CREATIVE MATH. & INF.
19 (2010), No. 2, 140 - 148

Online version available at http://creative-mathematics.ubm.ro/
Print Edition: ISSN 1584 - 286X Online Edition: ISSN 1843 - 441X

An implementation of the piecewise-linear homotopy algorithm for the computation of fixed points

ANDREI BOZANTAN

ABSTRACT.

This paper describes the main aspects of the "piecewise-linear homotopy method" for fixed point approximation proposed by Eaves and Saigal [*Homotopies for computation of fixed points on unbounded regions*, Mathematical Programming **3** (1972), 225-237]. The implementation of the method is developed using the modern programming language C# and some interesting details are given.

1. INTRODUCTION

In 1967 Herbert Scarf proposed a method for approximating fixed points of continuous mappings. The algorithm proposed by Scarf, which is also a numerically implementable constructive proof of the Brouwer fixed point theorem, has its origins in the Lemke-Howson complementary pivoting algorithm for solving linear complementarity problems [8]. Beside the generalization and applications in fixed point theory, the Lemke-Howson algorithm is also famous for its applications in finding Nash equilibrium points for bimatrix games, see for example the newer developments in [6]. Several improvements to the algorithm developed by Scarf were made by Terje Hansen in 1967, see [14] and by Harold W. Kuhn in 1968 [7]. But the decisive advancements came in 1972, when Eaves [4] and then Eaves and Saigal [5] described a piecewise-linear (PL) homotopy deformation algorithm as an improvement for the algorithm proposed by Scarf. Another PL algorithm, related to the one proposed by Eaves and Saigal, was presented by Orin H. Merrill in 1972 [9]. The main practical advantage of the PL homotopy methods is that they don't require smoothness of the underlying map, and in fact they can be used to calculate fixed points of set-valued maps. Although PL methods can be viewed in the more general context of complementary pivoting algorithms, see [10], usually they are considered to be included in the special class of homotopy or continuation methods [2].

Starting from this background, our primary aim in this paper is to describe the important aspects of the piecewise-linear homotopy method for fixed point approximation proposed by Eaves and Saigal. Using this presentation we will then describe our implementation of the method which is developed using the modern programming language C#, for which some interesting details are given.

2. BASIC CONCEPTS

The homotopy methods are useful alternatives and aides for the Newton methods in solving systems of n nonlinear equations in n variables:

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n \tag{2.1}$$

mainly when very little a priori knowledge regarding the zero points of F is available and so, a poor starting value could cause a divergent Newton iteration sequence.

The idea of the homotopy methods is to consider a new function $G : \mathbb{R}^n \to \mathbb{R}^n$, related to F, with a known solution, and then to gradually deform this new function into the original function F. Typically one can define the convex homotopy:

$$H(x, \lambda) = \lambda G(x) + (1 - \lambda)F(x) \tag{2.2}$$

and can try to trace the implicitly defined curve

$$H^{-1}(0) = \{x \in \mathbb{R}^n \mid \exists\, \lambda \in [0, 1] \text{ such that } H(x, \lambda) = 0\} \tag{2.3}$$

from a starting point $(x_0, 1)$ to a solution point $(x^*, 0)$. The implicit function theorem ensures that the set $H^{-1}(0)$ is at least locally a curve under the assumption that $(x_0, 1)$ is a regular value of H, i.e., the Jacobian $H'(x_0, 1)$ has full rank n.

However, because we don't require any smoothness conditions on F, a more complex approach involving piecewise-linear approximations is needed.

Definition 2.1. Let $v_1, v_2, \ldots, v_{k+1} \in \mathbb{R}^{n+1}$ be affinely independent points. We call k-**simplex** in $\mathbb{R}^{n+1}$ the convex hull $[v_1, v_2, \ldots, v_{k+1}] := \overline{\text{co}}\{v_1, v_2, \ldots, v_{k+1}\}$. The convex hull $[w_1, \ldots w_{j+1}]$ of any subset $\{w_1, \ldots, w_{j+1}\} \subset \{v_1, v_2, \ldots, v_{k+1}\}$ is an j-**face** of the simplex. Of particular interest are: the 0-faces called **vertices** (vertex), the 1-faces called **edges** and the $(k-1)$-faces called **facets** of the simplex. The facets are important in the PL methods and are obtained by eliminating one vertex of the simplex.

