CREAT. MATH. INFORM. **23** (2014), No. 1, 41 - 50
Online version at http://creative-mathematics.ubm.ro/
Print Edition: ISSN 1584 - 286X Online Edition: ISSN 1843 - 441X

About the implementation and some applications of the FIXPOINT software minipackage

ANDREI BOZANTAN and VASILE BERINDE

ABSTRACT. The main aim of this note is to briefly present the implementation, main features and some current and potential applications of the software minipackage FIXPOINT, designed and implemented by the first author and used by the second author in solving some concrete problems.

1. INTRODUCTION

In the iterative approximation of fixed points there exist various classical and new specific methods to compute the fixed points of concrete functions or mappings, see for example the monographs [9], [19] and [17]. Most of these methods are extensions of the Picard iteration or method of successive approximations: Krasnoselskij iterative method, Mann iterative method, Ishikawa iterative method and so on.

For Picard iteration there exist several software packages implemented in various large computer algebra (Mathematica, Maple, MathCAD etc.)

There exist also several fixed point type iterative methods, like Newton method, which are separately studied and implemented.

Having in view the fact that for Krasnoselskij, Mann and Ishikawa iterative methods there are no corresponding implementations, a first version of the FIXPOINT minipackage has been designed and implemented in [15].

It was later used to perform numerical experiments for the comparison study of fixed point iteration procedures that were reported in [9] and [10]. By inspecting the empirical results obtained by means of FIXPOINT for several fixed point iterative methods, some theoretical results have been also inferred and proved in [7], [8] and continued by other authors [3]-[5], [20]-[23], [27]-[37], [39].

Later, the FIXPOINT minipackage has been extended to include k-step fixed point iterative methods.

The aim of this note is to present the main features FIXPOINT minipackage, some details about its implementations and also to illustrate its use by some concrete applications.

2. FIXED POINT RELATED SOFTWARE

In this section we will analyse existing software alternatives for computing fixed points using iterative numerical methods and two different categories of software will be described: specialized software packages and general purpose computer algebra systems (CAS). We start our description with the CAS category and we will analyse the features

Received: 25.05.2014. In revised form: 26.05.2014. Accepted: date
2010 *Mathematics Subject Classification.* 47H10, 47J25, 49M15, 49M30.
Key words and phrases. Fixed point, piecewise linear, homotopy, algorithm, implementation, optimization problem, Schwefel's function .
Corresponding author: Vasile Berinde; vberinde@ubm.ro

related to fixed point iterations, offered in some of the most used and well known general purpose CAS: Mathematica, Maple and Matlab.

From all the software analysed in this paper, including the category of specialized software, **Maple** offers the most flexible and feature rich implementation for a fixed point iteration, in the form of the built in command **FixedPointIteration**, included in the Student [NumericalAnalysis] subpackage. The command will numerically approximate the roots of a real function f, by converting the problem to a fixed point problem and then using the Picard fixed point iteration with the supplied initial approximation for the root. The command can be invoked using one of the following forms:

- FixedPointIteration($f, x = a$, opts)
- FixedPointIteration(f, a, opts)

The arguments have the following significance:

- f - is an expression in the variable x representing a continuous function
- x - specifies the independent variable of f
- a - is the initial approximation of the root
- opts - optional arguments in the form keyword=value

Some of the most important other options for the command are:

- fixedpointiterator = expression - specifies directly the expression to be used in the fixed point iteration; if this option is present, the first argument, f, must be omitted.
- tolerance = value - specifies the error tolerance of the approximation
- stoppingcriterion = value - specifies the criterion that the approximation must meet before stopping the iteration, and can have one of the following values:
  - relative - $\frac{|x_n - x_{n-1}|}{|x_n|} < tolerance$
  - absolute - $|x_n - x_{n-1}| < tolerance$
  - function_value - $|f(x_n)| < tolerance$
- maxiterations = value - specifies the maximum iterations to perform, if the error tolerance is not achieved
- output = value - specifies the type of the return value of the command, and it can have on of the following values:
  - value - returns just the final numerical approximation of the root
  - sequence - returns the sequence of intermediate approximations produced by the fixed point iteration
  - plot - returns a plot of f with each iterative approximation shown
  - animation - returns an animation showing the iterations of the root approximation process
  - information - returns detailed information about the iterative approximations of the root of f

