**Mathematical Toolkit** **Autumn 2023**

Lecture 10: October 26, 2023

Lecturer: Madhur Tulsiani

## 1 Solving systems of linear equations: Gaussian elimination

Given a system of linear equations $Ax = b$ for $A \in \mathbb{F}^{m \times n}, b \in \mathbb{F}^m$, recall that we can solve the system or determine that there is no solution by converting the matrix $[A \mid b]$ to a row-reduced form using elementary row operations.

**Definition 1.1** *A matrix $M \in \mathbb{F}^{m \times n}$ is said to be in* row-reduced *form if*

- *The first non-zero entry in each row (known as the* leading entry*) is 1.*
- *If the leading entry in row $i_0$ is in column $j_0$, then $M_{ij} = 0$ for all $i > i_0$ and $j \leq j_0$.*
- *All non-zero rows occur above the zero rows.*

Notice that a matrix in the row-reduced form is always upper triangular. The system has no solution if and only if there is a non-zero row with a leading entry in the last column (corresponding to the entries of $b$). Also, if the system has a solution, then it can easily be found using back-substitution, starting from the last non-zero row.

Also, recall that an elementary row operations consist of the following (using $M_i$ to denote the $i^{th}$ row of $M$):

- Swapping the rows $M_i$ and $M_j$, for some $i, j, \in [m]$.
- $M_i \leftarrow c \cdot M_i$ for some $i \in [m], c \in \mathbb{F} \setminus \{0\}$.
- $M_i \leftarrow M_i + c \cdot M_j$ for some $i, j \in [m], c \in \mathbb{F}$.

A matrix $M$ can always be converted to a row-reduced form using elementary row operations, which gives a general algorithm for solving a system of linear equations over any field. However, the time taken by this algorithm can be as large as $\Omega(n^3)$, which is prohibitive for large matrices. In the next lecture, we will discuss methods which can take advantage of sparsity to significantly speed up the solution of linear systems.

**Exercise 1.2** *Prove that performing elementary row operations on a given matrix $M$ changes neither the row rank, nor the column rank of $M$. Use this to prove that for any matrix $M$, the row-rank and column-rank are equal.*

# 2 Solving sparse systems of linear equations

Given $A \in \mathbb{R}^{m \times n}$, if we have a representation of the non-zero entries of $A$ in "list form" i.e., a list of the non-zero entries in each row, then the for any vector $v$, if the matrix has a total of $N$ non-zero entries, then for any vector $v$, the product $Av$ can be computed using $O(N)$ arithmetic operations. We will keep this as our base cost and try to compute a solution to $Ax = b$ using as few matrix-vector multiplications as possible.

For the purposes of the discussion below, we will assume that $A \in \mathbb{R}^{n \times n}$ is a symmetric, positive-definite matrix (written as $A \succ 0$). This assumption is not as restrictive as it sounds, and in particular is no more restrictive than assuming that $A$ is invertible. Given a system $A_0 x = b_0$, we can always multiply both sides by $A_0^T$ and obtain $A_0^T A_0 x = A_0^T b_0$, where the matrix $A_0^T A_0$ is now positive-definite (if $A_0$ is invertible). Note that $A_0^T A_0$ may not be sparse, but we can still compute $A_0^T A_0 v$ in $O(N)$ operations for any vector $v$ using only $O(N)$ operations (we will also need the list of non-zero entries in every column for this). Taking $A = A_0^T A_0$ and $b = A_0^T b_0$ satisfies the required assumptions.

**Remark 2.1** *The methods we discuss here will require analyzing distances and inner products, and thus we will work with matrices with real entries (though everything we say will extend easily to complex matrices).*

## 2.1 Steepest descent

Given a system $Ax = b$ with $A \succ 0$, we apply a method for minimizing the function

$$f(x) \;=\; \frac{1}{2} \cdot \langle Ax, x \rangle - \langle b, x \rangle + c$$

for some arbitrary constant $c \in \mathbb{R}$. This can be motivated by recalling that we originally had the system $A_0 x = b_0$ and $Ax = b$ was obtained by multiplying both sides by $A_0^T$. If we consider minimizing the least square distance, we get

$$\|A_0 x - b_0\|^2 \;=\; \langle A_0 x, A_0 x \rangle - 2 \langle b_0, A_0 x \rangle + \|b_0\|^2 \;=\; \langle Ax, x \rangle - 2 \langle b, x \rangle + \|b_0\|^2 .$$

