Conjugate Gradient

Holden Lee

February 25, 2020

Good references for the conjugate gradient method are:

- `https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`
- $Lx = b$, Chapter 16: `https://theory.epfl.ch/vishnoi/Lxb-Web.pdf`

1 Introduction

Our goal is to solve the system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ are given. We assume that A is symmetric positive definite. (See Section 5 for how to deal with general A.) Then the solution x also satisfies

$$x = \operatorname{argmin}_{x\in\mathbb{R}^n} \underbrace{\left(\frac{1}{2}x^\top Ax - b^\top x\right)}_{f(x)}.$$

The method of **gradient descent** (or steepest descent) works by letting

$$x_{k+1} = x_k - \alpha_k \nabla f(x) = x_k + \alpha_k \underbrace{(b - Ax_k)}_{r_k}$$

for some step size α_k to be chosen. Here $-\nabla f(x)$ is the direction of steepest descent, and by calculation it equals the residual $r_k = b - Ax_k$. The step size α_k can be fixed, or it can be chosen to minimize $f(x_{k+1})$. In this case, we arrive at the following algorithm (not optimized for efficiency):

Algorithm 1 Gradient descent for solving $Ax = b$

Input: Symmetric positive definite $A \in \mathbb{R}^{n\times n}$, vector $b \in \mathbb{R}^n$, initial value x_0
for $k = 0, 1, \dots$ **do**
  Let $r_k = b - Ax_k$.
  Let $\alpha_k = \frac{r_k^\top r_k}{r_k^\top A r_k}$.
  Let $x_{k+1} = x_k + \alpha_k r_k$.
end for

To see the choice of α_k, we note that for any p_k, (we will take $p_k = r_k$, but we do the calculation more generally)

$$\begin{aligned} f(x_k + \alpha p_k) &= \frac{1}{2}(x_k + \alpha p_k)^\top A (x_k + \alpha p_k) - b^\top (x_k + \alpha p_k) \\ &= \frac{1}{2} p_k^\top A p_k \alpha^2 + \alpha p_k^\top (A x_k - b) + \cdots \end{aligned}$$

where the rest of the terms do not contain α. This is a quadratic at α, which is minimized at

$$-\frac{p_k^\top (A x_k - b)}{p_k^\top A p_k} = \frac{p_k^\top r_k}{p_k^\top A p_k}. \tag{1}$$

In our case, $p_k = r_k$, so we choose $\alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k}$.

Unfortunately, gradient descent can converge slowly when A has large condition number.

Theorem 1.1 (Convergence of gradient descent)**.** *Let $\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ denote the condition number of A. Then in Algorithm 2,*

$$\|x_n - x\|_A \le \left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right)^n \|x_0 - x\|_A$$

where $\|x\|_A := (x^\top A x)^{\frac{1}{2}}$ is the A-norm.

This ratio is $1 - O\left(\frac{1}{\kappa(A)}\right)$. The proof of Theorem 1.1 is somewhat involved. However, when a bound for $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ is known, a fixed step can be chosen which essentially attains the same bound; see the homework.

What can go wrong is that gradient descent can oscillate. Consider $A = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$, $b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, started at $x_0 = \begin{pmatrix} 3 \\ 1/3 \end{pmatrix}$. The matrix A is ill-conditioned with $\kappa(A) = 9$. The function $\frac{1}{2} x^\top A x$ is like a trough: shallow in the x direction and steep in the y direction. The solution is $x = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$. The iterates bounce back and forth in the trough and make little progress in the shallow direction, the x-direction. This kind of oscillation makes gradient descent impractical for solving $Ax = b$.

We would like to fix gradient descent. Consider a general iterative method in the form

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

where $p_k \in \mathbb{R}^n$ is the *search direction*. For example, in gradient descent, p_k is the residual $r_k = b - A x_k$. Let's dream big: instead of x_{k+1} just being the best point of the form $x_k + \alpha_k p_k$ for minimizing $f(x)$, we would like x_k to be the best point of the form $x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_k p_k$: in the entire x_0 plus the subspace generated by $p_0, \ldots, p_k$. In the case of $A \in \mathbb{R}^{2\times 2}$, this means that x_k converges to the solution in 2 iterations, and in general, for $A \in \mathbb{R}^{n\times n}$, it will converge to the solution in n iterations (if it has not converged in $n-1$ iterations, the first n search directions will span the whole space). This is certainly not satisfied by gradient

