PROGRAMMING THE TRANSIENT EXPLICIT FINITE ELEMENT ANALYSIS WITH MATLAB

Andrei Dragoş Mircea SÎRBU[1], László FARKAS[2]

Modern research in automotive crashworthiness relies extensively on explicit finite element analysis. Crash analysts require deep understanding of finite element method and explicit time integration routines. This work presents the most important elements of the theory behind the mathematical apparatus of transient explicit finite element method and explains the loop algorithm of model analysis commonly used in commercial and publicly available solvers.

Correctitude of the presented concepts and methodology are verified by using a computation example of a simple lumped mass spring system subjected to impact loading. The model is analysed both by using a commercial crash solver and by using a proposed MATLAB impact algorithm.

Keywords: FEA, explicit analysis, lumped mass system

1. Introduction

The mathematical apparatus used for describing the continuum evolved to reach today a high level of complexity with multiple formulations. Simulation environment gained increasing importance in automotive design work, sustained by rapidly growing technological capabilities of today. Computational cycle time and costs are also continuously reduced by sustainable improvements in computer aided design and analysis tools.

In a similar manner, finite element modelling capabilities evolved with technology, advancing with exponential growth of model detail throughout the last decades. Crashworthiness research, sharing an important part in automotive passive safety, requires considerable resources for design, modelling, analysis and testing procedures (model validation).

Explicit formulation of the finite element method is used for crash analysis, having a nonlinear transient dynamic character. Basic finite element method concepts will be presented, followed by particularities and work flow of explicit crash analysis. In the second part of the paper, a test example is used to illustrate the implementation of explicit finite element analysis into the MATLAB programming environment and its use for future research.

[1] PhD Student, Faculty of Transports, Department of Automotive Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: sarbudragos@gmail.com
[2] R&D Project Leader, Simulation Division, LMS International, Leuven, Belgium

2. The finite element method

Roughly, the history of the finite element starts from the early 1900s, when it was used for elastic bars continua using discrete equivalent formulation [1]. As time passed, the finite element method has progressed to become the most powerful and complex tool for engineering analysis. The rapid development of the finite element method (commonly abbreviated as "FEM" or "FE method") began as the supercomputers were introduced in the mid 1980's [2]. From this point on, the formulation and solution computation of mathematical models have greatly evolved, together with hardware processing technology. Understanding the processes behind finite element model analysis with the finite element method starts from the basics of continuum discretization.

Crash analysis requires a dynamic analysis formulation (body movements) and a nonlinear transient dynamic explicit type of finite element analysis. A nonlinear dependency describes the relation between the applied conditions and resulting effects (e.g. surfaces in contact change in time or large deformations of metal parts). Therefore, a nonlinear analysis is required in this case, since large amounts of plastic deformation occur in most crash scenarios. Transient (with time integration) character of the analysis is due to the movement and body deformations that are time dependent. In crash situations, the time frame is short and the analysis is highly dynamic. The explicit formulation of finite element analysis is a specific mathematical approach, mostly used for high velocity, short time frame scenarios.

Spatial approximation using the finite element method and time approximation where the ordinary differential equations are further approximated in time are computed by reaching the dynamic equilibrium equation at the end of every time step [3]. Considering $\{x\}$ as the displacement nodal values vector, $\{\dot{x}\}$ as the nodal velocities vector and $\{\ddot{x}\}$ as the nodal accelerations vector, the dynamic equilibrium equation is computed at the end of every time step and has the following formulation:

$$[M]\{\ddot{x}\} + [C]\{\dot{x}\} + [K]\{x\} = \{f\} \tag{1}$$

Where $\{\ddot{x}\}$, $\{\dot{x}\}$ and $\{x\}$ are $m \times 1$ vectors, $[M]$ is the $m \times m$ mass matrix, $[C]$ is the $m \times m$ damping matrix, $[K]$ is the $m \times m$ stiffness matrix and $\{f\}$ is the $m \times 1$ total forces vector, while m is the number of nodal degrees of freedom per finite element model [3].

