FILOMAT (Niš) 9:2 (1995), 261–271
Filomat '94, Niš, October 22-24, 1994
Geometry. Computer Sciences

HALLEY-LIKE ASYNCHRONOUS METHODS FOR POLYNOMIAL ROOTS

M. Trajković, S. Tričković and M. Petković

ABSTRACT. *In this paper we present the asynchronous implementation of Halley-like method for the simultaneous approximation of polynomial roots on a distributed memory multicomputer. It is shown that the lower bound of the order of convergence of asynchronous Halley-like method with the delay r is at least $\eta_A > 3$, where η_A is the unique positive root of the equation $\eta^{r+1} - 3\eta^r - 1 = 0$. The computational efficiency of the synchronous and asynchronous versions are studied in the case of hypercube topology.*

1. Some preliminary results

Simultaneous methods for the determination of polynomial roots run in several identical versions so that they are very convenient for the implementation on parallel computers (see, e.g., [4,5,6,8,9,10]). All n roots are found simultaneously, n versions of the same algorithm can be run on a distributed memory multicomputer consisting of k ($\leq n$) processors. The main advantage of parallel implementation comes from the fact that a great deal of computation can be performed simultaneously. The details concerning an application of simultaneous methods on parallel computers may be found in [4,5,6,7].

In practical implementation of simultaneous methods on parallel computers three standard network topologies are usually applied: rings, torus and hypercubes. The models assume k processors connected through a regular graph of diameter D and degree d. The efficiency of these methods depends on three parameters: the computation time of any arithmetical operation modeled by τ_a, the communication start up β_c and the throughput of the

1991 *Mathematics Subject Classification.* 65H05, 65W05.

links τ_c. Typical values of τ_c, τ_a and β_c for several types of multiprocessors can be found in the paper [3]. Since in our analysis we neglect the times for computing the starting points and checking the stopping criteria, the computational cost of algorithms is the sum of a computation time with a communication time. Besides, the number of basic arithmetical operations of the applied method appears as an additional parameter in the analysis of the total computational cost. This number for a wide class of simultaneous iteration methods can be given in the form (see [12, Ch. 6])

$$N(n) = \alpha n^2 + \beta n + \gamma,$$

where n is the polynomial degree. If n is sufficiently large, then we can take approximately that $N(n) = \alpha n^2 + o(n^2)$.

Following [6], the total time for the synchronous implementation can be expressed as the sum of the computation time $N(n)\tau_a/k$ and the communication time $D\beta_c + O(n/d)\tau_c$, that is

$$T_{\text{syn}} = \Big(\frac{\alpha n^2 + o(n^2)}{k}\Big)\tau_a + D\beta_c + O\big(\frac{n}{d}\big)\tau_c. \tag{1}$$

The communication time cannot be neglected in a synchronous implementation; moreover, it has a great influence on the total execution time and appears to be a major drawback of this parallelization of the simultaneous methods. In order to decrease the communicate time the following strategy can be applied [2,7,10]: In each iteration, a processor does not have to wait at predetermined points, for example, the end of the total-exchange, for predetermined messages to become available. This type of algorithms is called **asynchronous** by Baudet [1] indicating that, at each step, the local computation is performed using only a part of the global information.

Let $m = 0, 1, 2, \ldots$ be the iteration index and let us assume that the new approximation $z_i^{(m+1)}$ is calculated by a processor P_h, $h \in \{1, \ldots, k\}$. Evidently, to force the convergence, this processor *must know* the value of $z_i^{(m)}$. The improved approximation $z_i^{(m+1)}$ is calculated by a general iteration formula

$$z_i^{(m+1)} = F_i(\mathbf{z}^{(m^*)}), \quad \text{where} \quad \mathbf{z}^{(m^*)} = (z_1^{(m-r(1,m,h))}, \ldots, z_n^{(m-r(n,m,h))}). \tag{2}$$

In (2) $\mathbf{z}^{(m^*)}$ is the vector of the last values z_j known by the processor P_h at step m, represented by $z_j^{(m-r(j,m,h))}$. Here $r(j,m,h)$ is a **delay** depending on j, m and h and indicating that the processor P_h only knows the value of z_j computed at step $m - r(j,m,h)$. The maximum delay will be denoted by r, that is, $r = \max_{j,m,h} r(j,m,h)$.