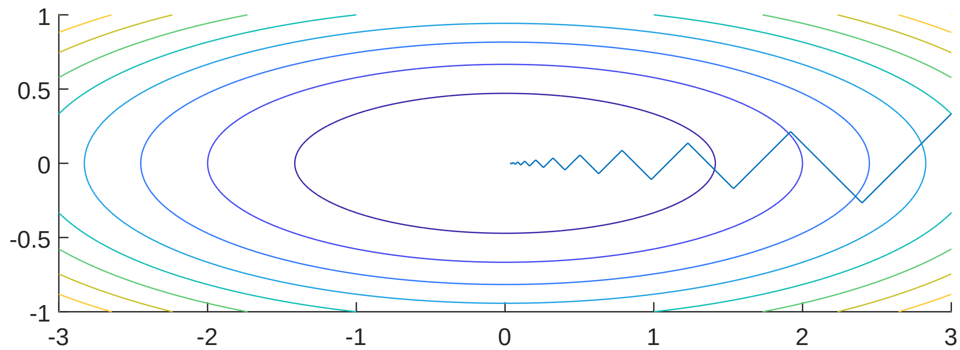

Figure 1: Gradient descent for $A = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$, $b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $x_0 = \begin{pmatrix} 3 \\ 1/3 \end{pmatrix}$ converges slowly because $\kappa(A)$ is large. Note that at each point x_{k+1}, the search direction r_k is tangent to the contour lines.

descent: in our problem, after 2 steps, it's still far from the solution; it overshot in the p_0 direction, and backtracked (too much).

How can we ensure that x_k is the best point in the form $x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_k p_k$? We can ensure this if the p_i are A**-orthogonal**: that is, $p_i A^\top p_j$ for $i \neq j$. To see this, note that this *decouples* the optimization problem: for $x_{k+1} = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_k p_k$, we have

$$\frac{1}{2} x_{k+1}^\top A x_{k+1} - b^\top x_{k+1} = \frac{1}{2} x_0^\top A x_0 - b_0^\top x_0 + \sum_{i=0}^{k} \left(\frac{1}{2} \alpha_i^2 p_i^\top A p_i + \alpha_i p_i^\top (A x_0 - b) \right). \tag{2}$$

Thus, if $x_k \in x_0 + \operatorname{span}\{p_0, \ldots, p_{k-1}\}$ was chosen to minimize $f(x)$, then choosing $x_{k+1} = x_k + \alpha_k p_k$ to minimize $f(x)$, is the same as choosing $x_{k+1} \in x_0 + \operatorname{span}\{p_0, \ldots, p_{k-1}, p_k\}$ to minimize $f(x)$. Progress in new directions does not undo progress in old directions.

Conjugate gradient chooses the search directions to be A-orthogonal. For this, we will need some background: how to convert an arbitrary basis into an orthogonal basis using Gram-Schmidt, and how to modify this to get an A-orthogonal basis.

2 Gram-Schmidt Orthogonalization

Given vectors $a_1, \ldots, a_n \in \mathbb{R}^n$ forming a basis, we would like a procedure that creates a basis of *orthogonal* vectors $q_1, \ldots, q_n$ such that each q_k is a linear combination of $a_1, \ldots, a_k$:

$$q_k = b_{1k} a_1 + \cdots + b_{kk} a_k.$$

for some $b_{1k}, \ldots, b_{kk}$. Note that this can also be expressed in matrix form as

$$\begin{bmatrix} | & & | \\ q_1 & \cdots & q_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & b_{nn} \end{bmatrix}$$

$$Q = AB$$

for some upper triangular matrix B. Note that because $q_1, \dots, q_n$ form a basis, Q is nonsingular, so B must be nonsingular. By letting $R = B^{-1}$, we can also write this in the form

$$A = QR.$$

Since B is upper-triangular, R is also upper-triangular; this instead expresses a_k as a linear combination of the orthogonal vectors $q_1, \dots, q_k$:

$$\begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_n \\ | & & | \end{bmatrix} \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & r_{nn} \end{bmatrix}$$

Algorithm 2 Gram-Schmidt Orthogonalization

```
Input: Basis a_1, ..., a_n ∈ R^n
for k = 1 to n do
    Let q_k = a_k − Σ_{i=1}^{k−1} (⟨a_k, q_i⟩ / ⟨q_i, q_i⟩) q_i.
end for
```

