periodica polytechnica

Civil Engineering
58/1 (2014) 23–31
doi: 10.3311/PPci.7032
http://periodicapolytechnica.org/ci

RESEARCH ARTICLE

Parallel scanning of implicit surfaces with the simplex algorithm

Róbert K. Németh

Received 2013-08-08, revised 2013-11-10, accepted 2013-12-23

Abstract

Solution of mechanical problems often requires the analytical or numerical calculation of equilibrium paths, while multidimensional solution sets are rare. From this requirement emerged numerous methods for the calculation of bifurcation diagrams. Two large groups of solution methods are the continuation methods and the scanning methods (however hybrid algorithm exists as well). The Simplex Algorithm is a robust approximative technique based on the Piecewise Linearization (PL)–algorithm, which has its application as a continuation and as a scanning algorithm as well. In this paper we will show the extension of the method for finding a 2-dimensional manifold (i.e. surface) with the scanning of the parameter space. We analyze the performance of the algorithm and its parallelization through two simple examples.

Keywords

scanning · equilibrium surface · piecewise linearization · simplex algorithm

Róbert K. Németh

Budapest University of Technology and Economics, Department of Structural Mechanics, Műegyetem rkp. 3, H-1111 Budapest, Hungary

e-mail: methro@eik.bme.hu

1 Introduction

In many engineering problems the equilibrium paths must be found in order to analyze the system's behavior. There are analytical methods to find the equilibrium path for many problems, but increasing complexity (geometrical and material nonlinearities, imperfections etc.) leads to necessity of numerical methods. There are two main types of numerical methods. Path-following techniques require a known solution point to start from. Most existing algorithms belong to this group, such as the Newton-Raphson iteration, the collocation method used in the program AUTO ([7] and [8]), and the approximate PL-algorithm ([1], [10]). Depending on the method of choice these techniques are able to find the bifurcation points along the path and decide which path to follow after that. The second type of numerical methods are the scanning methods. Scanning methods find all solutions inside a given region of the parameter-space, do not need any starting point, and without any further effort they give the bifurcations along the branches as well ([13] and [11]). Hence scanning methods do not need a starting point, and they are able to find disconnected branches as well.

For a few problems the solution set forms a two-dimensional manifold. This can happen when the loading of a structure depends on two independent parameters, or when we are looking for the equilibrium paths as a function of an additional structural parameter. An example is the clamped-clamped rod with a compression force and a twist moment at the end. In the space of parameters, that defines physical configurations uniquely, the points that correspond to equilibrium configurations are forming surfaces. Beyond analytical methods for finding these kinds of surfaces numerical algorithms can also be used. There are various methods for the surface following of these types of problems, differing in the shape of the surface elements of the continuation (see e.g. [4] and [15]). It is possible to find a whole surface inside a given part of the parameter space as long as we are able to define a starting point for the continuation. It is also possible to find the bifurcation points or lines on the surfaces. But, as usual with the continuation algorithms, it cannot find non-connected equilibrium points. If there are such paths with unknown starting points, one has to scan the parameter space

for solutions. Scanning plane sections of the parameter space can be a solution for this problem, but it gives no direct information about the connectedness of the lines between two plane section. In this paper we will introduce a robust, iteration free algorithm for the direct computation of such surfaces.

Outline of the paper is as follows. In Section 2 we will present the key idea of the Simplex Algorithm for finding 2-dimensional solution manifolds based on the 1-dimensional algorithm. At the end of the section we analyze the aspects of the parallelization of the computation. In Section 3 we present simple applications with numerical results. First, with a slight modification of the compressed-twisted rod problem we will analyze the equilibrium surfaces of the hemitropic rod. Second, the effect of bending springs on the elastic web of links will be analyzed. In Section 4 we analyze the speed-up of the parallelization, and at the end we draw the conclusions.

2 Parallel scanning for solutions of a 2D manifold

2.1 Error-functions in the Global Representation Space and division of the GRS

In some of the engineering problems the solution is not characterized by a line, but a surface. The solution surface can be a result of the continuous symmetry of the problem, but this should be avoided by a proper choice of variables and equilibrium conditions (see [12], and [20]). Other possibilities of a solution surface are the loading with a two-parameter load (see [19]) and the loading a structure with a one-parameter load while the system properties vary with respect to another parameter.

We restrict our analysis to the following problems. The equilibrium configuration of the structure can be described uniquely by n parameters. We refer to the $\mathfrak{R}^n$ space spanned by those parameters as the Global Representation Space (GRS). Any point in this space defines a configuration. We call a configuration an equilibrium configuration when it fulfills all equilibrium equations and boundary conditions.

For the equilibrium of a configuration we define the error-function

$$\mathbf{f}^*(\mathbf{X}) : \mathfrak{R}^n \to \mathfrak{R}^{n-2}, \tag{1}$$

which has its fix-points in the equilibrium configurations. The equilibrium surface is defined by the set of all points $\mathbf{X} \in \mathfrak{R}^n$, where $\mathbf{f}^*(\mathbf{X}) = \mathbf{0}$. In the Piecewise Linearization Algorithm the GRS is divided into simplices, and the $\mathbf{f}^*$ function is linearly approximated inside each simplex [1]. The approximation is based on the values of the function calculated in the vertices of the simplex.

A simplex is the simplest linear object occupying a measurable part of an n-dimensional space. It has $n+1$ vertices and facets. Each facet is a simplex on an $n-1$-dimensional hyperplane, containing n vertices of the simplex, i.e. only one vertex of the simplex is out of the facet, so we can identify each facet of the simplex by the non-contained vertex. Let us denote the coordinates of the $n+1$ vertices of the simplex in the GRS by $\mathbf{X}_k, (k = 0, \ldots, n)$. We refer to the facet of the simplex opposite to the kth vertex as the kth facet.

The GRS is divided into simplices in two steps. First we divide the space into orthotopes ([6]) to which we will refer to as hyperrectangles with the edges as vectors $\mathbf{b}_i, (i = 1, 2, \ldots, n)$ pointing parallel with the coordinate-axes of the GRS. (These hyperrectangles are often referred to as hypercubes or n-cubes, as it is done in [3], but the edges do not necessarily have the same length or even unit, so we use the term hyperrectangle here.)

