CPSC536

Lecture 8: Steepest descent and Conjugate Gradient

23-02-2026

Lecturer: Dr. Chen Greif *Scribe: Kevin K Thomas*

Contents

1 Introduction and motivation **1**
1.1 Fixed point methods 2
1.2 Convergence rates: 3
1.3 A sidenote on why we do the above 3

2 Steepest descent **3**

3 Conjugate Gradient Method **4**
3.1 A-conjugacy and search direction 5
3.2 Krylov subspaces 9
3.3 Convergence analysis 10

1 Introduction and motivation

The goal of this set of notes is to introduce the steepest descent algorithm and Conjugate gradient method. We motivate the algorithms by starting with the fixed point methods and their drawbacks.

Consider the problem of solving a system of linear equations:

$$Ax = b \tag{1}$$

An equivalent way of writing this as an optimization problem is to minimize the following quadratic function:

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

Throughout our discussion, we assume A is symmetric positive definite. So that the above problem is strongly convex and has a unique minimizer. i.e.

$$x^T Ax > 0 \quad \forall x \neq 0$$

One approach is to use direct solvers such as Gaussian elimination, LU etc. However, these techniques are computationally expensive and often impact the sparsity of A by introducing fill-in (zero

entries become non-zero). Many times in engineering and other applications, A is sparse (say finite difference grids for PDE's) and we only want an approximate solution x to the above problem and do not need an exact solution. This is where iterative methods come in.

1.1 Fixed point methods

The simplest iterative methods are of the form start with an initial guess x_0 and iterate as follows:

$$x_{k+1} = g(x_k)$$

One method splits A as

$$A = M - N$$

We can rewrite (1) as

$$\begin{aligned} (M - N)x &= b \\ Mx &= Nx + b \end{aligned}$$

Assuming M is invertible, this gives rise to our first iterative method:

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b = x_k + M^{-1}(b - Ax_k)$$

Denote $r_k := b - Ax_k$ the residual. So our basic iterative scheme is

$$x_{k+1} = x_k + M^{-1}r_k$$

Notice that we typically do not invert M but instead solve for p_k in $Mp_k = r_k$ and then iterate as

$$x_{k+1} = x_k + p_k$$

Hence, it is important to choose your M carefully. At one end, you want M to be as close to A as possible to decrease the number of iterations required. In the simplest case, we can take $M = A$ in which case you converge in one step! But you are basically just solving for x^* directly in that case. On the other end of the spectrum, we could take $M = I$ which makes solving for p_k trivial but convergence is really slow (and may in fact diverge based on conditioning). Based on the choice of M, we get different methods. Suppose $A = L + D + U$ where L is the strictly lower triangular part of A, D is the diagonal and U is the strictly upper triangular part, we get:

- $M = D$: The Jacobi method
- $M = L + D$: The Gauss-Seidel method

1.2 Convergence rates:

It can be shown that the iterative method

$$x_{k+1} = x_k + M^{-1} r_k$$

converges iff the iteration matrix $T = I - M^{-1}A$, (comes from expanding $r_k = b - Ax_k$ and combining Ax_k) has spectral radius

$$\rho(T) = \max_i |\lambda_i| < 1$$

It can also be shown that if $\rho(T) < 1$, the rate of convergence is linear with factor:

$$\|e_k\| \leq \rho(T)^k \|e_0\|$$

so smaller spectral radius lends to faster convergence.

1.3 A sidenote on why we do the above

Define the error at iteration k as $e_k = x^* - x_k$. Then, notice that

$$Ae_k = b - Ax_k = r_k$$

Obviously, we do not have access to the error e_k as if we did, we could trivially compute $x^* = e_0 + x_0$. Compare the above with our iterative scheme:

$$Mp_k = r_k; \qquad x_{k+1} = x_k + p_k$$

So we can think of iterative schemes as finding $p_k \sim e_k$.

