Research Article

Partitioning Inverse Monte Carlo Iterative Algorithm for Finding the Three Smallest Eigenpairs of Generalized Eigenvalue Problem

Behrouz Fathi Vajargah and Farshid Mehrdoust

Department of Statistics, Faculty of Mathematical Sciences, University of Guilan, P.O. Box 1914, Rasht, Iran

Correspondence should be addressed to Behrouz Fathi Vajargah, fmehrdoust@guilan.ac.ir

Received 5 December 2010; Accepted 15 February 2011

Academic Editor: Michele Benzi

Copyright © 2011 B. Fathi Vajargah and F. Mehrdoust. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new Monte Carlo approach for evaluating the generalized eigenpair of real symmetric matrices will be proposed. Algorithm for the three smallest eigenpairs based on the partitioning inverse Monte Carlo iterative (IMCI) method will be considered.

1. Introduction

It is well known that the problem of calculating the largest or smallest generalized eigenvalue problem is one of the most important problems in science and engineering [1, 2]. This problem arises naturally in many applications. Mathematically, it is a generalization of the symmetric eigenvalue problem, and it can be reduced to an equivalent symmetric eigenvalue problem. Let $A, B \in \mathbb{R}^{n \times n}$ be real symmetric matrices and the matrix $B$ a positive definite matrix. Consider the problem of evaluating the eigenvalues of the pencil $(A, B)$, that is, the values for which

$$Ax = \lambda Bx.$$  \hspace{1cm} (1.1)

A generalized eigenvalue problem (1.1) is said to be symmetric positive definite (S/PD) if $A$ is symmetric and $B$ is positive definite.
2. Inverse Vector Iteration Method

Another procedure for eigenvalue prediction is to use the Rayleigh quotient given by [3]

\[
\lambda = \mu(x) = \frac{x^T Ax}{x^T Bx} \tag{2.1}
\]

since \( B \) is positive definite, then (2.1) is well defined.

**Theorem 2.1.** Suppose that \( \lambda_{\min} = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_{n-1} < \lambda_n = \lambda_{\max} \) are \( n \) eigenvalues for pencil \((A, B)\) and \( v_1, \ldots, v_n \) corresponding eigenvectors. Then for arbitrary initial vector \( x \) one has [3]

\[
\lambda_1 < \mu(x) < \lambda_n \tag{2.2}
\]

where \( \mu(x) \) is as introduced in (2.1).

**Theorem 2.2.** The inverse vector iteration method for arbitrary choice vector \( x_0 \) is convergent to the smallest eigenvalue and corresponding eigenvector for pencil \((A, B)\). Also, the rate of convergence depends on \( O(\lambda_2/\lambda_1)^k \), where \( k \) is number of iterations [3].

Algorithm 1 evaluates the smallest eigenpair based on the inverse vector iteration [4].
3. **Monte Carlo Method for Matrix Computations**

Suppose that the matrix $A \in \mathbb{R}^{n \times n}$ and two vectors $f, h \in \mathbb{R}^n$ are given. Consider the following Markov chain $T_i$ with length $i$:

$$T_i : k_0 \rightarrow k_1 \cdots \rightarrow k_i,$$

where for $j = 1, \ldots, i$, $k_j \in \{1, 2, \ldots, n\}$. The statistical nature of constructing the chain (3.1) follows as

$$p(k_0 = \alpha) = p_{\alpha}, \quad p(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha \beta},$$

where $p_{\alpha}$ and $p_{\alpha \beta}$ show the probability of starting chain at $\alpha$ and transition probability from state $\alpha$ to $\beta$, respectively.