Of course, scaling by a factor of 2 and changing the constant term does not change the minimizer. If $x^*$ is the solution to the linear system, we can also re-write the above as

$$\|A_0 (x - x^*)\|^2 \;=\; \langle A(x - x^*), (x - x^*) \rangle \;=\; \langle x - x^*, x - x^* \rangle_A ,$$

where $\langle x, y \rangle_A$ denotes the function $\langle Ax, y \rangle$.

**Exercise 2.2** *Let $A \in \mathbb{R}^{n \times n}$ be a positive definite matrix. Let the function $\mu : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be defined as $\mu(x, y) = \langle Ax, y \rangle$. Check that $\mu$ is an inner product. We will use $\langle \cdot, \cdot \rangle_A$ to this inner product.*

The inner product $\langle \cdot, \cdot \rangle_A$ and the associated norm are sometimes more convenient for measuring the distance to the solution $x^*$ since this distance actually measures the least square error in the "output" $A_0 x$ rather than the "input" $x$. We will need this inner product when working with the conjugate gradient method.

We will use the following algorithm for solving the linear system:

- Start with an arbitrary vector $x_0$.
- At time $t$, update

$$x_{t+1} \;=\; x_t - \eta \cdot \nabla f(x_t) \;=\; \eta \cdot (Ax_t - b) \;.$$

The method can also be analyzed by choosing an optimal step size $\eta_t$ at each time $t$ but we will work with the simpler variant here. Let $x^*$ be the solution to the system $Ax = b$. We note that

$$x_{t+1} - x^* \;=\; x_t - x^* - \eta \cdot A(x_t - x^*) \;=\; (I - \eta A)(x_t - x^*)\,.$$

By induction,

$$x_t - x^* \;=\; (I - \eta A)^t (x_0 - x^*) \quad \Rightarrow \quad \|x_t - x^*\|_2 \;\leq\; \|I - \eta A\|_2^t \, \|x_0 - x^*\|_2 \;,$$

where we used the fact that if $\lambda$ is an eigenvalue of $M$, then $\lambda^t$ is an eigenvalue of $M^t$, which gives that $\left\|(I - \eta A)^t\right\|_2 = \|I - \eta A\|_2^t$. Thus, if $\|I - \eta A\|_2$ is small, we can reach a point close to the solution $x^*$ in a small number of steps. We now choose $\eta$ to minimize $\|I - \eta A\|_2$. Let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ denote the eigenvalues of $A$. Then, the eigenvalues of $I - \eta A$ are $1 - \eta\lambda_1 \geq \cdots \geq 1 - \eta\lambda_n$. Thus, we have

$$\|I - \eta A\|_2 \;=\; \max\left\{|1 - \eta\lambda_1|\,, |1 - \eta\lambda_n|\right\} \,.$$

Check that this is minimized for $\lambda = \frac{2}{\lambda_1 + \lambda_n}$. Plugging this, we get that

$$\|I - \eta A\|_2 \;=\; 1 - \frac{2}{\frac{\lambda_n}{\lambda_1} + 1} \;=\; 1 - \frac{2}{\kappa + 1}\,.$$

Here $\kappa = \lambda_n / \lambda_1$ is known as the condition number of the matrix $A$. Using this, we get that $\|x_t - x^*\| \leq \varepsilon \, \|x_0 - x^*\|$ after $O(\kappa \log(1/\varepsilon))$ iterations. Notice that the cost of each iteration is just $O(1)$ matrix-vector multiplications.

**Exercise 2.3** *Obtain a similar bound for the distance $\|x_t - x^*\|_A$ defined as $\sqrt{\langle (x - x^*), (x - x^*) \rangle_A}$.*

In the next lecture, we will discuss the conjugate gradient method, which can obtain a similar guarantee in $O(\sqrt{\kappa}\log(1/\varepsilon))$ iterations.

