Eighth Eurographics Workshop on Virtual Environments (2002)
S. Müller, W. Stürzlinger (Editors)

Real-time simulation of elastic objects in Virtual Environments using finite element method and precomputed Green's functions

Igor Nikitin, Lialia Nikitina, Pavel Frolov, Gernot Goebbels, Martin Göbel
Fraunhofer Institute for Media Communication, Sankt Augustin, Germany

Stanislav Klimenko
Institute for High Energy Physics, Protvino, Russia

Gregory M. Nielson
Arizona State University, Tempe, AZ, USA

Abstract

Simulation of an object's elastic deformation is an important feature in applications where three-dimensional object behavior is explored. In addition, the benefits of user-object interactions are best realized in interactive environments which require the rapid computation of deformations. In this paper we present a prototype of a system for the simulation of elastic objects in Virtual Environments (VE) under real-time conditions. The approach makes use of the method of finite elements and precomputed Green's functions. The simulation is interactively visualized in fully immersive rear-projection based Virtual Environments such as the CyberStage and semi-immersive ones such as the Responsive Workbench. Using pick-ray interaction techniques the user can interactively apply forces to the object causing its deformation. Our interactive visualization module, embedded in VE system Avango, supports real time deformations of high-resolution 3D model (10,000 nodes) at a speed >20 stereoimages/sec.

Keywords: Elastic deformation, Virtual Environments, Finite Element Method, Green's functions

Introduction

There is a great need in many engineering and medical applications to be able to simulate the material and geometric behavior of objects under forces. Medical applications are very often concerned with the simulation of soft tissue. Pioneering work in this area has been done by Bro-Nielsen and Cotin [1] using the Methods of Finite Elements to simulate elastic deformations. Engineering applications also require accurate methods for simulation of elastic deformations. The survey paper [2] describes much of the previous work on the modeling of deformable objects. Recent advances are the application of the Boundary Element Method and fast update Sherman-Morrison-Woodbury algorithm for the simulation of linearly elastic homogeneous objects [3], and implementation of St. Venant-Kirchhoff materials [4] and modified nested dissection [5] for the simulation of non-linear elasticity.

This paper describes a system that is capable of real-time simulation of linear elastic deformations of volumetric objects in Virtual Environments which allows interactive user-defined force application.

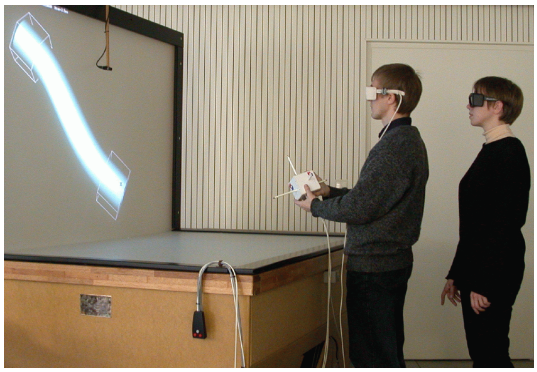

Fig.1. The two-sided Responsive Workbench.

Virtual Environment Systems

The vision of Virtual Environments is to provide single users or teams of engineers, designers, or surgeons with a virtual work space where they can view, manipulate and create in real-time virtual data of their interest. The hardware configuration used for the interactive visualization of deformable objects in the simulation approach of this paper includes input devices for interaction and rear projection display sys-

tems as output devices. In the following the most important ones, the Responsive Workbench and the CyberStage are introduced as well as created at our laboratory software framework Avango used for the visualization.