2.1. Explicit formulation

In crash analysis with explicit formulation, equation (1) is integrated in time domain by applying certain simplifications. The resulting mathematical expression is solved every time step:

$$\{\ddot{x}_n\} = [M]^{-1}\left\{F_n^{ext} - F_n^{int}\right\} \quad (2)$$

Where the mass matrix [*M*] is diagonalized for fast inversion (if it is not already diagonal) by specific mathematical algorithms, creating a lumped mass matrix, with procedures described in detail in [4, 5, 6, 7]. Stiffness or damping (if considered) matrix inversion are not required. Stiffness and damping effects are included in the internal force vector $\{F_n^{int}\}$ definition [8].

Newmark's algorithm is a mathematical routine generally used in dynamic analysis [9]. The scheme approximates displacements, velocities and accelerations at time step *n+1* by using the following routine [3, 10, 11]:

$$\{x_{n+1}\} = \{x_n\} + \Delta t\{\dot{x}_n\} + \frac{1}{2}(1-\beta_2)\Delta t^2\{\ddot{x}_n\} + \frac{1}{2}\beta_2\Delta t^2\{\ddot{x}_{n+1}\} \quad (3)$$

$$\{\dot{x}_{n+1}\} = \{\dot{x}_n\} + (1-\beta_1)\Delta t\{\ddot{x}_n\} + \beta_1\Delta t\{\ddot{x}_{n+1}\} \quad (4)$$

Where β_1 and β_2 are Newmark's constants and Δt is the analysis time step. The values given to Newmark's constants reduce the equations to either an implicit, either an explicit routine. The explicit equations are obtained by setting β_1=1/2 and β_2=0. Consequently, the explicit time integration algorithm becomes:

$$\{x_{n+1}\} = \{x_n\} + \Delta t\{\dot{x}_n\} + \frac{1}{2}\Delta t^2\{\ddot{x}_n\} \quad (5)$$

$$\{\dot{x}_{n+1}\} = \{\dot{x}_n\} + \frac{1}{2}\Delta t\{\ddot{x}_n\} + \frac{1}{2}\Delta t\{\ddot{x}_{n+1}\} \quad (6)$$

Looking at the explicit routine equations, it is observed that the information at time step *n+1* is obtained by using only the information from the previous step. The character of the algorithm by directly obtaining the present information from previous known data, without the necessity of equations solving (as implicit schemes require), gives its name, i.e. explicit formulation.

Numerical stability is maintained by choosing a small time step Δt. If the time step is over-much small, the solution will require long computational time, generally with little to no improvements in accuracy. On the other hand, if the time step is too large (exceeding the minimum time step size), the solution calculation may fail because of divergence. Consequently, there must be a maximum value of the time step at which the solution can reach convergence. The integration time step Δt must satisfy the Courant condition [2, 3]:

$$\Delta t \leq \frac{l_c}{c} \quad (7)$$

Where l_c is the smallest characteristic length of all the elements and c is the speed of sound in the material. The practical example used herein consists of discrete elements. Therefore, the time step is described by the following generic condition [12, 13]:

$$\Delta t \leq \frac{\left(\sqrt{mk+c^2}\right)-c}{k} \quad (8)$$

Where m is the mass lumped in nodes, k is the spring stiffness and c is the damping coefficient. Time step treatment is performed by using complex mathematical formulations, further described in literature [3, 8, 12].

Most finite element analysis solvers based on the explicit algorithm adapted the Newmark scheme into a staggered time marching routine, in which the nodal velocities are computed at half time steps ($\{\dot{x}_{n+\frac{1}{2}}\}$) and displacements at full time steps ($\{x_{n+1}\}$) [12]. The integration algorithm is referred to as the central difference routine:

$$\left\{\dot{x}_{n+\frac{1}{2}}\right\} = \{\dot{x}_{n-\frac{1}{2}}\} + \Delta t\{\ddot{x}_n\} \tag{9}$$

$$\{x_{n+1}\} = \{x_n\} + \Delta t\{\dot{x}_{n+\frac{1}{2}}\} \tag{10}$$

The computation cycle is presented in a more detailed manner in literature [2, 8, 10, 12, 13, 14]. Fig. 1 presents the general flow chart of the central difference time integration algorithm for time step n.