There are many other options available, used to control the plotting functionality of the command. More details about the Maple FixedPointIteration command are available on the Maple Online Help web page (http://www.maplesoft.com/support/help/Maple/view.aspx?path= Student/ NumericalAnalysis/ FixedPointIteration). As an example we use the Maple commands to solve the problem $x^2 - x - 2 = 0$ and to produce a plot of the fixed point iteration which are displayed in figure 1.

In **Mathematica** there are available two built in functions related to fixed point iterations: **FixedPoint** and **FixedPointList**.

with Student[NumericalAnalysis] :

$$FixedPointIteration\left(\text{fixedpointiterator} = 1 + \frac{2}{x},\ \text{x} = 1.0,\ \text{maxiterations} = 16,\ \text{output} = \text{sequence}\right)$$

1.0, 3.000000000, 1.666666667, 2.200000000, 1.909090909, 2.047619048, 1.976744186, 2.011764706, 1.994152047, 2.002932551, 1.998535871, 2.000732601, 1.999633834, 2.000183117, 1.999908450, 2.000045777

$$FixedPointIteration\left(\text{fixedpointiterator} = 1 + \frac{2}{x},\ \text{x} = 1.0,\ \text{output} = \text{plot},\ \text{maxiterations} = 16,\ view = [0..5, -1..4]\right)$$

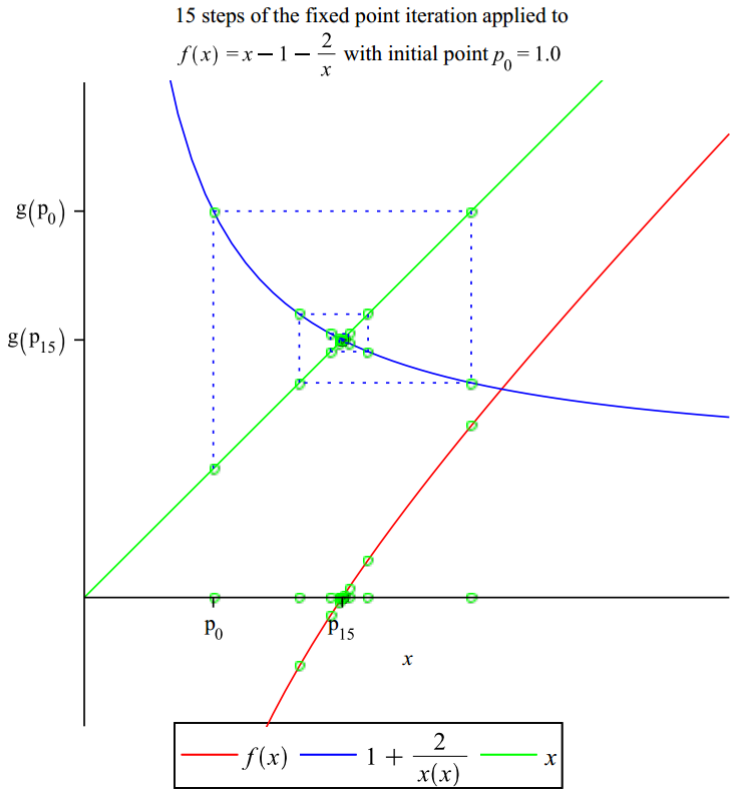

FIGURE 1. Fixed point iteration commands in Maple

Both functions use the Picard iteration in order to compute a fixed point and have the same parameters, with the difference that FixedPoint returns only the final value, while FixedPointList generates a list with all the intermediate values. The FixedPoint function can be called with using one of the following forms:

