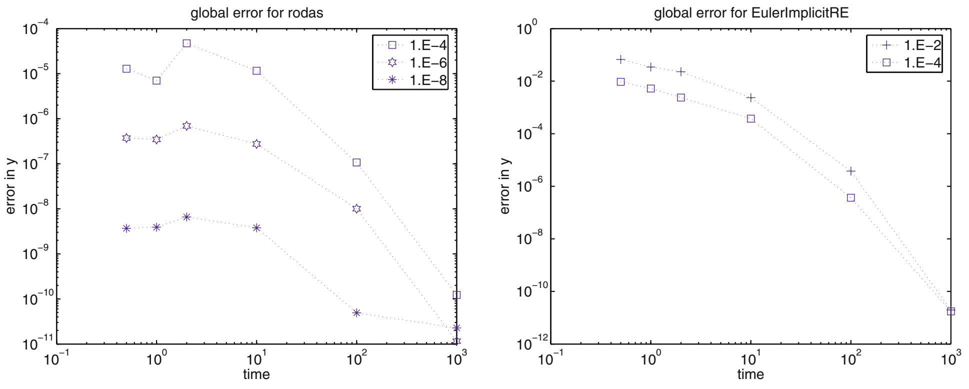


**Fig. 9** Error behaviour for $t \in [0, 1000]$ compared with the prescribed tolerances

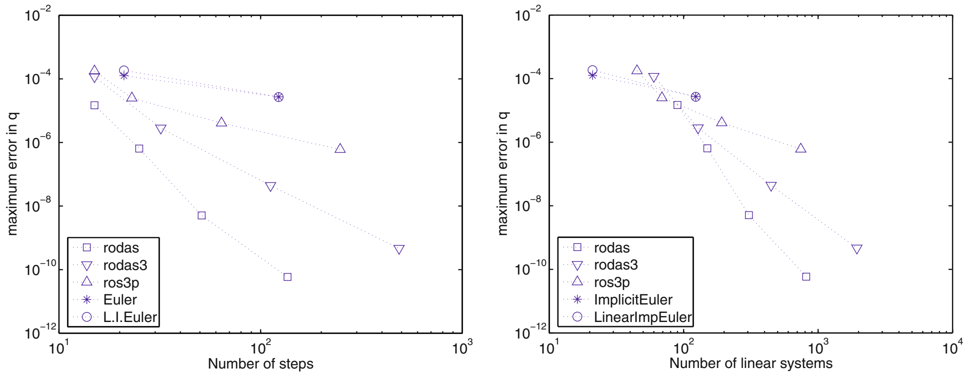


**Fig. 10** Efficiency of different methods for stepsize control

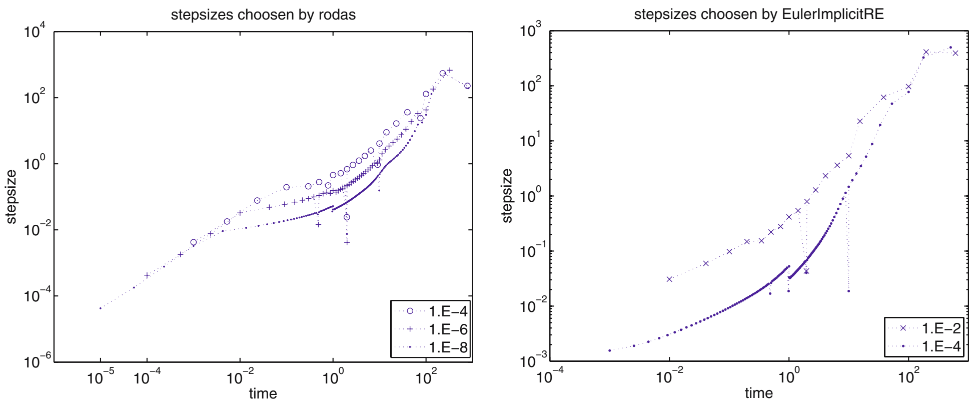


**Fig. 11** Stepsize chosen for different methods

$$l_1(t) := \begin{cases} 0 & \text{for } t \in [0, t_1 - d) \\ p_1(t) & \text{for } t \in [t_1 - d, t_1 + d] \\ 1 & \text{for } t \in (t_1 + d, t_2 - d) \\ p_2(t) & \text{for } t \in [t_2 - d, t_2 + d] \\ 0 & \text{for } t \in (t_2 + d, T] \end{cases} \tag{48}$$

The rectangular load function is approximated by the cubic polynomials $p_1(t)$ and $p_2(t)$ so that $l_1 \in C^1[0, T]$ is guaranteed. Here, $t_1 = 1$ s, $t_2 = 3$ s, $T = 1000$ s and $d = 0.05$ s are chosen, see Fig. 2b.