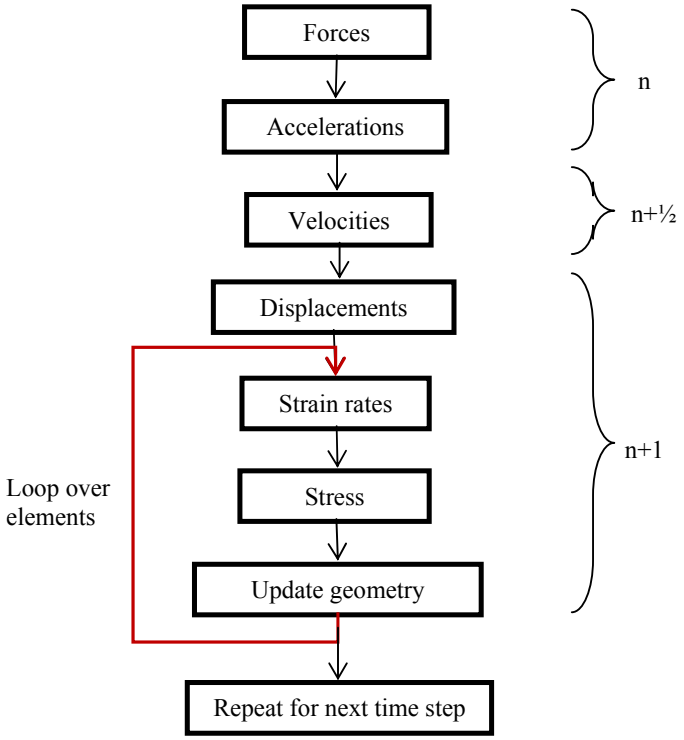

Fig. 1 – Flow chart of central difference time integration for time step n

The initiation of the time march starts from time step 0, with information known from the initial conditions. It is assumed that $\{\dot{x}_{-\frac{1}{2}}\}=\{\dot{x}_0\}$, in order to satisfy equation (8).

3. Programming an explicit impact algorithm

MATLAB [15] environment is chosen for programming the explicit finite element analysis impact algorithm. For simplicity, a simple 1D system is considered, i.e. a lumped mass spring system with three masses linked together by two springs. The three masses travel at a velocity of v_0, until the system suffers impact loading by contacting one of the rigid walls, see Figure 2.

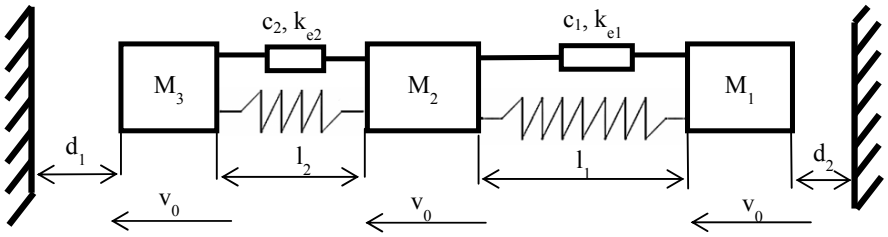

Figure 2 – Lumped mass spring system

Data values for the problem to be solved are presented in Table 1.

Table 1

Problem data

	1	2	3
Mass, M [kg]	3	2	1
Elastic stiffness, k_e [N/m]	900	500	-
Plastic stiffness, k_p [N/m]	400	800	-
Damping coefficient, c [N·s/m]	0.4	0.7	-
Yield limit, F_y [N]	200	150	-
Length, l [m]	0.5	0.3	-
Distance to wall, d [m]	0.2	0.7	-
v_0=5 m/s			

Spring material behaviour is described by a bilinear force displacement dependency with isotropic hardening, as presented in Fig. 3.