- FixedPoint[f, expr] - where expr is the initial approximation.
- FixedPoint[f, expr, n] - stops after at most n steps.
- FixedPoint[f, expr, ..., SameTest –> s] - a custom test, s, can be used to check if to consecutive results are equal

More details and examples are available online, on the Wolfram Mathematica Documentation website http://reference.wolfram.com/mathematica/ref/FixedPoint.html. In

Mathematica there is no built in functionality for plotting the fixed point iteration, like in Maple, but this can be achieved using few lines of code, as shown below.

```
g = Function[x, 1 + 2/x];
fpl = FixedPointList[g, 1.0];
fpCoords = Flatten[Table[
    {{fpl[[k]], fpl[[k]]}, {fpl[[k]], fpl[[k + 1]]}},
    {k, 10}],
  1];
iterationPlot = ListPlot[fpCoords,
  AxesOrigin -> {0, 0},
  PlotRange -> {{0, 4}, {0, 4}},
  Joined -> True, PlotStyle -> {Blue, Dashed}];
functionPlot = Plot[{x, g[x]}, {x, 0, 4}, PlotStyle -> {Thick}];
Show[iterationPlot, functionPlot]
```

LISTING 1. Fixed point iteration and plot in Mathematica

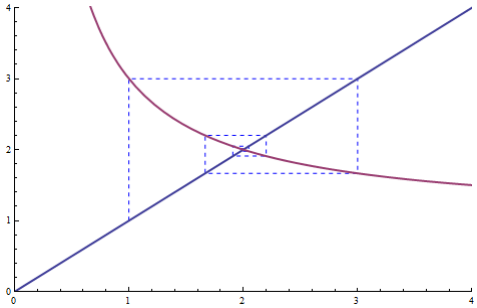

FIGURE 2. Mathematica plot: fixed point iteration applied to $f(x) = 1+\frac{2}{x}$ with initial point $x_0 = 1.0$

Matlab and MathCAD do not offer any built in functions related to fixed point iterations, but fixed point computation is possible using custom written functions, as demonstrated by the following Matlab listing.

```
function [p0,err,P] = fixpt(g, x0, tol, max1)
  x = x0; %initial guess
  P(0) = x0;
  xold = x;
  n = 1; %iteration counter
  while n < max1
    x = feval(g, x);
    P(n) = x;
    if abs(x-xold) < tol
      break;
  end
  xold = x;
  n = n + 1;
end
```

LISTING 2. Picard fixed point iteration and plot in Matlab

As alternatives to the CAS software there are several other basic implementations of the Picard fixed point iteration which can be used, from which we mention the following:

- EasyNumerics is a desktop application which, beside other numerical algorithms, has an implementation of the Picard fixed point iteration together with some basic plotting of the iteration. It is available for download from http://www.metu.edu.tr/ csert/EasyNumerics/.
- Fixed Point Iteration Java Applet is available online at http://www.csulb.edu/ wziemer/FixedPoint/FixedPoint.html and has another implementation of fixed point iteration plotting.
- other simple implementations of the Picard fixed point iterations are available online at:

- http://maccery.com/maths/#fixed-point
- http://planetcalc.com/2824/
- http://cs.laurentian.ca/badams/numeric/javascript/fpoint.htm

The above research indicates that all existing implementations for fixed point iterations have rather basic functionality, excepting Maple, which also has a good cobweb diagram potting and animation. Another interesting point is that even the implementations in well known CAS like Maple and Mathematica are lacking advanced features like oscillation detection (e.g. the case of logistic map, $\lambda x(1-x)$). Also, there are no available implementations for other fixed point iterative methods like Krasnoselskij, Mann and Ishikawa. Such implementations are possible in CAS using custom written code, but, beside expensive licenses, this requires at least some familiarity with the corresponding CAS programming language and functions.