Responsive Workbench^{TM} is is a high resolution tabletop display system (fig.1), which has been developed at the Department of Virtual Environments of the *German National Research Center for Information Technology (GMD)* [6]. In this system stereo images are projected over a mirror onto a display, sloped by 20° with respect to horizontal plane. The new two-sided Responsive Workbench has a horizontal and a vertical display, which are smoothly adjacent. By extending the normal Responsive Workbench with an additional vertical screen, the viewing frustum is substantially increased and virtual objects can be observed at the user's eye level which was not possible before. Using current PC and workstation technology the two-sided Responsive Workbench is operated with a resolution of $1280\times1024\times96$Hz stereo. Users wear shutter glasses, necessary for stereo-image perception. A Polhemus six degrees of freedom sensor is attached to the shutter glasses for head tracking. This allows the system to compute the correct perspective image for any user location. Users interact directly with three dimensional virtual objects within the viewing frustum using six degrees of freedom input devices. Typical application areas of the Responsive Workbench are delivered by engineering and medical visualization.

CyberStage^{TM} is immersive audio-visual projection system [7, 8]. It has room sizes ($3\text{m}\times3\text{m}\times2.4\text{m}$) and integrates a 4-side stereo image projection and 8-channel spatial sound projection, both controlled by the position of the user's head. An SGI Onyx 2 with 4 Infinite Reality 2 graphics subsystems and 12 MIPS R12000 processors generate eight user controlled images. Each pipe generates 11 million shaded triangles per second (peak rate). The used display resolution is 1280×1024 pixels at 120 Hz for each of the four displays.

For creating the illusion of presence in virtual spaces, the CyberStage and Responsive Workbench systems provide various interfaces and interaction metaphors to visually and acoustically respond to the user's actions. These interfaces allow for navigation in virtual spaces and manipulation of virtual objects. The software driving the CyberStage and the Responsive Workbench is the Avango [9] application development toolkit.

Avango^{TM} is a programming framework for building distributed, interactive VE applications. It uses the C++ programming language to define two categories of object classes. *Nodes* provide an object-oriented *scene graph* API which allows the representation and rendering of complex geometry. *Sensors* provide Avango with an interface to the real world and they are used to import external device data into the application.

All Avango objects are *fieldcontainers*, representing object state information as a collection of *fields*. They support a generic *streaming* interface, which allows objects and their state information to be written to a stream, and the subsequent reconstruction of the object from that stream. This interface is one of the basic building blocks used for the implementation of object distribution.

Avango uses connections between fields to build a *dataflow graph* which is conceptually orthogonal to the scene graph, and it is used to specify additional relationships between nodes, which cannot be expressed in terms of the standard scene graph. This facilitates the implementation of interactive behavior and the import of real world data into the scene graph.

In addition to the C++ API, Avango features a complete language binding to the interpreted language Scheme [10]. Scheme is a general purpose programming language descended from Algol and Lisp. It is a high level language, supporting operations on structured data such as strings, lists and vectors. All high level Avango objects can be created and manipulated from Scheme.

The Avango is originally based on OpenGL Performer to achieve the maximum possible performance for an application and addresses the special needs involved in application development of Virtual Environments. Advanced rendering tasks like culling, level-of-detail switching and communication with the graphics hardware are all handled by Performer. Whenever the underlying hardware allows, Performer utilizes multiple processors and multiple graphics pipelines. Currently we use Avango implementations on SGI computers and Linux-PCs.

Simulation of elasticity. Our goal is to be able to include deformable models in Virtual Environments. In the past, a variety of approaches have been used for elastic simulation [2], but, here we focus mainly on fast and accurate methods, which are capable of deformations of large models ($>$ 5,000 nodes) at real-time speed ($>$ 20 fps). The choice of the method depends on the shape of the object (fig.2). For objects that have approximately equal length in all three dimensions (3DE objects), it is usually considered acceptable to use linear theory of elasticity, but for objects such as plates and membranes (2DE) or rods and tubes (1DE) a small force creates a large displacement and linear theory does not apply.

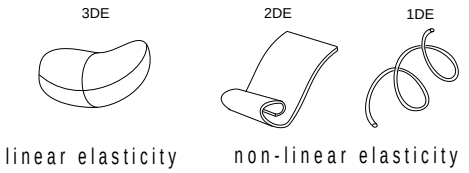

Fig.2. Different types of elastic objects.