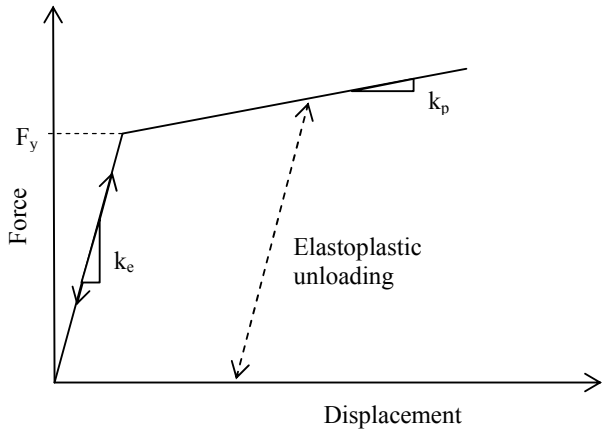

Fig. 3 – Force displacement relation describing spring elastoplasticity

Finite element translation of the abovementioned system is performed by using nodes and elements. Fig. 4 shows the discrete finite element model to be solved, where nodes are symbolized by *N* and elements by *E*. Masses are lumped in nodes. Elements have nonlinear (bilinear) spring material behaviour, with plasticity, isotropic hardening and damping properties.

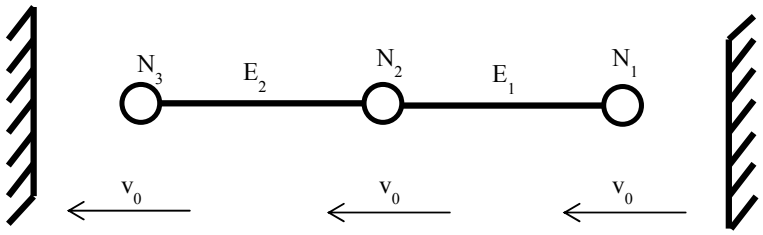

Fig. 4 – Equivalent discrete model of the lumped mass spring system

Result reference is considered to be the LS-DYNA [16] analysis of the model, which is presented in the next section.

3.1. LS-DYNA Reference

Spring behaviour is modelled by using *MAT_SPRING_ELASTOPLASTIC material, with elastoplastic translational properties and isotropic hardening [17]. It has bilinear force displacement dependence (as shown in Fig. 4), which is defined by two tangents (elastic and plastic) and a yield force [18].

Damping characteristics are described by *MAT_DAMPER_VISCOUS material [17]. The rigid walls are defined as completely rigid and constrained, allowing no deformation or penetration. Consequently, the contact is defined as rigid.

The critical time step is computed by LS-DYNA as 0.046 s for the first element and 0.027 s for the second element. However, in order to increase the analysis accuracy, a much lower time step is fixed, as 10^{-5} s. This ensures smooth data output plots and a more accurate reference data set.

The analysis is run for 2 seconds of impact time, in order to gather more data for validation. Results are pre-processed by using LS-PrePost 3.2 [19] and presented as plots in Fig. 5, Fig. 6 and Fig. 7. Time is expressed in seconds, coordinates in meters, velocities in m/s and accelerations in m/s^2.

As a convention, any vector oriented towards the left hand side has negative value, while the ones heading in the opposite direction have positive value.

