TMA 4180 Optimeringsteori
THE CONJUGATE GRADIENT METHOD

H. E. Krogstad, IMF, Rev. spring 2012/2013

1 INTRODUCTION

This note summarizes main points in the numerical analysis of the Conjugate Gradient (CG) method. Most of the material is also found in N&W, but the note contains the proof of Eq. 5.36, which is the main convergence result for CG. Some numerical experiments are also included.

2 DERIVATION

The Conjugate Gradient method is derived from the quadratic test problem,

$$\min_{x\in\mathbb{R}^n} \phi(x), \tag{1}$$

where

$$\phi(x) = \frac{1}{2}x'Ax - b'x,$$

and $A \in \mathbb{R}^{n\times n}$ is positive definite, $A > 0$. The method was originally considered to be a direct method for linear equations, but its favorable properties as an iterative method was soon realized, and it was later generalized to more general optimization problems.

A key point in the derivation is clever use of the *A-scalar product*, $\langle x, y\rangle_A = x'Ay$, and the corresponding norm, $\|x\|_A = (x'Ax)^{1/2}$ (see Basic Tools note).

We already know that $\nabla\phi(x)' = g = Ax - b$ and $\nabla^2\phi = A$, so that

$$x^* = \arg\min_{x\in\mathbb{R}^n} \phi(x) = A^{-1}b. \tag{2}$$

The first observation is that a set of n A-orthogonal vectors, $p_0, p_1, \cdots, p_{n-1}$, that is,

$$p_i'Ap_j = p_i'Ap_i \times \delta_{ij}, \tag{3}$$

is a *basis* for $\mathbb{R}^n$(recall TMA 4110/15 or a similar course for the terminology!). This property can be stated in several equivalent ways:

1. The n vectors $\{p_j\}$ are linearly independent.
2. $\operatorname{span}\{p_0, p_1, \cdots, p_{n-1}\} = \mathbb{R}^n$.
3. Every $x \in \mathbb{R}^n$ can be written uniquely as $x = \sum_{j=0}^{n-1} \alpha_j p_j$.

Let us for the derivation of the CG method assume that we already have a basis $\{p_0, p_1, \cdots, p_{n-1}\}$, that we start at $x_0 = 0$, and let $p_0 = -g_0 = b$ (the start point is easily adjusted later). The A-orthogonality enables us to write $x^* = \sum_{j=0}^{n-1} \alpha_j p_j$, where

$$\alpha_k = \frac{p_k'Ax^*}{p_k'Ap_k} = \frac{p_k'b}{p_k'Ap_k} = -\frac{p_k'g_0}{p_k'Ap_k}. \tag{4}$$

It is interesting to observe that we can compute the coefficients in the expansion for x^* without knowing x^* itself!

Below, the subspaces

$$B_k = \operatorname{span}\{p_0, \cdots, p_{k-1}\}, \ k = 0, \cdots, n-1, \tag{5}$$

will play an important role. Observe that

$$B_0 \subset B_1 \subset \cdots \subset B_{n-1} = \mathbb{R}^n. \tag{6}$$

It is a general property of expansions in *orthogonal bases* that partial sums are the *best approximations in the corresponding subspaces spanned by the basis-vectors.* This means that

$$x_k = \sum_{j=0}^{k-1} \alpha_j p_j = \arg \min_{y \in B_k} \|y - x^*\|_A. \tag{7}$$

As long as we look around for a good approximation to x^* in B_k, x_k is the best choice (in the A-norm)! In the present case, x_k is also *best* in another sense, namely, it actually solves the optimization problem

$$x_k = \arg \min_{y \in B_k} \phi(y). \tag{8}$$

Show this yourself by deriving that $\|y - x^*\|_A^2 = 2\phi(y) + b' A^{-1} b$.

Equation 8 represents a constrained problem (minimum constrained to B_k). The feasible directions d from x_k will all be vectors in B_k, and since $\nabla\phi(x_k)\, d \geq 0$, and $-d \in B_k$ if d is,

$$\nabla\phi(x_k)\, d = g_k' d = 0 \text{ for all } d \in B_k. \tag{9}$$

This could equally well be derived from the famous *Projection Theorem*, since the error, $x_k - x^*$, is A-orthogonal to B_k:

$$\begin{aligned} 0 &= (x_k - x^*)' A d \\ &= (A x_k - b)' d \\ &= g_k' d \ \text{ for all } \ d \in B_k. \end{aligned} \tag{10}$$

The only remaining problem is to actually find $\{p_j\}_{j=0}^{n-1}$, and the standard way would be to do this by a Gram-Schmidt procedure, – if we had a set of vectors to start with. If we know p_0, p_1, $\cdots$, p_{k-1}, it is tempting to try

$$p_k = -g_k + \beta_{k-1} p_{k-1} + \tilde{\beta}_{k-2} p_{k-2} + \cdots + \tilde{\beta}_0 p_0, \tag{11}$$

since we already know from above that g_k is orthogonal to and hence linearly independent of $\{p_0, p_1, \cdots, p_{k-1}\}$.

