Bul. Ştiinţ. Univ. Baia Mare, Ser. B,
Matematică-Informatică, Vol. XVIII(2002), Nr. 2, 205 - 210

SOME ASYNCHRONOUS ITERATIVE PROCEDURES FOR FIXED POINT SYSTEMS RESOLUTION

Ioan DZIŢAC, Horea OROS, Simona DZIŢAC

Abstract. C. M. Baudet introduced asynchronous methods for multiprocessors in [1]. In this paper we present some simulated asynchronous parallel variants of classical iterative methods for fixed point systems resolution and some parallel procedures based of algorithm MIARF [5]. The programs presented in this paper for solving the fixed point system of nonlinear algebraic equations of [7] is based on the theoretical results obtained by the first author in [2], [3], [4], [5]

MSC2000: 68W10, 65Y05

Keywords: fixed point system, asynchronous methods.

1. Procedure MIARF for parallel virtual machine

In [5] the first author presents a parallel asynchronous method for systems of nonlinear algebraic equations resolutions.

In this paper in Tabel 1 and Tabel 2 we propose a variant for Parallel Virtual Machine (PVM), *master-slave* model.

Table 1. Parallel asynchronous algorithm MIARF: Procedure for slave proc[i]

```
Procedure proc[i]
// proc[i] := slave processor with number i
// xold := old vector of solution approximation
// xactual := actual vector of solution approximation
// xnew := new vector of the solution approximation
// master-P := master peocessors (supervisor of program)
begin
    receive xold from master-P; xactual := f_i(xold)
if (a[i] <= x[i] & xactual[i] <= b[i]) then xnew[i] := xactual[i]
       else xnew[i] := a[i] + (b[i] - a[i]) * rnd;
endif;
    send xnew[i] to master-P; wait message from master-P;
end;
```

Table 2. Parallel asynchronous algorithm MIARF: Procedure for master-P

```
Procedure MIARF
// ε := approximation error
// xinit := initial vector of the solution approximation
begin
    select xinit; xold := xinit;
    for i = 1 to n do in parallel asynchronous
        send xnew to proc[i];
    parbegin proc[i];
    parend; {asynchronous}
    receive xnew[i] from proc[i]; xold[i] := xnew[i]
    if ||xnew - xold||_∞ ≤ ε & ||xnew - f(xnew)||_∞ ≤ ε
      then
        xsol := xnew; stop every proc[i];
      else
        xold := xnew;
        send xold to every freedim-task slave;
      endelse;
    endif;
    endfor;
    print xnew;
end MIARF.
```

2. Some programs in C++ for solving the fixed point sistem of [RCM75]

In Table 3 we present the fixed point system of nonlinear algebraic equations of [7] with matrix of contractions K presented in Table 4.

In Table 5 we present the numerical solution of system (1) by [7].

In Table 6 we present a program in C for sample succesive approximation.

In Table 7 we present a program in C for chaotic Gauss-Seidel variant.

In Table 8 we present a Serial Program in C for MIARF simulation.

In Table 9 we present comparative results for P1, P2 and P3 programs.

Table 3. System (1) by [RCM75]

$$x_1 = f_1(x) = (x_1 + x_2)/4 + x_3/20 \quad (1)$$

$$x_2 = f_2(x) = (\cos(x_1 + x_2 + x_4 - x_5 - x_8 - x_9))/6.6 \quad (2)$$

$$x_3 = f_3(x) = 0.5 * \cos(x_1/2 + x_3/3 - 2 * x_5/3) + x_7/5 \quad (3)$$

$$x_4 = f_4(x) = 0.2 * \sin(2 * x_3) + 0.2 * \cos(x_1 + x_4) + x_5/6 \quad (4)$$

$$x_5 = f_5(x) = (\exp(-x_1^2) + \exp(-x_5^2))/2 \quad (5)$$

$$x_6 = f_6(x) = (x_1 - x_2 + x_8 + x_9 + 3 * x_{10})/7.2 \quad (6)$$
$$x_7 = f_7(x) = \exp(-x_1) + x_9/11 \quad (7)$$
$$x_8 = f_8(x) = (\cos(x_1 + x_3 + x_6 + x_9))/5 \quad (8)$$
$$x_9 = f_9(x) = (\sin(x_2 + x_4 + x_5 + x_8))/5 \quad (9)$$
$$x_{10} = f_{10}(x) = (\exp(-x_3^2) + \exp(-x_8^2))/2 - x_{10}/11 \quad (10)$$