Theorem 2.1. *Given a basis $a_1, \dots, a_n \in \mathbb{R}^n$, Algorithm 2 produces an orthogonal basis $q_1, \dots, q_n$, such that each q_k is a linear combination of $a_1, \dots, a_k$.*

Proof. We would like to define $q_k = a_k + \sum_{j=1}^{k} b_{kj} q_j$ for some b_{kj}, but what b_{kj} should we choose? We would like the result to be orthogonal to all $q_1, \dots, q_{k-1}$. Taking the inner product with q_i, $i < k$ gives

$$\begin{aligned} \langle q_k, q_i \rangle &= \langle a_k, q_i \rangle + \sum_{j=1}^{k} b_{kj} \langle q_j, q_i \rangle \\ &= \langle a_k, q_j \rangle + b_{ki} \langle q_i, q_i \rangle \end{aligned}$$

because by orthogonality, $\langle q_i, q_j \rangle = 0$ for $i \neq j$. Thus, to make $\langle q_k, q_j \rangle = 0$, we take

$$b_{ki} = -\frac{\langle q_k, q_i \rangle}{\langle q_i, q_i \rangle}.$$

Then q_k is orthogonal to $q_1, \dots, q_{k-1}$. q_k is exactly defined using these coefficients b_{ki}.

Finally, note that $q_k \neq 0$. Indeed, if $q_k = 0$, then a_k is a linear combination of $q_1, \dots, q_{k-1}$, $a_k = -\sum_{i=1}^{k} b_{ki} q_i$. But $q_1, \dots, q_{k-1}$ are a linear combination of $a_1, \dots, a_{k-1}$, so a_k is not linearly independent of $a_1, \dots, a_{k-1}$, contradicting the fact that $a_1, \dots, a_n$ forms a basis. ☐

3 Inner products

Definition 3.1: An **inner product** on a (real) vector space V is a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ that satisfies the following:

1. $\langle \cdot, \cdot \rangle$ is symmetric: for all $x, y \in V$, $\langle x, y \rangle = \langle y, x \rangle$.
2. $\langle \cdot, \cdot \rangle$ is a bilinear form: for all $x, y, z \in V$ and $a \in \mathbb{R}$, $\langle ax + z, y \rangle = a \langle x, y \rangle + \langle z, y \rangle$ and $\langle x, ay + z \rangle = a \langle x, y \rangle + \langle x, z \rangle$.
3. $\langle x, x \rangle = 0$ only if $x = 0$.

An inner product defines a norm $\|x\| = \langle x, x \rangle^{\frac{1}{2}}$.

On $\mathbb{R}^n$, the usual dot product $\langle x, y \rangle = x^\top y$ is an inner product, but it is not the only one. We can define an inner product with respect to any symmetric positive definite matrix A.

Definition 3.2: Given a symmetric positive definite matrix A, define the inner product with respect to A by

$$\langle x, x \rangle_A = \langle x, Ax \rangle = x^\top Ax$$

and define the norm with respect to A by

$$\|x\|_A = \langle x, x \rangle_A^{\frac{1}{2}} = (x^\top Ax)^{\frac{1}{2}}.$$

Note that Gram-Schmidt Orthogonalization works with any inner product, not just the standard one $\langle x, y \rangle = x^\top y$. Indeed, we can verify that the proof of Theorem 2.1 only depends on the properties of $\langle \cdot, \cdot \rangle$ in Definition 3.1, and not on it being exactly $\langle x, y \rangle = x^\top y$. Thus, we can create an A-*orthogonal basis* $q_1, \ldots, q_n$, i.e., a basis such that $\langle q_i, Aq_j \rangle = 0$ for $i \neq j$, by letting

$$q_k = a_k - \sum_{i=1}^{k-1} \frac{\langle a_k, q_i \rangle_A}{\langle q_i, q_i \rangle_A} q_i = a_k - \sum_{i=1}^{k-1} \frac{\langle a_k, Aq_i \rangle}{\langle q_i, Aq_i \rangle} q_i.$$

4 Conjugate gradient method

We would like the search directions to be A-orthogonal. The natural search direction at the kth step is the residual $r_k = b - Ax_k$. However, the residuals $r_0, \ldots, r_k$ are not A-orthogonal to each other. Let's use Gram-Schmidt on the $r_0, \ldots, r_k$ to obtain $p_0, \ldots, p_k$, and use these as the search directions. Note $p_0, \ldots, p_{k-1}$ are computed with only knowledge of $r_0, \ldots, r_{k-1}$, so at the kth step, we just need to apply Gram-Schmidt to compute p_k.

