Computer Assisted Mechanics and Engineering Sciences, 4: 55–68, 1997.

A parallel algorithm for the global computation of elastic bar structures

Zsolt Gáspár, Gábor Domokos and Imre Szeberényi
Technical University of Budapest, H-1521 Budapest, Hungary

(Received May 27, 1996)

This paper presents an algorithm for parallel computers, which is suitable for the global (arbitrary displacements) computation of elastic bar structures subject to quasi-static loads. Our method is also capable to determine equilibria which are not connected to the initial, trivial configuration. The paper discusses the gains and the disadventages of the method, comparing it with other techniques.

1. Introduction

This paper introduces a method for the computation of elastic bar structures. We do not restrict the magnitude of the displacements, i.e. we are interested in *all* equilibria of the structure (within a given domain of the parameters), hence we call our method a global algorithm. Such nonlinear problems are commonly resolved by incremental-iterative techniques, starting from the initial, trivial configuration of the structure and following the equilibrium path in small steps, based on extrapolation. The error caused by the extrapolation is diminished by successive iterative steps, which are supposed to converge to the exact solution.

Our algorithm adopts a different approach. We define a finite dimensional space into which the global equilibrium path can be embedded. This equilibrium path is equivalent to the solution of a nonlinear equation system depending on one parameter. We solve the equation system by discretizing the mentioned space into simplices. The functions of our equation system can be piecewise linearly interpolated on this simplectic grid. The appeal of this approach is threefold:

1. it does not contain iterative steps
2. it is capable of finding equilibria not connected to the trivial solution
3. it is highly suitable for parallelization.

Simultaneously — and quite naturally — the method has its weak sides. For the global results one has to pay with huge computation effort — this can be partially neutralized by the application of powerful parallel machines. As already observed by several authors, ([6, 8]) discretization breeds spurious solutions. In our case a double discretization is applied: not only the embedding space but the structure itself has to be discretized. Consequently we obtain two distinct classes of spurious solutions. Although we expect these solutions to vanish as the meshsize goes to zero, in many cases it is hard to tell whether a given solution is relevant or not.

The paper is structured into 6 sections, including this one as the first.

Section 2 discusses the deep, but non-technical ideas from the theory of ODEs (Ordinary Differential Equations) which are necessary to construct the mentioned finite-dimensional embedding of the equilibrium path. We deal with BVPs (Boundary Value Problems) associated with ODEs and find a method to reduce these BVPs to parameter-dependent IVPs (Initial Value Problems).

Section 3 deals with the core of the numerical method: we describe here the Simplex Method. This method is based on the so-called PL algorithm presented by Allgower and Georg [1]. This section

provides also a discussion on the origin of the spurious solutions resulting from the discretization of the space. We describe the algorithm for the general, n-dimensional case.

The method appears to be particularly suitable for parallelization. We utilized this property by implementing it under the so-called PVM (Parallel Virtual Machine) system. The tests confirmed that the parallel architecture can be used with high efficiency. These ideas are presented in Section 4.

Section 5 provides some selected examples, intended to highlight the advantages and the drawbacks of our method. The examples presented here are partially published earlier in [4].

Finally, Section 6 draws conclusions and summarizes our results.

2. The basic concepts from mechanics

The shape of elastic bars subject to quasi-static loads is commonly described by BVPs associated with ODEs. As the simplest example we present the cantilever bar with compressive end-load, illustrated in Fig. 1:

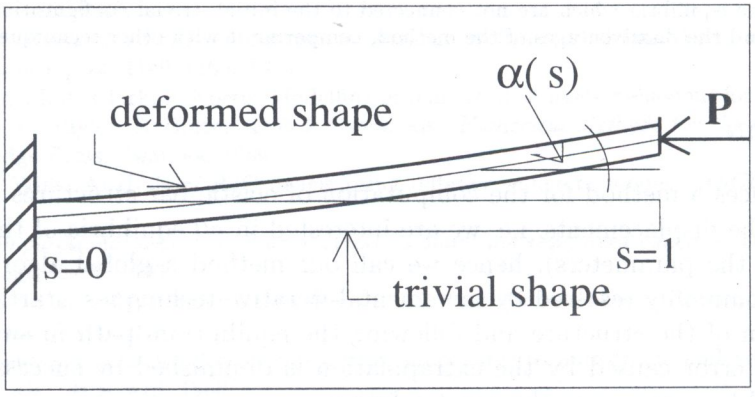


Fig. 1. The cantilever beam

The ODE describing the shape of the beam in terms of the slope α as a function of the arclength s was first described by Euler:

$$\alpha'' + P\sin(\alpha) = 0. \tag{1}$$

The trajectories of this equation are uniquely determined by the three scalars $\alpha(0), \alpha'(0)$ and P (the former ones being "true" initial conditions, the latter one a parameter). However, not all trajectories are of interest for us, only those which meet the boundary conditions

$$\begin{aligned} a/ \quad & \alpha(0) = 0, \\ b/ \quad & \alpha'(1) = 0 \end{aligned} \tag{2}$$

expressing zero slope and zero curvature at the left and right end, respectively. For the time being, we ignore the far-end condition (2/b) and concentrate on (2/a). This condition eliminates one of the "variable" initial conditions as a constant, so all trajectories which *might* meet the boundary conditions can be uniquely represented in the $[\alpha'(0), P]$ plane. Thus we managed to project the infinite-dimensional space of all geometrically possible configurations to a 2-dimensional space in such a way that the latter space is in a one-to-one correspondence with the set of IVPs. The relevant BVP solutions can be regarded as a subset of these IVPs. The scalars $\alpha'(0)$ and P are the *global coordinates* for this BVP, the plane (space) spanned by them will be called the *global representation space* (GRS) of the BVP. Since the BVP contains one parameter (P), the solutions will typically

