# Comparison of efficiency between Coordinate Descent method and Conjugate Gradient Descent+Inverse Iterative method

By

Lu Peian
( 1730005033)

A Final Year Project thesis (STAT4004)
submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Science (Honours)
in Statistics

at

BNU-HKBU
UNITED INTERNATIONAL COLLEGE

Dec, 2020

DECLARATION

I hereby declare that all the work done in this Project is of my independent effort. I also certify that I have never submitted the idea and product of this Project for academic or employment credits.

_____

Lu Peian
(1730005033)

Date:_____

Lu Peian

Science and Technology Division

# Abstract

Finding the smallest eigenvalue associated with eigenvector solves the minimization of $\frac{x^T Ax}{x^T x}$, and it is useful to construct rank 1 approximation for electronic structure calculations, or applied in principal component analysis. This paper shows efficiencies and accuracies mainly of two proposed methods to find the smallest eigenvector and corresponding eigenvalue when target matrix holds different properties. We designed our own method inspired from Coordinate Descent and compared the accuracy at the same level of calculation amount(number of iteration) with another combination of iterative and matrix-free method. By applying differen numerical simulations, we obtained advantages and disadvantages between these two methods according to matrix property

**Keywords**: Large computation demand, Coordinate Descent, Eigenvalue pair problem

# Acknowledgement

Although the most important thesis of the university has gone through many twists and turns, although topics changed several times, I finally achieved some achievements. This project made me realize the difficulty and difficulty of scientific research and theoretical innovation. Special thanks to Dennis teacher and my good friend Dylan who chose the same subject with me. Along the way, I adhered to the attitude of "no result is also a result" to complete this topic, trying best to achieved every given goal.It is their help and company, pushing me to complete this project.

This page left blank for purpose.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Introduction

### 1.1.1 Organization of this paper

This paper starts with problem formulation and notation declaration, followed by brief historical review of Conjugate Gradient(CG), short introduction of Inverse Iterative method(IIM) and Coordinate Descent(CD) method.In next section, we will demonstrate some mathematical conditions and detailed algorithms about three methods above, and then results of numerical experiments. The last part is conclusion of applicable scene and difficulty within CD compared with existed methods, which expands to discussion of future possibility to revise partly so as to obtain few smallest eigenvalues and associated vectors.

### 1.1.2 Problem Formulation

It is common circumstance that we need to minimize an equation form as

$$min : \frac{x^T Ax}{x^T x} \tag{1.1}$$

where A is positive-definite and symmetric matrix, x is column vector.
It can be shown that gradient of r(x) is:

$$dr = 2(x^T x)^{-1}(Ax - xr(x)) \tag{1.2}$$

**Theorem 1.1** *For a positive-definite symmetric matrix A of dimension n,*

$$\lambda_{min} \leq \frac{x^T Ax}{x^T x} \leq \lambda_{max} \tag{1.3}$$

Hence, obtaining the smallest eigenvalue is equivalent to minimize the (1)

**Corollary 1.1** *For a positive-definite symmetric matrix A of dimension n, its smallest eigenvalue equals to the largest eigenvalue of $A^{-1}$ . Similarly, its largest eigenvalue equals to the smallest eigenvalue of $A^{-1}$.*

### 1.1.3 Review of Conjugate Gradient(CG) method

The conjugate gradient method was proposed in the early 1950s. At high speed Developed digital computers and tried to lay the foundation for the available mathematical aspects of digital computer efficiency. Magnus Hestenes, Eduard Stiefel and Cornelius Lanczos participated in these equations in the seminar on linear numerical methods. This led to Lanczos and Hestenes and Stiefel discuss the conjugate gradient algorithm. This