In fact

$$\sum_{\alpha=1}^{n} p_{\alpha} = 1, \sum_{\beta=1}^{n} p_{\alpha \beta} = 1, \quad \alpha = 1, 2, \ldots, n.$$  \hspace{1cm} (3.3)

Define the random variable $W_j$ using the following recursion for

$$W_0 = 1, \quad W_j = W_{j-1} \frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \quad j = 1, 2, \ldots, i.$$  \hspace{1cm} (3.4)

Now, define the following random variable:

$$\Theta[h] = \frac{h_{k_i}}{p_{k_i}} \sum_{j=0}^{\infty} W_j f_{k_j}.$$  \hspace{1cm} (3.5)

**Theorem 3.1.** Consider the following system:

$$Ax = b.$$  \hspace{1cm} (3.6)

Let the nonsingular matrix $M \in \mathbb{R}^n$, such that $MA = I - L$, then the system (3.6) can be presented in the following form:

$$x = Lx + f,$$  \hspace{1cm} (3.7)

where $f = Mb$. Then under condition $\max_i \sum_{j=1}^{n} |l_{ij}| < 1$, one has [5]

$$E\{\Theta[h]\} = \langle h, x \rangle.$$  \hspace{1cm} (3.8)
Suppose that $x^{(i)}$ is the $i$th iterative solution of the following recursion relation with $x^{(0)} = f$. If we set the random variable

$$
\Theta_i[h] = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{i} W_j f_{k_j},
$$

then

$$
E[\Theta_i[h]] = \langle h, x^{(i+1)} \rangle.
$$

By simulating $N$ random paths with length $i$

$$
\tau_i^{(s)}: k_0^{(s)} \rightarrow k_1^{(s)} \rightarrow \cdots \rightarrow k_i^{(s)}, \quad s = 1, 2, \ldots, N,
$$

we can find

$$
\Theta_i^{(s)}(h) = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{i} W_j^{(s)} f_{k_j}, \quad s = 1, \ldots, N.
$$

The Monte Carlo estimation can be evaluated by

$$
\Theta_i = \frac{1}{N} \sum_{s=1}^{N} \Theta_i^{(s)}(h)
$$

which is an approximation of $\langle h, x^{(i+1)} \rangle$.

From all possible permissible densities, we apply the following:

$$
p_{\alpha} = \frac{|h_{\alpha}|}{\sum_{\alpha=1}^{n} |h_{\alpha}|},
$$

$$
p_{\alpha \beta} = \frac{|a_{\alpha \beta}|}{\sum_{\beta=1}^{n} |a_{\alpha \beta}|}, \quad \alpha = 1, 2, \ldots, n.
$$

The choice of the initial density vector and the transition probability matrix leads to an almost Optimal Monte Carlo (MAO) algorithm.

**Theorem 3.2.** Using the above choice $p = \{p_{\alpha}\}_{\alpha=1}^{n}$ and $P = \{p_{\alpha \beta}\}_{\alpha, \beta=1}^{n}$, the variance of the unbiased estimator for obtaining the inverse matrix is minimized [4].

There is a global algorithm that evaluates the solution of system (3.6) for every matrix $A$. The complexity of algorithm is $O(n^2lN)$, where $l$ and $N$ are the average length of Markov chain and the number of simulated paths, respectively [2].
4. Inverse Monte Carlo Iterative Algorithm (IMCI)

Inverse Monte Carlo iterative algorithm can be applied when $A$ is a nonsingular matrix. In this method, we calculate the following functional in each steps:

$$\frac{\langle Af_j, h_i \rangle}{\langle Bf_j, h_i \rangle} = \frac{\langle Bf_{j-1}, h_i \rangle}{\langle Bf_j, h_i \rangle}.$$  \hfill (4.1)

It is more efficient that we first evaluate the inverse matrix using the Monte Carlo algorithm [1, 2, 4]. The algorithm can be realized as in Algorithm 2.

