periodica polytechnica

Mechanical Engineering
56/2 (2012) 81–86
doi: 10.3311/pp.me.2012-2.01
web: http://www.pp.bme.hu/me

RESEARCH ARTICLE

Received 2012-06-30

Bisection method in higher dimensions and the efficiency number

Dániel Bachrathy / *Gábor* Stépán

Abstract

Several engineering applications need a robust method to find all the roots of a set of nonlinear equations automatically. The proposed method guarantees monotonous convergence, and it can determine whole submanifolds of the roots if the number of unknowns is larger than the number of equations. The critical steps of the multidimensional bisection method are described and possible solutions are proposed. An efficient computational scheme is introduced. The efficiency of the method is characterized by the box-counting fractal dimension of the evaluated points. The multidimensional bisection method is much more efficient than the brute force method. The proposed method can also be used to determine the fractal dimension of the submanifold of the solutions with satisfactory accuracy.

Keywords

Bisection method · multi dimension · system of non-linear equations · multiple roots · efficiency number

Acknowledgement

This work is connected to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project. This project is supported by the New Széchenyi Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002). The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 260073.

Dániel Bachrathy

HAS-BUTE Research Group on Dynamics of Machines and Vehicles, H-1111 Budapest, Műegyetem rkp. 5, Hungary
e-mail: bachrathy@mm.bme.hu

Gábor Stépán

Department of Applied Mechanics, BME, H-1111 Budapest, Műegyetem rkp. 5, Hungary
e-mail: stepan@mm.bme.hu

1 Introduction

The bisection method – or the so-called interval halving method – is one of the simplest root-finding algorithms which is used to find zeros of continuous non-linear functions. This method is very robust and it always tends to the solution if the signs of the function values are different at the borders of the chosen initial interval. Unfortunately, it has only linear convergence, more precisely, it doubles the accuracy with each iteration, which is relatively slow. Hence, it is usually used to find only a proper initial value for alternative root finding methods which have better rate of convergence (e.g.: Newton's method). A big advantage of the bisection method is, however, that it can be applied for non-differentiable continuous functions, too.

Geometrically, root-finding algorithms of $f(x) = 0$ find one intersection point of the graph of the function and the axis of the independent variable. In many applications, this 1-dimensional intersection problem must be extended to higher dimensions, e.g.: intersections of surfaces in a 3D space (volume), which can be described as a system on non-linear equations. In higher dimensions, the existence of multiple solutions become very important, since the intersections of two surfaces can create multiple intersection lines. In this study, we give a description and

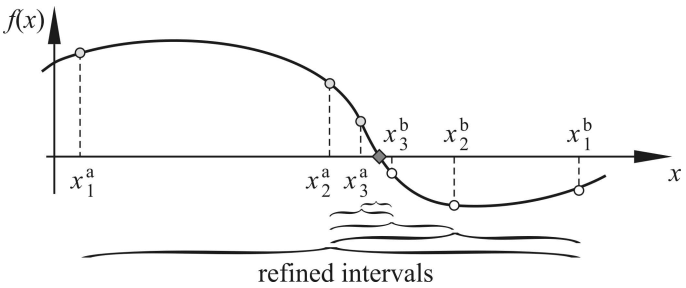

Fig. 1. Schematic representation of the iteration by bisection.

categorization of the root-finding problems in higher dimensions and we create a generalized version of the bisection method for higher dimensions. In Sec. 2, the bisection method is defined and generalized for multiple solutions. In Sec. 3, the basics of the generalization are detailed and the possible solutions of the occurring problems are analyzed. The steps of the multidimensional bisection method are presented in the form of a

flowchart (Fig. 4). Some hints for the numerical method are given in Sec. 4. The computational efficiency of the method is analyzed for different problems in Sec. 5, based on numerical results.