appear as one-dimensional manifolds, i.e. lines. Algebraically these lines can be expressed as the solutions of the nonlinear equation (corresponding to the far-end condition (2/b))

$$\alpha'(1) = f_1(\alpha'(0), P) = 0. \tag{3}$$

The far-end value of $\alpha'(1)$ is uniquely determined by our chosen variables $\alpha'(0)$ and P since the investigated ODE satisfies the conditions of Peano's Uniqueness Theorem [2]. This is always the case with ODEs describing elastic bars, non-uniqueness arises in the case of strings [3]. In the latter case our method has to be applied with strong caution. Our approach guarantees that whenever two solution lines intersect, the equilibria corresponding to those lines also coincide. We call such a diagram *topologically correct.*

On this simple example we introduced the basic concepts of our method. In the case of more complicated problems the GRS has more dimensions, however, it still remains a finite-dimensional space. If the GRS is n-dimensional, then the analogous equation system to (3) contains $n-1$ equations, determining — as in the case of the cantilever — 1-dimensional solution sets, i.e. lines. The assembly of those lines is called the global equilibrium path, or, the global bifurcation diagram. The global bifurcation diagram of the cantilever beam is illustrated (for a finite domain of the GRS) in Fig. 2:

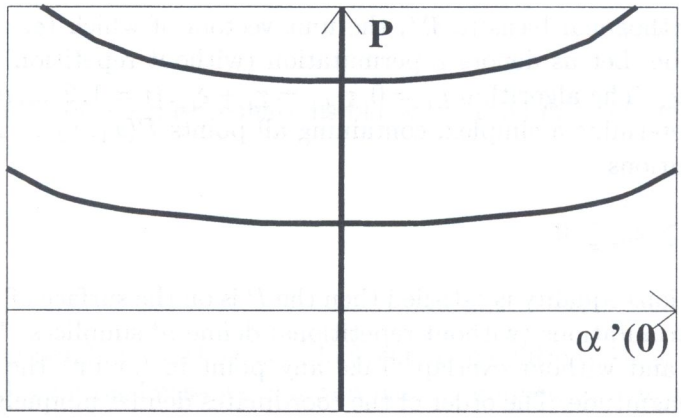


Fig. 2. The bifurcation diagram of the cantilever beam

The forthcoming sections investigate the problem how to compute this diagram.

3. The core of the algorithm

In the previous section we established the global representation space (GRS) into which the global bifurcation diagram can be topologically correctly embedded. The computation of the diagram relies on the discretization of the GRS. The most natural way to discretize is to choose a simplectic grid. An n-dimensional simplex is defined by $n+1$ points. (For example, a two-dimensional simplex is a triangle.) There are many ways to construct a simplectic grid. We choose the following method: in the first step we construct an orthogonal (cubic) grid and then, in the second step we subdivide each cube into $n!$ simplices. (For the sake of simplicity henceforth we refer to n-dimensional orthogonal objects as "cubes" even if a linear transformation $\bar{x}_i = \lambda_i x_i, (i = 1, 2, ..., n)$ is needed to transform them into cubes.) The just described two steps are illsutrated in Fig. 3 for the already introduced cantilever example.

In higher dimensions the construction of the simplectic (secondary) grid is non-trivial. In order to simplify the algorithm of subdivision we adopted the method yielding $n!$ simplices, which has the following algorithm:

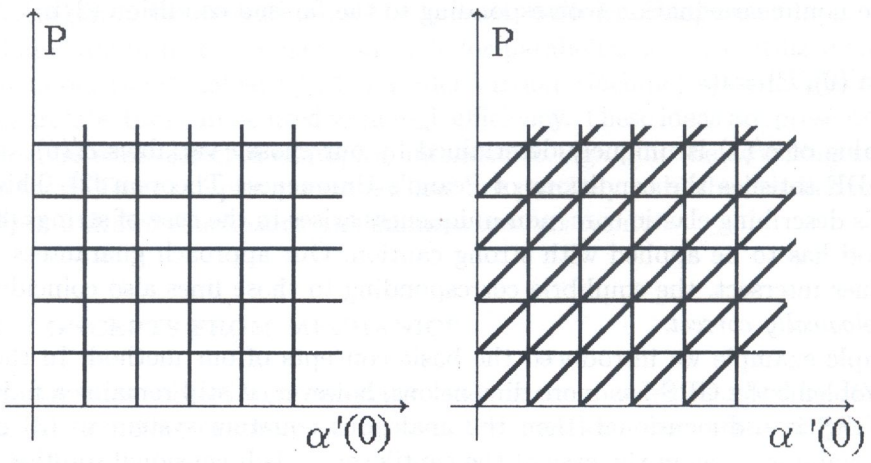


Fig. 3. The discretization of the GRS:
a) cubic (primary) grid b) simplectic (secondary) grid

Let us regard an orthogonal basis in $\Re^n$, the unit vectors of which $(\underline{e}_i, i = 1, 2, ..., n)$ span an n-dimensional unit cube. Let us denote a permutation (without repetition) of the first n natural numbers by $\nu_1, \nu_2, ..., \nu_n$. The algorithm $\underline{r}_1 = 0, \underline{r}_{i+1} = \underline{r}_i + \underline{e}_{\nu_i}, (i = 1, 2, ..., n)$ defines $n + 1$ points in $\Re^n$. These points determine a simplex, containing all points $P(x_1, x_2, ..., x_n)$ the coordinates of which satisfy the conditions

$$1 \geq x_{\nu_1} \geq x_{\nu_2} \geq ... \geq x_{\nu_n} \geq 0\,. \tag{4}$$

(If in any of these relations equality is satisfied then the P is on the surface of the simplex, otherwise in the interior.) All permutations (without repetitions) define $n!$ simplices. These simplices fill the unit cube completely and without overlap: Take any point in (or on) the cube and arrange its coordinates by their magnitude. The order of the coordinates defines uniquely the simplex to which the selected point belongs.