We first characterize the subspace spanned by the r_k's (which is also the subspace spanned by the p_k's).

Definition 4.1: For $A \in \mathbb{R}^{n \times n}$ and $y \in \mathbb{R}^n$, define the n**th Krylov subspace** $\mathcal{K}_k(A; y) := \text{span}\{y, Ay, A^2, \ldots, A^{k-1}y\}$.

Proposition 4.2: The residual r_k is in the Krylov subspace

$$\mathcal{K}_{k+1}(A; r_0) = \text{span}\{y, Ar_0, A^2 r_0, \ldots, A^k r_0\}.$$

Thus, if $r_k \neq 0$, $\{p_0, \ldots, p_k\}$ forms a basis for $\mathcal{K}_{k+1}(A; r_0)$, and for each k, $x_k - x_0 \in \mathcal{K}_k(A; r_0)$.

Algorithm 3 Conjugate gradient method for solving $Ax = b$ (not optimized)

Input: Symmetric positive definite $A \in \mathbb{R}^{n\times n}$, vector $b \in \mathbb{R}^n$, initial value x_0
Let $p_0 = r_0 = b - Ax_0$.
for $k = 0, 1, \dots$ **do**
    Let $\alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k}$.
    Let $x_{k+1} = x_k + \alpha_k p_k$.
    Let $r_{k+1} = b - Ax_{k+1}$.
    Let $p_{k+1} = r_{k+1} + \frac{r_k^\top r_k}{r_{k+1}^\top r_{k+1}} p_k$.
end for

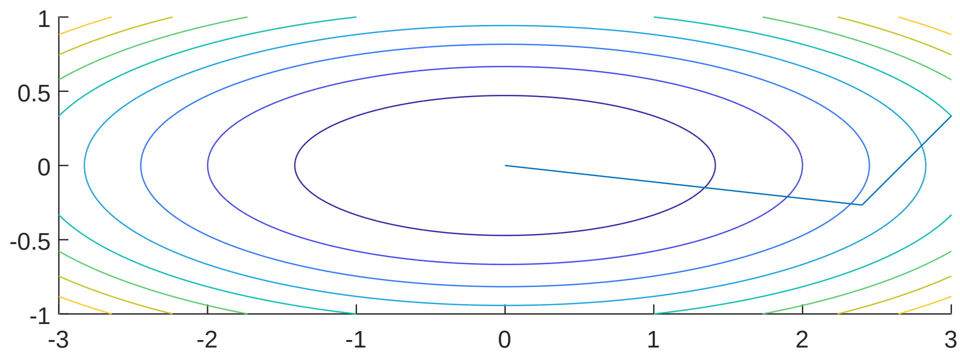

Figure 2: In the 2-D problem with $A = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$, $b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $x_0 = \begin{pmatrix} 3 \\ 1/3 \end{pmatrix}$, Conjugate Gradient converges in 2 steps.

Proof. Note that $A \cdot \mathcal{K}_k(A; r_0) \subseteq \mathcal{K}_{k+1}(A; r_0)$: that is, if $z \in \mathcal{K}_k(A; y)$, then $Az \in \mathcal{K}_{k+1}(A; y)$.

The claim is true for $k = 0$. We proceed by induction. If it's true for k, then

$$\begin{aligned} r_{k+1} &= b - Ax_{k+1} \\ &= b - A(x_k + \alpha_k p_k) \\ &= r_k - \alpha_k A p_k. \end{aligned}$$

Since $p_{k+1} \in \mathcal{K}_k(A; r_0)$, we have $Ap_k \in \mathcal{K}_{k+2}(A; r_0)$, and so $r_{k+1} \in \mathcal{K}_{k+1}(A; r_0)$. Note this does not depend on how α_k is defined! $\square$

We can summarize Conjugate Gradient in a line as: at each step k, go to the minimizer of $f(x) = \frac{1}{2} x^\top A x - b^\top x$ in the subspace $x_0 + \mathcal{K}_k(A; r_0)$.