In the following, two studies are carried out. First of all, the time-adaptive procedures are applied to the total time interval (no switching points are taken into account, i.e. the procedure does not have any information on the temporal load distribution). In the second investigation the procedure integrates piecewise in each interval [defined in (48)], i.e. the switching points are explicitly considered.

### *3.2.1 Computation without special treatment of switching points*

Again, computations using a Rosenbrock-type method are executed where an embedded lower order method is provided for stepsize control. To make the algorithms to work harder, no use is made of the information on the switching points in the load function (48). This can lead to several stepsize rejections, especially near the first switching point at $t = 1 - d$.

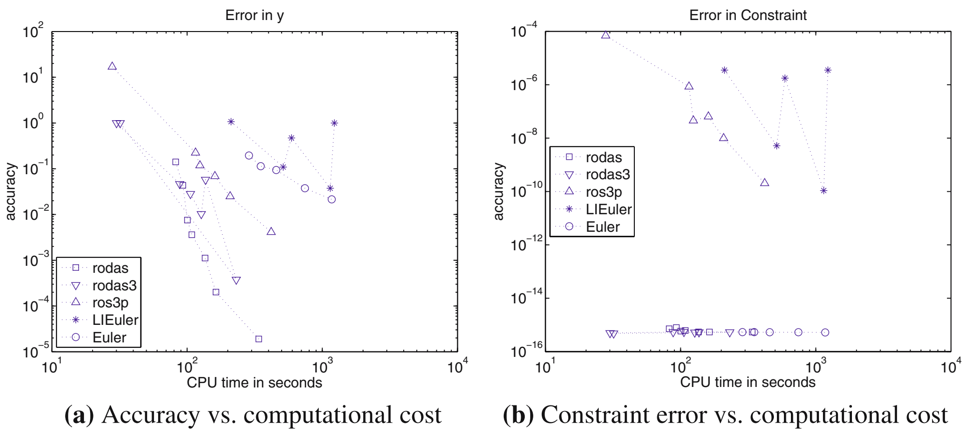
We have executed computations with tolerances of $10^{-k}$, $k = 1, 1.5, 2, 2.5, 3, 4, 6$ for the methods with order $p > 2$ and with tolerances $10^{-k}$, $k = 1, 1.5, 2, 2.5, 3$ for the methods of order $p = 1$. At points $t \in \{0.1, \ldots, 3.9, 4, 5, \ldots, 10, 15, 20, 50, 100, 500, 1000\}$ we have compared the numerical solution with a precomputed high accuracy solution. In order to compute the solution at the output points without influencing the stepsize control we used the following procedure: whenever such an output point lies within $[t_n, t_{n+1}]$, then a solution at the output point is computed by a fractional step which is discarded afterwards. We remark that, in practice, an embedded formula for continuous output or integration up to the output points are the methods of choice.

In Fig. 12, we compare the efficiency of the methods. The advantage of stiffly accurate methods is illustrated in Fig. 12b. RODAS and RODAS3 are stiffly accurate, i.e. $b_i = \alpha_{si} + \gamma_{si}$ and $c_s = 1$. This implies for our application (DAEs with constraints linear in $y, z$)

$$\boldsymbol{0} = \boldsymbol{g}(t_{n+1}, \boldsymbol{u}_{n+1}, \boldsymbol{q}_{n+1}) + \Delta t_n \sum_j \gamma_{sj} \frac{\partial \boldsymbol{g}}{\partial t}(t_n + c_j \Delta t_n).$$