method is attributed to the joint paper published by Hestenes and Stiefel in 1952, where they proposed the conjugate gradient method and the conjugate direction method including the conjugate Gram-Schmidt process. A closely related algorithm is proposed by Lanczos who is engaged in algorithm work to determine the eigenvalues of the matrix. His iterative algorithm generates similarity to transform the matrix into a tri-diagonal form, from which the eigenvalues can be obtained similar. The three recursive relations of the Lanczos program can be obtained by eliminating vectors from the conjugate direction algorithm scheme. Initially, the conjugate gradient algorithm was called the Hestenes-Stiefel-Lanczos method. Soon thereafter, vector computers and massive computer memory made it possible to use these methods to solve problems that could not be solved by any other means. Since then, the algorithm has been further improved, and has become a basic tool to solve various problems on various computer architectures. The conjugate gradient algorithm has also been extended to solve nonlinear equations and optimization problems, which are unconstrained and constrained optimization problems

### 1.1.4 Inverse Iterative(IM) method

Many problems in machine learning and statistics can be formulated as (generalized) eigenproblems. In terms of the associated optimization problem, computing linear eigenvectors amounts to finding critical points of a quadratic function subject to quadratic constraints. As inverse form of efficient power method, Inverse Iterative (also called Inverse Power) is widely applied to compute the smallest eigenvalues through calculating

the A$x^{k+1}$ to obtain x$^k$ , and thus subtly avoiding the difficulty of computing $A^{-1}$ . One of the drawback of Inverse Power method is that the convergence rate relies on ratio of the second smallest eigenvalue and the smallest eigenvalue. When the two smallest eigenvalues are close, or so called existence of multiple eigenvalues, it becomes inefficient. We cant trace the historical source of the Inverse Iterative method too much, only knowing that this method is accompanied by more and more problems in solving matrix eigenvalues. In China, this term was firstly approved and released by the National Scientific and Technical Terms Review Committee in 1933. In the later section,we will combine Conjugate Gradient method and Inverse Iterative method to find the smallest eigenvalue and associated eigenvector.

### 1.1.5 Coordinate-Descent(CD) method

CD method mainly solves optimization problems by successively performing approximate minimization along coordinate directions or coordinate hyperplanes.It obtains each iteration by fixing most components of the variable vector x from the current iteration, and minimizing the objective with respect to the remaining components.Genrally, the CD-method is similar to separated optimization in each iteration along a single axis and it is guaranteed to reach optimal point for convex problem. Each sub-problem is a minimization problem of lower dimensions (even scalar), hence it is usually easier to solve than the full problem. Most applications use block coordinate descent methods, which adjust groups of blocks of indices at each iteration, thus searching along a coordinate hyperplane rather than a single coordinate direction. Most derivation and analysis of

single-coordinate descent methods can be extended without great difficulty to the block-CD setting. In practical computation, its common to solve the equations in the form of Ax=b in each step of iteration, such as Newtons method and Gauss-Newtons method, when dealing with non-linear optimization problems. Furthermore, each calculation and storage of the entire matrix multiplication will make the algorithm inefficient and occupy a lot of storage space. Hence, under the setting of big data, coordinate descent method has become an important tool in the optimization toolbox for solving problems in machine learning and data analysis.For real problems(when the matrix is extremely large that it can only be downloaded part by part) , its worth sacrificing a certain degree of accuracy for availability of solution. However, it is developed by recent upsurge of machine learning with large amount of computation, which has brought this relatively unpopular technology back to peoples attention, and CD method has not yet found the best applied method. In this project, our goal is to borrow the inspiration of the greedy algorithm and choose the group with the largest gradients every time we define our compressed subproblem. Besides, we set the dimension of the subproblem be $\sqrt{n}$, so that the complexity of the entire algorithm is $O^3$

