The Parallel Implementation of Simultaneous Methods for Finding the Polynomial Zeros

Eglantina Kalluci[1*], Fatmir Hoxha[1], Brikena Preni[2]

[1] Faculty of Natural Sciences, Department of Applied Mathematics, University of Tirana, Albania.
[2] Faculty of Mathematical and Physical Engineering, Polytechnic University of Tirana, Albania.

* Corresponding author. Tel.: 00355682031231; email: eglantina.kalluci@fshn.edu.al, fatmir.hoxha@fshn.edu.al
Manuscript submitted August 12, 2018; accepted January 2, 2019.
doi: 10.17706/ijapm.2019.9.2.111-118

Abstract: In this paper we represent a parallel implementation of three simultaneous methods for finding the roots of polynomials. We have chosen two well-known simultaneous methods, Durand-Kerner and Ehrich-Aberth and a third new improvement in an asynchronous cluster with 9 processors. We have analysed the time of execution (= time of communication + time of computation) of these algorithms on polynomials with different powers (up to 200) and using different number of processors. For each of these methods we have we have calculated the speed-up. The numerical tests show the global convergence of these methods and through these tests we come into the conclusions when the parallelization is more effective. Some numerical tests are related to the case of using Estrin's scheme in the evaluation of the polynomials, which gives interesting results in the cases when the polynomial degree is a power of 2.

Key words: Speed-up, simultaneous methods, polynomial, parallel implementation.

1. Introduction

The problem of finding the roots of polynomials is encountered in different applications. Most of the numerical methods that deal with this problem are simultaneous ones. These methods start from the initial approximations of all the roots of the polynomial and give a sequence of approximations that converge to the roots of the polynomial. The main problem of the simultaneous methods is that the necessary time needed for the convergence is increased with the increasing of the degree of the polynomial. These methods are suitable to implement in a cluster and their parallelization considerably improves the time of convergence. In this paper we with treat the parallelization of Durand-Kerner method [1], [2] and Ehrlich-Aberth method [3], [4] comparing with the sequential case. These methods in practice have good global convergence. Other simultaneous methods as Laguerre, Euler-like and Halley-like methods can be implemented in parallel platforms. From the computational analyses in polynomial root-finding indicates that among all the mentioned methods, Ehrlich-Aberth method has the advantages of requiring a small number of iterations for convergence and having a small number of function evaluations per iteration. This is the main reason we have choosing this method to be compared with the modification of Durand-Kerner iterations.

2. Simultaneous Methods for Approximating Polynomial Zeros

Until 1960 all known methods for finding the roots of polynomials are based in the Deflation method, so in finding the roots of polynomials one by one. This method lead to the increasing of rounding errors. A group of methods that avoids this problem are the simultaneous ones, which in every iteration approximate all the roots simultaneously. These methods have many advantages even on a serial computer, but they are especially suited to be implemented in parallel computers. Several authors have considered this situation, McNamee, J. M., (2007) [5] has listed in chronological order most of them.

The first method of this group is the following:

$$\hat{z}_i = z_i - \frac{P(z_i)}{\prod_{i\neq j}(z_i - z_j)}. \tag{1}$$

This formula is mentioned for the first time from Weiestrass (1903) as part of the fundamental theorem of Algebra and is rediscovered from Ilieff (1950), Durand (1960), Docev (1962), Kerner (1966) and others. This is known as Durand-Kerner (DK) metod or Weiestrass-Durand-Kerner method. From a straightforward evaluation of (1), where P(z) is evaluated using the Horner's scheme, requires about $2n^2$ complex multiplications and $2n^2$ additions and subtractions. While Werner (1982) describes a more efficient scheme that requires only $\frac{1}{2}n^2$ multiplications, $\frac{1}{2}n^2$ divisions and $2n^2$ additions and subtractions.

Another method discovered from Borsch-Supan (1963) and also described and brought in the following form Ehrlich (1967) and Aberth (1973)

$$\hat{z}_i = z_i - \frac{1}{\frac{P'(z_i)}{P(z_i)} - \sum_{i\neq j}(z_i - z_j)}. \tag{2}$$

Aberth, Ehrlich and Farmer-Loizou (1983) have proved that the above method has cubic order of convergence for simple roots.

Besides the advantages of lending itself to parallel computations, the Weiestrass method is much more robust than e. g. Newton's method, i. e. it nearly always converges, no matter what the initial guess(es).