Next we divide each hyperrectangle into $n!$ simplex as follows. We choose one node of the hyperrectangle as an origin, its global coordinates will be collected in the vector $\mathbf{X}_0$. A permutation (without repetition) of the numbers $1, 2, \ldots, n$ is denoted by $p_1, p_2, \ldots, p_n$. Each of these permutations defines one simplex inside the hyperrectangle. We refer the vectors pointing from the $\mathbf{X}_0$ vertex to the other vertices of the simplex as the $\mathbf{d}_i$ directors of the simplex. The directors with a given permutation can be calculated as:

$$\mathbf{d}_i = \sum_{k=1}^{i} \mathbf{b}_{p_k} \tag{2}$$

To scan the GRS we have to linearize the error-functions in each simplex of each hyperrectangle. The linearization is based on the calculated values of the error-function in the vertices of the simplex. Then we have to check whether the solution has a segment inside the simplex or not. Due to the linarization it is a simple task, but the increase in the dimension of the problem results in an exponential increase in the number of simplices and hyperrectangles, leading in the end to an enormous computational need.

Before showing each above step in detail, we have to note in advance the following:

- As some vertices of every simplex inside a hyperrectangle coincide, it is recommended to compute the error-functions in all vertices of the hyperrectangle first, and then just take the necessary values to each simplex from the stored set.
- The computation of the hyperrectangles are independent of one another, which suggests the possibility of parallelization of the problem. In the parallel method different regions of the GRS are computed by different program threads, or even computers.
- While computing a neighboring hyperrectangle of an earlier computed hyperrectangle the error-functions are known in the common vertices. Using these stored values instead of calculating them again is definitely a further possibility to decrease the computation time.

2.2 Finding the solution inside a simplex

Any point in the n-dimensional GRS can be represented as:

$$\mathbf{X} = \mathbf{X}_0 + \sum_{i=1}^{n} x_i \mathbf{d}_i, \tag{3}$$

where $\mathbf{X}_0$ is the initial point of the hyperrectangle, the set of $\mathbf{d}_i$ directors identify the simplex inside the hyperrectangle and the n components of the vector $\mathbf{x}$ identify the point. The point $\mathbf{x}$ is inside the simplex or on its one or more facets, if:

$$\sum_{i=1}^{n} x_i =< 1, \text{ and} \tag{4}$$

$$x_i >= 0, \qquad i = 1, \ldots, n. \tag{5}$$

The equal sign of Eq. (4) means that the point $\mathbf{x}$ is in the hyperplane of the 0th facet of the simplex. The equal sign of Eq. (5) with $i = j$ means that the point $\mathbf{x}$ is in the hyperplane of the jth facet of the simplex. If every inequality of Eqs. (4)-(5) holds and the point lies in the plane of one facet (i.e. exactly one equality holds), then we say that point $\mathbf{x}$ is on the corresponding facet. If every inequality of Eqs. (4)-(5) holds and exactly two equality holds, then we say that point $\mathbf{x}$ is on a *hyperedge* of the simplex.

The linearization of the error-functions $\mathbf{f}^*(\mathbf{X})$ inside the simplex is based on the error-function values in the vertices of the simplex. The linearized error-function is denoted by $\mathbf{f}(\mathbf{x})$. For brevity we will denote the $\mathbf{f}^*(\mathbf{X}_i)$ values in the ith vertex by $\mathbf{f}^i$. Based on the $n+1$ vertices of the simplex the linear approximation of the error-functions is:

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}^0 + \sum_{i=1}^{n} x_i(\mathbf{f}^i - \mathbf{f}^0). \tag{6}$$

The $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ condition represents $n-2$ linear equations for the n unknown of $\mathbf{x}$. The solution set of these equations is typically a single 2-dimensional plane. In a general case the plane crosses each hyperedge, or its extension in one point. If this point is on the hyperedge, then the linearized solution plane goes through the simplex and has solution points on further hyperedges as well. These solution points form a convex polygon. The number of the vertices of the polygon depends on the state of the plane with respect to the simplex.

It was shown earlier that the hyperedge of every pair of facets can be characterized by a system of two linear equations. Every node of the solution polygon fulfills Eq. (6) and the equations of two facets. This gives a total of n equations of n unknowns, which can be solved using standard methods, like the Gaussian elimination with partial pivoting. The resulting x_i values represent one point of the linearized solution plane on two facets analyzed. If this point is *not* outside the simplex (i.e. Eq. (4)-(5) holds), then x_i is on both facets of the simplex so we can store the solution point $\mathbf{X} = \mathbf{X}_0 + \sum_{i=1}^{n} x_i \mathbf{d}_i$. (From numerical point of view, the constraints should be handled as $\sum_{i=j}^{n} x_j \leq 1 + \varepsilon$, and $x_i \geq -\varepsilon, (i = 1, \ldots, n)$ with a numerically small ε.) The Gaussian elimination must be produced for each combination of facet pairs of the simplex with the check of the solution at the end.

The $n+1$ facets of the simplex can be paired in $(n+1)n/2$ different ways. These $(n+1)n/2$ systems of linear equations differ only in their last two equations. If the first $n-2$ pivots of the Gaussian elimination are chosen only from the first $(n-2)$ equations, then the same manipulation is applied in those first $(n-2)$ steps of the operation. This equivalence can be applied for a numerically more efficient technique, where we make these $(n-2)$ steps on all equations of Eq. (6) and on all facets' equations. In order to do that we form the following system of linear equations:

$$\left[\frac{\mathbf{F}^0}{\mathbf{B}^0}\right][\mathbf{x}] = \left[\frac{\mathbf{g}^0}{\mathbf{e}^0}\right] \tag{7}$$

with

$$\mathbf{F}^0 = \begin{bmatrix} f_1^1 - f_1^0 & f_1^2 - f_1^0 & . & . & f_1^n - f_1^0 \\ f_2^1 - f_2^0 & f_2^2 - f_2^0 & . & . & f_2^n - f_2^0 \\ . & . & . & & . \\ . & . & & . & . \\ f_{n-2}^1 - f_{n-2}^0 & f_{n-2}^2 - f_{n-2}^0 & . & . & f_{n-2}^n - f_{n-2}^0 \end{bmatrix}, \tag{8}$$

