INTEGRATION OF THE EQUATIONS OF MOTION OF MULTIBODY SYSTEMS USING ABSOLUTE NODAL COORDINATE FORMULATION

Grzegorz ORZECHOWSKI*, Janusz FRĄCZEK*

*The Institute of Aeronautics and Applied Mechanics, The Faculty of Power and Aeronautical Engineering, Warsaw University of Technology, ul. Nowowiejska 24, 00-665 Warsaw, Poland

gorzech@meil.pw.edu.pl, jfraczek@meil.pw.edu.pl

Abstract: Recently, a finite element formulation, called the absolute nodal coordinate formulation (ANCF), was proposed for the large rotation and deformation analysis of flexible bodies. In this formulation, absolute position and slope coordinates are used to define the finite element configuration. Infinitesimal or finite rotations are not used as nodal coordinates. The ANCF finite elements have many unique features that distinguish them from other existing finite element methods used in the dynamic analysis of the flexible multibody systems. In such systems, there appears the necessity of solving systems of differential-algebraic equations (DAEs) of index 3. Accurate solving of the DAEs is a non-trivial problem. However, in the literature about the ANCF one can hardly find any detailed information about the procedures that are used to solve the DAEs. Therefore, the current paper is devoted to the analysis of selected DAE solvers, which are applied to simulations of simple mechanisms.

Key words: Flexible Bodies, Multibody Systems, Absolute Nodal Coordinate Formulation, ANCF, Differential-Algebraic Equations, DAE Solvers

1. INTRODUCTION

In flexible multibody systems (FMS), rigid, flexible and very flexible bodies are interconnected by mechanical joints that allow for large relative reference translations and rotations between bodies (Frączek, 2002). In recent years, a particular interest of many research centres concentrated on the problems of effective analysis of multibody systems, whose bodies undergo large deformations and displacements. One of such methods is the Absolute Nodal Coordinate Formulation (ANCF) (Shabana, 1997). The characteristic feature of this formulation is lack of rotational degrees of freedom. Instead, to define rotation, one uses three independent slope coordinate vectors. The result is that ANCF elements usually have more nodal coordinates than the standard elements used in the FEM analysis. However, a single ANCF element is capable to obtain some complex deformed shapes (Shabana and Yakoub, 2001). Moreover, beam and shell ANCF elements may be treated as isoparametric elements, so that an arbitrary rigid body displacements, including rigid body rotations, produces zero strains (Shabana, 2003). On top of that, in the case of fully parameterised elements (i.e. those, which have all first-order slope vectors in each node), elastic forces may be calculated not only from classical beam, plate or shell theories, but also by using general continuum mechanics approach based on linear or nonlinear strain-displacement relationships.

Additionally, in the ANCF method, mass matrix of system is a constant one, even in the case of three-dimensional analysis. This fact has an essential influence on effectiveness of methods used for solving equations of motion. Moreover, the formulae expressing external forces assume concise and simple forms. Because the mass matrix of the finite element is a constant matrix, centrifugal and Coriolis forces do not appear in the equations of motion. Instead, elastic forces and their Jacobian matrix depend strongly nonlinearly on nodal coordinates, even in the cases of simple systems consisting of two-dimensional elements.

From the point of view of numerical analysis, the equations of motion used to describe a flexible multibody system constitute a system of differential-algebraic equations (DAE), whose structure is similar to that of equations describing rigid multibody systems (Frączek, 2002). Such a system has a differentiation index equal to 3, and consists of differential equations and algebraic constraints equations (usually nonlinear) (Haug, 1989). Solving differential-algebraic equations is a much more complicated and less recognized task than solving ordinary differential equations (ODE) (Brenan et al., 1996). This is because the DAEs have some features of weakly conditioned ordinary differential equations. Furthermore, in the case of thin or stiff structures, the ANCF coupled deformation modes can be associated with very high frequencies that can be source of numerical problems. Because of that, effectiveness of some integration algorithms is low (Hussein et al., 2008). The mentioned characteristics of DAEs and ANCF cause that the choice of an adequate algorithm for integrating equations of motion is not a trivial task, and needs a thorough analysis (Frączek and Malczyk (in print)). In this work, we present an approach to the analysis of selection of DAE solvers in application to simple analyses of dynamics of flexible mechanisms.

Special, dedicated methods have been used for solving the differential-algebraic equations. In total, five different algorithms were tested. The algorithms *DASSL (*Brenan et al., 1996), *GAMD* (Iavernaro and Mazzia, 1998), as well as *Radau* and *Radau5* (Haier and Wanner, 1996) were provided for free by their authors. However, the algorithm *HHT-I3* (Hussein et al., 2008) is an original method, specially implemented for the needs of the ANCF. It is a generalization of the Hilbert-Hughes-Taylor (HHT) method (Bathe, 1996) applied to DAEs. All the algorithms, besides of the *DASSL*, are capable of directly solving DAEs

of index 3. In the case of the *DASSL*, it is necessary to convert the DAE system to the form of equations of index reduced to 1.

The presented algorithms, applied for solving the DAEs, have been tested with the use of three long-term analyses of multibody systems. The analysed model was a simple mechanism of a single physical pendulum moving in a gravitational field. In the first case, the pendulum was modelled as a rigid body, in the second case – as a flexible body of high stiffness (high value of Young's modulus), and the third model was a flexible pendulum of high flexibility. In each model, one analysed selected parameters, such as changes of period and amplitude of oscillations, as well as changes in total energy of the system. Because dissipation forces did not appear in the models, the total energy of the system should be preserved, so its value should not change in time. Moreover, for rigid pendulum, and for flexible pendulum of high stiffness, also period and amplitude of oscillations should keep their values unchanged. Then, the change in these values observed in a numerical solution may signify lack of accuracy of the applied method.

The presented work is organised in the following way. In section 2, we present the essentials of the absolute nodal coordinate formulation. Expressions defining basic quantities that appear in equations of motion are given there, as well as notation of differential-algebraic equations of the analysed motion. In section 3, the analysed methods of solving of differential-algebraic equations of motion are briefly characterised. The results of performed simulation of example tests are enclosed in section 4. The last section presents summary of results of the carried-out tests, and final conclusions.