# Chapter 2

## 2.1 Conjugate Gradient(CG) method

### 2.1.1 Assumption and Notation

Targeted matrix A: symmetric and positive-definite, A $\in R^{nn}, x \in R^n$ and $b \in R^n$

$$A^T = \text{A}$$
$$x^T Ax > 0 \text{ for all x}$$

### 2.1.2 Problem Formulation

Consider the problem of finding x $\in R^n$ satisfying

$$Ax = b \tag{2.1}$$

The reason that CG method is widely interested is that solution to this problem is also a solution of the following optimization problem :

$$min : f(x) = \frac{1}{2}x^T Ax - b^T x \tag{2.2}$$

Equivalently, consider the point $x^*$ such that

$$df(x^*) = g(x^*) = Ax^* - b = 0 \tag{2.3}$$

### 2.1.3 Gram-Schmidt Conjugation

**Definition 2.1** *Conjugacy Two vectors x and y are conjugate with respect to matrix A,or A-orthogonal, if*

$$x^T Ay = 0$$

*Similarly, the set of directions* $d_1, \cdots, d_k$ *is called conjugate directions if and only if*

$$d_j^T Ad_i = 0$$

*for all i=1,*$\cdots$ *,k and* $i \neq j$

Given idea of Gram-Schmidt Conjugation process, we can easily generate a set of search directions that are A-orthogonal. Suppose we have a set of n linearly independent vectors $u_1, \cdots, u_n$. Set $d_1 = u_1$,and for i > 1, set

$$d_i = u_i + \sum_{i=1}^{i} \beta_{ik} d_k \tag{2.4}$$

where i >k for $\beta_{ik}$
by multiplying $Ad_j$ in (3)

$$d_i^T Ad_j = u_i^T Ad_j + \sum_{k=1}^{i} \beta_{ik} d_k^T Ad_j$$

since $d_i^T A d_j = 0$ for all i≠ j

$$0 = u_i^T A d_j + \beta_{ik} d_k^T A d_j, i>j$$

$$\beta_{ik} = -\frac{u_i^T A d_j}{d_j^T A d_j} \tag{2.5}$$

### 2.1.4 Algorithm of CG-method

**Algorithm 1:** Conjugate Gradient method

**Input**: a appropriate small $\epsilon$; iteration index k

**Output**: x of

Initialize vector $x_1 \in R^n$; Let $r_k$ be the gradient of $k^{th} iteration$, which means $r_k = Ax_k - b, d_k = -r_k$;

**while** ($|r_k|$ <0 or k < given number of iteration) **do**

$$\alpha_k = \frac{r_k^T r_k}{d_k^T A d_k} \tag{2.6}$$

$$x_{k+1} = x_k + \alpha_k d_k \tag{2.7}$$

$$r_{k+1} = r_k + \alpha_k A d_k \tag{2.8}$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{2.9}$$

$$d_{k+1} = -r_{k+1} + \beta_k d_k \tag{2.10}$$

Update $\alpha_k, x_k, r_k, \beta_k, d_k$;

**end**

Output x

# Chapter 3

## 3.1 Inverse Iterative method

### 3.1.1 Assumption and Notation

Given a matrix A $\in R^{nn}$, which is non-singular. Let eigenvalues of A denoted as

$$|\lambda_1| \leq \lambda_2| \leq |\lambda_n|>0$$

and its associated eigenvectors be $x_1, x_2, ..., x_n$. Hence, eigenvalues of $A^{-1}$ is

$$|\frac{1}{\lambda_n}| \leq \frac{1}{\lambda_{n-1}}| \leq \cdots |\frac{1}{\lambda_1}|>0$$

and its associated eigenvectors are $x_n, x_{n-1}, \cdots, x_2, x_1$.