and

$$\mathbf{B}^0 = \begin{bmatrix} 1 & 1 & . & . & 1 \\ 1 & 0 & . & . & 0 \\ 0 & 1 & . & . & 0 \\ . & . & . & & . \\ 0 & . & & 1 & 0 \\ 0 & . & . & . & 1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{n-1} \\ x_n \end{bmatrix},$$

$$\mathbf{g}^0 = \begin{bmatrix} -f_1^0 \\ -f_2^0 \\ . \\ . \\ -f_{n-2}^0 \end{bmatrix}, \mathbf{e}^0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ . \\ 0 \\ 0 \end{bmatrix}, \tag{9}$$

or in a complex form:

$$\left[\begin{array}{ccccc} f_1^1 - f_1^0 & f_1^2 - f_1^0 & . & . & f_1^n - f_1^0 \\ f_2^1 - f_2^0 & f_2^2 - f_2^0 & . & . & f_2^n - f_2^0 \\ . & . & . & & . \\ . & . & & . & . \\ f_{n-2}^1 - f_{n-2}^0 & f_{n-2}^2 - f_{n-2}^0 & . & . & f_{n-2}^n - f_{n-2}^0 \\ \hline 1 & 1 & . & . & 1 \\ 1 & 0 & . & . & 0 \\ 0 & 1 & . & . & 0 \\ . & . & . & & . \\ 0 & . & & 1 & 0 \\ 0 & . & . & . & 1 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ . \\ . \\ x_{n-1} \\ x_n \end{array}\right] = \left[\begin{array}{c} -f_1^0 \\ -f_2^0 \\ . \\ . \\ -f_{n-2}^0 \\ \hline 1 \\ 0 \\ 0 \\ . \\ 0 \\ 0 \end{array}\right]. \tag{10}$$

Here we draw the attention of the reader to the fact, that in order to be on the hyperedge of the simplex one must fulfill only two of the last $n+1$ equations. We do a partial Gaussian elimination on Eq. (7) with a reduced line exchange, choosing the pivot $(n-2)$ times always from the first $n-2$ rows only (above the horizontal line in Eq.(10)). At the end of the partial elimination the structure of the system has the following, partial row echelon form (the stars represent the non-zero elements):

$$\left[\begin{array}{cccccc} 1 & * & . & . & & * \\ 0 & 1 & . & . & & * \\ . & . & . & . & & * \\ . & . & . & . & & * \\ 0 & 0 & . & 1 & * & * \\ \hline 0 & 0 & . & . & * & * \\ 0 & 0 & . & . & * & * \\ 0 & 0 & . & . & . & * \\ . & . & . & . & & . \\ 0 & 0 & . & . & * & * \end{array}\right] \left[\begin{array}{c} \\ x_1 \\ x_2 \\ . \\ . \\ x_{n-1} \\ x_n \end{array}\right] = \left[\begin{array}{c} * \\ * \\ * \\ * \\ * \\ \hline * \\ * \\ * \\ . \\ * \end{array}\right], \qquad (11)$$

or in a short form:

$$\left[\frac{\mathbf{F}}{\mathbf{B}}\right][\mathbf{x}] = \left[\frac{\mathbf{g}}{\mathbf{e}}\right]. \qquad (12)$$

This idea was originally used in the simplex algorithm introduced for the calculation of equilibrium paths in [13]. In the case of one-dimensional solution sets it is possible to decrease the number of non-zero columns below the horizontal line to one.

The back-substitution is done independently $n \times (n+1)/2$ times. We choose two rows from the last $n+1$ equations of the system $\mathbf{Bx} = \mathbf{e}$ and solve them for x_n and x_{n-1}. (It is possible that there is no unique solution for these two variables, but it would mean that the solution point is outside the simplex, so we would discard those solution anyway.) Then the next $n-2$ steps of the back-substitution must be performed on $\mathbf{Fx} = \mathbf{g}$. If some of the resulting $n(n+1)/2$ solution for x_i fulfill the inequality conditions in Eqs. (4)-(5), then we transform those local coordinates to the GRS with Eq. (3) and Eq. (2), and then store them.

2.3 Parallelization of the problem

The scanning algorithm explained in the previous subsection is a so-called massively parallelizable algorithm. Each simplex can be calculated independently from the others, and this holds for every hyperrectangle as well. The parallelization can be applied for the calculation of the error-functions in the nodes of hyperrectangles and for the building and solution of the sets of linear equations. A former method by Gáspár et al. in [13] used the PVM environment to divide the GRS between the collaborating computers.

Current direction in the field of parallel computation is the application of the computational power of graphical cards. For the real time creation of 3D scenes in games and CAD applications, similar calculations must be done on each pixel of the displayed images. This led the development of the video cards in a direction where numerous graphical processing unit (GPGPU) can be accessed from the main program. Depending on the manufacturer of the graphical processor, different extensions are available to access the GPGPUs for computational purposes, too. When programming the GPGPU directly, a crucial point is the careful planning of the data transfer between the system memory and the on-board memory of the graphical card.

In a preparation for this task, we developed a parallel version of the algorithm with the OpenMP environment. OpenMP is an API that supports shared memory multiprocessing programming and enables the easy creation of numerous threads during the running of the program [5]. From a programming point of view it is simple to call the API, but creating the threads and moving the data in the memory still requires precious computational time, so we implement the parallelization in a specific way described below.

As we described the method in the previous subsection, hyperrectangles are calculated in a series one after one another. The location of each hyperrectangle can be referred to with its starting point in the GRS. Those starting points form a grid in the GRS. In every hyperrectangle the error-functions are calculated in the vertices, and then the systems of linear equations of the simplices inside the hyperrectangle are built and solved. The white, wireframed cube in the center of Figure 1(a) shows one actually calculated cube. For further application let us denote the number of hyperrectangles in a row along the ith coordinate axis of the GRS by C_i.