For linear problems, there is the possibility to solve the equations of elasticity on-line and simultaneously perform the

graphical rendering, or to perform some of these computations off-line. Also, depending upon the assumptions about the internal composition of an object, there is the possibility to use the Finite Element Method (FEM) or Boundary Element Method (BEM). For 2DE and 1DE problems, there are available specially designed methods based upon non-linear theory of elasticity [11].

	linear	non-linear
on-line	FEM / BEM	FEM, special 1DE,2DE methods
off-line	FEM / BEM	-

We now set out to briefly describe these various possible approaches. At the same time, this discussion will lead to a description of the methods and technique we have developed for our present system which concentrates on the approach of using linear FEM with pre-computing a complete basis of solutions (Green's functions).

On-line vs off-line. Linear problems in theory of elasticity require a solution of large linear systems of the form $Ku = f$ with constant matrix K and variable right hand side f. Using on-line methods, one can achieve the real-time performance for moderately large models, typical figures are given in [4]: 1,400 nodes at 45 fps for PC Pentium III 500 MHz. Using off-line inversion of matrix K and representation of solution as $u = K^{-1} f$, we obtain better performance: 10,000 nodes at 85 fps for PC Athlon 1.3 GHz. The advantage of on-line approach is a possibility to solve non-linear problems, according to [4], this can be done at nearly real-time speed: 1,400 nodes at 8 fps. One more property, which is usually considered as an advantage of on-line approach, is a possibility to perform interactive change of system matrix K, including those associated with variation of boundary conditions and change of topology (cuts) of the model. However, in recent works [3, 12] this feature has been implemented for pre-computed models as well, using a fast update algorithm for evaluation of K^{-1}.

FEM vs BEM. Finite Element Method [13] subdivides the body to a finite set of primitives, using e.g. tetrahedral mesh, with subsequent definition of a physical equilibrium for each of these elements. Boundary Element Method [3] uses analytical reformulation of original partial differential equations in the theory of elasticity to an integral form, which includes only the surface variables (displacements and tractions). In BEM only the surface of the object should be meshed, practically one can take the same triangulation as used for the rendering. In FEM the interior also should be meshed, even in the case if it should not be visualized. The application of BEM is restricted to the objects with homogeneous interior. The objects with complex internal structure can be processed only by FEM. Linear systems, generated by FEM, are large and sparse, while the equivalent BEM systems are small and dense. Both methods can be used in pre-computation mode to produce the Green's functions, which can then be passed on to the on-line part of the simulator.

Linear FEM pre-computation of Green's functions. Deformation is described by a linear system of the form $Ku = f$, where f are forces, acting in the nodes of the mesh, u are unknown displacements of the nodes, K is stiffness matrix, defined by material properties and shape of the body. The matrix K is symmetric, very large and sparse, containing less than 1% nonzero entries, see fig.4. These data can be efficiently stored in data structures related to the mesh itself. The diagonal elements K_{ii} are stored in the nodes of the mesh, while the non-diagonal elements K_{ij} are stored on the edges.

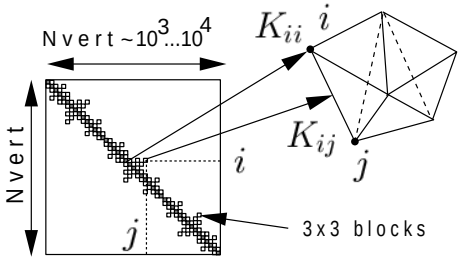

Fig.4. Storage of stiffness matrix.