The implementation of an asynchronous method is executed in such a way that, at each iteration step, a processor sends the most recently computed entries to its neighbors only, decreasing the communication time. As it was presented in [6], the total time per one iteration step is

$$T_{\text{asy}} = \Big(\frac{\alpha n^2 + o(n^2)}{k}\Big)\tau_a + \beta_c + O\big(\frac{n}{k}\big)\tau_c. \tag{3}$$

Comparing with (1), we obtain **one** start up instead of D and a propagation time of $O(n/k)$ instead of $O(n/d)$.

Let N_{syn} and N_{asy} be respectively the number of iteration steps of a synchronous and an asynchronous method. Evidently, the asynchronous method will be more efficient if $N_{\text{asy}}T_{\text{asy}} < N_{\text{syn}}T_{\text{syn}}$. By virtue of (1) and (3) this inequality may be written as follows:

$$N_{\text{asy}}\Big[\Big(\frac{\alpha n^2 + o(n^2)}{k}\Big)\tau_a + \beta_c + O\big(\frac{n}{k}\big)\tau_c\Big] < N_{\text{syn}}\Big[\Big(\frac{\alpha n^2 + o(n^2)}{k}\Big)\tau_a + D\beta_c + O\big(\frac{n}{d}\big)\tau_c\Big]. \tag{4}$$

Let us suppose that the inequality

$$\frac{\beta_c}{\tau_a} < \frac{\alpha n^2}{k} \tag{5}$$

holds. Namely, if $\beta_c/\tau_a > \alpha n^2/k$, then it could be faster to use less processors in the synchronous implementation (see [6]). Furthermore, since the relation $\beta_c \gg \tau_c$ generally occurs ([3]) on distributed memory computers, the inequality (4) becomes

$$\frac{N_{\text{asy}}}{N_{\text{syn}}} < \frac{\dfrac{\alpha n^2}{k} + D\dfrac{\beta_c}{\tau_a}}{\dfrac{\alpha n^2}{k} + \dfrac{\beta_c}{\tau_a}}. \tag{6}$$

Eventual validity of the inequality (6) can be suitable verified by a graphical interpretation in the plane $\big(N_{\text{asy}}/N_{\text{syn}}, \beta_c/\tau_a\big)$. Let $R = \beta_c/\tau_a$ denote a realistic parametric ratio depending on the applied network topology. For the hypercube topology this ratio usually belongs to the interval $[10^2, 10^3]$ (see [3]). Conditions for the dominance of asynchronous implementation has been discussed in [13]. Dominant area is bounded above by the curve

$$\frac{\beta_c}{\tau_a} = \frac{\dfrac{\alpha n^2}{k}\Big(1 - \dfrac{N_{\text{asy}}}{N_{\text{syn}}}\Big)}{\dfrac{N_{\text{asy}}}{N_{\text{syn}}} - D}, \tag{7}$$

which are obtained from (6) taking the sign "=" instead of "<", and the dashing line $\beta_c/\tau_a = R$.

Let V is the **critical ratio** given by the abscissa of the intersection of the curve (7) and the dashing line $\beta_c/\tau_a = R$, that is, $V = 1+(D-1)/(\frac{\alpha n^2}{kR}+1)$. As pointed out in [13], an asynchronous algorithm will be more efficient if the realistic ratio R is smaller than the bound value $\beta_c/\tau_a = \alpha n^2/k$ and, in addition, if the ratio of iteration steps $N_{\text{asy}}/N_{\text{syn}}$ can be realized in practice, that is, if $N_{\text{asy}}/N_{\text{syn}} < V$. We note that the following estimate for the ratio $N_{\text{asy}}/N_{\text{syn}}$ has been derived in [13]:

$$\frac{N_{\text{asy}}}{N_{\text{syn}}} \cong \frac{\log \eta_S}{\log \eta_A}. \tag{8}$$