The rather surprising result is now that it suffices to use

$$p_k = -g_k + \beta_{k-1} p_{k-1}, \tag{12}$$

and then determine β_{k-1} so that $p_k' A p_{k-1} = 0$,

$$\beta_{k-1} = \frac{p_{k-1}' A g_k}{p_{k-1}' A p_{k-1}}. \tag{13}$$

In order to prove that this is really true, we recall the inclusions in Eq. 6 and base the proof on induction. Starting with any p_0 (e.g., $p_0 = -g_0$), the next $p_1 = -g_1 + \beta_0 p_0$ can be made A-orthogonal to p_0 by choosing β_0 as in Eq. 13. One then needs to verify that $\{p_0, p_1, \cdots, p_{k-1}, p_k\}$ is an A-orthogonal set if the vectors $\{p_0, p_1, \cdots, p_{k-1}\}$ are A-orthogonal (the *induction hypothesis*). We already know that $p_k' A p_{k-1} = 0$, but what about p_k and $p_0, p_1, \cdots, p_{k-2}$?

Consider p_k and p_i for $i \le k-2$:

$$p_i' A p_k = p_i' A \left(-g_k + \beta_{k-1} p_{k-1}\right) = -p_i' A g_k, \tag{14}$$

since we already know $p_i' A p_{k-1} = 0$ by the induction hypothesis. The trick is now to write

$$x_{i+1} = x_i + \alpha_i p_i, \tag{15}$$

multiply by A and subtract b on both sides so that

$$g_{i+1} = g_i + \alpha_i A p_i. \tag{16}$$

In general,

$$\begin{aligned} g_i &\in B_{i+1} \subset B_{i+2} \subseteq B_k, \\ g_{i+1} &\in B_{i+2} \subseteq B_k, \end{aligned}$$

and then from Eq. 16, $Ap_i = (g_{i+1} - g_i)/\alpha_i \in B_k$, since $\alpha_i \neq 0$ if we have not reached the solution. But g_k is $\|\|_2$-orthogonal to B_k, so $(Ap_i)' g_k = 0$ for all $i \le k-2$!

Before stating the algorithm, let us consider the modifications when we start at an arbitrary $x_0 \neq 0$. The series expansion for x^* will then be

$$x^* = x_0 + \sum_{j=0}^{n-1} \alpha_j p_j, \tag{17}$$

and exactly as above,

$$\alpha_k = \frac{p_k' A (x^* - x_0)}{p_k' A p_k} = \frac{-p_k' (A x_0 - b)}{p_k' A p_k} = -\frac{p_k' g_0}{p_k' A p_k}. \tag{18}$$

The CG-algorithm may then be written:

```
Given $x = x_0$
Define $p = -g = -Ax + b$
for $k = 1 : k_{\max}$
    $\alpha = -p'g / p'Ap$
    $x = x + \alpha p$
    $g = g + \alpha A p$
    $\beta = p'Ag / p'Ap$
    $p = -g + \beta p$
end
```

The expression for β_k may be simplified in several ways, *e.g.* by utilizing that gradients are $\|\|_2$-orthogonal:

$$\beta_k = \frac{p_k' A g_{k+1}}{p_k' A p_k} = \frac{g_{k+1}' (g_{k+1} - g_k)/\alpha_k}{(-g_k + \beta_{k-1} p_{k-1})(g_{k+1} - g_k)/\alpha_k} = \frac{g_{k+1}' g_{k+1}}{g_k' g_k}. \tag{19}$$

These expressions have turned out to be useful in more general situations.

3 ERROR ANALYSIS

We know that $\{x_k\}$ is a sequence of best approximations, but how good is it really? It turns out that there exists a complete theory for the error analysis which involves some interesting mathematics.