Solution of the system is performed using standard techniques, such as preconditioned conjugate gradient method (PCG) [1, 14]. In pre-computation scheme [1] the matrix K is inverted in off-line mode, and the solution is determined on-line as $u = K^{-1} f$. The main problem with this approach is that the matrix K^{-1} is no longer sparse which impacts storage requirements and the cost of computing the product $K^{-1} f$. Fortunately, not the whole matrix needs to be stored, but only those rows, which correspond to the visible nodes and only those columns, which correspond to the nodes, where the external force can be applied:

$$\begin{bmatrix} u_S \\ * \end{bmatrix} = \begin{bmatrix} (K^{-1})_S & * \\ * & * \end{bmatrix} \begin{bmatrix} f_S \\ 0 \end{bmatrix}$$

For most of the applications it is sufficient to extract from K^{-1} the block $(K^{-1})_S$ of the size (3Nsvert)×(3Nsvert), corresponding to the surface nodes. For this purpose we solve the system $Ku = f$ with $f_i = \delta_{ik}$, where k is restricted to the surface nodes. The solution is computed using the PCG method. The obtained solutions $u_i^{(k)}$ (the columns of K^{-1}) are written to a file, for $i \in$ surface nodes. Notice that internal nodes contribute to the elements of K^{-1} during off-line stage of computations, and then can be omitted. The elements of obtained $(K^{-1})_S$ matrix describe how the influence of the unit force applied to a surface node propagates through

the body to other surface nodes. Mathematically, this coincides with the Green's function $G(x,y)$ which describes the propagation of the influence from point x to point y. From a rendering point of view, we are only interested in the values of these functions for the surface points.

The amount of memory needed to store $(K^{-1})_S$ is now reduced to an acceptable value (6Mb ... 450Mb for the number of surface vertices in the interval 600 ... 5000). However the computation of the product $u_S = (K^{-1})_S f_S$, which is necessary to perform on-line, is generally still expensive. Further optimization is possible if one uses extra assumptions about the character of force distributions. In particular, if the force is localized in a few nodes of the surface, the accelerated multiplication algorithm [1] can be used. For the force, localized in a single node, it is just necessary to take the corresponding column from $(K^{-1})_S$. In the other case the force is distributed in wider regions of the surface, but *few types* of force distributions are introduced, representing different types of interaction (stretch, shift, twist, etc). In this case there are few linear combinations of the columns in $(K^{-1})_S$, which can be computed off-line, stored, and combined on-line using non-expensive computation (in the simplest case reduced to the extraction of the necessary column from the matrix). One more example, appearing in engineering problems, is the deformation of elastic body, defined by few control elements, attached to the boundary. We will consider this model in more details.

Elastic deformation using control elements

Consider a boundary $\Gamma = \partial\Omega$ of the body and a region $\Gamma_b \subset \Gamma$ of the boundary, where non-zero displacements are set: $u = b$. The problem is to find the displacements u on $\Gamma \backslash \Gamma_b$. Such boundary problems are reduced to the problems of type "the force is given", using the following decomposition of matrix K:

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} u \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ f_b \end{bmatrix}$$

where f_b is the force, acting on Γ_b, while for other nodes no external force is assumed. We obtain the following two systems: $B^T u + Cb = f_b$, the definition of the force f_b, usable e.g. for haptic simulation [14, 15]; $Au = -Bb$, system for u, used for the visualization of the deformed object.

Note: the haptic system has smaller size and its solutions also can be precomputed. In the contrast to visual system, where the solution should be evaluated in each node of the surface, the haptic system requires the evaluation of a single vector of the force. Practically this allows to compute the haptic force at much higher update rate than solution of visual system, achieving the performance necessary for haptic devices (>1kHz). In this case the haptic simulation part should work in a separate thread [15] or even as a master application, transmitting the input data to slower visualization module.

