Proceedings of ICCC 2004, Băile Felix Spa-Oradea, Romania
pp. 149-156, 2004

Some parallel implementations of the Random Filtered Asynchronous Iterative Method

Ioan Dziţac, Horea Oros, Simona Dziţac

Abstract: In this paper we present synchronous/ asynchronous parallel procedure of RFAIM method (Random-Filtered Asynchronous Iterative Method). RFAIM method is introduced by first author in 2000 [3] and developed in 2002 [2]. This method let be used for calculating the fixed point of some map definite on subsets from $\mathbb{R}^n$ or for solving some system with n equations and n unknowns. In essence, the idea of RFAIM is the following: when the calculated value in the determinist way, by iterating one component, doesn't make part of so called waiting-interval (interval in which we aspect for the wanted value to be found, based on a certain criteria or only based on a presumption), it is substitute with a randomize value in the considerate interval. With the help of synchronous/ asynchronous parallel procedures of RFAIM we succeeded in calculating fixed points of map which aren't contractions points which cannot be determinate; for example, through the Jacobi method, the Gauss-Seidel'smethod, the Robert-Charnay-Musy's chaotiques serial- parallel iterative method [4] or Baudet's asynchronous iterative method [1].

Keywords: parallel algorithm, synchronous/asynchronous procedure, iterative method, fixed point, nonlinear system, RFAIM method.

1 Introduction

Definition 1. Let it be an algebraic equation system (linear or nonlinear), arranged under the form of fixed point system in $\mathbb{R}^n$:

$$x_k = f_k(x), k = 1..n, x \in I = \prod_{k=1}^{n} I_k \subset \mathbb{R}^n \tag{1}$$

where $I_k = [a_k, b_k] \subset \mathbb{R}, x = (x_1, x_2, \ldots, x_n) \in I$ and $f_k : I \subset \mathbb{R}^n \to \mathbb{R}$ are real continuous functions. We call the map $f = (f_1, f_2, \ldots, f_n) : I \subset \mathbb{R}^n \to \mathbb{R}$ the attached map of a fixed point system.

Remark 1. Upon the attached map $f = (f_1, f_2, \ldots, f_n) : I \to \mathbb{R}^n$ we do not know anything in advance whether it is or it is not a contraction on I, but "we wait" on having fixed points in I. This "waiting" can be based on some presumption or even on some certainties.

Remark 2. The fixed points of f, if they exist, are solution of the system 1 and reciprocal.

Definition 2. Let $N = 1, 2, \ldots, n \subset \mathbb{N}$ be the index set of components of f an n compact interval of real numbers $I_k = [a_k, b_k] \subset \mathbb{R}, k \in N$, which we shall call it waiting intervals. The product $I = \prod_{k=1}^{n} I_k \subset \mathbb{R}^n$ shall be called n-waiting interval.

Remark If $f(I) \subset I$ and f is a contraction on I, then f has a unique fixed point on I (the result of Perov's fixed point theorem).

In [2] is introduced the notion of *apparent fixed point* and an algorithm of numerical determination of apparent fixed point, which can be used although for determinate the regular fixed points.

Definition 3. Any nonempty part $J \subset \mathbb{N}$ shall be named index filter or short, filter. Let $(J_m)_{m \geq 1}$ be a sequence of filter with the terms: $J_1, J_2, \ldots, J_m, \ldots$. We shall denote with M_k the number which shows how many times the index $k \in N$ appears in the index filters until the m order included, if for all $k \in \mathbb{N}$ and $m \to \infty$, we have $M_k \to \infty$, then we shall say that the sequence of filters is exhaustive.

Remark 1. The sequence of filters we shall attached them to iterative algorithms. The simple successive vector iteration algorithm is corresponding the constant sequence of filters: $J_m = N, m = 1, 2, \ldots$, which, of course is exhaustive.