Cosnard and Fraignaud (1990) [6] compare three different parallel network topologies (ring, 2-D torus, and hypercube). They conclude that the hypercube is by far the fastest. In experiment they obtain almost perfect speed-up.

Bini (1996) [7] and Bini and Fiorentino (2000) [8] have written a highly efficient and robust program, based on Aberth's method, with cluster analysis to speed convergence for multiple roots, and adaptive multiprecision arithmetic. It never failed on 1000 polynomials of degree up to 25.

The Gauss-Seidel method in Linear Algebra consists in using the same iterations the already calculated values. Many authors have borrowed the same idea in the simultaneous methods for finding the roots of polynomials. For example, Niell (2010) [2] gives a modification of the following modification for the DK method:

$$\hat{z}_i = z_i^{(k)} - \frac{P(z_i^{(k)})}{\prod_{j=1}^{i-1}\left(z_i^{(k)} - z_j^{(k+1)}\right)\prod_{j=i+1}^{n}\left(z_i^{(k)} - z_j^{(k)}\right)}. \tag{3}$$

He has proved that the order of convergence is $2 < r < 3$, for polynomials with degree up to15.

In the paper Xhaja, Hoxha (2011) [3] we have proposed another modification. Let,

$$N_i = \frac{P(z_i)}{P'(z_i)}, \quad H_i = \frac{P(z_i)}{P'(z_i) - \frac{P(z_i)P''(z_i)}{2P'(z_i)}}$$

be Newton's and Halley's corrections appearing in the well-known iterative formulas $\hat{z}_i = z_i - N_i$ (Newton's method), $\hat{z}_i = z_i - H_i$ (Halley's method) of the second and third order, respectively.

Let be $z_1, z_2, \cdots, z_n$ the approximations to the zeros $\lambda_1, \cdots, \lambda_n$, of a monic polynomial of order *n*. Using the improved approximations $c_j = z_j - N_j$ or $c_j = z_j - H_j$ is defined the modified Weierstrass function,

$$\tilde{W}_i(z) = \frac{P(z)}{\prod_{\substack{j=1 \\ j \neq i}}^{n} (z - c_j)}.$$

This modification follows an idea borrowed from numerical linear algebra, where it leads from Jacobi's method to Gauss-Seidel's. The idea is to use at every moment the latest computed components of the approximate solution vector in order to compute the next component, rather than using the "old" approximate solution vector to compute the entire "new" vector. The Gauss-Seidel approach or serial mode is applied in different methods to accelerate the convergence speed. Using the modified Weierstrass function,

$$\vec{W}_i\left(z_i^{(k)}\right) = \frac{P\left(z_i^{(k)}\right)}{\prod_{j=1}^{i-1}\left(z_i^{(k)} - c_j^{(k+1)}\right)\prod_{j=i+1}^{n}\left(z_i^{(k)} - c_j^{(k)}\right)}, \tag{4}$$

and substituting this function to Newton method, achieving the following simultaneous method

$$\hat{z}_i = z_i - \frac{\vec{W}(z_i)}{\vec{W}_i'(z_i)}, \tag{5}$$

which has fifth order of convergence when we use the Newton correction and sixth order of convergence when we use Halley correction.

All applications are based on the fact that the rational function W (or $\tilde{W}$, $\vec{W}$) has the same zeros as the polynomial *P*. We emphasize that the use of corrections is justified only when its evaluations can be performed by the already calculated quantities. In this way the order of convergence is increased using negligible number of numerical operations giving a higher computation efficiency of the stated method.

3. The Implementation in a Cluster of the Simultaneous Methods

The simultaneous methods have a lot of advantages (for example very good convergence) even on sequential computers, but they are suitable to be implemented in parallel computers.

Many authors have treated the problem of implementation of simultaneous methods in a cluster. Freeman (1989) has tested the DK method, EA method and another method of the fourth order proposed from Farmer and Loizou (1983), on a 8- processor linear chain, for polynomial of degree up to 8. The third method often diverges, but the first two methods have speed-up 5.5 (speed-up = (Time on one processor)/(Time on p processors)). Later Freeman and Bane (1090) considered asynchronous algorithms, in which each processor continues to update its approximations even although the latest values of other $z_i^{(k)}$ have not received from the other processors, in difference with the synchronous version where it would wait.

In this paper we have considered the asynchronous version for DK method, EA method and the improvement given in (5).

