STUDIA UNIV. "BABEŞ–BOLYAI", MATHEMATICA, Volume **XLVI**, Number 4, December 2001

PARALLEL NUMERICAL METHODS FOR SOLVING NONLINEAR EQUATIONS

IOANA CHIOREAN

1. Introduction

The basis for constructing a parallel algorithm is either a serial algorithm or the problem itself. In trying to parallelize a serial algorithm a pragmatic approach would seem reasonable. Serial algorithm are analysed for frequently occurring basic elements which are then put into parallel form. These parallelization principles rely on definite serial algorithms. This corresponds to the serial way of thinking normally encountered in numerical analysis. What is needed is a parallel way of thinking.

In the following, we shall apply these principles to some numerical methods for solving single non-linear equations.

First we shall consider the one-dimensional case and assume that the real function $f(x)$ has only one zero in the interval $[a, b]$.

The methods for the determination of zeros can be subdivided into two types:

(i) search methods

(ii) locally iterative methods.

2. Search methods

2.1. **The Bisection Method.** The simplest search method is the bisection method, with the following code:

Repeat

$c := (a + b)/2$;

If $f(a) * f(c) > 0$ then $a := c$

else $b := c$

Until $abs(b - a) < \varepsilon$;

Considering f given as a function, continuous over the interval $[a, b]$, the serial bisection method needs $\log_2((b-a)/\varepsilon)$ function evaluations, additions and multiplications to enclose the zero in an interval of length ε.

We could adapt this serial algorithm at a parallel execution with 3 processors, by means of three sequences, a_n, b_n and c_n, every processor generating one of them:

$a[0] := a;$
$b[0] := b;$
$c[0] := (a+b)/2;$
$n := 0;$
Repeat in parallel
    $n := n+1;$
    If $f(a[n-1]) * f(c[n-1]) > 0$ then begin
        $a[n] := c[n-1];$
        $b[n] := b[n-1];$
        $c[n] := (a[n]+b[n])/2$
        end
    else begin
        $a[n] := a[n-1];$
        $b[n] := c[n-1];$
        $c[n] := (a[n]+b[n])/2$
    end
Until $abs(b[n]-a[n]) < \varepsilon$

Unfortunately, this parallel version of the bisection method does not bring a speed improvement, because, mainly, the number of operations is still of $\log_2((b-a)/\varepsilon)$ order, and we have to take into account the time spend for the processors communications.

But, if we think in parallel, the bisection method can clearly be performed on a computer consisting of r processors: for each iteration step the function is simultaneously evaluated at r points which thereby subdivide the actual interval in $r+1$ equidistant subintervals. The new interval points are chosen on the basis of the

signs of the function values. So, this parallel bisection method requires $\log_{r+1}((b-a)/\varepsilon)$ evaluations. The speed-up ratio is therefore

$$S = \frac{\log_2((b-a)/\varepsilon)}{\log_{r+1}((b-a)/\varepsilon)} = \log_2(r+1).$$

2.2. **Other Search Methods.** As we saw, in applying the bisection method it is sufficient to have a function which is continuous over the interval $[a, b]$. With functions having a high degree of smoothness, high order search methods can be constructed, which will converge yet faster.

The example given by Miranker [5] demonstrates this principle.

Let $f(x)$ be a function which is differentiable over $[0, 1]$ and let $f'(x) \in [m, M]$ $(m, M > 0)$ for all $x \in [0, 1]$. An algorithm is developed for a computer able to evaluate this function values $y_i = f(x_i)$, $x_i \in [0, 1]$, $(i = 1, 2)$, in parallel.

Initially, two piecewise linear functions $\underline{S}(x), \overline{S}(x)$, are defined which enclose $f(x)$ in $[x_1, x_2]$ (see Fig.1).

We see that $\underline{S}(x) \leq f(x) \leq \overline{S}(x)$, where

$$\overline{S}(x) := \begin{cases} y_1 + M(x - x_1), & x \leq \dfrac{(M-s)x_1 + (x-m)x_2}{M-m} \\ y_2 + m(x - x_2), & x \geq \dfrac{(M-x)x_1 + (s-m)x_2}{M-m} \end{cases}$$

where $s := \dfrac{y_2 - y_1}{x_2 - x_1}$. In a similar manner we define $\underline{S}(x)$.

The zero z of f is on the right-hand side of the zero x_1^* of $\overline{S}(x)$:

$$z \geq x_1^* = \max\left\{x_1 - \frac{y_1}{M}, x_2 - \frac{y_2}{m}\right\}$$

and

$$z \leq x_2^* = \min\left\{x - \frac{y_1}{m}, x_2 - \frac{y_2}{M}\right\}.$$

Considering each possible case, it can be shown that the inequality

$$\frac{x_2^* - x_1^*}{x_2 - x_1} \leq 1 - \frac{m}{M}$$

applies.