Received: 15.06.2010; In revised form: 28.07.2010; Accepted: 15.08.2010.
2000 *Mathematics Subject Classification.* 47H10, 65B99.
Key words and phrases. *Fixed point, piecewise linear, homotopy, algorithm, implementation.*

Definition 2.2. Let $\mathcal{T}$ be a non-empty family of $(n+1)$-simplexes in $\mathbb{R}^{n+1}$. We call $\mathcal{T}$ a **triangulation** of $\mathbb{R}^{n+1}$ if:

(1) the reunion of the simplexes cover all $\mathbb{R}^{n+1}$;
(2) any two simplexes in $\mathcal{T}$ intersect in a common face or not at all;
(3) any bounded set in $\mathbb{R}^{n+1}$ intersects only finitely many simplexes in $\mathcal{T}$.

In the following discussion we assume that $H : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is a map and $\mathcal{T}$ is a triangulation of $\mathbb{R}^{n+1}$. We want to approximate the components of $H^{-1}(0)$ by using only the values of H on the nodes of $\mathcal{T}$.

If σ is a simplex of $\mathcal{T}$ and $\tau \in \sigma$ is a facet of σ, then there is exactly one simplex $\tilde{\sigma} \in \mathcal{T}$ such that $\tilde{\sigma} \neq \sigma$ and $\sigma \cap \tilde{\sigma} = \tau$. Also there is exactly one vertex $v \in \sigma$ which is not a vertex of $\tilde{\sigma}$ and there is exactly one vertex $\tilde{v} \in \tilde{\sigma}$ which is not a vertex of σ. We call v the vertex of σ **opposite** τ and $\tilde{v}$ the vertex of $\tilde{\sigma}$ opposite τ. We say that the simplex σ is **pivoted** across τ into $\tilde{\sigma}$ and that the vertex v of σ is pivoted into $\tilde{v}$.

Definition 2.3. We call two simplexes $\sigma, \tilde{\sigma} \in \mathcal{T}$ **adjacent** if they meet in a common facet $\tau = \sigma \cap \tilde{\sigma}$.

In the PL algorithms, a triangulation is stored in the computer's memory using only a current simplex and a current facet of this simplex. A **pivoting step** is used to move from the current simplex to an adjacent simplex, which then becomes the new current simplex.The pivoting step is performed in the PL algorithms by a subroutine which implements the pivoting rules of the triangulation.

Definition 2.4. The **piecewise linear approximation** of H (with respect to $\mathcal{T}$) is defined as the union

$$H_{\mathcal{T}} = \bigcup_{\sigma \in \mathcal{T}} H_\sigma \tag{2.4}$$

where $H_\sigma : \sigma \to \mathbb{R}^n$ is the uniquely defined affine map which coincides with H on the vertices of σ.

H_σ can be uniquely extended to an affine map on the affine space spanned by σ. The Jacobian H'_σ has the property $H'_\sigma(x-y) = H_\sigma(x) - H_\sigma(y)$ for x, y in this affine space.

Definition 2.5. A point $x \in \mathbb{R}^{n+1}$ is called a **regular point** of the PL map $H_{\mathcal{T}}$ if x is not contained in any face of dimension smaller than n, and if H'_τ has maximal rank n for all faces τ containing x. A value $y \in \mathbb{R}^n$ is a **regular value** of $H_{\mathcal{T}}$ if all points in $H^{-1}(y)$ are regular. If a point is not regular it is called **singular**. If a value is not regular it is called singular.