The general formula for the simultaneous methods has the form:

$$z_i^{(k+1)} = z_i^{(k)} - \frac{P\left(z_i^{(k)}\right)}{\Phi_i\left(z_1^{(k)},z_2^{(k)},\cdots,z_n^{(k)}\right)} (i = 1,\cdots,n). \qquad (6)$$

The parallel version of (6) for p-processors in which the l-th processor calculates j_l approximations and $i_l = \sum_{m=1}^{l-1} j_m\ (l = 1,\cdots,p)(i_1 = 0)$:

Step 1: (i) k=1

(ii) Define the initial approximations $z_i^{(1)}$.

Step 2: In parallel, for$l = 1, 2, \dots, p$ and for$i = i_l + 1, i_l + 2, \dots, i_l + j_l$.

1) Calculate$p_i^{(k)} = P(z_i^{(k)})$.

2) Calculate$q_i^{(k)} = \Phi_i\left(z_1^{(k)}, z_2^{(k)}, \dots, z_n^{(k)}\right)$.

3) Set$z_i^{(k+1)} = z_i^{(k)} - \frac{p_i^{(k)}}{q_i^{(k)}}$.

Step 3: For$i = 1, 2, \dots, n$ communicate $z_i^{(k+1)}$ to all the processors.

Step 4: (i) Check for the convergence.

(ii) Perform a new iteration $k = k + 1$.

Go to Step 2.

Finding the appropriate initial approximations is very important for the efficiency of the algorithm, leading to a small number of iterations needed for the convergence. The point (ii) of Step 1 can be defined from different methods. For finding the initial approximation we have used Aberth method [1], which is suitable because it does not put conditions in the way how the roots should be spread, and also in their nature, real or complex.

We calculate the points (i) and (ii) of Step 2 using the classical method and not Horner's scheme, which is known as the fastest method for evaluating the polynomial in sequential computers. Another method which is designed for parallel environments is Estrin's scheme [4]. The condition that this method gives is that the degree of the polynomial should be a power of two. In this paper we have tested it in the cases of polynomials of power 8, 16, 32, 64 and 128.

Table 2.1. The Total Time of the Algorithm for Polynomials of Different Degrees (from 8 to 128) Using Estrin's Scheme

proc/n	**8**	**16**	**32**	**64**	**128**
1	0.000932	0.007696	0.110636	0.572907	0.950551
2	0.012506	0.029038	0.259891	0.550832	0.827741
5	0.023143	0.066454	0.267016	0.523447	0.696566
9	0.058312	0.113119	0.296654	0.56681	0.756231

We have implemented all the algorithms mentioned in the SEEUCluster, an asynchronous cluster, which has a node that makes the connection with the environment outside the cluster. He is composed of 9 processors PC computer 1.5 Ghz/128MB/20GB, a switch 16 ports and additional supply for the network. The cluster is built in the operating systems Linux Red Hat Enterprise 4.0, the clustering configuration is in software OSCAR 4.2 (Open Source Cluster Application Resources). The parallel environment for programming, which will be used is LAM 7.0.6/MPI 2 C++/ROMIO-Indiana University.

We have implemented the parallel version of DK algorithm in polynomials of different orders and we have evaluated the time of communication between the processors, the time of calculation of each processor and the total time (*the total time= the communication time + the calculation time*). For comparison we have implemented also EA algorithm.

From the tests with fixed tolerance 0.00001 we achieve the results shown in the following tables:

Table 2.2. The Communication Time between the Processors for Polynomials of Different Degrees (from 9 to 200)

proc/n	9	50	80	100	120	150	180	200
2	0.011976	0.084491	0.179627	0.236206	0.309865	0.454037	0.635534	0.809774
5	0.022858	0.142724	0.288124	0.384883	0.466556	0.658251	0.872588	1.03647
9	0.057796	0.171517	0.348915	0.482871	0.616082	0.92352	1.26034	1.54332

Table 2.3. The Calculation Time for Polynomials of Different Degrees (from 9 to 200)

proc/n	9	50	80	100	120	150	180	200
1	0.000932	0.07696	0.310636	0.57907	0.950551	1.90795	3.19035	4.30292
2	0.00053	0.44547	0.180264	0.314626	0.517876	1.02191	1.73541	2.40965
5	0.000285	0.02373	0.078892	0.138564	0.23001	0.441535	0.722316	0.973868
9	0.000515	0.011602	0.047739	0.08681	0.140149	0.267236	0.516066	0.680262