In order to decrease the time required for creating different threads, for computing the error-functions and for solving the systems of equations we create a larger block of hyperrectangles, and calculate them in two parallel session. The series of white, wireframed cubes in the center of Figure 1(b) shows one of these blocks. One can see that only the last coordinate of the starting points of the hyperrectangles varies in the block.

In the first parallel session the error-functions are calculated in the nodes of the hyperrectangles. This requires the calculation of the error-functions in the $T_1 = 2^{n-1}(C_n+1)$ nodes of the GRS. In the second parallel session we solve the systems of linear equations of every simplex in the current block. To build the B_0 matrix the error-functions are taken from the stored results of the previous session. This session requires the solution of $T_2 = n!C_n$ simplices. Both of the sessions are calculated in a parallelized cycle, where the threads are created at the beginning of the cycle, and the steps of the cycle are divided between the threads by the OpenMP library.

In both of the above sessions we group the T_1 and T_2 calculation into a prescribed number of threads. These numbers of threads may be different, of course. Generally, the total number of independent calculations, i.e. T_1 and T_2 are different, so the division of the calculation for the above described two sessions is highly recommended, to minimize time, when the threads have to wait for the unfinished ones. Further acceleration of the method by the application of more advanced load balancing techniques is beyond the goal of this paper.

3 Examples

3.1 Clamped-clamped rod with compression and twist

An essential example for the two parameter loading is the clamped-clamped rod problem, where a long, initially straight, linear elastic rod of circular cross-section with clamped-

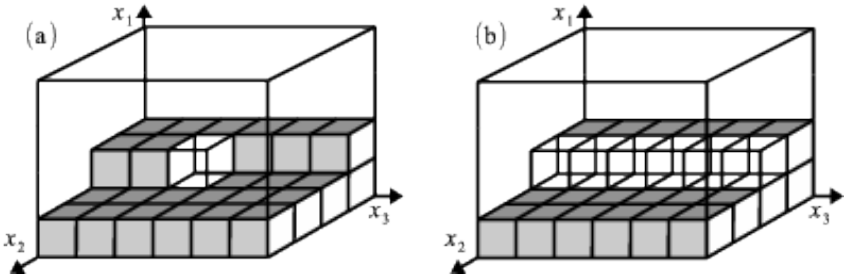


Fig. 1. Solution strategy with (a) a serial and (b) a parallel scanning for $n = 3$. White wireframed cubes are the currently calculated ones.

clamped ends is loaded at one end with a compression force and a twist moment as the two independent load-parameter. The simplicity of the rod model allows its analytical and numerical analysis, while it is still a complex enough model to approximate the behavior of long structures of the size in the range of molecule chains to undersea cables. Henderson and Neukirch presented a surface following method, which is capable of finding the surface elements connected to any given solution point [20]. (Moreover, it was shown in a related paper [16], that in this problem *all* solutions are connected to a given set of trivial solution points, so the method finds all of the solutions.) The continuous symmetry of the isotropic rod causes that every equilibrium configuration has an axis of flip-symmetry.

If the non-existence of disconnected surfaces is not proven, we have to know a solution point on each of them to use a surface following. (This is a drawback of every continuation method, not only of this specific one, of course.) Scanning of the parameter space avoids this need of starting points, as it finds all solutions in the analyzed region independent of their connectivity.

In this subsection we present the results of the scanning for a hemitropic rod under the same end-conditions. For the hemitropic rods no proof is known to the author, that all solution surfaces are connected, but the scanning algorithm is capable to find all solution with the same flip-symmetry, independent of the connectedness of the surfaces.

The rod shape is governed by the following system of differential equations:

$$\mathbf{r}'(s) = \nu_i(s)\mathbf{d}_i(s), \tag{13}$$

$$\mathbf{d}'_i(s) = \boldsymbol{\kappa}(s) \times \mathbf{d}_i(s), \tag{14}$$

$$\boldsymbol{\kappa}(s) = \kappa_i(s)\mathbf{d}_i(s), \tag{15}$$

where the vector $\mathbf{r}(s)$ describes the axis of the rod, $\mathbf{d}_i(s)$ are the directors (basis of the local reference system) of the cross section of the rod, $\nu_i(s)$ are the strain properties of the cross-section, $\boldsymbol{\kappa}(s)$ is the vector of specific rotation of the cross-section and $\kappa_i(s)$ are the specific rotations of the cross-section. The prime represents a differentiation with respect to the arc-length s. The strains of the cross-section are the $\nu_1(s)$ and $\nu_2(s)$ shears and the $\nu_3(s)$ stretch. The specific rotations of the cross-section are the $\kappa_1(s)$ and $\kappa_2(s)$ curvatures and the $\kappa_3(s)$ twist.

The internal forces and internal couples are:

$$\mathbf{n}(s) = n_i(s)\mathbf{d}_i(s), \quad \mathbf{m}(s) = m_i(s)\mathbf{d}_i(s). \tag{16}$$

The static equilibrium of an elementary rod segment are expressed by the equations:

$$\mathbf{n}'(s) = 0, \quad \mathbf{m}'(s) = \mathbf{r}'(s) \times \mathbf{n}(s). \tag{17}$$

Constitutive equations reflect the assumptions on the material of the rod. We neglect the shear deformations, thus $\nu_1(s) = \nu_2(s) = 0$. We assume circular cross-section and linear elastic material behavior, so the bending stiffness A is the same in the first and the second local direction:

$$m_1(s) = A\kappa_1(s), \quad m_2(s) = A\kappa_2(s). \tag{18}$$

Due to the hemitropy of the rod the twist and the stretch are coupled:

$$\begin{aligned} n_3(s) &= D(\nu_3(s) - 1) + E\kappa_3(s), \\ m_3(s) &= E(\nu_3(s) - 1) + C\kappa_3(s). \end{aligned} \tag{19}$$