2. Convergence analysis of asynchronous Halley-like method

Let $\epsilon_m^{(m)} = z_i^{(m)} - \zeta_i$ be the error of an asynchronous method of the form (2) which generates the sequences $(z_i^{(m)})$ of approximations to the roots $\zeta_1, \ldots, \zeta_n$. As mentioned in [13], for a wide class of iteration methods for the simultaneous determination of polynomial roots the following relation can be derived:

$$\epsilon_i^{(m+1)} = \alpha_i \left(\epsilon_i^{(m)}\right)^q \sum_{\substack{j=1 \\ j\neq i}}^{n} \beta_{ij} \epsilon_j^{(m-r(j,m,h))} \quad (i = 1, ..., n), \tag{9}$$

where α_i and β_{ij} are complex constants and $q \geq 1$ is integer. In that case following general convergence theorem has been proved in [13]:

Theorem 1. *Suppose that a polynomial P has only simple roots $\zeta_1, \ldots, \zeta_n$ and starting approximations $z_1^{(0)}, \ldots, z_n^{(0)}$ are reasonably close to these roots. Further, assume that $r(j,m,h)$ is bounded for all $j = 1,\ldots,n$ and all $h = 1,\ldots,k$. Then the asynchronous algorithm (2) for which the relations (9) are valid is locally convergent with the order of convergence at least $\eta_A(q) > q$, where $\eta_A(q)$ is the unique positive root of the equation*

$$\eta^{r+1} - q\eta^r - 1 = 0, \quad r = \max_{j,m,h} r(j,m,h). \tag{10}$$

In this section we give a convergence analysis of the asynchronous Halley-like root finding method and its efficiency compared to the synchronous implementation. Hypercube topology will be considered as the most efficient network topology for this kind of problems.

Halley-like method for the simultaneous approximation of all zeros of a polynomial P of degree n has been considered in [14] and [11]. For simplicity, the approximations $z_j^{(m-r)}$ to the roots $\zeta_1, \ldots, \zeta_n$ at the iteration step m will be shortly denoted with z_j if $r = 0$ and z_j^* if $r > 0$. According to this notation we introduce the errors $\epsilon_i = z_i - \zeta_i$ and $\epsilon_j^* = z_j^* - \zeta_j$. The new approximation $z_i^{(m+1)}$ will be denoted with $\hat{z}_i$ and the corresponding error with $\hat{\epsilon}_i = \hat{z}_i - \zeta_i$. Besides, we define the sums

$$S_{\lambda,i} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{1}{(z_i - z_j^*)^\lambda} \quad (i = 1, \ldots, n;\ \lambda = 1, 2),$$

$$\Sigma_{1,i} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{1}{z_i - \zeta_j} \quad (i = 1, \ldots, n),$$

the abbreviations

$$a_{ij} = (z_i - \zeta_j)(z_i - z_j^*), \quad b_{ij} = 2z_i - z_j^* - \zeta_j,$$

and the function

$$f(z) = \frac{P'(z)}{P(z)} - \frac{P''(z)}{2P'(z)}.$$

Then Halley-like method reads:

$$\hat{z}_i = z_i - \frac{1}{f(z_i) - \dfrac{P(z_i)}{2P'(z_i)}\left[S_{1,i}^2 + S_{2,i}\right]} \quad (i = 1, \ldots, n) \tag{11}$$

or in the form

$$\hat{z}_i = z_i - \frac{\dfrac{2P'(z_i)}{P(z_i)}}{\left[\dfrac{P'(z_i)}{P(z_i)}\right]^2 + \dfrac{P'(z_i)^2 - P''(z_i)P(z_i)}{P(z_i)^2} - S_{1,i}^2 - S_{2,i}}. \tag{12}$$

By the way, we observe that the function f in the denominator of (11) appears in the well-known Halley iteration formula

$$\hat{z} = z - \frac{1}{f(z)}$$

for the determination of a single root.