3. Main features of FIXPOINT software minipackage

The FIXPOINT software package was developed considering the previously described limitations of existing implementations of fixed point iterations numerical methods. The software package evolved during several versions from a C++ to a C# implementation, and in the latest version to a the Javascript implementation, complemented by a web-based application, which has great advantages for users in the form of: easy accessibility and usability, seamless updates and even enhanced collaboration. Considering the latest advancements in the programming tools for Javascript language, this new implementation also offers the unique possibility to run the same implementation of the algorithms on the client machine, using a smart client application or on servers using for example nodejs (http://nodejs.org/). Also the Javascript language has a straightforward syntax, it doesn't require a compiler and the support for developers is very good in the form of documentation, samples and tools, so it is expected that the Javascrit FIXPOINT package can be used very easily by other persons for different applications.

The FIXPOINT software package contains implementations for several fixed point iterative methods for real valued functions: Picard, Krasnoselskij, Mann, Ishikawa and k-step fixed point iterative methods. In the latest version two more algorithms were added for functions in $\mathbb{R}^n$: a simplicial fixed point algorithm and a piecewise linear homotopy algorithm [16]. The implementation of the algorithms for real valued functions are similar, so here we will describe the details just for the Picard algorithm.

The new Javascript implementation of the Picard iteration computes an approximate fixed point of real valued functions $f : \mathbb{R} \to \mathbb{R}$. The implementation is in the form of a Javascript function, which accepts the following parameters:

- f - the function for which the approximate fixed point will be computed, specified as a Javascript function which must accept a single argument;
- $x0$ - the real number which specifies the initial guess used to start the algorithm;
- $options$ - a Javascript object which can contain additional optional arguments used to control the behaviour of the algorithm, as described below;
- $options.maxRelativeError$ - specifies the minimum desired precision for the approximate fixed point; the algorithm stops if the relative error between the approximations obtained in two consecutive steps is smaller than the specified value; the relative error is used due to the precision limitations of the IEEE 754 floating point format; the algorithm is using standard IEEE 754 double precision floating point numbers and of course, the $maxRelativeError$ should be greater than the machine epsilon (or unit roundoff);
- $options.maxSteps$ - the maximum number of steps to be executed before stopping the algorithm, in the case that a fixed point with the desired approximation precision was not found;
- $options.checkCycles$ - by default, the implementation of the algorithm is also checking for cycles of the iteration (oscillations), but this increases the complexity of the algorithm to $O(n^2)$; if greater performance is required, this options can be used to disable this additional check.

For further details about the algorithm implementation, a simplified version of the code is presented in the following listing.

```
var fixpoint = function()
{
  function mixedErrorTest(maxError, x, xprev1, xprev2) {
    // compute an estimate of the absolute error
    var err = Math.abs(x - xprev1) + Math.abs(xprev1 - xprev2);
    // use a mixed error test for absolute and relative error
    return (err <= maxError * (1 + Math.abs(x)));
  }

  function iterate(f, x0, options) {
    // helper variables
    var values = [x0];
    var x = x0, xprev1 = x0, xprev2 = x0;
    var n = 1;
    var res = {};

    do {
      // perform an iteration step
      x = f(x, n);
      // store last result
      values.push(x);
      // perform the specified convergence test and stop if it is successful
      if (options.convergenceTest(options.maxError, x, xprev1, xprev2, values))
        break;
      // check max number of iterations
      if (n === options.maxSteps) {
        res.errorMessage = 'Maximum number of iterations was reached.';
        break;
      }
      // check for overflows and undefined operations
      if (!Number.isFinite(x)) {
        res.errorMessage = 'Iteration is divergent (numerical error).';
        break;
      }
      // check if the sequence already contains the current value
      if (options.checkCycles) {
        for (var i = 0; i < n; i++) {
          if (x === values[i]) {
            res.errorMessage = 'Iteration is divergent '
              + '(cycle detected between iterations #' + i.toString()
              + ' and #' + n.toString() + ').';
            break;
          }
        }
        if (res.errorMessage !== undefined)
          break;
      }
      n++;
      // save the values for the last two iterations
      // these will be used for the mixed error convergence test
      xprev2 = xprev1;
      xprev1 = x;
    }
    while (true);

    res.values = values;
    res.numSteps = n;
    res.x0 = x0;
    res.xn = x;
    return res;
  }

  return {
    picard : function(f, x0, options) {
      return iterate(f, x0, options);
    },
    mann : function(f, x0, alphan, options) {
      function m(x, n) {
        var an = alphan(n);
        return (1 - an) * x + an * f(x);
      }
      return iterate(m, x0, options);
    },
    errtest : { mixed : mixedErrorTest }
  };
}();
```