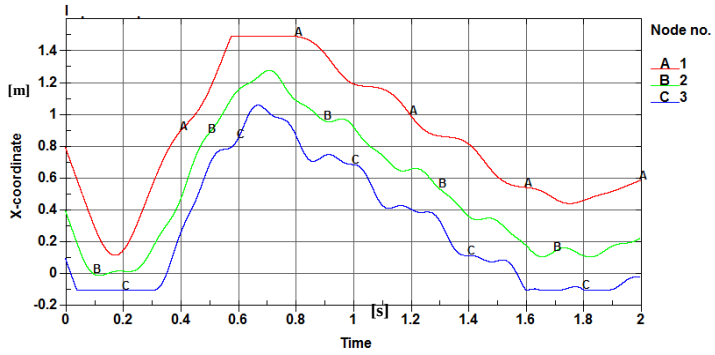

Fig. 5 – Time history of nodal coordinates in LS-DYNA

Fig. 5 shows the three impacts with the rigid walls. The model travels left and right during the 2 seconds of analysis, impacting the rigid walls at: 0.02-0.32 s, 0.57-0.8 s and finally 1.6-1.7 and 1.8-1.9 s. Fig. 6 presents the evolution of nodal velocities in time – as the model impacts the rigid walls, the velocities are being reduced by the energy absorption induced by the spring plasticity and the damping elements. The energy absorption is also shown by Fig. 7 with the decreasing acceleration peaks, which are most clearly observed from node 3.

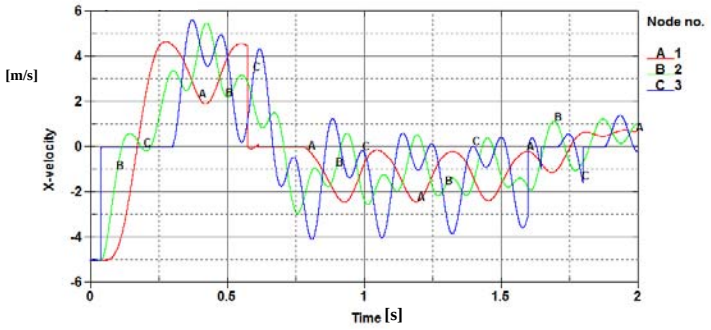

Fig. 6 – Time history of nodal velocities in LS-DYNA

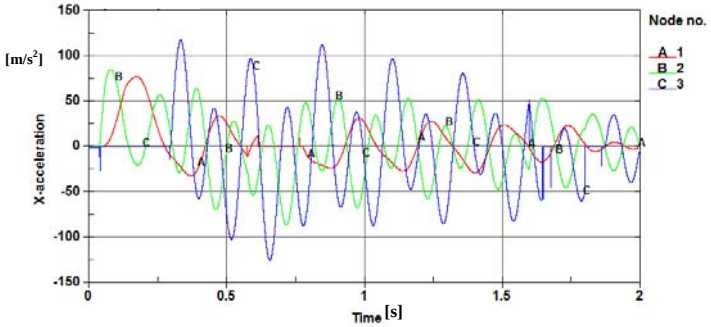

Fig. 7 – Time history of nodal accelerations in LS-DYNA

The plots containing time histories of nodal displacement, velocities and accelerations were presented in order to accurately verify correlation of all the three outputs. As the result reference was obtained, the same analysis results must be obtained by using the programmed impact algorithm.

3.2. Proposed impact algorithm

Identical model parameters (presented in Table 1) are considered for the MATLAB [15] code. The explicit finite element analysis solver is programmed within MATLAB version R2011b, including pre-processing, processing and post-processing phases. The time step is fixed throughout the analysis and identical to the reference analysis, as 10^{-5} s.

Data input and manipulation is done by using a matrix-oriented solving process, for compact and efficient information computation. Forces are checked every time step to determine the loading or unloading conditions of the elements, which are elastic or plastic. As presented in Fig. 1, a calculation loop over all

elements determines the loading conditions, resulting from the element deformation (spring force) and element deformation rate (damping force).

The contact between the model and the rigid walls is calculated by checking for penetration every time step and applying the required conditions where necessary. As stated, the contact is defined as rigid. Therefore, after each time step, if wall penetration is detected, a reaction force is computed as a function of the penetration length and the spring material. Afterwards, the node is repositioned so that penetration condition is cancelled.