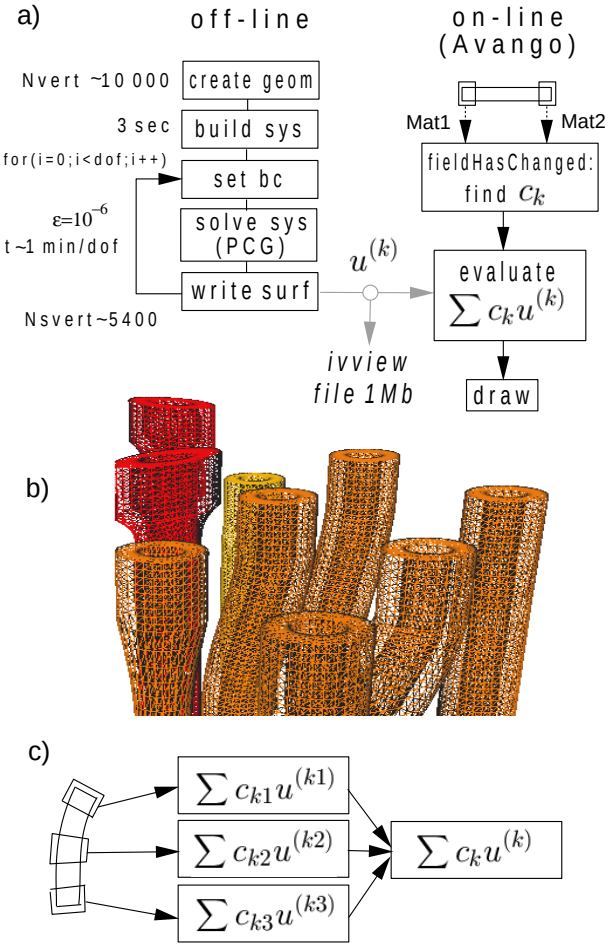

Fig.7. a) scheme of simulation; b) basis of solutions (iv-file); c) separability of computations in on-line module.

Now suppose that the displacements b are given by affine transformations $\vec{b} = \vec{T} + M\vec{b}_0$. This transformation is described by 12 parameters, namely 3 for the translations $\vec{T}$ and a 3×3 matrix of general linear transformation M (including 3 rotations and 3 scalings in particular), defining a 12-dimensional space per a control element. Let $b^{(k)}$, $k = 1..12$ be a basis in this space, corresponding to the unit placed sequentially to the entries of $\vec{T}$ and M, while other entries are filled by zeros. Let $u^{(k)}$ be corresponding solutions of the system, i.e. $u^{(k)} = -A^{-1}Bb^{(k)}$. Then, due to linearity of the system, the shape of the elastic body for a given interaction $b = \sum_k c_k b^{(k)}$ will be $u = \sum_k c_k u^{(k)}$. Actually, we need to precompute solutions for elementary interactions, corresponding to the basis vectors $b^{(k)}$, in the off-line mode, and then combine these solutions to find the shape of the body for any given interaction, using non-expensive on-line computation.

The amount of memory per control element required to store the solutions $u^{(k)}$ is small (100Kb ... 700Kb). The scheme of simulation is shown in fig.7. For 10,000 nodes model the pre-computation takes about 20 min. The resulting solutions $u^{(k)}$ are written to a file, of 1Mb size in a compressed form. We use the Inventor file format to control the solutions visually and to simplify the data transfer to the Virtual Environment. The visualization is performed by an on-line module, using the Avango Virtual Environment framework.

Implementation in Avango: Interaction with virtual objects in large-scale VE systems is performed using electromagnetic 3D pointing device (stylus), visually represented in the virtual world, e.g. by a green ray. Avango supports a useful feature, called *MatrixDragger*, which on a low level detects the intersection of the ray with the object and copies the matrix of the interaction device to the object's matrix, this allows easy positioning of virtual objects. In our application the draggers are attached to the control elements and their matrices are submitted to our visualization module using field connections. Whenever matrices are changed the method *fieldHasChanged* is activated, which extracts the coefficients c_k from the entries of the matrices. The computation of the sum $\sum_k c_k u^{(k)}$ is postponed to the *evaluate* method, activated only once per frame if any of the fields has been changed. Various methods can be used to speed up this computation. For example, if only one interaction device is used, only one control element can be displaced per the act of interaction, so that only its related coefficients c_k are changed and only the corresponding part of the sum should be recomputed. Additional possibilities for the acceleration are the usage of pointer arithmetics, placement of the data to fast accessible memory (cache) and parallelization of the computation. The described model can be interactively deformed in the Virtual Environment at a graphics speed of 20fps for SGI/Onyx2 300MHz MIPS R12000. The performance characteristics for Athlon 1.3GHz Linux PCs are better (85fps).