LISTING 3. FIXPOINT software minipackage implementation in Javascript

The functions defined in the software package will return the following values (wrapped in a Javascript object):

- xn - the approximate fixed point obtained by the iteration
- $values$ - an array containing all the intermediate approximations (including the initial approximation x_0)
- $numSteps$ - the number of iterations performed by the algorithm
- $error$ - it will contain a short description in case that the iteration is not successful (iteration is not convergent, the maximum number of iterations is reached or an undefined numerical operation is performed, e.g. $0/0$).

Complementary to the algorithms implementations, a simple web-based user interface is available at http://algonum.appspot.com/fixpoint.picard, which can be used very easily by persons without knowledge of Javascript programming for experimenting with the algorithm. The user interface allows the users to enter the input data for the algorithm in a very easy format. Some special attention was paid on editing the mathematical formulas, and as it is visible in figure 3, it is possible to edit complex formulas using a powerful editor.

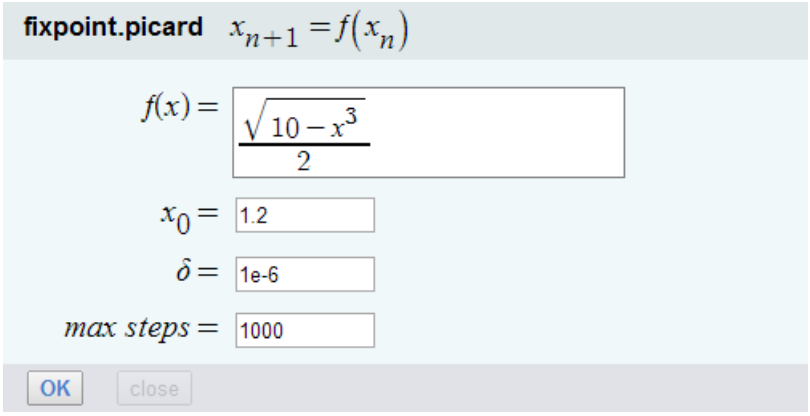

FIGURE 3. Web-based user interface for Picard iteration

The web-based application can be also used to obtain nice interactive visualisations of the function and the associated Picard iteration (as a cobweb plot), similar to the plots obtained in Maple.

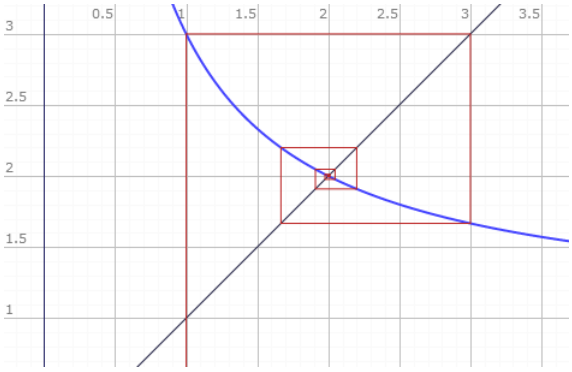

FIGURE 4. FIXPOINT software minipackage plot: fixed point iteration applied to $f(x) = 1 + \frac{2}{x}$ with initial point $x_0 = 1.0$

4. APPLICATIONS

The initial versions of the FIXPOINT package have been used by the second author in the study of convergence rate of various fixed point iterative algorithms. The main results reported on this topic were included in Chapter 7 of the monograph [6]. These were empirical studies of the rate of convergence of Picard, Mann and Ishikawa fixed point iteration procedures. We quote here some extracts from Example 9.4 from [9].

Example 4.1. Let $K = [0, 1]$ and $T : K \to K$ be given by $Tx = (1 - x)^6$.