In the following we will show that the asynchronous Halley-like method (11) belongs to the class of methods for which the relation (9) holds, which means that Theorem 1 can be also applied to this method. For that purpose we use the identities

$$\frac{P'(z)}{P(z)} = \sum_{j=1}^{n} \frac{1}{z - \zeta_j} \tag{13}$$

and

$$\frac{P'(z)^2 - P''(z)P(z)}{P(z)^2} = \sum_{j=1}^{n} \frac{1}{(z - \zeta_j)^2}, \tag{14}$$

which can be easily derived by logarithmic differentiation.

First, from (13) we have

$$\frac{2P'(z_i)}{P(z_i)} = 2\Big(\frac{1}{z_i - \zeta_i} + \sum_{j \neq i} \frac{1}{z_i - \zeta_j}\Big) = 2\Big(\frac{1}{\epsilon_i} + \Sigma_{1,i}\Big)$$

and

$$\Big(\frac{P'(z_i)}{P(z_i)}\Big)^2 - S_{1,i}^2 = \left(\sum_{j=1}^{n} \frac{1}{z_i - \zeta_j}\right)^2 - \left(\sum_{j \neq i} \frac{1}{z_i - z_j^*}\right)^2$$
$$= \Big(\frac{1}{\epsilon_i} - \sum_{j \neq i} \frac{\epsilon_j^*}{a_{ij}}\Big)\Big(\frac{1}{\epsilon_i} + S_{1,i} + \Sigma_{1,i}\Big).$$

Using the identity (14) we find

$$\frac{P'(z_i)^2 - P''(z_i)P(z_i)}{P(z_i)^2} - S_{2,i} = \sum_{j=1}^{n} \frac{1}{(z_i - \zeta_j)^2} - \sum_{j \neq i} \frac{1}{(z_i - z_j^*)^2} = \frac{1}{\epsilon_i^2} - \sum_{j \neq i} \frac{b_{ij}\epsilon_j^*}{a_{ij}^2}.$$

According the two last relations we find from (12)

$$\hat{z}_i - \zeta_i = z_i - \zeta_i - \frac{2\Big(\frac{1}{\epsilon_i} + \Sigma_{1,i}\Big)}{\Big(\frac{1}{\epsilon_i} - \sum_{j \neq i} \frac{\epsilon_j^*}{a_{ij}}\Big)\Big(\frac{1}{\epsilon_i} + S_{1,i} + \Sigma_{1,i}\Big) + \frac{1}{\epsilon_i^2} - \sum_{j \neq i} \frac{b_{ij}\epsilon_j^*}{a_{ij}^2}},$$

that is,

$$\hat{\epsilon}_i = \epsilon_i - \frac{2\epsilon_i\big(1+\epsilon_i\Sigma_{1,i}\big)}{\Big(1-\epsilon_i\sum\limits_{j\neq i}\frac{\epsilon_j^*}{a_{ij}}\Big)\big(1+\epsilon_i S_{1,i}+\epsilon_i\Sigma_{1,i}\big)+1-\epsilon_i^2\sum\limits_{j\neq i}\frac{b_{ij}\epsilon_j^*}{a_{ij}^2}}$$

$$= \epsilon_i - \frac{2\epsilon_i+2\epsilon_i^2\Sigma_{1,i}}{2+\epsilon_i\Big(S_{1,i}+\Sigma_{1,i}-\sum\limits_{j\neq i}\frac{\epsilon_j^*}{a_{ij}}\Big)-\epsilon_i^2\Big[\big(S_{1,i}+\Sigma_{1,i}\big)\sum\limits_{j\neq i}\frac{\epsilon_j^*}{a_{ij}}+\sum\limits_{j\neq i}\frac{b_{ij}\epsilon_j^*}{a_{ij}^2}\Big]}.$$

Let G_{ij} denotes the denominator in the last relation. After short rearrangement of the previous relation we obtain

$$\hat{\epsilon}_i = \frac{\epsilon_i^2}{G_{ij}}\Big\{S_{1,i}-\Sigma_{1,i}-\sum_{j\neq i}\frac{\epsilon_j^*}{a_{ij}}-\epsilon_i\Big[\big(S_{1,i}+\Sigma_{1,i}\big)\sum_{j\neq i}\frac{\epsilon_j^*}{a_{ij}}+\sum_{j\neq i}\frac{b_{ij}\epsilon_j^*}{a_{ij}^2}\Big]\Big\}. \quad (15)$$