Conclusion

In this paper we described our real-time capable approach for the simulation of elastic deformable objects. The approach uses the finite element method to solve the equations of elasticity. The most time consuming portions of the computations are performed in off-line mode. The resulting data, saved to a file, allows the acceleration of the on-line simulation process by a factor of one thousand and leads to an interactive visualization module operating in Virtual Environments at real-time speeds.

Acknowledgments. This work has been partially supported by RFBR grants: projects 02-01-01139 and 01-07-90327. We also wish to acknowledge the support of the National Science Foundation (IIS-9980166 & ACI-0083609), Office of Naval Research (ONR N00014-00-1-0281) and Defense Advance Research Projects Administration (DARPA MDA972-00-1-0027).

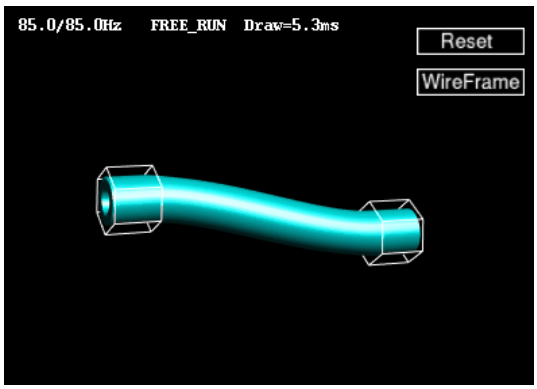

Fig.8. interactively deformable tube model in the Avango viewer.

References

1. Bro-Nielsen M., Cotin S., Real-Time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation, Eurographics'96, Vol.15 (1996) No.3, C57-C66 (note: in Eq.(23) the cyclic interchange of the indices should be followed by change of the sign).
2. Gibson S.F., Mirtich B., A survey of deformable models in computer graphics, Technical Report TR-97-18, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.
3. James D., Pai D., Artdefo Accurate Real Time Deformable Objects, Computer Graphics, vol.33, pp.65-72, 1999.
4. Picinbono G., Delingette H., Nicholas H.-A., Non-Linear Anisotropic Elasticity for Real-Time Surgery Simulation, INRIA Research Report 4028, October 2000.
5. Y.Zhuang, J.F.Canny, Real-time global deformations, In Algorithmic and Computational Robotics, pp.97-107, Natick MA, 2001. A.K.Peters.
6. Krüger W., Bohn C., Fröhlich B., Schüth H., Strauss W., Wesche G., The Responsive Workbench: A Virtual Work Environment, IEEE Computer, pp. 12-15, 1994.
7. Eckel G., Göbel M., Hasenbrink F., Heiden W., Lechner U., Tramberend H., Wesche G., Wind J. Benches and Caves. In: Bullinger H.J., Riedel O. (eds.) Proc. 1st Int. Immersive Projection Technology Workshop. Springer-Verlag, London, 1997.
8. Cruz-Neira C. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. Computer Graphics Proc., Annual Conference Series, 1993, pp.135-142.
9. Tramberend H., Avocado: A Distributed Virtual Reality Framework, *Proc. of the IEEE Virtual Reality, 1999.*
10. R. Kent Dybvig. The Scheme programming language: ANSI Scheme. P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, Edition 2, 1996.
11. L. D. Landau, E. M. Lifschitz, Theory of Elasticity, volume VII of Theoretical Physics, Moscow, Nauka, 1970.
12. U. Meier et al, Real-Time Simulation of Minimally-Invasive Surgery with Cutting Based on Boundary Element Methods, Lecture Notes in Computer Science (2001) V.2208, p.1263.