2 Bisection method

The bisection method is used to find the root of a nonlinear real valued scalar function $f(x) = 0$ ($f : \mathbb{R} \to \mathbb{R}$). The method is initialized with the limits of an interval $[x^a, x^b]$, where the function is defined (see Fig. 1). The signs of $f(x^a)$ and $f(x^b)$ must be different. In this case x^a and x^b are bracketing at least one root, since, by the Intermediate Value Theorem, the function f must have at least one zero in the interval (x^a, x^b). At the beginning of the iteration, the midpoint of the interval $x^c = (x^a + x^b)/2$ is computed and $f(x^c)$ is evaluated. If the sign of $f(x^c)$ is the same as the sign of $f(x^a)$, then x^c is set as a new value of x^a, otherwise, x^c is set as a new value of x^b and the iteration is repeated. If x^c happens to be a root of f (i.e., $f(x^c) = 0$) than the iteration is completed in finite number of steps. In this process, the length of the interval is reduced by 50% in each step, leading to a strictly monotone linear convergence. The drawback of this process is that it can find only one intersection point. In the next subsection, a generalized version of this method is described, which is able to find numerous roots.

2.1 Numerous solutions

The original form of the bisection method can easily be extended to find numerous roots of a non-linear equation in a given interval. If the function values are computed in an initial mesh on the examined interval, then the original method can be used for each neighboring points where the sign of the function values are different. This way, some roots may be omitted if the initial mesh is not fine enough, and even number of roots are placed inside one interval. An example is shown in Fig. 2. Note, that the number of the "useful" intervals which are bracketing roots, is the same as the number of the detected roots, so if we use a relatively fine initial mesh, then the detection of all roots can be ensured without the dramatic increas of the computational time. It is also important that the bracketing intervals are mapped into themselves after one iteration step. In case of Newton's method, e.g., the monotone convergence cannot always be guaranteed.

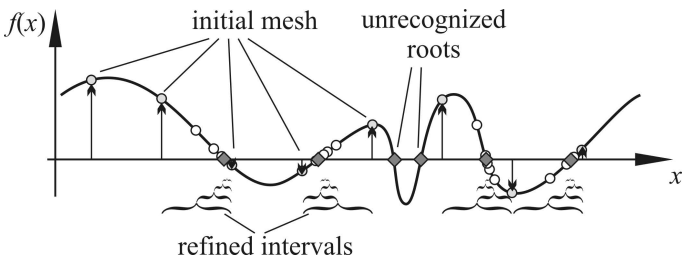

Fig. 2. Numerous solution computed by bisection method, evaluated on an initial mesh.

3 Generalization for higher dimensions

In many applications, the roots of a system of non-linear equations $\mathbf{f}(\mathbf{x})=\mathbf{0}$ must be computed. Let $\mathbf{f} : \mathbb{R}^{Ds} \to \mathbb{R}^{Dc}$ with $\mathbb{R}^{Ds}$ representing the D_S–dimensional vector space and D_C is the codimension (or relative dimension) of the subspace or submanifold of the solution. The topological dimension of the submanifold is $D_T = D_S - D_C$. For example: a surface ($D_T = 2$) can be defined by one equation $D_C = 1$ in a 3D space ($D_S = 3$). Note, that in case of fractal type submanifolds, the topological dimension D_T and the fractal dimension D_F of the object are not the same ($D_T \leq D_F \leq D_S$). For example: Julia set for $c = -1$ is defined by the periodic points of map $g(\mathbf{x}) = (x_1 + ix_2)^2 - 1$ in the complex plane resulting $D_T = 1$ and $D_F = 1.2683$ ($D_S = 2$, $D_C = 1$) [1].

Solution methods already exist and are well developed for the most typical cases. In case $D_F = D_T = 0$, the Newton's method can easily be generalized for higher dimensions by using the gradient of $\mathbf{f}$. Numerical solutions can be found by computer algebraic software for $D_C = 1$ and $D_S = 2, 3$ (see Wolfram Mathematica: *ContourPlot* and *ContourPlot3D*, Matlab: *contour* and *isosurface*, etc.) [2] [3]. For large values of D_S, the continuation methods are used usually (like [4–6]). These are typically applied for $D_F = D_T = 1$, however, multiparameter continuation methods also exist [7]. The main drawbacks of these continuation methods are that the proper initial value is needed and the closed submanifolds (compact submanifolds or isolas) can hardly be detected. From the engineering point of view, it is important to find all the roots in a given domain automatically, which could be a set of closed submanifolds. Our main goal is to create a robust algorithm based on the bisection method, which can determine a properly dense point cloud of all the submanifolds with a given accuracy.

In the following sections, the details of the suggested method and its main steps are described and analyzed.

