15

A SURVEY OF VARIOUS CONJUGATE GRADIENT ALGORITHMS FOR ITERATIVE SOLUTION OF THE LARGEST/SMALLEST EIGENVALUE AND EIGENVECTOR OF A SYMMETRIC MATRIX

X. Yang

15.1 Introduction

Conventional transformation methods, such as QR technique, for the eigen-problem of a symmetric matrix are quite popular. However, there has been an increasing interest in iterative algorithms due to the following two reasons.

First of all, unlike conventional methods which usually find all the eigenvalues and the eigenvectors simultaneously, iterative methods seek only the desired eigenvalues and eigenvectors. Hence, they should be more efficient. Moreover, the computational complexity of iterative algorithms depends largely on how easily the matrix-vector product Ax can be computed. Therefore, iterative methods could be even more efficient when A is sparse or when A has a certain special structure (like Hankel or Toplitz) for which FFT can be used to speed up the computation of Ax. Examples of the second case can be found in [1–3] where FFT has been used along with the conjugate gradient to solve Hankel systems.

Secondly, iterative methods can easily fit in the frame work of adaptive signal processing. Algorithms based on iterative methods were proposed in [4–9] to adaptively find the eigenvector corresponding to the minimum eigenvalue of a sample covariance matrix, while the covariance matrix is being continually updated.

In [10], it has been shown that the conjugate gradient method can be utilized in an efficient fashion for adaptively finding the eigenvector corresponding to the minimum eigenvalue. However, in this chapter, we present the conjugate gradient algorithm for the minimum/maximum eigen-problem of a fixed symmetric matrix. For convenience, only the minimum eigen-problem is mentioned throughout this chapter, unless it is necessary to distinguish it from the maximum eigen-problem, since the algorithms we shall discuss work equally well for both problems.

The chapter is organized as follows. In section 15.2 the essential basic concepts are outlined including the properties of the Rayleigh quotient, iterative algorithms for functional optimization, and their connections to the minimum eigen-problem. The conjugate gradient and steepest descent algorithms are presented and compared in section 15.3. The comparison of the two algorithms favors the conjugate gradient. Possible applications of the conjugate gradient algorithm are given in section 15.4. Finally, conclusive remarks are given in section 15.5.

15.2 Preliminaries

a. Rayleigh Quotient and its Properties

We shall consider a real symmetric matrix A of order n whose

eigenvalues and eigenvectors will be denoted by λ_i and x_i so that

$$Ax_i = \lambda_i x_i, \qquad i = 1, 2, \ldots, n, \tag{1}$$

with

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n, \tag{2}$$

and

$$\langle x_i, x_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{3}$$

where $\langle\ ,\ \rangle$ is the ordinary scalar product of vectors.

The Rayleigh quotient $\oslash(x)$ associated with the matrix A is a finite-dimensional nonquadratic functional which assigns to any non-zero real vector x a scalar quantity given by

$$\begin{aligned} \oslash(x) &= \langle x, Ax \rangle / \langle x, x \rangle \\ &= x^T Ax / x^T x \end{aligned} \tag{4}$$

where T denotes the transpose. Here both A and x are considered to be real.

The Rayleigh quotient has the following properties:

1) Homogeneity.

$\oslash(x)$ is homogeneous of degree one, i.e.,

$$\oslash(kx) = \oslash(x) \tag{5}$$

for any constant $k \neq 0$.

2) Minimal Residual.

Given $x \neq 0$, $\oslash(x)$ yields minimal residual, i.e.,

$$\|(A - \mu I)x\|_2^2 \geq \|Ax\|_2^2 - \oslash^2(x)\|x\|_2^2 \tag{6}$$

where $\|.\|_2$ is the 2-norm of a vector in R^n defined as

$$\|x\|_2 = \langle x, x \rangle^{1/2} \tag{7}$$

I is an identity matrix, and the equality holds only when $\mu = \oslash(x)$.

3) Boundedness.

The values of the Rayleigh quotient are in the real interval bounded by the extreme eigenvalues $[\lambda_1, \lambda_n]$, i.e.

$$\lambda_1 \leq \oslash(x) \leq \lambda_n \tag{8}$$

Also, $\oslash(x)$ is equal to λ_1 or λ_n when x is in the space spanned by x_1 or x_n.

4) Stationarity.

The gradient $g_{\oslash}(x)$ of $\oslash(x)$ is given by

$$g_{\oslash}(x) = 2 \star [Ax - \oslash(x)x]/\langle x, x \rangle \tag{9}$$

The Hessian of the Rayleigh quotient is

$$H_{\oslash} = 2 \star (A - g_{\oslash}x^T - xg_{\oslash}^T - \oslash I)/\langle x, x \rangle \tag{10}$$

From the definitions of gradient and Hessian of $\oslash(x)$, it is seen that the eigenvectors of A correspond to the stationary points of $\oslash(x)$ where the gradient is zero, and the eigenvalues of A are the values of $\oslash$ at the corresponding stationary points. Moreover, the minimum eigenvector x_1 is the only local and global minimum of $\oslash(x)$ which offers a stable stationary point (semi-positive Hessian) with respect to the *descent* search in the method to be discussed.