Since

$$S_{1,i}-\Sigma_{1,i}-\sum_{j\neq i}\frac{\epsilon_j^*}{a_{ij}} = \sum_{j\neq i}\Big(\frac{1}{z_i-z_j^*}-\frac{1}{z_i-\zeta_j}-\frac{z_j^*-\zeta_j}{(z_i-z_j^*)(z_i-\zeta_j)}\Big) = 0,$$

from (15) there follows

$$\hat{\epsilon}_i = -\frac{\epsilon_i^3}{G_{ij}}\Big[\big(S_{1,i}+\Sigma_{1,i}\big)\sum_{j\neq i}\frac{\epsilon_j^*}{a_{ij}}+\sum_{j\neq i}\frac{b_{ij}\epsilon_j^*}{a_{ij}^2}\Big]$$

or in the form

$$\hat{\epsilon}_i = \epsilon_i^3\sum_{j\neq i}c_{ij}\epsilon_j^*, \quad \text{with } c_{ij} = -\frac{a_{ij}\big(S_{1,i}+\Sigma_{1,i}\big)+b_{ij}}{a_{ij}^2G_{ij}}. \quad (16)$$

The quantities $a_{ij}, b_{ij}, S_{1,i}$ and $\Sigma_{1,i}$ are bounded, namely $a_{ij} \to (\zeta_i - \zeta_j)^2$, $b_{ij} \to 2(\zeta_i - \zeta_j)$, while $S_{1,i}$ and $\Sigma_{1,i}$ tend to $\Sigma_{j\neq i}(\zeta_i - \zeta_j)^{-1}$. Also, assuming that the starting approximations are sufficiently close to the exact zeros, the quantities ϵ_i will be small enough so that there exists a positive number $\mu < 2$ such that $|G_{ij}| > 2-\mu$. Therefore, c_{ij} is bounded in modulus; hence, the relation (16) is of the form (9) with $q = 3$ so that we can directly applied the assertion of Theorem 1. Thus, the order of convergence of the asynchronous Halley-like method is at least η_A, where $\eta_A > 3$ is the unique positive root of the equation $\eta^{r+1} - 3\eta^r - 1 = 0$. Since we assume reasonably good starting approximations, because of the very fast convergence the total number of iteration steps will be rather small (2 or 3 steps in practice). For these reason, greater values of r should not be expected.

3. Comparison of asynchronous and synchronous version

In this section we present a theoretical implementation of Halley-like method (11) on 4-dimensional hypercube with $k = 2^4 = 16$ processors where the diameter is $D = 4$. We take a realistic parameters ratio [3] $\beta_c/\tau_a \cong 10^3/6$ (dashing horizontal line). The total arithmetic cost of Halley-like method is $42n^2\tau_a + o(n^2)$, that is, $\alpha = 42$. Polynomials of the degrees $n = 16$ (the so-called full parallelization when the number of processors is equal to the polynomial degree) and $n = 30$ have been considered. The bound values $\beta_c/\tau_a = \frac{21}{8}n^2$ for these values of n are represented by the full horizontal lines in Fig. 1.

Realistic areas where the asynchronous Halley-like method can be more efficient are given by light shaded area for $n = 16$ and darker shaded area (partially invisible) for $n = 30$. The critical values which determine the necessary upper bound ratio $N_{\text{asy}}/N_{\text{syn}}$ are given by V_1 for $n = 16$ and V_2 for $n = 30$. Obviously, a more stronger requirement for the needed ratio $N_{\text{asy}}/N_{\text{syn}}$ appears in the case of the higher degree; namely, this ratio is closer to 1 when the degree n is higher, which is more difficult to realize in practice. Following (8) we find for the worst case model ($\eta_A = 3$) that the ratio of the number of iteration steps $N_{\text{asy}}/N_{\text{syn}}$ which provides a greater efficiency of asynchronous implementation must be smaller than $\log 4/\log 3 \cong 1.26$. For the considered ratio $\beta_c/\tau_a = 10^3/6$ this is available (theoretically) if $n < 26$. On the other side, the higher n permits the topology with the greater ratio β_c/τ_a (see Fig. 1).