2 Steepest descent

The fixed point methods described above are really slow. Instead of using a fixed M, we could use adaptive methods to speed up the convergence. This motivates steepest descent and the state of the art Conjugate gradient method. In this section, we iterate as follows:

$$x_{k+1} = x_k + \alpha_k p_k$$

here p_k is the search direction and α_k is the step size. Steepest descent is basically gradient descent for solving the problem $Ax = b$. Consider the optimization problem

$$\min_x f(x) := \frac{1}{2} x^T Ax - b^T x \tag{3}$$

And given that $A \succ 0$, the above problem is strongly convex and from the optimality condition we have

$$\nabla f(x^*) = Ax^* - b = 0$$

Hence, minimizing the problem above is exactly the same as solving the linear system $Ax = b$. Now, if we apply gradient descent to f, we get, $\nabla f(x_k) = Ax_k - b = -r_k$

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) = x_k + \alpha_k r_k$$

Now, we want to choose an α_k that tends to the largest decrease. Define

$$g(\alpha) = f(x_k + \alpha r_k)$$

we want the α that minimizes f in the step direction we took. Notice, g is convex as it is the composition of an affine function in α and a convex function f. If we expand the definition of $f(x_k + \alpha r_k)$ then use the first order optimality condition of g using the derivative with respect to α, we obtain the optimal α:

$$\alpha^* = \frac{r_k^T r_k}{r_k^T A r_k}$$

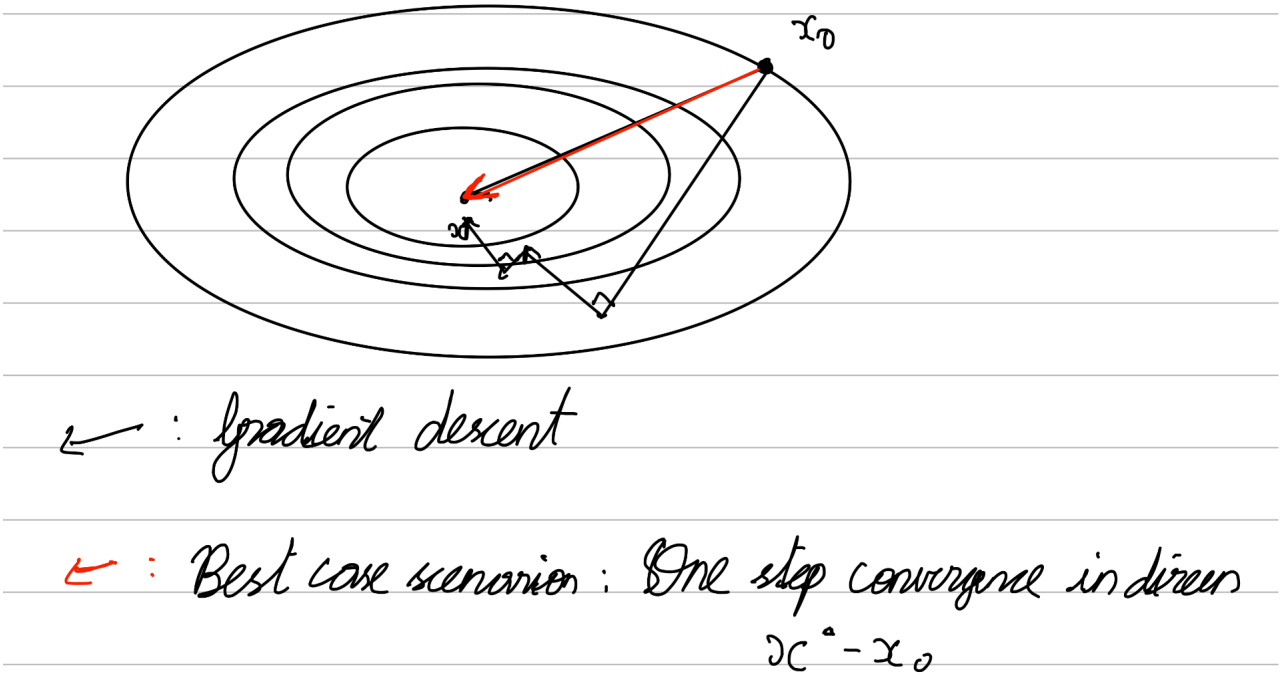
Hence, the steepest descent algorithm iterates as:

$$x_{k+1} = x_k + \frac{r_k^T r_k}{r_k^T A r_k} r_k \tag{4}$$

3 Conjugate Gradient Method

We now proceed to the conjugate gradient method. Steepest descent while good is still too slow for our purpose. This gives rise to the conjugate gradient method which does a clever choice in the search direction. In the steepest descent algorithm, at each step, our search direction is the residual $r_k = b - Ax_k$ which is the normal to the level curve of $f(x)$ defined earlier. While this is a good direction to take, we can do better which lends to the conjugate gradient method.