Table 4. A contraction matrix K associated to the opearator of the system (1) by [RCM75]

0,2500	0,2500	0,05000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
0,1520	0,1520	0,0000	0,1520	0,1520	0,0000	0,0000	0,1520	0,1520	0,0000
0,2500	0,0000	0,1670	0,0000	0,3340	0,2000	0,2000	0,0000	0,0000	0,0000
0,2000	0,0000	0,4000	0,2000	0,1670	0,0000	0,0000	0,0000	0,0000	0,0000
0,4289	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,4289	0,0000	0,0000
0,1390	0,1390	0,0000	0,0000	0,0000	0,0000	0,0000	0,1390	0,1490	0,4170
0,8579	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0910	0,0000
0,2000	0,0000	0,2000	0,0000	0,0000	0,2000	0,0000	0,0000	0,2000	0,0000
0,0000	0,2000	0,0000	0,2000	0,2000	0,0000	0,0000	0,2000	0,0000	0,0000
0,0000	0,0000	0,4289	0,0000	0,0000	0,0000	0,0000	0,4289	0,0000	0,0910

Table 5. The solution of system (1) by [RCM75]

x[1]	=	0.088446725	x[2]	=	0.132741218
x[3]	=	0.662994799	x[4]	=	0.523507573
x[5]	=	0.994592264	x[6]	=	0.342456943
x[7]	=	1.010223912	x[8]	=	0.056024954
x[9]	=	0.192178378	x[10]	=	0.752260362

Table 6. Program P1

```
//Program in C++ for sample successive approximation
// include iostream.h, ioamanip.h, math.h
long double eps = 1.0e-9;
long double f1(long double x[])
{return ((x[0] + x[1]) /4 + x[2] /20);}
//function f3, f4, ..., f10 are defined similarly
bool finish(long double x1[], long double x2[], int nr)
{ bool ret = true; int i;
for(i = 0; i < nr; i++){
if( fabsl(x1[i] - x2[i]) > eps ){
ret = false; break; }
} return ret;
}
void main( )
{ long double x1[10] ={0}, x2[10] ={0}; unsigned i, contor = 0;
```

```
long double (*pf[])(long double []) =
{f1, f2, f3, f4, f5, f6, f7, f8, f9, f10};
do{ contor++;
for(i = 0; i < 10 ; i++ ) x1[i] = x2[i];
for(i = 0; i < 10 ; i++ ) x2[i] = pf[i](x1);
} while(!finish(x1, x2, 10));
// print the values in tabel x2 and contor
}
```

Table 7. Program P2

```
// Program in C++ for simulation asynchronous variant of P1
// P2 is equivalent of chaotic Gass-Seidel variant
long double eps = 1.0e-9;
// definitions of f1, f2, ..., f10, finish and tipar are the same as in P1
void shuffle(unsigned tab[], int n)
{
/* shuffle the elements of tab */
}
void main()
{ long double x1[10] ={0},x2[10] ={0}, x3[10] ={0};
unsigned i, contor = 0;
long double (*pf[])(long double []) =
{f1, f2, f3, f4, f5, f6, f7, f8, f9, f10};
unsigned tab[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
do{ contor++;
shuffle(tab, 10);
for(i=0; i < 10; i++) x1[i]=x2[i];
for(i=0; i < 10; i++){
x2[tab[i]] = pf[tab[i]](x3);
x3[tab[i]] = x2[tab[i]]; }
} while(!finish(x1, x2, 10));
// print the values in tabel x2
and contor
}
```