In the above discussion, the algebraic multiplicity of unity of the extreme eigenvalues has been assumed. When the minimum eigenvalue has multiplicity $r < n$, i.e.,

$$\lambda_1 = \lambda_2 = \cdots = \lambda_r \tag{11}$$

then, $\oslash(x)$ will assume its minimum when x is in the space of

$$\text{span}\{x_1, x_2, \ldots, x_r\} \tag{12}$$

The close relation between the eigen-data of A and the stationary points of its Rayleigh quotient enable us to obtain the minimum eigenvalue of A by minimizing the Rayleigh quotient of A.

b. Descent Methods in Functional Minimization

Most iterative methods for finding the minimum of a functional $f(x)$ defined on R^n take the form

$$x(k+1) = x(k) + \alpha(k)d(k) \tag{13}$$

where $d(k)$ is called the "search direction," and $\alpha(k)$ is called the "step length" which is chosen to minimize, or at least reduce, $f(x)$

along the line that passes through $x(k)$ in the direction of $d(k)$. Thus there are two distinct problems in such an iterative method: the choice of $d(k)$, and the determination of $\alpha(k)$.

An iterative method is called a descent method if it generates a sequence of scalars $\{f[x(k)]\}$ which is descending, i.e.,

$$f[x(k+1)] < f[x(k)] \tag{14}$$

Since the sequence is bounded from below by $f(x_m)$, where x_m is the global minimum of $f(x)$, it seems likely that the sequence $\{x(k)\}$ will converge to at least a local minimum of $f(x)$. That is why so much attention has been drawn to the study of descent methods in the functional minimization.

Corresponding to the choices of the search directions, there are various iterative methods among which the most well known ones are the steepest descent (SD), Newton descent (ND), variable metric (VM), and conjugate gradient (as a special version of more generalized conjugate direction) methods [11].

In the *steepest descent* (SD) algorithm, the negative of the gradient $g[x(k)]$ of f at $x(k)$ is taken as search direction $d(k)$.

In the *Newton's descent* (ND) algorithm, the functional f is approximated by the three-term Taylor series

$$\begin{aligned} f(x) &\approx f[x(k)] + [x - x(k)]^T g(k) + (1/2)[x - x(k)]^T H(k)[x - x(k)] \\ &= f_q(x) \end{aligned} \tag{15}$$

From $x(k)$ we move to $x(k+1)$ to minimize the quadratic functional $f_q(x)$. The minimum occurs when

$$g(k) + H(k)[x(k+1) - x(k)] = 0 \tag{16}$$

or

$$x(k+1) = x(k) - H(k)^{-1} g(k) \tag{17}$$

Since f is not purely quadratic, greater amount of descent at each iteration may be obtained by performing the optimal line search rather than just using (17), i.e.,

$$x(k+1) = x(k) - \alpha(k) H(k)^{-1} g(k) \tag{18}$$

This is achieved through the scalar factor $\alpha(k)$.

In the *variable metric* (VM) algorithm, the search direction is formed by

$$d(k) = -G(k)g(k) \tag{19}$$

where $G(k)$ is an $n \times n$ positive definite matrix approximating the inverse of the Hessian at $x(k)$. The search directions are guaranteed to be descending. $G(k)$ is updated in such a way that the search direction $d(k)$ is more and more like that in the ND algorithm as $x(k)$ gets closer and closer to the minimum. The most successful variable metric algorithm is credited to Davidon [12] and Fletcher and Powell [13].

In the *conjugate gradient* (CG) algorithm, the search direction is a linear combination of the gradient and the previous search direction, i.e.

$$p(k) = r(k) + \beta(k-1)p(k-1) \tag{20}$$
$$r(k) = -g(k) \tag{21}$$

The advantages and disadvantages of descent methods mentioned above are as follows:

1) The steepest descent uses values of the functional to be minimized and its gradient. It often exhibits very slow convergence;

2) The conjugate gradient uses values of the functional, its gradient, and the second derivative. It gives the exact solution in n steps for an n-dimensional quadratic functional, and is faster than the steepest descent;

3) The Newton descent uses values of the functional, its gradient, and the inverse of the matrix of second derivative. It has very rapid convergence when it converges! However, the convergence is highly dependent on the initial guess;

4) The variable metric descent requires the computation and storage of the step direction matrices $G(k)$ in addition to those required by the conjugate gradient. It gives the exact solution in n steps for an n-dimensional quadratic functional due to its connection to the conjugate gradient, and offers higher asymptotic rate of convergence than the conjugate gradient due to its connection to the Newton descent.

15.3 Steepest Descent and Conjugate Gradient Methods for the Minimum Eigen-Problem

Among the descent methods which minimize the Rayleigh quotient, the Newton descent and quasi-Newton descent are not suitable since the former requires and the latter attempts to approximate the Hessian of $\oslash(x)$ at the minimum which is a singular matrix [14]. In what follows, we describe the methods of steepest descent and conjugate gradient for the minimum eigen-problem.

Both the steepest descent and conjugate gradient algorithms start with an initial guess $x(0)$ for the minimum eigenvector and update it as

$$x(k+1) = x(k) + \alpha(k)p(k) \tag{22}$$

In (22), $\alpha(k)$ is obtained by optimal line search, that is, by minimizing the functional $\oslash(x)$ along the direction $p(k)$ passing through the point $x(k)$.