2. BACKGROUND

A single node of fully-parameterised, three-dimensional ANCF element may be characterised by at least twelve parameters [13]:

$$\mathbf{e}^{i\mathrm{T}} = \left[\mathbf{r}^{i\mathrm{T}} \quad \frac{\partial \mathbf{r}^{i\mathrm{T}}}{\partial x} \quad \frac{\partial \mathbf{r}^{i\mathrm{T}}}{\partial y} \quad \frac{\partial \mathbf{r}^{i\mathrm{T}}}{\partial z}\right] \tag{1}$$

where i is node number, $\mathbf{r}^i$ is the vector defining global position of the node, and vectors $\partial \mathbf{r}^i/\partial\alpha$ are, for $\alpha = x, y, z$, the global slope vectors of the element nodes. Here x, y, z are the coordinates of an arbitrary point on the element in the undeformed configuration.

A standard ANCF beam element contains two fully-parameterised nodes [16]:

$$\mathbf{e}^{\mathrm{T}} = [\mathbf{e}^{1\mathrm{T}} \quad \mathbf{e}^{2\mathrm{T}}] \tag{2}$$

Using the definition of element nodal coordinates, given by equation (2), we can write position vector of an arbitrary point P in the form:

$$\mathbf{r}^{\mathrm{P}} = \mathbf{S}(x, y, z)\mathbf{e} \tag{3}$$

where $\mathbf{S}$ is the matrix containing the element's shape functions, which is independent of nodal coordinates. For the beam element, this matrix takes the form:

$$\mathbf{S} = [s_1\mathbf{I} \quad s_2\mathbf{I} \quad \cdots \quad s_8\mathbf{I}] \tag{4}$$

where $s_1, s_2, \ldots, s_8$ are the element's shape functions, and $\mathbf{I}$ is a 3×3 identity matrix. For the two-node ANCF beam element of twenty-four parameters, these functions can be represented as follows [16]:

$$\begin{cases} s_1 = 1 - 3\xi^2 + 2\xi^3, & s_2 = l(\xi - 2\xi^2 + \xi^3), \\ s_3 = l(\eta - \xi\eta), & s_4 = l(\zeta - \xi\zeta), \\ s_5 = 3\xi^2 - 2\xi^3, & s_6 = l(-\xi^2 + \xi^3), \\ s_7 = l\xi\eta, & s_8 = l\xi\zeta. \end{cases} \tag{5}$$

where l is length of the element, while $\xi = x/l$, $\eta = y/l$ and $\zeta = z/l$ are the element natural coordinates.

The mass matrix of ANCF element can be calculated based on kinetic energy relationships. The global velocity vector may be found by differentiating the position vector (3) with respect to time:

$$\mathbf{v}^{\mathrm{P}} = \dot{\mathbf{r}}^{\mathrm{P}} = \mathbf{S}\dot{\mathbf{e}} \tag{6}$$

Kinetic energy of the element can be defined as:

$$E_{\mathrm{k}} = \frac{1}{2}\int_V \rho \mathbf{v}^{\mathrm{T}}\mathbf{v}\mathrm{d}V \tag{7}$$

where ρ and V are, respectively, mass density and volume of a finite element. It is worth noting that the density and the volume pertain to the initial configuration of an undeformed element.

Substituting velocity equation (6) into equation (7) we obtain:

$$E_{\mathrm{k}} = \frac{1}{2}\dot{\mathbf{e}}^{\mathrm{T}}\int_V \rho \mathbf{S}^{\mathrm{T}}\mathbf{S}\mathrm{d}V\,\dot{\mathbf{e}} = \frac{1}{2}\dot{\mathbf{e}}^{\mathrm{T}}\mathbf{M}\dot{\mathbf{e}} \tag{8}$$

where $\mathbf{M}$ is the symmetric mass matrix of the finite element:

$$\mathbf{M} = \int_V \rho \mathbf{S}^{\mathrm{T}}\mathbf{S}\mathrm{d}V \tag{9}$$

It is worthy to note that the above matrix is a constant one.

In the ANCF the Coriolis forces, tangent and centrifugal forces, and other forces resulting from differentiation of kinetic energy are equal to zero. Therefore, non-zero force values in equations of motion can only originate from external forces, from reactions in nodes, and form elastic forces. The vector of elastic forces can be written as:

$$\mathbf{Q}_{\mathrm{s}}^{\mathrm{T}} = \frac{\partial E_{\mathrm{s}}}{\partial \mathbf{e}} \tag{10}$$

where E_{s} is the elastic energy. This energy can be expressed as a function of the Green-Lagrange strain vector $\boldsymbol{\varepsilon}$, and the second Piola-Kirchhoff stress vector $\boldsymbol{\sigma}$ in the following form:

$$E_{\mathrm{s}} = \frac{1}{2}\int_V \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}\mathrm{d}V \tag{11}$$

For the linear elastic model of material, which is actually taken into consideration, we may formulate the following relationship between vectors of stress and strain:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \tag{12}$$

where $\mathbf{D}$ is the matrix of elastic coefficients. Substituting equation (12) into (11) we obtain:

$$E_{\mathrm{s}} = \frac{1}{2}\int_V \boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{D}\boldsymbol{\varepsilon}\mathrm{d}V \tag{13}$$

Then, for calculating the value of elastic energy, it is enough to determine the strain vector. The latter is associated with the Green-Lagrange strain tensor, which can be found using the following relationship:

$$\boldsymbol{\varepsilon}_{\mathrm{m}} = \frac{1}{2}(\mathbf{J}^{\mathrm{T}}\mathbf{J} - \mathbf{I}) \tag{14}$$

where $\mathbf{J}$ is the deformation gradient matrix:

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \mathbf{x}} \tag{15}$$

Differentiating the function given by equation (13) with respect to the element nodal coordinates we obtain:

$$\mathbf{Q}_{\mathrm{s}}^{\mathrm{T}} = \int_V \left(\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{e}}\right)^{\mathrm{T}} \mathbf{D}\boldsymbol{\varepsilon}\mathrm{d}V \tag{16}$$