Result outputs are computed for nodal coordinates, velocities and accelerations. As a code convention, element compression strain generates negative force values, while element stretching produces positive force values.

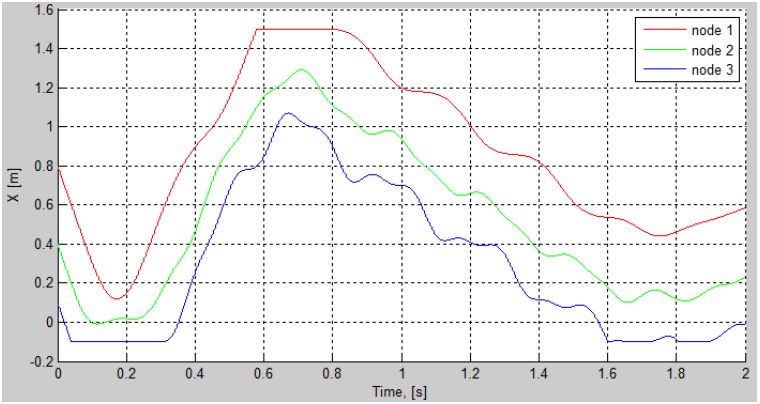

Fig. 8 – Time history of nodal coordinates in MATLAB

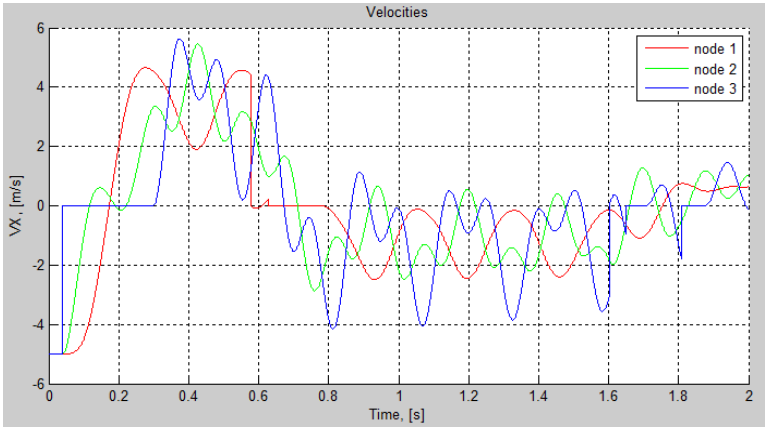

Fig. 9 – Time history of nodal velocities in MATLAB

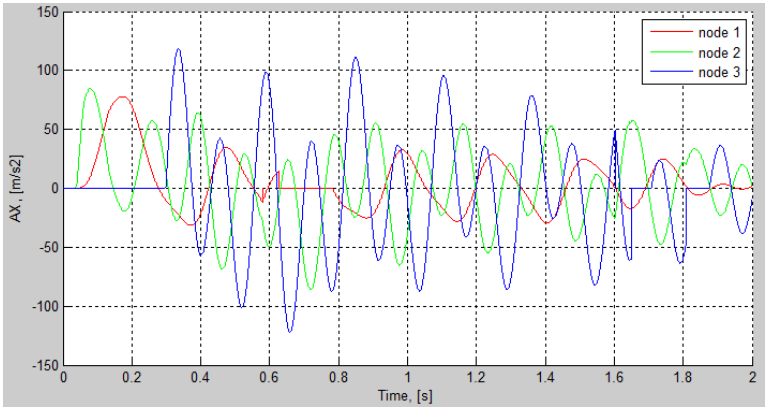

Fig. 10 – Time history of nodal accelerations in MATLAB

By comparing the result outputs generated from MATLAB shown in Fig. 8, Fig. 9 and Fig. 10 with the LS-DYNA reference results, it is concluded that the MATLAB code is perfectly validated for further research.

Initial modelling check is performed prior to the time march. Initial penetrations are verified: if any are found, the solver stops execution and returns an error message requiring rigid wall(s) or model repositioning.