The boundary conditions are the clamped constrains on the beginning of the rod:

$$\mathbf{r}(0) = \mathbf{0}, \quad \mathbf{d}_i(0) = \mathbf{e}_i, i = 1, 2, 3, \tag{20}$$

and the clamped constraints on the final end, with the load parameters λ_1, λ_2 on the compression and the twist moment:

$$\begin{aligned} &\mathbf{r}(L) \cdot \mathbf{e}_1 = \mathbf{r}(L) \cdot \mathbf{e}_2 = 0, \\ &\mathbf{d}_1(L) \cdot \mathbf{e}_3 = \mathbf{d}_2(L) \cdot \mathbf{e}_3 = 0, \\ &n_3(L) = \lambda_1, \quad m_3(0) = \lambda_2. \end{aligned} \tag{21}$$

The boundary conditions of Eqs. (20)-(21) include a continuous symmetry group, i.e. any equilibrium configuration rotated around the z-axis will fulfill the same boundary conditions as well. This symmetry increases the dimension of the solution set by one, which would make impossible to use any classical method for the calculation. The same continuous symmetry exists in the case of isotropic rods (where the normal- and the twist stiffness are decoupled). For that problem Neukirch and Henderson proved in [20] that this continuous symmetry causes at least one axis of C_2 symmetry. The axis of this flip-symmetry goes through the mid-point of the rod and orthogonal to the line connecting both ends of the rod.

In the case of a hemitropic rod, the continuous symmetry of the initial state and the boundary conditions results the same symmetry property, thus there are solutions with an axis of flip-symmetry [14]. To avoid multiple solutions from the rotation of a configuration around axis z, we only calculate those where the axis of flip-symmetry lies in the yz-plane (and is parallel to the y axis). Then, the global equilibrium of forces acting on the rod decreases the number of unknown parameters, because $\mathbf{n}\cdot\mathbf{e}_2 = 0$ must hold at any cross-section. The flip-symmetry leads to three constraints in the mid-point of the rod:

$$r_x(L/2) = 0, \quad \mathbf{d}_3(L/2)\cdot\mathbf{e}_2 = 0, \quad \mathbf{m}(L/2)\cdot\mathbf{e}_2 = 0. \tag{22}$$

By setting the origin of the coordinate system in the mid-point of the rod with axis z parallel to the rod in that point while keeping the axis of flip-symmetry parallel to axis y allows one to decrease the size of the problem. Describing a configuration that way the initial conditions are

$$\mathbf{r}(L/2) = \mathbf{0}, \quad \mathbf{d}_1(L/2) = \mathbf{e}_1, \quad \mathbf{d}_2(L/2) = \mathbf{e}_2, \quad \mathbf{d}_3(L/2) = \mathbf{e}_3 \tag{23}$$

with the force parameters

$$\mathbf{n}(L/2) = [n_x\ 0\ n_z]^T, \quad \mathbf{m}(L/2) = [m_x\ 0\ m_z]^T. \tag{24}$$

One can calculate the rod shape from its mid-point to its end. In the end-point the flip-symmetry requires that the line of the tangent must cross axis y orthogonally, leading to the constraints:

$$d_{3y}(L) = 0, \quad r_x(L)d_{3z}(L) - r_z(L)d_{3x}(L) = 0. \tag{25}$$

Calculation of the rod shape with this method results in a GRS with the dimension of 4. The results of the scanning in this space makes the comparison of the results to those presented in [16] easy.

We calculated the space of n_z, m_z, m_x, n_x with 70x50x50x50 hyperrectangles. The parameters of the rod were:

$$L = \pi, \quad A = 1, \quad C = 1, \quad D = 500, \quad E = 2. \tag{26}$$

The rod shape was calculated with a second-order Runge-Kutta method with a step size of $ds = 0.001\pi$. The results are shown in the rendered image in Figure 2. In the figures three coordinates are shown, and the fourth parameter serves as a color-code of the surface elements.

We made the scanning of the same part of the GRS for an isotropic rod and for a hemitropic rod with reversed handedness. These calculations were done by substituting the E values of 0 and -2 into Eq. (26), respectively. Figure 3 shows the results of these calculations. The viewpoint of each figure is the same, so the effect of the handedness of the rod can be observed. To make the difference between the three diagrams cleaner, Figure 4 shows three rod shapes from each diagram. The load parameters are for each solution: $n_z = -20$ and $m_z = -1$. The m_x bending moment varies between 6.940 and 6.650 while the n_x shear force varies between 0.501 and 0.912.

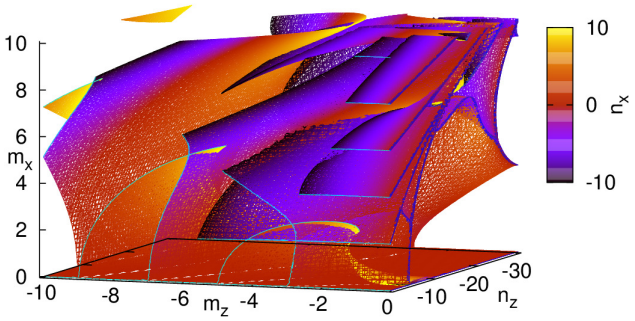


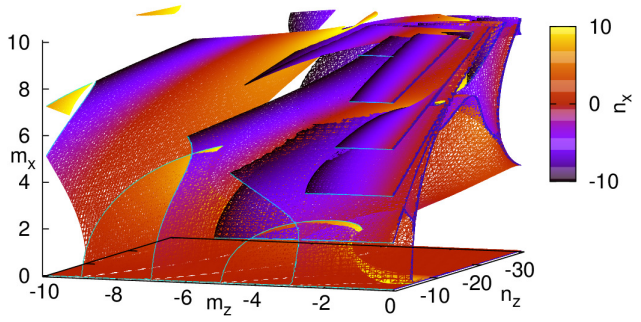
Fig. 2. Results of the scanning of the hemitropic rod. Blue and cyan lines represent the crossings of the scanned surfaces with the $m_z = 0$ and the $n_z = 0$ planes, respectively.