3. Piecewise Linear Methods

For $\epsilon > 0$, we introduce the following notation:

$$\overrightarrow{\epsilon} := \begin{pmatrix} \epsilon^1 \\ \vdots \\ \epsilon^N \end{pmatrix} \tag{3.5}$$

Definition 3.6. We call an n-simplex τ **completely labeled** if it contains solutions of the equation $H_\tau(v) = \vec{\epsilon}$ for all sufficiently small $\epsilon > 0$.

In other words, we define an n-simplex τ to be completely labeled if it contains a zero point of the PL approximation H_τ, and if this property of τ is stable under certain small perturbations in the above sense.

Definition 3.7. An $(n+1)$-simplex $\sigma \in \mathcal{T}$ is called **transverse** (with respect to H) if it contains a completely labeled n-face.

Definition 3.8. If $\tau = [v_1, \ldots, v_{n+1}] \in \mathcal{T}$ is an n-simplex, we call the matrix:

$$\mathcal{L}(\tau) := \begin{pmatrix} 1 & \cdots & 1 \\ H(v_1) & \cdots & H(v_{n+1}) \end{pmatrix} \tag{3.6}$$

the **labeling matrix** on τ induced by H.

Proposition 3.1. *The n-simplex $\tau = [v_1, \ldots, v_{n+1}]$ is completely labeled if and only if: the labeling matrix $\mathcal{L}(\tau)$ is nonsingular and $\mathcal{L}(\tau)^{-1}$ is **lexicographically positive** i.e. the first non-zero entry in any row of $\mathcal{L}(\tau)^{-1}$ is positive.*

Proposition 3.2 (Door-In-Door-Out Principle [2])**.** *An $(n+1)$-simplex has either zero or two completely labeled n-faces.*

We assume that σ is transverse and consider the equation $H_\sigma(v) = \vec{\epsilon}$ for $v \in \sigma$. An analogue version of Sard's theorem - the perturbation lemma - can be demonstrated for piecewise linear maps, see [2]. From this lemma we have that for any compact subset $C \subset \mathbb{R}^{n+1}$ there are at most finitely many $\epsilon > 0$ such that $C \cap H_{\mathcal{T}}^{-1}(\vec{\epsilon})$ contains a singular point of $H_{\mathcal{T}}$. As a consequence, $\vec{\epsilon}$ is a regular value of $H_{\mathcal{T}}$ for almost all $\epsilon > 0$. So for $\epsilon > 0$ being sufficiently small, the solutions v form a line which does not intersect lower-dimensional faces of σ. Hence, the line intersects exactly two n-faces of σ. These two n-faces cannot change as $\epsilon \to 0$, since otherwise a lower dimensional face would be traversed and the perturbation lemma would be contradicted. In other words, exactly two n-faces of σ contain solutions of the equation $H_\sigma(v) = \vec{\epsilon}$ for $\epsilon > 0$ being sufficiently small. □

The PL algorithm for tracing certain components of $H_{\mathcal{T}}^{-1}(0)$ can now be easily described via the Door-In-Door-Out-Principle. Heuristically, let us imagine that the $(n+1)$-simplexes $\sigma \in \mathcal{T}$ are "rooms" in an "infinite" building $\mathcal{T}$, and the "walls" of a room σ are its n-faces τ. A wall has a "door" if it is completely labeled. Hence a room has either no or exactly two doors. The algorithm consists of passing from one room to the next, and the following rule must be obeyed: if a room is entered through one door, it must be exited through the other door [2].

Generic PL Continuation Algorithm
input
begin
$\sigma_0 \in \mathcal{T}$ transverse (*starting simplex*)
τ_0 completely labeled n-face of σ_0;
end
repeat for $n := 1, 2, \ldots$
find $\sigma_n \in \mathcal{T}$, $\sigma_n \neq \sigma_{n-1}$ (*pivoting step*)
such that $\tau_{n-1} = \sigma_n \cap \sigma_{n-1}$;
find the completely labeled n-face τ_n of σ_n (*door-in-door-out step*)
such that $\tau_n \neq \tau_{n-1}$;
until traversing is stopped.

In the general PL methods a suitable starting simplex has to be constructed, but for the special case of homotopy methods the choice of the starting simplex is straightforward.

