Math 128B: Lecture notes

Spring 2025

M. Lindsey

Latest update: January 20

Contents

Part I

Iterative methods for linear systems

1 Introduction

- Review syllabus.
- Motivate iterative methods for $Ax = b$:
  - $O(n^3)$ to $O(n)$ scaling, hopefully.
  - Idea of a matrix-free method.
  - Useful structure: sparsity and low-rankness.
  - Need to discuss convergence, motivating vector norms....
- **Note:** Blue text indicates material outside the course textbook, *Numerical Analysis* by Burden and Faires, 10th ed.

2 Vector norms (§7.1)

- Definition: **vector norm** on $\mathbb{R}^n$, function $\|\cdot\|$ from $\mathbb{R}^n$ to $\mathbb{R}$ with the following properties:
  - Nonnegativity, definiteness (zero vector), homogeneity, triangle inequality.
- Vector notation $x = (x_1, \ldots, x_n)^\top$, write out column.
- Recall **dot product**, which is an inner product, and express as $\mathbf{x}^\top \mathbf{y}$.
- We need only two norms on $\mathbb{R}^n$, though there are many others.
  - Define l_2 norm or (2-norm / Euclidean norm), l_∞ norm (or infinity norm / max norm), l_1 norm (or 1-norm).
  - Define general p-norm.
  - Draw norm balls in $\mathbb{R}^2$.
- Show triangle inequality for infinity norm and 1-norm. (The rest of the properties are simple to verify.)
- Triangle inequality is harder for the 2-norm. (By the way, 2-norm is special, it comes from an inner product! By default assume we are using 2-norm.)
  - Theorem: **Cauchy-Schwarz** inequality.
    - Recall holds for general inner products! Review proof in case of dot product. Assume $x, y \neq 0$.

$$\begin{aligned} 0 &\leq \|x - \lambda y\|^2 \\ &= (x - \lambda y)^\top (x - \lambda y) \\ &= x^\top x - 2\lambda x^\top y + \lambda^2 y^\top y \\ &= \|x\|^2 - 2\lambda (x \cdot y) + \lambda^2 \|y\|^2 \end{aligned}$$

      Let $\lambda = \|x\|/\|y\|$. Then $\|x\|^2 - (\|x\|/\|y\|)\, x \cdot y \geq 0$.

  - Cauchy-Schwarz implies triangle inequality for 2-norm.

- Distance induced by norm. Convergence of sequence $\{x^{(k)}\}_{k=1}^{\infty}$.
- First reassuring theorem: sequence converges in infinity norm if and only if it converges componentwise.
- But what about other norms?
  - Theorem: All finite-dimensional norms are *equivalent*.
  - Corollary: Convergence in any norm is equivalent to convergence in any other norm.
  - Thus we can write $\lim_{k\to\infty} x^{(k)}$ *unambiguously* because the limit point (if it exists) is the same regardless of which norm we use to measure convergence.
  - More concretely...?
- Concrete theorem: $\|x\|_\infty \le \|x\|_2 \le \sqrt{n}\|x\|_\infty$. (Draw picture.)
  - $\|x\|_2^2 = \sum_i x_i^2 \le \sum_i \max_j(x_j^2) = n\left(\max_j |x_j|\right)^2 \le n\|x\|_\infty^2$.
  - Meanwhile for other inequality, let j be a maximizing index.

3 Matrix norms (§7.1)

- Definition: **matrix norm on** $\mathbb{R}^{n\times n}$ is a function $\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$.
  - Nonnegativity, definiteness (zero matrix), homogeneity, triangle inequality, **submultiplicativity**.
- Definition: **natural** or **induced norm**,

$$\|A\| = \max_{\|x\|=1} \|Ax\| \overset{?}{=} \max_{z\neq 0} \frac{\|Az\|}{\|z\|}.$$

- Theorem: this is a matrix norm. (Proof: **homework**.)
- Theorem: $\|Az\| \le \|A\|\|z\|$.
  - Can apply more generally to a string via submultiplicativity, e.g., $\|ABCz\| \le \|A\|\,\|B\|\,\|C\|\,\|z\|$.
- Notation $\|A\|_p$ induced by vector p-norm.
- Theorem: $A = (a_{ij})$, then $\|A\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$.
  - Define $C; = \max_i \sum_{j=1}^n |a_{ij}|$.
  - First WTS: $\|Ax\|_\infty \le C\|x\|_\infty$. This will imply $\|A\|_\infty \le C$.
    * Compute

$$\begin{aligned}\max_i \left|\sum_j a_{ij}x_j\right| &\le \max_i \sum_j |a_{ij}||x_j| \\ &\le \max_i \sum_j |a_{ij}|\,\|x_\infty\|.\end{aligned}$$

  - Next want to find x of unit norm such that $\|Ax\|_\infty \ge C$. This will imply $\|A\|_\infty \ge C$.

    * Let i be the maaximizing Set $x_j = \text{sign}(a_{ij})$.
    * Then

$$\|Ax\|_\infty \geq |[Ax]_i| = \left|\sum_j a_{ij}x_j\right| = \left|\sum_j a_{ij}\text{sign}(a_{ij})\right| = \sum_j |a_{ij}| = C.$$

- To interpret the matrix 2-norm we need to review eigendecomposition....
- But first, what if I wanted to simply view $\mathbb{R}^{n\times n}$ as $\mathbb{R}^{n^2}$ (i.e., view $n \times n$ matrices as n^2-dimensional vectors with no internal structure) and apply p-norms?
  - More specifically we can define a 'vectorization' map $\text{vec} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ which stacks all the columns of a matrix into a very long column vector. Then consider defining norms such as $\|A\| = \|\text{vec}(A)\|_p$, where $\|\cdot\|_p$ indicates a vector p-norm.
  - All the properties of a matrix norm except submultiplicativity are inherited from the vector norm. Thus you can think of these as defining **vector norms** on the **vector space** of $n \times n$ matrices. *But they do not define matrix norms in general.*
  - However, submultiplicativity **is** inherited in **one** special case: the Frobenius norm $\|A\|_F = \|\text{vec}(A)\|_2$. Therefore this norm actually does happen to be a matrix norm! But it is **not** a natural/induced norm. Meanwhile, the other cases $p \neq 2$ are **not very meaningful**.
  - Note with caution that $\|A\|_F \neq \|A\|_2$!
- More natural definition: **Frobenius norm** $\|A\|_F = \sqrt{\sum_{ij}|a_{ij}|^2} \overset{?}{=} \text{Tr}[A^\top A]$.
  - It is induced by the Frobenius inner product on the vector space of $n\times n$ matrices, defined by

$$\langle A, B\rangle = \sum_{ij} A_{ij}B_{ij} \overset{?}{=} \text{Tr}[A^\top B],$$

    in the sense that $\|A\|_F = \sqrt{\langle A, A\rangle}$.
  - Meanwhile, the natural norms are **not** induced by inner products on the space of matrices!
  - Submultiplicativity: suffices to show $\|AB\|_F^2 \leq \|A\|_F^2\|B\|_F^2$. First note that $[AB]_{ij} = \sum_k a_{ik}b_{kj}$, the (i,j) entry of AB, can be viewed as the dot product $[AB]_{ij} = a^{(i)} \cdot b^{(j)}$, where $a^{(i)}$ is the i-th row of A and $b^{(j)}$ is the j-th column of B. Then

$$\begin{aligned}
\|AB\|_F^2 &= \sum_{ij}([AB]_{ij})^2 \\
&= \sum_{ij}(a^{(i)} \cdot b^{(j)})^2 \\
&\overset{\text{(C-S)}}{\leq} \sum_{ij} \|a^{(i)}\|_2^2\|b^{(j)}\|_2^2 \\
&= \left(\sum_i \|a^{(i)}\|_2^2\right)\left(\sum_j \|b^{(j)}\|_2^2\right) \\
&\overset{?}{=} \|A\|_F^2\|B\|_F^2.
\end{aligned}$$

4 Spectral radius (§7.2)

- Recall for a square matrix A, an eigenvalue is a real or complex number λ such that $Ax = \lambda x$ for some $x \neq 0$. In this case x is an eigenvector. Eigenspace. The eigenvalues of A are the roots of the characteristic polynomial $p(\lambda) = \det(A - \lambda I)$. (Why?)
  - If you know an eigenvalue, you can find corresponding eigenspace as a null space.
- Definition: **spectral radius** $\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$. (Allows for possibly complex eigenvalues!)
- Theorem: $\|A\|_2 = \sqrt{\rho(A^\top A)} = \sigma_1(A)$
  - Proof outside of scope.
  - For a sketch of proof, note

    $$\|A\|_2^2 = \max_{\|x\|=1} \|Ax\|_2^2 = \max_{\|x\|=1} x^\top \left(A^\top A\right) x,$$

    and by Courant-Fischer maximum principle applied to the symmetric matrix $A^\top A$, the last expression is the precisely the largest eigenvalue of $A^\top A$. The eigenvalues of $A^\top A$ are nonnegative (follows from the fact that $A^\top A$ is positive semidefinite, or see proof via SVD below), so the largest eigenvalue of $A^\top A$ is equal to $\rho(A^\top A)$. Therefore $\|A\|_2^2 = \rho(A^\top A)$.
  - Note $A^\top A$ is symmetric. Recall spectral theorem says we can diagonalize as $A^\top A = V \Lambda V^\top$, where V is unitary, i.e., $V^{-1} = V^\top$, and Λ is diagonal.
  - Also recall SVD: $A = U \Sigma V^\top$, where U and V are unitary and Σ is nonnegative diagonal. Then $A^\top A = V \Sigma^2 V^\top$. So an SVD of A furnishes a diagonalization of $A^\top A$, and the eigenvalues of $A^\top A$ are the squares of the singular values of A.
  - Hence $\|A\|_2 = \sigma_1(A)$, where $\sigma_1(\,\cdot\,)$ is the function returning the largest singular value.
  - Meanwhile, another theorem says $\|A\|_F = \sqrt{\sum_{i=1}^n \sigma_i(A)^2}$. Thus the Frobenius norm is also related to singular values.
- Theorem: $\rho(A) \leq \|A\|$ for any natural norm $\|\cdot\|$
  - Direct proof.
- Question: when can we say that $\rho(A) = \|A\|$?
- Definition: we say that a matrix is **convergent** if $\lim_{k\to\infty} [A^k]_{ij} = 0$ for all i, j.
  - You could think of this as saying that $\|\text{vec}(A^k)\|_\infty$ goes to zero. All vector norms are equivalent on $\mathbb{R}^{n^2}$, so in fact a matrix is convergent if and only if $\lim_{k\to\infty} \|A^k\|$ for *any* matrix norm.
- Theorem: the following are equivalent
  1. A is a convergent matrix.
  2. $\lim_{n\to\infty} \|\text{vec}(A^n)\| = 0$ for some vector norm $\|\cdot\|$.
  3. $\lim_{n\to\infty} \|\text{vec}(A^n)\| = 0$ for all vector norms $\|\cdot\|$.
  4. $\lim_{n\to\infty} \|A^n\| = 0$ for some matrix norm $\|\cdot\|$.
  5. $\lim_{n\to\infty} \|A^n\| = 0$ for all matrix norms $\|\cdot\|$.
  6. $\rho(A) < 1$.

7. $\lim_{n\to\infty} A^n x = 0$ for all $x \in \mathbb{R}^n$.

   – We are glossing over further detailed proof of the fact that 1-5 are equivalent. Essentially the equivalences of these conditions follows from the fact that all vector norms on $\mathbb{R}^{n\times n}$ (viewed as a vector space), which include all matrix norms in particular, are equivalent.
   – To see (7) implies (1), plug in the standard basis vectors e_i, $i = 1, \ldots, n$.
   – To see (1) implies (7), directly expand and take limits to see $\lim_{k\to\infty}[A^k x]_i = 0$ for all $i = 1, \ldots, n$.
   – **Homework**: explore the equivalence of 1 and 6.

5 The Jacobi method (§7.3)

- Want to solve $Ax = b$.
- Split $A = D - L - U$ (diagonal + strictly lower triangular + strictly upper triangular).
  – **Caution:** L and U are **not** the same as in an LU factorization $A = LU$!
  – Assume that all the diagonal entries are nonzero (so D is **invertible**).
- Then equivalently want to solve $(D - L - U)x = b$
  – Equivalently, $Dx = b + (L + U)x$.
  – Equivalently, $x = D^{-1}[b + (L + U)x]$.
- Therefore we know that the true solution x is a **fixed point** (i.e., a point satisfying $F(x) = x$) of the map
$$F(x) := D^{-1}[b + (L + U)x]$$
(and vice versa).
- By repeated application of the map F, given an initial guess $x^{(0)}$, we can define a sequence via $x^{(k+1)} := F(x^{(k)})$. (In general this approach to finding fixed points is called **fixed point iteration**.)
  – ***If*** $\{x^{(k)}\}_{k=0}^{\infty}$ converges to some point $x^\star$, then taking limit of both sides reveals $x^\star = F(x^\star)$ by continuity of F. Therefore the limit point $x^\star$ is the true solution to $Ax = b$.
  – This method is called the **Jacobi method**. *Note that we do not know yet if/when it actually converges!*
- In summary, the **Jacobi method** is an iterative method with update defined by
$$x^{(k+1)} = D^{-1}[b + (L + U)x^{(k)}].$$
or written out in components:
$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j^{(k)} \right).$$

6 The Gauss-Seidel method (§7.3)

- Recall we want to solve $(D-L-U)x=b$
- We can group the first two terms instead of the last two terms:
  - Equivalently, $(D-L)x=b+Ux$.
  - Or: $x=(D-L)^{-1}(b+Ux)$.
- This suggests the iterative method:

$$x^{(k+1)}=(D-L)^{-1}[b+Ux^{(k)}].$$

  - This is called the **Gauss-Seidel method**.
  - Note that to implement this method we need to solve the **triangular** linear system

$$(D-L)x=\tilde{b},$$

  where $\tilde{b}:=b+Ux^{(k)}$. The solution x then defines the next iterate $x^{(k+1)}$.
  - Since this is a triangular linear system we can efficiently solve it by backsubstitution.
- We can write this out more concretely in components as follows.
  - Consider the equation $(D-L)x^{(k+1)}=b+Ux^{(k)}$ which defines the iteration.
  - Expand out in components as

$$a_{ii}x_i^{(k+1)}+\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}=b_i-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}.$$

  - Solve for $x_i^{(k+1)}$:

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right).$$

  - Note that (as in backsubstitution for a lower triangular matrix), given $x^{(k)}$, we can use this formula to find $x_1^{(k+1)},\dots,x_n^{(k+1)}$ sequentially without solving any equations! This is because the right-hand side of the formula depends only on $x_j^{(k+1)}$ where $j<i$.
- Compare/contrast with the componentwise formula for the Jacobi method.
  - Exercise: count floating point operations for each.

7 Convergence of Jacobi and Gauss-Seidel (§7.3)

- Both the Jacobi and Gauss-Seidel iterations can be rephrased in the general form

$$x^{(k+1)}=Tx^{(k)}+c$$

  for $k=0,1,\dots$.
  - Jacobi case: $T=D^{-1}(L+U)$ and $c=D^{-1}b$.

  - Gauss-seidel case: $T = (D-L)^{-1}U$ and $c = (D-L)^{-1}b$.

- **Recall:** looking for fixed points of the map $F(x) = Tx + c$, i.e., points satisfying $x = Tx + c$.
- **Idea:** Suppose that there exists a solution of the fixed point equation $x = Tx + c$. Then define the "error vector" $y^{(k)} = x^{(k)} - x$ at the k-th iteration, which tells us how far we are from the true solution.
- **Claim:** $y^{(k+1)} = Ty^{(k)}$.
  - To see this, simply compute

$$\begin{aligned} Ty^{(k)} &= T(x^{(k)} - x) \\ &= Tx^{(k)} - Tx \\ &= \left[x^{(k+1)} - c\right] - [x - c] \\ &= x^{(k+1)} - x \\ &= y^{(k+1)}, \end{aligned}$$

    as desired.

- It follows that $y^{(k)} = T^k y^{(0)} = T^k[x^{(0)} - x]$. By our earlier theorem, the limit $\lim_{k\to\infty} y^{(k)} = 0$ for all possible initial guesses $x^{(0)}$ if and only if$\rho(T) < 1$ (i.e., if and only if T is a convergent matrix).
  - Therefore $\lim_{k\to\infty} x^{(k)} = x$ for all initial guesses if and only if $\rho(T) < 1$, provided that there exists a solution to $x = Tx + c$.
- **Question:** but under what condition does there exist a solution to $x = Tx + c$ (for all $c \in \mathbb{R}^n$)?
  - Equivalently, a solution to $(I - T)x = c$ for all c.
  - Equivalently, $I - T$ must be surjective.
  - Since it is square, we have surjective $\iff$ injective $\iff$ invertible.
  - Hence there exists a solution for all c if and only if $I - T$ is invertible.
- Note that $\rho(T) < 1$ implies that $I - T$ is invertible.
  - This is because $I - T$ is singular $\iff$ $I - T$ has 0 as an eigenvalue $\iff$ T has 1 as an eigenvalue.
- **Theorem:** The sequence $\{x^{(k)}\}_{k=0}^{\infty}$ defined by

$$x^{(k+1)} = Tx^{(k)} + c$$

  for $k = 0, 1, \ldots.$ converges to the unique solution of $x = Tx + c$ for all initial conditions $x^{(0)}$ and all $c \in \mathbb{R}^n$ if and only if $\rho(T) < 1$.
- **Proof:**
  - We already proved $\Longleftarrow$.
    * Indeed, we showed $\rho(T) < 1 \implies I - T$ invertible $\implies$ there exists a (unique) solution to $x = Tx + c$. And then this together with $\rho(T) < 1$ imply that $\lim_{k\to\infty} x^{(k)} = x$.
  - Proof of $\implies$:

  * Consider the case $c = 0$. Then the unique solution is $x = 0$, and $x^{(k)} = T^k x^{(0)} \to 0$ for all choices of initial condition $x^{(0)}$.
  * This implies that T is a convergent matrix, so $\rho(T) < 1$.

- **Corollary:** If $\|T\| < 1$ for some natural matrix norm and $c \in \mathbb{R}^n$, then for the sequence $\{x^{(k)}\}$ as defined above, $\lim_{k\to\infty} x^{(k)}$ exists and is equal to the unique solution x of $x = Tx + c$. Moreover, we have the estimates on the rate of convergence:

$$\|x^{(k)} - x\| \leq \|T\|^k \|x^{(0)} - x\|, \quad \|x^{(k)} - x\| \leq \frac{\|T\|^k}{1 - \|T\|} \|x^{(1)} - x^{(0)}\|.$$

- **Proof:**
  - Recall $\|T\| < 1$ implies that $\rho(T) < 1$. This establishes everything but the estimates.
  - For the first estimate, again defining $y^{(k)} := x^{(k)} - x$, recall that $y^{(k)} = T^k y^{(0)}$. Therefore

$$\|y^{(k)}\| \leq \|T^k\| \|y^{(0)}\| \leq \|T\|^k \|y^{(0)}\|$$

(*why does the last inequality hold?*), which is exactly what we wanted.

  - For the second, consider $l > k$, and write

$$\begin{aligned}
x^{(k)} - x^{(l)} &= \left[x^{(k)} - x^{(k+1)}\right] + \left[x^{(k+1)} - x^{(k+2)}\right] \cdots + \cdots \left[x^{(l-1)} - x^{(l)}\right] \\
&= \sum_{j=0}^{l-k-1} \left[x^{(k+j)} - x^{(k+j+1)}\right] \\
&= \sum_{j=0}^{l-k-1} \left[y^{(k+j)} - y^{(k+j+1)}\right] \\
&= \sum_{j=0}^{l-k-1} T^{j+k} \left[y^{(0)} - y^{(1)}\right] \\
&= T^k \left(\sum_{j=0}^{l-k-1} T^j\right) \left[x^{(0)} - x^{(1)}\right]
\end{aligned}$$

so

$$\|x^{(k)} - x^{(l)}\| \leq \|T\|^k \left(\sum_{j=0}^{l-k-1} \|T\|^j\right) \|x^{(1)} - x^{(0)}\|.$$

  - Take the limit of both sides as $l \to \infty$ to obtain

$$\|x^{(k)} - x\| \leq \|T\|^k \left(\sum_{j=0}^{\infty} \|T\|^j\right) \|x^{(1)} - x^{(0)}\|,$$

and taking the sum of the infinite geometric series we obtain

$$\|x^{(k)} - x\| \leq \frac{\|T\|^k}{1 - \|T\|} \|x^{(1)} - x^{(0)}\|,$$

as was to be shown.

- What would it take to apply our convergence result to the Jacobi method, using the infinity norm $\|\cdot\|_\infty$?
  - In this case, $T = D^{-1}(L+U)$, and we need $\|T\|_\infty < 1$.
  - Equivalently, need $\max_i \sum_j |T_{ij}| < 1$.
  - Equivalently, need $\sum_j |T_{ij}| < 1$ for all i.
  - Now $T_{ij} = \begin{cases} -\frac{a_{ij}}{a_{ii}} & i \neq j, \\ 0, & i = j. \end{cases}$
  - Therefore, we need $\sum_{j\neq i} \frac{|a_{ij}|}{|a_{ii}|} < 1$ for all i.
  - Equivalently, need $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i.
- **Definition:** a matrix $A = (a_{ij})$ is called **strictly diagonally dominant** if $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$ for all i.
- Thus we have the following theorem:
- **Theorem:** If A is strictly diagonally dominant, the Jacobi method applied to $Ax = b$ converges for all b and all initial conditions $x^{(0)}$.
  - In fact, the Gauss-Seidel method also converges under the same condition, though we will not prove this.
- Ignoring parallelism, Jacobi and Gauss-Seidel cost the same number of flops per iteration. Which converges in fewer iterations? There is no clean general answer, but a set of sufficient conditions for Gauss-Seidel to be faster is provided by the **Stein-Rosenberg theorem** (see book for full statement).

8 Relaxation techniques (§7.4)

- Review of **positive (semi)definite matrices**.
- **Definition:** a square matrix A is called **positive definite** (written $A \succ 0$) if it is symmetric and $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$, with equality if only if $x = 0$. (A matrix is **positive semidefinite** (written $A \succeq 0$) if it is symmetric and $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$.)
- **Theorem:** the following are equivalent:
  - A is positive definite.
  - $A = B^\top B$ for some square invertible B.
  - A is symmetric with positive eigenvalues.
  - A has a unique positive definite square root, denoted $A^{1/2}$.
- Note that when A is positive definite, then by the spectral theorem we can write $A = U\Lambda U^\top$ where Λ is diagonal with positive diagonal entries and U is orthogonal. In this case, $A^{1/2} = U\Lambda^{1/2}U^\top$, where $\Lambda^{1/2}$ is the diagonal matrix with the square roots of the diagonal entries of Λ on its diagonal.
  - We also can often denote $A^{-1/2} = (A^{1/2})^{-1} = U\Lambda^{1/2}U^\top = U(\Lambda^{1/2})^{-1}U^\top$.
- The **weighted Jacobi method.**

- Recall the splitting $(D - L - U)x = b$.
- Let $\omega \in \mathbb{R}$ be a **relaxation parameter/factor** ($\omega \neq 0$ and consider the equivalent system
$$(\omega D - \omega L - \omega U)x = \omega b.$$
- Moreover, we can split $\omega D = D - (1-\omega)D$ to obtain
$$(D - (1-\omega)D - \omega L - \omega U)x = \omega b.$$
- Then rearrange:
$$Dx = \omega\left[b + (L+U)x\right] + (1-\omega)Dx$$
and solve
$$x = \omega D^{-1}\left[b + (L+U)x\right] + (1-\omega)x$$
or
$$x = \left[(1-\omega)I + \omega D^{-1}(L+U)\right]x + \omega D^{-1}b$$
- This suggests the iteration method
$$x^{(k+1)} = \omega D^{-1}\left[b + (L+U)x^{(k)}\right] + (1-\omega)x^{(k)}.$$
Note that for $\omega \in (0,1)$, this is like taking a a weighted average of the update that would be suggested by ordinary Jacobi with the update that does nothing. Colloquially this is called 'under-relaxation.' If $\omega > 1$, we are extrapolating in the Jacobi direction further than Jacobi prescribes. This is called 'over-relaxation.' Note that the ordinary Jacobi method is recovered in the case $\omega = 1$.
- Equivalently
$$x^{(k+1)} = T_\omega x^{(k)} + c_\omega,$$
where
$$T_\omega = (1-\omega)I + \omega D^{-1}(L+U), \quad c_\omega = \omega D^{-1}b.$$
- In fact, $L + U = D - A$, so $D^{-1}(L+U) = I - D^{-1}A$, and we can alternatively write
$$T_\omega = I - \omega D^{-1}A.$$

- We will analyze convergence under a different assumption, that A is positive definite. Recall that we need $\rho(T_\omega) < 1$.
- **Lemma:** If A is positive definite, then $D^{-1}A$ is diagonalizable with positive eigenvalues.
  - **Proof:** In fact a more general theorem is true. If A and B are positive definite, then BA is diagonalizable with positive eigenvalues. We'll prove this.
  - Note that $B^{1/2}AB^{1/2}$ is positive definite (why?), hence diagonalizable by the spectral theorem, and we can write $B^{1/2}AB^{1/2} = U\Lambda U^{-1}$ where Λ is diagonal with positive diagonal entries.