Let us start at a general x_0 with the first search direction $p_0 = -g_0 = -(Ax_0 - b)$. Then, from Eqns. 16 and 12,

$$\begin{aligned} p_1 &= -g_1 + \beta_{k-1} p_0 \\ &= -(g_0 - \alpha_0 A g_0) - \beta_{k-1} g_0 \in \text{span}\{g_0, Ag_0\}. \end{aligned} \tag{20}$$

Continuing in the same way, it is easy to prove that

$$p_k \in \text{span}\left\{g_0,\ Ag_0,\ A^2 g_0, \cdots,\ A^k g_0\right\}. \tag{21}$$

Hence, B_k could as well be defined as

$$B_k = \text{span}\left\{g_0,\ Ag_0,\ A^2 g_0, \cdots,\ A^{k-1} g_0\right\}. \tag{22}$$

Similarly,

$$x_k - x_0 \in \text{span}\left\{g_0,\ Ag_0,\ A^2 g_0, \cdots,\ A^{k-1} g_0\right\}. \tag{23}$$

The sequence

$$g_0,\ Ag_0,\ A^2 g_0, \cdots \tag{24}$$

is called a *Krylov Sequence*, and $\{B_k\}$ the *Krylov Subspaces*, after Alexei Nikolaevich Krylov (1863–1945), famous Russian naval architect and applied mathematician.

The group of algorithms based on Krylov sequences and subspaces (of which CG is the main one) belongs to the *2000th Century Top Ten Algorithms.*

Because of (23), we can write

$$x_k - x_0 = \sum_{j=0}^{k-1} \gamma_j \left(A^j g_0\right) = \left(\sum_{j=0}^{k-1} \gamma_j A^j\right) g_0 = P_k(A)\, g_0, \tag{25}$$

where $P_k(A)$ is a matrix polynomial of degree $k-1$. Furthermore,

$$\begin{aligned} x_k - x^* &= x_k - x_0 + x_0 - x^* \\ &= P(A)\, g_0 + x_0 - x^* \qquad (26) \\ &= P(A)(Ax_0 - Ax^*) + x_0 - x^* \qquad (27) \\ &= (P(A)A + I)(x_0 - x^*) = Q_k(A)(x_0 - x^*), \end{aligned}$$

where Q_k is a matrix polynomial of degree k satisfying $Q_k(0) = 1$.

Like we did for the analysis the Steepest Descent method, we observe that A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and a corresponding set of $\|\|_2$-orthogonal, normalized, eigenvectors, $e_1, e_2, \cdots, e_n$. Recall that for $y = \sum_{j=1}^{n} t_j e_j$,

$$\|y\|_A^2 = y' A y = y' \left(\sum_{j=1}^{n} \lambda_j t_j e_j\right) = \sum_{j=1}^{n} \lambda_j t_j^2. \tag{28}$$

Similarly, observing that also $Q(A)\, e_j = Q(\lambda_j)\, e_j$,

$$\|Q(A)\, y\|_A^2 = y' Q(A)\, A Q(A)\, y = \sum_{j=1}^{n} Q^2(\lambda_j)\, \lambda_j t_j^2. \tag{29}$$

If we apply Eq. 29 to Eq. 27 with $(x_0 - x^*) = \sum_{j=1}^{n} \xi_j e_j$, we obtain

$$\|x_k - x^*\|_A^2 = \sum_{j=1}^{n} Q_k^2(\lambda_j)\, \lambda_j \xi_j^2. \tag{30}$$

This is an *exact* expression, and since the norm on the left hand side is as small as possible, it will be impossible to decrease it by choosing a different polynomial $Q(A)$. Thus, the polynomial Q_k is the optimal solution to the problem

$$Q_k = \arg \min_{\substack{\deg(Q)=k, \\ Q(0)=1}} \left(\sum_{j=1}^{n} Q^2(\lambda_j)\, \lambda_j \xi_j^2 \right). \tag{31}$$

The expression is not particularly useful since the optimal Q_k will vary for each x^*, but we can also write

$$\|x_k - x^*\|_A^2 \le \max_j Q_k^2(\lambda_j) \sum_{j=1}^{n} \lambda_j \xi_j^2 = \max_j Q_k^2(\lambda_j)\, \|x_0 - x^*\|_A^2, \tag{32}$$

which shows that the relative error of x_k is always bound by $\max_j |Q_k(\lambda_j)|$,

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \le \max_j |Q_k(\lambda_j)|. \tag{33}$$

The quite general error estimate in Eq. 33 has some surprising implications. We are free to put in any polynomial we like as long as it has degree k and $Q(0) = 1$. Thus, if A happens to have only k different eigenvalues, there exists a k-th order polynomial having those k eigenvalues as zeros and, by a suitable scaling, we can also obtain $Q(0) = 1$. Then $\|x_k - x^*\|_A = 0$, proving that the method converges to x^* in k iterations!