### 3.1.2 Problem Formulation

$$min : \frac{x^T Ax}{x^T x}$$

Equivalently, finding the smallest eigenvalue and its associated eigenvector. Also, based on property of A, it can be solved by

$$max : \frac{x^T A^{-1} x}{x^T x} \tag{3.1}$$

Let $v_0$ be an arbitrary non-zero vector $\in R_n$, then it can be uniquely determined by

$$v_0 = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \tag{3.2}$$

Define:

$$v_k = A v_{k-1}, k = 1, 2, \cdots \tag{3.3}$$

then

$$\begin{aligned} v_k &= A v_{k=1} = A^2 v_{k-2} = \cdots = A^k v_0 \\ &= a_1 \lambda_1^T x_1 + a_2 \lambda_2^2 x_2 + \cdots + a_n \lambda_1^n x_n \end{aligned} \tag{3.4}$$

$$v_k = \lambda_1^k (a_1 x_1 + (\frac{\lambda_2}{\lambda_1})^k a_2 x_2 + \cdots + (\frac{\lambda_n}{\lambda_1})^k a_n x_n) \tag{3.5}$$

Hence, when $a_1 = 0$ and k $\to \infty$,

$$v_k = \lambda_1^k (a_1 x_1 + \epsilon_k),$$

,where $\epsilon_k = \sum j = 2^n a_j (\frac{\lambda_j}{\lambda_1})^k x_j$ since $\frac{\lambda_j}{\lambda_1} \leq 1$ for all j = 1 and then

$$\lim k \to \infty \frac{\lambda_j}{\lambda_1} = 0$$

Thus, we can obtain an approximation of eigenvector of $\lambda_1, v_k$ (only multiplying a constant). In our way, we directly substitute $v_k$ into x in $\frac{x^T A x}{x^T x}$ to compute $\lambda_1$. Similarly, if we set matrix A be $A^{-1}$, then applying above

procedure returns the eigenvector associated with the largest eigenvalue in $A^{-1}$, which is the smallest eigenvalue $\lambda_n$ in A. However, in order to avoid computing $A^{-1}$, II method comes up with a wise way. Recall (17) that

$$v_k = Av_{k-1}$$

Substituting $A^{-1}$ into A and times A at both sides yields that

$$Av_k = v_{k-1} \tag{3.6}$$

Hence, by successfully computing (19), we get the eigenvector associated with eigenvalue $\lambda_n$

### 3.1.3 Algorithm of II-method

**Algorithm 2:** Inverse Iterative method

**Input**: positive definite matrix A;iteration index k

**Output**: eigenvector associated with the smallest eigenvalue

Initialize vector $x_1 \in R^n$; $\nabla r_k$ (which means $r_k = Ax_k - b$);$d_k = -r_k$;

**while** $\frac{x^T Ax}{x^T x}$< given value **do**

  Use Conjugate Gradient method to solve following equation:

$$Ax_{k+1} = x_k \tag{3.7}$$

  Update $x_k$;

  k=k+1

**end**

Output final x

# Chapter 4

## 4.1 Coordinate Descent (CD) method

### 4.1.1 Assumption, Notation and Formulation

$$\arg\min_{x \in R^n} f(x_1, x_2, \cdot, x_n) \tag{4.1}$$

Hence, if we want to find the global solution, we need function f be convex. In our targeted problem,

$$f(x) = \frac{x^T A x}{x^T x}$$

which means A must also be positive-definite

### 4.1.2 Algorithm of CD-method

**Algorithm 3:** Coordinate Descent method

**Input**: positive definite matrix A;iteration index k

**Output**: eigenvector associated with the smallest eigenvalue

Initialize vector $x_1 \in R^n$; Set $y = Ax_0, r = \frac{x_0^T y}{x_0^T x_0}$,$g = 2\frac{y - x_0}{x_0^T x_0}$; **while** $\frac{x_k^T y}{x_k^T x_k} < |r|$ **do**

  **foreach** $j \in C^k$ **do**

$$x_j^{k+1} = \arg\min_{\alpha \in R^n} f(x^k + (\alpha - x_j^k)e_j) \tag{4.2}$$

  **end**

  Update $x_j^{k+1}$,y,r,g;

**end**

Output final x

### 4.1.3 CD method in matrix form

In matrix form of practical computation, we may need to find an index list s by sorting the gradient, and determine the number of selected coordinate, says $n'$