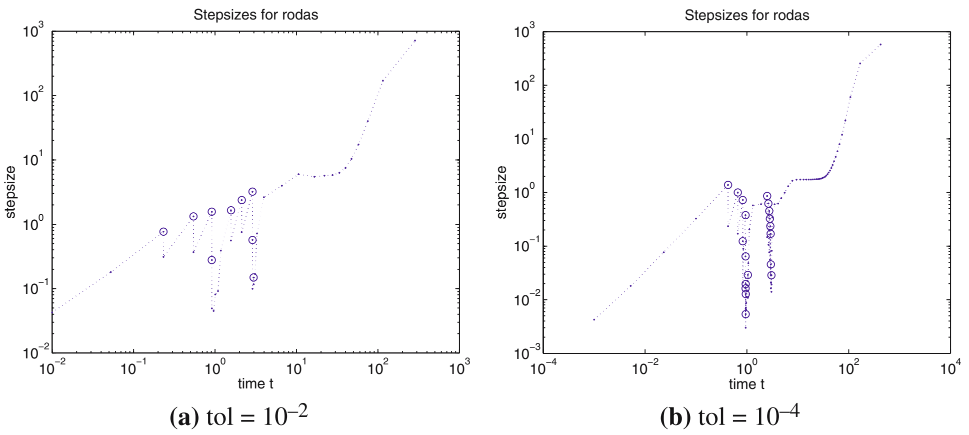
Therefore, the constraint is exactly satisfied in $t_{n+1}$ when $g$ is linear in $t$. ROS3P is not stiffly accurate and has stability function $R(\infty) = 0.73$. The result is an accumulation of discretisation errors in the constraint. The linearly implicit Euler method is stiffly accurate, but only of order 1. Nevertheless, the inaccuracy in the constraint is of magnitudes smaller than the discretisation error in the dynamic components for both ROS3P and the linearly implicit Euler method. To our experience this minor drawback is not essential in most applications. It is a point worth to be considered only when high accuracy in the constraints is explicitly required.

Figure 12a displays the maximum of $\|\boldsymbol{y}\|_1$ (global error in all components) among the output points versus the CPU time. Again, the high order methods provide more accurate results even for moderate and low tolerances than the first order method using the same amount of CPU time. Separate plots of algebraic ($\boldsymbol{u}$) and differential ($\boldsymbol{q}$) components show similar results. For several low tolerances and several methods the relative global error is in the magnitude of 1. This is particularly caused by the failure to detect the switching points – the method steps completely over the interval $[1-d, 3+d]$ and com-



**(a)** Accuracy vs. computational cost **(b)** Constraint error vs. computational cost

**Fig. 12** Accuracy behaviour without treatment of switching points with $\hat{G} = 100$ MPa



**(a)** tol = $10^{-2}$ **(b)** tol = $10^{-4}$

**Fig. 13** Stepsize behaviour of RODAS without treatment of the switching points

putes a zero solution. The probability of this behaviour increases when the method uses only nodes $c_i \in \{0, 1\}$.

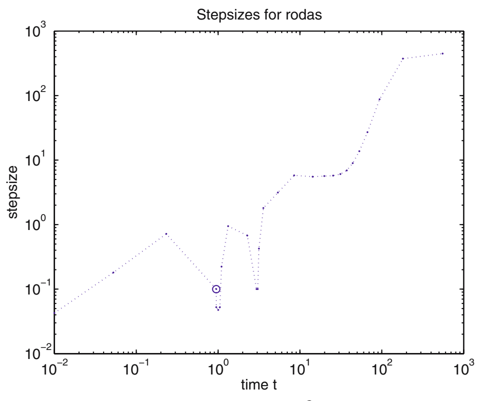
We restrict ourselves to the application of RODAS as a high-order method for an illustration of the stepsize behaviour near the switching points.

Figure 13 shows the behaviour of the time-adaptive higher order procedure. For $t < 1 - d$ the solution is constant zero. Therefore, the stepsize is increased by the maximum allowed factor 5 in each step. Accordingly, too large time-steps are estimated resulting in a number of rejections at $t = 1 - d$. Accepted time-steps (which do not step beyond $t = 1 - d$) and rejected steps (which step beyond $t = 1 - d$) alternate.