Finally, we wish to consider a more general problem. We recall that *Durand-Kerner* method (with a quadratic convergence) in a synchronous implementation have the best performances in a wide class of simultaneous methods although it possesses relatively low convergence rate (see [5,7]). The following question arises: *What is the influence of the convergence rate of applied methods in a practical realization when the parallel implementation is performed asynchronously?* In other words, we wish to investigate the case when (from (8))

$$\lambda_q = \frac{N_{\text{asy}}}{N_{\text{syn}}} \cong \frac{\log \eta_S}{\log \eta_A} = \frac{\log(q+1)}{\eta_A(q)} < V, \tag{17}$$

in dependendence on the parameter q which defines the convergence orders of synchronous and asynhronous versions. Here $\eta_A(q)$ is the unique positive root of the equation (10) and V is the critical ratio which is the upper bound of the possible area of dominance of the asynchronous version (see Section 1

and Fig. 1). For this purpose, we have solved the equation (10) and found the ratio λ_q for the delay $r = 0, 1, 2, 3, 4$ and the entries $q = 1, 2, 3, 4$ which are of a practical importance. The dependence λ_q against q with the delay r as a parameter is displayed in Fig. 2.

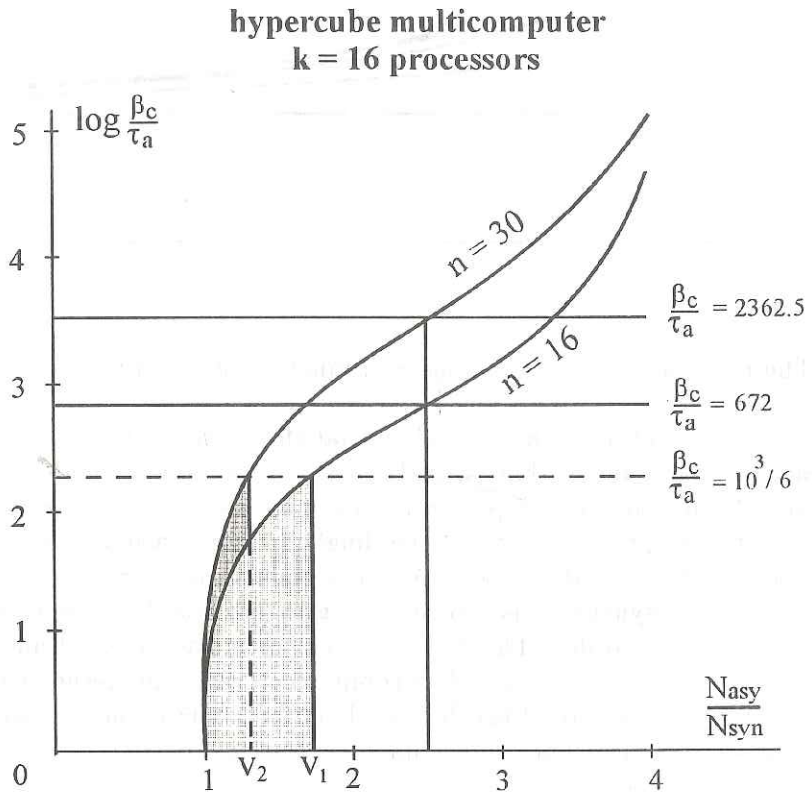

Fig. 1 Dominant areas of Halley-like asynchronous method

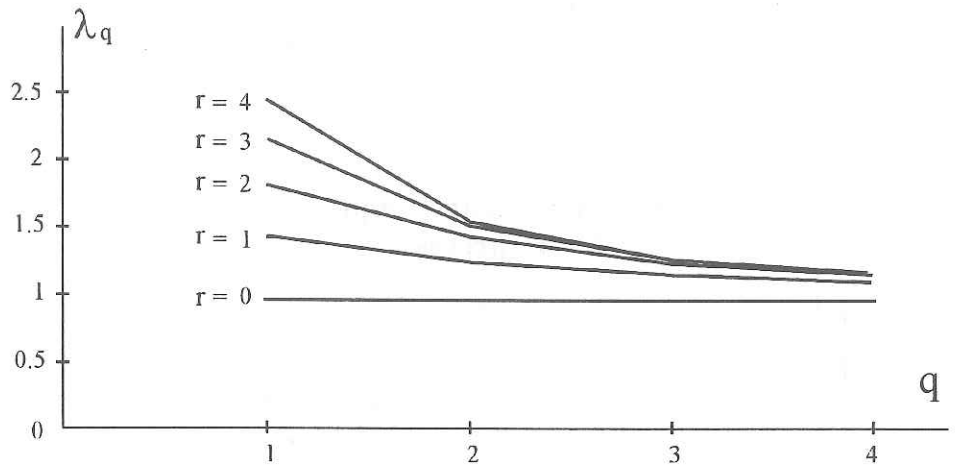

Fig. 2 The ratio of the iteration steps as a function of the convergence order

From Fig. 2 we observe that, for all r, that the ratio λ_q of iteration steps is smaller for a greater q, that is, in the case of methods of higher order. But, under fixed real network performances, a smaller ratio $\lambda = N_{\text{asy}}/N_{\text{syn}}$ means that the inequality (8) (and, accordingly, (17)) is feasible much easier. Hence, the possibility that an asynchronous algorithm be more efficient than the corresponding synchronous algorithm is greater if the basic method has a higher convergence order. This fact gives a slight advantage of Halley-like method (which is of the fourth order) compared to Durand-Kerner method (quadratic convergence) and Ehrlich-Aberth method (cubic convergence) but only in the case of the asynchronous implementation.

References