Remark 2. To the Gauss-Seidel's algorithm is corresponding a sequence of exhaustive filters, with the general term defined by:

$$J_m = \begin{cases} \{m\} & \text{if } m \in N \\ \{k\} & \text{if } m \equiv k(mod\ n) \text{ and n is not a factor of m} \\ \{n\} & \text{if n is a factor of m} \end{cases} \tag{2}$$

Definition 4. Let X be a discrete random variable with the uniform distribution:

$$X : \begin{pmatrix} 0 & 1 & \dots & n-1 \\ \frac{1}{n} & \frac{1}{n} & \dots & \frac{1}{n} \end{pmatrix} \tag{3}$$

We shall denote with RND(n) a generating set of randomize number which return a selection value of a randomize variable X.

Definition 5. Let X be the continuous randomize variable with the uniform distribution:

$$X : \begin{pmatrix} x \\ 1 \end{pmatrix}, x \in [0,1]. \tag{4}$$

We shall denote with RND a generating set of real random numbers which return a selection value of all random variable X.

Remark RND(n) and RND shall be considerate by the case as theoretical generating or generating sets obtained through a predefined method in a programming language or definite by the user, which generates pseudo-random numbers with good statistics qualities: satisfying a concordance test referring to the uniform econ, the stochastique independence and the big reproduction period.

A generating set of RND type is obtain like this:

$$RND = \frac{RND(m+1)}{m}, m \in \mathbb{N}^* \tag{5}$$

where m is conveniantly chosen.

Definition 6. Let $[a_k, b_k] = I_k$, be a compact interval of real numbers. We shall denote with $RND(I_k)$ the generating set given by the relation :

$$RND(I_k) = a_k + (b_k - a_k) * RND. \tag{6}$$

Remark It is clear that $RND(I_k)$ returns a selection value of a continuous randomize variable X, with the uniform distribution:

$$X : \begin{pmatrix} x_k \\ \frac{1}{b_k - a_k} \end{pmatrix}, x_k \in [a_k, b_k]. \tag{7}$$

Definition 7. Let $f_k : I \to \mathbb{R}, k = 1, 2, \dots, n$, be the continuous functions and $f = (f_1, f_2, \dots, f_n) : I \to \mathbb{R}^n$. For each index $k \in \mathbb{N}$ we shall consider an arbitrary value, but fixed, $r_k \in [0,1]$. To the vector $r = (r_1, \dots, r_k, \dots, r_n)$ (whom we shall call it the invariance selector) and the application f we shall associate the application $f^r : I \to I$, $f^r = (f_1^r, f_2^r, \dots, f_n^r)$, with the components given by:

$$f_k^r(x) = \begin{cases} f_k(x) & if\ f_k(x) \in I_k \\ a_k + r_k(b_k - a_k) & otherwise \end{cases} \tag{8}$$

The application $f^r : I \to I$ is named the invariance approximate selected by r selector.

Remark From the relation 8 results that if x_r^* is a fixed point of invariance approximate $f^r : I \to I$ of $f : I \to \mathbb{R}^n$, then x_r^* is an apparent fixed point of the f function.

In [2] we prove the following:

Theorem 8. *If the application $f = (f_1, f_2, \dots, f_n) : I \to \mathbb{R}^n$ has a fixed point $x^* \in I$, then f has at least one apparent fixed point in I.*

Remark The property of the previous theorem can be stated shortly this way: any fixed point of f is although an apparent fixed point of f. The reciprocal isn't true.

Definition 9. Let $f = (f_1, f_2, \ldots, f_n) : I \to \mathbb{R}^n$. We shall name the invariance approximate selected by the selector r and filtered by the J filter with elements from $r = (r1, \ldots, r_k, \ldots, r_n)$, the map: $f^{J(r)} = (f_1^{J(r)}, f_2^{J(r)}, \ldots, f_n^{J(r)}) : I \to I$, where:

$$f_k^{J(r)}(x) = \begin{cases} f_k^r(x) & \text{if } k \in J \\ x_k & otherwise \end{cases} \tag{9}$$

Definition 10. We shall name the partial sequence of randomize filtered iterations through an exhaustive sequence of filters $(Jm)_{m\geq 1}$ associated to the system 1, the recurrent sequence:

$$x_k^{m+1} = f_k^{J_m(r)}(x^m), m = 0, 1, \ldots, k = 1..n, \tag{10}$$

where $r = (r_1, \ldots, r_k, \ldots, r_n)$, $r_k = RND(I_k) = a_k + (b_k - a_k) * RND$, and $x^0 = (x_1^0, x_2^0, \ldots, x_n^0) \in I$ is a initial vector of approximation of $x^* = (x_1^*, x_2^*, \ldots, x_n^*) \in I$.