# 3 The conjugate gradient method

In the previous section, we saw the steepest descent or gradient descent method for finding a solution to the linear system $Ax = b$ for $A \succ 0$. The method guarantees $\|x_t - x^*\| \leq \varepsilon \cdot \|x_0 - x^*\|$ after $t = O(\kappa \cdot \log(1/\varepsilon))$ iterations, where $\kappa$ is the condition number of the matrix $A$. We will see that the conjugate gradient can obtain a similar guarantee in $O(\sqrt{\kappa} \cdot \log(1/\varepsilon))$ iterations.

For the steepest descent method, if we start from $x_0 = 0$, we get

$$x_t - x^* \;=\; (I - \eta A)(-x^*)\,,$$

which gives $x_t = p(A) \cdot b$ for some polynomial $p$ of degree at most $t$. The conjugate gradient method just takes this idea of finding an $x$ of the form $p(A) \cdot b$ and runs with it. The method finds an $x_t = p_t(A) \cdot b$ where $p_t$ is the *best* polynomial of degree at most $t$ i.e., the polynomial which minimizes the function $\frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$. However, the method does not explicitly work with polynomials. Instead we use the simple observation that any vector of the form $p_t(A) \cdot b$ lies in the subspace $\text{Span}\left(\{b, Ab, \ldots, A^t b\}\right)$ and the method finds the best vector in the subspace at every time $t$.

**Definition 3.1** *Let $\varphi : V \to V$ be a linear operator on a vector space $V$ and let $v \in V$ be a vector. The* Krylov subspace *of order $t$ defined by $\varphi$ and $v$ is defined as*

$$\mathcal{K}_t(\varphi, v) \;:=\; \text{Span}\left(\left\{v, \varphi(v), \ldots, \varphi^{t-1}(v)\right\}\right)\,.$$

Thus, at step $t$ of the conjugate gradient method, we find the best vector in the space $\mathcal{K}_t(A, b)$ (we will just write the subspace as $\mathcal{K}_t$ since $A$ and $b$ are fixed for the entire argument). The trick of course is to be able to do this in an iterative fashion so that we can quickly update the minimizer in the space $\mathcal{K}_{t-1}$ to the minimizer in the space $\mathcal{K}_t$. This can be done by expressing the minimizer in $\mathcal{K}_{t-1}$ in terms of a convenient orthonormal basis $\{u_0, \ldots, u_{t-1}\}$ for $\mathcal{K}_{t-1}$. It turns out that if we work with a basis which is orthonormal with respect to the inner product $\langle \cdot, \cdot \rangle_A$, at step $t$ we only need to update the component of the minimizer along the new vector $u_t$ we get to obtain a basis for $\mathcal{K}_t$.

## 3.1 The algorithm

Recall that we defined the inner product $\langle x, y\rangle_A := \langle Ax, y\rangle$ where $\langle \cdot, \cdot \rangle$ denotes the standard inner product on $\mathbb{R}^n$. As before we consider the function $\frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$ and pick

$$x_t \;:=\; \operatorname*{arg\,min}_{x \in \mathcal{K}_t} \; f(x)\,.$$

This can also be thought of as finding the closest point to $x^*$ in the space $\mathcal{K}_t$ (under the distance $\|\cdot\|_A$) since

$$f(x) \;=\; \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle \;=\; \frac{1}{2}\langle Ax, x\rangle - \langle Ax^*, x\rangle \;=\; \frac{1}{2}\langle x, x\rangle_A - \langle x^*, x\rangle_A$$
$$= \frac{1}{2}\cdot\left(\|x - x^*\|_A^2 - \|x^*\|_A^2\right),$$

which gives

$$x_t \;=\; \operatorname*{arg\,min}_{x\in\mathcal{K}_t} f(x) \;=\; \operatorname*{arg\,min}_{x\in\mathcal{K}_t} \|x - x^*\|_A \,.$$