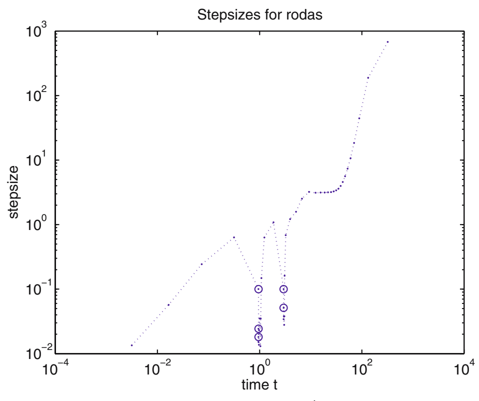
In principle, it is advisable to integrate up to the switching points when these are known in advance, see [15]. When the location of the discontinuity is not a priori known then strategies for the detection of switching points may be applied. However, these investigations are beyond the scope of the article. Moreover, in our experiments most methods did not suffer performance from these difficulties because the main computational effort is within the intervals $[1 - d, 1 + d]$ and $[3 - d, 3 + d]$ where very small time-steps have to be used due to the strongly non-linear behaviour of the load function. If the numerical solution has passed the point $t = 1 - d$, then the stepsize control works quite satisfactorily. Nevertheless, here are the switching points a priori known. Accordingly, we add further numerical experiments where the information on the discontinuities is exploited.

### *3.2.2 Computation with special treatment of switching points*

We continue the investigations and take into account the switching points, i.e. we integrate exactly up to the switching points $t \in \{1 - d, 1 + d, 3 - d, 3 + d\}$. The stepsize selections of RODAS are depicted in Fig. 14. Obviously, the number of rejected time-steps is reduced drastically, which implies that a priori known switching points should be taken into account so that a more efficient procedure is obtained. A similar behaviour can be seen for the linear implicit and the Backward–Euler

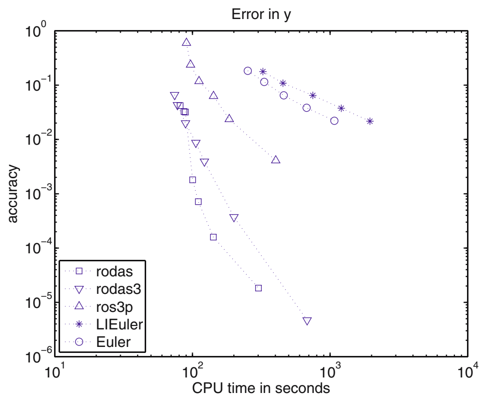


**(a)** tol = $10^{-2}$

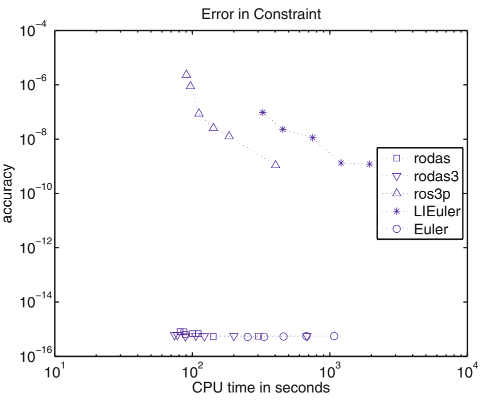


**(b)** tol = $10^{-4}$

**Fig. 14** Stepsize behaviour of RODAS with treatment of the switching points



**(a)** Accuracy vs. computational cost



**(b)** Constraint error vs. computational cost

**Fig. 15** Accuracy behaviour with treatment of switching points with $\hat{G} = 100$ MPa.

methods, where Richardson-extrapolation is used as a stepsize control technique, which is omitted for brevity.

The performance of all methods is compared in Fig. 15.

By including information on the switching points all methods work reliable. Note, that the Euler method is slightly more efficient than its linearly-implicit version.

## 3.3 Change of material parameters

In order to study the influence of the material properties, the material parameter $\hat{G}$ is changed to $\hat{G} = 10,000$ MPa leading to a more drastical change in the shear stress–shear strain behaviour, see Fig. 16.

This can be shown by applying Eqs. (1)–(6), where one obtains for the simple shear problem $\mathbf{E} = \gamma/2(\mathrm{e}_1 \otimes \mathrm{e}_2 + \mathrm{e}_2 \otimes \mathrm{e}_1)$ the ordinary differential equation

$$\dot{\tau}_{\text{ov}} + \frac{2\hat{G}}{\eta(\tau_{\text{ov}})}\tau_{\text{ov}} = \hat{G}\dot{\gamma}$$

for $\gamma(t) = \dot{\gamma}t$ with $\dot{\gamma} =$ const. and $\eta = \eta_0 \exp(-s_0|\tau_{\text{ov}}|)$.