The following result is somewhat deeper. It considers the case when we have an idea about the extreme eigenvalues of A, λ_1 and λ_n, but not the eigenvalues in between. It is then reasonable to ask how small the right hand side of Eq. 33 could be, since we clearly have

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \le \min_{\substack{Q \text{ of order } k \\ Q(0)=1}} \left(\max_{\lambda_1 \le \lambda \le \lambda_n} |Q(\lambda)| \right). \tag{34}$$

This turns out to be a classic so-called *min-max* problem in the approximation of functions, and the solution is known to be given by

$$Q^*(\lambda) = \frac{T_k\left(\frac{2\lambda - \lambda_1 - \lambda_n}{\lambda_n - \lambda_1}\right)}{T_k\left(\frac{\lambda_1 + \lambda_n}{\lambda_1 - \lambda_n}\right)}, \tag{35}$$

where T_k is the *Chebyshev polynomial* of degree k. Look up `http://mathworld.wolfram.com` for plots and properties of the Chebyshev polynomials (This is the spelling of Chebyshev accepted by Mathworld. When I was a student, we were told that there are 52 different spellings of the name Chebyshev, but Mathworld only cites 40).

Since $\max_{-1\leq x\leq 1}|T_k(x)| = 1$, inequality 33 now leads to

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \leq \frac{1}{\left|T_k\left(\frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\right)\right|}, \tag{36}$$

and

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \leq \frac{1}{\left|T_k\left(\frac{\kappa+1}{\kappa-1}\right)\right|}, \tag{37}$$

where $\kappa = \lambda_n/\lambda_1$ is the *condition number* of A. The Chebyshev polynomials satisfy a lot of relations, and one definition is simply

$$T_k(x) = \frac{1}{2}\left(\left(x+\sqrt{x^2-1}\right)^k + \left(x-\sqrt{x^2-1}\right)^k\right). \tag{38}$$

With $x = (\kappa+1)/(\kappa-1)$, we obtain

$$x+\sqrt{x^2-1} = \frac{\left(\sqrt{\kappa}+1\right)^2}{\kappa-1}\ ,\ x-\sqrt{x^2-1} = \frac{\left(\sqrt{\kappa}-1\right)^2}{\kappa-1}, \tag{39}$$

and

$$\begin{aligned} T_k\left(\frac{\kappa+1}{\kappa-1}\right) &= \frac{1}{2}\left(\left(\frac{\left(\sqrt{\kappa}+1\right)^2}{\kappa-1}\right)^k + \left(\frac{\left(\sqrt{\kappa}-1\right)^2}{\kappa-1}\right)^k\right) \\ &= \frac{1}{2}\frac{\left(\sqrt{\kappa}+1\right)^{2k} + \left(\sqrt{\kappa}-1\right)^{2k}}{\left(\sqrt{\kappa}-1\right)^k\left(\sqrt{\kappa}+1\right)^k} \\ &= \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \frac{1}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k > \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k. \end{aligned} \tag{40}$$

In addition,

$$\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \underset{k\to\infty}{\longrightarrow} 0, \tag{41}$$

when $\kappa > 1$.

This leads finally to Eq. 5.36 in N&W, 2nd Ed. (Note: Eq. 5.35 in N&W, 1st Ed. states the result with the wrong power, the wrong constant, and the wrong definition of κ!):

$$\|x_k - x^*\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_0 - x^*\|_A\,. \tag{42}$$

Figure 1 shows the polynomials Q_k for some k-s when $\lambda_1 = 0.3$ and $\lambda_n = 2$. The maximum deviation for Q_{10} in the interval $[0.3\ ,\ 2]$ is only 5.6×10^{-4}, – truly amazing!

Thus, if we have a system of equations and know that the coefficient matrix has eigenvalues in the interval [0.3 2], the condition number is less or equal to $\kappa = 20/3$, and after 10 CG-iterations, we obtain

$$\|x_{10} - x^*\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{10} \|x_0 - x^*\|_A \approx 5.6\times 10^{-4}\|x_0 - x^*\|_A\,, \tag{43}$$

regardless the size of the system!