Suppose we are in an ideal situation and we know x^*, in that case, starting from an x_0, the best direction to take to reach the minimizer is along the vector $x^* - x_0$. In fact, we will be able to converge in one iteration! However, we being lowly optimizers do not have access to the true solution :(.



In the above diagram, if we had access to the vector $x^* - x_0$, we could converge in one step! But obviously, we do not. Conjugate gradient uses some clever machinery to closely mimic this step direction.

3.1 A-conjugacy and search direction

Consider our original iteration step as in steepest descent

$$x_{k+1} = x_k + \alpha_k p_k$$

where α_k is our step and p_k is the search direction. Let us first compute the optimal step size like before. We minimize

$$g(\alpha) = f(x_k + \alpha p_k)$$

in α. If we set the derivative to 0 and solve, we obtain:

$$0 = g'(\alpha) = p_k^T(Ax_k - b) + \alpha p_k^T A p_k.$$

Using the residual $r_k = b - Ax_k$, we obtain

$$0 = g'(\alpha) = -p_k^T r_k + \alpha p_k^T A p_k.$$

On solving, we get

$$\alpha_k^* = \frac{r_k^T p_k}{p_k^T A p_k} = \frac{\langle p_k, r_k \rangle}{\langle p_k, Ap_k \rangle}$$

Now, let us try to find a good search direction p_k. Notice,

$$x_{k+1} = x_k + \alpha_k p_k$$

Multiply throughout by $-A$ and adding b, we then get:

$$r_{k+1} = r_k - \alpha_k A p_k$$

Multiply by p_k^T,

$$\langle p_k, r_{k+1} \rangle = \langle p_k, r_k \rangle - \alpha_k \langle p_k, A p_k \rangle$$

But from the optimal choice of α_k, we get that the RHS of the above is 0 (just substitute it in). Hence we have,

$$0 = \langle p_k, r_{k+1} \rangle = \langle p_k, A e_{k+1} \rangle \tag{5}$$

But this tells us that the error vector (which would be the ideal step direction) satisfies $\langle p_k, e_{k+1} \rangle = 0$. We define the following notion of conjugacy:

Definition 3.1. A-conjugate: Two vectors x and y are said to be A-conjugate if $\langle x, Ay \rangle = 0$.

We also define the following weighted norm which will be useful for our analysis later:

Definition 3.2. Energy norm (Weighted norm): Given a positive semi-definite matrix A, the energy norm of a vector x is given as

$$\|x\|_A = \sqrt{\langle x, Ax \rangle}$$

One reason why we care about A-conjugacy is given in (5). Our ideal step direction e_{k+1} is A-conjugate to the search direction p_k. Hence, if we can find search directions that are A-conjugate to each other, then we can mimic the ideal step direction and converge much faster. Additionally, notice that

$$\begin{aligned}
f(x) - f(x^*) &= \frac{1}{2} x^T A x - b^T x - \left(\frac{1}{2} x^{*T} A x^* - b^T x^* \right) \\
&= \frac{1}{2} x^T A x - (Ax^*)^T x + \frac{1}{2} x^{*T} A x^* \\
&= \frac{1}{2} x^T A x - x^{*T} A x + \frac{1}{2} x^{*T} A x^* \\
&= \frac{1}{2} (x - x^*)^T A (x - x^*) \\
&= \frac{1}{2} e^T A e \\
&= \frac{1}{2} \|e\|_A^2
\end{aligned} \tag{6}$$

Hence, minimizing the function value is the same as minimizing the error in the A norm. This is the main idea behind CG. At each step, we find a search direction p_k that is A-conjugate to all previous search directions. So a naive implementation of the algorithm proceeds as follows:

1. Start with an initial guess x_0 and compute the residual

$$r_0 = b - Ax_0.$$

   Set the first search direction

$$p_0 = r_0.$$

2. Construct search directions that are A-orthogonal using Gram–Schmidt:

$$p_k = r_k - \sum_{j=0}^{k-1} \frac{\langle r_k, Ap_j \rangle}{\langle p_j, Ap_j \rangle} p_j.$$