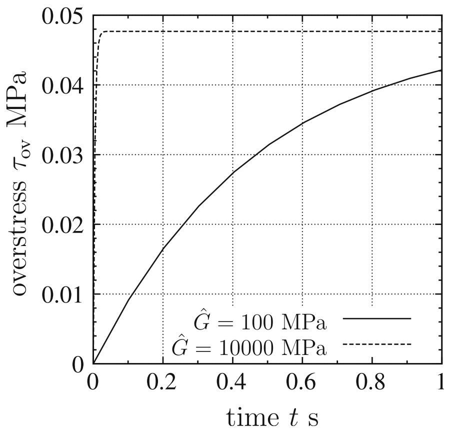
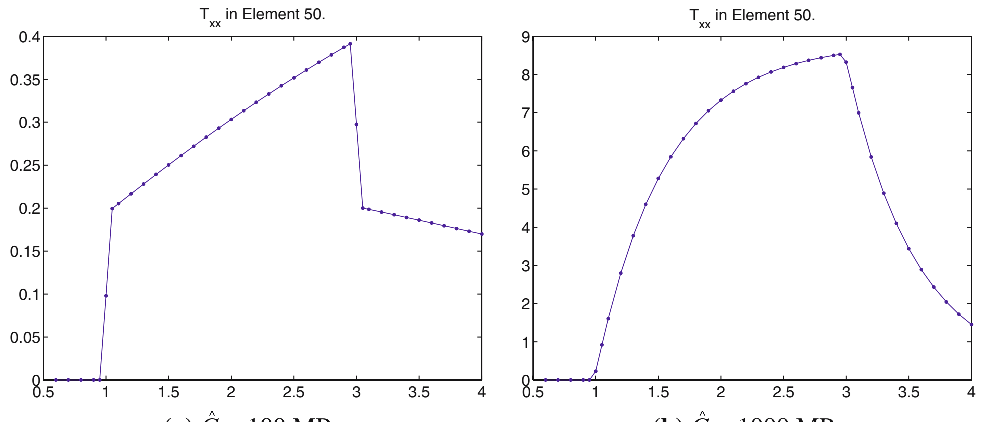


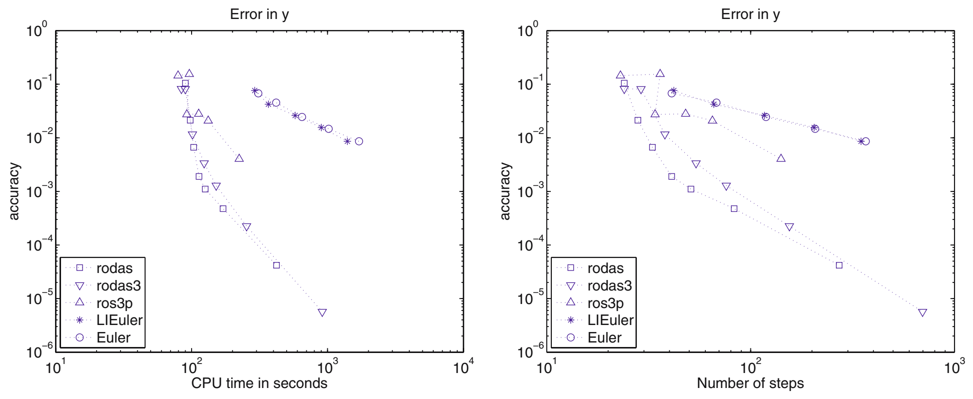
**Fig. 16** Shear strain – shear stress behaviour in dependence of $\hat{G}$ ($\dot{\gamma} = 10^{-3}$ s$^{-1}$)

In the following investigations, both the non-linear load function (48) and the more non-linear material behaviour are considered simultaneously. Here, the explicit consideration of the switching points is taken into account. Figure 17 shows the difference of the stress