It can be shown that $\alpha(k)$ is always real and given by

$$\alpha(k) = -B + \sqrt{B^2 - 4CD}/(2D) \tag{23}$$

where

$$D = p_b(k)p_c(k) - p_a(k)p_d(k) \tag{24}$$

$$B = p_b(k) - \lambda(k)p_d(k) \tag{25}$$

$$C = p_a(k) - \lambda(k)p_c(k) \tag{26}$$

$$p_a = \langle p(k), Ax(k)\rangle/\langle x(k), x(k)\rangle \tag{27}$$

$$p_b = \langle p(k), Ap(k)\rangle/\langle x(k), x(k)\rangle \tag{28}$$

$$p_c = \langle p(k), x(k)\rangle/\langle x(k), x(k)\rangle \tag{29}$$

$$p_d = \langle p(k), p(k)\rangle/\langle x(k), x(k)\rangle \tag{30}$$

$$\lambda(k) = \oslash[x(k)] = \langle x(k), Ax(k)\rangle/\langle x(k), x(k)\rangle \tag{31}$$

At $x(k+1)$, a new search direction needs to be selected. The basic difference between the method of steepest descent and the method of conjugate gradient is how the new search direction $p(k+1)$ is selected.

a. Method of Steepest Descent

For this case, the search directions are chosen as the residual, i.e.

$$\begin{aligned} p(k+1) &= r(k+1) \\ &= [\lambda(k+1)x(k+1) - Ax(k+1)]/\langle x(k+1), x(k+1)\rangle \end{aligned} \tag{32}$$

With this particular choice of $p(k+1)$, certain simplifications can be made in (27)–(30). For example,

$$p_a(k) = -p_d(k) = -\langle r(k), r(k)\rangle / \langle x(k), x(k)\rangle \tag{33}$$

$$p_b(k) = \langle Ar(k), r(k)\rangle / \langle x(k), x(k)\rangle \tag{34}$$

$$p_c(k) = 0 \tag{35}$$

Therefore,

$$\alpha(k) = -[p_b(k) - \lambda(k)p_d(k)] + \sqrt{[p_b(k) - \lambda(k)p_d(k)]^2 + 4p_d^3(k)}/[2p_d^2(k)] \tag{36}$$

b. Method of Conjugate Gradient

For the conjugate gradient method, the new search direction are selected as

$$p(k+1) = r(k+1) + \beta(k)p(k) \tag{36}$$

where $\beta(-1) = 0$, and $\beta(k)$, $k = 0, 1, \ldots$, is such that $p(k+1)$ is conjugate to $p(k)$ with respect to some weighting matrix H, i.e., $p(k)$ is H-orthogonal

$$\langle p(k+1), Hp(k)\rangle = 0 \tag{37}$$

We shall discuss the choice of H in subsection 15.3.d.

The equations (36), (37) along with (22)–(31) constitute the conjugate gradient algorithm for the minimum eigenproblem of a symmetric matrix.

c. Comparison of SD and CG Algorithms

The rates of convergence of the SD and CG algorithms have been studied by many researchers. For example, it is shown in the book by Axelsson and Barker [16] that when the SD and CG algorithms are applied to minimize a quadratic functional, their rates of convergence are respectively

$$|x(k) - x_m|_A \leq [(\lambda_L - \lambda_S)/(\lambda_L + \lambda_S)]^k |x(0) - x_m|_A \tag{38}$$

and

$$|x(k) - x_m|_A \leq T_k[(\lambda_L + \lambda_S)/(\lambda_L - \lambda_S)]^{-1} |x(0) - x_m|_A \tag{39}$$

where λ_L and λ_s are the largest and smallest eigenvalues of the matrix A, T_k is the Chebyshev polynomial of degree k, and $|\cdot|_A$ denotes the vector norm with respect to the positive definite matrix A defined as

$$|x|_A = \langle x, Ax \rangle^{1/2} \tag{40}$$

Further if $p(\epsilon)$ is defined for any $\epsilon > 0$ to be the smallest integer k such that

$$|x(k) - x_m|_A \leq \epsilon |x(0) - x_m|_A, \qquad x(0) \epsilon R^n \tag{41}$$

then, from (38) and (39), bounds are found for $p_{SD}(\epsilon)$ and $p_{CG}(\epsilon)$ as

$$p_{SD}(\epsilon) \leq (1/2)\kappa(A)\ln(1/\epsilon) + 1 \tag{42}$$

and

$$p_{CG}(\epsilon) \leq (1/2)\sqrt{\kappa(A)}\ln(1/\epsilon) + 1 \tag{43}$$

where $\kappa(A)$ is the spectral condition number of A, and ln() represents natural logarithm. It is seen that when $\kappa(A)$ is large the bound for $p_{CG}(\epsilon)$ is much smaller than that for $p_{SD}(\epsilon)$. For example, if $\kappa(A) = 10^4$ and $\epsilon = 10^{-4}$, then $p_{CG} \leq 496$, whereas $p_{SD} \leq 46052$.

It is important to notice that (39) and (43) are derived without assuming anything about the eigenvalue distribution of A. Depending on the eigenvalue distribution, they may be quite pessimistic. Several examples are given in [16] to demonstrate the above point and lead to the well known fact that the "clustering" of eigenvalues tends to increase the rate of convergence of the CG algorithm. For a nonquadratic functional, convergence behavior similar to (38) and (39) should be expected in the latter stages of the minimization since the functional is typically near quadratic in the immediate vicinity of a minimum. Therefore, as far as the rate of convergence is concerned, the conjugate gradient algorithm is generally much better than the steepest descent algorithm.

Since near the minimum the Hessian of the Rayleigh quotient is nonnegative definite instead of positive definite, what we have discussed doesn't seem applicable. However, it is shown [14] that similar convergence behavior exists when the SD and CG algorithm are applied to a quadratic functional whose Hessian is nonnegative.