The pivoting step finds the simplex which is adjacent to the current simplex on a given facet. The implementation of the pivoting step depends on the chosen triangulation and typically is performed using only a few operations which define the pivoting rules of the triangulation.

The door-in-door-out step finds the second completely labeled n-face of a transverse simplex. This step is computationally more expensive than the pivoting step because it involves the solving of linear equations in a manner typical for linear programming methods and so it is also referred as linear programming (LP) step.

4. Piecewise Linear Homotopy Algorithms

The idea of the piecewise linear continuation algorithm can be used to approximate a fixed point of a continuous bounded map $F : \mathbb{R}^n \to \mathbb{R}^n$. We consider the homotopy:

$$H : \mathbb{R}^{n+1} \to \mathbb{R}^n, \quad H(x, \lambda) = x - \lambda x_0 - (1 - \lambda)F(x) \tag{4.7}$$

and we try to follow the solution path from $(x_0, 1)$ to $(x^*, 0)$, where x_0 is the initial approximation of the solution and x^* is a fixed point of F which we try to approximate. Because there are no smoothness conditions on F, we use the piecewise linear continuation algorithm in conjunction with a special triangulation.

We consider a triangulation $\mathcal{T}$ of $\mathbb{R}^n \times (0, 1]$ into $(n+1)$-simplexes σ such that every simplex is contained in some slab $\mathbb{R}^n \times [2^{-k}, 2^{-k-1}], k \in \mathbf{N}$. Let $\sigma = [v_1, v_2, \ldots, v_{n+1}, v_{n+2}] \in \mathcal{T}$ be an $(n+1)$-simplex and let $\pi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ be the following canonical projection: $\pi(x, \lambda) = \lambda$. We define the level of σ as $\max_{i=\overline{1,n+2}} \pi(v_i)$, which is the maximum of the last co-ordinates of all vertices of σ. We call $\mathcal{T}$ a refining triangulation if the diameter of σ tends to zero as the level of σ tends to zero.

We define the piecewise linear map $H_{\mathcal{T}}$ which interpolates H on the vertices of the given refining triangulation $\mathcal{T}$. It can be shown that $\mathcal{T}$ induces a triangulation on the frontier of $\mathbb{R}^n \times (0, 1]$ so we have an induced triangulation over $\mathbb{R}^n \times \{1\}$. If we assume that the starting point $u_0 := (x_0, 1)$ is in the interior of a facet τ_0, then it is clear that τ_0 is the only completely labeled facet on $\mathbb{R}^n \times \{1\}$. So the algorithm cannot terminate on the boundary $\mathbb{R}^n \times \{1\}$, so it generates a sequence $\tau_0, \tau_1, \ldots$ of completely labeled facets. It is possible to follow the polygonal solution path parametrized by arclength $0 \leq s < s_0 \leq \infty$, $c(s) = (x(s), \lambda(s))$ in $H^{-1}(0)$, starting at $c(0) = (x_0, 1)$. From the boundedness of the map it follows that $\lambda(s)$ tends to zero as s tends to s_0 and

$$\lim_{s \to s_0} ||x(s) - F(x(s))|| = 0 \tag{4.8}$$

Since $x(s)$ remains bounded as s tends to s_0, this implies that every accumulation point of $x(s)$ is a fixed point of F.

5. Implementation details

5.1. **Input and Output.** The input data required by the algorithm is:

- n - the dimension of the problem;
- $F : \mathbb{R}^n \to \mathbb{R}^n$ - the function for which we calculate a fixed point, specified as an expression depending on the variables $x_1, x_2, \ldots, x_n$;
- $v_1, \ldots, v_{n+1} \in \mathbb{R}^n$ - the vertices of a starting simplex;
- $x_0 \in \mathbb{R}^n$ - an initial approximation of the fixed point.

There are also some parameters for flow controlling:

- the minimum required contraction factor for the Newton steps;

- the maximal allowed bisection level of the triangulation;
- the maximal number of steps to be performed;
- the stopping tolerance for $||F||$.