Making use of equation (16), one can directly determine elastic force values for a given element. It is known, however, that when this relationship is used in the case of a classic ANCF beam element, there may appear volume locking [7]. To avoid this effect, one can apply selective reduced integration [6]. In order to do so, the matrix of elastic coefficients $\mathbf{D}$ should be divided into two parts:

$$\mathbf{D} = \mathbf{D}^0 + \mathbf{D}^\nu \tag{17}$$

where Poisson effect is taken into account only in the matrix $\mathbf{D}^\nu$. Then, $\mathbf{D}^0$ is a diagonal matrix having the following form:

$$\mathbf{D}^0 = \begin{bmatrix} E & 0 & 0 & 0 & 0 & 0 \\ 0 & E & 0 & 0 & 0 & 0 \\ 0 & 0 & E & 0 & 0 & 0 \\ 0 & 0 & 0 & G & 0 & 0 \\ 0 & 0 & 0 & 0 & Gk_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & Gk_3 \end{bmatrix} \tag{18}$$

where E is the Young's modulus, G is the Kirchhoff's modulus, while k_2 and k_3 are shear correction factors that are used for correction of transverse shear stiffness. As one can see, the values in matrix $\mathbf{D}^0$ do not depend on the value of Poisson's ratio ν.

The following equation shows how Poisson effect is taken into account in matrix $\mathbf{D}^\nu$:

$$\mathbf{D}^\nu = \frac{E\nu}{(1+\nu)(1-2\nu)} \begin{bmatrix} 2\nu & 1 & 1 & 0 & 0 & 0 \\ 1 & 2\nu & 1 & 0 & 0 & 0 \\ 1 & 1 & 2\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

We substitute the formula of elastic coefficients (17) into elastic energy equation (13) and obtain:

$$E_{\mathrm{s}} = \frac{1}{2}\int_V \boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{D}^0\boldsymbol{\varepsilon}\mathrm{d}V + \frac{1}{2}\int_V^{\mathrm{RED}} \boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{D}^\nu\boldsymbol{\varepsilon}\mathrm{d}V \tag{20}$$

where the component containing matrix $\mathbf{D}^0$ is computed using full integration, while the component with matrix $\mathbf{D}^\nu$ is integrated in a reduced way. In the case of the considered beam element, the reduced integration means that the function is integrated only along the beam's axis.

The only relationship to be determined is that of the generalised external force $\mathbf{Q}_{\mathrm{e}}$ associated with the gravity forces whose acceleration equals $\mathbf{g}$:

$$\mathbf{Q}_{\mathrm{e}} = \int_V \mathbf{S}^{\mathrm{T}}\rho\mathbf{g}\mathrm{d}V \tag{21}$$

Using the following expressions the dynamic equations of the flexible multibody system can be obtained in a matrix form as (Haug, 1989):

$$\begin{cases} \mathbf{M}\ddot{\mathbf{e}} + \mathbf{Q}_{\mathrm{s}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q}_{\mathrm{e}} \\ \boldsymbol{\Phi} = \mathbf{0} \end{cases} \tag{22}$$

where $\boldsymbol{\lambda}$ is vector of the Lagrange multipliers, vector $\boldsymbol{\Phi}$ contains nonlinear constrains equations, while $\boldsymbol{\Phi}_{\mathbf{e}} = \partial\boldsymbol{\Phi}/\partial\mathbf{e}$ is Jacobian matrix of the vector of constrains relative to system parameters.

Because constrains equations and their Jacobian matrix take a standard form, well known from rigid multibody systems Haug, 1989; Nikravesh, 1988), they will not be described here in detail.

Equations (22) can be solved by means of typical integration algorithms used for ordinary differential equations. It can be done after decreasing their differentiation index to one, i.e. by double differentiation of constrains equations. However, solving equations of such a type may lead to computational problems, so it is necessary to apply measures stabilizing equations of constrains (i.e. Baumgarte stabilization (Nikravesh, 1988)). For that reason, a more effective way might be the application of algorithms dedicated for solving differential-algebraic equations.

3. AGORITHMS FOR SOLVING DIFFERENTIAL-ALGEBRAIC EQUATIONS

Four algorithms applicable for solving equation (22) will be analysed in this section: the algorithm *DASSL* based on the backward differentiation formulae, the algorithms *Radau* and *Radau5* based on implicit Runge-Kutta method, the algorithm *GAMD* based on generalised Adams method, and the algorithm *HHT-I3* based on generalised HTT method. These algorithms represent different groups of numerical methods applicable for solving DAE systems. Therefore, the obtained results may contain valuable hints on applicability of various numerical schemes in solution of DAEs in the proposed initial formulation.

In most of these algorithms, it is necessary to modify the solved equations, so that they take the forms adequate for the applied algorithm. Appropriate transformations will be presented in further part of this section.

3.1. Method of backward differentiation – algorithm *DASSL*

The software of *DASSL* is an implementation of the method of backward differentiation (*Backward Differentiation Formulae*, BDF) [2], with variable order of the method changing from one to five. This was one of the first numerical methods developed for integrating DAE systems. It allows solving differential-algebraic equations of index not greater than one.

The differential scheme implemented in the *DASSL* procedure makes it possible to solve an equation system given in the following, implicit form:

$$\mathbf{G}(t, \mathbf{y}_{\mathrm{d}}, \dot{\mathbf{y}}_{\mathrm{d}}) = \mathbf{0} \tag{23}$$

where $\mathbf{y}_{\mathrm{d}}$ is the system state vector, and $\mathbf{G}$ is a vector of state equations. The program *DASSL* requires differential-algebraic equations expressed as a first-order system of equations. In order to reduce the order of equation (22), one can use the new set of variables:

$$\mathbf{u} = \dot{\mathbf{e}} \tag{24}$$

However, to decrease the integration index, one can perform

one differentiation of the constrains vector with respect to time:

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{\Phi}_{\mathbf{e}}\dot{\mathbf{e}} + \boldsymbol{\Phi}_{\mathrm{t}} = \mathbf{0} \tag{25}$$