    * Here $B^{1/2}$ indicates the unique positive definite square root of B, satisfying $B^{1/2}B^{1/2} = B$, and $B^{-1/2}$ will indicate its inverse.

  - Then

$$\begin{aligned} BA &= B^{1/2}\left(B^{1/2}AB^{1/2}\right)B^{-1/2} \\ &= B^{1/2}U\Lambda U^{-1}B^{-1/2} \\ &= (B^{1/2}U)\Lambda(B^{1/2}U)^{-1} \\ &= P\Lambda P^{-1}, \end{aligned}$$

  where $P := B^{1/2}U$ and we have diagonalized BA with positive eigenvalues.

- **Lemma:** λ is an eigenvalue of $D^{-1}A$ if and only if $1-\omega\lambda$ is an eigenvalue of T_ω.
- **Corollary:** If A is positive definite and $0 < \omega < \frac{2}{\rho(D^{-1}A)}$, then the weighted Jacobi method with parameter ω always converges.
- How to determine the best choice of ω? Discuss.
- **Successive over-relaxation (SOR)** for Gauss-Seidel:
  - Recall the equation

$$(D-(1-\omega)D-\omega L-\omega U)x = \omega b,$$

  but now rearrange as

$$(D-\omega L)x = \omega[b+Ux]+(1-\omega)Dx,$$

  or

$$(D-\omega L)x = [(1-\omega)D+\omega U]\,x+\omega b,$$

  and solve to obtain

$$x = (D-\omega L)^{-1}\left\{[(1-\omega)D+\omega U]\,x+\omega b\right\},$$

  which suggests the iteration

$$x^{(k+1)} = (D-\omega L)^{-1}\left\{[(1-\omega)D+\omega U]\,x^{(k)}+\omega b\right\}.$$

  - This can equivalently be viewed as an iteration of the form

$$x^{(k+1)} = T_\omega x^{(k)} + c_\omega,$$

  where

$$T_\omega = (D-\omega L)^{-1}\left[(1-\omega)D+\omega U\right], \quad c_\omega = \omega(D-\omega L)^{-1}b.$$

  - This method is called **successive over-relaxation (SOR)**.
- **Theorem (Kahan):** If $a_{ii} \neq 0$ for all i, then $\rho(T_\omega) \geq |\omega-1|$. Therefore SOR can only converge if $\omega \in (0,2)$.
  - **Proof:** exercise.
- **Theorem (Ostrowski-Reich):** If A is a positive definite matrix and $\omega \in (0,2)$, then SOR always converges.
  - Note that in particular.
- How to determine the best choice of ω? In general, it is unclear, but an optimal choice is provided in the book in the case of tridiagonal positive definite matrices.

9 Condition numbers (§7.5)

- Given a guess $\tilde{x}$ for the solution of $Ax = b$, define the **residual vector** $r = b - A\tilde{x}$**.**
- The residual vector tells us how far we are from satisfying the equations $Ax = b$.
- When $\|r\|$ is small, it should mean we are close to a solution. How closely does it correspond to the error of the solution vector $\|\tilde{x} - x\|$?
- **Examples:**
  - Consider the linear system
$$\begin{pmatrix} 1 & 0 \\ 0 & 0.0001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$
    which by inspection admits the solution $x = (1, 0)^\top$.
    * Consider the guess $\tilde{x} = (1, 10)^\top$. The residual $r = (0, 0.01)^\top$ is small, but the guess is not very good.
  - Consider the linear system
$$\begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3.0001 \end{pmatrix},$$
    which by inspection admits the solution $x = (1, 1)^\top$.
    * Consider the guess $\tilde{x} = (3, -0.0001)$. The residual $r = (0.0002, 0)^\top$ is small, but the guess is not very good.
- **Theorem:** Suppose $Ax = b$, where A is invertible, and $\tilde{x}$ is a guess defining the residual $r = b - A\tilde{x}$. Then for any natural norm,
$$\|x - \tilde{x}\| \leq \|r\| \, \|A^{-1}\|.$$
  Moreover, if $x, b \neq 0$, then
$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \|A\| \, \|A^{-1}\| \, \frac{\|r\|}{\|b\|}.$$
  - **Proof:**
    * Note $x = A^{-1}b$ and $\tilde{x} = A^{-1}(b - r)$. Therefore
$$x - \tilde{x} = A^{-1}r,$$
      and the first desired inequality
$$\|x - \tilde{x}\| \leq \|r\| \, \|A^{-1}\|.$$
      follows.
    * Meanwhile, since $b = Ax$, we have $\|b\| \leq \|A\| \, \|x\|$, so $1/\|x\| \leq \|A\|/\|b\|$, from which the second inequality follows.
- **Definition:** the **condition number** $K(A)$ of an invertible matrix A relative to a norm $\| \cdot \|$ is defined $K(A) = \|A\| \, \|A^{-1}\|$.
  - Informally, we say that a matrix is well-conditioned if $K(A) \approx 1$, ill-conditioned if $K(A) \gg 1$.
- **Theorem:** $K(A) \geq 1$.
  - **Proof:** Note $AA^{-1} = I$, take norms of both sides and use submultiplicativity.

10 Positive definite systems (§7.6)

- Again consider $Ax = b$, where A is positive definite.
- **Aside:** any linear system can be reduced to a positive definite linear system. Indeed, consider the general linear system $Bx = c$ where B is invertible.
  - We can view the problem of solving this system as an optimization problem:
  $$\min_{x \in \mathbb{R}^n} \|Bx - c\|_2^2$$
  with the unique optimizer B. However, note that
  $$\begin{aligned} \|Bx - c\|_2^2 &= (Bx - c)^\top (Bx - c) \\ &= (Bx)^\top Bx - c^\top Bx - (Bx)^\top c + c^\top c \\ &= x^\top B^\top Bx - 2(B^\top c) \cdot x + c^\top c. \end{aligned}$$
  - Then taking gradient and setting equal to zero we obtain the new equations
  $$B^\top Bx = B^\top c.$$
    * We have used the general facts that $\nabla(x^\top Cx) = 2Cx$ and $\nabla(a \cdot x) = a$ for any symmetric matrix C and any vector a, which will be verified in **homework**.
  - Note that this is a positive definite linear system with matrix $B^\top B$ and right-hand side $B^\top c$. Note that $B^\top B$ is automatically positive definite.
  - A simpler point of view is simply to note that the solution $x = B^{-1}c$ can be written
  $$B^{-1}c = (B^\top B)^{-1} B^\top c,$$
  and we can view the expression on the right as the solution of $B^\top Bx = B^\top c$.
  - However, more generally, even when B is a rectangular $m \times n$ matrix so that usually it is impossible to solve $Bx = c$ exactly, we can instead solve the least squares problem $\min_{x \in \mathbb{R}^n} \|Bx - c\|_2^2$, which always leads to a square positive definite (as long as B has full column rank) linear system $B^\top Bx = B^\top c$. In general this linear system can be referred to as the **normal equations**.
  - Still, given a square invertible linear system $Bx = c$, it is not always advantageous to instead solve $B^\top Bx = B^\top c$. Often approaching $Bx = c$ directly is a better idea. Nonetheless it is good to keep in mind that **once you have the power to solve positive definite linear systems, you have the power to solve all linear systems!**
- Let us now suppose that A *positive definite* and b are given to us.
- **Theorem:** the solution x of a positive definite linear system is the unique optimizer of the objective function
$$g(x) = \frac{1}{2} x^\top Ax - b^\top x.$$
  - **Proof:**

        * *First-order optimality condition* is that $\nabla g(x) = 0$. But

$$\nabla g(x) = Ax - b.$$

          So $\nabla g(x) = 0$ if and only if $Ax = b$. Hence the only critical point is the unique solution.

        * In fact the Hessian matrx $\nabla^2 g(x) \equiv A$ is positive definite everywhere, so g is a (strictly) *convex* function. Therefore any critical point is a global optimizer. See the book for a more detailed proof.

    - **Alternative proof:**
        * View the optmization problem as equivalent to a least squares problem.
        * Recall that a least squares problem of minimizing $\|Bx - c\|_2^2$ yields a linear system $B^\top Bx = B^\top c$.
        * Motivation: find B such that $B^\top B = A$ and then $B^\top c = b$.
            · There exist many choices of B, but simply take $B = A^{1/2}$. Then accordingly take $c = (B^\top)^{-1}b$.
        * Then the least squares problem of minimizing $f(x) = \|Bx - c\|_2^2$ is (by earlier derivation) yields the unique solution. By earlier computations:

$$\begin{aligned} f(x) &= x^\top B^\top Bx - 2(B^\top c) \cdot x + c^\top c \\ &= x^\top Ax - 2b \cdot x + c^\top c, \end{aligned}$$

          which differs from g only by a constant and a rescaling by a factor of 2.

        * Therefore optimizing f is equivalent to optimizing g.

- Draw some pictures illustrating the difficulty of optimizing $g(x)$ for ill-conditioned A via steepest descent.

11 Steepest descent (§7.6)

- **Idea (to be improved):** optimize $g(x)$ by *steepest descent*.
    - Given a guess $x^{(k-1)}$, define

$$x^{(k)} = x^{(k-1)} + t^{(k)}v^{(k)},$$

      where $v^{(k)}$ is the direction of *steepest descent* for the objective $g(x)$ at the guess $x^{(k-1)}$ and $t^{(k)} \in \mathbb{R}$ is a scalar.

    - Multivariate calculus tells us that the direction of steepest descent is

$$v^{(k)} = -\nabla g(x^{(k-1)}) = b - Ax^{(k-1)} =: r^{(k-1)},$$

      which is the *residual*!

    - Moreover we want to find an optimal scalar $t^{(k)}$ given the search direction which we can find by solving

$$\min_{t \in \mathbb{R}} h(t), \quad h(t) := g(x^{(k-1)} + tv^{(k)}).$$

  - For simplicity of notation, fix $x = x^{(k-1)}$, $r = r^{(k-1)}$, $v = v^{(k)}$, and expand

$$\begin{aligned} h(t) &= \frac{1}{2}(x+tv)^\top A(x+tv) - b\cdot(x+tv) \\ &= \frac{t^2}{2}v^\top Av + t\,(v^\top Ax) - t(v^\top b) + \left[\frac{1}{2}x^\top Ax - b\cdot x\right]. \end{aligned}$$

  - Take derivative and set equal to zero to obtain

$$t = \frac{v^\top(b-Ax)}{v^\top Av} = \frac{v^\top r}{v^\top Av}.$$

  - Using the inner product notation for the dot product $\langle x,y\rangle = x\cdot y = x^\top y = y^\top x$, we can also write

$$t^{(k)} = \frac{\left\langle v^{(k)}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle}.$$

  - For steepest descent, it happens to be the case that $v^{(k)} = r^{(k-1)}$, but for a general search direction, this derivation of $t^{(k)}$ still holds.

- In summary, **steepest descent pseudocode**:
  - Given initial guess $x^{(0)} \in \mathbb{R}^n$, right-hand side b, the ability to compute matrix-vector products by a positive definite matrix A, maximum number of iterations m, and a relative residual norm tolerance ε for some choice of norm $\|\cdot\|$.
  - Set $r^{(0)} = b - Ax^{(0)}$.
  - For $k = 1, 2, \dots, m$:
    * Set search direction $v^{(k)} = r^{(k-1)}$.
    * Set $t^{(k)} = \frac{\left\langle v^{(k)}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle}$.
    * Set $x^{(k)} = x^{(k-1)} + t^{(k)}v^{(k)}$.
    * Set $r^{(k)} = b - Ax^{(k)}$.
    * If $\|r^{(k)}\|/\|b\| \le \varepsilon$, **BREAK**.

12 The conjugate gradient method (§7.6)

- Back to picture.
  - Notice that with steepest descent we make the same mistake repeatedly. *We want to improve our search directions $v^{(k)}$ to take into account past history.* Once a search direction is chosen, we will choose t by th same univariate optimization procedure.
  - Imagine the situation if $A = I$. As long as our search directions are all orthogonal, we will converge ***exactly*** in n steps.
  - Consider the change of variable $y = Bx$, where B is such that $B^\top B = A$. Therefore $B^{-\top}AB^{-1} = I$. Here $B^{-\top}$ denotes $(B^{-1})^\top = (B^\top)^{-1}$. For instance we could take $B = A^{1/2}$ which is symmetric.

- . Then we could seek to optimize the transformed objective function

$$\tilde{g}(y) := g(B^{-1}y)$$

over y and then recover the solution as $x = B^{-1}y$.

- Expand the transformed objective as

$$\begin{aligned}\tilde{g}(y) &= \frac{1}{2}(B^{-1}y)^\top AB^{-1}y - b^\top B^{-1}y \\ &= \frac{1}{2}(B^{-1}y)^\top A(B^{-1}y) - (B^{-\top}b)^\top y \\ &= \frac{1}{2}y^\top B^{-\top}AB^{-1} - \tilde{b}\cdot y \\ &= \frac{1}{2}y^\top y - \tilde{b}\cdot y.\end{aligned}$$

- In other words, we are solving the linear system with matrix I and right-hand side $\tilde{b} = B^{-\top}$.

- Therefore we want a sequence of search directions $v^{(k)}$ such that after transformation, they are all orthogonal.
  - Search directions v transform to Bv.
  - Therfore we want $\left[Bv^{(k)}\right]\cdot\left[Bv^{(l)}\right] = 0$ if $k \neq l$.
  - This means we want

$$0 = \left[Bv^{(k)}\right]^\top\left[Bv^{(l)}\right] = v^{(k)\top}B^\top Bv^{(l)} = v^{(k)\top}Av^{(l)}.$$

  - This motivates the following definition.

- **Definition:** a collection of vectors $v^{(1)},\ldots,v^{(m)}$ is called A-**orthogonal** if

$$\left\langle v^{(k)}, v^{(l)}\right\rangle_A := v^{(k)\top}Av^{(l)} = \left\langle v^{(k)}, Av^{(l)}\right\rangle = 0$$

whenever $k \neq l$. It is called A-**orthonormal** if additionally $\left\langle v^{(k)}, v^{(k)}\right\rangle_A = 1$ for all $k = 1,\ldots,m$.

  - The definition $\langle u, v\rangle_A := u^\top Av$ actually defines an ***inner product*** in the general sense of Math 110. Therefore we can A-orthogonalize/orthonormalize a collection of vectors by the generalized Gram-Schmidt procedure.

- **Idea:** at each iteration, consider a starter guess for the search direction given by the steepest descent direction $r^{(k-1)} := b - Ax^{(k-1)}$, which is the residual. Then correct this guess to our actual choice of search direction $v^{(k)}$ by A-orthogonalizing with respect to the preceding $v^{(1)},\ldots,v^{(k-1)}$. This way, we maintain that our collection of search directions is A-orthogonal. Then pick $t^{(k)}$ by line search and update $x^{(k)} = x^{(k-1)} + t^{(k)}v^{(k)}$.
  - How to A-orthogonalize? Let

$$v^{(k)} = r^{(k-1)} - \sum_{j=1}^{k-1}\frac{\left\langle r^{(k-1)}, v^{(j)}\right\rangle_A}{\left\langle v^{(j)}, v^{(j)}\right\rangle_A}v^{(j)},$$

just like Gram-Schmidt. Can verify that this extends $v^{(1)},\ldots,v^{(k-1)}$ to remain A-orthogonal.

- **Amazing, amazing fact:** Following this procedure, we in fact have at every iteration that $\left\langle r^{(k)}, v^{(j)} \right\rangle_A$ for all $j < k-1$. Hence, for the first iteration we choose

$$v^{(k)} = r^{(k-1)}$$

and for all subsequent iterations we choose

$$v^{(k)} = r^{(k-1)} - \frac{\left\langle r^{(k-1)}, v^{(k-1)} \right\rangle_A}{\left\langle v^{(k-1)}, v^{(k-1)} \right\rangle_A} v^{(k-1)},$$

or

$$v^{(k)} = r^{(k-1)} + s^{(k-1)} v^{(k-1)},$$

where

$$s^{(k-1)} := -\frac{\left\langle r^{(k-1)}, Av^{(k-1)} \right\rangle}{\left\langle v^{(k-1)}, Av^{(k-1)} \right\rangle}.$$

- In summary, pseudocode for the **conjugate gradient (CG) method**:

```
– Given initial guess $x^{(0)} \in \mathbb{R}^n$, right-hand side $b$, the ability to compute matrix-vector
  products by a positive definite matrix $A$, maximum number of iterations $m$, and a relative
  residual norm tolerance $\varepsilon$ for some choice of norm $\| \cdot \|$.
– Compute $r^{(0)} = b - Ax^{(0)}$.
– For $k = 1, \ldots, m$:
    ∗ If $k = 1$:
        · Set $v^{(1)} = r^{(0)}$.
    ∗ Else:
        · Set $v^{(k)} = r^{(k-1)} + s^{(k-1)} v^{(k-1)}$.
    ∗ Set $t^{(k)} = \frac{\langle v^{(k)}, r^{(k-1)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$.
    ∗ Set $x^{(k)} = x^{(k-1)} + t^{(k)} v^{(k)}$.
    ∗ Set $r^{(k)} = b - Ax^{(k)}$.
    ∗ If $\|r^{(k)}\| / \|b\| \le \varepsilon$, BREAK.
    ∗ Compute $s^{(k)} = -\frac{\langle r^{(k)}, Av^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$.
```

- Computational complexity / bottleneck?
  - Usually the total cost of the algorithm can be roughly measured in terms of the total number of matrix-vector multiplications (***matvecs***) by A.
  - If you store the results $u^{(k)} := Av^{(k)}$ so that you never have to recompute them, we actually need only ***one*** matvec per iteration of CG (*same as steepest descent!*)
  - But wait, it looks like we also need to compute $Ax^{(k)}$ in order to update the residual!
    - ∗ Not really, because $x^{(k)} = x^{(k-1)} + t^{(k)} v^{(k)}$, hence $Ax^{(k)} = Ax^{(k-1)} + t^{(k)} Av^{(k)}$.
    - ∗ Therefore if we make sure have have stored $Ax^{(k-1)}$, we can keep track of the $Ax^{(0)}, Ax^{(1)}, Ax^{(2)}, \ldots$ recursively without additional matvecs per iteration (after the first one).
    - ∗ Define $w^{(k)} := Ax^{(k)}$ so
$$w^{(k)} = w^{(k-1)} + t^{(k)} u^{(k)}.$$

- Matvec-efficient pseudocode for the **conjugate gradient (CG) method**:

```
  – Given initial guess $x^{(0)} \in \mathbb{R}^n$, right-hand side $b$, the ability to compute matrix-vector
    products by a positive definite matrix $A$, maximum number of iterations $m$, and a relative
    residual norm tolerance $\varepsilon$ for some choice of norm $\|\cdot\|$.
  – Compute $w^{(0)} = Ax^{(0)}$.
  – Set $r^{(0)} = b - w^{(0)}$.
  – For $k = 1, \ldots, m$:
      ∗ If $k = 1$:
          · Set $v^{(1)} = r^{(0)}$.
      ∗ Else:
          · Set $v^{(k)} = r^{(k-1)} + s^{(k-1)} v^{(k-1)}$.
      ∗ Compute $u^{(k)} = Av^{(k)}$.
      ∗ Set $t^{(k)} = \frac{\langle v^{(k)}, r^{(k-1)} \rangle}{\langle v^{(k)}, u^{(k)} \rangle}$.
      ∗ Set $x^{(k)} = x^{(k-1)} + t^{(k)} v^{(k)}$.
      ∗ Set $w^{(k)} = w^{(k-1)} + t^{(k)} u^{(k)}$.
      ∗ Set $r^{(k)} = b - Ax^{(k)}$.
      ∗ If $\|r^{(k)}\| / \|b\| \leq \varepsilon$, BREAK.
      ∗ Compute $s^{(k)} = -\frac{\langle r^{(k)}, u^{(k)} \rangle}{\langle v^{(k)}, u^{(k)} \rangle}$.
```

- This pseudocode is still not memory-efficient because we can throw away some vectors once computed that we won't need again. See the textbook for a terse efficient pseudocode.

13 Convergence of CG (§7.6)

- How fast does CG converge?
- Our earlier picture proof arguments via A-orthogonality yield a handwavy proof of the following:
- **Theorem:** The CG algorithm, applied to $Ax = b$ where A is positive definite, recovers the exact solution in n iterations.
  - In other words, $x^{(k)} = x$ for all $k \geq n$.
- Therefore, if one matvec costs $O(n)$ operations (which is the case for many sparse matrices A), we can solve the linear system in only $O(n^2)$ operations, which is better than the $O(n^3)$ general scaling of a direct method.
- However, *the true power of CG lies in the fact that CG can converge approximately in* $\ll n$ *iterations*, depending on the conditioning in A.
- **Theorem:** If $K(A)$ is the condition number of a positive definite matrix A with respect to the 2-norm, then the iterates of the CG algorithm as applied to $Ax = b$ satisfy

$$\|x^{(k)} - x\|_2 \leq 2K(A) \left(\frac{\sqrt{K(A)} - 1}{\sqrt{K(A)} + 1} \right)^k \|x^{(0)} - x\|_2.$$

- **Recall** (computation in class): for positive definite A, $K(A)$ with respect to the 2-norm is the ratio of the largest and smallest eigenvalues of A.
- When $K(A)$ is very large, the base of the exponent is approximately

$$1 - \frac{2}{\sqrt{K(A)}}.$$

- Meanwhile, Jacobi and Gauss-Seidel can be shown to converge exponentially with base approximately $1 - \frac{2}{K(A)}$.

14 Preconditioning (§7.6)

- The previous convergence theorem motivates the question: if the condition number of A is large, what should we do?
- Suppose $A \approx CC^\top$ and C and $C^\top$ are easy to invert.
- Then $\tilde{A} := C^{-1}AC^{-\top} \approx I$, hence is well-conditioned (and still positive definite).
- We still have the exact equation $A = C\tilde{A}C^\top$
- Then instead of solving $Ax = b$, we can solve $C\tilde{A}C^\top x = b \iff \tilde{A}(C^\top x) = C^{-1}b$.
- Defining $\tilde{b} := C^{-1}b$, then we can instead solve $\tilde{A}\tilde{x} = \tilde{b}$ for $\tilde{x}$ and then recover $x = C^{-\top}\tilde{x}$.
- Idea of **preconditioned CG**: apply CG to the positive definite linear system $\tilde{A}\tilde{x} = \tilde{b}$.
  - See book for pseudocode.
- This requires us to perform matvecs by $\tilde{A}$, which can be achieved via matvecs by A, C^{-1}, and $C^{-\top}$.
- Finding a good choice of C (i.e., achieving $C^{-1}AC^{-\top} \approx I$) such that matvecs by C^{-1} and $C^{-\top}$ are efficient is *hard in general*.
- One choice that is often worth trying is **diagonal preconditioner**:

$$C = D^{1/2},$$

where D is the diagonal part of A. Then $C^{-1} = C^{-\top}$ are just diagonal matrices with diagonal entries $\frac{1}{\sqrt{a_{ii}}}$. The resulting $\tilde{A} = (\tilde{a}_{ij})$ has $\tilde{a}_{ii} = 1$ for all i.

Part II

Approximation theory

15 Least squares approximation (§8.1)

- Suppose we are given 'data' (x_i, y_i), $i = 1, \ldots, m$. Here $x_i, y_i \in \mathbb{R}$.

- We want to find a function f that fits the data as well as possible, i.e., achieves

$$f(x_i) \approx y_i,$$

and moreover, we want to restrict f to lie within a certain restricted class of functions (usually to avoid *overfitting*).

- For example, we could consider only linear fitting functions.
  - These are parametrized by two parameters, a slope a_1 and in intercept a_0, via

$$f_{\mathbf{a}}(x) = a_1 x + a_0,$$

  where $\mathbf{a} = (a_0, a_1)$ emphasizes the parametric dependence of f on the parameters collected in the vector $\mathbf{a}$.

- More generally, we can consider a more general collection of parameters a parametrizing a family of functions $f_{\mathbf{a}}$.
  - For example, we can consider all polynomials of order n, parametrized by

$$f_{\mathbf{a}}(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = \sum_{j=0}^{n} a_j x^j,$$

  where $\mathbf{a} = (a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$.