After having constructed the simplectic grid the $n - 1$ nonlinear functions are evaluated at the meshpoints. In the case of the cantilever example we have $n = 2$, thus a single fuction $\alpha'(1) = f_1(\alpha'(0), P)$ has to be evaluated at the meshpoints in the $[\alpha'(0), P]$ plane. This evaluation is realized by integrating the ODE (1) with initial conditions

$$\begin{aligned} \alpha(0) &= 0\,, \\ \alpha'(0) &= i\Delta_1\,, \\ P &= j\Delta_2\,, \\ i, j &= 0, \pm 1, \pm 2, ... \end{aligned} \tag{5}$$

where Δ_1 and Δ_2 denote the meshsize in the $\alpha'(0)$ and P direction, respectively. One can apply any numerical scheme to integrate this equation. We remark that these schemes often yield spurious solutions for the BVP, so, the meshsize Δs for the arclength s has to be chosen as suitably small. The integration yields the desired values of f_1 at the meshpoints. (In the general case we obtain $n - 1$ function values at each meshpoint.) In the case of the cantilever beam the function f_1 can be interpreted as a surface in the $[\alpha'(0), P, \alpha'(1)]$ space; the bifurcation diagram is the intersection of this surface with the GRS.(In our case the GRS is the $[\alpha'(0), P]$ plane.) Based on our just computed function values the function f_1 can be piece-wise linearly interpolated over the given domain by

the C^0-continuous function f_1^L. Over each simplectic domain the function f_1^L is constructed as a linear combination of base functions, the coefficients of these latter are derived from the function values. The permutation $\nu_1, \nu_2, ..., \nu_n$ defines $n+1$ points, thus a simplex of our grid. The values of the functions f_i $(i = 1, 2, ..., n-1)$ computed at the vertices of the simplex will be denoted by f_{ij}, $(j = 1, 2, ..., n)$. In our example we have $n = 2$, and our single function f_1^L can be expressed as

$$f_1^L = f_{11} + (f_{12} - f_{11})x_{\nu_1} + (f_{13} - f_{12})x_{\nu_2}\,, \tag{6}$$

and in the general, n-dimensional case we have

$$f_i^L = f_{i1} + \sum_{j=1}^{n}(f_{i,j+1} - f_{ij})x_{\nu_j}. \tag{7}$$

The intersection of f_1^L with the GRS is a polygonal line, this is the numerical approximation of the global bifurcation diagram. A simplex only contains a segment of this line if the sign of f_1 was not identical at the three vertices. If there are function values with different signs then the end-points of the line segment in the simplex can be determined by solving three equations systems, each with two unknowns. The first equation in each system is

$$f_1^L = 0, \tag{8}$$

the second one is taken from the three equations below, determining the sides of our simplex:

$$\begin{aligned} x_{\nu_1} &= 1\,, \\ x_{\nu_2} &= x_{\nu 1}\,, \\ 0 &= x_{\nu_2}. \end{aligned} \tag{9}$$

From the three solution points we pick those which satisfy condition (4).

In the genereal, n-dimensional case we can only have a segment of the solution poligon in the investigated simplex if *neither* of the functions f_i $(i = 1, 2, ..., n-1)$ has identical signs at the vertices of the simplex. If this is the case, then the equation system

$$f_i^L = 0, \quad (i = 1, 2, ..., n-1) \tag{10}$$

is subseqently complemented by the equations describing the faces of the simplex:

$$\begin{aligned} x_{\nu_1} &= 1\,, \\ x_{\nu_i} &= x_{\nu_{i-1}}\,, \\ 0 &= x_{\nu_n}\,, \end{aligned} \tag{11}$$

where $i = 2, 3, ..., n$. Thus we have to solve $n+1$ equation systems with n unknowns. After solving them we look for those two solution points the coordinates of which satisfy (4).

When solving the equation system it is worth utilizing the fact that the first $n-1$ equations are always identical, so identical operations have to be performed only once. To reach this goal let us write below the equation system (10) all equations of (11):

$$\begin{bmatrix} \underline{\underline{A}} \\ \underline{\underline{B}} \end{bmatrix} \underline{x} = \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix}, \tag{12}$$

where $\underline{\underline{A}}$ is an $(n-1)\times n$ matrix, $\underline{\underline{B}}$ is an $(n+1)\times n$ matrix, $\underline{x}$ is an n-vector, $\underline{a}$ is an $(n-1)$-vector, $\underline{b}$ is an $(n+1)$-vector and and

$$\begin{aligned}
a_{ij} &= f_{ij} - f_{i,j+1}\,,\\
b_{kj} &= \begin{cases} 1 & \text{if} \quad k = j\,,\\ -1 & \text{if} \quad k = j+1\,,\\ 0 & \text{otherwise}\,, \end{cases}\\
x_j &= x_{\nu_j}\,,\\
a_i &= f_{i1}\,,\\
b_k &= \begin{cases} 1 & \text{if} \quad k = 1\,,\\ 0 & \text{otherwise}\,, \end{cases}\\
i &= 1,2,...,n-1\,,\\
j &= 1,2,...,n\,,\\
k &= 1,2,...,n+1\,.
\end{aligned} \tag{13}$$