If an approximate solution is found the algorithm will output this solution and the number of steps performed in order to find it. During the execution, the algorithm will output the current point of the followed polygonal path.

5.2. **The Triangulation and Pivoting Step.** The triangulation used in the algorithm is the refining triangulation J_3 of $\mathbb{R}^n \times (0,1]$, but in the implementation any other refining triangulation can be easily used instead of J_3. The presentation of the J_3 triangulation and the corresponding pivoting rules follows the description given by Saigal [12].

The vertices of the triangulation J_3 are given by the following set of points:

$$J_3^0 = \{v = (v_1, \ldots, v_{n+1}) \mid v_{n+1} = 2^{-k}, k \in \mathbf{N} \text{ and } \frac{v_i}{v_{n+1}} \in \mathbb{Z}, i = i, \ldots, n\}. \tag{5.9}$$

In addition, the following subset of vertices in J_3^0 are called central vertices:

$$\overline{J}_3^0 = \{v = (v_1, \ldots, v_{n+1}) \in J_3^0 \mid \frac{v_i}{v_{n+1}} \text{ is an odd integer}, i = 1, \ldots, n\}. \tag{5.10}$$

A vertex $v \in J_3^0$ has the level k if $v_{n+1} = 2^{-k}$. Each central vertex of depth $k \geq 1$ has an unique nearest central vertex of depth $k-1$ obtained as

$$y(v) = v - v_{n+1} \cdot \nu(v) \tag{5.11}$$

where

$$\nu(v) = (\nu_1(v), \nu_2(v), \ldots, \nu_{n+1}(v))$$

$$\nu_i(v) = \begin{cases} -1, & \text{if } \frac{v_i}{v_{n+1}} \bmod 4 = 1 \\ +1, & \text{if } \frac{v_i}{v_{n+1}} \bmod 4 = 3. \end{cases} \tag{5.12}$$

Each simplex $\sigma \in J_3$ has an unique representation by a triplet (v, π, s) where $v = (v_1, \ldots, v_{n+1})$ is a central vertex, π a permutation of $\{1, \ldots, n+1\}$ and $s = (s_1, \ldots, s_{n+1})$ with $s_i \in \{-1, +1\}$. The vertices of this simplex are obtained as follows:

$$\begin{aligned} &\gamma_1 = v \\ &\gamma_{i+1} = \gamma_i + v_{n+1} s_{\pi(i)} e_{\pi(i)} && i = 1, \ldots, j-1 \\ &\gamma_{j+1} = \gamma_j - v_{n+1} \sum_{k=j+1}^{n+1} \nu_{\pi(k)} e_{\pi(k)} + v_{n+1} e_{n+1} \\ &\gamma_{k+1} = \gamma_k + 2x_{n+1} \nu_{\pi(k)} e_{\pi(k)} && j+1 \leq k \leq n+1 \end{aligned} \tag{5.13}$$

where $\nu = \nu(v)$, $e_1, \ldots, e_{n+1}$ is the standard unit basis in $\mathbb{R}^{n+1}$ and j is the index such that $\pi(j) = n+1$.

Next we will present the pivoting rules for the J_3 triangulation. Let (v, π, s) be a simplex in J_3, and let $\pi(j) = n+1$. Also, let $\gamma_1, \ldots, \gamma_{n+2}$ be the vertices of this simplex generated by the above relations. We define the n-dimensional vector b such that $b_{\pi(i)} = s_{\pi(i)}$ for $1 \leq i \leq j-1$ and $b_{\pi(i)} = \nu_{\pi(i)}$ for $j+1 \leq i \leq n+1$. We assume that γ_i is the vertex which has to be pivoted. The new simplex (v', π', s') is obtained by the following rules.