- Even more generally, suppose we have an arbitrary set of basis functions $g_0, \ldots, g_{p-1}$. Then we can consider

$$f_{\mathbf{a}}(x) = \sum_{j=0}^{p-1} a_j g_j(x). \quad (\star)$$

  - This recovers polynomial choice via the choice $g_j(x) = x^j$, $j = 0, \ldots, p-1$.
  - The vector of parameters $\mathbf{a} = (a_0, \ldots, a_{p-1}) \in \mathbb{R}^p$ is p-dimensional, but we will adopt the convention that we use zero-indexing for its components, i.e., we start with index zero.

- In **least squares** approximation, we seek to solve

$$\min_{\mathbf{a} \in \mathbb{R}^p} F(\mathbf{a}), \quad F(\mathbf{a}) := \sum_{i=1}^{n} (f_{\mathbf{a}}(x_i) - y_i)^2.$$

  - In the setting of $(\star)$, this can be reduced to a linear algebra problem!
  - Otherwise, it can only be viewed as a general optimization problem.

- Let us unpack:

$$\begin{aligned}
\sum_{i=1}^{n} (f_{\mathbf{a}}(x_i) - y_i)^2 &= \sum_{i=1}^{n} \left(\left[\sum_{j=0}^{p-1} a_j g_j(x_i) \right] - y_i \right)^2 \\
&= \sum_{i=1}^{n} \left(\sum_{j=0}^{p-1} g_{ij} a_j - y_i \right)^2 \\
&= \sum_{i=1}^{n} \left([G\mathbf{a}]_i - y_i \right)^2 \\
&= \|G\mathbf{a} - y\|_2^2,
\end{aligned}$$

where we have defined the matrix $G = (g_{ij}) = (g_j(x_i))$, for $i = 1, \ldots, m$ and $j = 0, \ldots, p-1$.

  - Somewhat confusingly, we have the row indexing starting at 1 and the column indexing starting at 0. Sorry, the reason is that this is more intuitive based on the interpretation, e.g., in polynomial least squares!
  - Therefore the matrix is $m \times p$.

- Therefore we want so solve
$$\min_{\mathbf{a}\in\mathbb{R}^p} \|G\mathbf{a} - y\|_2^2.$$

- We solved precisely this type of problem in the last part of the course!
  - The answer is given by solving the **normal equations**, which form a positive definite linear system:
$$(G^\top G)\mathbf{a} = G^\top y.$$
  - The normal equations admit a unique solution if G has rank p. In particular, since G is $m \times p$ we must have $m \geq p$.
  - In the setting of **polynomial least squares** corresponding to the choice $g_j(x) = x^j$, it can be shown that a unique solution exists if the data points x_i, $i = 1, \ldots, m$ are all distinct.

- Sometimes we want to fit the data (x_i, y_i), $i = 1, \ldots, m$ within a function class that is not parametrized in the form of $(\star)$.
  - This becomes a general optimization problem that cannot be solved simply by solving a linear system! For example, neural networks.
  - However, sometimes, you can transform your problem to the form $(\star)$.

- For example, suppose I want to fit within the class of exponential functions
$$f_{b,c}(x) = be^{cx}$$
parametrized by $b, c \in \mathbb{R}$.
  - I want to find b, c that achieve
$$y_i \approx f_{b,c}(x_i) = be^{cx_i},$$
but I can take the log of both sides and instead try to achieve
$$\log y_i \approx \log b + cx_i.$$
  - Define modified data $\tilde{y}_i = \log y_i$ and transformed parameters $a_0 = \log b$, $a_1 = c$, so we instead seek to achieve
$$\tilde{y}_i = a_0 + a_1 x_i$$
for all i.
  - Therefore we are motivated to try ***linear*** least squares fitting on the data $(x_i, \tilde{y}_i)$.
    * **Note with caution:** this isn't the same thing as doing least squares fitting within the original nonlinear function class! But at least we can do it without resorting to general optimization.
  - Then after we solve for the optimal parameters a_0, a_1, we can recover $b = e^{a_0}$ and $c = a_1$.

16 Interlude: inner products and Gram-Schmidt

- We will next be considering functions $f : [a, b] \to \mathbb{R}$ defined on an entire continuous interval.
- We will review a few relevant concepts from Math 110 about general inner products.
- **Definition:** an inner product $\langle \cdot, \cdot \rangle$ on a vector space $\mathcal{V}$ is a function $\mathcal{V} \times \mathcal{V} \to \mathbb{R}$ satisfying the following properties for all $u, v, w \in \mathcal{V}$ and all $\lambda \in \mathbb{R}$:
  - Linearity in the first slot: $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$
  - Homogeneity in the first slot: $\langle \lambda u, v \rangle = \lambda \langle u, w \rangle$
  - Symmetry: $\langle u, v \rangle = \langle v, u \rangle$
  - Positive definiteness: $\langle u, u \rangle \geq 0$ with equality if and only if $u = 0$.
- The most famous inner product is the dot product on $\mathbb{R}^n$.
  - Note that earlier in the discussion of CG we saw that "A inner product."
  - Most of the properties of dot products are general to inner products.
- Any inner product induces a vector space norm via $\|u\| = \sqrt{\langle u, u \rangle}$. In the case of the dot product this yields the 2-norm.
- Inner products define a general notion of **orthogonality**: we say that vectors u, v are orthogonal if $\langle u, v \rangle = 0$.
  - This allows us to do Gram-Schmidt.
- The next most famous inner product is the L^2 inner product defined on the vector space $C([a, b])$ of continuous functions $[a, b] \to \mathbb{R}$.
  - Given $f, g : [a, b] \to \mathbb{R}$, we define

$$\langle f, g \rangle_{L^2([a,b])} = \int_a^b f(x) g(x)\, dx.$$

  - It is like a continuous version of the dot product, replacing a sum with an integral.
- Recall we can think of $C([a, b])$ as a vector space...
  - $(f + g)(x) = f(x) + g(x)$, for $f, g \in C([a, b])$.
  - $(\lambda f)(x) = \lambda f(x)$, for $\lambda \in \mathbb{R}$ and $f \in C([a, b])$.
  - The function that is uniformly zero is the zero vector.
  - A collection of functions $f_1, \ldots, f_k$ are **linearly independent** if the only choice of coefficients $c_1, \ldots, c_k$ such that the linear combination $\sum_{i=1}^k c_i f_i$ is the zero vector, i.e., $\sum_{i=1}^k c_i f_i(x) = 0$ for all $x \in [a, b]$, is the choice $c_1 = \cdots = c_k = 0$.
  - An expression of the form $\sum_{i=1}^k c_i f_i$ is called a **linear combination** of the functions $f_1, \ldots, f_k$.
  - The span of $f_1, \ldots, f_n$, denoted $\operatorname{span}(f_1, \ldots, f_n)$ is the set of all linear combinations of $f_1, \ldots, f_n$.
  - A linearly independent set forms a **basis** for its span.

- We can also think of $\mathbb{P}_n$, the set of all polynomials of order at most n, as a subspace of $C([a,b])$, since all polynomials define continuous functions on $[a,b]$. Moreover, the set of polynomials of order at most n is closed under addition and scalar multiplication.
  - It is useful to notice that the polynomials $M_k(x) = x^k$, $k = 0, \dots, n$ form a basis for $\mathbb{P}_n$.
  - Indeed, for coefficients c_k, the linear combination $\sum_{k=0}^n c_k M_k$ is the polynomial which evaluates to $\sum_{k=0}^n c_k x^k$, which is an arbitrary polynomial of order at most n. You can also check that the M_k are linearly independent.
- We will also consider the weighted L^2 inner product, defined by a nonnegative-valued weight function $w : (a,b) \to [0,\infty)$.
  - Given $f, g : [a,b] \to \mathbb{R}$, we define the weighted L^2 inner product

$$\langle f, g \rangle_{L^2([a,b];w)} = \int_a^b f(x)g(x)\,w(x)\,dx.$$

- Given an orthogonal collection of vectors $v_1, \dots, v_k$ and a new vector w, we can find a new vector v_{k+1} completing $v_1, \dots, v_{k+1}$ to an orthogonal collection, with the same span (set of linear combinations) as $v_1, \dots, v_k, w$.
  - This is the **Gram-Schmidt procedure** and is specified by the formula

$$v_{k+1} = w - \sum_{i=1}^k \frac{\langle w, v_i \rangle}{\langle v_i, v_i \rangle} v_i.$$

  - One can check directly that $\langle v_{k+1}, v_i \rangle = 0$ for all $i = 1, \dots, k$.

17 Least squares over an interval (§8.2)

- In the preceding section on least squares approximation, we were given a *discrete* set $x_1, \dots, x_m$ of points on which we intend to approximate a function.
- In this section, we will be concerned with approximating a function $f(x)$ on an *entire interval* $[a,b]$.
- Can again suppose we want to fit f with a function of the form

$$f_{\mathbf{a}}(x) = \sum_{j=0}^{p-1} a_j g_j(x),$$

  except now we have 'infinitely many' data points $(x, f(x))$, $x \in [a,b]$.
- What would happen if we took $x_1, \dots, x_m$ to be an increasingly fine grid, e.g.,

$$x_i = a + ih,$$

  where $h = \Delta x = (b-a)/m$ and our $y_i = f(x_i)$ to be the exact function values?
- Note that the least squares objective is (up to overall scaling factor which does not affect the optimizer):

$$\frac{1}{m}\sum_{i=1}^m (f_{\mathbf{a}}(x_i) - f(x_i))^2 \underset{m\to\infty}{\longrightarrow} \int (f_{\mathbf{a}}(x) - f(x))^2\,dx = \|f_{\mathbf{a}} - f\|^2_{L^2([a,b])}.$$

- Let us also take the limit of the normal equations and see what happens!
  - Recall that the normal equations were
$$G^\top G\mathbf{a} = G^\top y.$$
  - It is equivalent to consider
$$hG^\top G\mathbf{a} = hG^\top y, \quad (\star)$$
    and we will actually take the limit of both sides of this equation.
  - Observe
$$\begin{aligned}[hG^\top G]_{ij} &= \sum_{k=1}^{m} G_{ki}G_{kj}\, h \\ &= \sum_{k=1}^{m} g_i(x_k)g_j(x_k)\, h \\ &\approx \int_a^b g_i(x)g_j(x)\, dx,\end{aligned}$$
    where we note in the last line that we have a Riemann sum approximation, which becomes exact in the limit as $m \to \infty$.
  - Meanwhile
$$\begin{aligned}[hG^\top y]_i &= \sum_{k=1}^{m} G_{ki}y_k\, h \\ &= \sum_{k=1}^{m} g_i(x_k)f(x_k)\, h \\ &\approx \int_a^b g_i(x)f(x)\, dx.\end{aligned}$$
  - Therefore define the matrix
$$a_{ij} = [hG^\top G]_{ij} = \int_a^b g_i(x)g_j(x)\, dx = \langle g_i, g_j\rangle_{L^2([a,b])}$$
    and the vector
$$b_i = \int_a^b g_i(x)f(x)\, dx = \langle g_i, f\rangle_{L^2([a,b])}$$
    then the normal equations become
$$A\mathbf{a} = b.$$
- **Summary:** to solve
$$\min_{\mathbf{a}\in\mathbb{R}^p} \|f_{\mathbf{a}} - f\|^2_{L^2([a,b])},$$
  we form the **'Gram matrix'** (a general name for a matrix of inner products)
$$A_{ij} = \langle g_i, g_j\rangle_{L^2([a,b])}$$
  and the vector
$$b_i = \langle g_i, f\rangle_{L^2([a,b])}.$$
  Then we solve for **a** and recover
$$f_{\mathbf{a}}(x) = \sum_{j=0}^{p-1} a_j g_j(x).$$

– In some cases, we may be able to compute A and b analytically. In others, we may need to estimate them by numerical quadrature.

18 Weighted least squares over an interval (§8.2)

- What if instead we want to find

$$\min_{\mathbf{a}\in\mathbb{R}^p} \|f_{\mathbf{a}} - f\|^2_{L^2([a,b];w)},$$

where w is a weight function as mentioned earlier?

- Skipping to the conclusion, we again form the Gram matrix

$$A_{ij} = \langle g_i, g_j \rangle_{L^2([a,b];w)}$$

and the vector

$$b_i = \langle g_i, f \rangle_{L^2([a,b];w)}$$

using the appropriate inner product. Then we solve for $\mathbf{a}$ and recover

$$f_{\mathbf{a}}(x) = \sum_{j=0}^{p-1} a_j g_j(x).$$

- Note that we recover unweighted least squares via the choice $w \equiv 1$.

19 Orthogonal polynomials (§8.2)

- For short we will use the shorthand

$$\langle\,\cdot\,,\,\cdot\,\rangle_w = \langle\,\cdot\,,\,\cdot\,\rangle_{L^2([a,b];w)}$$

- One motivation for the theory of orthogonal polynomials: find $g_0, \ldots, g_{p-1}$ such that the Gram matrix is ***diagonal***, i.e.,

$$\langle g_i, g_j \rangle_w = 0 \text{ if } i \neq j.$$

- We also insist that these functions are polynomials of increasing order up to n, and henceforth in this discussion we will use the notation $p_0, \ldots, p_n$ in place of $g_0, \ldots, g_{p-1}$. **[*The book notation for the orthogonal polynomials is* $\phi_0, \ldots, \phi_n$.]**

- A sequence of polynomials $p_0, p_1, \ldots$ such that p_k has order k and such that

$$\langle p_i, p_j \rangle_w = 0 \text{ if } i \neq j.$$

is called an **orthogonal polynomial** sequence with respect to the weight function w.

  – We would have **orthonormal** polynomials if $\langle p_i, p_i \rangle_w = 1$ for all i.

  – Often we do not choose this normalization! Instead we often ensure that each p_k is a **monic** polynomial, meaning that its leading order term has coefficient 1, i.e., $p_k(x) = x^k + \ldots$

- How are orthogonal polynomial sequences constructed?

- Consider the **monomial sequence** $M_k(x) = x^k$, $k = 0, 1, 2, \ldots$.
- A monic orthogonal polynomial sequence with respect to the weight function by applying Gram-Schmidt with the weighted L^2 inner product $\langle \cdot, \cdot \rangle_w$.
- To wit, start with $p_0 = M_0$. Then inductively, given $p_0, \ldots, p_k$, define p_{k+1} via

$$p_{k+1} = M_{k+1} - \sum_{i=0}^{k} \frac{\langle M_{k+1}, p_i \rangle_w}{\langle p_i, p_i \rangle_w} p_i.$$

- By construction, the sequence $p_0, p_1, \ldots$ is **orthogonal** with respect to $\langle \cdot, \cdot \rangle_w$ and in fact **monic**.
- Moreover, since applying Gram-Schmidt to the sequence $M_0, M_1, \ldots$ preserves the span at every stage, we moreover have

$$\operatorname{span}(p_0, \ldots, p_k) = \operatorname{span}(M_0, \ldots, M_k)$$

  for every k.
- Notice that $\operatorname{span}(M_0, \ldots, M_k) = \mathbb{P}_k = \Pi_k$, the **set of all polynomials of degree at most** k, where Π_k ***is the book notation***.
- Therefore $\operatorname{span}(p_0, \ldots, p_k) = \mathbb{P}_k$ as well.
- **HW:** orthogonality implies linear independence.
  - Therefore in fact $p_0, \ldots, p_k$ form a **basis** for $\mathbb{P}_k$.
- Note that if $l < k$, since p_k is orthogonal to $p_0, \ldots, p_l$, it is also orthogonal to everything in $\mathbb{P}_l = \operatorname{span}(p_0, \ldots, p_l)$. Therefore in fact p_k is orthogonal to all polynomials of degree less than k...!

- **Theorem:** Let $\{M_k\}_{k=0}^{\infty}$ denote the monomial sequence $M_k(x) = x^k$. The monic orthogonal polynomial sequence $p_0, p_1, \ldots$ with respect to the weight function w on the interval $[a, b]$ is produced by setting $p_0 = M_0$ and then inductively defining

$$p_{k+1} = M_{k+1} - \sum_{i=0}^{k} \frac{\langle M_{k+1}, p_i, \rangle_w}{\langle p_i, p_i \rangle_w} p_i.$$

  The p_k so produced are monic of degree k and orthogonal with respect to $\langle \cdot, \cdot \rangle_w$,. For every n, the collection $p_0, \ldots, p_n$ forms a basis for $\mathbb{P}_n$, the set of polynomials of degree at most k. For any $k > l$, p_k is orthogonal to all polynomials in $\mathbb{P}_l$..

20 Back to least squares (§8.2)

- A major advantage of orthogonal polynomial sequences it simplifies the weighted least squares approximation of f over an interval $[a, b]$ with weight function w.
- With $p_0, \ldots, p_n$ in place of $g_0, \ldots, g_{p-1}$ in our previous discussion, the Gram matrix A_{ij} is now diagonal by orthogonality:

$$A_{ij} = \begin{cases} \langle p_i, p_i \rangle_w, & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

- Therefore the normal equations $A\mathbf{a} = b$, where

$$b_i = \langle p_i, f \rangle_w$$

can be solved as

$$a_i = \frac{\langle f, p_i \rangle_w}{\langle p_i, p_i \rangle_w},$$

and we recover the optimal approximator

$$f_{\mathbf{a}} = \sum_{i=0}^{n} \frac{\langle f, p_i \rangle_w}{\langle p_i, p_i \rangle_w} p_i.$$

- Notice that

$$\{f_{\mathbf{a}} \,:\, \mathbf{a} \in \mathbb{R}^{n+1}\} = \left\{ \sum_{i=0}^{n} a_i p_i \,:\, \mathbf{a} \in \mathbb{R}^{n+1} \right\} = \operatorname{span}(p_0, \ldots, p_n) = \mathbb{P}_n,$$

so the optimal $f_{\mathbf{a}}$ is in fact precisely the same as the optimal polynomial of order $\leq n$. In summary:

- **Theorem:** The optimization problem

$$\min_{p \in \mathbb{P}_n} \|p - f\|_w^2$$

is solved by

$$p = \sum_{k=0}^{n} \frac{\langle f, p_k \rangle_w}{\langle p_k, p_k \rangle_w} p_k,$$

where $\{p_k\}_{k=0}^{\infty}$ is an orthogonal polynomial sequence with respect to the weight function w on $[a, b]$.

21 Computing with orthogonal polynomials (§8.2)

- How can we compute orthogonal polynomial sequences?
  - In principle it is often possible to carry out the Gram-Schmidt formula analytically.
  - In the **HW**, you will do this for the weight function $w \equiv 1$ on an interval.

- In fact, many of the terms in the Gram-Schmidt formula are zero. The justification is beyond the scope of the course (though see supplementary handout if interested), but the conclusion is that any orthogonal polynomial sequence satisfies a property called the **three-term recurrence**.

- **Theorem:** Let $\{p_k\}_{k=0}^{\infty}$ denote the orthogonal polynomial sequence with respect to the weight function w on $[a, b]$. Then the p_k satisfy the three-term recurrence

$$p_{k+1}(x) = (x - \alpha_k) p_k(x) - \beta_k p_{k-1}(x)$$

for $k \geq 1$, where

$$\alpha_k := \frac{\langle x p_k, p_k \rangle}{\langle p_k, p_k \rangle}, \quad \beta_k := \frac{\langle p_k, p_k \rangle}{\langle p_{k-1}, p_{k-1} \rangle} > 0.$$

Here xp_k is shorthand for the polynomial of degree $k+1$ which takes value $xp_k(x)$ at the point x.

- **Notation:** I will use p_k to denote general polynomial . But I will use capital letters for specific polynomial sequences. We have already defined M_k as the monomial sequence. We will also reserve P_k and T_k for the Legendre and Chebyshev polynomials
- **First example:** (monic) Legendre polynomials. The Legendre polynomials P_k are the orthogonal polynomials with respect to the weight function $w \equiv 1$ on the interval $[-1, 1]$. (*The book follows our same convention, but beware that Legendre polynomials are usually not chosen to be monic!*)
  - The Legendre polynomials satisfy the three-term recurrence

    $$P_{k+1}(x) = xP_k(x) - \frac{k^2}{4k^2 - 1} P_{k-1}(x)$$

    for $k \geq 1$.
  - We can use the three-term recurrence to easily construct the first few Legendre polynomials.
    * $P_0(x) = 1$ as always.
    * $P_1(x) = x$ can be constructed by Gram-Schmidt, but also we can easily verify orthogonality as $\int_{-1}^{1} P_0(x)P_1(x)\,dx = 0$, and P_1 as defined here is monic, so it is the right choice!
    * Now plugging into the three-term recurrence for $k = 1$, we have

      $$\begin{aligned} P_2(x) &= xP_1(x) - \frac{1}{3}P_0(x). \\ &= x^2 - \frac{1}{3}. \end{aligned}$$

    * Similarly,

      $$\begin{aligned} P_3(x) &= xP_2(x) - \frac{4}{15}P_1(x) \\ &= x\left(x^2 - \frac{1}{3}\right) - \frac{4}{15}x \\ &= x^3 - \left(\frac{1}{3} + \frac{4}{15}\right)x \\ &= x^3 - \frac{3}{5}x. \end{aligned}$$

    * And

      $$\begin{aligned} P_4(x) &= xP_3(x) - \frac{9}{35}P_2(x) \\ &= x\left[x^3 - \frac{3}{5}x\right] - \frac{9}{35}\left[x^2 - \frac{1}{3}\right] \\ &= x^4 - \frac{3}{5}x^2 - \frac{9}{35}x^2 + \frac{9}{35}\cdot\frac{1}{3} \\ &= x^4 - \left(\frac{3}{5} + \frac{9}{35}\right)x^2 + \frac{3}{35} \\ &= x^4 - \frac{6}{7}x^2 + \frac{3}{35}. \end{aligned}$$

- **How to evaluate orthogonal polynomials?**

- Counterintuitive!
- In principle we could carry out the three-term recurrence to get a formula

$$p_n(x) = a_n x^n + a_{n-1} x^{k-1} + \cdots + a_1 x + a_0$$

  for the n-th orthogonal polynomial. Then we could use this formula to evaluate.
- However, this is ***not*** how one should proceed. The reason is that the coefficients $a_0, \ldots, a_n$ can become ***extremely large*** as n becomes large, resulting in numerical overflow.
- In fact, the numerically stable way to evaluate $p_n(x)$ is using the three-term recurrence!
- If the recurrence coefficients α_k and β_k are known simply use the recurrence relation

$$p_{k+1}(x) = (x - \alpha_k) p_k(x) - \beta_k p_{k-1}(x)$$

  to evaluate build the sequence $p_0(x), \ldots, p_n(x)$ from scratch.

22 Chebyshev polynomials (§8.3)

- The ***Chebyshev polynomials*** are an extremely useful gadget in numerical analysis, due to their ***equioscillation property*** on the interval $[-1, 1]$, which can be transferred to arbitrary intervals via apprropriate shifting and scaling.
- They are orthogonal polynomials with respect to a suitable weight function on $[-1, 1]$, but we will construct them a different way and then see later that they are orthogonal polynomials!
- The n-th Chebyshev polynomial is defined for $x \in [-1, 1]$ by

$$T_n(x) = \cos(n \cos^{-1}(x)).$$

- It is not immediately obvious from the definition that T_n is actually a polynomial! However, this fact is guaranteed by the three-term recurrence

$$T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x), \quad (\star)$$

  together with the obvious identities $T_0 \equiv 1$ and $T_1(t) = x$.
  - To see $(\star)$, we use the classic trigonometric identity

$$\cos(a + b) = \cos(a)\cos(b) - \sin(a)\sin(b).$$

  - Substituting $-b$ in for b and adding identities yields another identity

$$\cos(a + b) + \cos(a - b) = 2\cos(a)\cos(b),$$

    or

$$\cos(a + b) = 2\cos(a)\cos(b) - \cos(a - b).$$

  - We can then compute

$$\begin{aligned} T_{n+1}(x) &= \cos(n \cos^{-1}(x) + \cos^{-1}(x)) \\ &= 2\cos(n \cos^{-1}(x)) \cos(\cos^{-1}(x)) - \cos((n-1)\cos^{-1}(x)) \\ &= 2x T_n(x) - T_{n-1}(x), \end{aligned}$$

    verifying $(\star)$.

- Note that the zeros of T_n (called the ***Chebyshev nodes***) are therefore

$$\cos\left(\frac{2j-1}{2n}\pi\right), \quad j = 1, \ldots, n,$$

and moreover

$$|T_n(t)| \le 1, \quad t \in [-1,1].$$

- Moreover, the local extrema of T_n, attained at

$$\cos\left(\frac{j\pi}{n}\right), \; j = 1, \ldots, n-1$$

are all ± 1 (as are the values at the interval endpoints -1 and 1). This is the **equioscillation property**.