3.2 Perturbation of the bifurcation diagram of the elastic web of links

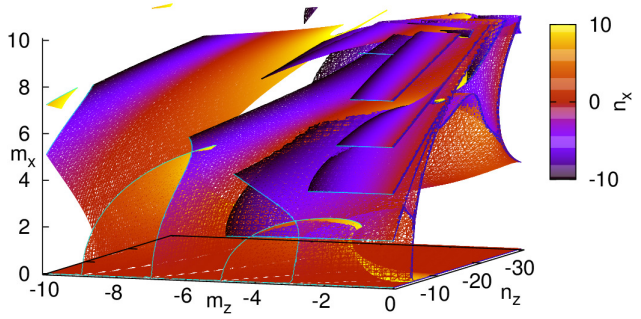
Kocsis et al. presented a discrete model for the analysis of static equilibrium of rods with shear stiffness in [19]. The model consists of stiff elements of the same length ℓ (so-called links) arranged in a squared mesh. The initially horizontal and vertical links are connected via internal joints and internal spiral springs of stiffness c. If we combine this model with spiral springs of stiffness k, connecting the consecutive columns as shown in Figure 5, we get a discrete model with bending and shear stiffness [21]. The k-springs themselves have the same role as the springs of the elastic linkage, which is a discrete model of the Euler-problem. The elastic linkage is a widely analyzed model, because its simple nonlinearity allows the occurrence of spatial chaotic behavior. This chaotic behavior of the elastic linkage was first analyzed by [9], and recently by [17, 18] under the effect of non-conservative loading. The equilibrium equations of the mixed model can be derived from the potential energy. Similarly to the problem without bending stiffness, the tilt angle of the initially horizontal bars will be the same for each floor. This lets us simplify the non-empty sides of the equilibrium equations into the form:

$$\begin{aligned} f_1 &= -\lambda\sin(\beta_1) + \beta_1 - (1-r)\bar{\beta} - \frac{r}{2}\beta_2 \\ f_i &= -\lambda\sin(\beta_i) + \beta_i - (1-r)\bar{\beta} - \frac{r}{2}(\beta_{i-1} - \beta_i + \beta_{i+1}), \\ &\qquad\qquad i = 2,\ldots,N-1 \\ f_N &= -\lambda\sin(\beta_N) + \beta_N - (1-r)\bar{\beta} - \frac{r}{2}\beta_{N-1} \end{aligned} \tag{27}$$

where $\lambda = (M+1)F/(2k+4Mc)$ is the non-dimensional load parameter, β_i is the angle of the columns on the ith floor to the vertical, $\bar{\beta}$ is the average of the angles of the columns, $r = 2k/(2k+4Mc)$ is the ratio of the bending stiffness of one floor to the total stiffness of the same floor. In the equilibrium state the $f_j (j = 1,\ldots,N)$ functions have fix-points, so they can be treated as the error-functions for the calculation of bifurcation diagrams in the $N+2$-dimensional space of $\lambda, r, \beta_j (j = 1,\ldots,N)$. Thus, with the notation of Section 2 the GRS is $n = N+2$ dimensional.



(a)



(b)

Fig. 3. Equilibrium surfaces of (a) the isotropic rod and (b) the hemitropic rod with reversed handedness. The viewpoint and the form of the figures are the same as in Fig.2

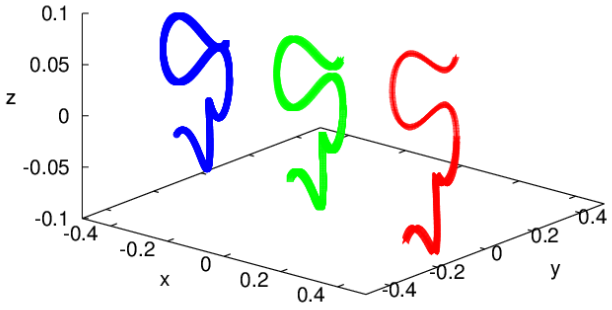


Fig. 4. Three equilibrium configurations of the same load level $n_z = -20$, $m_z = -1$. The isotropic rod is in the middle, the hemitropic rods with $E = -2$ and $E = 2$ are shifted parallel by 0.4 along the axis x in negative and positive direction, respectively.

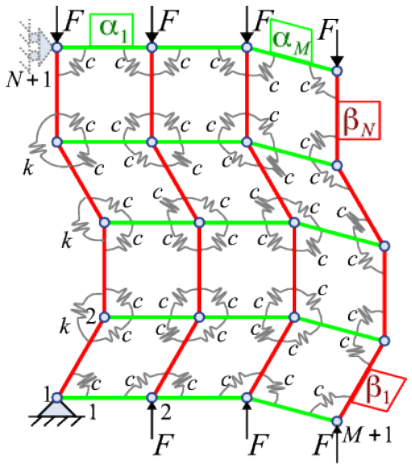


Fig. 5. Elastic web of links

We calculated the bifurcation diagram of a three-level web (i.e. $N = 3$) unraveling the double cusp catastrophe of the model. The load parameter varied between 0.4 and 2.0, the stiffness ratio varied between 0 and 1, and the column angles varied between -2 and 2. The GRS was divided into 25×40×40×40×40 hyperrectangles.

The results are shown in Figure 6. We present here load parameter λ as a function of the angle of the first and the top column. The graph is colored depending on the stiffness ratio r, so isocolor lines are the equilibrium paths of a web with fixed stiffness ratio. Black lines on the boundaries of surfaces are the equilibrium paths of the elastic web of links (starting from $\lambda = 1$) or of the elastic linkage (starting from $\lambda = 0.5$ and $\lambda = 1.5$). Near the $\lambda = 1$ bifurcation the parasitic solutions as a direct result of the higher order catastrophe can be observed. The coloring methods allows one to see the fast changing behavior of some paths depending on the r parameter, while the linkage-like bifurcated paths undergo a simple translation along the λ axis. The blue frames are the typical deformed shapes of an elastic web with no bending springs (i.e. $r = 0$), their secondary branches on the surface are indicated by the green lines.