Substituting equations (24) and (25) into (22), we obtain:

$$\begin{cases} \mathbf{M}\dot{\mathbf{u}} + \mathbf{Q}_{\mathrm{s}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q}_{\mathrm{e}} \\ \mathbf{u} - \dot{\mathbf{e}} = \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{e}}\mathbf{u} + \boldsymbol{\Phi}_{\mathrm{t}} = \mathbf{0} \end{cases} \tag{26}$$

Exclusion of equations of constrains from the above system of equations causes that the constrains may be violated (Haug, 1989). In order to prevent it, a new set of variables should be introduced, which leads to a formulation with index stabilization (the Gear-Gupta-Leimkuhler method (Gear at al., 1985)):

$$\begin{cases} \mathbf{M}\dot{\mathbf{u}} + \mathbf{Q}_{\mathrm{s}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q}_{\mathrm{e}} \\ \mathbf{u} - \dot{\mathbf{e}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\mu} = \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{e}}\mathbf{u} + \boldsymbol{\Phi}_{\mathrm{t}} = \mathbf{0} \\ \boldsymbol{\Phi} = \mathbf{0} \end{cases} \tag{27}$$

where $\boldsymbol{\mu}$ is the vector of the new variables, which have the character of Lagrange multipliers. One considers solution to the above equation system as a correct one if the vector $\boldsymbol{\mu}$ is permanently equal to zero. Numerical solution to equation (27) guarantees that the equations of position and velocity constrains are fully satisfied. The index of the above equation is equal to two. It can be further decreased to the value of one by substituting the multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ with the new variables, $\dot{\boldsymbol{\eta}} = \boldsymbol{\lambda}$ and $\dot{\boldsymbol{\zeta}} = \boldsymbol{\mu}$. The obtained equation has a differentiation index equal to one. The method has many advantages, first of all one does not need to perform the double differentiation of constrains equations, thus avoiding high computational costs. After introducing the new variables, we obtain:

$$\begin{cases} \mathbf{M}\dot{\mathbf{u}} + \mathbf{Q}_{\mathrm{s}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\dot{\boldsymbol{\eta}} = \mathbf{Q}_{\mathrm{e}} \\ \mathbf{u} - \dot{\mathbf{e}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\zeta} = \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{e}}\mathbf{u} + \boldsymbol{\Phi}_{\mathrm{t}} = \mathbf{0} \\ \boldsymbol{\Phi} = \mathbf{0} \end{cases} \tag{28}$$

The state vector and the vector of state equations, used in the *DASSL* program, have the forms:

$$\mathbf{y}_{\mathrm{d}} = [\mathbf{u}^{\mathrm{T}} \quad \mathbf{e}^{\mathrm{T}} \quad \boldsymbol{\eta}^{\mathrm{T}} \quad \boldsymbol{\zeta}^{\mathrm{T}}]^{\mathrm{T}} \tag{29}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{M}\dot{\mathbf{u}} + \mathbf{Q}_{\mathrm{s}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\dot{\boldsymbol{\eta}} - \mathbf{Q}_{\mathrm{e}} \\ \mathbf{u} - \dot{\mathbf{e}} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\zeta} \\ \boldsymbol{\Phi}_{\mathbf{e}}\mathbf{u} + \boldsymbol{\Phi}_{\mathrm{t}} \\ \boldsymbol{\Phi} \end{bmatrix} \tag{30}$$

It is noticeable that the vector $\mathbf{G}$ depends both on the state vector, and its derivative. In order to effectively carry out the simulation, one not only needs to calculate the value of vector $\mathbf{G}$, but also calculate the Jacobian matrix for this vector with respect to the state vector, as well as its derivative.

Making use of the relationships presented above, we can prepare appropriate procedures needed by the *DASSL* program for the analysis of a multibody system. The advantage of the presented method, called the *Stabilized Index 1* (SI1) formulation, is that it takes into account the influence of constrains on positions and velocities in the system. Owing to that fact, the obtained results are often more accurate in comparison to those which don't take advantage of stabilization. A disadvantage of the SI1 formulation is high computational cost associated with calculation of the vector $\mathbf{G}$ and its Jacobian matrix.

3.2. Implicit Runge-Kutta methods – algorithms *Radau* and *Radau5*

The programs *Radau* and *Radau5* are based on implicit Runge-Kutta methods (*Radau IIA* method) [8] with an automatic control of integration step. In the *Radau* method, the order is variable, equal to 5, 9 or 13, while the *Radau5* method has a constant order, equal to 5.

Both algorithms are used for solving weekly-conditioned ordinary differential equations, as well as differential-algebraic equations of index not greater than 3. The algorithms are capable of solving first-order equations of the explicit form:

$$\mathbf{M}_{\mathrm{r}}\dot{\mathbf{y}}_{\mathrm{r}} = \mathbf{F}_{\mathrm{r}}(t, \mathbf{y}_{\mathrm{r}}) \tag{31}$$

where $\mathbf{M}_{\mathrm{r}}$ is a constant matrix of masses, $\mathbf{y}_{\mathrm{r}}$ is the state vector, and $\mathbf{F}_{\mathrm{r}}$ is a vector function of the right sides. In the case of solving DAEs $\mathbf{M}_{\mathrm{r}}$ is a singular matrix. This matrix must not be confused with the mass matrix $\mathbf{M}$ of a multibody system, i.e. that appearing in equation (22).

Similarly as in the case of DASSL program, it is necessary to lower the order of equation to one, which can be done by substitution defined by equation (24). One obtains the equation of motion in the form:

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} & \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{e}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{e}} - \mathbf{Q}_{\mathrm{s}} \\ \mathbf{u} \\ \boldsymbol{\Phi} \end{bmatrix} \tag{32}$$

The structure of the above equation reflects the structure of equation (31). However, it must be noted that the matrix corresponding to the matrix $\mathbf{M}_{\mathrm{r}}$ is not a constant one. Due to this fact, the above equation should be re-written in a different form. In order to do so, one introduces new variables $\mathbf{w} = \dot{\mathbf{u}}$, which represent accelerations of the vector $\mathbf{e}$. Then, the differential equations may be written as additional constrains equations:

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{w}} \\ \dot{\mathbf{u}} \\ \dot{\mathbf{e}} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{w} + \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\lambda} + \mathbf{Q}_{\mathrm{s}} - \mathbf{Q}_{\mathrm{e}} \\ \mathbf{w} \\ \mathbf{u} \\ \boldsymbol{\Phi} \end{bmatrix} \tag{33}$$