Definition 11. We shall call the iteration randomize filtered sequence through the exhaustive sequence $(J_m)_{m\geq 1}$ associated of (1), the vector sequence $(x^{m+1})_{m\geq 0}$, with those n components given by the formulas (10).

2 A parallel implementation based of master-slave model

Suppose that for solving the system (1) we have available a parallel calculus system formed by n slave processors: $P_1, P_2, \ldots, P_n$, which are working under the control and under the command of one master processor P. Each processor P_i shall evaluate the x_i^{k+1} component from the partial sequence of filtered iterations (10) and shall deposit the result into a vector of current approximation from the common memory administrated by P and accessible in every moment, even simultaneously, to all slave processors. The processor P shall momentary evaluate on every deposit (it can be made deposit at once) if the obtained approximation satisfies the fixed stop criteria. In the case in which this criteria is not satisfied, the processors which have not finished their previous job shall make a new iteration, taking as entering date the current approximation vector (at first all processors have as entrance the initial approximation vector x_0). That way, the sequence of filters shall be automatically generated by the asynchronous process of computation and it shall be accomplish the theoretical condition of being exhaustive. We shall call this sequence of filters, the asynchronous sequence of filters.

Definition 12. We consider the sequence of asynchronous filtrated iterations described by the sequence of asynchronous filtrated iteration described by the relations (9) and (10), in which the filter J is automate generated by the asynchronous iterative process. The iterative process of obtaining the terms of the sequence (10) shall be named the asynchronous randomize filtered iterative algorithm, and this method shall be assign through the acronym RFAIM.

In [2] we proved the following:

Theorem 13. *The asynchronous randomize filtered iteration (the RFAIM algorithm) converge to an apparent fixed point $x_a^* \in I$ of the attached map f. If, moreover, the attached map $f = (f_1, f_2, \ldots, f_n)$ has in n waiting intervals I a unique fixed point $x^* \in I$, then $x_a^* = x^*$.*

Theorem 14. *Let $I \subset \mathbb{R}^n$. If $f(I) \subset I$ and f is a contraction in the canonical vector norm on I, then the randomize filtered iteration sequence converges to a unique fixed point of f, $x^* = (x_1^*, x_2^*, \ldots, x_n^*) \in I$.*

Remark The theorem 13 is available in the contraction case too in a scalar norm.

Let $I \subseteq \mathbb{R}^n$, $f : I \to \mathbb{R}^n$, $f = (f_1, f_2, \ldots, f_n)$, $f_i : X \to \mathbb{R}(i = 1..n)$ are real function of n variable, continuous.

We shall consider in $\mathbb{R}^n$ the canonical vector norm:

$$\|x\|_{\rightarrow} : \mathbb{R}^n \to \mathbb{R}^n, \|x\|_{\rightarrow} = (|x_1|, |x_2|, \ldots, |x_n|), x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n.$$

Remark If we stop the iterative process when the following stop condition:

$$\|x_{new} - x_{old}\|_{\rightarrow} \leq \varepsilon, \tag{11}$$

where ϵ is a vector with n components pretty small, a priori fixed, then the vector x_{new} (obtained on the last asynchronous iteration) shall be a numerical approximation for the apparent fixed point of f (with x_{old} we have denoted the vector obtained on the penultimate asynchronous iteration).

If the condition is accomplish:

$$\|x_{new} - f(x_{new})\|_{\rightarrow} \leq \varepsilon, \tag{12}$$

then we can consider that $x_n ew$ has a pretty good approximation of x^*, a fixed point of f from the n-interval I. If we know from before that f has a fixed point in I, then it can be determine through RFAIM's algorithm, the iterative process is stopping when the condition (11) or (12) are satisfied.