4 Speed-up of the parallel computation

For every parallel algorithm an important question is the speed-up of the computation. Using J threads can accelerate the computation by a factor of J only if the whole algorithm can be parallelized. Otherwise, the serial parts of the program run with the serial mode. So, there is a theoretical maximum for the acceleration of the code referred to as Amdahl's law [2]. This upper limit of the acceleration is still a theoretical value, because creating the independently running threads, using the same I/O-devices results in further overheads. We analyzed the speed-up of our algorithm on a desktop computer with the 4 cores of an Intel(R) Core(TM) i7-3770 CPU, allowing 8 threads as a maximum. We have done the calculation on both of our previously presented problem with the following results.

The scanning of the five-dimensional GRS of the problem analyzed with the elastic web was done using 1 to 10, 12 and 16 threads. Table 1 shows the measured time for the calculations, and their ratio to the measured time with one thread multiplied by the number of actual treads. Figure 7 shows the measured time for the calculations as a function of the threads on a log-log scale. The ratio in Table 1 has the lower limit of 1 (and, depending on the part of the serial code an even higher one), but the smaller it is, the better the efficiency of the parallelization is. From the results we can conclude that the speed-up is almost linear to the number of threads as long as there is a core for every thread. For the thread numbers 5 to 8 the efficiency decreases,

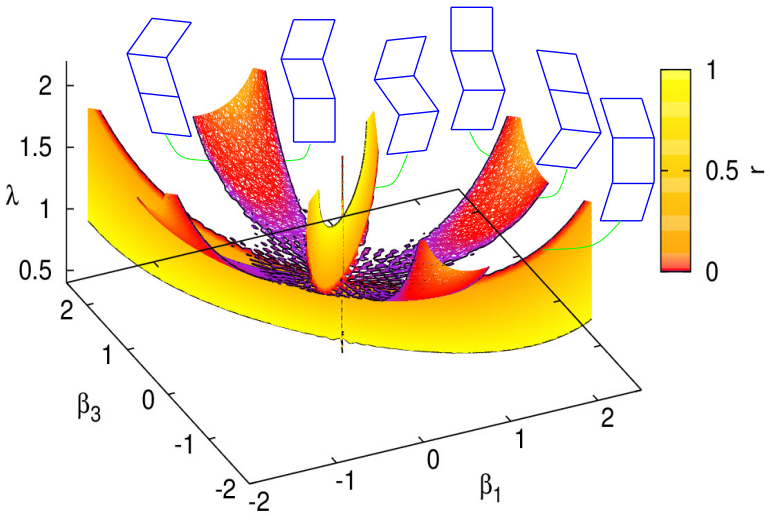


Fig. 6. Bifurcation diagram of the bielastic web of links with $N = 3$. Color represents the ratio r of the shear stiffness to one floor's total stiffness. The six blue webs are typical deformed shapes of the elastic web of links.

Tab. 1. Measured times of calculations with various numbers of threads

Threads (J)	1	2	3	4	5	6	7	8	9
Time [min] (T_J)	78.5	39.6	27.2	21.0	26.0	21.9	18.9	16.8	24.8
Proc. time (JT_J)	78.5	79.2	81.6	84.0	130.0	131.4	132.3	134.4	223.2
Ratio (JT_J/T_1)	1	1.01	1.04	1.07	1.65	1.67	1.68	1.71	2.85

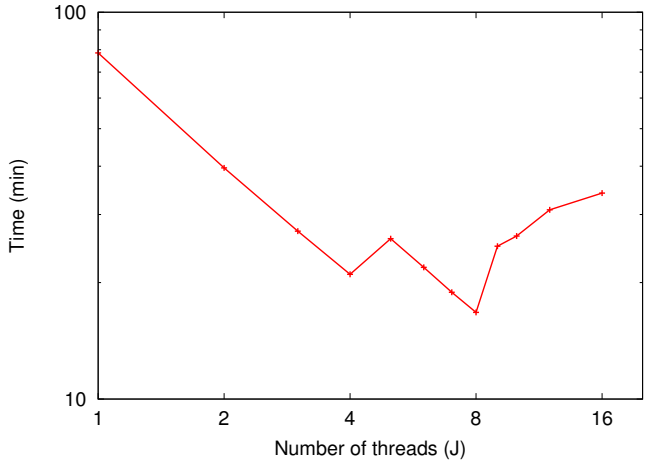


Fig. 7. Log-log scale diagram of the running time (t) as a function of applied threads. Almost linear segments for $J = 1, 2, 3, 4$ and for $J = 5, 6, 7, 8$ represents linear scalability of the problem as long the maximum usage of cores is similar.

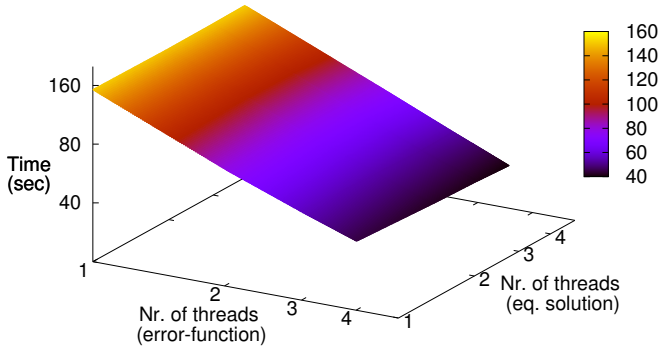


Fig. 8. Log-log scale diagram of the running time (t) as a function of applied threads in the calculation of the error-function and the solution of the system of linear equations.

but the acceleration is still close to linear. More threads per core will results inevitably in a performance drop. The reason is that some of the threads must move from the cache of the processor to a slower memory and back. These conflicting threads result in a breakdown of the overall performance, especially, when the number of threads exceeds the total number of available parallel threads. This can be seen as an extreme increase in the time for the cases $J = 9, J = 10, J = 12, J = 16$. The optimal case is $J = 8$, where the number of threads equals to the number of available threads. But this optimum requires the system to operate on that program only. Any system call of the operational system may break down the performance in this case.

For the hemitropic rod we analyzed the effect of dividing the number of threads between the calculation of the error-functions and the solution of the system of linear equations. Figure 8 shows the results of the 4×4 runs in a triple-log scale. One can see that multithreading results in a much larger speed-up, when used for the calculation of the error-functions, hence the numerical solution of the IVP has a much larger computational need. Faster calculation of the error-functions (with respect to the solution of the system of linear equations) leeds to a smaller gap between the change of the accelerations, of course.