Then T has $p_1 \approx 0.2219$ and $p_1 \approx 2.1347$ as fixed points. Both of them are repulsive fixed points with respect to the Picard iteration. However, p_1 is attractive with respect to Krasnoselskij, Mann and Ishikawa iterations, while p_2 stays repulsive, as indicated by the numerical results obtained by running the new version of the program FIXPOINT.

Krasnoselskij iteration: if we start from $x_0 = 2$ and the parameter that defines the iteration is $\lambda = 0.5$, then we obtain $x_1 = 1.5$, $x_2 = 0.757$, $x_3 = 0.379$, $x_4 = 0.2181$, $x_5 = 0.2232$ and $x_6 = 0.2214$;

Mann iteration: if we start from $x_0 = 2$ and the parameter sequence is $\alpha_n = 1/(n + 1)$, then we obtain $x_1 = 1.0$, $x_2 = 0.5$, $x_3 = 0.338$, $x_4 = 0.2748$, $x_5 = 0.2489$ and $x_6 = 0.2378$;

Ishikawa iteration: if we start from $x_0 = 2$ and the parameter sequences are $\alpha_n = 1/(n + 1)$ and $\beta_n = 1/(n + 2)$, respectively, then we obtain $x_1 = 0.01$, $x_2 = 0.55$, $x_3 = 0.346$, $x_4 = 0.2851$, $x_5 = 0.2527$ and $x_6 = 0.2392$;

These empirical results suggest that Krasnoselskij iteration converges faster than both Mann and Ishikawa iterations. This fact is more clearer illustrated if we choose $x_0 = p_2$, the repulsive fixed point of T: after 20 iterations, Krasnoselskij method gives $x_{20} = 0.2219$, while Mann and Ishikawa iteration procedures give $x_{20} = 0.6346$ and $x_{20} = 0.6347$, respectively. The convergence of Mann and Ishikawa iteration procedures is indeed very slow in this case: after 500 iterations we get $x_{500} = 0.222$ for both methods.

Note that for $x_0 \in \{-2, 3, 4\}$ and the previous values of the parameters λ, α_n and β_n, all three iteration procedures: Krasnoselskij, Mann and Ishikawa, converge to 1, which is not a fixed point of T.

Starting from such kind of numerical results, we tried to infer that, for certain classes of mappings, Picard iteration *always* converges faster than Mann or Ishikawa iterations. The first results of this kind have been reported in [7], which opened a fruitful direction of research that has been later considered by many other authors, see, for an incomplete list, [1], [2], [3], [4], [5], [8], [10], [20], [21]-[23], [24], [25], [26], [27], [28], [29], [36], [37], [39], [40].

5. CONCLUSIONS AND FUTURE WORK

All our empirical studies based on the use of the FIXPOINT package have been mainly devoted to continuous single-valued self mappings of the form $T : X \to X$. It is therefore our aim to continue the study by considering discontinuous mappings (which are governed by more general fixed point principles than the classical Banach contraction mapping principle) or single-valued self mappings defined on product spaces $T : X^k \to X$, in view of the recent papers [11], [12], [13], [14], [18], [30]-[35], where more sophisticated k-step fixed point algorithms implemented in recent versions of FIXPOINT have to be used.

Acknowledgements. The second author's research was supported by the Grants PN-II-RU-TE-2011-3-239 and PN-II-ID-PCE-2011-3-0087 of the Romanian Ministry of Education and Research.