5. Partitioning IMCI

Let the matrix $A$ be partitioned into four blocks $A_1$, $A_2$, $A_3$, and $A_4$, where $A_1$ and $A_4$ are square matrices of order $p$ and $q$ such that $p + q = n$:

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}.$$  \hfill (5.1)

By assumption that all the indicated matrix inversions are realized, it is easy to verify that

$$A^{-1} = \begin{pmatrix} B & L \\ M & N \end{pmatrix}.$$  \hfill (5.2)
Partitioning inverse $(S,n)$

begin:

\[ n = \text{rank}(S); \quad p = n/2 \]
\[ A = S[1:p,1:p]; \quad B = S[1:p,p+1:n] \]
\[ C = S[p+1:n,1:p]; \quad D = S[p+1:n,p+1:n] \]
\[ m = \text{size}(A) \]

if \( m \leq \text{threshold} \)

\[ AA = \text{Monte Carlo procedure } (A) \]

else begin:

\[ AA = \text{Partitioning inverse } (A,m) \]
\[ N = \text{Partitioning inverse } (D - C \ast AA \ast B) \]
\[ M = -N \ast C \ast AA; \quad L = -AA \ast B \ast N \]
\[ K = AA - AA \ast B \ast M \]
\[ SS[1:p,1:p] = K; \quad SS[1:p,p+1:n] = L \]
\[ SS[p+1:n,1:p] = M; \quad SS[p+1:n,p+1:n] = N \]

end

end

Algorithm 3

where

\[ N = \left( A_4 - A_3A_1^{-1}A_2 \right)^{-1}, \quad M = -N A_3A_1^{-1}, \]
\[ L = -A_1^{-1}A_2N, \quad K = A_1^{-1} - A_1^{-1}A_2M. \] (5.3)

Thus inverting a matrix of order \( n \) comes down to inverting four matrices, of which two have order \( p \) and two have order \( q \), and several matrix multiplications. Therefore the basic Monte Carlo for solving \( Af_j = Bf_{j-1} \) will be called as the dimension of matrix \( A \) and equals to \( \text{threshold} \). This action causes the convergence acceleration. Now, we can use the following recursion algorithm to obtain the inverse of matrix \( A \) (see Algorithm 3).

6. Finding More Than One Generalized Eigenvalues

Assume that an eigenvalue \( \lambda_1 \) and its corresponding eigenvector \( v_1 \) have been computed using the partitioning IMCI algorithm. In the first step of the above algorithm, we deflate the matrix \( A \) to the matrix \( B \). Then, we repeat again the first step of the algorithm to obtain the dominant eigenvalue of \( B \) which is the second dominant eigenvalue of \( A \). Let \( p \) values of eigenvalues of pencil \((A,B)\) be computed. Suppose that \( V_p \) is a matrix such that the columns of \( V_p \) are \( p \) vector of eigenvector of pencil \((A,B)\), that is,

\[ V_p = [v_1, \ldots, v_p], \] (6.1)

where \( v_i \) is eigenvector corresponding eigenvale \( \lambda_i, \quad i = 1,2, \ldots, p. \)
Table 1: Number of chains $N = 80$.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$\lambda_3$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$64 \times 64$</td>
<td>$1.26 \times 10^{-7}$</td>
<td>$4.5 \times 10^{-7}$</td>
<td>$1.26 \times 10^{-6}$</td>
<td>$8.36 \times 10^{-4}$</td>
<td>.002813</td>
<td>.008154</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>$1.18 \times 10^{-7}$</td>
<td>$4.46 \times 10^{-7}$</td>
<td>$1.33 \times 10^{-6}$</td>
<td>$6.33 \times 10^{-4}$</td>
<td>.002534</td>
<td>.008074</td>
</tr>
<tr>
<td>$256 \times 256$</td>
<td>$6.48 \times 10^{-8}$</td>
<td>$4.37 \times 10^{-7}$</td>
<td>$1.51 \times 10^{-6}$</td>
<td>$3.16 \times 10^{-4}$</td>
<td>.002151</td>
<td>.008070</td>
</tr>
<tr>
<td>$512 \times 512$</td>
<td>$7.91 \times 10^{-8}$</td>
<td>$2.87 \times 10^{-7}$</td>
<td>$1.72 \times 10^{-6}$</td>
<td>$1.83 \times 10^{-4}$</td>
<td>.001284</td>
<td>.007759</td>
</tr>
<tr>
<td>$1024 \times 1024$</td>
<td>$7.04 \times 10^{-8}$</td>
<td>$1.99 \times 10^{-7}$</td>
<td>$1.79 \times 10^{-6}$</td>
<td>$1.06 \times 10^{-4}$</td>
<td>.000712</td>
<td>.00769</td>
</tr>
</tbody>
</table>

Figure 1: Computational times for general and partition methods.