Table 8. Program P3

```
// Serial program in C++ for simulation MIARF}
long double eps = 1.0e-9;
long double intervale[][2] = {{eps, 1.2}, {eps, 1.2}, {eps, 1.2}, {eps, 1.2},
{eps, 1.2}, {eps, 1.2}, {eps, 1.2}, {eps, 0.2}, {eps, 0.2}, {eps, 1.2}};
long double f1(long double x[], int i)
{long double ret = ((x[0] + x[1]) /4 + x[2] /20);
if (ret > intervale[i][0] && ret < intervale[i][1])
```

```
return ret;
else return intervale[i][0] +
(intervale[i][1] - intervale[i][0])*(long double)rand()/RAND_MAX;
}
// functions f2, f3, f10 are defined analogously
void main()
{ long double x1[10] ={0}, x2[10] ={0}, x3[10] ={0};
unsigned nr_comp = sizeof(x1) / sizeof(x1[0]), i, contor = 0;
long double (*pf[])(long double [], int) =
{f1, f2, f3, f4, f5, f6, f7, f8, f9, f10};
unsigned tab[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
do{ contor++;
shuffle(tab, nr_comp);
for(i=0; i<nr_comp; i++){
x1[i]=x2[i];
for(i=0; i<nr_comp; i++){
x2[tab[i]] = pf[tab[i]](x3, i);
x3[tab[i]] = x2[tab[i]];
}
} while(!finish(x1, x2, nr_comp));
//print the values in tabel x2 and contor }
```

Table 9. The numerical results obtained by P1, P2 and P3

Vectorial steps	Program P1, 19	program P2, 12-14	Program P3, 14-19
x[1]	0.0872168780	0.0872168781	0.872168781
x[2]	0.1323158804	0.1323158805	0.1323158805
x[3]	0.6466737695	0.6466737693	0.6466737692
x[4]	0.5221144436	0.5221144435	0.5221144435
x[5]	0.9945407606	0.9945407605	0.9945407605
x[6]	0.3453312686	0.3453312687	0.3453312688
x[7]	0.9344921911	0.9344921909	0.9344921907
x[8]	0.0478458776	0.0578458776	0.0578458776
x[9]	0.1981526895	0.1981526895	0.19811526895
x[10]	0.7584951902	0.7584951902	0.7584951902

Conclusion:

The results in Table 9 confirm the theoretical anticipations: **The asynchronous iterative procedures increased the speed of he iterative processes and reduced the time of execution.**

REFERENCES

[1] BAUDET, G. M. *Asyncronous Iterative Methods for Multiprocessors* Research report of the Depratment of Computer Science Pittsburg Carnegie-Mellon University, Pennsylvania 15213, USA 1976

[2] DZITAC, I. *Rezolvarea sistemelor neliniare de punct fix prin metoda iteratiilor asincrone*, An. Univ. din Oradea, Tom III, 1993, pag 97-102.

[3] DZITAC, I. *Asincronizarea unor algoritmi paraleli clasici de rezolvare a sistemelor de ecuatii neliniare*, An. Univ. din Oradea, Tom IV, 1994, pag. 51-53.

[4] DZITAC, I. *Asyncrhronous Iterative Methods for Fixed-Point System Resolution*. 8-th SEFI European Seminar on Mathematics in Engineering Education, Prague, 1995

[5] DZITAC, I. *Random-filtered asynchronous iterative methods*, Bul. St. Univ. Baia Mare, Ser. B, Matematica Informatica, Vol. XVI (2000), Nr. 1, Pag. 17-24.

[6] DZITAC, I. *Parallel Solving a System of Nonlinear Equations*, The PAMM's periodical Bulletins for Applied & Mathemathics, Budapest, BAM-1905/2001 (XCVI-C), p. 141-150.

[7] ROBERT, F., CHARNAY, M., MUSY, F. *Iterations chaotique serie-paralleie pour des equations nonlinineaire de point fixe*. Aplikace Matematiky 20 (1975), p. 1-38.

Received: 11.09.2002

University of Oradea,
Department of Mathematics
3700 Oradea, Bihor.
E-mail: idzitac@uoradea.ro
horos@uoradea.ro
idzitac@yahoo.com