Intuitively, convergence of the CG algorithm is faster than the SD algorithm because the trial eigenvector is updated by a vector chosen

from a larger space of vectors in the CG algorithm than the corresponding space in the SD algorithm. Moreover, the CG algorithm tends to converge in less than n iterations once the trial eigenvector is in the region close to the minimum eigenvector.

d. Comparison of Various Versions of the CG Algorithm

Equation (37) is the most fundamental relation of the conjugate gradient method. The choice of the weighting matrix H in (37) renders various algorithms. For example, in [10], Chen et al. utilized the matrix A for H. This results in

$$\beta(k) = -\langle r(k+1); Ap(k)\rangle / \langle p(k); Ap(k)\rangle \tag{44}$$

Townsend and Johnson [23] suggested the use of $\underline{H}$ defined by

$$r(k+1) = r(k) + \underline{H}(k)[x(k) - x(k+1)] \tag{45}$$

which leads to

$$\beta(k) = -\langle r(k+1); r(k) - r(k+1)\rangle / \langle p(k); r(k) - r(k+1)\rangle \tag{46}$$

A better known formula for $\beta(k)$ is

$$\beta(k) = \langle r(k+1); r(k+1)\rangle / \langle r(k); r(k)\rangle \tag{47}$$

which was developed by Fletcher and Reeves [24]. They were among the first to apply a conjugate gradient algorithm to the minimization of a nonquadratic functional.

When the conjugate gradient method is applied to minimize a quadratic functional of the form

$$(1/2)x^T Ax - x^T b + \text{constant} \tag{48}$$

the conjugacy is made among search directions with respect to the matrix A, which is the Hessian of the functional. Here T denotes the transpose of the matrix. In the case of minimizing the Rayleigh quotient, which is a nonquadratic functional, we are inclined to think that the most natural choice of H is the Hessian matrix of $\varnothing(x)$. This is because near the minimum point x_1 of $\varnothing(x)$, $\varnothing(x)$ behaves like a quadratic functional, i.e.,

$$\varnothing(x) \approx \varnothing(x_1) + (1/2)(x - x_1)^T H(x_1)(x - x_1) \tag{49}$$

Generally, there are two factors that make the conjugacy with respect to Hessians less desirable. One factor is that the Hessian of a nonquadratic functional is not always easily available. The other is the extra computation and storage required to evaluate the Hessian. However, these problems do not appear for the case of extreme eigenproblems.

For the Rayleigh quotient, the Hessian is given by (10). This particular choice of the Hessian of $\oslash(x)$ at x_{k+1}, denoted by $H(k+1)$, for the H in (37) yields

$$\beta(k) = -\langle r(k+1); H(k+1)p(k)\rangle / \langle p(k); H(k+1)p(k)\rangle \tag{50}$$

This results in

$$\beta(k) = -[\langle r(k+1); Ap(k)\rangle + \langle r(k+1); r(k+1)\rangle\langle x(k+1); p(k)\rangle] \\ /[\langle p(k); Ap(k)\rangle - \lambda(k+1)\langle p(k); p(k)\rangle] \tag{51}$$

The above result is arrived at by utilizing

$$\langle r(k+1); p(k)\rangle = 0 \tag{52}$$

which is the consequence of the optimal line search used to obtain $\alpha(k)$ in (22). It is seen from (51) that there is no need to compute and store $H(k+1)$ at all!

Besides the choice of $\beta(k)$, another issue of concern is the effect of normalization. The normalization was adopted in [10], partially to prevent numerical unstability and overflow caused by possible extremely large magnitudes of the trial eigenvector. Specifically, an additional step following the optimal line search is used to ensure the unity magnitude of the trial eigenvector in each iteration, i.e., (22) is replaced by

$$x'(k+1) = x(k) + \alpha(k)p(k) \tag{53}$$

and

$$x(k+1) = x'(k+1)/|x'(k+1)| \tag{54}$$

In summary, various versions of the CG algorithm include:

Algorithms CA and CN: both use (44) for $\beta(k)$, with and without the normalization, respectively.

Algorithms TJ and TN: both use (46) for $\beta(k)$, with and without the normalization, respectively.

Algorithms FR and FN: both use (47) for $\beta(k)$, with and without the normalization, respectively.

Algorithms HE and HN: both use (51) for $\beta(k)$, with and without the normalization, respectively.

Several numerical examples are given for the comparison among various versions of the CG-algorithm [15]. The computer programs are implemented in single precision.

As a first example, we consider the same problem as described in [10], where matrix A is the sixteenth order covariance matrix of a typical speech signal written as

$$A = \begin{bmatrix} r_0 & r_1 & \cdots & r_{15} \\ r_1 & r_0 & \cdots & r_{14} \\ \vdots & \vdots & \ddots & \vdots \\ r_{15} & r_{14} & \cdots & r_0 \end{bmatrix} \tag{55}$$

where the various autocorrelation coefficients are listed below:

$$\begin{aligned} r_0 &= 1.00000000, & r_1 &= 0.91189350, \\ r_2 &= 0.75982820, & r_3 &= 0.59792770, \\ r_4 &= 0.41953610, & r_5 &= 0.27267350, \\ r_6 &= 0.13446390, & r_7 &= 0.00821722, \\ r_8 &= -0.09794101, & r_9 &= -0.21197350, \\ r_{10} &= -0.30446960, & r_{11} &= -0.34471370, \\ r_{12} &= -0.34736840, & r_{13} &= -0.32881280, \\ r_{14} &= -0.29269750, & r_{15} &= -0.24512650 \end{aligned} \tag{56}$$

The minimum eigenvalue is

$$\lambda_{\min} = 0.0032584817 \tag{57}$$

which is obtained using the QR algorithm for the eigen-problem of a symmetric matrix in double precision. The specific algorithm utilized was EIGRS, which is available in the IMSL library.