On the basis of the above equations, one can easily write, in an explicit form, the matrix of masses and the vector of right sides for the *Radau* programs. It is clear that now the mass matrix is a constant (and singular) one. The state vector can be written as:

$$\mathbf{y}_{\mathrm{r}} = [\mathbf{w}^{\mathrm{T}} \quad \mathbf{u}^{\mathrm{T}} \quad \mathbf{e}^{\mathrm{T}} \quad \boldsymbol{\lambda}^{\mathrm{T}}]^{\mathrm{T}} \tag{34}$$

To use the programs *Radau* and *Radau5* effectively, one needs to calculate the Jacobian matrix for vector $\mathbf{F}_{\mathrm{r}}$ with respect to the state vector $\mathbf{y}_{\mathrm{r}}$. This matrix takes a more simple form than the Jacobian matrix for the program *DASSL* (although computational cost is still high).

3.3. Generalized Adams method – algorithm *GAMD*

The *GAMD* software is an implementation of the generalized Adams method (GAM – acronym of *Generalized Adams Method*) (Iavernaro and Mazzia, 1998). This is a method of variable order (3, 5, 7 or 9), with automatic control of integration step. By using this algorithm, one can solve differential-algebraic equations of index not greater than 3. In this algorithm, the authors used

a part of the code from the procedure *Radau5*.

The *GAMD* algorithm can solve equations having identical form as those solved by *Radau5*, i.e. given by equation (31). The algorithm solves equations in an explicit form, such as equation (33), with the state vector from equation (34). In practice, both *Radau* programs and the *GAMD* program use the same procedures for calculating the mass matrix, the vector of right sides, and the Jacobian matrix of this vector.

3.4. Generalized HHT method – algorithm *HHT-I3*

The implicit algorithm *HHT-I3* serves for directly solving differential-algebraic equations of index equal to 3, having the form represented by equation (22) (Hussein et al., 2008). The algorithm is based on classic Hilbert-Hughes-Taylor method (Bathe, 1996) adapted for solving DAEs of index equal to 3. This algorithm has the possibility of selecting the magnitude of numerical damping, and was tested in application to flexible bodies modelled with the use of the ANCF method. The algorithm offers the possibility of automatic selection of integration step. In contrast to previously described methods used for solving DAEs, any widely available implementation of this algorithm has not been known to the authors. Because of that, the algorithm was originally implemented by the authors for the needs of this paper.

The HHT method is often used for solving second-order ordinary differential equations. It is based on the Newmark method (Bathe, 1996). Numerical damping, introduced into this method, facilitates eliminating undesirable, high-frequency oscillations. The Newmark method is unconditionally stable, but is characterised by first-order convergence. The HHT method also introduces numerical damping, is unconditionally stable, and guarantees second-order (quadratic) convergence.

When using the *HHT-I3* method, we write equation (22) in the form:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{e}} = \mathbf{H} \\ \boldsymbol{\Phi} = \mathbf{0} \end{cases} \tag{35}$$

where the vector $\mathbf{H}(t, \mathbf{e}, \dot{\mathbf{e}}) = \mathbf{Q}_{\mathrm{e}} - \mathbf{Q}_{\mathrm{s}} - \boldsymbol{\Phi}_{\mathbf{e}}^{\mathrm{T}}\boldsymbol{\lambda}$.

The algorithm *HHT-I3* solves equation (35) in a direct way, so that there is no simple division into the vector of right sides and the state vector, unlike to what was needed in the case of previously described general-purpose algorithms. It is necessary, however, to assume initial conditions for positions and velocities, and prepare appropriate procedures for computation of the Jacobian matrix. It is worth emphasizing that computational cost of determining Jacobian matrix in the procedure *HHT-I3* is comparable to that encountered in the programs *Radau* and *GAMD*. A detailed description of the *HHT-I3* algorithm can be found in literature (Hussein et al., 2008).

It should be emphasized that, in the program HHT-I3, there is no implemented procedure for assessing the necessity of recalculating the Jacobian matrix in a consecutive steps of integration. In consequence, this matrix is calculated in each integration step. This characteristic of the procedure causes that its effectiveness might be much lower than effectiveness of other algorithms, in which a method for assessing Jacobian matrix is implemented.

4. TEST EXAMPLE

Simple numerical tests are proposed for the programs presented in section 3. The tests are concerned with computations of dynamics of three-dimensional mechanisms, both rigid and flexible ones. The purpose of the tests is an initial assessment of effectiveness of the integrating algorithms through evaluation of their computational accuracy, numerical stability and time of computations.

The computations were performed with the use of Fortran program on a computer equipped with a four-core, eight-thread processor with 3.4 GHz nominal frequency and 8 MB cache memory (Intel Core i7-2600K).

As a test example, we have chosen a problem of dynamic analysis of a single physical pendulum fixed at one of its ends to a base by the revolute joint. The pendulum moves under the influence of gravitation forces. The element has a cuboidal shape of 40cm length, 4cm height and 2cm depth. In all models, the element's material has a density of 7801kg/m^3 and the Poisson ratio equal to 0,3. At the initial moment, the pendulum is in a horizontal position.

It is known that rigid physical pendulum should move with constant frequency of oscillations and constant amplitude, irrespective of duration of stimulation time. In such a system, there are no dissipation forces, so that total energy of the system must be preserved. Of course, because of errors arising in the process of integration (including the integration and numerical errors) the value of the mentioned quantities might change in time. One can assess the range of this variability for individual algorithms.

In the case of analysis of dynamics of a rigid physical pendulum it is possible to calculate the theoretical period of oscillations. For an infinitesimal angle of oscillation, the theoretical period of oscillation can be expressed by the formula:

$$T_0 = 2\pi\sqrt{\frac{I_0}{mgr}} \tag{36}$$

where I_0 is moment of inertia of the body with respect to the point of rotation, m is mass of the body, g is acceleration of gravity, and r is the distance between the centre of mass and the point of rotation. For the presented data, this period equals $T_0 = 1.04$s.