Table 2.4. The Total Time of the Algorithm for Polynomials of Different Degrees (from 9 to 200)

proc/n	9	50	80	100	120	150	180	200
1	0.000932	0.07696	0.310636	0.572907	0.950551	1.90795	3.19035	4.30292
2	0.012506	0.129038	0.359891	0.550832	0.827741	1.475947	2.370944	3.219424
5	0.023143	0.166454	0.367016	0.523447	0.696566	1.099786	1.594904	2.010338
9	0.058312	0.183119	0.396654	0.56681	0.756231	1.190756	1.776406	2.223582

If we analyze the values in the tables, we notice that for a fixed degree of the polynomial if we increase the number of processors, the communication time is increased, because each processor will communicate with other processors in the cluster. The same situation is encountered even if we move in the other direction, so if we fix the number of processors and increase the degree of the polynomial. We can emphasize that if we compare the rows of Table 2.2 and of Table 2.3, so with the increasing of the degree of the polynomial and for a fixed number of processors, we notice that the calculation time is smaller than the communication one. If we move vertically in Table 2.3 we notice that with the increasing of the number of processors, the calculation time will be decreased. In Table 2.4 is presented the total time which is needed from the algorithm to give the desired result and we notice that with the increasing of the degree of the polynomial, this time is quite doubled for a given number of processors. If we move vertically in this table for a fixed degree of the polynomial, we notice that we do not gain much in time, because arrives a moment when adding more processors does not bring much difference in the number of evaluation performed, but increases the possibility of passing in stand-by mode and increases the communication through processors.

Finally for both DK and EA algorithms we have compared the sequential and parallel implementations and is measured the speed-up for polynomials with different degrees. This is useful for finding the degree of polynomial for which we will achieve a good speed-up, and for which the speed-up remains quite constant.

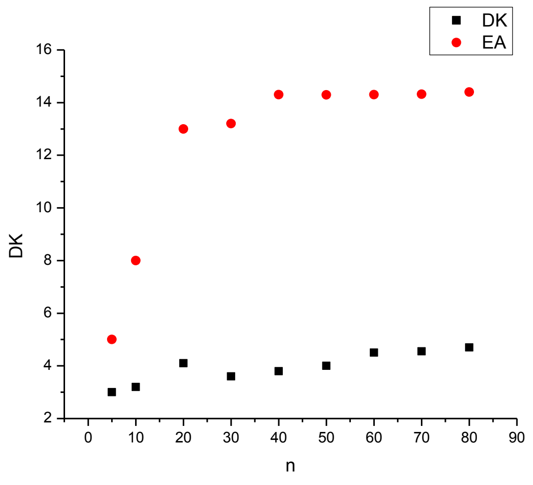

Fig. 1. The speed-up of DK and EA method for polynomial for different degrees.

As seen from Fig. 1 the speed-up of DK method converges at 4.5, while the speed-up of EA method converges at 14.5.

In Fig. 2 is added the speed-up of the modified method (5) DKN, but without the Gauss-Seidel approach and the numerical tests agree with the theoretical ones, because EA method and DKN method have the same order of convergence.

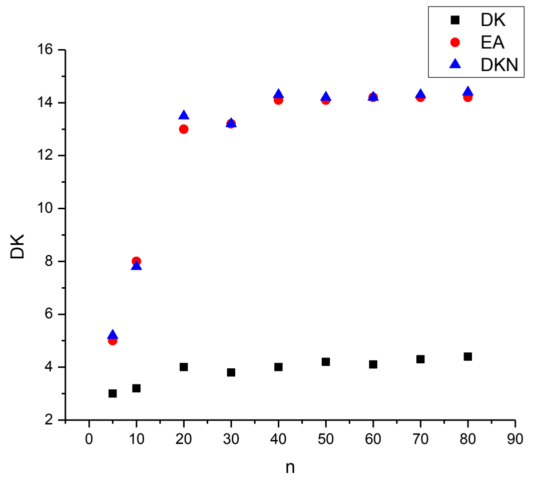

Fig. 2. The speed-up of DK, EA and DKN method for polynomials of different degrees.

If we have less than 6 processors then it would be better to run the program in a number of processors that is a submultiples of 6 (the degree of the polynomial) for using at the maximum all the processors. In this case should be better to run the program in 2 or 3 processors.