We apply the eight versions of CG-algorithm, and compare the number of iterations and the cpu time taken for convergence starting with different initial values of the trial eigenvector. For the purpose of comparison, we list results of all versions of CG-algorithm in Table 15.1. The symbol N denotes the number of iterations, and T denotes

x_0	random vector			$(-1,1,-1,\ldots)^T$			$(1,0,0,\ldots,0)^T$		
Alg.	N	T	λ_1	N	T	λ_1	N	T	λ_1
HE	66	15	0.0032588	24	4	0.0032585	77	16	0.0032586
HN	82	17	0.0032586	24	6	0.0032585	96	21	0.0032586
TJ	68	16	0.0032588	26	5	0.0032585	65	14	0.0032586
TN	77	18	0.0032586	26	5	0.0032586	65	15	0.0032586
FR	81	17	0.0032586	18	4	0.0032585	88	18	0.0032586
FN	84	19	0.0032587	17	4	0.0032586	87	19	0.0032585
CA	119	26	0.0032588	32	7	0.0032585	124	25	0.0032586
CN	124	26	0.0032587	32	7	0.0032586	132	30	0.0032585

Table 15.1 Stimulation results of varous CG-algorithms for the first example.

the total cpu time taken to obtain the minimum eigenvalue λ_1 (to an accuracy of 10^{-4}) shown in the table.

It is observed from Table 15.1 that all versions of CG-algorithm converge to the minimum eigenvalue with an accuracy of 4 decimal digits regardless of the different trial eigenvectors they start with. The initial value $(-1,1,-1,1,\ldots,)^T$ produces the fastest convergence among the three different initial values in this example. The performance of the algorithms HE, HN, TJ, TN, FR, and FN is evidently better than that of algorithms CA, and CN. There is little difference in performance between algorithms in which the trial eigenvector is normalized in each iteration and those without the normalization.

As a second example we choose a 50×21 Hankel matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_{21} \\ a_2 & a_3 & \cdots & a_{22} \\ \vdots & \vdots & \ddots & \vdots \\ a_{50} & a_{51} & \cdots & a_{70} \end{bmatrix} \tag{58}$$

which is obtained from sampling the sum of 10 sinusoidal signals with random phases plus white Gaussian noise of zero mean, i.e.,

$$s(t) = \sum_{k=1}^{10} \sin[2\pi(0.08 + 0.4k)t + 2\pi \mathrm{ran}(k)] + w(t) \tag{59}$$

where $\mathrm{ran}(k)$'s are random samples of a random variable uniformly distributed between -1 and 1, and $w(t)$ is the Gaussian noise.

To determine the sinusoids from the data samples a_i, it is necessary in Pisarenko's method to obtain the minimum eigenvector of

x_0	random vector			$(-1,1,-1,\ldots)^T$			$(1,0,0,\ldots,0)^T$		
Alg.	N	T	λ_1	N	T	λ_1	N	T	λ_1
HE	39	11	0.205751	19	6	0.205754	36	14	0.205749
HN	39	13	0.205749	19	7	0.205753	33	12	0.205750
TJ	22	7	0.205751	20	6	0.205754	21	7	0.205750
TN	21	8	0.205751	20	6	0.205749	21	7	0.205750
FR	21	6	0.205748	22	7	0.205753	20	5	0.205750
FN	22	7	0.205747	62	20	0.205751	21	7	0.205752
CA	49	16	0.205747	20	7	0.205750	49	17	0.205753
CN	49	15	0.205747	20	7	0.205754	49	16	0.205749

Table 15.2 Comparison of various CG-algorithms for the second example.

A^TA. Once again, we apply the eight versions of the CG-algorithm, and compare the number of iterations and the cpu time taken by each of them to converge for different initial values for the trial eigenvector. The results listed in Table 15.2 are similar to the observations of Table 15.1.

As a third example, we apply CG-algorithms to a symmetric matrix of order 40 created by performing a similarity transformation on a diagonal matrix, i.e.,

$$A = Q^T D Q \tag{60}$$

where D is the matrix of eigenvalues given by

$$D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{40}) \tag{61}$$

and Q is an orthogonal matrix generated by

$$Q = I - 2vv^T/(v^Tv) \tag{62}$$

with I being an identity matrix, and v being a random vector.

By specifying the matrix D in different ways, one gets some insight into the performance of the CG-algorithms. Specifically, four sets of eigenvalues are specified to test CG-algorithms for cases of multiple, negative minimum eigenvalues, and clustering eigenvalues. They are:

1) $(-1.5,-1.5,-1.5,4,5,\ldots,40)$;
2) $(-1.5,-1.5,-1.5,4,4,\ldots,4)$;
3) $(-1.5,-1.5,-1.48,4,5,\ldots,40)$;
4) $(1.5,1.5,1.52,4,5,\ldots,40)$.