REFERENCES

[1] Akbulut, S. and Zdemir, M., *Picard iteration converges faster than Noor iteration for a class of Quasi-contractive operators*, Chiang Mai J. Sci., **39** (2012), No. 4, 688–692

[2] Alotaibi, A., Kumar, V. and Hussain, N., *Convergence comparison and stability of Jungck-Kirk-type algorithms for common fixed point problems*, Fixed Point Theory Appl., **2013**, 2013:173, 30 pp

[3] Babu, G. V. R. and Vara Prasad, K. N. V. V., *Mann iteration converges faster than Ishikawa iteration for the class of Zamfirescu operators*, Fixed Point Theory Appl, **2006**, Art. ID 49615, 6 pp

[4] Babu, G. V. R. and Vara Prasad, K. N. V. V., *Erratum: "Mann iteration converges faster than Ishikawa iteration for the class of Zamfirescu operators" [Fixed Point Theory Appl. 2006, Art. ID 49615, 6 pp.; MR2210912]*, Fixed Point Theory Appl, **2007**, Art. ID 97986, 2 pp

[5] Babu, G. V. R. and Vara Prasad, K. N. V. V., *Comparison of fastness of the convergence among Krasnoselskij, Mann, and Ishikawa iterations in arbitrary real Banach spaces*, Fixed Point Theory Appl., **2006**, Art. ID 35704, 12 pp

[6] Berinde, V., *Iterative Approximation of Fixed Points*, Efemeride Publishuing House, Baia Mare, 2002

[7] Berinde, V., *Picard iteration converges faster than Mann iteration for a class of quasi-contractive operators*, Fixed Point Theory Appl, 2004, No. 2, 97–105

[8] Berinde, V. and Berinde, M., *The fastest Krasnoselskij iteration for approximating fixed points of strictly pseudo-contractive mappings*, Carpathian J. Math., **21** (2005), No. 1-2, 13–20

[9] Berinde, V., *Iterative Approximation of Fixed Points*, 2nd edition, Springer, Berlin Heidelberg New York, 2007

[10] Berinde, V. and Păcurar, M., *Empirical study of the rate of convergence of some fixed point iterative methods*, Proc. Appl. Math. Mech., **7** (2007), 20300152030016. doi: 10.1002/pamm.200700254

[11] Berinde, V. and Păcurar, M., *An iterative method for approximating fixed points of Prešić nonexpansive mappings*, Rev. d'Anal. Numer. Theor. Approx., **38** (2009), No. 2, 144–153

[12] Berinde, V. and Păcurar, M., *Two elementary applications of some Prešić type fixed point theorems.*, Creat. Math. Inform., **20** (2011), No. 1, 32–42

[13] Berinde, V. and Păcurar, M., *O metodă de tip punct fix pentru rezolvarea sistemelor ciclice*, Gazeta Matematică, Seria B, **116** (2011), No. 3, 113–123

[14] Berinde, V. and Păcurar, M., *Stability of k-step fixed point iterative methods for some Prešić type contractive mappings*, J. Ineq. Appl., 2014, 2014:149

[15] Bozantan, A., *Algorithms for approximating fixed points*, Dissertation Thesis, North University of Baia Mare, 2007

[16] Bozantan, A., *An implementation of the piecewise-linear homotopy algorithm for the computation of fixed points*, Creat. Math. Inform., **19** (2010), No. 2, 140–148

[17] Cegielski, A., *Iterative Methods for Fixed Point Problems in Hilbert Spaces*, Lecture Notes in Mathematics, **2057**, Springer-Verlag, Berlin, 2013

[18] Chen, Y.-Z., *A Prešić type contractive condition and its applications*, Nonlinear Anal., **71** (2009), No. 12, 2012–2017

[19] Chidume, C. E., *Geometric Properties of Banach Spaces and Nonlinear Iteration*, Springer, Berlin Heidelberg New York, 2009

[20] Duong V. T., *The comparison of the convergence speed between Picard, Mann, Ishikawa and two-step iterations in Banach spaces*, Acta Math. Vietnam, **37** (2012), No. 2, 243–249

[21] Hussain, N., Rafiq, A., Damjanović, B. and Lazović, R., *On rate of convergence of various iterative schemes.*, Fixed Point Theory Appl., **2011**, 2011:45, 6 pp

[22] Hussain, N., Chugh, R., Kumar, V. and Raffia, A., *On the rate of convergence of Kirk-type iterative schemes*, J. Appl. Math, **2012**, Art. ID 526503, 22 pp