3.1 n-simplexes and n-cubes

The generalized version of the initial interval of the bisection method must be a higher dimensional object, which can be divided into multiple self-similar objects. The first obvious choice could be the n-simplex, which is a generalization of a triangle or a tetrahedron to higher dimensions. The second obvious choice is the n-cube (or hypercube), which is a generalization of a square or a cube. In a numerical implementation, the n-cube has many advantages. The data corresponding to a node (or vertex) of an n-cube can easily be stored in multilevel arrays (or hypermatrices), the coordinates of the neighbors of an n-cube can easily be determined. Furthermore, the halving of an n-cube along each dimension is trivial, while halving of an n-simplex is complicated in higher dimensions.

3.2 Selection of bracketing n-cubes

In higher dimensions, an initial mesh must be defined to ensure the detection of multiple submanifolds similarly as it

was shown in Sec. 2.1. First, two limit value x_j^a and x_j^b and the size of the mesh N_j must be defined for each dimension ($j \in \{1, 2, ...D_S\}$). The edge length of the initialized n-cube in the j^{th} dimension is $\Delta x_j = (x_j^b - x_j^a)/(N_j - 1)$. In the next step, the function value must be computed in all the $\prod_{j=1}^{D_S} N_j$ points. Before the refinement, the bracketing n-cubes which contain parts of the submanifold must be selected from all the initialized n-cubes. This step also has to be modified in case $D_C > 1$ according to the given problem.

Safe selection

If the sign of f_k is different at any two nodes of an n-cube for all $k = 1, 2, ...D_C$, then it is possible, that the n-cube brackets a part of a submanifold. Unfortunately, this is just a necessary condition, however, this condition can be used to define the n-cube as a bracketing one. In this case, non-bracketing n-cubes will be also selected and refined. This does not lead to computational error if further analysis or iteration is applied, because for smaller refined n-cubes, the apparent bracketing will disappear. The only minor problem is that the unnecessary selections will increase the computational time.

Secant based selection

A more precise selection of the bracketing n-cubes is based on the secant method. In this case, the submanifold inside the n-cube is to be found by linear interpolations. The iteration starts with connecting all the nodes with positive f_1 values to all the nodes with negative f_1 values, then the roots of f_1 along these lines are linearly approximated by the secant method. Then, **f** is linearly interpolated in these points. The iteration is repeated for $f_2, f_3, ...$and f_{D_C}. If in any stage of the iteration, the sign of f_k are the same in all interpolated points, then the next step of the iteration is impossible, which means that the current n-cube is non-bracketing. If an exact **0** is found, then it is a bracketing n-cube.

Fitting hyper-planes

Another possible solution is to fit a hyper-plane to the function values of the nodes at the current n-cube. The normal vectors of the hyper-plane can be computed as well as the closest point of the hyper-plane to the center of the n-cube. If this point is "close enough" to the n-cube then it can be selected as a bracketing cube, since it will be a bracketing n-cube indeed, with high probability.

An advantage of the selection methods presented in this subsection is that the additional evaluation of **f** is unnecessary. Other types of selection methods could be created if we compute **f** at additional points.

The appropriate method of the selection depends on the given problem. The safe selection is very fast, but the computational time increases if the codimension-1 surfaces ($f_j(x) = 0$) are almost parallel at the intersection points. However, it is still advised to use it if the evaluation of **f** is fast enough. In this case the selection of the bracketing n-cubes can be the bottleneck of the computation. Otherwise, if the evaluation of **f** is slow, then the selection based on the secant method or the fitting of hyper-plane give better performance. Based on our experiments, if $D_S < 4$ then the secant based selection is faster than fitting a hyper-plane. Still, these two methods give poor results if the curvatures of the surfaces are relatively large compared to the size of the n-cube. These two methods could possibly mark a bracketing n-cube as a non-bracketing one which could lead to the loss of some parts of the submanifold.

If $D_T \geq 1$, the missing parts of the submanifold can be found by using some kind of continuation method at the very end of the iteration.

3.3 Continuation – neighboring n-cubes

At the end of the iterations all the neighbors of the bracketing n-cubes must be examined. If additional bracketing n-cubes are found then a missing part of the submanifold is detected. These tests should be repeated till all the neighboring n-cubes of the bracketing n-cubes are non-bracketing. In this case, the whole submanifold is found in the selected region. This method is similar to a multiparameter continuation method, because if a small part of the submanifold is detected, then it is extended during the iterations until it is closed into itself (in case of an isola) or until the boundary of the selected range is reached.