By using complete (or partial) selection of principal elements, the matrix on left side of (12) can be transformed by Gaussian elimination to the following form:

$$\begin{array}{r} 1)\\ 2)\\ \vdots\\ n-1)\\ n)\\ n+1)\\ \vdots\\ 2n) \end{array} \begin{bmatrix} 1 & * & * & \dots & & * \\ 0 & 1 & * & \dots & & * \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & * \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}, \tag{14}$$

where $*$ denotes elements which might differ from zero. After this step the $n+1$ equation systems can be solved by solving equations with one unknown. If we did not interchange columns while selecting the principal elements then it is worth after the computation of each coordinate to check inequality (4). If this condition is not satisfied then the computation of this point can be terminated. (Because of the inavoidable numerical errors one prefers to use instead of $u \geq v$ $u > v + \epsilon$, ϵ being a small positive number.)

4. The parallel implementation

The previous section showed how a segment of the solution line can be obtained in a single simplex. Now we proceed to the description of how this process can be efficiently organized for a large number of simplices by utilizing the advantages of parallel computer architecture. The algorithm was implemented under the PVM (Parallel Virtual Machine) system, enabling the user to run individual computers on a distributed network as nodes of a virtual parallel machine. The program was designed according the so-called master-slave model.

The mechanical example is presented in Section 2. is very simple and the number of dimension is small ($n = 2$). A complex problem involves bigger GRS, and the number of dimension grows

rapidly with the complexity of the problem. The CPU and memory requirements of the algorithm grow exponentially with the number of dimensions. In order to solve the equation system with prescribed precision we have to a choose sufficiently small grid-size (Δ_i), so the number of cubes can be very large if the precision requirements are tight. Supposing that the number of points on each coordinate axis is N and the number of dimensions is n, the numbers of points where we have to evaluate each function will be N^n. This exponential expression can yield huge numbers, justifying the parallelization of the algorithm.
To design the parallel algorithm we have examined the simplex algorithm. The main observations are:

a. Every simplex is independent from the results of the calculations in the other simplices.

b. Since adjacent simplices have common vertices, the function values should not be re-computed for each simplex.

c. We assume that the computation time for the the functions f_i is not negligible, and could be different at different points.

d. We expect solution points only in a few simplices, and the most of the simplices do not contain any solution points. (In the limit where the size of the simplices goes to zero we expect solution points "almost nowhere", on a subset of measure zero.)

Considering the first statement, i.e. that the computation in every simplex is independent, it suggests that the simplex could be the element of the parallelization, however the computation steps could be also parallelized in a simplex (e.g. solving the linearized functions and the n different equations of the simplex facets in parallel way). The main disadvantage of the parallelization inside the simplex is that the cost of the communication between two processes in PVM is very high.

The statement 'b' tells that the neighbour simplices have common vertices, so the same function values can be used. Regarding statements 'c', and 'd', the load-balancing is also an important aspect in our case, however, the capacity of the computers of the virtual machine is also different.

The implemented parallel program based on a master-slave structure, where the master program distributes the phase space to smaller pieces (domains) and the slaves figure out the equation system in this these domains. The major functions of the master program:

- reading the configuration files
- starting and stopping the slaves
- collecting the results from slaves
- load-balancing

The slave program essentially contains the serial version of the described Simplex Algorithm, and solves the equations in the domain given by the master. The values of the functions are computed once by the slave, when the slave gets a new domain from the master program. In this manner the function values are multiply computes only on the boundary points between the domains. To minimize the number of boundary points the master program tries to create domains with approximately equal orthogonal sizes.

The load-balancing is provided by the master, because the GRS is divided into more domains than the number of processors. When the computation in any domain has been finished, the master sends the next domain to the next free slave. In this way the faster processors will get more jobs then the slower processors.

The parallel program was developed on heterogeneous environment. We have used for developing some different machines (HP 9000, VAX-750, SUN IPC). This environment is good for testing, but the measurement of the performance is quite difficult since the different computers have different

performances. However, we could test the application, and the load-balancing as well. The presented results were measured on an IBM SP1 computer with 8 processors and 3 RS6000/580 computers connected by Ethernet with TCP/IP protocol. All the computers are connected by NFS.

5. Examples

In this section we present some less trivial examples. The first one is seemingly simple, however, as we will see, this appearance is deceptive.

5.1. The bar clamped at both ends

We investigate the behaviour of the already discussed elastic bar with the difference that now *both* ends are clamped. Since both supports admit vertical forces we have to add a term into the ODE, so, instead of (1) we now have

$$\alpha'' + P\sin(\alpha) - F\cos(\alpha) = 0, \tag{15}$$

where F denotes the vertical force. Instead of P and F one could equally write $H(0)$ and $V(0)$, respectively, expressing that these forces are equal to the horizontal and vertical reactions at the origin. The bar is illustrated in Fig. 4. The GRS is three dimensional, the global coordinates are: $H(0), V(0)$ and $M(0)$, the latter denoting the moment at the origin which could be expressed as $M(0) = \alpha'(0)EI$. (EI denotes the bending stiffness of the bar which was taken as unit, similarly to the length.) The nonlinear equation system, prescribing the boundary conditions at the far end, has two equations:

$$\begin{aligned} \alpha(1) &= f_1(H(0), V(0), M(0)) = 0, \\ y(1) &= f_2(H(0), V(0), M(0)) = 0, \end{aligned} \tag{16}$$

where $y(s)$ is the vertical displacement and is defined by $y'(s) = \sin\alpha$. The bifurcation diagram is plotted in Fig. 4. The physical configurations corresponding to the numerated points of the diagram are illustrated in Fig. 5. The singular behaviour of the "figure 8" configuration was already discovered by Maddocks [7], the fact that this singular branch connects the first two modes was first mentioned by Domokos [4].