At step k, move to the f-minimizer in $x_0 + \mathcal{K}_k(A; r_0)$.

Lemma 4.3. *The following hold.*

1. $x_k = \operatorname{argmin}_{x \in x_0 + \mathcal{K}_k(A; r_0)} f(x)$.

2. *The residual* r_k *is orthogonal (in the usual sense) to* $\mathcal{K}_k(A; r_0)$, *and hence to* $p_0, \dots, p_{k-1}$ *and* $r_0, \dots, r_{k-1}$.

3. *The residual r_k is A-orthogonal to $\mathcal{K}_{k-1}(A; r_0)$, and hence to $p_0, \ldots, p_{k-2}$ and $r_0, \ldots, r_{k-2}$.*

4. *The search directions are A-orthogonal: for any $j < k$, p_k is A-orthogonal to p_j.*

As we will see, the magic fact that makes conjugate gradient efficient is that r_k is A-orthogonal to $p_0, \ldots, p_{k-2}$. This means that when doing Gram-Schmidt orthogonalization, we only need to subtract out one previous term p_{k-1}, rather than k terms $p_0, \ldots, p_{k-1}$. If we had to do that, then conjugate gradient would not be efficient—it would take $O(kd)$ flops at the kth iteration!

Proof.

(1) $\Longrightarrow$ (2): Note that x_k being the minimizer of $f(x)$ on the hyperplane $x_0 + \mathcal{K}_k(A; r_0)$ means that the gradient $\nabla f(x)$ must be perpendicular to the subspace $\mathcal{K}_k(A; r_0)$. But the gradient is just $-r_k$, so r_k is orthogonal to $\mathcal{K}_k(A; r_0)$, and to $p_0, \ldots, p_{k-1}$ and $r_0, \ldots, r_{k-1}$.

(2) $\Longrightarrow$ (3): If $z \in \mathcal{K}_{k-1}(A; r_0)$, then $Az \in \mathcal{K}_k(A; r_0)$, so by (2), r_k is orthogonal to Az, or equivalently, r_k is A-orthogonal to z.

(2,3) $\Longrightarrow$ (4): We note that p_k is obtained by Gram-Schmidt orthogonalization. To see this, note that if p_k is defined using Gram-Schmidt on $\langle \cdot, \cdot \rangle_A$, then

$$\begin{aligned} p_k &= r_k - \sum_{i=0}^{k-1} \frac{\langle r_k, p_i \rangle_A}{\langle p_i, p_i \rangle_A} p_i \\ &= r_k - \frac{\langle r_k, p_{k-1} \rangle_A}{\langle p_{k-1}, p_{k-1} \rangle_A} p_{k-1} \\ &= -\frac{r_k^\top p_{k-1}}{p_{k-1}^\top A p_{k-1}} \end{aligned}$$

since r_k is A-orthogonal to $p_{k-2}, \ldots, p_0$. We rewrite this in the form in the algorithm. Note

$$\begin{aligned} r_k &= b - Ax_k \\ &= b - A(x_{k-1} + \alpha_{k-1} A p_{k-1}) \\ &= r_{k-1} - \alpha_{k-1} A p_{k-1} \\ \Longrightarrow \quad Ap_{k-1} &= \frac{1}{\alpha_{k-1}} (r_k - r_{k-1}) \end{aligned} \tag{3}$$

and

$$p_{k-1} = p_{k-1} + \beta_{k-2} p_{k-2} \tag{4}$$

for some β_{k-2}. Substituting this in, we have

$$\begin{aligned}
-\frac{\langle r_k, p_{k-1}\rangle_A}{\langle p_{k-1}, p_{k-1}\rangle_A} &= -\frac{r_k^\top A p_{k-1}}{p_{k-1}^\top A p_{k-1}} \\
&= -\frac{\frac{1}{\alpha_{k-1}} r_k^\top (r_k - r_{k-1})}{(r_{k-1} + \beta_{k-2} p_{k-2})^\top A p_{k-1}} && \text{by (3) and (4)} \\
&= -\frac{\frac{1}{\alpha_{k-1}} r_k^\top (r_k - r_{k-1})}{\frac{1}{\alpha_{k-1}} r_{k-1}^\top (r_k - r_{k-1})} && p_{k-1} \perp_A p_{k-2} \\
&= \frac{r_k^\top r_k}{r_{k-1}^\top r_{k-1}} && r_k \perp r_{k-1}
\end{aligned}$$

using the fact that $r_k \perp r_{k-1}$ and $p_{k-1} \perp_A p_{k-2}$. This is exactly the update in the algorithm.