$i = 1, j = 1$	$v' = v - v_{n+1}(-1, b)$ $\pi' = (\pi(2), \ldots, \pi(n+1), \pi(1))$ $b' = b$
$i = 1, j > 1$	$v' = v + 2v_{n+1}b_{\pi(1)}e_{\pi(1)}$ $\pi' = \pi$ $b' = b - 2b_{\pi(1)}e_{\pi(1)}$
$2 \leq i \leq j - 1$	$v' = v$ $\pi' = (\pi(1), \ldots, \pi(i+1), \pi(i), \ldots, \pi(n+1))$ $b' = b$
$i = j, b_{\pi(j-1)} = \nu_{\pi(j-1)}$	$v' = v$ $\pi' = (\pi(1), \ldots, \pi(j), \pi(j-1), \ldots, \pi(n-1))$ $b' = b$
$i = j, b_{\pi(j-1)} = -\nu_{\pi(j-1)}$	$v' = v$ $\pi' = (\pi(1), \ldots, \pi(j-2), \pi(j), \ldots, \pi(n+1), \pi(j-1))$ $b' = b - 2b_{\pi(j-1)}e_{\pi(j-1)}$
$j + 1 \leq i \leq n + 1$	$v' = v$ $\pi' = (\pi(1), \ldots, \pi(i+1), \pi(i), \ldots, \pi(n+1))$ $b' = b$
$i = n + 2, j = n + 1$	$v' = v + \frac{1}{2}v_{n+1}(-1, b)$ $\pi' = (\pi(n+1), \pi(1), \ldots, \pi(n))$ $b' = b$
$i = n + 2, j < n + 1$	$v' = v$ $\pi' = (\pi(1), \ldots, \pi(j-1), \pi(n+1), \pi(j), \ldots, \pi(n))$ $b' = b - 2b_{\pi(n+1)}e_{\pi(n+1)}$

5.3. **The Linear Programming Step.** In the linear programming step we try to find the second completely labeled n-face of a transverse simplex. This is usually a computational intensive step since involves solving of linear equations in a manner typical for linear programming methods, so it has to be implemented using efficient methods.

We denote the current transverse $(n + 1)$-simplex by $\sigma_n = [v_1, \ldots, v_{n+2}]$ and its current completely labeled n-face by $\tau_{n-1} = [v_1, \ldots, v_{n+1}]$. Hence the vertex v_{n+2} was just pivoted in the preceding step. Our aim in the linear programming step is to find a completely labeled n-face τ_n of σ_n which is different from τ_{n-1}. So we have to find an index $i \in \{1, \ldots, n + 1\}$ such that the right inverse of the matrix obtained by deleting the i-th column of $A = \mathcal{L}(\sigma_n)$ is lexicographically positive, see 3.1. One index which satisfies this is $i = n + 2$, since the n-face opposite v_{n+2} is completely labeled. The other index which will be obtained is the new index we are seeking in the linear programming step.

For numerical purposes, the lexicographical operations in linear programming are usually performed only over the first coordinate of the rows. Theoretically, in cases of degeneracies, an algorithm based on this simplified test could cycle, but this is rarely observed in practice.

Using this simplification, the numerical linear algebra of the "door-in-door-out" step consists of solving the equations

$$A\gamma = 0, \gamma[j] = -1$$

$$A\alpha = e_1$$

for $\gamma, \alpha \in \mathbb{R}^{n+2}$ and where the index j is given and corresponds to the known completely labeled n-face. Then a minimization

$$\min \left\{ \frac{\alpha[i]}{\gamma[i]} \;\middle|\; \gamma[i] > 0, i = 1, \ldots, n + 2 \right\}$$

is performed to find the index i corresponding to the vertex to be pivoted next. The vertex v_i is pivoted into the vertex $\tilde{v}_i$. The corresponding label

$$y := \begin{pmatrix} 1 \\ H(\tilde{v}_i) \end{pmatrix}$$

is calculated, and the new labeling matrix $\tilde{A}$ is obtained by replacing the i-th column of A by y:

$$\tilde{A} = A + (y - Ae_i)e_i^{\intercal}.$$

At each step, a standard decomposition of $\tilde{A}$ is updated from a given decomposition of A, which enables us to solve the linear equations at each step. The cheapest such method directly updates the right inverse B of A such that $e_i^{\intercal}B = 0$ for the current pivot index i. However, this update method is not always stable, so we actually update a QR factorization of $\tilde{A}$ in each cycle. The required matrix operations are performed using the ALGLIB library [1].