In the computation the GRS was divided into 120*200*300=7200000 cubes, and each cube into 2*3=6 simplices. We used different number of computers and only one slave per processor. In each case one slave got a domain not bigger than 37*37*36=49284 cubes. As we have mentioned in the previous section, for getting reasonable load-balancing the master program creates more domains than the number of processors. The IVP for the rod was integrated by the Euler method, using with step-size 1/65=0.0154. (Recall that the length of the rod is unit.)

The slave processes have measured the elapsed times (user, system, real). This time values were gathered by the master program and it was reported at the end of the running. Table 1 displays the measured times with respect to the number of processors. We choose always the worst (maximal) time values. The speed up is computed from the max user + max system times, however the real speed up would come from real times, we have used the user + system times, because we couldn't exclude the extra load on the computers caused by other processes during the measurement period.

As we can see from the table, the parallel algorithm proved to be very efficient, achieving speed-ups up to 6.66.

The difficulty of spurious solutions can be nicely illustrated on this problem. We can obtain two different kinds of these unwanted solutions: the first kind arises from insufficient density of the *physical* mesh (in our case this means the step-size for the Euler method); the second kind is created by the insufficient density of the *mesh in the GRS*. We re-computed the same problem with

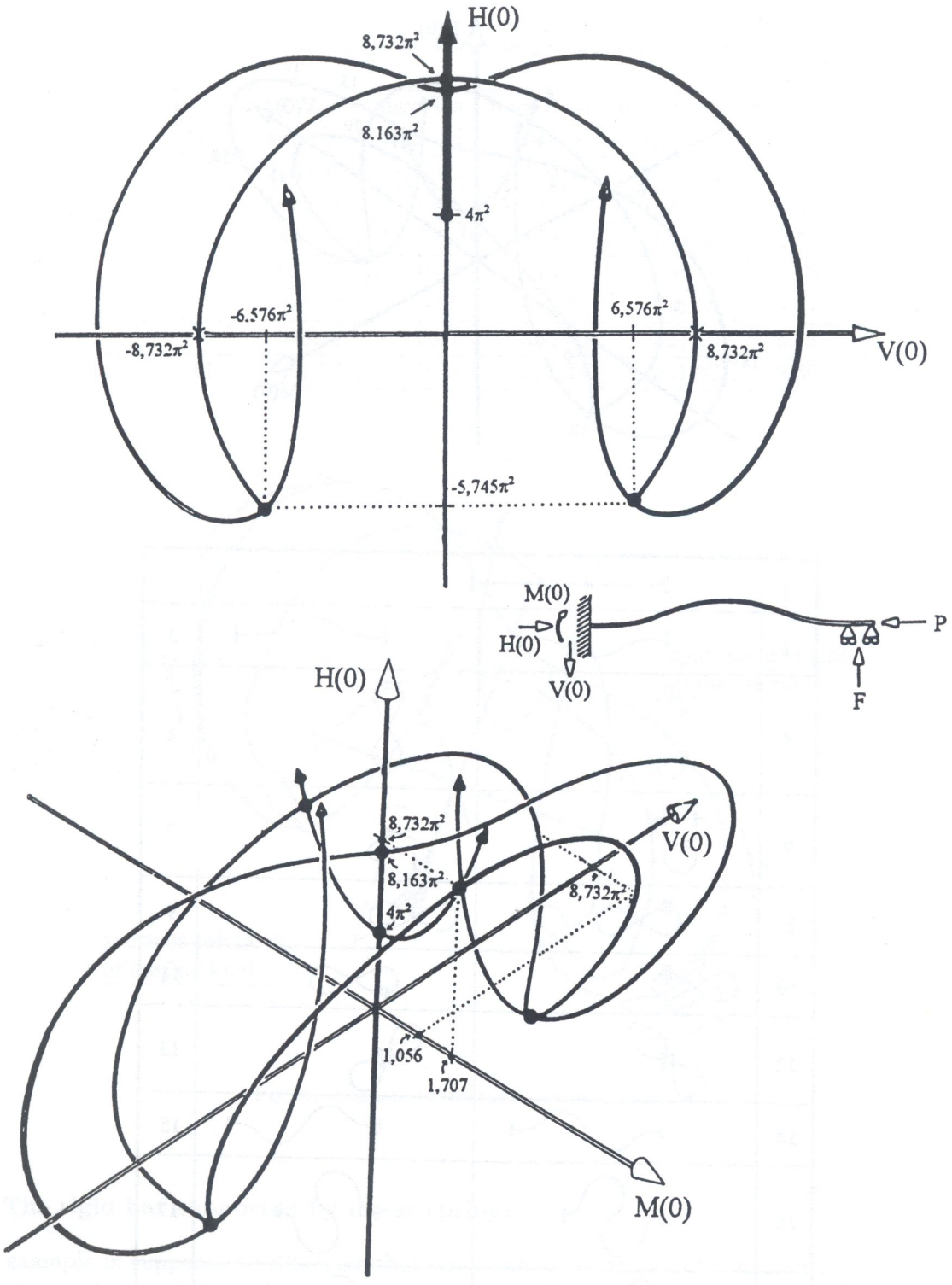
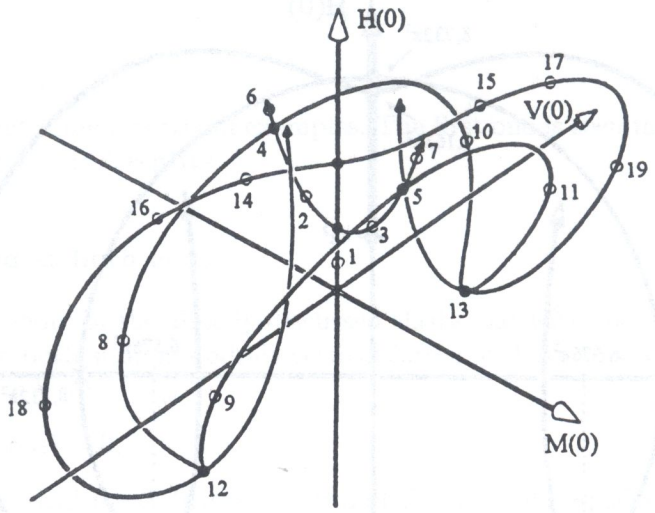