**Algorithm 4:** Coordinate Descent method in matrix form

**Input**: positive definite matrix A;iteration index k

**Output**: eigenvector associated with the smallest eigenvalue

Initialize vector $x_1 \in R^n$; Set $y = Ax_0, r = \frac{x_0^T y}{x_0^T x_0}, g = 2\frac{y - x_0}{x_0^T x_0}$

**while** $\frac{x_k^T y}{x_k^T x_k} < |r|$ **do**

  Set $np = \sqrt{n}$

  Select non-decreasing list **s** such that

  $|g(s(1))| > g(s(2)) > \cdot\, |g(s(n))|$

  Compute smaller matrix $A'$

| | | |
|---|---|---|
| $Matrix\ A' =$ | $A(s(1:n'-1), s(1:n'-1))$ | $A(s(1:n'-1), s(n':n)x_n)$ |
| | $B$ | $x_n^T(A(n':n, n':n))x_n$ |

  where B = $A'(1:n'-1, n')^T$ should be determined

  finally.(Notice that the complexity of step 3 is $O(n^{1.5})$)

  **foreach** $A'$ **do**

    Find the smallest eigenvalue of $A'$ and its associated eigenvector, says **xp**

  **end**

  update:

$$xp = xp\frac{||x(s(n':n))||_2}{xp(np)}$$

$$x(s(1:np-1)) = xp(1:np-1)$$

$$y = y + A(:, s(1:np-1)(xp(1:np-1))x(s(1:np-1)))$$

$$x(s(1:np-1)) = xp(1:np-1)$$

$$r = \frac{x^T y}{x^T x} \text{ and } g = 2\frac{y - xr}{x^T x}$$

**end**

output x

(For simplicity and clarity, here s(i:j) represents $s_{i:j} and x(s(i:j)) represents x(k)$, where $k \in s(i:j)$. Also A(:,i:j) represents the $i^{th}$ to $j^{th}$ columns of A)

# Chapter 5

## 5.1 Numerical experiment

### 5.1.1 Ways to generate matrix A

Generate matrix B by matlab code rand(n,n), satisfying entries in B follow N(0,1). Let $B' = B^T B$ so that $B'$ is positive-definite. Lastly, set A=$B'$ + B to guarantee A is symmetric

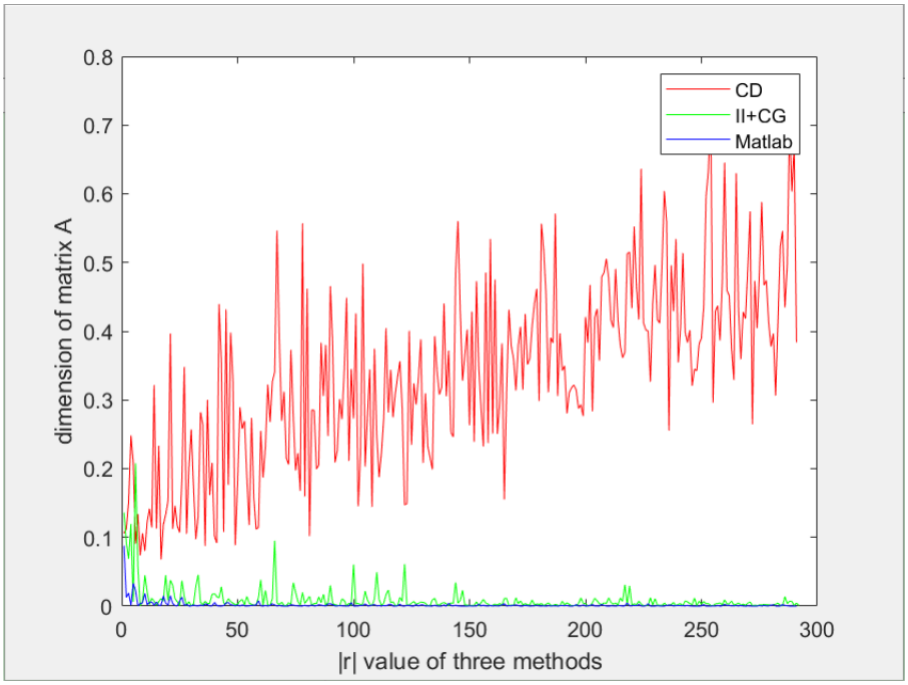


Figure 5.1: When A is generated by method (1)