The numerical results corresponding to the random initial trial eigenvector are in Table 15.3 in which the obtained minimum eigenvalue and the required number of iterations are recorded. It is seen

D	(1)		(2)		(3)		(4)	
Alg.	N	λ_1	N	λ_1	N	λ_1	N	λ_1
HE	15	-1.50000	1	-1.50000	35	-1.50000	44	1.50000
HN	15	-1.50000	1	-1.50000	26	-1.49967	45	1.50000
TJ	14	-1.50000	1	-1.50000	25	-1.49998	33	1.50010
TN	15	-1.50000	1	-1.50000	25	-1.49998	32	1.50011
FR	14	-1.50000	1	-1.50000	25	-1.49994	32	1.50002
FN	14	-1.50000	1	-1.50000	34	-1.49999	45	1.50000
CA	50	-1.50000	1	-1.50000	66	-1.48697	161	1.50005
CN	50	-1.50000	1	-1.50000	62	-1.48696	161	1.50005

Table 15.3 Simulation results of various CG-algorithms for the third example.

that the CG-algorithms still converge for the multiple and/or negative minimum eigenvalue. The rate of convergence of the CG algorithm for the minimum eigen-problem is dependent on the eigenvalue distribution of $H_{\mathcal{Q}}(x_1)$. In particular, the "clustering" of nonzero eigenvalues of $H_{\mathcal{Q}}(x_1)$ results in fast convergence as evident from the results for the second set of eigenvalues. It is apparent that the observations made in previous examples are still true in this example.

We conclude from those numerical examples that the performance of algorithms HE, HN, TJ, TN, FR, and FN is generally better than that of algorithms CA, and CN. Since $\underline{H}(k)$ in (45) appears to be an approximation of $H(k)$, and (46) and (47) are equivalent near the minimum where the orthogonality of the residulas $r(k+1)$ and $r(k)$ are approximately true, it seems that as far as the rate of convergence is concerned, the matrix A should be replaced by the Hessian-like matrices for generating mutual conjugate directions. The comparison among the algorithms utilizing (46), (47), and (48) for $\beta(k)$ shows that the difference in performance is not significant. Also, it is shown that the normalization of x_k in each iteration seems to have very little effect on the rate of convergence.

15.4 Applications of the CG Algorithm for Extreme Eigen-Problem

The applications of the CG algorithm are mainly for efficiency in computational effort and flexibility in real time implementation. In this section, we give some examples.

a. Computation of the Prolate Spheroidal Functions [17]

The computation of the prolate spheroidal functions demonstrates the efficiency of the CG algorithm. The prolate spheroidal functions are the eigenfunctions of the following integral equation [18]

$$\int_{-1}^{1} \varnothing(x) \sin[c(t-x)]/[\pi(t-x)]\, dx = \lambda \varnothing(t) \tag{63}$$

where c is a parameter, λ is the eigenvalue and the eigenfunction $\varnothing(x)$ is the prolate spheroidal function.

It is shown that the operator

$$\underline{A}X(t) = \int_{-1}^{1} X(v) \sin[c(t-v)]/[\pi(t-v)]\, dv \tag{64}$$

is self-adjoint and positive definite. Therefore, according to Daniel [19] we can use the conjugate gradient algorithm for the computation of the prolate spheroidal functions. For computational purpose, (63) can be written as

$$\lambda_c \varnothing(c,t) \approx \sum_{i=1}^{N} w_i \sin[c(t-t_i)]/[\pi(t-t_i)]\, \varnothing(c,t_i) \tag{65}$$

In (65), the integral has been replaced by a summation. This can be achieved numerically by the N point Gauss-Legendre quadrature formula [20] which has the least truncation error for a given number of samples of the function among all numerical integration routines. The Gauss-Legendre quadrature is given by

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{i=1}^{N} w_i f(x_i) \tag{66}$$

where w_i's are the weights, and x_i's are the abscissa points at which the function $f(x_i)$ is evaluated.

By defining the eigenvector x as

$$x = (x_1, x_2, \ldots, x_N)^T \tag{67}$$

with

$$x_i = \sqrt{w_i}\, \varnothing(c,t_i), \qquad i = 1,2,\ldots,N \tag{68}$$

n	$c = 0.5$			$c = 1.0$			$c = 2.0$			$c = 4.0$		
	L	p	m	L	p	m	L	p	m	L	p	m
0	3.0989557	1	2	5.7258178	1	3	8.8055992	1	3	9.9588544	1	4
1	8.5810737	3	3	6.2791273	2	4	3.5205866	1	4	9.1210736	1	4
2	3.9174534	5	2	1.2374793	3	2	3.5867687	2	3	5.1905483	1	3

Table 15.4 Eigenvalues of prolate spheroidal equation obtained by CG-algorithm ($\lambda_n = L \times 10^{-p}$.)

the operator equation of (63) is transformed into a matrix equation

$$Ax = \lambda x \tag{69}$$

where A is a symmetric matrix whose elements are given by

$$a_{ij} = \sqrt{w_i w_j}\,\sin[c(t_i - t_j)]/[\pi(t_i - t_j)] \tag{70}$$

and the eigenvector of A corresponds to scaled values of the prolate spheroidal function at the quadrature abscissas of the Gauss-Legendre quadrature formula, as seen from (68).

Transformation methods like the QR algorithm could be used to solve (69). However, for fixed value of c the magnitude of eigenvalue λ_i falls off to zero rapidly with increasing i once i has exceeded $(2/\pi)c$, so only the eigenfunctions corresponding to a few largest eigenvalues are needed in practice. In that case, the CG algorithm is likely to be more efficient.