Remark RFAIM shall assure the convergence to the solution x^*, because the relation doesn't permit the components to exit from the waiting intervals, and the components selection way which shall be iterated every step of the way, assures the theoretical iteration of an infinite times of each one of them.

3 Synchronous parallel procedure

In procedure 1 and 2 we give the pseudo-code description of the synchronous parallel implementation of the RFAIM algorithm, based of master- slave model for parallel virtual machine implementation. In the expression "synchronous-RFAIM" is only an apparent contradiction between the terms "synchronous" and "asynchronous", because the synchronous case can be accident accomplished through an asynchronous execution (is true with a very small probability). Here we have an explicit synchronization, accomplished through a program.

```
Procedure 1. The synchronous procedure for slave processors
procedure processor[i] begin
    receive x_pre from  master_P;
     x[i] := f[i](x_pre);
     send  x[i] to master_P;
     wait message from master_P;
end.
```

```
Procedure 2. The synchronous procedure for master P
program RFAIM_SYNC
begin
   select x_init;
   for  i = 1 to n do in parallel
     send x_pre to P[i];
     cobegin processor[i];
     coend;
   endfor;
   for  i= 1 to n do in parallel
     receive  x[i] from P[i];
     if (a[i]< x[i] and x[i]< b[i])
        then  x_new[i]:=x[i]:
           else
           x[i]:=a[i]+(b[i]-a[i])*rnd;
           x_new[i]:=x[i];
           endelse;
        x_prec:=(x_new[1], x_new[2],.., x_new[n]);
   endfor ;
   if no stop condition then
     for i = 1 to n do in parallel
         send x_preto P[i];
     endfor ;
     else
            print x_pre;
```

```
     endelse;
   endif;
end RFAIM_SYNC.
```

4 Asynchronous parallel procedure

In procedure 3 and 4 we give the pseudo-code description of the asynchronous parallel implementation of the RFAIM algorithm, based of master- slave model for parallel virtual machine.

```
Procedure 3. The asynchronous procedure for slave processors
procedure processor[i]
begin
    receive x_old from P;
     x_pro[i]:=fi(x_old);
if (a[i]<=x_pro[i] and x_pro[i]<=b[i]) then x_new[i]:=x_pro[i]
      else
        x_new[i]:=a[i]+(b[i]-a[i])*rnd;
      endelse;
endif;
    send x_new[i] to P;
     wait message from P;
end.
```

```
Procedure 4. The asynchronous parallel procedure for master processor
procedure RFAIM_ASYNC
begin
    select x_init;
    x_old:= x_init;
    for i = 1 to n do in parallel asynchronous
        send x_new to processor[i];
    parbegin processor[i];
    parend; asynchronous
    receive x_new[i] from processor[i];
    changes in x_old the x_old[i] component x_new[i]
    and redefine  the obtained vector through x_new;
    xold xold[I] xnew[i] xnew;
    if  ||x_new − x_old||→ ≤ ε & ||x_new − f(x_new)||→ ≤ ε
      then
        x_fixed:=x_new and stops all processors;
        else
        x_old:=x_new;
        send x_old to all processors witch have finished task;
        endelse;
    endif;
endfor;
    print x_new;
end RFAIM_ASYNC.
```

5 Numerical examples

Example 1. [4]

$$\begin{aligned}
x_1 = f_1(x) &= (x_1 + x_2)/4 + x_3/20 \\
x_2 = f_2(x) &= (\cos(x_1 + x_2 + x_4 - x_5 - x_8 - x_9))/6.6 \\
x_3 = f_3(x) &= 0.5 * \cos(x_1/2 + x_3/3 - 2 * x_5/3) + x_7/5 \\
x_4 = f_4(x) &= 0.2 * \sin(2 * x_3) + 0.2 * \cos(x_1 + x_4) + x_5/6 \\
x_5 = f_5(x) &= (\exp(-x_1^2) + \exp(-x_8^2))/2 \\
x_6 = f_6(x) &= (x_1 - x_2 + x_8 + x_9 + 3 * x_{10})/7.2 \\
x_7 = f_7(x) &= \exp(-x_1) + x_9/11 \\
x_8 = f_8(x) &= (\cos(x_1 + x_3 + x_6 + x_9))/5 \\
x_9 = f_9(x) &= (\sin(x_2 + x_4 + x_5 + x_8))/5 \\
x_{10} = f_{10}(x) &= (\exp(-x_3^2) + \exp(-x_8^2))/2 - x_{10}/11
\end{aligned}$$