The hereby programmed MATLAB explicit impact solver can perform calculations with an unlimited number of nodes and elements. The only limitations can arise from MATLAB version or license related maximum matrix dimensions that may limit the number of nodes.

4. Conclusions

This work briefly presented the mathematical apparatus of explicit crash solvers together with a flow chart of internal solver processes. The presented theory is a good base for understanding and creating any crash analysis code based on the explicit formulation of the finite element method.

MATLAB environment was chosen for programming the proposed explicit impact algorithm. It was chosen for its mathematically-oriented environment and for its code flexibility which recommends it for programming finite element analysis routines.

Couplings with other analysis concepts can be performed in future works, like coupling with the fuzzy concept for uncertainty modelling or with optimization processes. Additionally, the proposed algorithm can be used at the early stage of product design, in concept modelling, as a simplification of detailed models by using a lumped mass spring system approach.

The programmed impact algorithm can be extended towards using integrated elements and various material formulations (like adding failure parameters). Also, the modelling space can be extended to 2D or 3D environment for increased system complexity.

Acknowledgements

Andrei Dragoş Mircea Sîrbu gratefully acknowledges the PhD funding from the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88/1.5/S/61178. Additionally, Andrei Drago Mircea Sîrbu would like to thank LMS International for hosting his PhD visiting research stage at their headquarters in Leuven, Belgium.

B I B L I O G R A P H Y

[1] *S. Moaveni*, Finite Element Analysis. Theory and application with ANSYS, Prentice-Hall, USA, 1999;
[2] *P. Du Bois et.al.*, Vehicle crashworthiness and occupant protection, American Iron and Steel Institute, Michigan, USA, 2004;
[3] *J. N. Reddy*, An Introduction to Nonlinear Finite Element Analysis, Oxford University Press, USA, 2004;
[4] *A. A. Shabana*, Computational Continuum Mechanics, Cambridge University Press, 2008;
[5] *C. Felippa*, Introduction to Finite Element Methods, University of Colorado courses, retrieved in 2012 from www.colorado.edu/engineering
[6] *S. R. Wu and W. Qiu*, Nonlinear transient dynamic analysis by explicit finite element with iterative consistent mass matrix, in Communications in Numerical Methods in Engineering, **vol. 25**, no. 3, Apr. 2008, pp. 201-217;
[7] *P. Solin*, Partial Differential Equations and the Finite Element Method, John Wiley & Sons, USA, 2006;
[8] ANSYS, ANSYS LS-DYNA User's Guide Release 13.0, USA, 2010;
[9] *O. C. Zienkiewicz and R. L. Taylor*, The Finite Element Method – Fifth Edition – Volume 1: The Basics, Butterworth-Heinemann, USA, 2000;
[10] *O. C. Zienkiewicz and R. L. Taylor*, The Finite Element Method – Fifth Edition – Volume 2: Solid Mechanics, Butterworth-Heinemann, USA, 2000;
[11] *G. R. Liu and S. S. Quek*, The Finite Element Method: A Practical Course, Butterworth-Heinemann, 2003;
[12] *M. A. Crisfield*, Non-linear Finite Element Analysis of Solids and Structures – Volume 2: Advanced Topics, John Wiley & Sons, England, 1997;
[13] RADIOSS, Radioss Theory Manual 10.0, USA, 2009;
[14] *J. Bonet and R. D. Wood*, Nonlinear continuum mechanics for finite element analysis, Cambridge University Press, USA, 1997;

[15] MathWorks, MATLAB, www.mathworks.com/products/matlab, 2012;
[16] LSTC, LS-DYNA, www.lstc.com/products/ls-dyna, 2011;
[17] LSTC, LS-DYNA Keyword User's Manual, California, USA, May, 2010;
[18] LSTC, LS-DYNA Theory Manual, California, USA, March 2006;
[19] LSTC, LS-PrePost, www.lstc.com/lspp, 2012.