In Table 15.4, the eigenvalues computed by the CG-algorithm are given. The results are obtained with double precision 256 point Gauss-Legendre formula. The program is run in double precision with x_0 as $(1, -1, 1, -1, \ldots)^T$. The eigenvalues have been computed for four different c, as defined in (63). In the table, the first column n describes the order of eigenvalues, p denotes the magnitude of the exponent and m the number of iterations required to obtain the eigenvalue with eight significant digits of accuracy. In order to find the higher order eigenvalues the initial guess is taken as orthogonal to the eigenvectors associated with the already obtained eigenvalues. The efficiency of the algorithm is evident because it finds a few desired eigenvalues directly each of which takes only a few iterations to converge. This contrasts the conventional algorithm which seeks all 256 eigenvalues in spite of the need for only a few of them. The accuracy of the given results are also quite satisfactory since the eigenvalues obtained by CG-algorithm have more significant digits than those reported in Table I of [18].

b. Application in Pisarenko's Method of Frequency Estimation

The Pisarenko harmonic decomposition (PHD) method is one of the earliest procedure for spectral estimation by eigenanalysis [21]. Assuming the process consisting of M sinusoids in additive white noise, PHD derives the sinusoidal frequencies, its powers, and white noise variance from the known autocorrelation sequence $r_{xx}(0)$ to $r_{xx}(2M)$. It is shown that the sinusoid frequencies can be determined by factoring the eigenfilter polynomial

$$\sum_{k=0}^{2M} v_{k+1} z^{-k} \tag{71}$$

whose coefficients v_k's are given by the minimum eigenvector of autocorrelation matrix R_{2M+1}, where R_p is defined by

$$R_p = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(p) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(p) & r_{xx}(p-1) & \cdots & r_{xx}(0) \end{bmatrix} \tag{72}$$

Therefore, PHD leads to a symmetric matrix minimum eigenvalue problem which can be solved by any one of the several standard algorithms. However, the autocorrelation sequence normally is not known so we have to deal with the eigenequation

$$X^T X v = \lambda v \tag{73}$$

where X is the covariance data matrix of the form

$$X = \begin{bmatrix} x_0 & x_1 & \cdots & x_{2M} \\ x_1 & x_2 & \cdots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_N & x_{N+1} & \cdots & x_{N+2M} \end{bmatrix} \tag{74}$$

One obvious way to obtain the minimum eigenvalue and eigenvector in the matrix in (73) is to form the symmetric matrix $X^T X$, and then to invoke any one of the standard algorithms such as the QR algorithm. However, we may choose to use the CG algorithm, which in addition to the possible gain in efficiency and accuracy (as discussed in

previous chapters) has the following advantages from technical point of view:

Practically, for the ability to track slowly time-varying process it is often necessary to implement PHD in an adaptive fashion. Algorithms of this sort were first proposed by Thompson [4], followed by Reddy et al. [5], Sarkar et al. [6], Vaccaro [22], and others. In this application, the underlying matrix is not fixed but a changing matrix which is updated with each new data sample. Since the minimum eigenvector of the changing matrix corresponds to the true PHD solution only when a large number of data samples are taken, the adaptive algorithms do not compute the minimum eigenvector of the matrix, but only update the trial eigenvector by one iteration of the algorithm for every new data sample until the trial eigenvector converges to the minimum eigenvector of the matrix. Apparently, conventional transformation methods, such as the QR method, can't be used here. The use of conjugate gradient algorithm in adaptive PHD was proposed by Chen et al. [10] and Sarkar et al. [6].

Secondly, the CG algorithm is more flexible in taking advantage of the special structure of matrix X to achieve savings in the computational effort, storage, and easier hardware implementation.

It is seen that most of the computation time in the conjugate gradient algorithm is used for the evaluation of Xv, Xp, and $X^T(Xv)$ which are required each iteration. These are products of a matrix and a vector each of which requires $N \times M$ operations. We can reduce the number of operations required from $N \times M$ to approximately $2(M+N-1)\log(M+N-1)$ by employing FFT which is much faster than the conventional multiplication method when N and M are large.

By observing the Hankel structure of matrix X, the product

$$X_p = \begin{bmatrix} x_0 & x_1 & x_2 \\ x_1 & x_2 & x_3 \\ x_2 & x_3 & x_4 \\ x_3 & x_4 & x_5 \\ x_4 & x_5 & x_6 \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} \tag{75}$$

as the convolution of two sequences $\{a\}$ and $\{b\}$, where

$$\{a\} = \{x_0; x_1; x_2; x_3; x_4; x_5; x_6\} \tag{76}$$

and

$$\{b\} = \{p_0; p_1; p_2; 0; 0; 0; 0\} \tag{77}$$

The product is embedded in the convolution of the two sequences $\{a\}$ and $\{b\}$. This is clear as one observes the 3rd, 4th, 5th, 6th and 7th elements in the convolution of $\{a\}$ and $\{b\}$. We know that the convolution can be efficiently carried out by FFT. So if we perform the FFT of the sequences $\{a\}$ and $\{b\}$ and multiply them and then take the inverse FFT to obtain the sequence $\{c\}$ as

$$\{A\} = \text{FFT}\{a\} \tag{78}$$

$$\{B\} = \text{FFT}\{b\} \tag{79}$$

$$\{c\} = \{a\} * \{b\} = (\text{FFT})^{-1}[\text{FFT}\{a\}\text{FFT}\{b\}] \tag{80}$$

then the desired result is the 3rd to the 7th element of $\{c\}$. The total number of operation is MN for direct multiplication and $2(M+N-1)\log(M+N-1)$ using FFT. If $(M+N-1)$ is not a number of type 2^n, zero padding could be used. It is seen that the application of the FFT becomes more efficient as the matrix size increases. It is interesting to point out that the FFT of the sequence of $(M+N-1)$ elements of X needs to be computed only once, and then it could be used to compute Xp, Xv, and $X^T(Xv)$ as well in each iteration. There is also a saving from a storage point of view. Instead of requiring the storage of $M \times N$ elements of matrix X, one needs to stor only $(M+N-1)$ elements in an array. This idea of utilizing FFT to evaluate matrix products has been used before [1–3].