It has been establish in [3] that the attached operator of this system is a contraction in canonical vector norm, so it has a unique fixed point.

Let us consider now system from example 2 which is in fact system from example 1, with the components 1, 8, 9 and 10 written in another form, so that it isn't satisfied the contraction condition.

Example 2. (Example 1 modified)

$$\begin{aligned}
x_1 = g_1(x) &= x_2/3 + x_3/15 \\
x_2 = g_2(x) &= (\cos(x_1 + x_2 + x_4 - x_5 - x_8 - x_9))/6.6 \\
x_3 = g_3(x) &= 0.5 * \cos(x_1/2 + x_3/3 - 2 * x_5/3) + x_7/5 \\
x_4 = g_4(x) &= 0.2 * \sin(2 * x_3) + 0.2 * \cos(x_1 + x_4) + x_5/6 \\
x_5 = g_5(x) &= (\exp(-x_1^2) + \exp(-x_8^2))/2 \\
x_6 = g_6(x) &= (x_1 - x_2 + x_8 + x_9 + 3 * x_{10})/7.2 \\
x_7 = g_7(x) &= \exp(-x_1) + x_9/11 \\
x_8 = g_8(x) &= \arcsin(5 * x_9) - x_2 - x_4 - x_5 \\
x_9 = g_9(x) &= \arccos(5 * x_8) - x_1 - x_3 - x_6 \\
x_{10} = g_{10}(x) &= 11(\exp(-x_3^2) + \exp(-x_8^2))/24
\end{aligned}$$

The numerical results of some classical methods for the start vector $x_0 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and $x_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$, are presented in table 1.

The map $g : D \subset \mathbb{R}^{10} \to \mathbb{R}^{10}$, $g = (g_1, g_2, \ldots, g_{10})$, from the second example doesn't satisfy the request of the contraction theorem.

The first four methods applied on Example 2, aren't convergent or they cannot be applied because of the restrictive definition domain, of some of the function Jacobi, Gauss-Seidel programs whether they give an error message or they cycled on infinite. In change the RFAIM synchronous program leads us to the results from table 1.

Solution of the system from the examples 1 and 2, obtained through different methods or through the same method, but with different start vectors is presented in table 1

In the example 3 we have a system of 3 algebraic non-linear equation with 3 real unknown.

From the table 2 we see that the map associated to the system isn't a contraction on the hole space, so the used algorithm is being use for solving (simple successive iteration algorithm), in accordance with Perov's theorem it should be convergent for any choice of the start vector.

Method	Jacobi for Ex.1 and x^1	Gauss-Seidel for Ex.1 and x^1	RFAIM for Ex.2 and x^1	RFAIM for Ex.1 and x^0	RFAIM for Ex.2 and x^0
No. of vect. iter.	19	16	13-15	10-12	16-18
x_1	0.0872168780	0.0872168782	0.0889569301	0.0853306672	0.0872168781
x_2	0.1323158804	0.1323158805	0.1377887907	0.1263607698	0.1323158805
x_3	0.6466737693	0.6466737692	0.6449144164	0.6466737693	0.6466737692
x_4	0.5221144435	0.5221144435	0.5215423665	0.5198021063	0.5221144435
x_5	0.9945407606	0.9945407605	0.9930747880	0.9768165122	0.9945407605
x_6	0.3453312689	0.3453312688	0.3343999880	0.3503781711	0.3453312688
x_7	0.9344921910	0.9344921908	0.9244585190	0.9308854467	0.9344921907
x_8	0.0578458776	0.0578458776	0,0773705252	0.1997436445	0.0578458776
x_9	0.1981526895	0.1981526895	0.1053089665	0.1394451735	0.1981526895
x_{10}	0.7584951902	0.7584951901	0.7579771328	0.7415213722	0.7584951902