3. Compute the step size

$$\alpha_k = \frac{\langle r_k, p_k \rangle}{\langle p_k, Ap_k \rangle}.$$

4. Update

$$x_{k+1} = x_k + \alpha_k p_k$$

5. Update the residual

$$r_{k+1} = r_k - \alpha_k A p_k.$$

This method converges in at most n steps in the worst case. That is when $\{p_k\}_{k=1}^n$ spans the entire space $\mathbb{R}^n$. Of course if at any point a $p_k = 0$, we have found the solution as $r_k = 0$ then. Now I shall prove why we do not need a full Gram-Schmidt.
But before that, I prove the following claim:

Lemma 3.3. *The algorithm above gives iterates where*

$$x_k = \arg \min_{x_0 + \textit{span}\, \{p_0, p_1, \ldots, p_{k-1}\}} f(x)$$

Proof. From first order optimality condition for constrained optimization, we have that x_{k+1} is the minimizer of f over the space $x_0 + \text{span}\, \{p_0, p_1, \ldots, p_k\}$ iff $\nabla f(x_{k+1})$ is in the orthogonal complement of span $\{p_0, p_1, \ldots, p_k\}$. Or equivalently, it suffices to show that $\langle \nabla f(x_{k+1}), p_i \rangle = 0$ for all $0 \leq i \leq k$. Notice,

$$\nabla f(x_{k+1}) = A(x_{k+1} - x^*) = -r_{k+1}$$

Hence, we want to show that $\langle r_{k+1}, p_i \rangle = 0$ for all $0 \leq i \leq k$. (5) shows that $\langle r_{k+1}, p_k \rangle = 0$. Now we proceed by induction.
Base case: $k = 0$: We have $p_0 = r_0$ and hence

$$\begin{aligned} \langle r_1, p_0 \rangle &= \langle r_0 - \alpha_0 A p_0, p_0 \rangle \\ &= \langle r_0, p_0 \rangle - \alpha_0 \langle A p_0, p_0 \rangle \\ &= 0 \qquad \text{(by definition of } \alpha_0) \end{aligned}$$

Induction step: Assume $\langle r_k, p_i \rangle = 0$ for all $0 \leq i \leq k-1$. We want to show that $\langle r_{k+1}, p_i \rangle = 0$ for all $0 \leq i \leq k$. We have already shown that $\langle r_{k+1}, p_k \rangle = 0$. Now, for $0 \leq i \leq k-1$, we have

$$\begin{aligned}\langle r_{k+1}, p_i \rangle &= \langle r_k - \alpha_k A p_k, p_i \rangle \\ &= \langle r_k, p_i \rangle - \alpha_k \langle A p_k, p_i \rangle \\ &= 0\end{aligned}$$

(by induction hypothesis and using the construction of p_k from the Gram-Schmidt step)

□

Now I claim that we do not need a full Gram-Schmidt process to construct the search directions.

Lemma 3.4. *The search direction p_k can be constructed using only the previous search direction p_{k-1} and the current residual r_k. We do not need the full Gram-Schmidt process:*

$$p_k = r_k - \sum_{j=0}^{k-1} \frac{\langle r_k, A p_j \rangle}{\langle p_j, A p_j \rangle} p_j.$$

Instead we can do:

$$p_k = r_k - \frac{\langle r_k, A p_{k-1} \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} p_{k-1}$$

Proof. It suffices to show that in the full Gram-Schmidt $\langle r_k, A p_j \rangle = 0$ for all $0 \leq j \leq k-2$. We already showed earlier that $\langle r_k, p_j \rangle = 0$ for all $0 \leq j \leq k-1$. So it suffices to show that $A p_j \in \text{span}\{p_0, p_1, \ldots, p_{k-1}\}$. Notice,

$$r_{j+1} = r_j - \alpha_j A p_j$$

Hence, we can rewrite $A p_j$ as

$$A p_j = \frac{1}{\alpha_j}(r_j - r_{j+1})$$

So now it suffices to show $r_j \in \text{span}\{p_0, p_1, \ldots, p_j\}$. Notice, from the full Gram-Schmidt process, we have

$$p_{j+1} = r_{j+1} - \sum_{i=0}^{j} \frac{\langle r_{j+1}, A p_i \rangle}{\langle p_i, A p_i \rangle} p_i$$