(2):

Based on (1), we add a relatively large number $||A||_F$ to diagonal elements of A increase the ratio of

$$\frac{\lambda n - 1}{\lambda n}$$

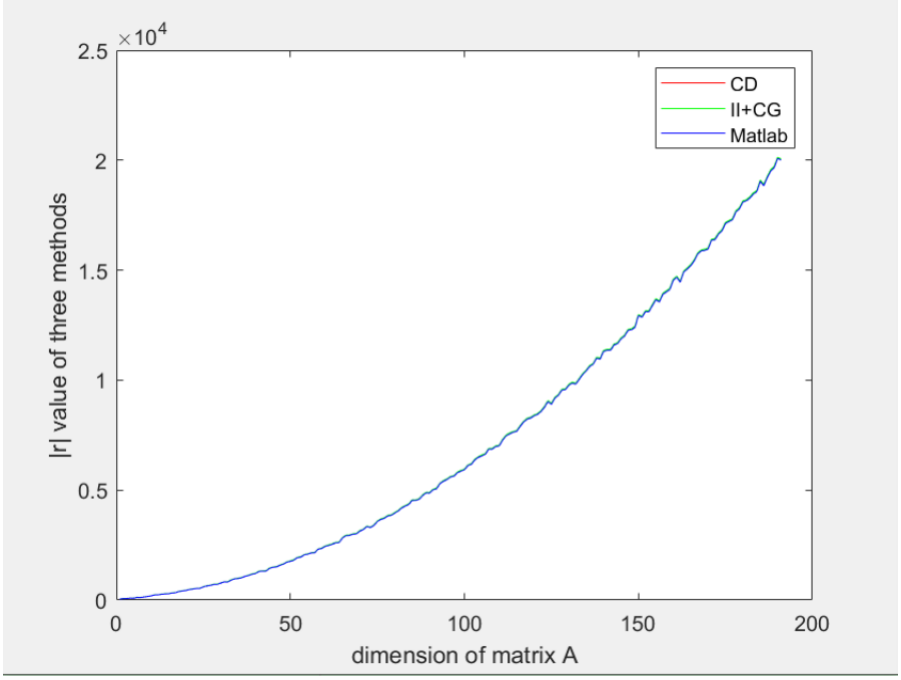


Figure 5.2: When A is generated by method (2)

(3): By eigendecomposition:

1.Generate a random matrix U with each element following N(0,1).

2.Geneate diagonal matrix D as given eigenvalues.

$$D = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1-\varepsilon & & \\ \vdots & & \ddots & \\ 0 & 0 & \dots & 1-(n-1)*\varepsilon \end{bmatrix}$$