15.5 Conclusive Remarks

For the minimum/maximum eigen-problem of a symmetric matrix, the conjugate gradient is often more efficient and more accurate than the conventional eigen-problem algorithms. Moreover, it is more convenient to use with signal processing techniques such as the adaptation and the FFT. Hence, it offers a good alternative to conventional algorithms for the solution of the maximum/minimum eigen-problem.

References

[1] Fienup, J. R., "Phase retrieval algorithms: a comparison," *Applied Optics,* 2758–2769, Aug., 1982.

[2] Sarkar, T. K., and X. Yang, "Accurate and efficient solution of Hankel matrix systems by FFT and the conjugate gradient methods," *Proc. ICASSP84,* 1835–1838, 1987.

[3] Sarkar, T. K., E. Arvas, and S. M. Rao, "Application of FFT and the conjugate gradient method for the solution of electromagnetic radiation from dielectrically large and small conduction bodies," *IEEE Trans. Antennas Propagat.,* 635–640, May, 1986.

[4] Thompson, P. A., "An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise," *Proc. 13th Asilomar Conf. Circuits, Syst., Comput.,* 529–533, 1980.

[5] Reddy, V. U., B. Egardt, and T. Kailath, "Least-squares-type algorithm for adaptive implementation of Pisarenko's harmonic retrieval method," *IEEE Trans. Acoust., Speech, Signal Processing,* **ASSP-30**, 399–405, June, 1982.

[6] Sarkar, T. K., S. A. Dianat, and S. M. Rao, "A finite step adaptive implementation of the Pisarenko harmonic retrieval method in colored nose," *Proc. ICASSP 83,* 1102–1105, Apr., 1983.

[7] Bellanger, M. G., *Adaptive Digital Filters and Signal Analysis,* Marcel Dekker, New York, 1987.

[8] Vaccaro, R., "On adaptive implementations of Pisarenko harmonic retrieval," *Proc. ICASSP 84,* 6.1.1, Mar., 1984.

[9] Fuhrmann, D. D., and B. Liu, "Rotational search methods for adaptive Pisarenko harmonic retrieval," *IEEE Trans. Acoust., Speech, Signal Processing,* **ASSP-34**, 1550–1565, Dec., 1986.

[10] Chen, H., T. K. Sarkar, J. Brule, and S. A. Dianat, "Adaptive spectral estimation by the conjugate gradient method," *IEEE Trans. Acoust., Speech, Signal Processing,* **ASSP-34**, 272–284, Apr., 1986.

[11] Dorny, C. N., *A Vector Space Approach to Models and Optimization,* Huntington, New York, 1975.

[12] Davidon, W. C., "Variable Metric Method for Minimization," *Ar-*

gonne National Laboratory Report, ANL–5990.

[13] Fletcher, R., and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal,* **6,** 163–168, 1963.

[14] Yang, X., *Conjugate Gradient Algorithms for the Solution of Extreme Eigen-problem of a Hermitian Matrix and its Applications in Signal Processing,* Ph.D. Dissertation, Syracuse University, 24–28.

[15] Yang, X., T. K. Sarkar, and E. Arvas, "A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix," *IEEE Trans. Acoust., Speech, Signal Processing,* **ASSP-37,** 1550–1556, Oct., 1989.

[16] Axelsson, O., and V. A. Barker, *Finite Element Solution of Boundary Value Problems,* Academic Press, New York, 1984.

[17] Sarkar, T. K., and X. Yang, "Application of the Conjugate Gradient and Steepest Descent for Computing the eigenvalues of an operator," *Signal Processing,* **17,** 31–38, May, 1989.

[18] Slepian, D., and H. O. Pollack, "Prolate Spheroidal Wave Functions, Fourier Analysis and Uncertainty—I.," *B.S.T.J.,* 40, **1,** 43–63, Jan., 1961.

[19] Daniel, J. W., *The Conjugate Gradient Method for Linear and Nonlinear Operator Equations,* Ph.D. dissertation, Stanford University, 53.

[20] Krylov, V. I., trans. A. H. Stroud, *Approximate Calculation of Integrals,* The Macmillan Company, New York, 1962.

[21] Pisarenko, V. F., "The Retrieval of Harmonics from a Covariance Function," *Geophys. J. R. Astron. Soc.,* **33,** 347–366, 1973.

[22] Vaccaro, R., "On Adaptive Implementations of Pisarenko Harmonic Retrival," *Proc. ICASSP 84,* 6.1.1, Mar.,1984.

[23] Townsend, M. A., and G. E. Johnson, "In favor of conjugate directions: a generalized acceptable-point algorithm for function minimization," *J. Franklin Inst.,* **306,** Nov., 1978.

[24] Fletcher, R., and C. M. Reeves, "Function minimization by conjugate gradients," *Comput. J.,* **7,** 149–154, 1964.