Now, let

$$(A_p, B) = \left( A + (BV_p) \Lambda (BV_p)^T, B \right),$$

(6.2)

where

$$\Lambda = \text{diag} \{ \delta_i - \lambda_i \}, \quad \delta_i > \lambda_p, \ i = 1, \ldots, p.$$  \hspace{1cm} (6.3)

Hence, if we find the $p$th smallest eigenpair of pencil $(A, B)$, then we can evaluate $\lambda_{p+1}$, that is, $(p + 1)$th smallest eigenvalue of pencil $(A, B)$.

### 7. Numerical Results

In this section, the experimental results for obtaining the three smallest eigenpairs outlined in Tables 1, 2, and 3. The numerical tests are performed on Intel(R) Core(TM)2 CPU, 1.83 GHz personal machine.
Table 2: The solution when the number of chains increases.

<table>
<thead>
<tr>
<th>Number of chains ( N )</th>
<th>Calculated eigenvalues ( \lambda_1 )</th>
<th>( \lambda_2 )</th>
<th>( \lambda_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Exact ( \lambda_1^{(1)} = .74529395 )</td>
<td>Exact ( \lambda_2^{(1)} = .85537131 )</td>
<td>Exact ( \lambda_3^{(1)} = .91032740 )</td>
</tr>
<tr>
<td>40</td>
<td>.74737713</td>
<td>.85513704</td>
<td>.91008238</td>
</tr>
<tr>
<td>80</td>
<td>.74529979</td>
<td>.85537157</td>
<td>.91034304</td>
</tr>
</tbody>
</table>

Table 3: Total computational time for general and partitioning methods.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Time (Sec.) General method</th>
<th>Time (Sec.) Partitioning method</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 × 20</td>
<td>.21</td>
<td>.20</td>
</tr>
<tr>
<td>40 × 40</td>
<td>.40</td>
<td>.32</td>
</tr>
<tr>
<td>60 × 60</td>
<td>.71</td>
<td>.54</td>
</tr>
<tr>
<td>80 × 80</td>
<td>1.25</td>
<td>.60</td>
</tr>
<tr>
<td>100 × 100</td>
<td>1.83</td>
<td>.75</td>
</tr>
<tr>
<td>120 × 120</td>
<td>2.54</td>
<td>1.18</td>
</tr>
<tr>
<td>140 × 140</td>
<td>3.67</td>
<td>1.57</td>
</tr>
<tr>
<td>160 × 160</td>
<td>4.42</td>
<td>2.04</td>
</tr>
<tr>
<td>180 × 180</td>
<td>5.58</td>
<td>2.54</td>
</tr>
<tr>
<td>200 × 200</td>
<td>7.20</td>
<td>3.18</td>
</tr>
</tbody>
</table>

Figure 2: Regression function \( y = 0.0000658008x + 0.1048607485 \).

8. Conclusion and Future Study

We have seen that Monte Carlo algorithms can be used for finding more than one eigenpair of generalized eigenvalue problems. We analyze the computational complexity, speedup, and efficiency of the algorithm in the case of dealing with sparse matrices. Finally, a new method
for computing eigenpairs as the partitioned method is presented. In Figure 1 the comparison of computational times between general Monte Carlo algorithm and partitioning algorithm is shown. The scatter diagram in Figure 2, shows that there is a linear relationship between matrix dimension (equivalently, the number of matrix elements) and total computational time for partitioning IMCI.

References

Submit your manuscripts at http://www.hindawi.com