3.Orthogonalizize matrix U

4.Let

$$A = UDU^T$$

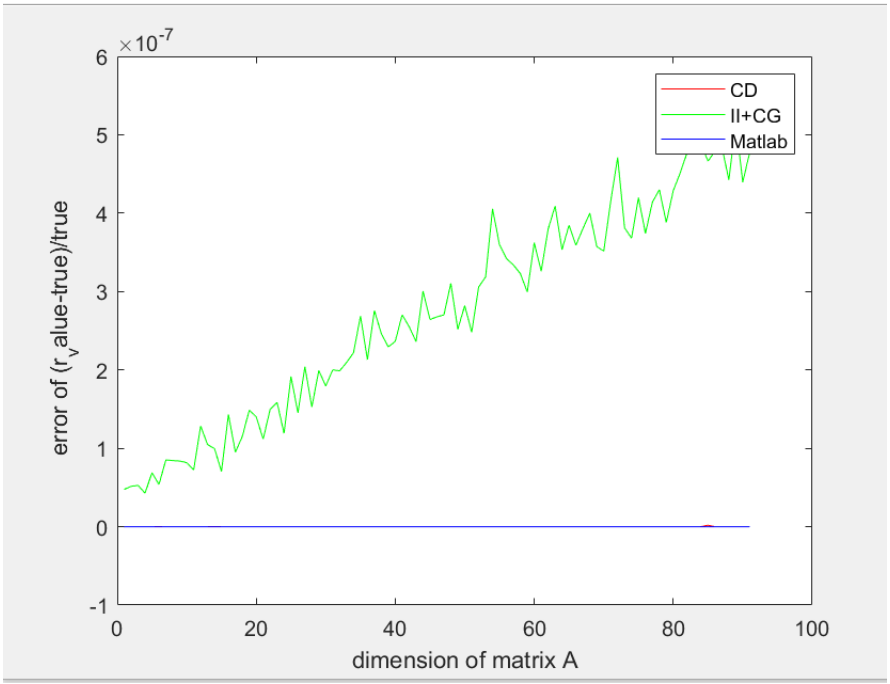


Figure 5.3: When A is generated by method (3) with $\epsilon = 10^{-8}$

As we can see from Figure 2, there is no significant difference among three methods. The mean of $|r|$ value among these three methods are $7.0757 \times 10^3$for CD, $7.1061 \times 10^3$ for II+CG,$7.0754 \times 10^3$ for matlab built-in function. Hence, accuracy:

$$\text{Matlab} > CD > II + CD$$

Similarly, from Figure 3, by comparing with determinate eigenvalue, Coordinate Descent method performances much better than CG + II method.

the mean error for Matlab, CD and CG+II are

$$9.5162 \times 10^{-17}, 2.3169 \times 10^{-11}, 2.711 \times 10^{-7},$$

respectively.

# Chapter 6

## 6.1 Conclusion

Precision of our CD method can be very close to matlab built-in function while relieving storage space when the smallest eigenvalues of matrix A are far away from each others. It also wins II+CG method under same level of complexity. However, when matrix A has multiple-eigenvalues or close smallest eigenvalues, CD method behaves poorly. Hence, we deplore that, in the real practice of dealing with huge matrix, applying Coordinate Descent or Inverse Iterative method combining Conjugate Gradient can efficiently solve the smallest eigen-pair problems while maintaining high accuracy. Many scholars also choose a specific matrix generation method when applying the CD method (such as method 3), so based on the foundation of the predecessors, we limit the storage capacity of the CD method to O(n) instead of the traditional matrix factorization method $O(n^2)$. At the same time, under the same amount of calculation, the CD method has obvious advantages over the iterative method in terms of accuracy. Furthermore, our achievement is an efficient approximation method when iterative method fails based on specific matrix properties.

# Appendix A

# Code

## A.1 Matlab Code for Conjugate Gradient

```
function [x,k] = conjgrad(A, b, x)
    r = b - A * x;
    p = r;
    rsold = r' * r;
    k=0;
  for k=1:length(b)
        k=k+1;
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-11
```

```
            break;
        end
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
end
```