13. O. C. Zienkiewcz, R. L. Taylor, The Finite Element Method, Vol.1, Edition 5, Butterworth-Heinemann, Oxford 2000 (note: in Eq.(6.5b) the cyclic interchange of the indices should be followed by change of the sign).

14. A. O. Frank, I. A. Twombly, J. D. Smith, Finite Element Methods for real-time Haptic Feedback of Soft-Tissue Models in Virtual Reality Simulators, Proceedings of VR2001.

15. J. Berkley et al, Banded Matrix Approach to Finite Element Modeling for Soft Tissue Simulation, Virtual Reality: Research, Development, and Applications (1999) 4:203-212.

16. Hirota, G., Fisher, S. and Lin, M., Simulation of Non-penetrating Elastic Bodies Using Distance Fields. UNC Technical Report TR00-018, 2000.

Appendix: Beyond the small strain theory

There are two types of non-linearity, which can complicate the simulation: material and geometrical ones. The material non-linearity, caused by violation of stress-strain linear relation (Hooke's law), usually, at least in engineering applications, occurs only under extremal conditions when the material looses the elastic properties. The geometrical non-linearity occurs at large displacements of points in the body relative to each other and is associated with a non-linear term of purely geometrical nature, appearing in the strain tensor [11]. For 3DE-bodies (fig.2) the geometrical non-linearity is usually negligible, because the large relative displacements for this kind of bodies can appear only under strong stress. In the contrast, for the deformations of long rods (1DE-bodies) and cylindric bending of thin plates (2DE-bodies) the large displacements are possible without creation of much stress. A significant part of the elasticity theory is dedicated to these special cases.

The most of developed numerical methods for simulation of elasticity are based on the linear small strain theory. For 1DE or 2DE objects the solutions of linear theory possess non-physical properties. In particular, if one end of the long rod is fixed, while another one is twisted about its axis, performing one complete revolution, the boundary condition b is returned to the non-deformed state, so that the solution of small strain theory $u = -A^{-1}Bb$ is also continuously returned to the non-deformed state, in a complex way penetrating through itself. This problem is caused by the omitted quadratic term in the definition of the strain tensor [11], which is not small for the considered displacements. This term is also responsible for the complete rotational invariance, so that the energy in the linear theory is invariant only under infinitesimal but not under finite rotations [4]. Incorrect behavior of linear system with respect to local rotations of the parts of the body relative to each other is a kernel of the problems, associated with the non-linear effects.

Below we summarize several approaches capable to take account of geometrical non-linear effects in the simulation. We separate the approaches to those which can be applied on-line only, those which can use the advantage of pre-computation and the composite ones.

On-line methods:

1a) use strain tensor with quadratic term (St.Venant-Kirchhoff elasticity), as proposed by [4];

1b) introduce such a system of local coordinate frames [11], where the quadratic term in strain tensor is small and can be neglected.

Pre-computation methods (applicable for moderately large rotation angles, e.g. for more precise simulation of 3DE bodies):

2a) in the scheme, using control elements, extract the average rotation matrix of the set of control elements and transfer it to the rigid rotation of the whole model. This will reduce the rotation angles of control elements relative to the average position, significantly decreasing non-linear terms (which are proportional to the angles squared).

2b) solve non-linear system by means of perturbation series with respect to non-linear term, pre-computing sufficient number of coefficients of expansion.

Composite methods:

3a) subdivide 1DE and 2DE bodies to a set of 3DE parts, for which the small strain theory is applicable and whose behavior can be described in terms of precomputed Green's functions, link the parts along the boundaries, express the energy of the system in terms of a smaller set of boundary coordinates and minimize it on-line using an appropriate non-linear minimizer [16].

3b) considering the non-linear behavior of long rods and thin plates, use the methods, based on partially analytical resolution of elastic PDEs, done in the non-linear theory of elasticity [11].