- We believe now that the Chebyshev polynomials are *polynomials*, but why are they ***orthogonal polynomials***?
  - **Preliminary observation:** the collection $f_n(y) = \cos(ny)$ are orthogonal with respect to the unweighted L^2 inner product on $[0,\pi]$, i.e., $\int_0^\pi \cos(my)\cos(ny)\,dx = 0$ whenever $m \neq n$
    - To show this, recall our earlier identity: $\cos(a+b) + \cos(a-b) = 2\cos(a)\cos(b)$
    - Therefore, when $m \neq n$, we can compute:

$$\begin{aligned}\int_0^\pi \cos(my)\cos(ny)\,dy &= \frac{1}{2}\int_0^\pi \cos((m+n)y)\,dy + \frac{1}{2}\int_0^\pi \cos((m-n)y)\,dy \\ &= \frac{1}{2}\frac{1}{m+n}\sin((m+n)y)|_{y=0}^{\pi} + \frac{1}{2}\frac{1}{m-n}\sin((m-n)y)|_{y=0}^{\pi} \\ &= \frac{1}{2(m+n)}\sin((m+n)\pi) + \frac{1}{2(m-n)}\sin((m-n)\pi) \\ &= 0,\end{aligned}$$

    where we have used the facts that $\sin(0) = 0$ and then more generally $\sin(k\pi) = 0$ for all integers k.
  - Therefore if we change variables to x where $y = \cos^{-1}(x)$, which changes the domain of integration from $[0,\pi]$ to $[-1,1]$, we obtain

$$\begin{aligned}0 &= \int_0^\pi f_m(y)f_n(y)\,dy \\ &= \int_{-1}^1 f_m(\cos^{-1}(x))f_n(\cos^{-1}(x))\left|\frac{d}{dx}[\cos^{-1}(x)]\right|\,dx \\ &= \int_{-1}^1 T_m(x)T_n(x)\frac{1}{\sqrt{1-x^2}}\,dx \\ &= \langle T_m, T_n\rangle_{L^2([-1,1];w)},\end{aligned}$$

    where $w(x) := \frac{1}{\sqrt{1-x^2}}$.
  - Therefore the **Chebyshev polynomials are orthogonal polynomials on** $[-1,1]$ **with respect to the weight function** $w(x) = \frac{1}{\sqrt{1-x^2}}$**.**

- Notice that the Chebyshev polynomials as we have defined them are **not monic**!

- Recall the 3-term recurrence: $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$, and the starter polynomials $T_0(x) = 1$, $T_1(x) = x$
- The recurrence implies that the leading order term of the next polynomials are $2x^2$, $4x^3$, $8x^4$, etc.
- In general, the leading-order term of T_n is $2^{n-1}x^n$
- We can define the **monic Chebyshev polynomials** by normalizing as $\tilde{T}_n(x) := \frac{1}{2^{n-1}}T_n(x)$ for $n \geq 1$ and $\tilde{T}_0(x) = 1$.

- If we are interested in approximating functions on a more general interval $[a, b]$ (not necessarily $[-1, 1]$), it is useful to consider the change of variables

$$\tilde{x} = \frac{1}{2}\left[(b-a)x + a + b\right],$$

which sends -1 to a and 1 to b. The inverse transformation is

$$x = \frac{2\tilde{x} - (a+b)}{b-a},$$

which sends a to -1 and b to 1.

  - This allows us, for example, to transport the Chebyshev nodes to an arbitrary interval.
  - One can also define suitable orthogonal polynomials $p_n(\tilde{x}) = T_n\left(\frac{2\tilde{x}-(a+b)}{b-a}\right)$ on the interval $[a, b]$ with the appropriate weight function $w(\tilde{x}) = w\left(\frac{2\tilde{x}-(a+b)}{b-a}\right)$.
  - Similar comments apply to the Legendre polynomials.
  - In our discussions, for simplicity we will just focus on $[-1, 1]$.

23 Economization of power series (§8.3)

- Let $\tilde{\mathbb{P}}_n$ denote the set of all monic polynomials of degree n.
- **Theorem:** For all $\tilde{p}_n \in \tilde{\mathbb{P}}_n$,

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\tilde{T}_n(x)| \leq \max_{x\in[-1,1]} |\tilde{p}_n(x)|.$$

  - In this sense, the (monic) Chebyshev polynomials take the smallest possible extreme values on $[-1, 1]$ among all other monic polynomials.
  - The proof is omitted but can be found in the textbook.

- We can apply this result in the folowing setting: suppose that we are given a polynomial $p_n(x) = \sum_{k=0}^n a_k x^k$, but we are not happy with it because the degree n is too large. We want to obtain a lower-degree polynomial $p_m \in \mathbb{P}_m$, where $m < n$, which approximates the given polynomial of high degree on the interval $[-1, 1]$.
  - The high degree polynomial might be obtained, for example, from a Taylor series truncated at very high order.
- We will try to obtain p_m *greedily*:
  - First obtain $p_{n-1} \in \mathbb{P}_{n-1}$ that approximates p_n well.

  - Then obtain $p_{n-2} \in \mathbb{P}_{n-2}$ that approximates p_{n-1} well, etc., etc., until we reach p_k.
  - Therefore we can focus simply on the first step.

- **Criterion:** choose $p_{n-1} \in \mathbb{P}_{n-1}$ that minimizes the quantity

$$\max_{x\in[-1,1]} |p_n(x) - p_{n-1}(x)|.$$

(This is called a 'minimax' optimization problem.)

  - Assume that $a_n \neq 0$. (Otherwise we can simply take $p_{n-1} = p_n$, and then proceed to the next step.)
  - Note then that for all $p_{n-1} \in \mathbb{P}_{n-1}$, the polynomial $(p_n - p_{n-1})/a_n$ is monic of degree n, i.e., lies in $\tilde{\mathbb{P}}_n$.
  - Then it is equivalent to minimize

$$|a_n| \max_{x\in[-1,1]} \left| \frac{p_n(x) - p_{n-1}(x)}{a_n} \right|$$

over all $p_{n-1} \in \mathbb{P}_{n-1}$.

  - But the theorem says that the minimum value for the maximum is attained precisely when

$$\frac{p_n(x) - p_{n-1}(x)}{a_n} = \tilde{T}_n(x),$$

and solving for p_{n-1} yields the formula

$$p_{n-1}(x) = p_n(x) - a_n\tilde{T}_n(x)$$

for the maximizing polynomial $p_{n-1} \in \mathbb{P}_{n-1}$.

  - Therefore in summary, we simply take

$$p_{n-1}(x) = p_n(x) - a_n\tilde{T}_n(x),$$

and note that this formula even works in the $a_n = 0$ case, so we realize that we can apply it without worrying about cases.

24 Lagrange interpolation meets Chebyshev polynomials (§8.3)

- First we will review the topic of Lagrange interpolation from Math 128A.
- The idea of Lagrange interpolation is motivated by the following problem. Suppose we are given points $x_0, \ldots, x_m \in \mathbb{R}$ (which are distinct) and values $y_1, \ldots, y_m$. The **Lagrange interpolating polynomial** for the data (x_i, y_i), $i = 0, \ldots, m$, is the **unique** polynomial $p \in \mathbb{P}_m$ such that $p(x_i) = y_i$ for all $i = 0, \ldots, m$.
- Given a function f and interpolation points x_i, by evaluating $y_i = f(x_i)$ at the interpolation points, we can construct the Lagrange interpolating polynomial p for the data $(x_i, y_i), i = 0, \ldots, m$.
  - **Question:** does p approximate f on some interval, as we take $m \to \infty$?
  - **Answer:** depends dramatically on how we choose the interpolation points x_i!

    * If we choose wrong (e.g., equispaced points), the approximation can ***diverge***! (This is called Runge's phenomenon.)
    * If we choose right (***Chebyshev nodes***, as we shall see), the approximation converges.

- Consider the ***Lagrange basis polynomials*** subordinate to the choice of interpolation points $\mathbf{x} = (x_0, \ldots, x_m)$
$$L_j(x;\mathbf{x}) := \prod_{i\in\{0,\ldots,m\}\setminus\{j\}} \frac{x - x_i}{x_j - x_i}, \ j = 0, \ldots, m.$$
  - Note that the Lagrange basis polynomials depend on the collection $\mathbf{x}$ of interpolation points.
  - For simplicity of notation, we will omit the dependence on $\mathbf{x}$ when the context is clear, i.e., write $L_j(x;\mathbf{x}) = L_j(x)$.
  - **Key properties:** $L_j \in \mathbb{P}_m$, and
$$L_j(x_i) = \delta_{ij},$$
    for all i, j.

- Therefore if we define $p \in \mathbb{P}_m$ via
$$p = \sum_{j=0}^{m} y_j L_j,$$
then by construction $p(x_i) = \sum_{j=0}^{m} y_j \delta_{ij} = y_i$, achieving the desired interpolating property.
  - In fact, the p so defined is the **unique** choice in $\mathbb{P}_m$ achieving the interpolating property, called the Lagrange interpolating polynomial.

- The following theorem characterizes the interpolation error. It is useful first to define the norm
$$\|g\|_{L^\infty([a,b])} = \max_{x\in[a,b]} |g(x)|.$$

- **Theorem:** Suppose $f \in C^{m+1}([a,b])$, and let $x_0, \ldots, x_m \in [a,b]$ be distinct interpolation points. Let $y_i = f(x_i)$ for $i = 0, \ldots, m$, and let p denote the Lagrange interpolating polynomial for the data (x_i, y_i), $i = 0, \ldots, m$. Then for every $x \in [a,b]$ there exists $\xi = \xi(x) \in [a,b]$ such that
$$f(x) - p(x) = \frac{f^{(m+1)}(\xi)}{(m+1)!} \prod_{i=0}^{m} (x - x_i).$$
Hence
$$\|f - p\|_{L^\infty([a,b])} \leq \frac{\|f^{(m+1)}\|_{L^\infty([a,b])}}{(m+1)!} \|q_{\mathbf{x}}\|_{L^\infty([a,b])},$$
where $q_{\mathbf{x}}(x) := \prod_{i=0}^{m} (x - x_i)$.
  - It follows that if we fix the function f and the number $m+1$ of interpolation points, but change the interval length $b - a$, then $\|f - p\|_{L^\infty([a,b])} = O(|b-a|^{m+1})$. This is true because $\|q_{\mathbf{x}}\|_{L^\infty([a,b])} \leq |b-a|^{m+1}$.
  - However, if we fix the function f and the interval $[a,b]$ but change the number of interpolation points, ***anything could happen***, depending on the choice of interpolation points.

- What if we the interval is $[-1,1]$ and we choose the Chebyshev nodes as interpolation points?

  - Other intervals can be handled similarly by linearly transforming the Chebyshev nodes as described above.

- Note that we have $m+1$ interpolation points, so the appropriate Chebyshev nodes are the zeros of T_{m+1}.
  - Recall the zeros of T_n are

$$\cos\left(\frac{2j-1}{2n}\pi\right), \quad j = 1, \dots, n.$$

  - Hence plugging in $n = m+1$, we obtain points $\mathbf{x} = (x_0, \dots, x_m)$ given by

$$x_j = \cos\left(\frac{2j+1}{2(m+1)}\pi\right), \quad j = 0, \dots, m.$$

- Note that with this choice, in fact (!) $q_{\mathbf{x}} = \tilde{T}_{m+1}$.
  - This follows from the fundamental theorem of algebra: since $\tilde{T}_{m+1}$ has zeros $x_0, \dots, x_m$, we must have $\tilde{T}_{m+1}(x) = C\prod_{i=0}^{m}(x - x_i) = Cq_{\mathbf{x}}(x)$ for some C. But since $q_{\mathbf{x}}$ and $\tilde{T}_{m+1}$ are both monic, we must have $C = 1$.

- Moreover $|\tilde{T}_{m+1}(x)| \leq \frac{1}{2^m}$ for $x \in [-1,1]$, hence $\|q_{\mathbf{x}}\|_{L^\infty([a,b])} \leq \frac{1}{2^m}$, and we have:

- **Theorem:** Suppose $f \in C^{m+1}([-1,1])$, and let $\mathbf{x} = (x_0, \dots, x_m)$ where

$$x_j = \cos\left(\frac{2j+1}{2(m+1)}\pi\right), \quad j = 0, \dots, m.$$

  Let $y_i = f(x_i)$ for $i = 0, \dots, m$, and let p denote the Lagrange interpolating polynomial for the data (x_i, y_i), $i = 0, \dots, m$. Then

$$\|f - p\|_{L^\infty([-1,1])} \leq \frac{\|f^{(m+1)}\|_{L^\infty([-1,1])}}{2^m(m+1)!}.$$

  - **Extension:** If we use appropriately transformed Chebyshev nodes $\tilde{x}_j$ for the interval $[a,b]$ as our interpolation points, then we obtain

$$\|f - p\|_{L^\infty([a,b])} \leq \left(\frac{b-a}{2}\right)^{m+1} \frac{\|f^{(m+1)}\|_{L^\infty([a,b])}}{2^m(m+1)!}.$$

25 Gauss quadrature for approximation theory

- Given an interval $[a,b]$, possibly with $a = -\infty$, $b = +\infty$ and a weight function $w : [a,b] \to [0,\infty)$, ***Gauss quadrature*** concerns the estimation of integrals of the form

$$\int_a^b g(x)\,w(x)\,dx$$

  with quadrature rules of the form

$$\sum_{i=0}^{m} w_i g(x_i).$$

  - The points x_i are called the ***nodes*** and the values w_i are called the ***weights*** of the Gauss quadrature rule.

- We will ask for an integration rule that is exact when g is a polynomial, up to whatever order we can achieve.
  - Gauss-Legendre quadrature is induced by the choice $w \equiv 1$ on the interval $[-1, 1]$, and Gauss-Chebyshev quadrature is induced by the choice $w(x) = \frac{1}{\sqrt{1-x^2}}$ on the interval $[-1, 1]$.
  - Note that if the interval is infinite or semi-infinite, we must demand that the weight function $w(x)$ has sufficient decay. For example, the choice $w(x) = e^{-x^2/2}$ for the interval $(-\infty, \infty)$ will induce the so-called Gauss-Hermite quadrature.

- **Theorem:** Let $a < x_0 < \ldots < x_m < b$ be the zeros of p_m, the $(m+1)$-th orthogonal polynomial with respect to the weight function w on the interval $[a, b]$, and let $\mathbf{x} = (x_0, \ldots, x_m)$. Define weights

$$w_i := \int_a^b L_i(x; \mathbf{x})\, w(x)\, dx$$

for $i = 0, \ldots, m$. Then

$$\int_a^b g(x)w(t)\, dt = \sum_{i=0}^{m} w_i g(x_i)$$

for all $g \in \mathbb{P}_{2m+1}$.
  - Given arbitrary nodes, can derive weights by realizing that any $g \in \mathbb{P}_m$ is equal to its own interpolating polynomial on the interpolation points $x_0, \ldots, x_m$.
  - The fact that the formula is exact on all of $\mathbb{P}_{2m+1}$ is a more subtle point and is only true for the choice of nodes derived from orthogonal polynomials!

- Recall our earlier theorem on weighted least squares:

- **Theorem:** The optimization problem

$$\min_{p \in \mathbb{P}_n} \|p - f\|_w^2$$

is solved by

$$p = \sum_{k=0}^{n} \frac{\langle f, p_k \rangle_w}{\langle p_k, p_k \rangle_w} p_k,$$

where $\{p_k\}_{k=0}^{\infty}$ is an orthogonal polynomial sequence with respect to the weight function w on $[a, b]$.

- To implement this theorem, we need to compute integrals of the form

$$\langle h_1, h_2 \rangle_w = \int_a^b h_1(x)h_2(x)\, w(x)\, dx.$$

  - Can view these as integrals of the form $\int_a^b g(x)\, w(x)\, dx$, where $g = h_1 h_2$.
  - Hence apply Gauss quadrature:

$$\langle h_1, h_2 \rangle_w \approx \sum_{i=0}^{m} w_i h_1(x_i) h_2(x_i).$$

26 Connecting interpolation and least-squares projection

- Recall our formula for $p \in \mathbb{P}_n$ which approximates f optimally in the least-squares sense (weighted by w on $[a, b]$):
$$p = \sum_{k=0}^{n} \frac{\langle f, p_k \rangle_w}{\langle p_k, p_k \rangle_w} p_k$$
- Consider the 'approximate inner product' $\langle \cdot\,, \cdot \rangle_w^{(m)}$ defined by
$$\langle \phi, \psi \rangle_w^{(m)} := \sum_{i=0}^{m} \phi(x_i)\psi(x_i)w_i,$$
where w_i and x_i, $i = 0, \ldots, m$, are the Gauss quadrature weights and nodes for w, as defined in the last section
- In the last section we suggested approximating the least-squares-optimal p by
$$q := \sum_{k=0}^{n} \frac{\langle f, p_k \rangle_w^{(m)}}{\langle p_k, p_k \rangle_w^{(m)}} p_k$$
where we take the m to be 'sufficiently large' (possibly $m \gg n$)
- Amazingly, if we pick $m = n$, then q coincides ***exactly*** with the Lagrange interpolating polynomial for f using interpolation points x_i, $i = 0, \ldots, m$
- Therefore we can perform Lagrange interpolation ***exactly*** by a kind of 'lazy' evaluation of Gauss quadrature
  - This allows us to express the Lagrange interpolating polynomial as a linear combination of orthogonal polynomials, which is otherwise annoying to do (since we have otherwise only constructed it as a linear combination of Lagrange basis polynomials)
  - Very useful, especially in Chebyshev case!
- **Theorem:** With notation as in the preceding discussion, let $q = \sum_{k=0}^{n} \frac{\langle f, p_k \rangle_w^{(n)}}{\langle p_k, p_k \rangle_w^{(n)}} p_k \in \mathbb{P}_n$. Then q coincides exactly with the Lagrange interpolating polynomial for f using interpolation points x_i, $i = 0, \ldots, n$. That is, $q(x_i) = f(x_i)$ for all $i = 0, \ldots, n$.
- **Proof:**
  - Let ℓ be the Lagrange interpolating polynomial. **WTS:** $q = \ell$, where q is defined as above in the statement.
  - **Claim (1):** $\langle p_k, p_k \rangle_w^{(n)} = \langle p_k, p_k \rangle_w$ For all $k = 0, \ldots, n$
    * To see this note that $\langle p_k, p_k \rangle_w = \int_a^b p_k^2(x)\, w(x)\, dx$
    * Since $p_k^2 \in \mathbb{P}_{2n}$, the Gauss quadrature rule is exact, i.e.,
$$\int_a^b p_k^2(x)\, w(x)\, dx = \sum_{i=0}^{n} p_k^2(x_i)\, w_i\, dx = \langle p_k, p_k \rangle_w^{(n)},$$
    which proves the claim.
  - **Claim (2):** $\langle f, p_k \rangle_w^{(n)} = \langle \ell, p_k \rangle_w^{(n)}$

    * We can verify by computing

$$\langle f, p_k \rangle_w^{(n)} = \sum_{i=0}^{n} f(x_i) p_k(x_i) w_i = \sum_{i=0}^{n} \ell(x_i) p_k(x_i) w_i = \langle \ell, p_k \rangle_w^{(n)},$$

      where the equality in the middle follows from the fact that the interpolating polynomial must satisfy $\ell(x_i) = f(x_i)$.

  - **Claim (3):** $\langle \ell, p_k \rangle_w^{(n)} = \langle \ell, p_k \rangle_w$
    * This claim is proved in the same way as Claim (1), noting that $\ell p_k \in \mathbb{P}_{2n}$.
  - Putting the three claims together and substituting into the formula for q, we have that

$$q = \sum_{k=0}^{n} \frac{\langle \ell, p_k \rangle_w}{\langle p_k, p_k \rangle_w} p_k$$

  - But the right-hand side of the last equation is precisely the solution of the problem

$$\min_{\tilde{q} \in \mathbb{P}_n} \|\tilde{q} - \ell\|_w^2$$

  - This problem trivially admits the unique solution $\tilde{q} = \ell$. Therefore $q = \ell$, as was to be shown.

27 Padé approximation (§8.4)

- **Definition:** a ***rational function*** r of degree N is a function of the form

$$r(x) = \frac{p(x)}{q(x)},$$

  where p and q are polynomials whose degrees sum to N.

- We are now interested in approximating a given function f not just by polynomials, but by rational functions.

- Padé approximation (about $x = 0$).
  - Let us expand $p(x) = \sum_{k=0}^{n} p_k x^k$ and $q(x) = \sum_{k=0}^{m} q_k x^k$.
  - For simplicity later on, extend the sequence $p_0, \ldots, p_n$ and $q_0, \ldots, q_m$ by zeros, so that $p_k = 0$ for $k > n$ and $q_k = 0$ for $k > m$.
    * Hence we can view $p(x) = \sum_{k=0}^{\infty} p_k x^k$ and $q(x) = \sum_{k=0}^{\infty} q_k x^k$.
  - Without loss of generality we can assume $q_0 = 1$, leaving $n+m+1 = N+1$ free parameters, $p_0, \ldots, p_n, q_1, \ldots, q_m$.
  - The $[n/m]$ ***Padé approximant*** of a function f is defined by choosing these free parameters such that $r^{(k)}(0) = f^{(k)}(0)$ for $k = 0, \ldots, N$.
    * This matches as many derivatives at the origin as possible since we are solving $N+1$ equations in $N+1$ unknowns.

- How to construct the Padé approximant?

- Let $f(x) = \sum_{k=0}^{\infty} a_k x^k$ denote a Maclaurin series expansion for f.
- Then

$$\begin{aligned} f(x) - r(x) &= f(x) - \frac{p(x)}{q(x)} \\ &= \frac{f(x)q(x) - p(x)}{q(x)}. \end{aligned}$$

- In order to have $[f-r]^{(k)}(0) = 0$ for $k = 0, \ldots, N$, in fact it is equivalent for the first k derivatives of the numerator to all be zero. (Exercise in lecture: check.)
- This is equivalent to all terms up to order N in the series expansion for the numerator being zero.
-

$$\begin{aligned} f(x)q(x) - p(x) &= \left(\sum_{i=0}^{\infty} a_i x^i\right)\left(\sum_{j=0}^{\infty} q_j x^j\right) - \sum_{k=0}^{\infty} p_k x^k \\ &= \sum_{i,j=0}^{\infty} a_i q_j x^{i+j} - \sum_{k=0}^{\infty} p_k x^k \\ &= \sum_{k=0}^{\infty}\sum_{i=0}^{k} a_i q_{k-i} x^k - \sum_{k=0}^{n} p_k x^k \\ &= \sum_{k=0}^{\infty}\left(\sum_{i=0}^{k} a_i q_{k-i} - p_k\right) x^k. \end{aligned}$$

- In order for the terms of order up to N to drop out, we require

$$p_k = \sum_{i=0}^{k} a_i q_{k-i}, \quad \text{for all } k = 0, 1, \ldots, N.$$

- Observe that this is a system of $N+1$ linear equations in the $N+1$ unknowns $p_0, \ldots, p_n, q_1, \ldots, q_m$.

• How to rephrase the linear system in the standard format $Ax = b$?

- It is equivalent to rephrase the k-th equation as

$$p_k - \sum_{i=0}^{k} a_{k-i} q_i = 0,$$

which is equivalent to

$$p_k - \sum_{i=1}^{k} a_{k-i} q_i = a_k.$$

- Then define an $(N+1) \times m$ matrix $C = (C_{ki})$ where the row index is 'zero-indexed' $0, \ldots, N$ and the column index is 'one-indexed' $1, \ldots, m$, and

$$C_{ki} = \begin{cases} a_{k-i}, & i \le k, \\ 0, & \text{otherwise,} \end{cases}$$

so

$$p_k - \sum_{i=1}^{m} C_{ki} q_i = a_k.$$

* As a small example, consider $n = 3$, $m = 3$, hence $N = 6$:

$$C = \begin{pmatrix} 0 & 0 & 0 \\ a_0 & 0 & 0 \\ a_1 & a_0 & 0 \\ a_2 & a_1 & a_0 \\ a_3 & a_2 & a_1 \\ a_4 & a_3 & a_2 \\ a_5 & a_4 & a_3 \end{pmatrix}.$$

– Meanwhile $p_k = \sum_{ki} E_{ki} p_i$, where $E = (E_{ki})$ is the $(N+1) \times (n+1)$ matrix where both indices are zero-indexed and

$$E_{ki} = \begin{cases} 1, & i = k \leq n \\ 0, & \text{otherwise,} \end{cases}$$

so

$$\sum_{i=0}^{n} E_{ki} p_i - \sum_{i=1}^{n} C_{ki} q_i = a_k, \quad k = 0, \ldots, N.$$

* In the same case $m = 3$, $n = 3$, $N = 6$, we have:

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

* In general we can write

$$E = \begin{pmatrix} I_{n+1} \\ 0_{(N-n)\times(n+1)} \end{pmatrix}.$$

– Then if we let $\mathbf{a} = (a_0, \ldots, a_N)^\top$, $\mathbf{p} = (p_0, \ldots, p_n)^\top$, and $\mathbf{q} = (q_1, \ldots, q_m)^\top$, we can vectorize our equations as

$$E\mathbf{p} - C\mathbf{q} = \mathbf{a},$$

or

$$\underbrace{\begin{pmatrix} E & -C \end{pmatrix}}_{=:A} \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix} = \mathbf{a},$$

which is a linear system $Ax = \mathbf{a}$ in matrix-vector format that can be solved for $x = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix}$.