We have already seen how to compute find the characterize the closest point in a subspace, to a given point. Let $\{u_0, \ldots, u_{t-1}\}$ be an orthonormal basis for $\mathcal{K}_t$ under the inner product $\langle\cdot,\cdot\rangle_A$. Completing this to an orthonormal basis $\{u_0, \ldots, u_{n-1}\}$ for $\mathbb{R}^n$, let $x^*$ be expressible as

$$x^* \;=\; \sum_{i=0}^{n-1} c_i \cdot u_i \;=\; \sum_{i=0}^{n-1} \langle x^*, u_i\rangle_A \cdot u_i \,.$$

Then we know that the closest point $x_t$ in $\mathcal{K}_t$, under the distance $\|\cdot\|_A$ is given by

$$x_t \;=\; \sum_{i=0}^{t-1} \langle x^*, u_i\rangle_A \cdot u_i \;=\; \sum_{i=0}^{t-1} \langle Ax^*, u_i\rangle \cdot u_i \;=\; \sum_{i=0}^{t-1} \langle b, u_i\rangle \cdot u_i$$

Note that even thhough we do not know $x^*$, we can find $x_t$ given an orthonormal basis $\{u_0, \ldots, u_{t-1}\}$, since we can compute $\langle b, u_i\rangle$ for all $u_i$. This gives the following algorithm:

- Start with $u_0 = b / \|b\|_A$ as an orthonormal basis for $\mathcal{K}_1$.
- Let $x_t \;=\; \sum_{i=0}^{t-1} \langle b, u_i\rangle \cdot u_i$ for a basis $\{u_0, \ldots, u_{t-1}\}$ orthonormal under the inner product $\langle\cdot,\cdot\rangle_A$.
- Extend $\{u_0, \ldots, u_{t-1}\}$ to a basis of $\mathcal{K}_{t+1}$ by defining

$$v_t \;=\; A^t b - \sum_{i=0}^{t-1} \left\langle A^t b, u_i\right\rangle_A \cdot u_i \quad \text{and} \quad u_t \;=\; \frac{v_t}{\sqrt{\langle v_t, v_t\rangle_A}}\,.$$

- Update $x_{t+1} \;=\; x_t + \langle b, u_t\rangle \cdot u_t$.

Notice that the basis extension step here seems to require $O(t)$ matrix-vector multiplications in the $t^{th}$ iteration and thus we will need $O(t^2)$ matrix-vector multiplications in total for $t$ iterations. This would negate the quadratic advantage we are trying to gain over steepest descent. However, in the homework you will see a way of extending the basis using only $O(1)$ matrix-vector multiplications in each step.

### 3.2 Bounding the number of iterations

Since $x_t$ lies in the subspace $\mathcal{K}_t$, we have $x_t = p(A) \cdot b$ for some polynomial $p$ of degree at most $t-1$. Thus,

$$x_t - x^* \;=\; p(A) \cdot b - x^* \;=\; p(A) \cdot A \cdot x^* - x^* \;=\; (I - p(A) \cdot A) \cdot (x_0 - x^*)\,,$$

since $x_0 = 0$. We can think of $I - p(A)A$ as a polynomial $q(A)$, where $\deg(q) \le t$ and $q(0) = 1$. Recall from last lecture that the minimizer of $f(x)$ is the same as the minimizer of $\langle x - x^*, x - x^* \rangle_A = \|x - x^*\|_A^2$. Since $p(A)b$ is the minimizer of $f(x)$ in $\mathcal{K}_t$, we have

$$\|x_t - x^*\|_A^2 \;=\; \min_{q \in Q_t} \|q(A)(x_o - x^*)\|_A^2\,,$$

where $Q_t$ is the set of polynomials defined as

$$Q_t \;:=\; \{q \in \mathbb{R}[z] \mid \deg(q) \le t, q(0) = 1\}\,.$$

Use the fact that if $\lambda$ is an eigenvalue of a matrix $M$, then $\lambda^t$ is an eigenvalue of $M^t$ (with the same eigenvector) to prove that the following.

