PARALLELIZED ALGORITHMS FOR THE EIGENVALUE PROBLEM

Rinela Kapçiu[1], Eglantina Kalluçi[2]

[1]Faculty of Information Technology "Aleksander Moisiu" University of Durres, Computer Science Department. Durres. Albania E mail: topalli_r78@hotmail.com
[2]University of Tirana, Faculty of Natural Sciences, Department of Applied Mathematics. Durres. Albania E mail: eglaxhaja@yahoo.com

Abstract

Matrix methods are very popular for polynomial root-finding. The idea is to approximate the zeros a given polynomial as the eigenvalues of the companion or generalized companion matrix of the given polynomial. In this paper we give some new results related to a parallel algorithm given by Bini and Gemignani. The algorithm is a reformulation of Householder's sequential algorithm that is based in computation of the polynomial remainder sequence generated by the Euclidean scheme. As a comparison method we use a generalized version of the bisection method, which also is considered in both versions sequential and parallel ones. The parallel implementation is done in an asynchronous cluster.

Key words: *eigenvalue problem, matrix, parallel programming, algorithm.*

Introduction

The problem of finding the zeros of a polynomial in a parallel model of computation has been of main interest in these last years.

Finding the eigenvalues and eigenvectors of matrices is a classic problem in linear algebra, and computationally efficient solutions are enormously important.

Ben et al [2] have presented a fast parallel algorithm based on some properties of the Euclidean algorithm and on the divide-and-conquer strategy.

Bini & Gemignani [3] have redefined Householder's sequential algorithm in terms of a parallel algorithm that works on tri diagonal matrices.

Monte Carlo methods give statistical estimates for the functional of the solution by performing random sampling of a certain variable whose expectation is the desired functional.

Let J be any functional that we estimate by Monte Carlo method; θ_N be the estimator, where N is the numbers of trials. The probable error is r_N for which $P_r\{|J-\theta_N| \geq r_N\} = \frac{1}{2} = P_r\{|J-\theta_N| \leq r_N\}$.

In this paper we have presented modification of Monte – Carlo algorithms for evaluating the spectral radius of large sparse matrices and we have tested the parallel implementation of these methods using MPI.

2. The power method

Let suppose that $A \in R^{n\times n}$ is diagonalizable, $X^{-1}AX = diag(\lambda_1,\ldots,\lambda_n)$, $X = (x_1,\ldots,x_n)$ and $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Given $f^{(0)} \in C^n$, the Power method [6] produces a sequence of vectors $f^{(k)}$ as follows:

$$z^{(k)} = Af^{(k-1)},$$
$$f^{(k)} = \frac{z^{(k)}}{\left\|z^{(k)}\right\|_2},$$
$$\lambda^{(k)} = \left[\, f^{(k)} \right]^H Af^{(k)}, \quad k = 1,2,\ldots$$

Except for special starting points, the iterations converge to an eigenvector corresponding to the eigenvalue of *A* with largest magnitude (dominant eigenvalue) with rate of convergence:

$$\left|\lambda_1 - \lambda^{(k)}\right| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right).$$

Consider the case when we want to compute the smallest eigenvalue. The iteration matrix A is replaced by B, where A and B have the same eigenvectors, but different eigenvalues. Letting σ denote a scalar, then the three common choices for B are: $B = A - \sigma I$ which is called the shifted Power method, $B = A^{-1}$ which is called the inverse Power method and $B = (A - \sigma I)^{-1}$ which is called the inverse shifted Power method.

Table 1. Relationship between eigenvalues of A and B.

B	Eigenvalue of B	Eigenvalue of A
A^{-1}	$1/\lambda_A$	$1/\lambda_B$
$A-\sigma I$	$\lambda_A-\sigma$	$\lambda_B+\sigma$
$(A-\sigma I)^{-1}$	$1/(\lambda_A-\sigma)$	$\sigma+(1/\lambda_B)$

Having *k* iterations, the number of arithmetic operations in Power method is $O(4kn^2+3kn)$, so the power method is not suitable for large sparse matrices.

3. The improved algorithm and numerical results

In this paper we present Monte Carlo algorithms for evaluating the dominant eigenvalue of large real sparse matrices and their implementation on a cluster of workstations using MPI.

Let's consider the matrix $A=\{a_{ij}\}_{i,j=1}^{n}, A\in R^{n\times n}$ and the vectors $f=(f_1,\cdots f_n)^t\in R^{n\times 1}$, $f=(h_1,\cdots h_n)^t\in R^{n\times 1}$. The algebraic transformation $Af\in R^{n\times 1}$ is called a Monte Carlo iteration.