28 Chebyshev rational approximation (§8.4)

- The Padé approximant, like Taylor polynomials, is based on an expansion around a base point (which we took to be $x = 0$), leading to approximations that are of nonuniform quality over an interval such as $[-1, 1]$.
- Assume we want to construct a rational approximation for f of roughly uniform quality over $[-1, 1]$ (and we can reduce to this case from the general case $[a, b]$ by the appropriate linear transformation as before).

- First step, replace Taylor series in numerator and denominator with Chebyshev expansions:

$$r(x) = \frac{\sum_{k=0}^{n} p_k T_k(x)}{\sum_{k=0}^{m} q_k T_k(x)},$$

  where again we can assume WLOG that $q_0 = 1$.

  - Again we can extend p_k and q_k by zeros for $k > n$ and $k > m$, respectively, to obtain $r(x) = \frac{\sum_{k=0}^{\infty} p_k T_k(x)}{\sum_{k=0}^{\infty} q_k T_k(x)}$
  - In fact we will also extend q_k by zeros for $k < 0$

- Suppose that

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x),$$

  where the coefficients are defined

$$a_k = \frac{\langle f, T_k \rangle_w}{\langle T_k, T_k \rangle_w}$$

  and w is the Chebyshev weight function $w(x) = \frac{1}{\sqrt{1-x^2}}$.

  - We shall only need to compute $a_0, \ldots, a_{N+m}$.
  - The numerators $\langle f, T_k \rangle_w$ can be obtained in general by Gauss-Chebyshev quadrature.
  - The denominators are problem-independent and can be computed directly to be

$$\langle T_k, T_k \rangle_w = \begin{cases} \pi, & k = 0 \\ \pi/2, & \text{otherwise.} \end{cases}$$

- **Idea:** again expand

$$f(x)q(x) - p(x)$$

  but now in the Chebyshev polynomials $T_0, T_1, \ldots$, and insist that all terms up to order N vanish.

  - Compute

$$\begin{aligned} f(x)q(x) - p(x) &= \left(\sum_{i=0}^{\infty} a_i T_i(x) \right) \left(\sum_{j=0}^{\infty} q_j T_j(x) \right) - \sum_{k=0}^{\infty} p_k T_k(x) \\ &= \sum_{i,j=0}^{\infty} a_i q_j T_i(x) T_j(x) - \sum_{k=0}^{\infty} p_k T_k(x). \end{aligned}$$

  - We have reached an obstacle: before we could write $x^i x^j = x^{i+j}$ as another element of the monomial basis, but $T_i T_j$ is itself not a Chebyshev polynomial.
  - However, we do have the ***important*** identity (from the homework)

$$T_i T_j = \frac{1}{2} \left[T_{i+j} + T_{|i-j|} \right],$$

    which is still good enough.

– Continue computing:

$$\begin{aligned}
fq-p &= \frac{1}{2}\sum_{i,j=0}^{\infty} a_i q_j T_{i+j} + \frac{1}{2}\sum_{i,j=0}^{\infty} a_i q_j T_{|i-j|} - \sum_{k=0}^{\infty} p_k T_k \\
&= \frac{1}{2}\sum_{k=0}^{\infty}\sum_{i=0}^{k} a_i q_{k-i} T_k + \frac{1}{2}\sum_{i=0}^{\infty} a_i q_i T_0 + \frac{1}{2}\sum_{k=1}^{\infty}\sum_{i=0}^{\infty} a_i\left(q_{i+k}+q_{i-k}\right) T_k - \sum_{k=0}^{\infty} p_k T_k \\
&= \frac{1}{2}\sum_{k=0}^{\infty}\left(\sum_{i=0}^{k} a_i q_{k-i}\right) T_k + \frac{1}{2}\left(\sum_{i=0}^{m} a_i q_i\right) T_0 + \frac{1}{2}\sum_{k=1}^{\infty}\left(\sum_{i=0}^{m-k} a_i q_{i+k}\right) T_k + \frac{1}{2}\sum_{k=1}^{\infty}\sum_{i=0}^{\infty} a_i q_{i-k} T_k - \sum_{k=0}^{\infty} p_k T_k \\
&= \frac{1}{2}\sum_{k=0}^{\infty}\left(\sum_{i=0}^{k} a_i q_{k-i}\right) T_k + \frac{1}{2}\sum_{k=0}^{\infty}\left(\sum_{i=0}^{m-k} a_i q_{i+k}\right) T_k + \frac{1}{2}\sum_{k=1}^{\infty}\sum_{i=k}^{m+k} a_i q_{i-k} T_k - \sum_{k=0}^{\infty} p_k T_k
\end{aligned}$$

– This yields the equations

$$\begin{aligned}
k=0: &\quad \sum_{i=0}^{k} a_i q_{k-i} + \sum_{i=0}^{m-k} a_i q_{i+k} - 2p_k \\
k\geq 1: &\quad \sum_{i=0}^{k} a_i q_{k-i} + \sum_{i=0}^{m-k} a_i q_{i+k} + \sum_{i=k}^{m+k} a_i q_{i-k} - 2p_k = 0
\end{aligned}$$

– These are equivalent to

$$\begin{aligned}
k=0: &\quad \sum_{i=0}^{k} a_{k-i} q_i + \sum_{i=k}^{m} a_{i-k} q_i - 2p_k = 0 \\
k\geq 1: &\quad \sum_{i=0}^{k} a_{k-i} q_i + \sum_{i=k}^{m} a_{i-k} q_i + \sum_{i=0}^{m} a_{i+k} q_i - 2p_k = 0
\end{aligned}$$

– This time we will ignore the constraint $q_0 = 1$ for now and treat q_0 as a variable ***for the time being***.

– Form the $(N+1)\times(m+1)$ matrices $B^{(1)} = (B^{(1)}_{ki})$, $B^{(2)} = (B^{(2)}_{ki})$ and $B^{(3)} = (B^{(3)}_{ki})$, with both rows and columns ***zero-indexed*** this time, defined by

$$B^{(1)}_{ki} = \begin{cases} a_{k-i}, & i \leq k, \\ 0, & \text{otherwise,} \end{cases} \qquad B^{(2)}_{ki} = \begin{cases} a_{i-k}, & i \geq k, \\ 0, & \text{otherwise,} \end{cases} \qquad B^{(3)}_{ki} = \begin{cases} a_{i+k}, & k \geq 1, \\ 0, & \text{otherwise,} \end{cases}$$

and let

$$\tilde{C} = \frac{1}{2}\left(B^{(1)} + B^{(2)} + B^{(3)}\right).$$

– Then letting $\tilde{\mathbf{q}} = (q_0, \ldots, q_m)^\top \in \mathbb{R}^{m+1}$ now includes q_0, we recover the equations

$$E\mathbf{p} - \tilde{C}\tilde{\mathbf{q}} = 0.$$

– Now we remember the constraint $q_0 = 1$ and let $\mathbf{c} \in \mathbb{R}^{N+1}$ and $C \in \mathbb{R}^{(N+1)\times m}$ be defined by the block equation

$$\tilde{C} = \begin{pmatrix} \mathbf{c} & C \end{pmatrix}, \quad \tilde{\mathbf{q}} = \begin{pmatrix} 1 \\ \mathbf{q} \end{pmatrix}$$

to obtain

$$E\mathbf{p} - C\mathbf{q} = \mathbf{c}.$$

– We may vectorize the equations again as

$$\underbrace{\begin{pmatrix} E & -C \end{pmatrix}}_{=:A} \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix} = \mathbf{c}$$

to obtain the $A\mathbf{x} = \mathbf{c}$ format that can be solved for $x = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix}$.

29 Fourier series (§8.5)

- **Caution:** *notation here is a bit different from book.*
- **Caution:** *for the discussion of Fourier stuff, we will reserve the letter i for the complex number ($i^2 = -1$).*
- We now consider functions $f : [-\pi, \pi] \to \mathbb{R}$ (by shifting and scaling we can reduce to this domain from an arbitrary interval).
  – It is useful to view the domain of f as extending to the entire real line $\mathbb{R}$ by stipulating that it is 2π-***periodic***, i.e., that $f(x + 2\pi k) = f(x)$ for all $k \in \mathbb{Z}$.
- Consider the functions
$$\phi_k(x) = \cos(kx), \quad k = 0, \ldots, n,$$
so in particular $\phi_0 \equiv 1$, and
$$\psi_k(x) = \sin(kx), \quad k = 0, \ldots, n.$$
- In fact the collection $\{\phi_0, \ldots, \phi_n, \psi_1, \ldots, \psi_n\}$ is *orthogonal* with respect to the $L^2([-\pi, \pi])$ inner product. (In this section we will fix the notation $\langle \cdot, \cdot \rangle$ for this inner product.)
  – This means $\langle \phi_k, \phi_l \rangle = 0$ whenever $i \neq j$, $\langle \psi_k, \psi_l \rangle$ whenever $k \neq l$, and $\langle \phi_k, \psi_l \rangle = 0$ for all k, l.
  – In fact also
$$\langle \phi_0, \phi_0 \rangle = 2\pi, \quad \langle \phi_k, \phi_k \rangle = \langle \psi_k, \psi_k \rangle = \pi \text{ for } k = 1, \ldots, n.$$
  – The verification is similar to our verification of orthogonality for Chebyshev polynomials, so we'll skip it.
- **Definition:** Set $\mathcal{T}_n := \operatorname{span}(\phi_0, \ldots, \phi_n, \psi_1, \ldots, \psi_n)$ is called the set of ***trigonometric polynomials*** of degree $\leq n$. (We say that a trigonometric polynomial in $\mathcal{T}_n$ has ***degree*** n if the coefficient of either ϕ_n or ψ_n is nonzero.)
  – **Caution:** For some reason, the book does not include ψ_n within $\mathcal{T}_n$, but that is not conventional and it would mess up some clarifying discussion below.
- Orthogonality ensures that the problem
$$\underset{\phi \in \mathcal{T}_n}{\text{minimize}} \ \|\phi - f\|^2$$
is solved by
$$\phi = \frac{\langle f, \phi_0 \rangle}{\langle \phi_0, \phi_0 \rangle} + \sum_{k=1}^{n} \left[\frac{\langle f, \phi_k \rangle}{\langle \phi_k, \phi_k \rangle} \phi_k + \frac{\langle f, \psi_k \rangle}{\langle \psi_k, \psi_k \rangle} \psi_k \right],$$

or concretely

$$\phi(x) = \frac{a_0}{2} + \sum_{k=1}^{n} \left[a_k \cos(kx) + b_k \sin(kx)\right],$$

where

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,\cos(kx)\,dx, \quad k = 0,\dots,n,$$
$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,\sin(kx)\,dx, \quad k = 1,\dots,n.$$

- Clarifying perspective is gained by extending the range of our functions to the *complex numbers...*
  - We can extend the L^2 inner product to functions $g, h : [-\pi,\pi] \to \mathbb{C}$ via the formula

    $$\langle g, h\rangle = \int_{-\pi}^{\pi} g(x)\overline{h(x)}\,dx.$$

  - **Note:** in general, inner product over a complex vector vector space is defined the same as an inner product over a real vector space, except that it is required to be linear in only the *first slot* and instead of symmetry we have conjugate symmetry

    $$\langle g, h\rangle = \overline{\langle h, g\rangle}.$$

    * From this, it follows that we have 'conjugate linearity' in the second slot:

      $$\langle g, h_1 + \lambda h_2\rangle = \langle g, h_1\rangle + \overline{\lambda}\,\langle g, h_2\rangle.$$

- In fact our basis $\phi_0,\dots,\phi_n,\psi_1,\dots,\psi_n$ for $\mathcal{T}_n$ is closely related to a much more elegant basis that involves complex values.
- Define $e_k : [-\pi,\pi] \to \mathbb{C}$ by

  $$e_k(x) = e^{ikx}$$

  for every $k \in \mathbb{Z}$
- In fact $\langle e_k, e_l\rangle = 2\pi\delta_{kl}$, so in particular the functions $\{e_{-n},\dots,e_n\}$ are orthogonal.
  - We can check this

    $$\begin{aligned}\langle e_k, e_l\rangle &= \int_{-\pi}^{\pi} e_k(x)\,\overline{e_l(x)}\,dx\\ &= \int_{-\pi}^{\pi} e^{ikx}e^{-ilx}\,dx\\ &= \int_{-\pi}^{\pi} e^{i(k-l)x}\,dx.\end{aligned}$$

    In the case $k = l$, $e^{i(k-l)x} = 1$ and the integral is evidently 2π. Otherwise we have

    $$\langle e_k, e_l\rangle = \frac{1}{i(k-l)}\left[e^{i(k-l)x}\right]_{x=-\pi}^{\pi} = 0,$$

    and the claim is verified.
- **Theorem:** $\mathcal{T}_n \subset \mathcal{T}_n^{\mathbb{C}} := \text{span}^{\mathbb{C}}(\phi_0,\dots,\phi_n,\psi_1,\dots,\psi_n) = \text{span}^{\mathbb{C}}(e_{-n},\dots,e_n)$, where 'span$^{\mathbb{C}}$' indicates that the span is the set of all linear combinations with complex coefficients.

- **Proof:** Note that $\phi_0 = e_0$. Then it suffices to show that for $k \geq 1$, $\operatorname{span}(\phi_k, \psi_k) = \operatorname{span}(e_k, e_{-k})$.
- Euler's formula says

$$e^{ikx} = \cos(kx) + i\sin(kx), \quad e^{-ikx} = \cos(kx) - i\sin(kx),$$

  which implies that $e_k, e_{-k} \in \operatorname{span}(\phi_k, \psi_k)$, hence $\operatorname{span}(\phi_k, \psi_k) \subset \operatorname{span}(e_k, e_{-k})$.
- Then it suffices to show that $\phi_k, \psi_k \in \operatorname{span}(e_k, e_{-k})$.
- To see this, note that by adding and subtracting the above two identities we get

$$\cos(kx) = \frac{e^{ikx} + e^{-ikx}}{2}, \quad \sin(kx) = \frac{e^{ikx} - e^{-ikx}}{2i},$$

  which proves the claim.

• It follows that

$$\underset{\phi \in \mathcal{T}_n^{\mathbb{C}}}{\text{minimize}} \; \|\phi - f\|^2$$

is equivalently solved by

$$\phi = \sum_{k=-n}^{n} \frac{\langle f, e_n \rangle}{2\pi} e_n,$$

or

$$\phi(x) = \sum_{k=-n}^{n} c_k e^{ikx},$$

where

$$c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx}\, dx.$$

• In fact if f is real-valued, the minimizer over $\mathcal{T}_n^{\mathbb{C}}$ coincides with the minimizer over $\mathcal{T}_n$, and we can relate the coefficients c_k with the coefficients a_k, b_k as follows, allowing us to recover either decomposition from the other:

- First consider $k = 0$:

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\, dx = \frac{a_0}{2}.$$

- Now for $k \geq 1$, apply Euler's identity $e^{-ikx} = \cos(kx) - i\sin(kx)$ within the formula for c_k to obtain

$$\begin{aligned} c_k &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\cos(kx)\, dx - \frac{i}{2\pi} \int_{-\pi}^{\pi} f(x)\sin(kx)\, dx \\ &= \frac{a_k - ib_k}{2} \end{aligned}$$

  and similarly, $c_{-k} = \frac{a_k + ib_k}{2}$.
- Then we can add and subtract our expressions for c_k and c_{-k} to obtain

$$a_k = c_k + c_{-k}, \quad b_k = \frac{c_{-k} - c_k}{i}.$$

- In summary,
$$c_0 = \frac{a_0}{2}, \quad c_k = \frac{a_k - ib_k}{2}, \; c_{-k} = \frac{a_k + ib_k}{2}, \quad k = 1, \ldots, n$$
and
$$a_0 = 2c_0, \quad a_k = c_k + c_{-k}, \; b_k = i(c_k - c_{-k}), \quad k = 1, \ldots, n.$$

- In general, the coefficients c_k, or alternatively a_k, b_k, must be computed by numerical quadrature.
  - In fact, the best choice for numerical quadrature in this context is the ***trapezoidal rule*** on an equispaced grid
  $$x_k = -\pi + kh, \quad k = 0, \ldots, M+1,$$
  where $h = \frac{2\pi}{M+1}$ is the grid spacing.
  - Concretely the quadrature rule is specified by
  $$\int_{-\pi}^{\pi} g(x)\,dx \approx h \sum_{k=0}^{M+1} \frac{g(x_k) + g(x_{k+1})}{2} = \frac{h}{2}\left[g(x_0) + g(x_{M+1})\right] + h \sum_{k=1}^{M} g(x_k)$$
  - One motivation is that it is exact for any trigonometric polynomial in $\mathcal{T}_M$.
    * Analogous to exactness of Gauss quadrature with $M+1$ points on $\mathbb{P}_{2M+1}$.
  - Since f generally does not lie in $\mathcal{T}_M$ for any finite M, in practice we choose a large enough value (possibly $M \gg n$) to yield an excellent approximation.
  - In fact, if $f(-\pi) = f(\pi)$ (which should hold in the case of 2π-periodicity at the endpoints), then $g(x_0) = g(x_{M+1})$, and the trapezoidal rule simplifies to
  $$\int_{-\pi}^{\pi} g(x)\,dx \approx h \sum_{k=0}^{M} g(x_k),$$
  which is in fact the left-endpoint Riemann sum!

30 Discrete least squares with trigonometric polynomials (§8.5)

- Sometimes in practice we are given only a 'subsampling' of a periodic function on an equispaced grid
$$x_j = -\pi + jh, \quad j = 0, \ldots, 2m-1,$$
where $h = \pi/m$.
  - Note that x_{2m} would be π, which is a redundant point assuming periodicity.
- Suppose $y_j = f(x_j)$ for $j = 0, \ldots, 2m-1$.
- We want to perform *discrete least squares* using the data (x_j, y_j), $j = 0, \ldots, 2m-1$ and basis functions $e_k(x) = e^{-ikx}$.
- We actually want to be careful about the range of the index k:

- Indeed observe that for all $j = 0, \dots, 2m-1$,
$$\begin{aligned} e_{k+2m}(x_j) &= e^{ikx_j} e^{2imx_j} \\ &= e^{ikx_j} e^{2im(-\pi + j\pi/m)} \\ &= e^{ikx_j} e^{-2\pi i m} e^{2\pi i j} \\ &= e^{ikx_j} \\ &= e_k(x_j), \end{aligned}$$
i.e., $e_{k+2m} = e_k$.
- Therefore it does not make any sense to choose a range for k that contains more than $2m-1$ consecutive integers! The extra functions we get from considering a longer range will just be redundant.
- Hence let us choose the range $k = -m, \dots, m-1$.
- Note that this gives us $2m-1$ free parameters, and we choose them to fit the data at $2m-1$ points, so this is precisely enough points to fit the discrete data *exactly*.
- After fitting, we can expand in the basis $e_{-m}, \dots, e_{m-1}$ to extrapolate the fit to the entire interval $[-\pi, \pi]$.
- We can always zero out some of the coefficients later to get the least squares fit within ny smaller basis $e_{-n}, \dots, e_n$ for $n < m$.

- Normal equations are $A\mathbf{c} = \mathbf{b}$, where
$$A_{kl} = \sum_{j=0}^{2m-1} \overline{e_k(x_j)} e_l(x_j), \quad b_k = \sum_{j=0}^{2m-1} \overline{e_k(x_j)} y_j,$$
where the indexes k, l ***run from*** $-m, \dots, m-1$, from which the least squares fit can be recovered as
$$\phi(x) = \sum_{k=-m}^{m-1} c_k e_k(x) = \sum_{k=-m}^{m-1} c_k e^{ikx}.$$

- Can write more concretely:
$$\begin{aligned} b_k &= \sum_{j=0}^{2m-1} e^{-ikx_j} y_j \\ &= \sum_{j=0}^{2m-1} e^{-ik(-\pi + jh)} y_j \\ &= e^{ik\pi} \sum_{j=0}^{2m-1} e^{-\frac{2\pi i k j}{2m}} y_j. \end{aligned}$$

- Can also compute:
$$\begin{aligned} A_{kl} &= \sum_{j=0}^{2m-1} e^{i(l-k)x_j} \\ &= \sum_{j=0}^{2m-1} e^{i(l-k)(-\pi + j\pi/m)} \\ &= e^{i(k-l)\pi} \sum_{j=0}^{2m-1} e^{\frac{2\pi i (l-k) j}{2m}}. \end{aligned}$$

- The sum $\sum_{j=0}^{2m-1} e^{\frac{2\pi i(l-k)j}{2m}} = \sum_{j=0}^{2m-1} r^j$ is geometric with $r = e^{\frac{2\pi i(l-k)}{2m}}$, hence the sum is $\frac{1-r^{2m}}{1-r} = \frac{1-e^{2\pi i(l-k)}}{1-e^{\frac{2\pi i(l-k)}{2m}}} = 0$ assuming $r \neq 1$, which holds precisely when $k-l$ is not a multiple of $2m$.
- But since k and l range from $-m$ to $m-1$, the only way $k-l$ can be a multiple of $2m$ is if $k = l$.
- When $k = l$, the sum is straightforwardly $\sum_{j=0}^{2m-1} 1 = 2m$.
- Therefore $A_{kl} = 2m\,\delta_{kl}$, and $A = 2m\,I$.

- It follows that we can recover $\mathbf{c} = \frac{1}{2m}\mathbf{b}$, or

$$c_k = \frac{e^{ik\pi}}{2m} \sum_{j=0}^{2m-1} e^{-\frac{2\pi ikj}{2m}} y_j, \quad k = -m, \ldots, m-1.$$

31 Reducing to a discrete Fourier transform (§8.5)

- The difficulty computationally lies in computing the sums $\sum_{j=0}^{2m-1} e^{-\frac{2\pi ikj}{2m}} y_j$ for $k = -m, \ldots, m-1$.
  - We will package this computation in a standard format defined in terms of the so-called ***discrete Fourier transform (DFT)***.
- The negative indices are annoying. Let's get rid of them.
  - Recall $e_{k+2m}(x_j) = e_k(x_j)$ for all $j = 0, \ldots, 2m-1$.
  - This suggests that perhaps we can instead consider a range $k = 0, \ldots, 2m-1$, by using the correspondence $-m \to m$, $-m+1 \to m+1$, ... , $-1 \to 2m-1$.
  - This implies that we can instead define
  - Indeed
$$e^{-\frac{2\pi i(k+2m)j}{2m}} = e^{-\frac{2\pi ikj}{2m}} e^{-2\pi ij} = e^{-\frac{2\pi ikj}{2m}},$$
    and
$$e^{i(k+2m)\pi} = e^{ik\pi} e^{2\pi im} = e^{ik\pi}.$$
- Therefore we can define

$$\tilde{c}_k := \frac{e^{ik\pi}}{2m} \sum_{j=0}^{2m-1} e^{-\frac{2\pi ikj}{2m}} y_j, \quad k = 0, \ldots, 2m-1$$

  and recover $\mathbf{c}$ via

$$c_0 = \tilde{c}_0, \ldots, c_{m-1} = \tilde{c}_{m-1}, \quad c_{-m} = \tilde{c}_m, \ldots, c_{-1} = \tilde{c}_{2m-1}.$$

  - In other words, if we take the last m entries of $\tilde{\mathbf{c}}$ and put them on the top of the vector, we recover $\mathbf{c}$.
- Now since $n = m$, we can actually change the index $k = -n, \ldots, n$ in our formula for c_k to an index $k = 0, \ldots, 2n$ (where all the old negative indices now correspond to .

- Finally, define

$$\hat{y}_k = \sum_{j=0}^{2m-1} e^{-\frac{2\pi ikj}{2m}} y_j, \quad k = 0, \dots, 2m-1,$$

so we can recover $\tilde{\mathbf{c}}$ from $\hat{\mathbf{y}}$ by taking $\tilde{c}_k = \frac{e^{ik\pi}}{2m}\hat{y}_k$.

- Now it is convenient to define $N = 2m-1$. Observe that $\mathbf{y} \in \mathbb{R}^N$, $\hat{\mathbf{y}} \in \mathbb{R}^N$, and both of these vectors are ***zero-indexed***.

