SOME PARALLEL PROCEDURES FOR COMPUTING THE EIGENVALUES OF A REAL SYMMETRIC MATRIX

by
Ioan Dzițac, Simona Dzițac, Horea Oros

Abstract. The determination of the eigenvalues (also known as characteristic roots, proper values, or latent roots) of a matrix is extremely important in physics and engineering, where it is equivalent to matrix diagonalization and arises in such common applications as stability analysis, the physics of rotating bodies, and small oscillations of vibrating systems etc. In this paper we present some parallel procedure for numerical computing of the eigenvalues of a real symmetric matrix (in this case every characteristic roots are real). For solution of the characteristic equation in this particular conditions let be used asynchronous parallel implementation of the simplified Newton's method and some parallel procedures based of RFAIM [1] and particularized in [2] and [3].

Keywords: parallel processing, parallel implementation, eigenvalues, RFAIM method.

1. Introduction

Let

$$\textbf{(1)} \qquad pol[n](\lambda) = \sum_{i=0}^{n} p[i] * \lambda^{n-i} = 0,\ p[0] = 1 \ and\ \ p[n] \neq 0,$$

be the normalized characteristic equation associated of a real symmetric matrix A (nxn- type).

In general case, we know that equation (1) has n real roots, which we can isolate as in [3] and may be computed as in [1] and [2].

For example, the quadratic symmetric matrix $A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ has the characteristic equation $\begin{vmatrix} 1-\lambda & 0 \\ 0 & 2-\lambda \end{vmatrix} = \lambda^2 - 3\lambda + 2 = 0$ and two real eigenvalues $\lambda_1 = 1,\ \lambda_2 = 2.$

In this paper we present a parallel implementation of the simplified Newton procedure and a method based on the recurrent relation of Newton and on the RFAIM algorithm presented in [1].

2. Parallel implementation of simplified Newton's method

If the number of real positive roots of equation (1) is *poz*, then the number of negative roots is $neg = n - poz$. Let's assume that we have at our disposal a computing system, composed of $p + q$ processors. We will use p processors to find the positive roots and q processors to find the negative roots, proportionally with the numbers *poz* and *neg*. We call that procedure *news_par*.

In the process of finding the negative roots, the *news* processes are activated with an initial value equal to the right margin of the interval and are stopped when the number of found positive roots *nradp* is equal to *poz*. For finding the negative roots, the *news* processes are activated with an initial value equal to the left margin of the interval and are stopped when the number of found positive roots *nradn* is equal to *neg*. These choices are made to guarantee the convergence of the method.

The main program presented in Table 1 activates $p + q$ processors in parallel (see instruction *cobegin* and *coend*), and those processors will operate in parallel asynchronously.

Table 1 Equation having only real roots (procedure news_par)

```
procedure news_par(n,pol[n],eps);
select eps;
begin
              a:=rinf(n,pol[n],eps);
              b:=rsup(n,pol[n],eps);
                  call vars(vs);
               poz:=vars(a)-vars(b);
                    neg:=n-poz;
cobegin
for k=1 to p do in parallel asynchronous
    a[k]:=a+(k-1)*(b-a)/p;
    b[k]:=a+k*(b-a)/p;
    radp:=news(n,pol[n],a[k],b[k],b[k],eps);
    if (nradp=poz) then exit;
endfor;
for k=1 to q do in parallel asynchronous
    a[k]:=a+(k-1)*(b-a)/q;
    b[k]:=a+k*(b-a)/q;
    radn:=news(n,pol[n],-b[k],-a[k],-b[k],eps);
```

```
        if (nradn=neg) then exit;
    endfor;
    coend;
    end.
```

3. Using Newton's relation of recurrence

In such situation the polynomial algebraic equation (1) is equivalent with a system of algebraic non-linear equations by means of Newton's relation of recurrence:

(2)
$$\begin{aligned}&\lambda_1+\lambda_2+\ldots+\lambda_n=b_1\\&(\lambda_1)^2+(\lambda_2)^2+\ldots+(\lambda_n)^2=b_2\\&\ldots\\&(\lambda_1)^n+(\lambda_2)^n+\ldots+(\lambda_n)^n=b_n\end{aligned}$$

We are able to determine b_i using recurrence from the coefficients of the equation (1).

The resolution of system (2) can be done using RFAIM algorithm presented in [1] with some specific customization.