Table 1:

Example 3 (nonlinear system of 3x3)

$$\begin{aligned} x = f_1(x,y,z) &= 0.2\cos(x) + z^2 \\ y = f_2(x,y,z) &= x^2 + 0.2y - z \\ z = f_3(x,y,z) &= y^2 + 0.1\cos(z) \end{aligned}$$

Start vector	No. of vector iterations	Solution
x_0 = 0 y_0 = 0 z_0 = 0	25	$x* = 0.2068875811$ $y* = -0.0785041483$ $z* = 0.1056057901$
x_1 = 0 y_1 = -0.5 z_1 = 0.5	31	$x* = 0.2068875812$ $y* = -0,0785041494$ $z* = 0,1056057905$
x_2 = 1 y_2 = 1 z_2 = 1	∞	It doesn't obtain

Table 2: Solution generated by the Jacobi algorithm

Start vector	No. of vector iterations	Solution
x_0 = 0 y_0 = 0 z_0 = 0	18-22	x*= 0,206887581x y*=- 0,07850414xx z*= 0,105605790x
x_1 = 0 y_1 = -0.5 z_1 = 0.5	26-32	Idem
x_2 = 1 y_2 = 1 z_2 = 1	∞	It doesn't obtain

Table 3: Solution generated by the Baudet's algorithm

Applying Baudet's algorithm simulated by randomization we have a similar situation with the simple iteration case (see table 3). In table 4 we transcribed some results obtained through asynchronous RFAIM method, on diverse waiting intervals, which we have taken here as symmetrical intervals towards the origin and of equal length, and the components of the start vector are being random generated from the waiting interval.

In practice, about the solving system it is usually anticipated, more useful information such as the existence of a solution and the waiting interval in which are big intervals such as those from our table.

6 Conclusion

Arranging the systems so that the associated application to be a contraction is difficult and sometimes impossible. RFAIM permits de resolution of the fixed point for some applications that are not contractions, but have a fixed point.

The procedures presented in this paper have been implemented using the PVM package on a Linux network and the statistics shows that the asynchronous implementations converges more rapidly that the synchronous one.

The random start vector from the waiting interval	Number of vector iterations (on 10th experience)	Solution
[-1,1] or $[-0.5, 0.5]$	18-28	$x* = 0.206887581x$ $y* = -0.07850414xx$ $z* = 0.105605790xx$
[-10,10]	23-57	Idem
[-100,100]	32-574	Idem
[-1000,1000]	202-6337	Idem
[-10000,10000]	134-25380	Idem
[-1000000,100000]	$>$ 1.000.000	Idem

Table 4: Solution generated by the synchronous parallel procedure of the RFAIM method

References

[1] G. M. Baudet, Asynchronous iterative methods for multiprocessors, Research report of the Department of Computer Science, Carnegie-Mellon Univ. Pittsburg, Pennsylvania, 1976.

[2] I. Dziţac, Parallel and Distributed Procedures in Solving of some Operator Equations, Ph. D. Thesis, "Babeş Bolyai" University of Cluj Napoca, (2002), supervisor prof. dr. Grigor Moldovan. (Romanian).

[3] I. Dziţac, Random-filtered asynchronous iterative method, Bul. Stiint. Univ. Baia Mare, Ser. B, Matematica-Informatica, Vol. XVI (2000), Nr. 1, pp. 17-24.

[4] F. Robert, M. Charnay, F. Musy, Iterations chaotique serie- parallele pour des equations nonlineaire de point fixe, Aplikace Matimatiky 20 (1975), pp. 1- 38. (French)

University of Oradea, Romania
Department of Mathematics
Armatei Române St. No. 3-5, Oradea, Romania
E-mail: {idzitac,horos}@uoradea.ro