- Moreover, the two vectors are related by the matrix-vector multiplication

$$\hat{\mathbf{y}} = \mathbf{F}\mathbf{y},$$

where $\mathbf{F} = (F_{kl})$ is an $N \times N$ complex matrix (with ***zero-indexed rows and columns***), specified by

$$F_{kl} = e^{-\frac{2\pi ikl}{N}}.$$

  - F is the matrix of the ***discrete Fourier transform (DFT)***, which is the linear transformation $\mathbb{R}^N \to \mathbb{R}^N$ that corresponds to F.
  - For clarity we may write $\mathbf{F}^{(N)}$ to emphasize that we are talking about the DFT on $\mathbb{R}^N$.

- Hence we have reduced the problem of least squares approximation by trigonometric polynomials to applying a DFT.
  - In fact we will achieve this by applying the FFT algorithm.

32 Summary of pipeline for discrete least squares with trigonometric polynomials

- Recall we are given $y_j = f(x_j)$, $j = 0, \dots, 2m-1$, where

$$x_j = -\pi + j\frac{\pi}{m}, \quad j = 0, \dots, 2m-1.$$

- Remember our goal was to find $\phi \in \mathcal{T}_n^{\mathbb{C}}$ that is the best least squares fit of the data (x_j, y_j), $j = 0, \dots, 2m-1$.
  - We assume $n < m$.

- We can think of $\mathbf{y} = (y_0, \dots, y_{2m-1})^\top \in \mathbb{C}^{2m}$ as a vector.

- Here is the pipeline we constructed:
  - First compute $\hat{\mathbf{y}} \in \mathbb{C}^{2m}$ as the FFT of $\mathbf{y}$.
  - Compute $\tilde{\mathbf{c}} = (c_0, \dots, c_{2m-1})^\top \in \mathbb{C}^{2m}$ via the formula $\tilde{c}_k = \frac{e^{ik\pi}}{2m}\hat{y}_k$, $k = 0, \dots, 2m-1$.
  - Compute $\mathbf{c} = (c_{-m}, \dots, c_{m-1})^\top \in \mathbb{C}^{2m}$ by stacking the last m entries of $\tilde{\mathbf{c}}$ onto the top.
  - Deduce

$$\phi(x) = \sum_{k=-n}^{n} c_k e^{ikx}.$$

33 Fast Fourier transform (§8.6)

- A priori, it seems that forming the matrix $\mathbf{F}$ and the performing matvecs $\mathbf{Fy}$ should cost $\sim N^2$ operations.
- It turns out that we never need to explicitly form the matrix F, and a fast algorithm is available that achieves matvecs by F in $O(N \log N)$ time.
  - This algorithm is called the ***fast Fourier transform*** (FFT).
  - *Arguably the greatest numerical algorithm of all time!*
    - In fact Gauss derived it in 1805 to perform astronomical calculations by hand, but it was rediscovered in the computing age by Cooley and Tukey in 1965.
  - Whenever you are dealing with trigonometric polynomials, Fourier series, approximation of periodic functions, *you always want to figure out how to reduce the problem to applying the right FFTs!*
- For simplicity we assume $N = 2^n$ for some n.
  - It is possible to reduce to this case with a little work, and when you call `fft` in Matlab, you don't need to worry about making sure N is a power of 2.
- Let us break the matvec $\hat{\mathbf{y}} = \mathbf{F}^{(N)}\mathbf{y}$ into two pieces:

$$\begin{aligned}
\hat{y}_k &= \sum_{l=0}^{N-1} e^{-\frac{2\pi i}{N}kl} y_l \\
&= \underbrace{\sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N}k(2m)} y_{2m}}_{\text{even-indexed part}} + \underbrace{\sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N}k(2m+1)} y_{2m+1}}_{\text{odd-indexed part}} \\
&= \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N}k(2m)} y_{2m} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N}k(2m)} y_{2m+1} \\
&= \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N/2}km} y_{2m} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N/2}km} y_{2m+1}.
\end{aligned}$$

- If we only consider indices $k = 0, \ldots, N/2-1$, these expressions look like smaller DFTs.
- To wit, define $\hat{\mathbf{y}} = \begin{pmatrix} \mathbf{w} \\ \mathbf{z} \end{pmatrix}$. Also define $\mathbf{y}^{(\mathrm{e})}$ and $\mathbf{y}^{(\mathrm{o})}$ in terms of $\mathbf{y}$ by selecting the even and odd indices, respectively, and define $\mathbf{D}^{(N)}$ to be the diagonal matrix with $D^{(N)}_{kk} = e^{-\frac{2\pi i}{N}k}$ for $k = 0, \ldots, N/2-1$.
- Then

$$\mathbf{w} = \mathbf{F}^{(N/2)}\mathbf{y}^{(\mathrm{e})} + \mathbf{D}^{(N)}\mathbf{F}^{(N/2)}\mathbf{y}^{(\mathrm{o})}.$$

- Thus we can compute $\mathbf{w}$ using a **smaller** DFT (twice) and performing one diagonal matvec (which costs only $O(N)$ operations).
- What about $\mathbf{z}$? Remarkably we can recover it from the same vectors $\mathbf{F}^{(N/2)}\mathbf{y}^{(\mathrm{e})}$ and $\mathbf{D}^{(N)}\mathbf{F}^{(N/2)}\mathbf{y}^{(\mathrm{o})}$, which don't need to be recomputed.

- Indeed compute for $k = 0, \dots, N/2-1$,

$$\begin{aligned}
\hat{y}_{k+N/2} &= \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N/2}(k+N/2)m} y_{2m} + e^{-\frac{2\pi i}{N}(k+N/2)} \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N/2}(k+N/2)m} y_{2m+1} \\
&= \sum_{m=0}^{N/2-1} e^{-2\pi i m} e^{-\frac{2\pi i}{N/2}km} y_{2m} + e^{-\pi i} e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} e^{-2\pi i m} e^{-\frac{2\pi i}{N/2}km} y_{2m+1} \\
&= \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N/2}km} y_{2m} - e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} e^{-\frac{2\pi i}{N/2}km} y_{2m+1},
\end{aligned}$$

  so

$$\mathbf{z} = \mathbf{F}^{(N/2)}\mathbf{y}^{(\mathrm{e})} - \mathbf{D}^{(N)}\mathbf{F}^{(N/2)}\mathbf{y}^{(\mathrm{o})}.$$

- Therefore here is a pseudocode for computing $\hat{\mathbf{y}} = \mathbf{F}^{(N)}\mathbf{y}$.

```
– If $N = 1$:
    ∗ Return $\hat{\mathbf{y}} = \mathbf{y}$.
– Else:
    ∗ Split $\mathbf{y}$ into even and odd parts $\mathbf{y}^{(e)}$ and $\mathbf{y}^{(o)}$.
    ∗ Compute $\mathbf{u} = \mathbf{F}^{(N/2)}\mathbf{y}^{(e)}$ and $\mathbf{v} = \mathbf{D}^{(N)}\left[\mathbf{F}^{(N/2)}\mathbf{y}^{(o)}\right]$.
    ∗ Return $\hat{\mathbf{y}} = \begin{pmatrix} \mathbf{u}+\mathbf{v} \\ \mathbf{u}-\mathbf{v} \end{pmatrix}$.
```

- Note that this is a ***recursive*** algorithm, which requires us to apply a DFT of smaller size, until we hit the base case $N = 1$.
- What is the computational cost?
  - Within each call (ignoring the cost of the recursive calls within the call), we must perform $O(N)$ operations.
  - In total we will perform
    - 1 call of size $N \to O(N)$ total cost
    - 2 calls of size $N/2 \to O(N)$ total cost
    - 4 calls of size $N/4 \to O(N)$ total cost
    - ...
    - N calls of size $1 \to O(N)$ total cost
  - There are $\log_2(N) + 1$ different call sizes, yielding $O(N \log N)$ total cost!

Part III
Eigenvalue problems

34 Appetizer: power method

- We have constructed eigenvectors in theory, but how does one compute them in practice?

- **Power method:** the simplest *iterative method* for computing an eigenvector (in particular, the *dominant eigenvector*)
  - *Iterative* means that the eigenvector is constructed as a limit of a convergent sequence of vectors
- It can also be viewed as the simplest *matrix-free* method
  - **Recall:** *matrix-free* means that we only rely on matrix-vector multiplication (do not necessarily need to construct the full matrix A)
  - **Recall:** sometimes matvecs are cheaper to implement than forming the entire matrix A and doing a full matvec
    - e.g., A is sparse + low-rank
- In practice, one often only wants one or a few (i.e., $O(1)$) key eigenpairs
  - Important examples to follow
  - If matvec costs $O(n)$, then it is possible to achieve this in $O(n)$ time
  - Much better than a full diagonalization, which is always $O(n^3)$
- Start with any initial vector $x \in \mathbb{R}^n$ (or $\mathbb{C}^n$).
  - Consider the sequence Ax, A^2x, A^3x...
  - What does this converge to?
    - **Trick question:** usually doesn't converge, or converges to zero vector
  - But if we 'suitably normalize' the sequence, what does it converge to? (Assume for simplicity that A is diagonalizable)
    - The *dominant eigenvector* (if it exists), i.e., the eigenvector whose eigenvalue has maximal absolute value
- Indeed, suppose A has a (strictly) dominant eigenpair (λ_1, v_1) and A is diagonalizable
  - In particular, $\lambda_1 \neq 0$.
  - Note that if A is real, since all complex eigenvalues come in conjugate pairs, any dominant eigenvalue must also be real.
  - Then write $x = \sum_{i=1}^n c_i v_i$, where $v_2, \ldots, v_n$ complete v_1 to a basis of eigenvectors.
  - Then $A^k x = \sum_{i=1}^n c_i \lambda_i^k v_i$.
  - So $\lambda_1^{-k} A^k x = \sum_{i=1}^n c_i \left(\frac{\lambda_i}{\lambda_1}\right)^k v_i$.
  - But for $i > 1$, $\left|\frac{\lambda_i}{\lambda_1}\right| < 1$, so $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$.
  - Hence $\lambda_1^{-k} A^k x \to c_1 v_1$.
  - Note that we're in trouble if $c_1 = 0$. We'll come back to this later.
- In practice, we don't know λ_1, so we can't form the sequence $\{\lambda_1^{-k} A^k x\}_{k=0}^{\infty}$.
- It is not important to know λ_1, we just need to normalize the iterates so that they don't collapse to zero or fly off to infinity.

- Define
$$u^{(k)} := \frac{A^k x}{\|A^k x\|},$$
where we adopt for now the 2-norm as our norm without including in the notation.
  - Note that as $k \to \infty$, as long as $c_1 \neq 0$, we have
$$u^{(k)} = \frac{\sum_{i=1}^n c_i \lambda_i^k v_i}{\|\sum_{i=1}^n c_i \lambda_i^k v_i\|} \approx \frac{c_1 \lambda_1^k v_1}{|c_1 \lambda_1^k| \|v_1\|} = \underbrace{\frac{c_1}{|c_1|}\left(\frac{\lambda_1}{|\lambda_1|}\right)^k}_{=:z^{(k)}} \frac{v_1}{\|v_1\|},$$
where $|z^{(k)}| = 1$. (Hence in the real case, $z^{(k)} = \pm 1$.)
  - This implies the following.
- **Theorem:** With notation as above there exists a complex unit $z^{(k)}$ (which is ± 1 in the real case) such that $\lim_{k\to\infty} \frac{u^{(k)}}{z^{(k)}} = \frac{v_1}{\|v_1\|}$.
- This suggests an algorithm that converges to v_1 (up to scaling), but how to terminate?
  - Given a guess $u^{(k)}$, we want to consider a stopping criterion based on the 'relative residual norm'
$$\frac{\|Au^{(k)} - \lambda_1 u^{(k)}\|}{\lambda_1}.$$
  - However, we don't know λ_1, so we need to come up with an iterative guess for it....
- How to estimate eigenvalue given eigenvector guess?
  - Note that if $Au = \lambda u$, then $u \cdot (Au) = \lambda \|u\|^2$, hence $\lambda = \frac{u^\top A u}{\|u\|^2}$.
  - If u is normalized, then we have $\lambda = u \cdot (Au) = u^\top A u$.
  - Therefore guess $\lambda_1^{(k)} = u^{(k)} \cdot (Au^{(k)})$, and estimate relative residual norm as
$$\frac{\|Au^{(k)} - \lambda_1^{(k)} u^{(k)}\|}{\lambda_1^{(k)}}$$
- This suggests the practical algorithm (***power method***):
  - Given initial guess $x^{(0)} \in \mathbb{R}^n$, maximum number of iterations m, ability to perform matvecs by A, tolerance ε.
  - For $k = 0, \ldots, m$:
    - Set $u^{(k)} = x^{(k)}/\|x^{(k)}\|$.
    - Set $x^{(k+1)} = Au^{(k)}$.
    - Set $\lambda_1^{(k)} = u^{(k)} \cdot x^{(k+1)}$.
    - If $\|x^{(k+1)} - \lambda_1^{(k)} u^{(k)}\|/\lambda_1^{(k)} \leq \varepsilon$:
      - BREAK.
  - Return $(\lambda_1^{(k)}, u^{(k)})$.
- The power method works even if A is not diagonalizable, as long as there is a strictly dominant eigenvector
  - Proof more subtle, relies on Jordan normal form
- Can generalize to the problem of finding $k \geq 1$ most dominant eigenvectors (***subspace iteration***)

35 Application: Markov chains

- Think of a system with n different states, labeled $1, \dots, n$
  - Interested in random transitions between states at discrete time steps
  - Specifically, suppose that for each state j, we are given a column of probabilities $p_j := (p_{1j}, \dots, p_{nj})^\top$
  - Given that we are in state j at time t, p_{ij} represents the probability that we move to state i at time $t+1$
  - The transition probabilities only depend on the current state (*Markov property*)
  - To ensure the probabilistic interpretation we must have
    * $p_{ij} \geq 0$ for all i, j
    * $\sum_{i=1}^n p_{ij} = 1$ for all j
    * Can also write $\mathbf{1}^\top p_j = 1$ for all j, where $\mathbf{1}$ is the vector of all ones
  - $P = (p_{ij})$ satisfying these two properties is called a ***(column) stochastic matrix***
    * Can rephrase second property concisely as $P^\top \mathbf{1} = \mathbf{1}$.
    * Can write $P \geq 0$ to indicate entrywise nonnegativity.
  - P is in addition **positive** if $p_{ij} > 0$ for all i, j, in which case we may write $P > 0$.
    * We will assume positivity later for simplicity.
- Note that $\pi \in \mathbb{R}^n$ defines a **probability distribution** over $\{1, \dots, n\}$ if $\pi_i \geq 0$ for all i and $\sum_{i=1}^n \pi_i = 1$. (We can write these properties alternatively as $\pi \geq 0$ and $\mathbf{1}^\top \pi = 1$.)
- Let $X_t \in \{1, \dots, n\}$ be the random variable denoting our state at time t
  - **Claim:** If $\pi = (\pi_1, \dots, \pi_n)^\top$ is the probability distribution for X_t at time t, then the probability distribution π' at time $t+1$ is $P\pi$
    * *Check*:

$$\begin{aligned} \pi'_i &= \mathbf{P}(X_{t+1} = i) \\ &= \sum_{j=1}^n \mathbf{P}(X_{t+1} = i \,|\, X_t = j)\ \mathbf{P}(X_t = j) \\ &= \sum_{j=1}^n p_{ij}\pi_j \\ &= [P\pi]_i \end{aligned}$$

  - Therefore the evolution of the probability distribution of our state can be understood purely in terms of repeated application of the stochastic matrix P
- Notice that if $\mathbf{1}^\top \pi = 1$, then

$$\mathbf{1}^\top (P\pi) = (P^\top \mathbf{1})^\top \pi = \mathbf{1}^\top \pi = 1$$

  - Moreover if $\pi \geq 0$ (entrywise), then $P\pi \geq 0$ as well.
  - This is good because it confirms that P maps probability distributions to probability distributions, as we suspected.

- For arbitrary initial distribution π, interested in the limit

$$\pi_\infty = \lim_{k\to\infty} P^k \pi$$

  - If it exists, π_∞ should then be a probability vector as well.
  - If π_∞ exists, then by taking the limit of both sides of $\pi_{k+1} = P(\pi_k)$, we obtain

$$\pi_\infty = P\pi_\infty.$$

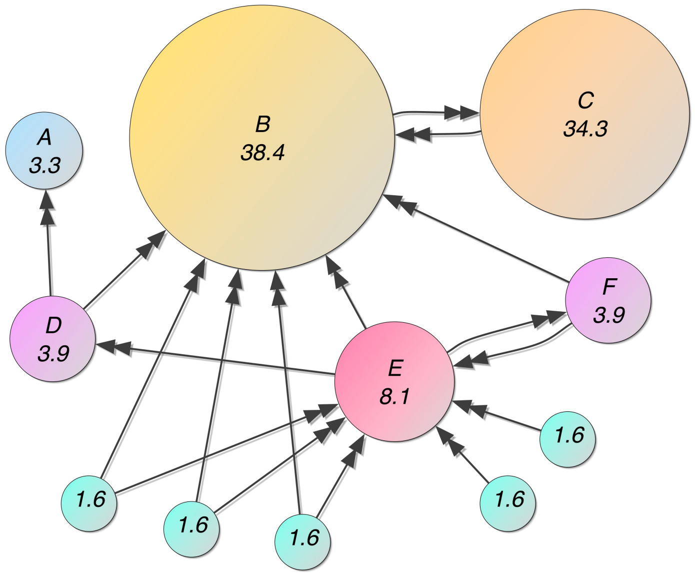
- If the limit exists, π_∞ is called the ***equilibrium / invariant distribution***.
  - Thusly led to ask: are the conditions for the power method satisfied?
  - If so, then π_∞ is the (suitably normalized) dominant eigenvector of P.
- **Perron-Frobenius Theorem:** Let P be a positive stochastic matrix. Then P admits a dominant eigenvector with eigenvalue 1.
  - More refined versions of the theorem are available with weaker assumptions.

36 Sub-application: PageRank

- The Internet...is a directed graph.
  - Again given a collection of states (webpages) indexed $1, \ldots, n$.
  - A directed graph is a set E of edges (i, j) (order matters).
    - Indicates a hyperlink from page i to page j.
  - A directed graph is equivalently specified as an *adjacency matrix* A, defined by

$$A_{ij} = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E. \end{cases}$$

  - Want to *rank* pages according to some score.
    - Score should measure the 'centrality' of a webpage.
      - Not really satisfactory just to measure how many pages link to it (why?).
    - PageRank offers such a score, an example of an 'eigenvector centrality' measure.



- We will build a Markov chain (stochastic matrix) out of our adjacency matrix A.
  - In fact, example of a general recipe for producing a stochastic matrix from a nonnegative matrix .
  - Just appropriately normalize each column so that its sum is one.
    - Assume each column has a nonzero entry, i.e., each page has a link to it.
    - Otherwise, if there is a page that no one links to, we can throw it out.
  - Formula: define $\tilde{P}$ stochastic via $\tilde{P}^\top = \operatorname{diag}(A\mathbf{1})^{-1}A$, or

$$\tilde{P}_{ji} = \begin{cases} \frac{1}{\deg^-(i)}, & (i,j) \in E \\ 0, & (i,j) \notin E, \end{cases}$$

    where $\deg^-(i)$ is the 'outdegree' of a page.
  - *Probabilistic interpretation*: Markov chain on the webpages, at each time step pick a uniformly random hyperlink from current page and click it.
  - *Not quite satisfactory yet....*
    - Power method may be very slow to converge, as you will explore in the homework. By changing the problem slightly, we can ensure that it always converges rapidly.
- Introduce a damping factor $\tau \in (0,1)$.
  - Instead of clicking a random link, do the following:
    - With probability τ, pick a uniformly random link on current page and click it.
    - With probability $1-\tau$, go to a random webpage picked uniformly from the *entire* set of webpages.
  - Corresponds to defining a new Markov chain

$$P = \tau\tilde{P} + (1-\tau)\frac{1}{n}\mathbf{1}\mathbf{1}^\top$$

    (note that $\mathbf{1}\mathbf{1}^\top$ is just the matrix of all ones).

  - Note P is stochastic and positive.
  - Hence (by Perron-Frobenius) can let $\pi^{(\star)}$ be its dominant eigenvector, scaled so that $\pi^{(\star)} \geq 0$ and $\mathbf{1}^\top \pi^{(\star)} = 1$.
    * $\pi_i^{(\star)}$ is the PageRank score for webpage i.
    * Corresponds to the fraction of time that our Markov chain spends on state i in the long-time limit.
  - Can compute with power method.

- The ***PageRank algorithm*** is then *just the power method applied to this matrix P*, except even simpler: we don't need to normalize at each iteration (since the dominant eigenvalue is exactly 1).
  - Indeed, if we start with initial guess $\pi^{(0)}$ that is a probability distribution and then define $\pi^{(k+1)} = P\pi^{(k)}$ for all $k = 0, 1, 2, \ldots$, then it is guaranteed that $\pi^{(k)}$ is a probability distribution at every iteration, and $\lim_{k\to\infty} \pi^{(k)} = \pi^{(\star)}$.

37 Linear algebra review

- Review:
  - eigenvalues
  - diagonalization
  - spectral theorem for symmetric matrices
  - viewing matrix factorizations as sums of rank-one matrices
  - orthogonal matrices
  - orthogonal projectors
  - SVD (full, thin, compact, truncated)
  - relation between SVD and spectral theorem

38 Principal component analysis (PCA)

- Suppose we have a set of 'data points' $x_j \in \mathbb{R}^m$, $j = 1, \ldots, n$
  - m is the dimension of the data space, n is the number of data points
- **Goal:** want to find a subspace of dimension k (given) that 'captures' the data as well as possible
  - In statistics terms, subspace that 'explains as much variance' as possible
- Assume that data has zero *empirical mean*, i.e., $\frac{1}{n}\sum_{i=1}^n x_i = 0$
  - Always possible by replacing $x_i \leftarrow x_i - \frac{1}{n}\sum_{i=1}^n x_i$.
  - This is called ***de-meaning*** the data.
- Can form data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{m\times n}$
  - **FYI:** $\frac{1}{n}XX^\top = \frac{1}{n}\sum_i x_i x_i^\top$ is called the *empirical covariance* matrix

- Need quantitative formulation as an optimization problem
  - Want to optimize over $q_1, \ldots, q_k$ (forming an orthonormal basis for best subspace)
  - Let $Q = [q_1, \ldots, q_k]$, so alternatively can optimize over $Q \in \mathbb{R}^{m\times k}$ such that $Q^\top Q = I_k$
  - There is not a completely standard name for a matrix satisfying $Q^\top Q = I_k$, but we can say that such Q is ***partially orthogonal*** or simply ***has orthonormal columns***.
  - Note: $P_Q := QQ^\top$ is orthogonal projection onto $\mathrm{span}(Q) = \mathrm{col}(Q)$. [Exercise: check this!]
- Objective function to be **minimized** is $f(Q) := \sum_{i=1}^n \|x_i - P_Q x_i\|^2$
  - Measures the deviation of x_i from its orthogonal projection onto the subspace (i.e., the closest point in the subspace), summed over all the data points i
  - Clearly zero if $x_i \in \mathrm{span}(Q)$ for all i, i.e., if all the data lies in the subspace
  - Measures deviation from this ideal scenario
- **Notation:** the 'span' of a matrix indicates the span of its columns. It is literally the same as the column space.
- Compute

$$f(Q) = \sum_{i=1}^n \|x_i - P_Q x_i\|^2 = \sum_{i=1}^n \|(I - QQ^\top)x_i\|^2$$

  - **Note:** $(I - QQ^\top)x_i$ is the i-th column of $(I - QQ^\top)X = X - QQ^\top X$, and the squared Frobenius norm is the same as the sum of the squared column norms.
  - Hence $f(Q) = \|X - QQ^\top X\|_{\mathrm{F}}^2$.
  - Let $X = U\Sigma V^\top = \sum_{i=1}^r \sigma_i u_i v_i^\top$ be a ***compact*** SVD for X, where $r := \mathrm{rank}(X)$.
    - Here $U = [u_1, \ldots, u_r]$ and $V = [v_1, \ldots, v_r]$.
  - Let $X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top = \sum_{i=1}^k \sigma_i u_i v_i^\top$ be a ***truncated*** SVD (to rank k).
    - Here $\tilde{U} = [u_1, \ldots, u_k]$ and $\tilde{V} = [v_1, \ldots, v_k]$.
  - Note that $\mathrm{rank}(QQ^\top X) \le k$, so we know from the 'best rank-k approximation' property of the truncated SVD that if we could choose Q such that $QQ^\top X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$, it would be optimal.
  - But then just choose $Q = \tilde{U}$, so

$$QQ^\top X = \sum_{i=1}^r \tilde{U}\tilde{U}^\top \sigma_i u_i v_i^\top = \sum_{i=1}^k \sigma_i u_i v_i^\top = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$$

- In summary, the optimal subspace is given by the *span of the top k left singular vectors of the (de-meaned) data matrix X.*
  - Alternatively, it is given by *span of the dominant k eigenvectors of the empirical covariance matrix $\frac{1}{n}XX^\top$.*
- For a data point x_j, its component $u_i \cdot x_j$ in the i-th singular vector is called the i-th *principal component* of the data.
  - Thus the j-th column of the matrix $Q^\top X \in \mathbb{R}^{k\times n}$, which is $(u_1 \cdot x_j, \ldots, u_k \cdot x_j)^\top \in \mathbb{R}^k$ is the vector consisting of the first k principal components of the j-th data point.