$\Rightarrow(1)(k+1)$ We now show the induction step.

As explained by the decomposition 2, when the search directions p_j's are A-orthogonal, choosing $x_{k+1} = x_k + \alpha_k p_k$ to minimize $f(x)$, actually gives the minimum over $x_0 + \text{span}\{p_0, \ldots, p_k\} = x_0 + \mathcal{K}_{k+1}(A; r_0)$.

The choice of α_k is given by (1):

$$\alpha_k = \frac{p_k^\top r_k}{p_k^\top A p_k} = \frac{(r_k + \beta_k p_{k-1})^\top r_k}{p_k^\top A p_k} = \frac{r_k^\top r_k}{p_k^\top A p_k}$$

because p_{k-1} is perpendicular to r_k.

$\square$

Conjugate gradient improves the dependence on $\kappa(A)$ by a square-root factor.

Theorem 4.4 (Convergence of conjugate gradient)**.** *Let $\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ denote the condition number of A. Then in Algorithm 3,*

$$\|x_n - x\|_A \le 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^n \|x_0 - x\|_A .$$

The proof of this involves *Chebyshev polynomials*; we will carry out the proof in the unit on polynomial interpolation.

We relate the convergence of conjugate gradient to a problem about polynomial interpolation.

Lemma 4.5. *Let $\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ denote the condition number of A. Then in Algorithm 3,*

$$\|x_n - x\|_A \le \min_{\deg q \le n, q(0)=1} \max_{\lambda \text{ eigenvalue of } A} |q(\lambda)| \, \|x_0 - x\|_A .$$

Proof. Because $x_n \in x_0 + \mathcal{K}_n(A; r_0)$, we can write

$$x_n = x_0 + p(A)r_0 = x_0 + p(A)(b - Ax_0) = x_0 + p(A)A(x - x_0)$$
$$x_n - x = (I - p(A)A)(x_0 - x).$$

for some polynomial p of degree $\leq n - 1$.

Intuition: In order for x_n to be close to x, we would like $p(A)A \approx I$. Thinking of p as a function on $\mathbb{R}$ rather than on matrices, this is like saying that $p(x) \approx \frac{1}{x}$, or that $p(x)$ is a good interpolation of $\frac{1}{x}$ on some interval. It turns out that we can bound $p(A)A$ by its evaluation on eigenvalues of A, so that we want $p(x) \approx \frac{1}{x}$ for $x \in [\lambda_{\min}, \lambda_{\max}]$. It will be easier for us to work with the polynomial $1 - xp(x)$, which we do below.

Moreover, by construction, $p(x)$ is the polynomial of degree $\leq k - 1$ that minimizes $\frac{1}{2}x_n^\top Ax_n - b^\top x_n$ or equivalently minimizes $(x_n - x)^\top A(x_n - x)$ when $x_n = x_0 + p(A)r_0$. This is the A-norm of the error e_n. Letting $q(x) = 1 - p(x)x$, we note that $\deg q \leq k$, and we have the restriction $q(0) = 1$. Hence

$$\|e_n\|_A = \min_{\deg p \leq n, p(0)=1} \|q(A)e_0\|_A \leq \min_{\deg p \leq n, p(0)=1} \|q(A)\|_A \|e_0\|_A.$$

If the condition number of A is $\kappa(A)$, then all eigenvalues are in $[\lambda_{\min}, \lambda_{\max}]$ where $\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$. Now $\|q(A)\|_A = \max_x \frac{x^\top q(A)Aq(A)x}{x^\top Ax} = \max_y \frac{y^\top q(A)q(A)y}{y^\top y} = \|q(A)\|_2$ by setting $y = A^{\frac{1}{2}}x$.

Now, because A is symmetric, we can diagonalize it as $UDU^\top$ where U is orthogonal and D is diagonal. Then $q(A) = Up(D)U^\top$, and

$$\|q(A)\|_A = \|q(A)\| = \|q(D)\| = \max_{\lambda \text{ eigenvalue of } A} |q(\lambda)|.$$

□

Proof of Theorem 4.4. By Lemma 4.5, we have reduced to solving the following problem: *Find the polynomial p such that $p(0) = 1$ and $\deg p \leq n$, such that $\max_{x\in[\lambda_{\min},\lambda_{\max}]} |p(x)|$ is minimized.* Then this will be the factor that we get.