## A.2 Main Matlab Code for numerical experiments

```
rand('seed',0)
k_iter=[];
rmat_check=[];
r_concheck=[];
iter=[];
acc=[];
rcheck=[];
no=0;
ind=0;
for n=10:20
n=n^2;
np=floor(n^0.5);
Ap=zeros(np,np);
A=rand(n,n);
A=A'*A;
%method 2:
```

```
% no=norm(A, 'fro ');
% B=A;
% A=B+100*no*eye(n);


%method 3:
% d=ones(n,1);
% for i=1:n
%       d(i)=d(i)-i*10^(-8)
% end

% A=randn(n,n);
% A=A'*A;
% D=diag(d);
% u=randn(n,n);
% u=orth(u);
% A=u*D*u';
x=randn(n,1);
% x=zeros(n,1);
% x(1)=1;
[true_ve,true_va]=eig(A);
ttar=min(abs(diag(true_va)));
iindex=find(abs(diag(true_va))==ttar);
true_x=true_ve(:,iindex);
y=A*x;
r=(x'*y)/(x'*x);
g=(y-x*r);
```

```
k=0;
r_mat=(true_x'*A*true_x/(true_x'*true_x));
rmat_check=[rmat_check,r_mat];
xcheck=[];
xpcheck=[];
sold=zeros(1,n)
s=ones(1,n)
order=[];
%CD
while(k<n^1.5)
k=k+1;
[a,s]=sort(abs(g),'descend');
Ap=A(s(1:np-1), s(1:np-1));
x_N=x(s(np:n))/norm(x(s(np:n)));
Ap(1:np-1, np) = A( s(1:np-1),s(np:n) )*x_N;
Ap(np, 1:np-1) = Ap(1:np-1, np)';
Ap(np, np) = x_N'*A(s(np:n), s(np:n))*x_N;
[vector,value]=eig(Ap);
tar=min(abs(diag(value)));
index=find(abs(diag(value))==tar);
xp=vector(:,index);
xp =xp*( norm(x(s(np:n)))/ xp(np));
xpcheck=[xpcheck,xp];
x(s(1:np-1)) = xp(1:np-1);
% xcheck=[xcheck,x];
y = A*x;
% if ( abs( x'*y/(x'*x)) > 0.99*abs(r) )
```

```
%       break
% end
r=(x'*y)/(x'*x);
g=2*(y-x*r)/(x'*x);
end
rcheck=[rcheck,r];
k_con=0;
xold=zeros(n,1);
xold(1)=1;
while(1)
k_con=k_con+1;
xnew=conjgrad(A,xold,xold);
r_con=(xnew'*A*xnew)/(xnew'*xnew);
if (r_con < r)
    break
end
xold=xnew;
end
r_concheck=[r_concheck,r_con]
k_iter=[k_iter,k_con];
iter=[iter,k];
end
plot(rcheck,'r')
hold on
plot(r_concheck,'g')
hold on
plot(rmat_check,'b')
```

```
legend( 'CD','II+CG','Matlab')
ylabel('␣r␣')
xlabel('dimension␣of␣matrix␣A')
mean(rcheck)
mean(r_concheck)
mean(rmat_check)
```

# Bibliography

[1] Powell, M. J. D. (1977). Restart procedures for the conjugate gradient method. Mathematical programming , 12 (1), 241-254.

[2] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain.

[3] Lasdon, L., Mitter, S., & Waren, A. (1967). The conjugate gradient method for optimal control problems. IEEE Transactions on Automatic Control , 12 (2), 132-138.

[4] Brinch Hansen, P. E. R. (1998). Conjugate gradient solution of linear equations. Concurrency: Practice and Experience , 10 (2), 139-156.

[5] Miller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. Neural networks , 6 (4), 525-533.

[6] Wright, S. J. (2015). Coordinate descent algorithms. Mathematical Programming , 151 (1), 3-34.

[7] Pardalos, P., Birge, J. R., Du, D. Z., Floudas, C. A., Mockus, J., Sherali, H. D., & Stavroulakis, G. (1994). Nonconvex Optimization and Its Applications.

[8] Wang, J., Wang, W., Garber, D., & Srebro, N. (2018, April). Efficient coordinate-wise leading eigenvector computation. In Algorithmic Learning Theory (pp. 806-820).

[9] Golub, G. H., & OLeary, D. P. (1989). Some history of the conjugate gradient and Lanczos algorithms: 1948C1976. SIAM review , 31 (1),50-102.