This continuation method is very efficient, because the evaluation of **f** is carried out only along the submanifold.

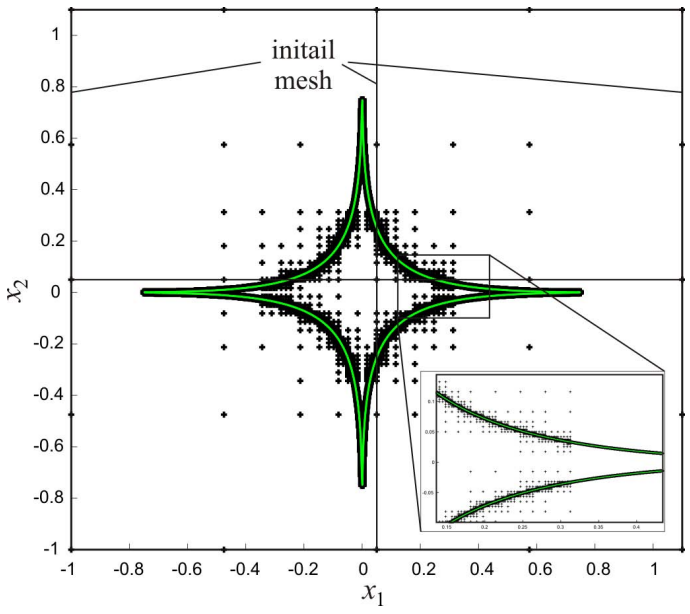

Fig. 3. Numerical result with continuation algorithm. The black crosses denote the points where f is evaluated, the dense bright green dots show up as a line and denote the roots of f computed by the secant method in the final bracketing n-cubes. Parameters are $D_S = 2$, $D_C = 1$, $D_T = D_F = 1$, $x_1^a = x_2^a = -1$, $x_1^b = x_2^b = 1.1$, $N_1 = N_2 = 3$.

An example is shown in Fig. 3, where a unit circle in $1/3$ metric ($f(\mathbf{x}) = ||\mathbf{x}||_{1/3} - 1$, where $||\mathbf{x}||_p = (\sum_{i=1}^{D_S} |x_i|^p)^{1/p}$) is evaluated in a plane. We can clearly see, that the missed thin parts of the submanifold are discovered efficiently by the continuation. The

examination of the neighbouring cubes is not necessary in all cases since it slightly increases the computational time.

The final step of the method is the interpolation of the roots inside the bracketing n-cubes, which can be done by the secant method, the hyper-plane fitting or other more advanced methods. Now, all the necessary steps of the proposed algorithm are described and collected in proper sequence on the flowchart in Fig. 4.

4 Numerical implementation

The proposed method is implemented and tested in Matlab. To create a fast method, indexing of the n-cubes and the nodes should be carried out optimally. It should be stored which n-cubes are bracketing or non-bracketing and which nodes had already been computed, in order to avoid the redundant computation and analysis of a node or an n-cube.

The memory problems can be eliminated in case of large D_S if spares matrices or linearly indexed arrays are used to store the values of $\mathbf{f}$ which are typically computed very sparsely in the selected interval. Still, memory problems could occur due to the large number of computed points if the fractal dimension of the object D_F is large and many iterations are used.

The result of the proposed method is a dense point cloud. In some cases, it is necessary to connect the neighboring points and determine the edges and the surfaces of the detected submanifold (e.g.: for plotting). This additional computation can be done by means of a simplex based algorithm, the details of which are out of the focus of this study.

We computed the point cloud that corresponds to the stability boundary surface of a shimmying wheel, which is described and analyzed deeply in [8]. The surface was determined by a simplex based method. Equation (4.5) in [8] defines the stability boundary surface of a shimmy problem for a 4-parameter space ($D_S = 4, D_C = 2, D_T = D_F = 2$). Our final result in Fig. 5 is the same as Fig. 4, Fig. 5 of the dissertation [8].