Penalty and barrier methods (N&W, Chapter 17) always lead to Hessians which have a few very large eigenvalues (actually tending to ∞ as the iteration proceeds), whereas the rest are clustered

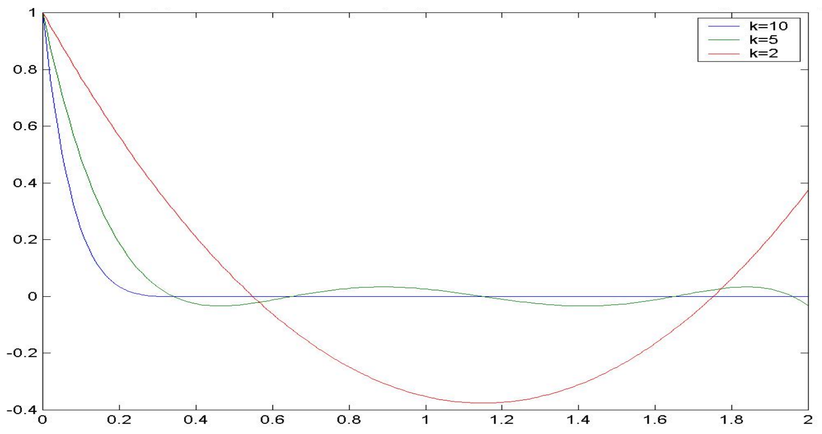

Figure 1: The optimal Chebyshev polynomials for the interval $[0.3, 2]$ when $k = 2,\ 5$ and 10.

in a fixed range. Theorem 5.5, p. 115, in N&W has some quite important implications for this case:

$$\|x_{k+1} - x^*\|_A \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right) \|x_0 - x^*\|_A \,. \tag{44}$$

The bound is thus independent of the size of $\lambda_{n-k+1}, \cdots, \lambda_n$. It is easy to derive the bound from 33 by constructing a $(k+1)$-th order polynomial with zeros at $(\lambda_1 + \lambda_{n-k})/2$, $\lambda_{n-k+1}, \cdots, \lambda_n$, when $\lambda_{n-k+1}, \cdots, \lambda_n$ are all different (try it!). We recognize (44) as similar to the Steepest Descent error bound for a matrix with extreme eigenvalues λ_1 and λ_{n-k}, and this is a key observation: *By restarting the CG method every $k+1$ iterations we obtain a convergence rate similar to the SD method, but now with a greatly reduced condition number* (counting k CG steps as one iteration).

The convergence behavior with several clustered groups of eigenvalues is illustrated on p. 116-117 in N&W.

Another good reference to the Conjugate Gradient method is the book of G. H. Golub and C. E. van Loan: *Matrix Computations*, Johns Hopkins University Press (An excellent general reference book in numerical linear algebra!).

4 SOME NUMERICAL EXPERIMENTS

It is easy to experiment with the CG-method using Matlab. In the following code we first generate a positive definite matrix $R'R$, and then the matrix $A = (R'R)^{npot}$. The power *npot* steers the eigenvalue distribution and hence the condition number (Consider the change in the eigenvalues of A when *npot* varies).

```
ndim = 100;

R = randn(ndim);

for npot = .1:.4:2

     A = (R'*R)^npot;

     lamb = eig(A);  kappa= max(lamb)/min(lamb);

     xsol = rand(ndim,1); b = A*xsol;
```