- We can use PCA for dimensionality reduction:
  - The columns y_j of $Y := Q^\top X$ are the images of the data points x_j under the linear transformation $Q^\top$ mapping $\mathbb{R}^m \to \mathbb{R}^k$.
  - Thus the map $Q^\top$ can be thought of as 'compressing' our dataset.
  - To recover the original dataset approximately from the compressed dataset, we can form $\tilde{X} = QY = QQ^\top X$.
    - Recall that Q was optimized to minimize the error $\|\tilde{X} - X\|_{\mathrm{F}}$ of our recovery.
    - Thus, although the compression is 'lossy,' it is as faithful as possible given that we are compressing the data into dimension k.

39 Subspace iteration

39.1 Basic algorithm

- Let $A \in \mathbb{R}^{n\times n}$ be diagonalizable (for simplicity). Let $\lambda_1, \ldots, \lambda_n$ denote its eigenvalues (possibly repeated according to multiplicity), ordered such that

$$|\lambda_1| \geq \cdots \geq |\lambda_k| > |\lambda_{k+1}| \geq \cdots \geq |\lambda_n|.$$

- Let $u_1, \ldots, u_n$ denote a corresponding basis of eigenvectors, and let $U = [u_1, \ldots, u_k] \in \mathbb{R}^{n\times k}$.
  - For now we are not assuming that A is ymmetric so these are not necessarily orthonormal!
- We want to find the ***subspace*** $\mathrm{span}(U)$, in some sense.
- Let $W = [w_1, \ldots, w_k] \in \mathbb{R}^{n\times k}$. (Initial guess.)
- **Theorem:** For 'generic' initial guess $W \in \mathbb{R}^{n\times k}$, there exists a suitable sequence of invertible matrices $C_l \in \mathbb{R}^{k\times k}$, $l = 0, 1, 2, \ldots$, such that

$$\lim_{l\to\infty} A^l W C_l = U.$$

  - In the proof, we will say what we mean by 'generic' (generalizing condition for power method), and we will construct the sequence C_l.
- **Implication:** As $l \to \infty$, $A^l W = A(A(A(\cdots AW)\cdots)))$ does not literally 'converge' to U, but its *span* 'converges' to the *span* of U as a subspace.
  - However, if we literally form $A^l W$ for l large, we will have issues with numerical overflow/underflow.
  - Therefore the actual practical algorithm will include a generalization of the normalization procedure used in the power method.
- This suggests the practical algorithm (***subspace iteration***), which we present for now without a stopping criterion:
  - Given generic initial guess $W \in \mathbb{R}^{n\times k}$, maximum number of iterations m, ability to perform matrix-vector multiplications by A.
  - Let V have orthonormal columns with the same span as the columns of W. (*This can be achieved by the Gram-Schmidt procedure, or equivalently, a rectangular QR factorization. The Matlab function* `orth` *can also be used.*)

  - For $j = 0, 1, \ldots, m$:
    * Set $W \leftarrow AV$. (*Note that this entails k matrix-vector multiplications by A, one for each column.*)
    * Let V have orthonormal columns with the same span as the columns of W.
  - Return V.

- Note that the bulk of the work in each iteration consists of (1) the k matvecs by A and (2) the orthonormalization of the k columns of W.
  - The orthonormalization cost is $O(nk^2)$ as can verified by counting operations in Gram-Schmidt.
  - For example, if$A$ is sparse with $O(n)$ nonzero entries, then the cost of the matvecs is $O(nk)$ per iteration. If A is completely dense (worst case), it is still only $O(n^2k)$ per iteration.
  - Compare to the cost $O(n^3)$ of a complete diagonalization!
- If the number of iterations m is sufficiently large, then the final output V will have columns forming an orthonormal basis for a subspace very close to span(U).
- Later we will improve the algorithm to include a criterion for stopping when the algorithm has converged within a specified tolerance. However, this takes some more care than in the case of the power method.
- We will also explain later how to recover *actual eigenpairs* from the recovered subspace span(V)

39.2 Convergence proof

- **Proof (of theorem):**
  - We will construct C_l below and present the genericity condition on W in bold italics.
  - First let P have columns $u_1, \ldots, u_n$, so $P = (U\ V)$ for suitable V. (Note that the columns are not necessarily orthogonal since A is not assumed symmetric.)
  - Then $A = P\Lambda P^{-1}$, with

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) =: \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix},$$

    where the top-left block is $k \times k$. Note that the diagonal matrix Λ_1 is invertible.
  - Let

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} := P^{-1}W,$$

    where the top block Y_1 is $k \times k$. ***We assume that*** Y_1 ***has full rank, i.e., is invertible.***
  - In this case, let $C_l = Y_1^{-1}\Lambda_1^{-l}$, so

$$A^l W C_l = P\Lambda^l P^{-1} W C_l = P \begin{pmatrix} \Lambda_1^l Y_1 \\ \Lambda_2^l Y_2 \end{pmatrix} Y_1^{-1}\Lambda_1^{-l} = P \begin{pmatrix} I_k \\ \Lambda_2^l Y_2 Y_1^{-1} \Lambda_1^{-l} \end{pmatrix}.$$

  - We claim that the entire lower block $\Lambda_2^l Y_2 Y_1^{-1} \Lambda_1^{-l}$ in the last expression converges to zero as $l \to \infty$.
    * To see this, it suffices to show that the entries of $M := \Lambda_2^l B \Lambda_1^{-l}$ all converge to zero, where $B := Y_2 Y_1^{-1}$.

  - ∗ But the (i,j)-th entry of this $(n-k)\times k$ matrix is $|\lambda_{i+k}/\lambda_j|^l B_{ij}$, and $|\lambda_{i+k}/\lambda_j| < 1$ for all $i = 1,\dots,n-k$, $j = 1,\dots,k$, so the claim follows.
  - Therefore $\lim_{l\to\infty} A^l X C_l = P \begin{pmatrix} I_k \\ 0 \end{pmatrix} = (U\ V) \begin{pmatrix} I_k \\ 0 \end{pmatrix} = U$, as was to be shown.

39.3 Distance between subspaces

- We need to include a condition that allows us to detect when subspace iteration has converged to within a desired tolerance.
- It is tricky to do this because we want to think of subspace iteration as providing a sequence of *subspaces*? How do we measure the distance between two subspaces?
- One way to measure the distance between two subspaces is to compute a matrix norm distance between their orthogonal projection matrices.
- Consider two subspcaes span(V) and span($\tilde{V}$), where $V^\top V = I_k$ and $\tilde{V}^\top \tilde{V}$ (i.e., V and $\tilde{V}$ have orthonormal columns).
  - Then $P := VV^\top$ and $Q := \tilde{V}\tilde{V}^\top$ are the corresponding orthogonal projection matrices.
  - We will compute $\|P-Q\|_{\mathrm{F}}^2$.
- Need two facts involving the matrix trace 'Tr[B]' (also denoted 'tr(B)' or 'trace(B)') which is the sum of the diagonal elements of a square matrix B.
  - **Fact 1:** for any square matrix B, $\|B\|_{\mathrm{F}}^2 = \mathrm{Tr}[BB^\top] = \mathrm{Tr}[B^\top B]$.
  - **Fact 2:** for any matrices B and C such that B and $C^\top$ are the same size, $\mathrm{Tr}[BC] = \mathrm{Tr}[CB]$.
  - We verified in class, but it is a good exercise to try yourself!
- Computation:
  - First expand using Fact 1:
$$\|P-Q\|_{\mathrm{F}}^2 = \mathrm{Tr}[(P-Q)(P-Q)] = \mathrm{Tr}[PP] - \mathrm{Tr}[QP] - \mathrm{Tr}[PQ] - \mathrm{Tr}[QQ].$$
  - Since P and Q are projectors, $PP = P$ and $QQ = Q$. Moreover, $\mathrm{Tr}[QP] = \mathrm{Tr}[PQ]$ by Fact 2.
  - Then
$$\begin{aligned}\|P-Q\|_{\mathrm{F}}^2 &= \mathrm{Tr}[P] + \mathrm{Tr}[Q] - 2\mathrm{Tr}[PQ] \\ &= \mathrm{Tr}[VV^\top] + \mathrm{Tr}[\tilde{V}\tilde{V}^\top] - 2\mathrm{Tr}[VV^\top\tilde{V}\tilde{V}^\top].\end{aligned}$$
  - Now use Fact 2 again to rearrange the matrices within the traces:
$$\begin{aligned}\|P-Q\|_{\mathrm{F}}^2 &= \mathrm{Tr}[V^\top V] + \mathrm{Tr}[\tilde{V}^\top\tilde{V}] - 2\mathrm{Tr}[\tilde{V}^\top VV^\top\tilde{V}] \\ &= \mathrm{Tr}[I_k] + \mathrm{Tr}[I_k] - 2\mathrm{Tr}[(V^\top\tilde{V})^\top(V^\top\tilde{V})] \\ &= 2k - 2\|V^\top\tilde{V}\|_{\mathrm{F}}^2,\end{aligned}$$

    where we have used the fact that $\mathrm{Tr}[I_k] = k$, as well as Fact 1 once again.

- It is also useful to note that

$$\|P\|_{\mathrm{F}}^2 = \mathrm{Tr}[VV^\top VV^\top] = \mathrm{Tr}[V^\top VV^\top V] = \mathrm{Tr}[I_k I_k] = \mathrm{Tr}[I_k] = k,$$

so the "relative distance" can be computed as

$$\frac{\|P-Q\|_{\mathrm{F}}}{\|P\|_{\mathrm{F}}} = \sqrt{2}\cdot\sqrt{1-k^{-1}\|V^\top\tilde{V}\|_{\mathrm{F}}^2}.$$

- Therefore we have proved the following:

- **Theorem:** If V and $\tilde{V}$ have orthonormal columns and P and Q are the orthogonal projection matrices onto span(V) and span$(\tilde{V})$, respectively, then

$$\|P-Q\|_{\mathrm{F}}^2 = 2\left(k-\|V^\top\tilde{V}\|_{\mathrm{F}}^2\right),$$

and

$$\frac{\|P-Q\|_{\mathrm{F}}}{\|P\|_{\mathrm{F}}} = \sqrt{2}\cdot\sqrt{1-k^{-1}\|V^\top\tilde{V}\|_{\mathrm{F}}^2}.$$

- The key point is that we can compute the Frobenius norm of the difference without every forming P and Q explicitly, which would take $O(n^2k)$ operations!
  - Instead the cost of evaluation is only $O(nk^2)$, which is dominated by the much smaller cost of forming $V^\top\tilde{V}$.

39.4 Full pseudocode

- If V is our current guess in subspace iteration and $\tilde{V}$ is the next guess, then we can use the relative Frobenius norm distance between the orthogonal projection matrices onto their spans as a measure of convergence.
- This suggests the following pseudocode for ***subspace iteration***.

```
– Given generic initial guess $W \in \mathbb{R}^{n\times k}$, maximum number of iterations $m$, ability to
  perform matrix-vector multiplications by $A$, tolerance $\varepsilon$.
– Let $V$ have orthonormal columns with the same span as the columns of $W$. (This can be
  achieved by the Gram-Schmidt procedure, or equivalently, a rectangular QR factorization.
  The Matlab function orth can also be used.)
– For $j = 0, 1, \ldots, m$:
    ∗ Set $W \leftarrow AV$. (Note that this entails $k$ matrix-vector multiplications by $A$, one for
      each column.)
    ∗ Let $\tilde{V}$ have orthonormal columns with the same span as the columns of $W$.
    ∗ Compute $\delta := \sqrt{2}\cdot\sqrt{1-k^{-1}\|V^\top\tilde{V}\|_{\mathrm{F}}^2}$.
    ∗ $V \leftarrow \tilde{V}$.
    ∗ If $\delta < \varepsilon$.
        · BREAK.
– Return $V$.
```

- The output V satisfies span(V) = span(U) approximately.

39.5 Recovering eigenpairs via Rayleigh-Ritz

- **We will assume in this section that A is symmetric.**
- Subspace iteration (assuming for simplicity that we have fully converged it) furnishes a matrix $V \in \mathbb{R}^{n\times k}$ whose columns form an orthonormal basis for span(U), so span(U) = span(V), the span of the dominant k eigenvectors.
  - Since A is symmetric, by the spectral theorem we can assume that the columns $u_1, \dots, u_k$ of U are othonormal, i.e., $U^\top U = I_k$.
- However, we have not actually recovered eigenpairs!
  - There is some ambiguity in the task of recovering "THE" eigenvectors, due to the arbitrary scaling of the eigenvectors and possibly due to repeated eigenvalues, so we cannot hope to recover U exactly.
- Instead we hope to recover the dominant eigenvalues $\lambda_1, \dots, \lambda_k$ exactly, as well as ***some*** corresponding orthonormal eigenvectors $q_1, \dots, q_k$ such that

$$Aq_j = \lambda_j q_j, \quad j = 1, \dots, k.$$

- It is useful to notice that the eigenvector equations $Au_j = \lambda_j u_j$, $j = 1, \dots, k$, can be repackaged in the single equation

$$AU = U\Lambda_1,$$

  where $\Lambda_1 = \mathrm{diag}(\lambda_1, \dots, \lambda_k)$ is the same diagonal $k \times k$ matrix appearing in our earlier proof.
- Therefore similarly we seek $Q = [q_1, \dots, q_k]$ with orthonormal columns such that $AQ = Q\Lambda_1$.
- The ***Rayleigh-Ritz procedur***e at the end of subspace iteration tells us how to do this.
  - First, form $\tilde{A} = V^\top AV$, which is a $k \times k$ symmetric matrix.
  - Then perform a complete diagonalization $\tilde{A} = \tilde{U}\tilde{\Lambda}\tilde{U}^\top$ where $\tilde{U}$ is a $k \times k$ orthogonal matrix and $\tilde{\Lambda}$ is a diagonal matrix with diagonal entries ordered non-increasingly by absolute value.
    - Such a factorization exists by the spectral theorem. We will not describe how it is computed, but note that the problem size is $k \times k$ (independent of n), and in fact the cost is $O(k^3)$.
  - Then recover $Q = V\tilde{U}$.
- There are two claims about this procedure:
  - **Claim 1:** $\tilde{\Lambda}$ produced by Rayleigh-Ritz satisfies $\tilde{\Lambda} = \Lambda_1$
  - **Claim 2:** Q produced by Rayleigh-Ritz satisfies the eigenvector equations $AQ = Q\Lambda_1$.
- We begin with a **sub-claim:** $V = UO$ for the orthogonal $k \times k$ matrix $O = U^\top V$.
  - To see that $V = UO$, observe that

$$UO = (UU^\top)V = P_U V = V,$$

    where the last equality follows from the fact that span(U) = span(V) and here $P_U = UU^\top$ is the orthogonal projection matrix onto span(U).

  - To see that O is orthogonal, observe that

$$O^\top O = (U^\top V)^\top (U^\top V) = V^\top U U^\top V = V^\top (P_U V) = V^\top V = I_k,$$

    so $O^\top O = I_k$ and O is orthogonal.
  - This completes the proof of the subclaim.

- Then we can write the matrix

$$\tilde{A} = V^\top A V = (UO)^\top A (UO) = O^\top U^\top A U O,$$

  and using the fact that $AU = U\Lambda_1$, we further compute

$$\tilde{A} = O^\top (U^\top U)\Lambda_1 O = O^\top I_k \Lambda_1 O = O^\top \Lambda_1 O,$$

  i.e.,

$$\tilde{A} = O^\top \Lambda_1 O.$$

  - In fact this furnishes a diagonalization of $\tilde{A}$. To make this clear, observe that if we take $P = O^\top$, then $\tilde{A} = P\Lambda_1 P^{-1}$. Therefore the eigenvalues of $\tilde{A}$ are precisely $\lambda_1, \dots, \lambda_k$, and since the eigenvalues in $\tilde{\Lambda}$ have been ordered appropriately, it follow that $\Lambda_1 = \tilde{\Lambda}$, i.e., **Claim 1** holds.

- Then to prove **Claim 2**, compute:

$$\begin{aligned}
AQ &= AV\tilde{U} \\
&= AUO\tilde{U} \\
&= (AU)O\tilde{U} \\
&= U\Lambda_1 O\tilde{U} \\
&= U(I_k)\Lambda_1 O\tilde{U} \\
&= U(OO^\top)\Lambda_1 O\tilde{U} \\
&= UO(O^\top \Lambda_1 O)\tilde{U} \\
&= UO(\tilde{A})\tilde{U} \\
&= UO(\tilde{U}\Lambda_1\tilde{U}^\top)\tilde{U} \\
&= UO\tilde{U}\Lambda_1(\tilde{U}^\top\tilde{U}) \\
&= (UO)\tilde{U}\Lambda_1 \\
&= (V\tilde{U})\Lambda_1 \\
&= Q\Lambda_1,
\end{aligned}$$

  which establishes the claim!

40 Randomized SVD

- Suppose that $A \in \mathbb{R}^{m\times n}$ and assume $m \geq n$.
- The entire information of A is contained within the thin SVD:

$$A = \hat{U}\hat{\Sigma}\hat{V}^\top$$

  where $\hat{U} \in \mathbb{R}^{m\times n}$ has orthonormal columns and $\tilde{V} \in \mathbb{R}^{n\times n}$ is orthogonal.

- Suppose we want to compute the rank-k truncated SVD

$$A_k := U\Sigma V^\top,$$

where $U = \hat{U}_{:,1:k}$ consists of the top k left singular vectors, $\Sigma = \Sigma_{1:k,1:k}$ contains the top k singular values, and $V = V_{:,1:k}$ consists of the top k right singular vectors.

- Note that A_k can be computed by applying subspace iteration + Rayleigh-Ritz ***twice*** separately:
  - We know that the k dominant eigenvectors of $AA^\top = \hat{U}\hat{\Sigma}^2\hat{U}^\top$ furnish the top k left singular vectors U. (The corresponding eigenvalues are the squares of the top k singular values.)
  - Meanwhile the k dominant eigenvectors of $A^\top A = \hat{V}\hat{\Sigma}^2\hat{V}^\top$ furnish the top k right singular vectors U. (The corresponding eigenvalues are again the squares of the top k singular values.)
  - Thus we can apply subspace iteration (looking for the dominant k eigenvectors) separately to both $AA^\top$ and $A^\top A$ to deduce the top k left and right singular vectors, i.e., U and V, as well as the top k singular values Σ.
  - ***Note carefully that the meaning of the letter V now is different than in the discussion of subspace iteration! Here it is indicating the right singular vectors. Sorry we sometimes run out of intuitive letters!***

- The ***randomized SVD*** provides an alternative approach to approximating the truncated SVD A_k.
  - Unlike subspace iteration, we do not rely on the convergence of an iterative algorithm (except within a small full SVD analogous to the Rayleigh-Ritz problem, the cost of which we will view as negligible).
    - And indeed, if the gap between σ_k and σ_{k+1} is small, the convergence of subspace iteration will be slow!
  - On the other hand, with the randomized SVD, we may have to accept some approximation based on how rapidly the singular values of A decay (as we will see).

40.1 Toy problem: the rank k case

- Suppose for simplicity that $\mathrm{rank}(A) = k$, meaning that the dimension of the range of A is k.

- The let the columns of $Z = [z_1, \ldots, z_k] \in \mathbb{R}^{n\times k}$ be k "random" vectors (the details of which shall be spelled out more carefully later).

- Note that $Az_1, \ldots, Az_k \in \mathrm{range}(A)$. If Z is chosen randomly, we expect with probability 100% that these vectors will be linearly independent, hence span the entire subspace range(A), i.e., we expect that

$$\mathrm{span}(Az_1, \ldots, Az_k) = \mathrm{span}(AZ) = \mathrm{range}(A).$$

- In fact in this simple case, $\mathrm{range}(A) = \mathrm{span}(U)$.
  - The reason is that if $\mathrm{rank}(A) = k$, then $\sigma_1, \ldots, \sigma_k > 0$, but $\sigma_i = 0$ for all $i > k$.
  - Therefore in the SVD, $A = \sum_{i=1}^k \sigma_i u_i v_i^\top$, the sum terminates at the k-th summand.
  - It follows that $\mathrm{range}(A) = \mathrm{span}(u_1, \ldots, u_k) = \mathrm{span}(U)$, which we verified in class. (Good exercise!)

- Therefore in this simple case we have $\operatorname{span}(AZ) = \operatorname{span}(U)$.
- So let X have orthonormal columns spanning $\operatorname{span}(AZ)$, hence spanning $\operatorname{span}(U)$.
- Then the ***Rayleigh-Ritz procedure*** applied to the matrix $AA^\top$ using the subspace $\operatorname{span}(X)$ will furnish the top k left singular vectors (and top k singular values).
    - **Notation caution:** ***here X is playing the role of V in our earlier discussion of subspace iteration!*** (Now the notation V is reserved for the right singular vectors.)
    - Concretely, here we must form the $k \times k$ matrix
    $$B := X^\top AA^\top X = (X^\top A)(X^\top A)^\top,$$
    which we diagonalize via spectral theorem as $B = \tilde{U}\tilde{\Lambda}\tilde{U}^\top$ (with eigenvalues ordered non-increasingly).
    - Then recover $U = X\tilde{U}$, $\Sigma = \sqrt{\tilde{\Lambda}}$.
- Similar procedure applied to $A^\top$ can be used to deduce V because $\operatorname{rank}(A^\top) = \operatorname{rank}(A) = k$ as well:
    - Let the columns of $W = [w_1, \ldots, w_k] \in \mathbb{R}^{m\times k}$ be random.
    - Form $A^\top W$ and orthonormalize the columns to produce Y.
    - Form $C := Y^\top A^\top AY = (AY)^\top(AY)$.
    - Diagonalize via spectral theorem as $C = \tilde{V}\tilde{\Lambda}\tilde{V}^\top$ (with eigenvalues ordered non-increasingly).
    - Recover $V = Y\tilde{V}$.
- *But there is a way to do both left and right steps at once*!
    - Observe that $XX^\top A = A$.
        - The reason is that $P := XX^\top$ is the orthogonal projection matrix onto $\operatorname{span}(X)$, which is the same as $\operatorname{range}(A)$, hence $PAx = Ax$ for all x, i.e., $PA = A$.
    - Likewise $YY^\top A^\top = A^\top$, or taking the transpose $AYY^\top = A$.
    - Therefore
    $$A = XX^\top A = XX^\top(AYY^\top) = X(X^\top AY)Y^\top.$$
- Neater / more compact presentation:
    - Let the columns of $Z = [z_1, \ldots, z_k] \in \mathbb{R}^{n\times k}$ and $W = [w_1, \ldots, w_k] \in \mathbb{R}^{m\times k}$ be random.
    - Form AZ and $A^\top W$ and orthonormalize their columns to produce X andy Y, respectively.
    - Form $\tilde{A} := X^\top AY$ (not symmetric!) and perform a ***full SVD*** to obtain $\tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$, in which all factors are $k \times k$.
    - Then recover $U = X\tilde{U}$, $V = Y\tilde{V}$, and $\Sigma = \tilde{\Sigma}$.
- Why does this give the same result?
    - Note that
    $$\tilde{A}\tilde{A}^\top = X^\top(AYY^\top)A^\top X = X^\top AA^\top X = B,$$
    so the left singular vectors of $\tilde{A}$ are the eigenvectors of B.
        - Hence our two recipes for producing $\tilde{U}$ coincide.
    - Meanwhile
    $$\tilde{A}^\top\tilde{A} = Y^\top A^\top(XX^\top A)Y = Y^\top A^\top AY = C,$$
    so the right singular vectors of $\tilde{A}$ are the eigenvectors of C.
        - Hence our two recipes for producing $\tilde{V}$ coincide.