5 Efficiency number

In this section, the number points N_P where the function evaluationed is determined in order to analyze the efficiency of the proposed method. The efficiency of the computation time can be measured similarly to the definition of the Hausdorff dimension or the fractal dimension [9] [10]. This dimension is closely related to the box-counting dimension D_0, which considers the number of boxes N_B of size ε that contains the submanifold (attractor) and is given by

$$D_0 = \lim_{\varepsilon \to 0} \frac{\log N_B(\varepsilon)}{\log 1/\varepsilon}. \tag{1}$$

This definition can be applied directly to our situation, where n-cubes are refined:

$$D_x = \lim_{I \to \infty} \frac{\log N_x(I)}{\log 2^I}, \tag{2}$$

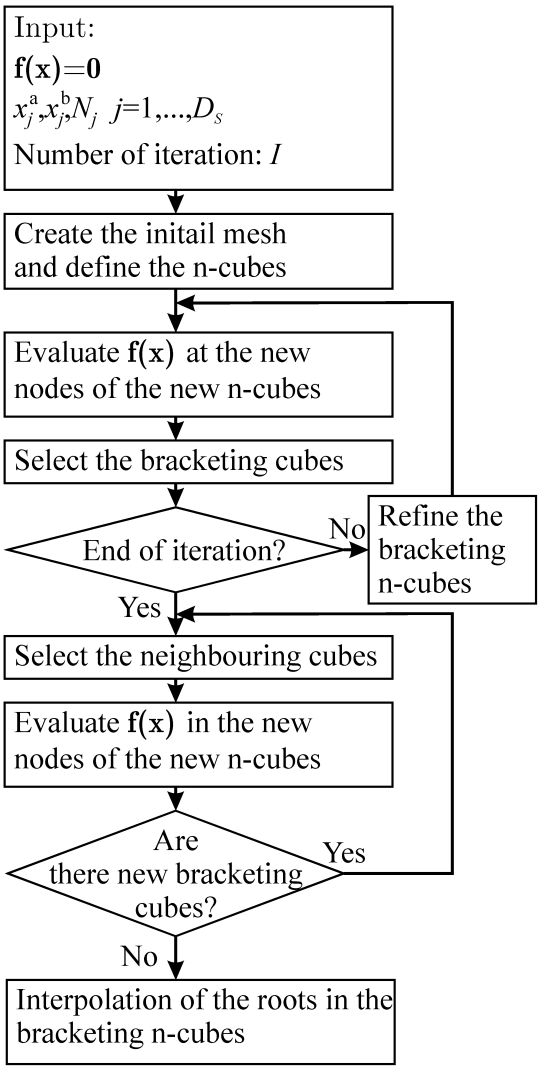

Fig. 4. Flowchart of the proposed Multidimensional Bisection method

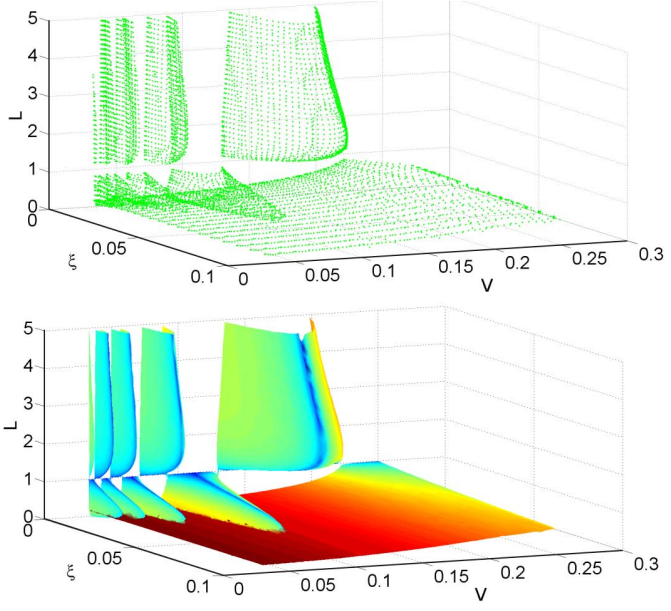

Fig. 5. Point cloud and the fitted surface of the stability boundary of the shimmy problem [8] computed by mutidimensional bisection method. Color map refers to the 4^{th} dimension.

where I is the number of the iteration steps of the refinements

of the n-cubes. If we substitute N_x with the number of bracketing cubes then the fractal dimension of the submanifold D_F is determined, but if the number of function evaluation N_P is substituted, then we get the fractal dimension of the evaluated points D_P. This number shows, how dense the evaluated points are in the space. We define the efficiency number E of the computation method as follows.

$$E = \frac{1}{D_P - D_F}. \tag{3}$$

First, look at the so-called "brute force method", where the function is evaluated at all points in the final mesh for which N_P is given by

$$N_P = \prod_{j=1}^{D_S} \left(\left(N_j - 1 \right) 2^I + 1 \right) \approx \left(2^I \right)^{D_S} \prod_{j=1}^{D_S} N_j. \tag{4}$$

It is clear, that for the brute force method, the fractal dimension of the evaluated points D_P tends rapidly to the dimension of the whole vectorspace D_S, which leads to a small value of the efficiency number $E = 1/(D_S - D_F)$. Note, that $E = 1/D_C$ in case of non-fractal type manifolds.