5 Conclusions and outlooks

Scanning version of the simplex algorithm is a powerful and robust numerical method for approximate solution of nonlinear problems with a one-dimensional solution set (e.g. finding the equilibrium paths of a mechanical problem formulated as a boundary value problem). We have shown, that a slight modification of the method makes it capable for finding 2-dimensional solution sets of nonlinear problems. These solution sets are surfaces in the parameter space. For the implementation we derived a parallel code with the OpenMP environment, which is capable of using multiple cores of the CPU. We applied our method for the calculation of the equilibrium surfaces of the clamped-clamped hemitropic rod to see the effect of the handedness of the rod, and calculated equilibrium paths for the perturbed elastic web of links.

We analyzed the speed-up of the method with various numbers of separated threads, with differentiated thread numbers for the various sessions of a calculation block. The current results can be a good base for the parallel code making use of the GPGPUs of the graphical cards.

Acknowledgement

Financial supports from OTKA No. K100894 are greatly acknowledged.

References

1 **Allgower EL, Georg K**, *Numerical Continuation Methods: An Introduction*, Series in Computational Mathematics, Vol. 13, Springer Verlag; Berlin, Heidelberg, New York, 1990.

2 **Amdahl GM**, *Validity Of The Single Processor Approach To Achieving Large Scale Computing Capabilities*, In: Proceedings of the April 18-20, 1967, spring joint computer conference, AFIPS '67 (Spring), ACM; New York, NY, USA, 1967, pp. 483–485, DOI 10.1016/0898-1221(94)00175-8.

3 **Bachrathy D, Stépán G**, *Bisection Method In Higher Dimensions And The Efficiency Number*, Periodica Polytechnica Mechanical Engineering, **56**(2), (2012), 81–86, DOI 10.3311/pp.me.2012-2.01.

4 **Brodzik ML, Rheinboldt WC**, *The Computation Of Simplicial Approximations Of Implicitly Defined Two-Dimensional Manifolds*, Computers & Mathematics with Applications, **28**(9), (1994), 9–21, DOI 10.1016/0898-1221(94)00175-8.

5 **Chapman B, Jost G, van der Pas R**, *Using OpenMP: Portable Shared Memory Parallel Programming*, Scientific And Engineering Computation, MIT Press, 2008.

6 **Coxeter HSM**, *Regular Polytopes*, 3rd, Dover; New York, NY, USA, 1973.

7 **Doedel E, Keller HB, Kernevez JP**, *Numerical Analysis And Control Of Bifurcation Problems (I): Bifurcation In Finite Dimensions*, International Journal of Bifurcation and Chaos, **01**(03), (1991), 493–520, DOI 10.1142/S0218127491000397.

8 **Doedel E, Keller HB, Kernevez JP**, *Numerical Analysis And Control Of Bifurcation Problems (II): Bifurcation In Infinite Dimensions*, International Journal of Bifurcation and Chaos, **01**(04), (1991), 745–772, DOI 10.1142/S0218127491000555.

9 **Domokos G, Holmes P**, *Euler's problem, Euler's method, and the standard map; or, the discrete charm of buckling*, Journal of Nonlinear Science, **3**(1), (1993), 109–151, DOI 10.1007/BF02429861.

10 **Domokos G, Gáspár Z**, *A Global, Direct Algorithm for Path-Following and Active Static Control of Elastic Bar Structures*, Mechanics of Structures and Machines, **23**(4), (1995), 549–571, DOI 10.1080/08905459508905251.

11 **Domokos G, Szeberényi I**, *A Hybrid Parallel Approach To One-Parameter Nonlinear Boundary Value Problems*, Computer Assisted Mechanics and Engineering Sciences, **11**(1), (2004), 15–34.

12 **Domokos G, Healey TJ**, *Multiple Helical Perversions Of Finite, Intristically Curved Rods*, International Journal of Bifurcation and Chaos, **15**(03), (2005), 871–890, DOI 10.1142/S0218127405012430.

13 **Gáspár Z, Domokos G, Szeberényi I**, *A Parallel Algorithm For The Global Computation Of Elastic Bar Structures*, Computer Assisted Mechanics and Engineering Sciences, **4**(1), (1997), 55–68.

14 **Healey TJ, Papadopoulos CM**, *Bifurcation Of Hemitropic Elastic Rods Under Axial Thrust*, Quarterly of Applied Mathematics, **71**, (2013), 729–753.

15 **Henderson ME**, *Multiple Parameter Continuation: Computing Implicitly Defined K-Manifolds*, International Journal of Bifurcation and Chaos, **12**(03), (2002), 451–476, DOI 10.1142/S0218127402004498.

16 **Henderson ME, Neukirch S**, *Classification Of The Spatial Equilibria Of The Clamped Elastica: Numerical Continuation Of The Solution Set*, International Journal of Bifurcation and Chaos, **14**(04), (2004), 1223–1239, DOI 10.1142/S0218127404009971.

17 **Kocsis A, Károlyi Gy**, *Buckling Under Nonconservative Load: Conservative Spatial Chaos*, Periodica Polytechnica Civil Engineering, **49**(2), (2005), 85–98.

18 **Kocsis A, Károlyi Gy**, *Conservative spatial chaos of buckled elastic linkages*, Chaos, **16**, (2006), 033111/1–7.

19 **Kocsis A, Németh RK, Károlyi Gy**, *Spatially Chaotic Bifurcations Of An Elastic Web Of Links*, International Journal of Bifurcation and Chaos, **20**(12), (2010), 4011–4028, DOI 10.1142/S021812741002815X.

20 **Neukirch S, Henderson ME**, *Classification Of The Spatial Equilibria Of The Clamped Elastica: Symmetries And Zoology Of Solutions*, Journal of Elasticity, **68**(1-3), (2002), 95–121, DOI 10.1023/A:1026064603932.

21 **Németh RK, Kocsis A**, *Bielastic web of links: a discrete model of* Csonka's *beam*, International Journal of Non-Linear Mechanics. Accepted.