Let $\kappa = \kappa(A)$. We now construct a $p(x)$ such that $\max_{x\in[\lambda_{\min},\lambda_{\max}]} |p(x)| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n$, using Chebyshev polynoimals.

To obtain a polynomial that is as small as possible on an interval, and with the given value $p(0) = 1$, we take p to be a Chebyshev polynomial suitably scaled. Let

$$p(x) = \frac{1}{T_n\left(-\frac{\lambda_{\min}+\lambda_{\max}}{\lambda_{\max}-\lambda_{\min}}\right)} T_n\left(\frac{x - \frac{\lambda_{\min}+\lambda_{\max}}{2}}{\frac{\lambda_{\max}-\lambda_{\min}}{2}}\right).$$

The scaling factor in front was chosen so that $p(0) = 1$, the linear function in the argument takes the interval $[\lambda_{\min}, \lambda_{\max}]$ to $[-1, 1]$. Note that the maximum of $T_n\left(\frac{x-\frac{\lambda_{\min}+\lambda_{\max}}{2}}{\frac{\lambda_{\max}-\lambda_{\min}}{2}}\right)$ on $[\lambda_{\min}, \lambda_{\max}]$ is the maximum of $T_n(x)$ on $[-1, 1]$, which is 1. Hence

$$\max_{x\in[\lambda_{\min},\lambda_{\max}]} |p(x)| = \left|T_n\left(-\frac{\lambda_{\min} + \lambda_{\max}}{\lambda_{\max} - \lambda_{\min}}\right)\right| = \left|T_n\left(-\frac{\kappa + 1}{\kappa - 1}\right)\right|.$$

It is left as an exercise to show that $\left|T_n\left(-\frac{\kappa+1}{\kappa-1}\right)\right| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n$. (Hint: Use the fact that $T_n\left(z + \frac{1}{z}\right) = z^n + \frac{1}{z^n}$.) □

5 Remarks

- We can apply *preconditioning* to gradient descent or conjugate gradient by considering the system

$$(P^{-1/2}AP^{-1/2})(P^{1/2}x) = P^{-1/2}b$$

  where P is chosen to reduce the condition number: $\kappa(P^{-1/2}AP^{-1/2}) < \kappa(A)$. Preconditioning can be done implicitly in the algorithm.

- For nonsymmetric, nonsingular A, we can write $Ax = b$ as $A^\top Ax = A^\top b$, where $A^\top A$ is now symmetric positive definite, and then apply conjugate gradient to $A^\top A$.

  However, $\kappa(A^\top A) = \kappa(A)^2$, so convergence becomes slow. It is better to use more sophisticated methods that work with A directly (see Section 6.5 of Ascher and Greif).

- Conjugate gradient is a "direct method in theory, but an iterative method in practice." If exact arithmetic is used, then for $A \in \mathbb{R}^{n\times n}$, $x_n = x$: the exact solution is obtained in n steps. This is because each r_k is A-orthogonal to $r_0, \ldots, r_{k-1}$, so if $r_k \neq 0$, then $r_0, \ldots, r_k$ are linearly independent, and we must have $k < n$. In other words, if $r_k \neq 0$, then conjugate gradient explores a new linearly independent direction, and there are only n dimensions. (Another way to see this is from Lemma 4.5: the degree-n polynomial q can be chosen to have all the eigenvalues as zeros.)

  However, in practice, it is useless as a direct method because Conjugate Gradient is unstable: round-off error blows up. This instability is due to instability in the Gram-Schmidt orthogonalization process when the input vectors are "close" to linearly dependent.

  It is fine however, to run Conjugate Gradient for a number of iterations $k \ll n$. If you run many iterations, you may want to "restart" the algorithm periodically to prevent the instability.

  Historically, CG was proposed as a direct method, and people lost interest because of its instability, but then it made a comeback as an iterative method.

- One application where symmetric positive definite matrices come up naturally is graph Laplacians. For a network of resistors, putting the inverse resistances between nodes in the L term, given the outgoing currents in b, the voltages x are given by Ohm's Law $Lx = b$.