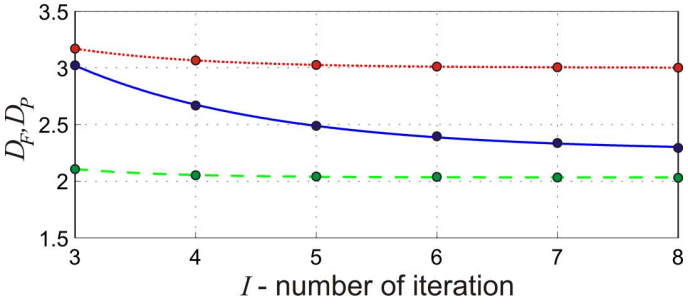

Fig. 6. Convergence of the fractal dimension as a function of the iteration. Dots denote the evaluated box-counting dimension for the integer values of I. Lines denote the fitted exponential curve in a form $D = A + Be^{-\lambda I}$. Green dashed line denotes the fractal dimension of the submanifold D_S. The fractal dimension D_P of the evaluated points for the brute force method and for the proposed multidimensional bisection method are denoted by the red dotted and the blue continuous lines, respectively.

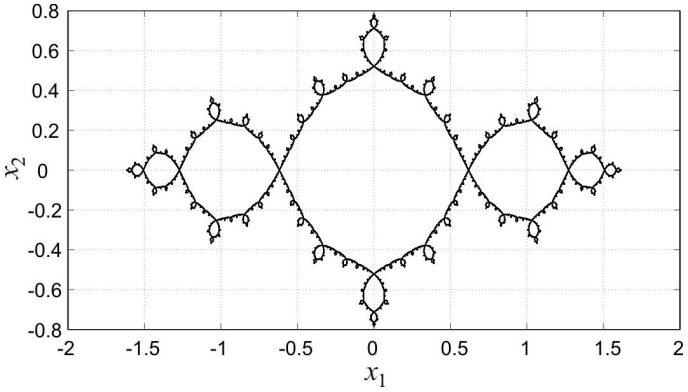

Fig. 7. Julia set for $c = -1$ computed by mutidimensional bisection method.

The limit of expression 2 cannot be determined by a numerical method, however the tendency can be analyzed by increasing the number of iteration I. In the next example, we determine the surface of a unit sphere in Euclidean norm by $f(\mathbf{x}) = \|\mathbf{x}\|_2 - 1$ in a volume defined by $x_j^a = -2$, $x_j^b = 2$, $N_j = 3$ and $j \in \{1, 2, 3\}$ ($D_S = 3$, $D_C = 1$, $D_T = D_F = 2$).

It can be seen in Fig. 6, that the fractal dimensions converge almost exponentially to a given value, which can be approximated by the limit of the fitted exponential curve. The efficiency number of the afore mentioned brute force method tends to 0.9989, which is very close to the correct value $E = 1$. The efficiency number of the proposed multidimensional bisection method tends to $E = 3.7821$ ($D_P = 2.2644$). An ideal perfect method, which computes the dense pointcloud only on the submanifold would have $D_P = D_F$ and $E = \infty$.

The examined unit sphere does not have fractal shape, so, the fractal and the topological dimension must be the same ($D_T = D_F = 2$). The fitted green curve in Fig. 6 tends to $D_F = 2.0328$ which is very close to the topological dimension $D_T = 2$.

The same analysis was performed for the afore mentioned Julia set [1] with an initial mesh $x_j^a = -2$, $x_j^b = 2$, $N_j = 3$ and $j \in \{1, 2\}$ ($\max(I) = 12$), see Fig. 7. During the numerical evaluation, the fractal dimension tends to $D_F = 1.2753$, which has only ~0.5% error. Thus, as an side result, the proposed multidimensional bisection method can be also used to determine the fractal dimension of an object.