Fig. 4. The bar clamped at both ends



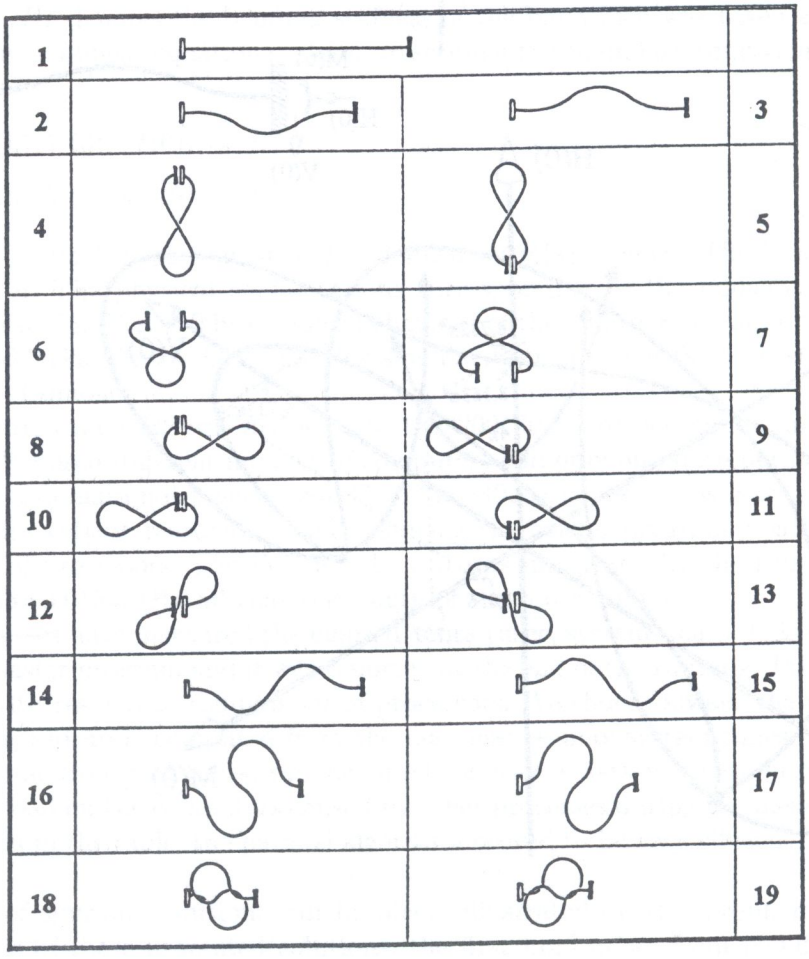


Fig. 5. The physical configurations corresponding of the clamped-clamped bar

a more sparse mesh: we considered 60*100*150=900000 cubes and choose the step-size to be 1/9=0.111. The result of this computation is illustrated in Fig. 6, we indicated the spurious solutions of the first and second kind.

Table 1. The elapsed times depending on number of processors

Processor no	max User time [s]	max Sys time [s]	max Real time [s]	max U+S time [s]	Speed up
1	2143.80	0.46	2166.64	2144.26	1.00
2	1427.57	0.51	1443.93	1428.08	1.50
4	844.44	0.54	860.20	844.98	2.54
6	601.37	0.43	613.44	601.80	3.56
8	431.24	0.35	447.20	431.59	4.97
10	346.78	0.41	355.86	347.19	6.18
11	321.57	0.30	331.38	321.87	6.66