In order to calculate the period of oscillation of a rigid physical pendulum moving in a finite rage of oscillation angles, one should determine the value of the following integral:

$$T_{\mathrm{t}} = \frac{2T_0}{\pi}\int_0^{\pi/2}\frac{\mathrm{d}\varphi}{\sqrt{1-k^2\sin^2\varphi}} \tag{37}$$

where φ is the independent variable, and $k = \sin\frac{1}{2}\alpha_{\max}$, in which $\alpha_{\max}$ is the maximum angle of pendulum, measured with respect to the equilibrium position (in the considered case, this angle equals $90°$). The integral was calculated numerically, and the period of oscillations was found equal to $T_{\mathrm{t}} = 1.22$s.

In the simulations described in this section, one assumed the ending time equal to $t_{\mathrm{k}} = 600$s. During the simulations, we calculated the parameters such as total energy value, amplitude decay, and elongation of oscillation period. The value of total energy was calculated directly. The percentage amplitude decay was calculated from the formula:

$$A_{\mathrm{z}} = \frac{A_0 - A}{A_0}\times 100\% \tag{38}$$

where A_0 is the theoretical value of amplitude, and A is its current value. Positive values of this parameter mean that oscillations are damped (the maximal value of 100% indicates complete decay of oscillations), while negative values of A_{z} denote oscillations

of increasing amplitude.

The period elongation of oscillations can be calculated in a similar way:

$$T_{\mathrm{w}} = \frac{T - T_{\mathrm{t}}}{T_{\mathrm{t}}} \times 100\% \tag{39}$$

where T is the current value of oscillation period, and T_{t} is its theoretical value calculated from equation (37). For a rigid pendulum, T_{w} should be always equal to zero. Its negative value means shortening of the period.

The results presented in Tab. 1, 2 and 3 are extreme values of the analysed quantities calculated for the whole duration of simulation. All the analyses were carried out for default values of majority of the parameters in solving procedures. The variable values were absolute errors (denoted as *Atol*) and relative errors (*Rtol*). When analysing the procedure *HHT-I3*, it was only possible to determine the value of relative error – the *Atol* value was ignored in this procedure.

4.1. Rigid physical pendulum

In this case, position of the body is described with a vector of seven independent coordinates (three of them describing position of the centre of mass, and four others are Euler parameters defining orientation of the body), and there are six equations of constrains (one equation for Euler parameters, and five equations for the revolute joint). Therefore, the analysed system has one degree of freedom.

For a rigid body, total energy is a sum of element's kinetic energy and its potential energy in gravitation field. In Tab. 1, there are presented results of performed simulations. In the cases of procedures *Radau* and *Radau5*, some of the simulations ended ahead of time, because the selected integration step was too small. In such a situation, there are no results in Table 1, and the value in the column of computational time means the time of simulation termination due to the procedure error.

Tab. 1 presents results for three selected tolerance values. For the greatest tolerance value (test I), we managed to obtain acceptable results only in the case of the algorithm *HHT-I3.* For the remaining programs, simulation failed, or oscillations of pendulum were strongly damped. The results obtained in test II showed that the algorithms *DASSL*, *GAMD* and *HHT-I3* solved the problem without difficulties, and the simulation results could be considered as acceptable. In the cases of algorithms *GAMD* and *HHT-I3*, the oscillations were mildly damped; however, the algorithm *DASSL* produced oscillations of increasing amplitude. This was an undesirable effect which, for a longer duration of simulation, might indicate lack of convergence of the solution.

When the lowest values of tolerances (test III) were applied, all the algorithms gave satisfactory results. In this case, the algorithm *DASSL* proved the most effective, and gave very good overall results. It might result from the fact that the applied SI1 formulation takes into account not only the equations of constrains (which are considered in all remaining algorithms), but also their velocities.

The algorithm DASSL proved the best one as far as time of computations was concerned. The calculation step, selected by the program, was relatively big, and computation of the Jacobian matrix was performed not very frequently. Slower than this one were the algorithms from the *Radau* group. At the same time it should be noted that the results obtained with different algorithms of this group, and the times of computations for the constant-order algorithm (*Randau5*) and the variable-order algorithm (*Radau*) were comparable.

The *GAMD* algorithm turned out to be twice slower than the *Radau* algorithms, however, it could solve the problem in which tolerance values were much higher. The *HHT-I3* algorithm turned out to be the slowest one. It was due to the fact that, in this algorithm, the Jacobian matrix was computed in each iteration step. Moreover, this method was designed especially for the needs of the ANCF method, therefore the algorithm might not optimally select integration step in the case of analysis of rigid bodies (where mass matrix is not a constant one). Nevertheless, by using this algorithm, we could obtain correct solution even for the greatest tolerance values.

Tab. 1. Results of simulation of rigid physical pendulum

Tolerance	Algorithm	Total energy [J]	Amplitude decay [%]	Period elongation [%]	Computation time [s]
Test I: *Atol* 10^{-6} *Rtol* 10^{-3}	*DASSL*	-2.40	48.8	-8.99	0.82
	Rasau5	-	-	-	sim. to 1.3
	Radau	-	-	-	sim. to 3.3
	GAMD	-1.73	35.4	-6.85	9.67
	HHT-I3	-0.11	0.99	-0.22	9.87
Test II: *Atol* 10^{-7} *Rtol* 10^{-5}	*DASSL*	0.27	-5.50	1.29	1.5
	Rasau5	-	-	-	sim. to 37.6
	Radau	-	-	-	sim. to 130
	GAMD	-0.59	12.0	-2.58	12.8
	HHT-I3	-0.39	7.9	-1.73	8.6
Test III: *Atol* 10^{-9} *Rtol* 10^{-7}	*DASSL*	0.004	-0.08	0.02	2.9
	Rasau5	0.05	0.32	-0.07	8.5
	Radau	0.05	0.28	-0.06	8.6
	GAMD	0.05	-0.07	0.01	20.3
	HHT-I3	-0.03	0.6	-0.13	35.3

4.2. Flexible physical pendulum

The pendulum was constructed with the use of standard, fully parameterised, ANCF beam elements. Elastic forces were computed by applying selective, reduced integration, in order to avoid the influence of volume locking on the results. The pendulum, shown in Fig. 1, was divided into six finite elements, so that the system was described by eighty-four differential equations. Additionally, one must take into account six algebraic equations of constrains describing the revolute joint. In comparison to the rigid pendulum, this system of equation of motion consisted of the same number of algebraic equations, and twelve times greater number of differential equations. We also carried out simulations with the use of a pendulum consisting of ten finite elements. However, the differences between the results of simulation of this pendulum, and those obtained for the six-element pendulum were insignificant, so that only the latter would be presented here.