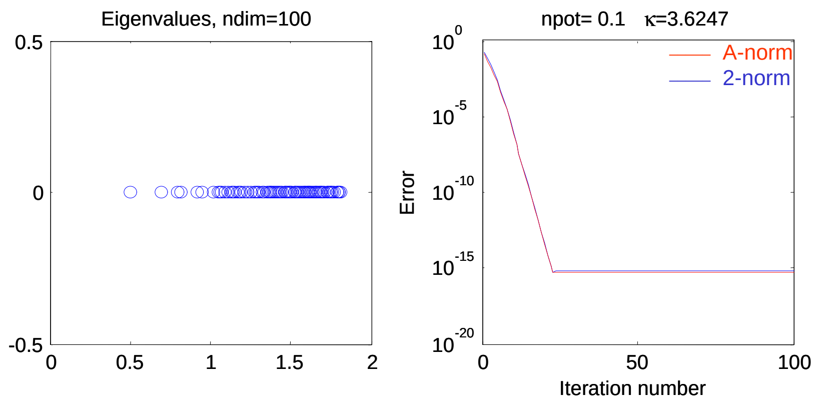

Figure 2: Well-conditioned matrix. Fast convergence.

```
        Norm2 = sqrt(xsol'*xsol); NormA = sqrt(xsol'*A*xsol);
        x = zeros(size(b)); g = Ax-b ; p = -g;
        for loop = 1:ndim
            Ap = A*p; % Only one matrix-vector product!
            alfa = -(p'*g)./(p'*Ap);
            x = x + alfa*p;
            g = g + alfa*Ap; % g = A*x-b;
            beta = (g'*Ap)./ (p'*Ap);
            p = -g + beta*p;
            err2(loop) = sqrt((x-xsol)'*(x-xsol))/Norm2;
            errA(loop) = sqrt((x-xsol)'*A*(x-xsol))/NormA;
        end;
        subplot(1,2,1); plot(lamb,zeros(size(lamb)),'o');
        Tittel = ['Eigenvalues, ndim=' num2str(ndim)];
        title(Tittel);
        subplot(1,2,2); semilogy(1:ndim, err2,1:ndim,errA);
        legend( '2-norm' , 'A-norm' );
        xlabel('Iteration number'); ylabel('Error')
        Tittel = ['npot= ' num2str(npot) '
       'kappa=',num2str(kappa)];
        title(Tittel);
        pause
end
```

Figures 2 – 4 shows some results from the simulations.

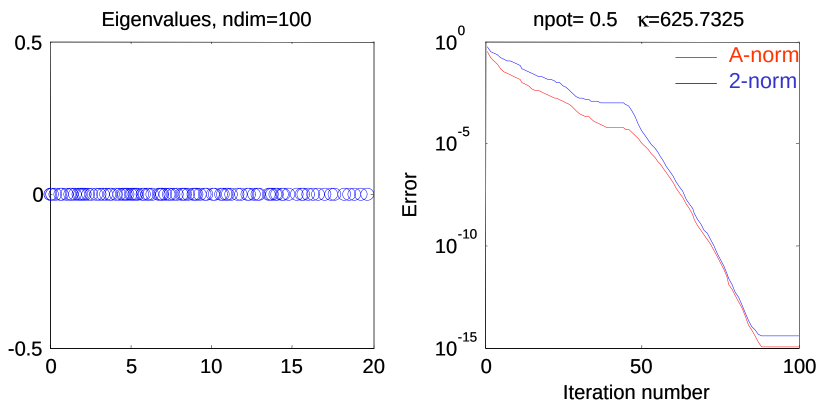

Figure 3: Medium conditioned matrix.

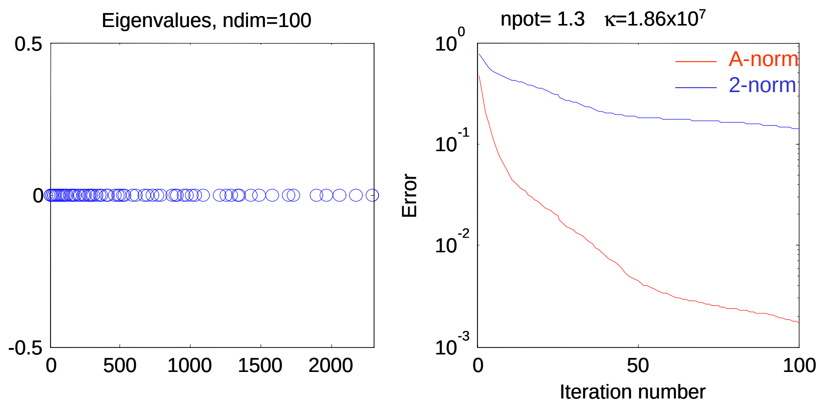

Figure 4: Ill-conditioned matrix. The convergence is very slow, and rounding errors distroy the accuracy severely.

4.1 Exercise

(a) Implement and plot the error bound in Eq. 42 in the Matlab code. How does this compare with the actual decrease of the error? (N&W say "*This bound often gives a large overestimate*" Is this true?).

(b) Modify the matrix A (say, $A \in \mathbb{R}^{100\times100}$) so that it has m (say, $m = 5$ –10) large eigenvalues by adding a rank-m matrix LL', $A = (R'R)^{npot} + \mu LL'$, $\mu \gg 1$. Here L is $n \times m$ and consists of m random column vectors. Test the performance of the CG-method. Is a SD restart every k iteration really necessary?