**(a)** $\hat{G} = 100$ MPa **(b)** $\hat{G} = 1000$ MPa

**Fig. 17** Stress-time response in element (50/1), $0 \leq t \leq 4$, for different material parameters



**(a)** Accuracy vs. computational cost **(b)** Accuracy of the constraint

**Fig. 18** Accuracy behaviour in the (more) non-linear material behaviour example ($\hat{G} = 10{,}000$ MPa)

response at element (50/1) showing the increasing non-linearity ($\hat{G} = 100$ vs. $\hat{G} = 10{,}000$). In Fig. 18a the accuracy of all unknowns, $\boldsymbol{y}$, versus the CPU-time of the time-adaptive procedure is depicted. Obviously, the higher order methods behave more efficient than the order one methods. Even for low tolerances they are much faster due to the application of the Richardson-extrapolation. The behaviour of the Backward–Euler and the linearly implicit Euler method is quite similar. The linearly implicit method needs approximately the same number of steps (Fig. 18b) as the implicit Euler method and therefore less function evaluations and CPU time in order to satisfy the prescribed error tolerances.

## 4 Conclusions

In this paper, an iteration-less time integration method for the solution of coupled systems of differential-algebraic equations, which arise in quasi-static solids mechanics of inelastic structures, is applied. These Rosenbrock-type methods work efficiently and reliably for the class of problems under consideration. Since the algorithms do not have an iterative structure, they are easy to implement. Only a fixed number of linear systems has to be solved. In other words, there is neither a local nor a global iteration to solve the equilibrium state and the internal variables. Furthermore, an embedded stepsize control technique is applicable so that only a tiny additional amount of computing time is necessary to obtain the local integration error. In comparison to implicit methods, where a coupling between the error control of the time discretisation and the error control of the iterative procedure has to be taken into account, here, only the time discretisation error has to be controlled. Since the applied constitutive model shows only moderate non-linearities, the Rosenbrock-type methods under consideration have to be investigated for the case of problems with larger non-linearities. However, this approach of applying an implicit method is an attractive alternative to currently applied implicit non-linear finite elements and has to be investigated for more complex problems in future works.