[23] Hussain, N., Kumar, V. and Kutbi, M. A., *On rate of convergence of Jungck-type iterative schemes.*, Abstr. Appl. Anal., **2013**, Art. ID 132626, 15 pp

[24] Kang, S. M., Ćirić, L. B., Rafiq, A., Ali, F., and Kwun, Y. C., *Faster Multistep Iterations for the Approximation of Fixed Points Applied to Zamfirescu Operators*, Abstr. Appl. Anal., Vol. **2013**

[25] Karahan, I. and Murat, O., *A general iterative method for approximation of fixed points and their applications*, Adv. Fixed Point Theory, **3** (2013), No. 3, 510–526

[26] Khan, A. R., Kumar, V. and Hussain, N., *Analytical and numerical treatment of Jungck-type iterative schemes*, Appl. Math. Comput., **231** (2014), 521535

[27] Kumar, V., *Comments on convergence rates of Mann and Ishikawa iterative schemes for generalized contractive operators*, Int. J. Math. Anal. (Ruse), **7** (2013), No. 25-28, 1317–1321

[28] Olaleru, J. O., *A comparison of Picard and Mann iterations for quasi-contraction maps*, Fixed Point Theory, **8** (2007), No. 1, 87–95

[29] Olaleru, J. O., *On the convergence rates of Picard, Mann and Ishikawa iterations of generalized contractive operators*, Stud. Univ. Babeş-Bolyai Math, **54** (2009), No. 4, 103–114

[30] Păcurar, M., *Approximating common fixed points of Prešić-Kannan type operators by a multi-step iterative method*, An. Ştiinţ,. Univ. "Ovidius" Constanţa Ser. Mat., **17** (2009), No. 1, 153–168

[31] Păcurar, M., *Iterative Methods for Fixed Point Approximation*, Risoprint, Cluj-Napoca, 2010

[32] Păcurar, M., *A multi-step iterative method for approximating fixed points of Prešić-Kannan operators*, Acta Math. Univ. Comen. New Ser., **79** (2010), No. 1, 77-88

[33] Păcurar, M., *A multi-step iterative method for approximating common fixed points of Prešić-Rus type operators on metric spaces*, Stud. Univ. Babeş-Bolyai Math., **55** (2010), No. 1, 149–162
[34] Păcurar, M., *Fixed points of almost Prešić operators by a k-step iterative method*, An. Ştiint,. Univ. Al. I. Cuza Iaşi, Ser. Noua, Mat., **57** (2011), Supliment 199–210
[35] Păcurar, M., *Common fixed points for almost Presić type operators*, Carpathian J. Math., **28** (2012), No. 1, 117–126
[36] Phuengrattana, W. and Suantai, S., *Strong convergence theorems and rate of convergence of multi-step iterative methods for continuous mappings on an arbitrary interval*, Fixed Point Theory Appl., 2012, 2012:9
[37] Popescu, O., *Picard iteration converges faster than Mann iteration for a class of quasi-contractive operators*, Math. Commun., **12** (2007), No. 2, 195–202
[38] Rhoades, B. E. and Xue, Z., *Comparison of the rate of convergence among Picard, Mann, Ishikawa, and Noor iterations applied to quasicontractive maps*, Fixed Point Theory Appl, **2010**, Art. ID 169062, 12 pp
[39] Xue, Z., *The comparison of the convergence speed between Picard, Mann, Krasnoselskij and Ishikawa iterations in Banach spaces*, Fixed Point Theory Appl, **2008**, Art. ID 387056, 5 pp
[40] Xue, Z. and Rhoades, B. E., *Comparison of the rate of convergence among Picard, Mann, Ishikawa, and Noor iterations applied to quasicontractive maps*, Fixed Point Theory Appl., 2010, Art. No. 169062

Department of Mathematics and Computer Science
North University Center at Baia Mare
Technical University of Cluj-Napoca
Victoriei 76, 430122, Baia Mare, Romania
E-mail address: andrei.bozantan@gmail.com
E-mail address: vberinde@ubm.ro