[1] G.M. Baudet, *Asynchronous iterative methods for multiprocessors.* J. of ACM **2** (1978), 226-244.

[2] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and distributed computation - numerical methods.* Prentice-Hall Inc. 1989.

[3] L. Bomans and D. Roose, *Communication benchmarks for the iPSC/2.* Hypercube and Distributed Computers (Proc. I European Workshop on hypercube and Distributed Computers, eds. F. Andre and J. P. Verjus), North Holland, Amsterdam 1989, pp. 93-104.

[4] M. Cosnard and P. Fraigniaud, *Asynchronous Durand-Kerner and Aberth polynomial root finding methods on a distributed memory multicomputer.* Parallel Computing **9** (1989) 79-84.

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

[6] M. Cosnard and P. Fraigniaud, *Asynchronous polynomial root finding methods.* Research report 90-21, LIP-IMAG, Ecole Normale Supérieure de Lyon, France 1990.

[7] M. Cosnard and P. Fraigniaud, *Analysis of asynchronous polynomial root finding methods on a distributed memory multicomputer.* IEEE Transaction on Parallel and Distributed Systems (to appear).

[8] P. Fraigniaud, *Performance analysis of broadcasting in hypercubes.* Hypercube and Distributed Computers (Proc. I European Workshop on hypercube and Distributed Computers, eds. F. Andre and J. P. Verjus), North Holland, Amsterdam 1989, pp. 311-328.

[9] T.L. Freeman, *Calculating polynomial zeros on a local memory parallel computer.* Parallel Computing **12** (1989) 351-358.

[10] T.L. Freeman and M.K. Bane, *Asynchronous polynomial zero-finding algorithms.* Parallel Computing **17** (1991) 673-681.

[11] M.S. Petković, *On Halley-like algorithms for simultaneous approximation of polynomial complex zeros.* SIAM J. Numer. Anal. **3** (1989), 740-763.

[12] M.S. Petković, *Iterative methods for simultaneous inclusion of polynomial zeros.* Springer-Verlag, Berlin-Heidelberg-New York 1989.

[13] S. Tričković, M. Trajković and M. Petković, *Asynchronous methods for simultaneous determination of polynomial roots* (submitted).

[14] X. Wang and S. Zheng, *A family of parallel and interval iterations for finding all roots of a polynomial with rapid convergence (I).* J. Comput. Math. **1** (1984), 70-76.

Faculty of Electronic Engineering, P.O. Box 73, 18 000 Niš