The dominant eigenvalue based on Direct Monte Carlo method is obtained using the following iteration process:

$$\lambda_{max}=lim_{i\to\infty}\frac{(h,A^i f)}{(h,A^{i-1}f)}.$$

In order to find the smallest by modulus eigenvalue of A we can use the Inverse Monte Carlo algorithm, which can be implemented in two ways, where the most effective is calculating the inversion of the matrix A and then applying Direct Monte Carlo algorithm using the iterations with the inverse matrix.

These algorithms use the idea of the Power method combined with Monte Carlo iterations by the given matrix, the resolvent matrix and the inverse matrix respectively.

The power method produces a sequence of vectors. The iterations converge to an eigenvector corresponding to the eigenvalue of A with largest magnitude (dominant eigenvalue).

Consider the case when we want to compute the smallest eigenvalue. The iteration matrix A is replaced by B, where A and B have the same eigenvectors, but different eigenvalues.

Now consider an algorithm based on Monte Carlo iterations by the resolvent matrix $R_q = [I - qA]^{-1} \in R^{n \times n}$. The following presentation holds

$$[I - qA]^{-m} = \sum_{i=0}^{\infty} q^i C_{m+i-1}^i A^i, \quad |qA| < 1.$$

Having in mind that

$$\left([I - qA]^{-m} \ f, h \right) = E \left\{ \sum_{i=0}^{\infty} q^i C_{m+i-1}^i \left(A^i f, h \right) \right\}$$

we have the following Monte Carlo algorithm:

$$\lambda_{\min} \approx \frac{1}{q} \left(1 - \frac{1}{\mu^{(m)}} \right) = \frac{\left(A[I - qA]^{-m} \ f, h \right)}{\left([I - qA]^{-m} \ f, h \right)}$$

We note that if $q > 0$ the algorithm evaluates $\lambda_{\max}$, if $q < 0$ the algorithm evaluates $\lambda_{\min}$ without matrix inversion.

4. Numerical tests

The numerical tests are made on a cluster with 10 processors workstation under MPI. Each processor executes the same program for *N/p* number of trajectories, i. e. it computes *N/p* independent realizations of the random variable (where *p* is the number of processors). At the end the host processor collects the results of all realizations and computes the desired value. The computational time does not include the time for initial loading of the matrix because we consider our problem as a part of bigger problem and suppose that every processor constructs it.th results for average time (T) and efficiency (E) are given in Table 1, with relative accuracy10^{-3}.

Table 2. Implementation of **Direct Monte Carlo Algorithm** using MPI.

Matrix dim	**1 pr. T**	**2 pr. T**	**2 pr. E**	**3 pr. T**	**3 pr. E**	**4 pr. T**	**4 pr. E**	**5 pr. T**	**5 pr. E**
n=128	34	17	1	11	1.03	8	1.06	7	0.97

n=1024	111	56	0.99	37	1	27	1.003	21	1.06
n=2000	167	83	1	56	1	42	1	35	0.96

Table 3. Implementation of Improved Algorithm using MPI.

Matrix dim	**1 pr. T**	**2 pr. T**	**2 pr. E**	**3 pr. T**	**3 pr. E**	**4 pr. T**	**4 pr. E**	**5 pr. T**	**5 pr. E**
n=128	18	9	1	6	1	4	1.1	3	1.2
n=1024	30	15	1	10	1	7	1.06	6	1
n=2000	21	11	0.99	7	1	5	1.04	4	1.04

5. Conclusion

Parallel Monte Carlo algorithm for calculating the eigenvalues are presented and studied. They can be applied for well balanced matrices (which have nearly equal sums of elements per row) in order to provide good accuracy.

We propose to use them when one have to calculate the dominant eigenvalue of very large sparse matrices since the computational time is almost independent of the dimensions of the matrix and their parallel efficiency is super linear.

References

[1] F. L. Lucio On the convergence of a parallel algorithm for finding polynomial zeros, 1997.

[2] M. Ben et al, A fast parallel algorithm for determining all roots of a polynomial with real roots, SIAM J. Comp. 11, 1988.

[3] D. Bini, L. Germignani. On the complexity of polynomial zeros, SIAM, J. Comp. 21, 1992.

[4] T. Auckenthaler et al, Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations, Parallel Comp. 2011.

[5] E. R. Alcalde, Parallel implementation of Davidson-type methods for large-scale eigenvalue problem, 2012.

[6] G. H. Golub, Ch. Van. Loon, Matrix Computations. The Johns Hopkins Univ. Press, Baltimore and London, 1996.