Hence, we can rewrite r_{j+1} as

$$r_{j+1} = p_{j+1} + \sum_{i=0}^{j} \frac{\langle r_{j+1}, A p_i \rangle}{\langle p_i, A p_i \rangle} p_i$$

Hence, $r_{j+1} \in \text{span}\{p_0, p_1, \ldots, p_{j+1}\}$. Therefore, we have shown that $A p_j \in \text{span}\{p_0, p_1, \ldots, p_{j+1}\} \subset \text{span}\{p_0, p_1, \ldots, p_{k-1}\}$ and hence $\langle r_k, A p_j \rangle = 0$ for all $0 \leq j \leq k-2$. This gives us the desired result. □

Hence, we get the new update rule:

$$p_{k+1} = r_{k+1} + \beta_k p_k$$

where β_k is given by

$$\beta_k = -\frac{\langle r_{k+1}, Ap_k \rangle}{\langle p_k, Ap_k \rangle}$$

This gives us the standard Conjugate gradient algorithm.

1. Initialize
$$x_0 \text{ given}, \quad r_0 = b - Ax_0, \quad p_0 = r_0.$$
2. For $k = 0, 1, 2, \ldots$
$$\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle p_k, Ap_k \rangle}$$
$$x_{k+1} = x_k + \alpha_k p_k$$
$$r_{k+1} = r_k - \alpha_k A p_k$$
$$\beta_k = -\frac{\langle r_{k+1}, Ap_k \rangle}{\langle p_k, Ap_k \rangle}$$
$$p_{k+1} = r_{k+1} + \beta_k p_k$$

3.2 Krylov subspaces

In order to analyze the conjugate gradient method, we have to introduce the notion of Krylov subspaces.

Definition 3.5. Krylov subspace: Given a matrix A and a vector r_0, the Krylov subspace of order k is defined as

$$K_k(A; r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\}$$

We now form a connection between Krylov subspaces and the CG method. From the CG update rule, we have

$$x_{k+1} = x_k + \alpha_k p_k,$$

We expand recursively:

$$x_k = x_{k-1} + \alpha_{k-1} p_{k-1} = x_{k-2} + \alpha_{k-2} p_{k-2} + \alpha_{k-1} p_{k-1} = \cdots = x_0 + \sum_{j=0}^{k-1} \alpha_j p_j.$$

Hence

$$x_k \in x_0 + \text{span}\{p_0, p_1, \ldots, p_{k-1}\}.$$

It can be shown that $\text{span}\{p_0, p_1, \ldots, p_{k-1}\}$ forms a A-orthogonal basis for the Krylov subspace:

$$K_k(A; r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\}$$

We now show that the search directions $\text{span}\{p_0, p_1, \ldots, p_{k-1}\}$ form an A-orthogonal basis for the Krylov subspace $K_k(A; r_0)$. We know

$$p_k = r_k + \beta_{k-1} p_{k-1}$$

Unrolling this recursively, we have that p_k is in the span$\{r_0, r_1, \ldots, r_{k-1}, r_k\}$. So it suffices to show that $r_k \in K_{k+1}(A; r_0)$. We proceed by induction.
Base case: $k = 0$: We have $r_0 \in K_1(A; r_0) = \text{span}\{r_0\}$ which is true.
Induction step: Assume $r_k \in K_{k+1}(A; r_0)$. We want to show that $r_{k+1} \in K_{k+2}(A; r_0)$. Notice from the update rule of p_k we have

$$p_k = r_k + \beta_{k-1} p_{k-1}$$

Inductively, this means that $p_k \in K_{k+1}(A; r_0)$ (Using the fact $r_k \in K_{k+1}(A; r_0)$ and $p_{k-1} \in K_k(A; r_0) \subset K_{k+1}(A; r_0)$). Applying A to this gives:

$$Ap_k \in K_{k+2}(A; r_0)$$

Now, notice from the update rule of r_k we have

$$r_{k+1} = r_k - \alpha_k A p_k$$

Hence, from the above two equations, we have

$$r_{k+1} \in K_{k+2}(A; r_0)$$

This gives us the desired result.
So we are searching for solutions of the form:

$$x_k \in x_0 + K_k(A; r_0)$$

3.3 Convergence analysis

From our earlier discussion, we know

$$x_k \in x_0 + K_k(A; r_0)$$

So we can write

$$x_{k+1} = x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + \cdots + \gamma_k A^k r_0$$