The efficiency number was also computed for other non-fractal type problems and the results are collected in Tab. 1. The dimension of the vectorspace D_S and the number of codimensions D_C were varied from 1 to 4. The initial mesh was defined by $x_j^a = -2$, $x_j^b = 2$, $N_j = 3$ and $j \in \{1, 2, ..., D_S\}$. The corresponding equations are

$$f_1(\mathbf{x}) = \|\mathbf{x}\|_2 - 1, \tag{5}$$

$$f_k(\mathbf{x}) = x_k - x_{k-1} \tag{6}$$

for $k > 1, k \in \{1, ..., D_C\}$ which define a unit n-circle in Euclidean space and codimension–1 hyper-planes, respectively.

Tab. 1. The efficiency number of the multidimensional bisection method for Eqs. (5, 6).

$D_S \setminus D_C$	1	2	3	4
1	2.9878	×	×	×
2	5.4795	2.0500	×	×
3	3.7821	1.6603	1.1860	×
4	7.7821	1.3978	0.9936	0.8103

Tab. 1 shows that the computational efficiency is varying based on the given problem. For the ideal method, these numbers would tend to infinity. In all cases, the efficiency number is much better than the one of the brute force method. A linear surface can be fitted well on the computed values of the fractaldimnesion of the evaluated points $D_P = 0.9764D_S - 0.6692D_C \approx Ds - 2/3D_C$, which leads to $E \approx 3/D_C$, which shows that the efficiency is quite large.

6 Conclusions

In this paper, the multidimensional bisection method is derived and deeply analyzed. It is able to find the submanifolds of the roots of a system of nonlinear equations, where the number of uknowns are larger than or equal to the number of equations. The proposed method can be used for the determination of multiple solutions in the selected interval for any high dimensions and codimension. An efficiency number was introduced to characterize the performace of the numerical methods. It is based on the box-counting dimension of the evaluated points. The efficiency number was dermined for the mutlidimensional bisection method in case various problems. It has much better efficiency number than the brute force method, and tends to the ideal one. The efficiency number of the method is smaller than the efficiency number of the continuation methods, but it can find all the isolated submanifolds (including the closed ones) automatically. The main advantage of this method is that the convergence is guaranteed, although, it is only linear. This method can be used as an initialization of more advanced methods which have better convergence. The linear interpolation-based secant method within the small refined final n-cubes gives precise enough results for most of the applications without any additional evaluation of the function. The proposed multidimensional bisection method can also be used for the estimation of the fractal dimension of the computed submanifold. In the given example, the fractal dimension of a Julia set was also determined with a relative error less than 1%.

References

1 **Saupe D**, *Efficient computation of Julia sets and their fractal dimension*, Physica D: Nonlinear Phenomena **28** (1987), no. 3, 358–370, DOI 10.1016/0167-2789(87)90024-8, http://www.sciencedirect.com/science/article/pii/0167278987900248.

2 *Mathematica*, Version 6.0, Wolfram Research, Inc, Champaign, Illinois, 2007.

3 **MathWorks I.**, *Matlab*, R2011b ed, MathWorks, Inc, Natick, Massachusetts, United States, 2011.

4 **Doedel E., Champneys A., Fairgrieve T., Kuznetsov Y. A, Sandstede B, Wang X**, *AUTO97: Continuation and bifurcation software for ordinary differential equations (with HomCont)*, Tech Rep (1997).

5 **Engelborghs K, Luzyanina T, Roose D**, *Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL*, ACM Transactions on Mathematical Software **28** (2002), no. 1, 1–21, DOI 10.1145/513001.513002.

6 **Szalai R**, *Knut: A continuation and bifurcation software for delay-differential equations*, 2009.

7 **Henderson M.**, *Multiple parameter continuation: Computing implicitly defined k-manifolds*, International Journal of Bifurcation and Chaos **12** (2002), no. 3, 451–476, DOI 10.1142/S0218127402004498.

8 **Takács D**, *Dynamics of Towed Wheels - Nonlinear Theory and Experiments*, Budapest University of Technology and Economics, Department of Applied Mechanics, 2011.

9 **Mandelbrot B.**, *The fractal geometry of nature*, W. H. Freeman Press, 1982.

10 **Tél T, Gruiz M**, *Chaotic dynamics: an introduction based on classical mechanics*, Cambridge University Press, 2006.