4. Conclusions

The simultaneous methods calculate all the roots of polynomials simultaneously and this is an advantage, which leads to the parallelization of these methods. In this paper we considered the parallelization of DK, EA and DKN method in a cluster with 9 processors. From the numerical tests we see that the increasing of the number of processors should be done till the communication time is considerable with the total time of the execution of the algorithm.

It is shown clearly in the scatter plot given in Fig. 1 that the parallelization is efficient for polynomials with large degrees (≥ 20). From the Fig. 2 we prove numerically that Erlich-Aberth method and Durand-Kerner method with Newton corrections have the same order of convergence. In all the polynomial evaluations we haven't use the Horner's role, because it will decrease the speed-up, for its sequential nature. In the case when the degree of polynomials is a power of two can be used the Estrin's method for the evaluation of polynomials. All these aspects goes to the discussion of reducing the operations per iteration.

References

[1] Aberth, O. (1973). Iterative methods for finding all zeros of a polynomial simultaneously. *Math. Comp.*, *27*, 339-344.

[2] Niell, A. M. (2001). The simultaneous approximation of polynomial roots. *Comp. Math. Appl.*, *41*, 1-14.

[3] Xhaja, E., & Hoxha, F. (2011). Accelerating simultaneous methods for the determination of polynomial multiple roots. *AKTET, IV(3)*, 418-422.

[4] Reynolds, S. G., (2010). Investigation of different methods of fast polynomial evaluation.

[5] McNamee, J. M. (2007). Numerical methods for roots of polynomials. *Studies in Computational Mathematics*, *14*. Elsevier.

[6] Cosnard, M., & Fraignaud, P. (1990), Finding the roots of a polynomial on an MIMD multicomputer. *Parallel Computing*, *15*, 75-85.

[7] Bini, D. A. (1996), Numerical computation of polynomial zeros by means of Aberth's method. *Numer. Algorithms*, *13*, 179-200.

[8] Bini, D. A., & Fiorentino, G. (2000), Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numer. Algorithms*, *23*, 127-173.

Eglantina Kalluçi was born on September 17, 1979, in Tirana. She was a student in the 5-years branch of Mathematics, Faculty of Natural Sciences, University of Tirana from 1998 to 2003. She got the master degree in numerical analysis with thesis "A new improvement of Durand-Kerner method in the determination of multiple roots of the polynomial", Faculty of Natural Sciences, University of Tirana, Albania; the PhD degree in mathematics (numerical analysis) with theme "Numerical solution of polynomial equations even in the case of multiple roots and the implementation in a cluster of processors", Faculty of Natural Sciences, University of Tirana. She was a professor at the Department of Mathematics, Faculty of Natural Sciences, University of Tirana from October 2003 to June 2016. From June 2016 to continuing, she is vice dean of the Faculty of Natural Sciences, University of Tirana.

Her research interest includes numerical methods for finding polynomial zeros; iterative, simultaneous and matrix ones; parallel implementations of PDE; monte-carlo simulations in complex networks.

Fatmir Hoxha was born on November 12, 1951, in Tirana, Albania. He is professor, head of Applied Mathematics Department, Faculty of Natural Sciences, University of Tirana.

He was a Student in the 5-years branch of Mathematics, Faculty of Natural Sciences, University of Tirana from 1969 to 1974. He graduated with excellence in mathematics from the University of Tirana, after one year training period in industry, obligatory for all students at that time in May 1975. He was a postgraduate studies in France from March 1984 to October 1988. He got the Master Degree from Institut National Polytechnique de Toulouse (INPT) France; the Ph.D.

degree from INPT, France (Advisor Prof. J. Noailles). He was a Professor of Mathematics at the University of Tirana from December 1994 to May 2016. From May 2016 to now, he is head of Applied Mathematics Department, Faculty of Natural Sciences, University of Tirana.

Brikena Preni was born on February 11, 1983. She was a student in the 5-years branch of Mathematics, Faculty of Natural Sciences, University of Tirana from 2000-2005. She got the master degree in probability and statistics with thesis "The Logit and Probit models, applications in insurance", Faculty of Natural sciences, University of Tirana; the PhD degree in mathematics (numerical analysis) with theme "Wavelet analysis, application in signal processing", Faculty of Natural Sciences, University of Tirana. From February 2009 to now, she is in Polytechnic University of Tirana, Faculty of Mathematical Engineering and Physics Engineering. Her research interest includes numerical methods in signal and image processing theory; Forecast in insurance models.