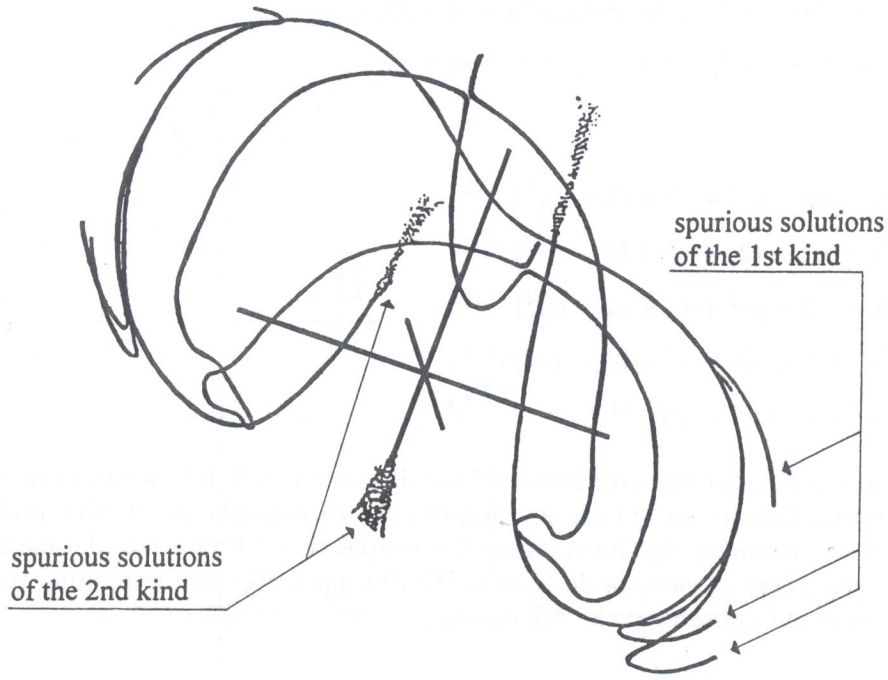


Fig. 6. Spurious solutions in the beam problem

5.2. The rigid bar supported by linear springs

This example is supposed to illustrate that our method is capable of computing the bifurcation diagram even in the vicinity of highly degenerate points. Simultaneously we would like to emphasize the embarassing appearance of spurious solutions.

We investigate the behaviour of the structure illustrated in Fig. 7, consisting of five linear springs.

All connections are spatial (spherical) hinges, transmitting only forces and no moments. The structure is loaded by the vertical force P at the single node. The axial stiffness of each spring is taken as unit, we assume that in the unloaded state they are stress-free and that they remain

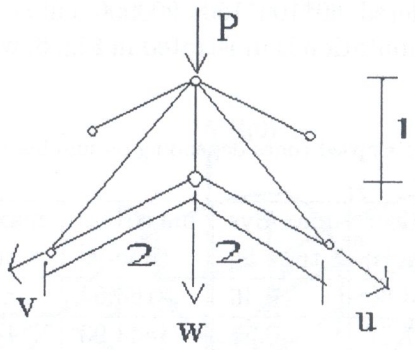


Fig. 7. The structure consisting of five linear springs

straight at all load levels. So this structure has 3 degrees of freedom, in this case the construction of the GRS is trivial. The displacements u, v, w of the node define uniquely the geometry of the structure and from this the bar forces may be computed. The equation system to be solved expresses the horizontal force equilibrium:

$$\begin{aligned} f_1(u,v,w) &= 2(s_1 - s_3) - u(s_1 + s_2 + s_3 + s_4 + s_5)\,, \\ f_2(u,v,w) &= 2(s_2 - s_4) - v(s_1 + s_2 + s_3 + s_4 + s_5)\,, \end{aligned} \tag{17}$$

where

$$\begin{aligned} s_1 &= 1/\sqrt{5} - ((2-u)^2 + v^2 + (1-w)^2)^{-1/2}\,, \\ s_2 &= 1/\sqrt{5} - (u^2 + (2-v)^2 + (1-w)^2)^{-1/2}\,, \\ s_3 &= 1/\sqrt{5} - ((2+u)^2 + v^2 + (1-w)^2)^{-1/2}\,, \\ s_4 &= 1/\sqrt{5} - (u^2 + (2+v)^2 + (1-w)^2)^{-1/2}\,, \\ s_5 &= 1 - (u^2 + v^2 + (1-w)^2)^{-1/2}\,. \end{aligned} \tag{18}$$

Because of the symmetry the primary solution branch is $u = v = 0$. It is worth noting that the first bifurcation point on this trivial branch is a double cusp catastrophe point. The problem has four planes of reflection symmetry. By introducing the simplectic grid with parallel main diagonals in the cubes, two symmetry planes are destroyed. We did not utilize the remaining two symmetries in our computation but considered the full domain

$$\begin{aligned} -0.9999 &\le u \le 1.0000\,, \\ -1.0000 &\le v \le 0.9999\,, \\ -0.00001 &\le w \le 1.0000\,. \end{aligned} \tag{19}$$

(The slight asymmetry in the domain boundaries is aimed to avoid the trivial solution coincidig with the edges of the cubes.) This domain was first subdivided into 100*100*100 cubes (Fig. 8), then into 800*800*800 cubes (Fig.9).

In Fig. 8 we can clearly observe the spurious solutions of the second kind (spurious solutions of the first kind can not arise in this problem since no integration is needed). The second, more dense mesh limited the spurious solutions to the vicinity of the bifurcation point. Similarly to the previous example, we provide a table displaying the speed-up factors depending on the number of processors. The columns of the table have the same meaning as in the previous subsection. The speed-up data was measured for a smaller domain of the GRS.