5.4. **Acceleration Techniques.** In order to improve the convergence rate of the algorithm a technique of mixing PL steps with Newton steps can be used, see [11]. This follows from the fact that a modified Newton's method expressed in barycentric coordinates leads to a system of linear equations which is closely related to the linear equations obtained in the linear programming step and in fact it is possible to continue the updates of the labeling matrix during the Newton steps on the column corresponding to the vertex which has to be pivoted next. The Newton steps are performed when each new level $\mathbb{R}^n \times \{k\}$ is reached in the algorithm. If the contraction of the Newton steps is sufficiently strong the algorithm will continue to perform these steps until it stops with an approximate solution. If the Newton steps are not successful then the pivoting step is executed and the PL homotopy algorithm continues normally.

6. Conclusions and future work

The main practical advantages of the PL homotopy methods is that they don't require smoothness of the underlying map, so in theory they have a very large range of applicability. Also another important feature of these methods is that they can be applied when no a priori knowledge regarding the solutions of the system to be solved is available.

The implementation of this algorithm in a modern programming language will make possible a comparative study with some other methods for solving non-linear equations, e.g. Newton's method. Some preliminary results are given in [3]. We also want to study possible applications for the newer developments related to the piecewise-linear homotopy methods, see for example [6]. Another important research direction is to study the feasibility of parallelizing some parts of this algorithm.

Acknowledgements. This work was partially financed by European Social Fund through the Sectoral Operational Programme Human Resources Development 2007-2013.

References

[1] ALGLIB - a cross-platform numerical analysis and data processing library, www.alglib.net
[2] Allgower, E. L. and Georg, K., Introduction to numerical continuation methods, Colorado State University, 1990
[3] Bozântan, A. and Berinde, V., *Applications of the PL homotopy algorithm for the computation of fixed points to unconstrained optimization problems*, Proceedings of SYNASC 2010 (accepted)
[4] Eaves, B. C., *Homotopies for computation of fixed points*, Mathematical Programming **3** (1972), 1-22
[5] Eaves, B. C. and Saigal, R., *Homotopies for computation of fixed points on unbounded regions*, Mathematical Programming **3** (1972), 225-237
[6] Herings, P. J-J. and Peeters, R., *Homotopy methods to compute equilibria in game theory* Economic Theory, **42** (2010), 119-156
[7] Kuhn, H. W., Simplicial approximation of fixed points, *Proceedings of the National Academy of Sciences*, 1968
[8] Lemke, C. E., *Bimatrix equilibrium points and mathematical programming*, Management Science **11** (1965), No. 7, 681-689
[9] Merril, O. H., Applications and extensions of an algorithm that computes fixed points of a certain upper semi-continuous point to set mapping, Ph. D. Disertation, The University of Michigan, 1972. Ann Arbor, Michigan, 48106, USA
[10] Murty, K. G., Linear complementarity, linear and nonlinear programming, Heldermann, Berlin, 1988
[11] Saigal, R. and Todd, M. J., *Efficient acceleration techniques for fixed points algorithms*, SIAM Journal of Mathematics **15** (1978), 997-1007
[12] Saigal, R., *Fixed point computing methods*, Encyclopedia of computer science an technology **8** (1979), 545-566
[13] Scarf, H. E., *The approximation of fixed points of a continuous mapping*, SIAM Journal on Applied Mathematics **15** (1967), No. 5, 1328-1343
[14] Scarf, H. E. (in collaboration with Hansen, T.), The computation of economic equilibria, Yale University Press, New Haven, Connecticut, 1973

North University of Baia Mare
Department of Mathematics and Computer Science
Victoriei 76, 430122, Baia Mare, Romania
E-mail address: andrei.bozantan@gmail.com