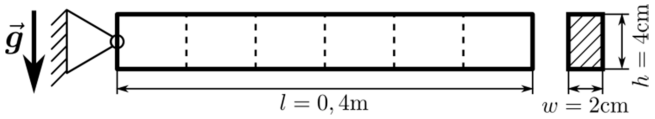

Fig. 1. Flexible physical pendulum

The simulations were carried out for two flexible models differing from one another by the values of Young's modulus. In the first case, Young's modulus was equal to $E = 2 \times 10^{11}$ Pa. Such a stiff pendulum should deform only insignificantly when moving in the field of gravitational forces. Because of that, the courses obtained in simulation should be similar to those of determined for rigid pendulum. It means that the values of amplitude and period of oscillation are comparable to respective values of oscillation amplitude and period of the rigid pendulum.

The difficulty of solving a system of high stiffness is associated with to the fact that the equations of a standard ANCF beam element with high value of Young's modulus contain high-frequency components, which causes numerical difficulties. The following example will illustrate how the above-described algorithms could cope with solving a very stiff system of equations.

In the second model, the value of Young's modulus was decreased to $E = 2 \times 10^{7}$ Pa. The ANCF formulation is exceptionally effective in the case of models of low stiffness (Gerstmayr and Shabana, 2006), so that the described algorithms should solve this problem with relatively low computational cost. Low stiffness of the pendulum means that there might appear significant deformations of various frequencies, so that making comparisons between amplitudes and oscillation periods is useless. Therefore, only total energy value was calculated for this model. The results shown in Tab. 1 and 2 indicate that the information about total energy value is sufficient for assessment of accuracy of the analysed algorithms. For the systems with flexible bodies, total energy value was calculated as a sum of kinetic energy, potential energy of gravity forces and elastic energy (according to formula (20)).

Tab. 2 summarises simulation results obtained for the first, stiffer pendulum model. The tests carried out with the use of the *Radau* algorithms ended ahead of time in each case, because of too small integration step applied by the procedure. The results obtained by using the *DASSL* algorithm are printed in italics, because duration of all simulations performed with this algorithm exceeded the maximum allowable computation time equal to 10 hours. In such cases, the value in the "computation time" column means the time of calculation end reached after 10 hours of computations. We managed to obtain correct results only when using the algorithms *GAMD* and *HHT-I3*.

None of the algorithms gave satisfactory results for the highest tolerance values (test I). The results obtained with the use of the *GAMD* algorithm indicated that the oscillations were completely damped, while those produced by the *HHT-I3* were at variance with the physical sense of the problem. Simulation results obtained for lower tolerance values were fully acceptable, and their accuracy increased with decreasing tolerance values. The algorithm *HHT-I3* proved much more effective than the *GAMD* (despite the fact that its implementation was not optimal) for medium and low tolerance values. In the case of the highest tolerances (test I), the computations performed with the use of this algorithm took a much longer time, which was due to the fact that the computed results were erroneous (oscillations of increasing amplitude).

Tab. 2. Results of simulation of flexible physical pendulum ($E = 2 \times 10^{11}$Pa)

Tolerance	Algorithm	Total energy [J]	Amplitude decay [%]	Period elongation [%]	Computation time
Test I: *Atol* 10^{-6} *Rtol* 10^{-3}	*DASSL*	*0.01*	*0.32*	*0.18*	*sim. to 30.3s*
	Rasau5	-	-	-	sim. to 0.6s
	Radau	-	-	-	sim. to 1s
	GAMD	-4.90	100	-15.5	18m 19s
	HHT-I3	138.4	-102	98.9	1h 16m 28s
Test II: *Atol* 10^{-7} *Rtol* 10^{-5}	*DASSL*	*-0.001*	*0.06*	*0.09*	*sim. to 75s*
	Rasau5	-	-	-	sim. to 8.7s
	Radau	-	-	-	sim. to 2.3s
	GAMD	0.07	-1.36	0.32	1h 25m 57s
	HHT-I3	-0.02	0.44	-0.09	17m 38s
Test III: *Atol* 10^{-9} *Rtol* 10^{-7}	*DASSL*	*5x10-6*	*-0.02*	*0.12*	*sim. to 83.3s*
	Rasau5	-	-	-	sim. to 16.0s
	Radau	-	-	-	sim. to 16.4s
	GAMD	0.07	0.02	-0.008	2h 47m 20s
	HHT-I3	-0.002	0.04	-0.02	1h 03m 44s

Tab. 3. Results of simulation of flexible physical pendulum ($E = 2 \times 10^7$Pa)

Tolerance	Algorithm	Total energy [J]	Computation time
Test I: *Atol* 10^{-6} *Rtol* 10^{-3}	*DASSL*	0.007	2h 57m 04s
	Rasau5	1.27	17m 02s
	Radau	3.67	27m 45s
	GAMD	0.07	11m 31s
	HHT-I3	2.46	3m 54s
Test II: *Atol* 10^{-7} *Rtol* 10^{-5}	*DASSL*	0.007	42m 48s
	Rasau5	0.07	13m 38s
	Radau	0.09	28m 51s
	GAMD	0.07	14m 50s
	HHT-I3	-0.25	11m 36s
Test III: *Atol* 10^{-9} *Rtol* 10^{-7}	*DASSL*	0.007	43m 08s
	Rasau5	0.07	28m 34s
	Radau	0.08	28m 44s
	GAMD	0.07	1h 00m 48s
	HHT-I3	0.007	1h 14m 44s

In Tab. 3, there are shown results of simulation of the pendulum which can be subject to significant deformations because of low value of Young's modulus. In the case of this model, it was possible to obtain some results for all algorithms and all tolerance values. In the case of test I, acceptable results were obtained only when using the algorithms *DASSL* and *GAMD*. The results obtained with the use of the remaining methods significantly differ from the nominal values. However, for lower tolerance values, all the results can be treated as acceptable. With the exception of the algorithm *HHT-I3*, all other algorithms gave results which did not differ much from one another when one assumed medium or low tolerance values.