Comparing this method with the parallelized bisection method it is possible to obtain a speed-up, provided that

$$\frac{m}{M} \geq \frac{2}{3}$$

applies.

Remark. Miranker shows, also, that is $f \in C^2[0,1]$, the algorithm even has quadratic convergence.

3. **Locally iterative methods**

The best known locally iterative methods are the Newton's method and the secant method. Because the second one is a discrete version of the first one (the derivative is replaced by the difference quotient), we shall discuss only the secant method.

For x_0 an initial approximation and supposing, without restriction of generality, that $f \in C^d[a,b]$, $f'(x) \neq 0$, for all $x \in (a,b)$ and $f(a) < 0$ and $f(b) > 0$.

The the serial secant method is the following:

if $f(a) \cdot f''(a) > 0$ then $x_0 = b$ else $x_0 = a$; $n := 0$;

Repeat

$n := n + 1;$

$x[n] := a - \dfrac{b-a}{f(b)-f(a)} f(a)$

If $f(x[n]) < 0$ then $a := x[n];$

else $b := x[n]$

Until $abs(x[n] - x[n-1]) < \varepsilon$

Denoting

$$\delta_k := \max |z - x[k]|,$$

where z is the zero of f, we speak of convergence of order λ, if

$$\lim_{k\to\infty} \frac{\delta_{k+1}}{(\delta_k)^\lambda} = c > 0$$

applies.

It is known that the serial secant method has the order of convergence $\dfrac{1+\sqrt{5}}{2} \simeq 1.618\ldots$. Trying to parallelize this serial algorithm, we can use 3 processors to generate the sequences a_n, b_n and c_n, according to the following code:

$a[0] := a;$

$b[0] := b;$

$c[0] := (a[0] * f(b[0]) - b[0] * f(a[0]))/(f(b[0]) - f(a[0]));$

$n := 0;$

Repeat in parallel

$n := n + 1;$

if $f(c[n-1]) < 0$ then begin

$a[n] := c[n-1];$

$b[n] := b[n-1];$

$c[n] := (a[n] * f(b[n]) - b[n] * f(a[n]))/(f(b[n]) - f(a[n]))$

end

else if $f(c[n-1]) > 0$ then

begin

$a[n] := a[n-1];$

$b[n] := c[n-1];$

$c[n] := (a[n] * f(b[n]) - b[n] * f(a[n]))/(f(b[n]) - f(a[n]))$

end

Until $f(c[n-1]) = 0$;

Unfortunately, this algorithm does not bring an important improvement in speed-up, because of the time of communication between processors. But we may think the secant method directly in parallel, as follows.

We imagine an SIMD computer with r processors (see Chiorean [1]). Starting with approximations $x_{0,1}; x_{0,2}; \ldots, x_{0,r}$ of z, it is requires to determine r improuved approximations

$$x_{k+1,i} = \phi_{k,i}(x_{k,1}; x_{k-1,1}; \ldots, x_{0,r})$$

at every iteration step. Here,

$$\phi_{k,i} : \mathbb{R}^{(k+1)r} \to \mathbb{R}^r, \quad k \geq 0.$$

According to Corliss [3], the iteration series $x_{k,i}$ remains close to the zero z if the starting approximation is suitable, and that it will thus finally converge to the zero.

Taking into account all this, and considering an SIMD parallel computer with $r = 3$ processors, the serie $x_{k,i}$, $i = 1, 2, 3$ for the secant parallel method is generated in the following way:

$$x_{k+1,1} = x_{k,1} - \frac{x_{k,1} - x_{k,2}}{f(x_{k,1}) - f(x_{k,2})} f(x_{k,1})$$

$$x_{k+1,2} = x_{k,2} - \frac{x_{k,2} - x_{k,3}}{f(x_{k,2}) - f(x_{k,3)}} f(x_{k,2})$$

$$x_{k+1,3} = x_{k,3} - \frac{x_{k,3} - x_{k,1}}{f(x_{k,3}) - f(x_{k,1})} f(x_{k,3}),$$

where $x_{k,i}$ are those in the Fig.2.

It can be proved that the order of convergence for this parallel secant method is 2, compared with 1,618... for the serial secant method.

References

[1] Chiorean, I., *Calcul paralel. Fundamente*, Ed. Microinformatica, 1998.
[2] Coman, Gh., *Analiză mumerică*, Ed. Libris, Cluj, 1995.
[3] Corliss, G.F., *Parallel root finding algorithms*, Ph.D. Dep. of Mathematics, Michigan State Univ., 1974.
[4] Mateescu, G.D., Mateescu, I.C., *Analiză numerică, Proiect de manual pentru clasa a XII-a*, Ed. Petrion, 1996.
[5] Miranker, W.L., *Parallel search methods for solving equations*, Math. Comp. Simul., 20, 2(1978), pp.93-101.

Babeş-Bolyai University, str. Kogălniceanu nr. 1, 3400 Cluj-Napoca, Romania