## References

1. Drozdov A (1998) Viscoelastic structures. Mechanics of growth and aging. Academic, San Diego
2. Durran DR (1999) Numerical methods for wave equations in geophysical fluid dynamics. Springer, Berlin Heidelberg New york
3. Eckert S, Baaser H, Gross D, Scherf O (2004) A BDF2 integration method with stepsize control for elastoplasticity. Comput Mech 34(5):377–386
4. Ellsiepen P, Hartmann S (2001) Remarks on the interpretation of current non-linear finite-element-analyses as differential-algebraic equations. Int J Numer Methods Eng 51:679–707
5. Fritzen P (1997) Numerische Behandlung nichtlinearer Probleme der Elastizitäts- und Plastizitätstheorie. Doctoral Thesis, Department of Mathematics, University of Darmstadt
6. Gustafsson K, Lundh M, Söderlind G (1988) A pi stepsize control for the numerical solution of ordinary differential equations. BIT 28:270–287
7. Hairer E, Wanner G (1991) Solving ordinary differential equations II. Springer, Berlin Heidelberg New york
8. Hairer E, Wanner G (1996) Solving ordinary differential equations II, 2nd eds. Springer, Berlin Heidelberg New york
9. Hartmann S (2000) A time adaptive finite-element procedure applied to creep and relaxation processes. ZAMM Z Angew Math Mech 80(Suppl. 2):S515–S516
10. Hartmann S (2002) Computation in finite strain viscoelasticity: finite elements based on the interpretation as differential-algebraic equations. Comput Methods Appl Mech Eng 191(13–14):1439–1470
11. Hartmann S (2002) On displacement control within the DIRK/MLNA approach in non-linear finite element analysis. In: Bathe K-J (ed) Computational Fluid and Solid Mechanics 2003, vol 1. Elsevier, Amsterdam, pp 316–319
12. Hartmann S (2005) A remark on the application of the Newton–Raphson method in non-linear finite element analysis. Comput Mech 36(2):100–116
13. Haupt P (2000) Continuum Mechanics and theory of materials. Springer, Berlin Heidelberg New york
14. Haupt P, Lion A (1995) Experimental identification and mathematical modelling of viscoplastic material behavior. J Continuum Mech Thermodyn 7:73–96
15. Higham DJ (1993) Error control for initial value problems with discontinuities and delays. Appl Numer Math 12(4): 315–330
16. Hundsdorfer W, Verwer J (2003) Numerical solution of time-dependent advection-diffusion-reaction equations. Springer series in computational mathematics, vol 33 In: Springer, Berlin Heidelberg New york
17. Kaps P, Ostermann A (1989) Rosenbrock methods using few LU-decompositions. IMA J Numer Anal 9:15–27
18. Kaps P, Rentrop P (1979) Generalized Runge–Kutta methods of order four with stepsize control for stiff ordinary differential equations. Numer Math 38:55–68
19. Kaps P, Wanner G (1981) A study of Rosenbrock-type methods of high order. Numer Math 38:279–298
20. Kavetski D, Binning P, Sloan SW (2004) Truncation error and stability analysis of iterative and non-iterative Thomas-Gladwell methods for first-order nonlinear differential equations. Int J Numer Methods Eng 60:2031–2043
21. Kirchner E, Simeon B (1999) A higher-order time integration method for viscoplasticity. Comput Methods Appl Mech Eng 175:1–18
22. Lang J (2001) Adaptive multilevel solution of nonlinear parabolic PDE systems: theory, algorithms and applications. Lecture notes in computational science and engineering, vol 16 In: Springer, Berlin Heidelberg New york
23. Lang J, Verwer J (2001) Ros3p—an accurate third-order rosenbrock solver designed for parabolic problems. BIT 41:731–738
24. Nrsett SP, Wolfbrandt A (1979) Order conditiones for Rosenbrock-type methods. Numer Math 32:1–15
25. Rabbat NBG, Sangiovanni-Vincentelli AL, Hsieh HY (1979) A multilevel Newton algorithm with macromodeling and latency for the analysis of large-scale nonlinear circuits in the time domain. IEEE Trans Circ Syst 26:733–740
26. Reich S (2006) Linearly implicit time stepping methods for numerical weather prediction. BIT (to appear)
27. Roche M (1988) Rosenbrock methods for differential-algebraic systems. Numer Math 52:45–63
28. Scherf O (2000) Numerische Simulation inelastischer Körper. Fortschritt-Berichte VDI, Reihe 20 (Rechnerunterstützte Verfahren) Nr. 321. VDI-Verlag, Düsseldorf
29. Simo JC, Taylor RL (1985) Consistent tangent operators for rate-independent elastoplasticity. Comput Methods Appl Mech Eng 48:101–118
30. Steihaug T, Wolfbrandt A (1979) An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff ordinary differential equations. Math Comput 33: 521–534
31. Strehmel K, Weiner R (1995) Numerik gewöhnlicher Differentialgleichungen. Teubner Verlag, Stuttgart
32. Strehmel K, Weiner R, Dannehl I (1988) A study of B- convergence of linearly implicit Runge–Kutta methods. Computing 40:241–253
33. Strehmel K, Weiner R, Dannehl I (1990) On error behaviour of partitioned linearly implicit Runge–Kutta methods for stiff and differential algebraic systems. BIT 30:358–375
34. Verwer JG, Hundsdorfer WH, Blom JG (2001) Numerical time integration for air pollution models. Surv Math Ind 10(2):107–147
35. Wensch J (1998) An eight stage fourth order partitioned rosenbrock method for multibody systems in index-3 formulation. Appl Numer Math 27(2):171–183
36. Wensch J (2004) Beiträge zur geometrischen integration und anwendungen in der numerischen simulation. Habilitation Thesis, Mensch und Buch Verlag, Berlin
37. Wensch J (2005) Krylov-ROW methods for DAEs of index 1 with applications to viscoelasticity. Appl Numer Math 53(2–4):527–541
38. Wensch J, Podhaisky H, Hartmann S (2003) Time integration of index 1 DAEs with Rosenbrock methods using Krylov subspace techniques. PAMM Proc Appl Math Mech 3:573–574
39. Wittekindt J (1991) Die numerische Lösung von Anfangs-Randwertproblemen zur Beschreibung inelastischen Werkstoffverhaltens. Doctoral Thesis, Department of Mathematics, University of Darmstadt