Using Newton's relation of recurrence:

(3)
$$S_i=(\lambda_1)^i+(\lambda_2)^i+\ldots+(\lambda_n)^i=b_i$$

we are able to form the partial series (parallel-synchronous):

$$\lambda_1^{k+1}=f_1(\lambda^k)=\lambda_1^k+b_1-S_1^k$$
$$\lambda_{2i+1}^{k+1}=f_i(\lambda^k)=\sqrt[2i+1]{b_{2i+1}+\lambda_{2i+1}^k-S_{2i+1}^k}\ ,$$

for the components with odd indices,

$$\lambda_{2i}^{k+1}=f_{2i}(\lambda^k)=s\ \sqrt[2i]{\left|\ b_{2i}+\lambda_{2i}^k-S_{2i}^k\ \right|}\ ,$$

for the components with even indices,

where: $$s = \begin{cases} 1, & if \quad \lambda_{2i} \ is \ positive \\ -1, & if \quad \lambda_{2i}, \ is \ negative \end{cases}.$$

If the number of positive roots of (1) is *poz*, then the number of negative roots is *neg* = *n* - *poz*.

In **Table 2** we present a procedure for the resolution of the number of real positive and negative roots, by means of the procedure for sign variation from Sturm's sequence (procedure `vars(vs)`).

Table 2 Procedure for the resolution of the number of positive and negative roots

```
procedure nradr(n,pol[n],eps);
select eps;
begin
              a:=rinf(n,pol[n],eps);
              b:=rsup(n,pol[n],eps);
                  call vars(vs);
              poz:=vars(a)-vars(b);
                   neg:=n-poz;
end.
```

We assume that we have a computing system composed of at least *n = poz + neg* processors (otherwise we redistribute proportionally the computing tasks).

For symmetry, we may assume that:

- positive roots: *rootp[i], i = 1…poz*
- negative roots: *rootn[i], j = neg…n*

We use two RFAIM [1] parallel type procedures which communicate and exchange data, through the memory of the host processor P, while executing asynchronously. The master processor P manages the execution of the RFAIM_POZ and RFAIM_NEG processes in parallel and asynchronously, those processes also being parallel.

Processor P receives the first subvectors formed by the positive and negative components respectively, and combines those subvectors to form the vector of current approximation at each step of asynchronous iteration.

We will denote with *ex* the expression whose square root is taken.

Table 3 Procedure for the resolution of positive roots

```
      procedure RFAIM_POZ
      receive previous approach vector pav from processor
P;
      begin
      for i=1 to poz do in parallel
                        if ex(pav[i])>0
                 then rootp[i]:=rad[i](ex(pavv[i]))
                   else rootp[i]:=rand(rinf,rsup);
                             endelse;
                            endif;
                  send rootp[i] to processor P;
      endfor;
      wait mesaj from processor P;
      end.
```

Table 4 Procedure for the resolution of the negative roots

```
      procedure RFAIM_NEG
      receive previous approach vector pav from processor
P;
      begin
      for j=poz to n do in parallel
                      if (j este par) then
                        if ex(pavv[j])> 0
                 then rootn[i]:=-rad[i](ex(pav[j]))
                  else rootn[j]:=rand(-rsup,rinf);
                              endelse;
                             endif;
                else rootn[j]:=rad[j](ex(pav[j]));
                            endelse;
                            endif;
                  send rootp[j] to processor P;
      endfor;
      wait message from processor P;
      end.
```

REFERENCES:

1. Ioan Dziţac**,** *Random-filtered asynchronous iterative method,* Bul. St. Univ. Baia Mare, Seria B, Mat. Inf., Vol. XVI(2000), Nr. 1, p. 17-24.
2. Ioan Dziţac, Grigor Moldovan, Horea Oros*, A Parallel Procedure for the Exhaustive Numerical Resolution of the Polynomial Equations*, Proceedings of The 11th Conference on Applied and Industrial Mathematics, Oradea, Edit. Univ. of Oradea, 2003, vol. I, p. 94-96.
3. Ioan Dziţac, Simona Dziţac, Mădălina Văleanu, *A Parallel Algorithm for the Real Roots Isolation of the Polynomial Equation*, Proceedings of The 11th Conference on Applied and Industrial Mathematics, Oradea, Edit. Univ. din Oradea, 2003, vol. I, p. 97-99.

Autors:

Ioan Dziţac, Simona Dziţac, Horea Oros - University of Oradea, Romania, email: idzitac@uoradea.ro