**Exercise 3.2** *Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A$. Then for any polynomial $q$ and any $v \in \mathbb{R}^n$,*

$$\|q(A)v\|_A \;\le\; \left(\max_i |q(\lambda_i)|\right) \cdot \|v\|_A\,.$$

Using the above, we get that

$$\|x_t - x^*\|_A \;\le\; \left(\min_{q \in Q_t} \max_i |q(\lambda_i)|\right) \cdot \|x_0 - x^*\|_A\,.$$

Thus, the problem of bounding the norm of $x_t - x^*$ is reduced to finding a polynomial $q$ of degree at most $t$ such that $q(0) = 1$ and $q(\lambda_i)$ is small for all $i$.

**Exercise 3.3** *Verify that using $q(z) = \left(1 - \frac{2z}{\lambda_1 + \lambda_n}\right)^t$ recovers the guarantee of the steepest descent method.*

Note that the conjugate gradient method itself does not need to know anything about the optimal polynomials in the above bound. The polynomials are only used in the analysis of the bound. The following claim, which can be proved by using slightly modified Chebyshev polynomials, suffices to obtain the desired bound on the number of iterations.

**Claim 3.4** *For each $t \in \mathbb{N}$, there exists a polynomial $q_t \in Q_t$ such that*

$$|q_t(z)| \;\le\; 2 \cdot \left(1 - \frac{2}{\sqrt{\kappa} + 1}\right)^t \quad \forall z \in [\lambda_1, \lambda_n]\,.$$

We will prove the claim later using Chebyshev polynomials. However, using the claim we have that

$$\|x_t - x^*\|_A \;\leq\; \left(\min_{q \in Q_t} \max_i |q(\lambda_i)|\right) \cdot \|x_0 - x^*\|_A \;\leq\; 2 \cdot \left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^t \cdot \|x_0 - x^*\|_A \;.$$

Thus, $O(\sqrt{\kappa}\log(1/\varepsilon))$ iterations suffice to ensure that $\|x_t - x^*\|_A \leq \varepsilon \cdot \|x_0 - x^*\|_A$.

### 3.3 Chebyshev polynomials

The Chebyshev polynomial of degree $t$ is given by the expression

$$P_t(z) \;=\; \frac{1}{2} \cdot \left[\left(z + \sqrt{z^2-1}\right)^t + \left(z - \sqrt{z^2-1}\right)^t\right] .$$

Note that this is a polynomial since the odd powers of $\sqrt{z^2-1}$ will cancel from the two expansions. For $z \in [-1, 1]$ this can also be written as

$$P_t(z) \;=\; \cos\left(t \cos^{-1}(z)\right) ,$$

which shows that $P_t(z) \in [-1, 1]$ for all $z \in [-1, 1]$.

Using these polynomials, we can define the required polynomials $q_t$ as

$$q_t(z) \;=\; \frac{P_t\left(\frac{\lambda_1+\lambda_n-2z}{\lambda_n-\lambda_1}\right)}{P_t\left(\frac{\lambda_1+\lambda_n}{\lambda_n-\lambda_1}\right)} .$$

The denominator is a constant which does not depend on $z$ and the numerator is a polynomial of degree $t$ in $z$. Hence $\deg(q_t) = t$. Also, the denominator ensures that $q_t(0) = 1$. Finally, for $z \in [\lambda_1, \lambda_n]$, we have $\left|\frac{\lambda_1+\lambda_n-2z}{\lambda_n-\lambda_1}\right| \leq 1$. Hence, the numerator is in the range $[-1, 1]$ for all $z \in [\lambda_1, \lambda_n]$. This gives

$$|q_t(z)| \;\leq\; \frac{1}{P_t\left(\frac{\lambda_1+\lambda_n}{\lambda_n-\lambda_1}\right)} \;\leq\; 2 \cdot \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t \;=\; 2 \cdot \left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^t .$$

The last bound above can be computed directly from the first definition of the Chebyshev polynomials.

An detailed treatment of the conjugate gradient method, and a related method called the Lanczos Method, which also uses the Krylov subspace, can be found in the excellent monograph by Vishnoi [Vis13].

## References

[Vis13] Nisheeth K. Vishnoi, *Lx = b*, Foundations and Trends® in Theoretical Computer Science **8** (2013), no. 1–2, 1–141. 7