Calculation time in the carried-out simulations proved strongly dependent on the tolerance values. The lowest impact of tolerances on computation time was observed in the algorithms *Radau* and *Radau5*. The algorithms *GAMD* and *HHT-I3* exhibited significant increase in computation time with decreasing tolerance value. In the case of algorithm *DASSL*, all the simulations lasted relatively long, and computations time needed in test I was much longer than that in tests II and III.

For this model, application of the algorithms *GAMD* and *HHT-I3* gave correct results in a shortest time (for all tolerance values). However, for the lowest tolerance values, the algorithms *Radau* and *Radau5* proved much more effective.

5. SUMMARY AND CONCLUSION

The results presented in this paper have shown significant differences in effectiveness of algorithms used for solving differential-algebraic equations in application to multibody systems. Correct results were obtained for all analysed models only when applying the algorithms *GAMD* and *HHT-I3*. The algorithms *Radau* and *Radau5* were not able to solve the system of a flexible pendulum of high stiffness, and the calculations performed for this system with the use of the *DASSL* algorithm lasted unacceptably long.

Considering the time of computation, one could conclude that the algorithm *HHT-I3* was the most effective one in the cases of simulations where tolerance values were medium or high. In this respect, the characteristics of algorithm *GAMD* were similar. The algorithm *DASSL* proved very effective in the analysis of pendulum modelled with the use of a rigid body, while in the analysis of the flexible pendulum this algorithm turned out to be the slowest one. Nevertheless, the results obtained with the use of this algorithm were often the most accurate ones. The algorithms *Radau* and *Radau5* did not differ much, neither in accuracy of computations, nor in effectiveness (in some simulations, however, the algorithm *Radau5* was much faster). These algorithm proved to be effective in cases of low tolerance values.

On the basis of the presented results we can state that, for solving a multibody system, one should apply, in the first place, the algorithms *GAMD* or *HHT-I3*. If it turns out that these algorithms have too low effectiveness, one can try to apply the algorithm *Radau5,* which in some cases may give a higher effectiveness. The algorithm *Radau* is based on the same method of solving DAEs as the algorithm *Radau5*. However, we have not notice any substantial advantages of using this algorithm in comparison to the algorithm *Radau5*. Unlike the previous ones, the algorithm *DASSL* turns out to be a good one only in the cases of systems with low number of degrees of freedom, for example when analysing a rigid body. In the analysis of flexible systems, this algorithm was usually the slowest one.

It should be emphasised that this publication does not fully exhaust the presented topic. The works are carried out, aimed at complementing the results with the analyses of more complex examples. The algorithms have been tested, so far, only on relatively simple models; neither have we taken into account such properties of integrating procedures as, e.g., the ability of solving equations with inconsistent initial conditions. The mentioned problems are beyond the scope of this publication and need further analyses.

REFERENCES

1. **Bathe K. J.** (1996), *Finite Element Procedures*, Prentice Hall, New Jersay.
2. **Brenan K. E., Campbell S. L., Petzold L. R.** (1996), *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia.
3. **Frączek J.** (2002), *Modelowanie mechanizmów przestrzennych metodą układów wieloczłonowych*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa.
4. **Frączek J., Malczyk P.**, Parallel Divide and Conquer Algorithm for Constrained Multibody System Dynamics based on Augmented Lagrangian Method with Projections-based Error Correction, *Nonlinear Dynamics*, DOI: 10.1007/s11071-012-0503-22012, to be printed.
5. **Gear C. W., Leimkuhler B., Gupta G. K.** (1985), Automatic integration of Euler-Lagrange equations with constraints, *Journal of Computational and Applied Mathematics*, Vol. 12-13, No. 0, 77-90.
6. **Gerstmayr J., Matikainen M., Mikkola A.** (2008), A geometrically exact beam element based on the absolute nodal coordinate formulation, *Multibody System Dynamics*, Vol. 20, No. 4, 359-384.
7. **Gerstmayr J., Shabana A. A.** (2006), Analysis of Thin Beams and Cables Using the Absolute Nodal Co-ordinate Formulation, *Nonlinear Dynamics*, Vol. 45, No. 1, 109-130.
8. **Hairer E., Wanner G.** (1996), *Solving oridinary differential equations II. Stiff and differential-algebraic problems*, Springer-Verlag, Berlin.
9. **Haug E. J.** (1989), *Computer-Aided Kinematics and Dynamics of Mechanical Systems, Volume I: Basic Methods*, Allyn and Bacon, Massachusetts.

10. **Hussein B., Negrut D., Shabana A. A.** (2008), Implicit and explicit integration in the solution of the absolute nodal coordinate differential/algebraic equations, *Nonlinear Dynamics*, Vol. 54, No. 4, 283-296.
11. **Iavernaro F., Mazzia F.** (1998), Solving ordinary differential equations by generalized Adams methods: properties and implementation techniques, *Applied Numerical Mathematics*, Vol. 28, No. 2-4, 107-126.
12. **Nikravesh P. E.** (1988), *Computer-Aided Analysis of Mechanical Systems*, Prentice Hall, New Jersey.
13. **Shabana A. A.** (1997), Definition of the Slopes and the Finite Element Absolute Nodal Coordinate Formulation, *Multibody System Dynamics*, Vol. 1, No. 3, 339-348.
14. **Shabana A. A.** (2003), *Dynamics of Multibody Systems*, Cambridge University Press, Cambridge.
15. **Shabana A. A.** (2008), *Computational Continuum Mechanics*, Cambridge University Press, Cambridge.
16. **Shabana A. A., Yakoub R. Y.** (2001), Three Dimensional Absolute Nodal Coordinate Formulation for Beam Elements: Part I and II, *Journal of Mechanical Design*, Vol. 123, No. 4, 606-621.

Acknowledgements: This work was realised as a part of research project No. N N514 673340 financed by the National Science Centre.