Define the polynomial,

$$P_k(A) = \gamma_0 I + \gamma_1 A + \cdots + \gamma_k A^k$$

So we can rewrite

$$x_{k+1} = x_0 + P_k(A) r_0$$

We now have the following theorem regarding the convergence rate of CG

Theorem 3.6. *Let A be a symmetric positive definite matrix, and consider solving the problem $Ax = b$ using the conjugate gradient method. Let $e_k = x^* - x_k$. Then the error vector e_k satisfies,*

$$\|e_k\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|e_0\|_A$$

where $\kappa(A)$ is the condition number and for SPD matrices it is given by:

$$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

Proof. We shall first rewrite the error vector as a polynomial similar to x_{k+1}. Notice that $r_0 = b - Ax_0 = A(x^* - x_0) = Ae_0$. Hence, we have,

$$e_k = x^* - x_k = x^* - x_0 - P_{k-1}(A) r_0 = e_0 - P_{k-1}(A) A e_0 = (I - P_{k-1}(A) A) e_0$$

Define the polynomial $q_k(A) = I - q_{k-1}(A) A$ q_k has degree atmost k and $q_k(0) = I$ So,

$$e_k = q_k(A) e_0$$

Now, from our earlier discussion in (6), we have that minimizing the function value is the same as minimizing the error in the A norm.
Hence, we can rewrite the error in terms of the function value as

$$f(x_k) - f(x^*) = \frac{1}{2} \|e_k\|_A^2$$

Now, at each iteration, we are minimizing $f(x)$ over the space $x_0 + K_k(A, r_0)$ which is the same as minimizing the error in the A norm over the same space. Hence, we have

$$\|e_k\|_A = \min_{x \in x_0 + K_k(A, r_0)} \|x^* - x\|_A$$

Using the polynomial representation of the error gives

$$\|e_k\|_A = \min_{\substack{q_k \in \Pi_k \\ q_k(0) = 1}} \|q_k(A) e_0\|_A$$

Now, notice that q_k is a polynomial of degree atmost k and $q_k(0) = I$. Hence, we can write

$$\begin{aligned}
\|e_k\|_A^2 &= \min_{\substack{q_k \in \Pi_k \\ q_k(0)=1}} \|q_k(A)e_0\|_A^2 \\
&= \min_{\substack{q_k \in \Pi_k \\ q_k(0)=1}} \sum_{i=1}^{n} \lambda_i q_k(\lambda_i)^2 \langle e_0, u_i \rangle^2 \qquad \text{(where } u_i \text{ are the eigenvectors of } A\text{)} \\
&\leq \min_{\substack{q_k \in \Pi_k \\ q_k(0)=1}} \max_i |q_k(\lambda_i)|^2 \sum_{i=1}^{n} \lambda_i \langle e_0, u_i \rangle^2 \\
&= \min_{\substack{q_k \in \Pi_k \\ q_k(0)=1}} \max_i |q_k(\lambda_i)|^2 \|e_0\|_A^2
\end{aligned}$$

From theorem 6.25 and theorem 6.29 in [2], we have that the optimal polynomial q_k that minimizes $\max_i |q_k(\lambda_i)|$ is given by a Chebyshev polynomial. Using that result, we can conclude the rate of convergence:

$$\|e_k\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|e_0\|_A$$

□

Remark: I used [3], [2], [1] and Dr. Greif's lecture notes to write this set of notes.

References

[1] J. NOCEDAL AND S. J. WRIGHT, *Numerical optimization*, Springer, 2006.

[2] Y. SAAD, *Iterative methods for sparse linear systems*, SIAM, 2003.

[3] J. R. SHEWCHUK ET AL., *An introduction to the conjugate gradient method without the agonizing pain*, 1994.