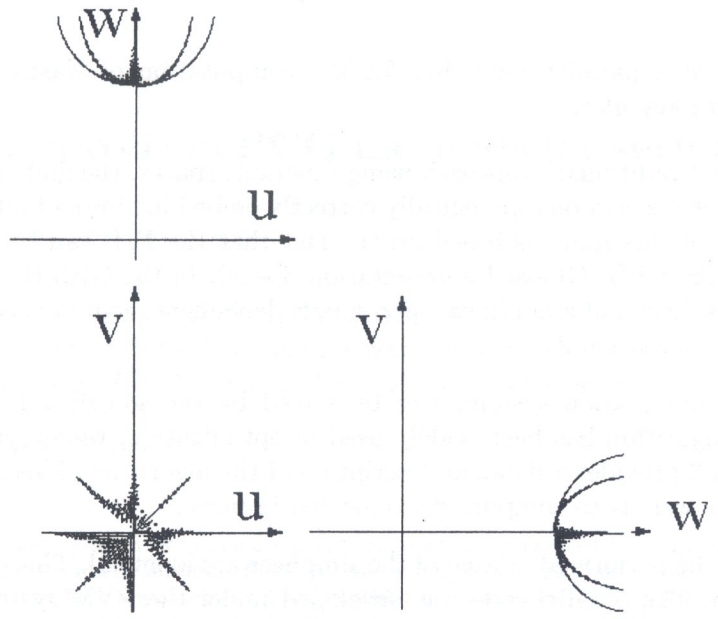


Fig. 8. Computation with 100*100*100 cubes

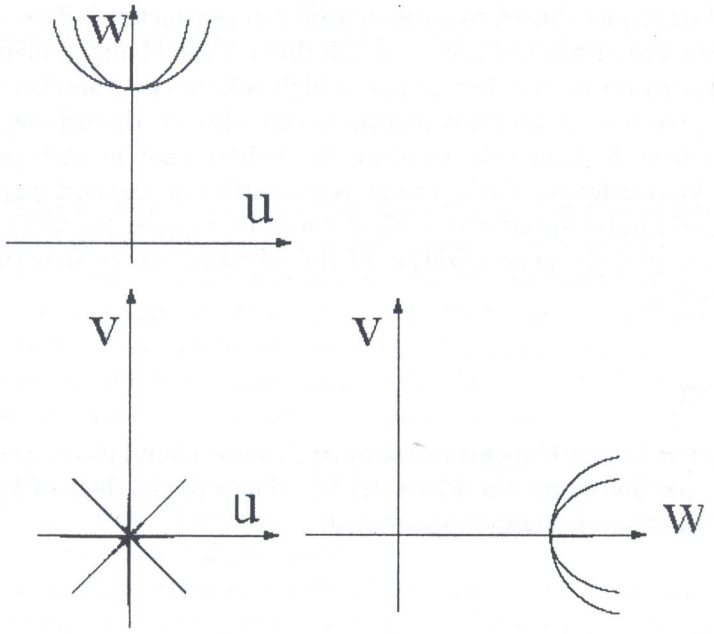


Fig. 9. Computation with 800*800*800 cubes

Table 2. The elapsed times depending on number of processors

Processor no	max User time [s]	max Sys time [s]	max Real time [s]	max U+S time [s]	Speed up
1	875.39	2.95	931.31	878.34	1.00
2	457.80	3.17	469.81	460.97	1.91
4	263.30	3.57	270.99	266.87	3.29
6	159.88	4.12	191.64	164.00	5.36
8	139.53	4.30	178.30	143.83	6.11

6. Summary

This paper introduced a parallel algorithm for the computation of elastic bar structures. The method relies on three key ideas:

1. In contrast to the "traditional" approach using functions spaces, the global bifurcation diagram of elastic bar structures can be topologically correctly embedded into a finite dimensional space. The construction of this space is based on the fact that the IVP can be uniquely solved. We called this space the GRS (Global Representation Space). In the GRS the solution of the BVP is reduced to the solution of a nonlinear, parameter-dependent, algebraic equation system. This ideas are outlined in section 2.

2. Nonlinear algebraic equation systems can be solved by the so-called PL (Piecewise Linear) algorithm. This algorithm has been widely used in optimization, we adopted it for mechanical problems. Section 3 provides a detailed description of the algorithm. This method enables us to avoid iterations as well as to compute disconnected branches.

3. The operations to be performed in each of the simplices are identical. This offers an ideal ground for parallelization. The parallel code was developed under the PVM system and proved to be very efficient, achieving high speed-up factors. The approach to parallelization is described in section 4.

The two presented examples illustrate many features of our method. First of all one can realize that hardly any other method would have been suitable for the clamped-clamped beam problem. The examples also illustrate that one has to pay a high toll on computation time to achieve such general results. The presence of spurious solutions can also be disturbing, however, nearly all discretization method have to fight this problem. We believe that in such problems where none, or very little a priori knowledge on the solutions is available our method might be a good choice. Also, with increasing computer speed the method could be applied to larger problems. Although not mentioned in this paper, the generalization of this ideas to elastic structures (such as frames) is rather straightforward [5].

Acknowledgment

This work was supported by the Hungarian National Science Foundation, in particular, grant No. T 015851 (Zs.G. and I.Sz) and grant No. F7690 (G.D.). The valuable help of Ms. Antónia Szabados in the preparation of the figures is very appreciated.

References

[1] E.L. Allgower, K. Georg. *Numerical continuation methods: an introduction.*Springer, Berlin, 1990.
[2] L. Bieberbach. *Differentialgleichungen.*Springer, Berlin, 1923.
[3] G. Domokos. Can strings buckle? In: *Recent development in stability, vibration and control of structural systems*, AMD -Vol 167, 167–174, ASME 1993, Applied Mechanics Division, ISBN 0-7918-1146-8, 1993.
[4] G. Domokos. Global Description of Elastic Bars *ZAMM*, **74** (4): 289-291, 1994.
[5] G. Domokos, Zs. Gáspár. A Global, direct algorithm for path-following and active static control of elastic bar structures. *J. of Structures and Machines*, **23**(4): 549–579, 1995.
[6] G. Domokos, P.J. Holmes. Euler's problem, Euler's method and the standard map. *J. of Nonlinear Sci.*, **3**: 109–151, 1993.
[7] J.H. Maddocks. Stability of nonlinearly elastic rods. *Arch. Rat. Mech. Anal.*, **85** (4): 311–354, 1984.
[8] H.O. Peitgen, D. Saupe, K. Schmitt. Nonlinear Elliptic Boundary Value Problems Versus Finite Difference Approximations: Numerically Irrelevant Solutions. *J. Reine u. Angew. Math. (Crelle)*, **322**: 74–117, 1981.