40.2 Beyond rank k

- The more general algorithm is similar, except that we allow ourselves to multiply by $\tilde{k} \geq k$ random vectors, where $\tilde{k}$ is a parameter that must be set.
  - By taking $Z \in \mathbb{R}^{n \times \tilde{k}}$ and $W \in \mathbb{R}^{m \times \tilde{k}}$ By taking $\tilde{k}$ sufficiently large, we may still hope that $\operatorname{span}(AZ)$ and $\operatorname{span}(A^\top W)$ *approximately* contain the ranges of A and $A^\top$, respectively.
  - Therefore by defining $X = \operatorname{orth}(AZ) \in \mathbb{R}^{m \times \tilde{k}}$ and $Y = \operatorname{orth}(A^\top W) \in \mathbb{R}^{n \times \tilde{k}}$ as above, we can still hope for an approximation

$$A \approx XX^\top A \approx XX^\top AYY^\top = X\tilde{A}Y^\top,$$

    where $\tilde{A} = X^\top AY \in \mathbb{R}^{\tilde{k} \times \tilde{k}}$ is now possibly enlarged.
- This intuition is justified by the following technical result, which is outside the scope of the course. Specifically we now assume that X and Y have independent and identically distributed standard normal entries. (They can be instantiated in Matlab with `randn`.)

Theorem 1. *Let $\delta > 0$ be a failure probability. Let $A \in \mathbb{R}^{m \times n}$, and let $X = \operatorname{orth}(AZ)$, where $Z \in \mathbb{R}^{n \times \tilde{k}}$ has independent and identically distributed standard normal entries. There exists a universal constant C such that if $\tilde{k} \geq C\,(k + \log(1/\delta))$, then with probability at least $1 - \delta$*

$$\|XX^\top A\|_{\mathrm{F}} \leq 2 \sum_{i>k} \sigma_i^2,$$

where σ_i denote the singular values of A.

- Therefore if A has a 'numerical rank' of k, in the sense that $\sum_{i>k} \sigma_i^2$ is negligibly small, then by taking $\tilde{k}$ to be a few times as large as k, the randomized SVD will be guaranteed to work accurately with high probability.

40.3 Actual algorithm

- Given the ability to perform matvecs by A and $A^\top$, a target rank k, and a number of random vectors $\tilde{k} \geq k$:
  - Let the columns of $Z \in \mathbb{R}^{n \times \tilde{k}}$ and $W \in \mathbb{R}^{m \times \tilde{k}}$ have independent and identically distributed standard normal entries.
  - Form AZ and $A^\top W$ and orthonormalize their columns to produce X and Y, respectively.
  - Form $\tilde{A} := X^\top AY \in \mathbb{R}^{\tilde{k} \times \tilde{k}}$ (not symmetric!) and perform a ***full SVD*** to obtain $\tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$, in which all factors are $k \times k$.
  - Then recover $\check{U} = [X\tilde{U}]_{:,1:k}$, $\check{V} = [Y\tilde{V}]_{:,1:k}$, and $\check{\Sigma} = [\tilde{\Sigma}]_{1:k,1:k}$.
- Then $A \approx \check{U}\check{\Sigma}\check{V}^\top$ forms an approximate rank-k truncated SVD for A.

41 Spectral clustering

41.1 Constructing a weighted graph from data

- Suppose we are given data $x_1, \ldots, x_n \in \mathbb{R}^m$. We will use it to create a weighted graph.

- Given also a ***similarity kernel*** $K(x,y) \geq 0$ for $x, y \in \mathbb{R}^m$, satisfying $K(x,y) = K(y,x)$ for all x, y.
  - Popular choice: Gaussian similarity kernel with width parameter $\sigma > 0$:

$$K(x,y) = e^{-\frac{1}{2\sigma^2}\|x-y\|_2^2}.$$

- The kernel can be used to form a ***weighted adjacency matrix*** $A \in \mathbb{R}^{n\times n}$ via

$$A_{ij} = K(x_i, x_j),$$

  which is in particular symmetric, i.e., $A^\top = A$.

41.2 Graph Laplacians

- The weighted adjacency matrix can then be used to form the corresponding ***graph Laplacian*** $L \in \mathbb{R}^{n\times n}$, which is also a symmetric matrix defined by

$$L = D - A,$$

  where D is the diagonal matrix with $D_{ii} = \sum_{j=1}^n A_{ij}$, or for short $D = \mathrm{diag}(A\mathbf{1}_n)$.
  - Note that entrywise $L_{ij} = \delta_{ij} D_{ii} - A_{ij}$.
- ***Aside***: part of the intuition for the graph Laplacian comes from considering the ***left normalized graph Laplacian***

$$\tilde{L} = D^{-1}L = I - D^{-1}A,$$

  which is ***not symmetric***!
  - Entrywise we have $\tilde{L}_{ij} = \delta_{ij} - \frac{A_{ij}}{D_{ii}}$.
  - Then note that for any $v \in \mathbb{R}^n$,

$$[\tilde{L}v]_i = v_i - \frac{\sum_{j=1}^n A_{ij} v_j}{\sum_{j=1}^n A_{ij}}.$$

  - Therefore if we think of v as defining a 'function on the graph' via $i \mapsto v_i$, then $\tilde{L}v$ corresponds to the function that measures the difference between the value of v at a point and a weighted average of the values of its neighbors.
- Two facts about L (proofs from class are left here as exercises):
  - **Fact 1:** $L\mathbf{1}_n = 0$.
  - **Fact 2:** L is diagonally dominant, i.e., $|L_{ii}| \geq \sum_{j\neq i} |L_{ij}|$ for all $i = 1, \ldots, n$.
- Since L is diagonally dominant with nonnegative diagonal, a third fact follows (proof omitted)—
  - **Fact 3:** L is positive semidefinite (i.e., all eigenvalues are nonnegative).
- In fact spectral embedding / clustering is usually achieved by the ***symmetric normalized graph Laplacian***

$$\hat{L} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2},$$

  with entries

$$\hat{L}_{ij} = \delta_{ij} - \frac{A_{ij}}{\sqrt{D_{ii} D_{jj}}}.$$

- We conclude our discussion with one more fact
- **Fact 4:** The eigenvalues of $\hat{L}$ all lie in $[0,2]$.
  - *Proof* [omitted from lecture]:
    - In fact $\hat{L}$ is positive semidefinite, due to the construction as $\hat{L} = D^{-1/2} L D^{-1/2}$ and the fact that L is positive semidefinite. Hence we know the eigenvalues of $\hat{L}$ are all nonnegative.
    - Note that $\hat{L}$ and $\tilde{L}$ are similar via $\tilde{L} = D^{-1/2} \hat{L} D^{1/2}$, hence share the same eigenvalues.
    - So it would suffice to show that the spectral radius $\rho(\tilde{L}) \leq 2$.
    - For this it in turn suffices to show that $\|\tilde{L}\|_\infty \leq 2$.
    - But we can bound

$$\begin{aligned} \sum_{j=1}^{n} |\tilde{L}_{ij}| &= \sum_{j=1}^{n} \left| \delta_{ij} - \frac{A_{ij}}{D_{ii}} \right| \\ &\leq \sum_{j=1}^{n} \delta_{ij} + \frac{1}{D_{ii}} \sum_{j=1}^{n} A_{ij} \\ &= 1 + \frac{1}{D_{ii}} D_{ii} \\ &= 2, \end{aligned}$$

      so the 1-norms of all the rows of $\tilde{L}$ are bounded by 2, meaning that indeed $\|\tilde{L}\|_\infty \leq 2$, as desired.

41.3 Spectral embedding

- Spectral embedding is a type of graph embedding, which defines a map $\Phi : \{1, \ldots, n\} \to \mathbb{R}^d$ so that $\Phi(i)$ is the image of the i-th graph vertex (coming from the i-th data point) under the embedding.
- After embedding, we can define a new dataset $y_1, \ldots, y_n \in \mathbb{R}^d$ by setting $y_i = \Phi(i)$.
- To define the embedding, let the columns of $U = [u_1, \ldots, u_d] \in \mathbb{R}^{n \times d}$ be the eigenvectors corresponding to the d ***lowest*** eigenvalues of $\hat{L}$.
  - These can be computed by subspace iteration, applied to the matrix $2 - \hat{L}$, due to **Fact 4** proved above.
- Then set $\Phi(i) = [U_{i,:}]^\top \in \mathbb{R}^d$ to be the i-th row of U.

41.4 Clustering

- Given a dataset $x_1, \ldots, x_n \in \mathbb{R}^m$, we will describe how to partition it into k clusters.
  - ***Note carefully***: the algorithm we describe can be applied either to the original dataset $x_1, \ldots, x_m \in \mathbb{R}^m$ from the beginning of this section, or to the spectrally embedded dataset $y_1, \ldots, y_n \in \mathbb{R}^d$ obtained via the spectral embedding Φ. ***The results will differ!***
- The standard algorithm for achieving such a clustering is called ***k-means clustering***.
  - In k-means clustering, k 'cluster centroids,' $z_1, \ldots, z_k \in \mathbb{R}^m$ are defined.

  - Moreover, a cluster assignment map $c : \{1, \dots, n\} \to \{1, \dots, k\}$ is also defined. For every data point index i, the value $c(i)$ indicates the index of the cluster to which it has been assigned.

- k-means clustering seeks to optimize the following objective

$$F(z_1, \dots, z_k, c) = \sum_{j=1}^{k} \sum_{i\,:\,c(i)=j} \|x_i - z_j\|_2^2,$$

  over the cluster centroids $z_1, \dots, z_k$ ***as well as*** the cluster assignment map c itself!

- This is usually achieved by ***Lloyd's algorithm***, an iterative algorithm, which updates the cluster centroids and the cluster assignment map by alternating optimization:

  1. For fixed cluster assignment c, let $z_1, \dots, z_k$ be chosen to optimize the objective.
  2. For fixed cluster centroids $z_1, \dots, z_k$, let the cluster assignment be chosen to optimize the objective.

- In fact both steps can be achieved quite simply in closed form:

  1. For $j = 1, \dots, k$, set

$$z_j \leftarrow \frac{\sum_{i\,:\,c(i)=j} x_i}{\#\{i \,:\, c(i) = j\}}$$

     to be the mean of the data points assigned to the j-th cluster.

  2. For $i = 1, \dots, n$, assign the i-th data point to the cluster

$$c(i) = \operatorname*{argmin}_{j=1,\dots,k} \|x_i - z_j\|_2^2$$

     whose cluster centroid is closest.

- Usually a random initial guess for the cluster centroids is chosen, but note that ***there is no guarantee of convergence to a global optimum***!

  - In fact the algorithm can frequently get stuck, and many random restarts can be used.
  - The algorithm ***is*** however guaranteed to never increase the objective.

Part IV
Nonlinear equations and optimization

42 General nonlinear equations

- A general system of n ***nonlinear*** equations in n unknowns can be phrased as solving

$$F(x) = 0,$$

  where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a given function.

  - In general, there could exist any number (including zero) of solutions.

- We will focus on iterative methods for finding ***some*** solution (usually the solution 'closest' to some initial guess), provided that it exists.
- It is useful to think of

$$F(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{pmatrix}$$

as a vector of ***coordinate functions*** $f_i : \mathbb{R}^n \to \mathbb{R}$, which are each scalar-valued.

- Examples include:
  - Linear equations, which are furnished by the case $F(x) = Ax - b$.
    - We have already discussed linear equations a fair amount!
    - Later (Newton's method) we will see how the solution of linear equations can help with the solution of general nonlinear equations.
  - Optimization of a function $u : \mathbb{R}^n \to \mathbb{R}$ yields a system of nonlinear equations for the critical points

$$\nabla u(x) = 0,$$

  so in this case we view $F = \nabla u$.
    - We will return to this setting a few times below.
    - In particular, the best guarantees can be made in the case where u is a strictly convex function, meaning that the Hessian matrix $\nabla^2 u(x) = \left(\frac{\partial^2 u}{\partial x_i \partial x_j}(x) \right)$ is a positive definite matrix for all x. (We will return to this.)
  - Nonlinear equations arise in countless settings, from implicit time-stepping methods for nonlinear ODEs, to the discretization of nonlinear PDEs, and beyond....

43 Fixed-point iteration

43.1 Motivation

- **Definition:** Given a function $G : \mathbb{R}^n \to \mathbb{R}^n$, we call x a ***fixed point*** of G if $G(x) = x$.
- The idea of fixed-point iteration starts with finding a function G such that the fixed points of G are precisely the solutions of $F(x) = 0$.
  - ***One way*** to produce such a G is, given arbitrary $\varepsilon \neq 0$, to take

$$G(x) := x - \varepsilon F(x).$$

  - Then note that

$$G(x) = x \iff x - \varepsilon F(x) = x \iff F(x) = 0,$$

  as desired.
  - Although any choice of $\varepsilon \neq 0$ 'works' in the sense we have described, different choices may yield convergent or divergent fixed-point iterations, to be introduced below.
- Given such G, ***fixed-point iteration*** refers to the sequence $x^{(k)}$ produced by recursively defining

$$x^{(k+1)} = G(x^{(k)}), \quad k = 0, 1, 2, \ldots,$$

where $x^{(0)} \in \mathbb{R}^n$ is some initial guess. Alternatively we can think of

$$x^{(k)} = \underbrace{G \circ \cdots \circ G}_{k \text{ times}}(x^{(0)}).$$

- To proceed we will need to assume that our function G is ***continuous***. For completeness we provide a definition in the multivariate case.
  - **Definition:** we say that a function $G : \mathbb{R}^n \to \mathbb{R}^n$ is ***continuous*** if for all $x \in \mathbb{R}^n$, and all sequences $\{x^{(k)}\}$ with $\lim_{k\to\infty} x^{(k)} = x$, we have $\lim_{k\to\infty} G(x^{(k)}) = G(x)$.
- **Idea:** hope that fixed-point iteration converges. **Claim:** If it does, it furnishes a fixed point as its limit (assuming that G is continuous).
  - **Proof of claim:** Let $x = \lim_{k\to\infty} x^{(k)}$, which is assumed here to exist. Simply take the limit on both sides of
$$x^{(k+1)} = G(x^{(k)})$$
as $k \to \infty$ and use the continuity of G to obtain the equation $x = G(x)$, which means precisely that the limit x is a fixed point, as desired.

43.2 Banach fixed-point theorem

- When does fixed point iteration converge?
- A sufficient condition is provided by the Banach fixed-point theorem. To prove it, we will need to make a technical definition and recall one point of mathematical analysis. In the following discussion we make some consistent choice of norm $\|\cdot\|$ on $\mathbb{R}^n$ and let $\|\cdot\|$ also denote the corresponding natural norm on matrix in $\mathbb{R}^{n\times n}$.
  - **Definition:** a sequence $\{x^{(k)}\}$ is ***Cauchy*** if for every $\varepsilon > 0$, there exists K such that for all $k, l \geq K$, it holds that $\|x_k - x_l\| \leq \varepsilon$.
  - **Fact:** Any Cauchy sequence converges to some limit point.
    - This more generally is true of Cauchy sequences in complete metric spaces. ($\mathbb{R}^n$, equipped with any norm, is a complete metric space.)
- **Definition:** a map $G : \mathbb{R}^n \to \mathbb{R}^n$ is called a ***contraction map*** (with parameter $\alpha \in [0, 1)$) if for all $x, y \in \mathbb{R}^n$,
$$\|G(x) - G(y)\| \leq \alpha \|x - y\|.$$

Theorem (Banach fixed-point)**.** *Suppose that G is a contraction map with parameter $\alpha \in [0, 1)$. Let $x^{(0)} \in \mathbb{R}^n$ and define the sequence $\{x^{(k)}\}$ recursively via $x^{(k+1)} = G(x^{(k)})$ for $k = 0, 1, 2, \ldots$. Then* $\lim_{k\to\infty} x^{(k)} = x^{(\star)}$ *for some limit point $x^{(\star)} \in \mathbb{R}^n$, and moreover*
$$\|x^{(k)} - x^{(\star)}\| \leq \frac{\alpha^k}{1-\alpha} \|x^{(1)} - x^{(0)}\|.$$

- Not only does the theorem guarantee that fixed-point iteration converges, but also it provides us with a rate of convergence.
  - The error decays exponentially in k with rate $\alpha \in [0, 1)$.
  - In numerical analysis, we actually call this ***linear convergence*** because the log of the error converges linearly with slope $\log \alpha < 0$.
  - One interpretation of linear convergence is that each digit of accuracy costs as much to obtain as the last one.
  - We will see later that Newton's method achieves ***superlinear convergence***, in which each digit of accuracy is (asymptotically) cheaper to obtain that the last one.

- **Proof of theorem:**
  - First we want to show that the sequence is Cauchy.
  - For $k \leq l$, we can bound

    $$\|x^{(k)} - x^{(l)}\| \leq \sum_{i=k}^{l-1} \|x^{(i)} - x^{(i+1)}\|$$

    using a telescoping sum and the triangle inequality.
  - For $j \geq 0$, let G^j denote the j-fold composition $G \circ \cdots \circ G$.
    * **Claim:** G^j is a contraction map with parameter α^j.
    * **Proof of claim:** observe that for $j \geq 1$,

      $$\begin{aligned}\|G^j(x) - G^j(y)\| &= \|G(G^{j-1}(x)) - G(G^{j-1}(y))\| \\ &\leq \alpha \|G^{j-1}(x) - G^{j-1}(y)\|,\end{aligned}$$

      where we have used the contraction map property of G in the last step. Repeating this argument j times yields the claim.
  - Then

    $$\|x^{(i)} - x^{(i+1)}\| = \|G^i(x^{(0)}) - G^i(G(x^{0}))\| \leq \alpha^i \|x^{(0)} - G(x^{(0)})\| = \alpha^i \|x^{(0)} - x^{(1)}\|.$$

  - Plugging into our bound yields

    $$\|x^{(k)} - x^{(l)}\| \leq \left(\sum_{i=k}^{l-1} \alpha^i \right) \|x^{(0)} - x^{(1)}\| \leq \left(\sum_{i=k}^{\infty} \alpha^i \right) \|x^{(0)} - x^{(1)}\|.$$

  - Performing the infinite sum yields

    $$\|x^{(k)} - x^{(l)}\| \leq \frac{\alpha^k}{1-\alpha} \|x^{(0)} - x^{(1)}\|, \quad (\star)$$

    and recall that this holds for arbitrary $k \leq l$. This fact implies that the sequence is Cauchy. (Exercise, done in lecture: fill in this argument.)
  - Therefore the sequence has some limit point $x^{(\star)} = \lim_{k\to\infty} x^{(k)}$.
  - But then take the limit of the inequality $(\star)$ as $l \to \infty$ and use the fact that the norm is continuous to obtain

    $$\frac{\alpha^k}{1-\alpha} \|x^{(0)} - x^{(1)}\| \geq \lim_{l\to\infty} \|x^{(k)} - x^{(l)}\| = \|x^{(k)} - x^{(\star)}\|,$$

    and the proof is complete.

43.3 Sufficient condition for contraction map

- There is a useful sufficient condition for a map to be a contraction map, phrased in terms of the first partial derivatives. To state it, we will need to define the Jacobian of a vector-valued function.

- **Definition:** For a function $G : \mathbb{R}^n \to \mathbb{R}^n$ with coordinate functions $g_1, \ldots, g_n$, the ***Jacobian*** DG is an $n \times n$ matrix-valued function on $\mathbb{R}^n$ defined by

  $$[DG(x)]_{ij} = \frac{\partial g_i}{\partial x_j}(x).$$

Theorem. *Suppose that there exists $\alpha \in [0,1)$ such that $\|DG(x)\| \leq \alpha$ for all $x \in \mathbb{R}^n$. Then G is a contraction map with parameter α.*

- **Proof:**
  - Let $x, y \in \mathbb{R}^n$. We want to bound $\|G(x) - G(y)\|$
  - We will use the fundamental theorem of calculus. Define the parametric curve $z(t) = (1-t)x + ty$ for $t \in [0,1]$, so $z(0) = x$ and $z(1) = y$.
  - Then

$$\begin{aligned} G(y) - G(x) &= G(z(1)) - G(z(0)) \\ &= \int_0^1 \frac{d}{dt} G(z(t))\, dt \\ &= \int_0^1 DG(z(t))\, z'(t)\, dt \\ &= \int_0^1 DG(z(t))\, [y-x]\, dt \end{aligned}$$

    where we have used the chain rule.
  - Then take norms and apply the triangle inequality through the integral to obtain

$$\begin{aligned} \|G(y) - G(x)\| &\leq \int_0^1 \|DG(z(t))\, [y-x]\|\, dt \\ &\leq \int_0^1 \|DG(z(t))\|\, \|y-x\|\, dt \\ &\leq \int_0^1 \alpha\, \|y-x\|\, dt \\ &= \alpha \|y-x\|, \end{aligned}$$

    which completes the proof.

43.4 Convergence of gradient descent for convex optimization

- Consider the case of optimization of the function $u : \mathbb{R}^n \to \mathbb{R}$, where we took $F(x) = \nabla u(x)$. Moreover, we will consider fixed-point iteration with $G(x) = x - \varepsilon \nabla u(x)$, where ε is a ***step size*** parameter whose role we shall analyze.
- **Definition:** We say that $u \in C^2(\mathbb{R}^n)$ (meaning that u has continuous second partial derivatives) is ***strictly convex*** if the ***Hessian matrix*** $\nabla^2 u(x) := \left(\frac{\partial^2 u}{\partial x_i \partial x_j}(x) \right)$ is positive definite for all x.
  - The C^2 assumption guarantees that the Hessian exists and moreover is a symmetric matrix (since mixed partial derivatives commute in this case).
- We will assume in particular that u is strictly convex, which means equivalently that the eigenvalues of $\nabla^2 u(x)$ all lie in $(0, \infty)$.
  - Any critical point x satisfying $\nabla u(x) = 0$ of a strictly convex function must be the global optimizer.

  - We will make an even more detailed assumption.

- **Assumption:** Assume that there exist $c, C > 0$ such that the eigenvalues of $\nabla^2 u(x)$ lie in $[c, C]$ for all x.
  - As we shall see, it is the ***ratio*** C/c that really matters, and this can informally be called the ***condition number*** of the optimization problem, motivated by the fact that the condition number of a positive definite matrix (with respect to the 2-norm) is the ratio of the largest to smallest eigenvalue.

- Fixed-point iteration with $G(x) = x - \varepsilon \nabla u(x)$ concretely yields the update rule

$$x^{(k+1)} = x^{(k)} - \varepsilon \nabla u(x^{(k)}).$$

  This is called ***gradient descent*** with step size $\varepsilon > 0$.

Theorem. *For u satisfying our assumption, gradient descent with step size $\varepsilon = \frac{1}{C}$ converges to the global optimizer $x^{(\star)}$ of u. Moreover, the gradient descent update is a contraction map with parameter $1 - \frac{c}{C}$, so by the Banach fixed-point theorem we have*

$$\|x^{(k)} - x^{(\star)}\|_2 \leq \left(1 - \frac{c}{C}\right)^k \cdot \frac{C}{c} \|x^{(1)} - x^{(0)}\|_2$$

- **Proof:**
  - Observe (exercise, done in lecture) that with $G(x) = x - \varepsilon \nabla u(x)$,

$$DG(x) = I_n - \varepsilon \nabla^2 u(x).$$

  - Therefore if the eigenvalues of $\nabla^2 u(x)$ always lie in $[c, C]$, then the eigenvalues of $DG(x)$ always lie in

$$[1 - \varepsilon C, 1 - \varepsilon c] = [0, 1 - c/C],$$

    where we have taken $\varepsilon = 1/C$ as in the statement of the theorem.
  - Since $DG(x)$ is a symmetric matrix, $\|DG(x)\|_2 = \rho(DG(x))$, and moreover we know that $\rho(DG(x)) \leq 1 - \frac{c}{C}$, so $\|DG(x)\|_2 \leq 1 - \frac{c}{C}$.
  - It follows from our last theorem that G is a contraction map with parameter $1 - \frac{c}{C}$, which completes the proof.

- Observe that if the 'condition number' C/c of our problem is large, then the convergence of gradient descent will be slow!

44 Newton's method

44.1 Motivation and algorithm

- Again we are interested in solving $F(x) = 0$.
- **Idea:** alternate the following two steps
  - Linearize the equations near our current guess.
  - Solve linearized equations to obtain new guess.

- Let's be more concrete.... Suppose we have a guess $x^{(k)}$. Near $x^{(k)}$ we can expand $F(x)$ to first order using multivariate Taylor expansion (exercise, reviewed in lecture):

$$F(x) \approx F(x^{(k)}) + DF(x^{(k)})\,(x - x^{(k)}).$$

- Let us replace $F(x)$ with this approximation in the nonlinear equations $F(x) = 0$ to obtain the linearized equations

$$0 = F(x^{(k)}) + DF(x^{(k)})\,(x - x^{(k)}),$$

  which we can solve for x to obtain

$$x = x^{(k)} - DF(x^{(k)})^{-1}F(x^{(k)}).$$

- We set the solution of these equations to be the next guess $x^{(k+1)}$, thereby fixing the following iteration

$$x^{(k+1)} = x^{(k)} - DF(x^{(k)})^{-1}F(x^{(k)}),$$

  which is called ***Newton's method***.

  - Observe that in the scalar case $n = 1$, Newton's method reads as

$$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})}.$$

- In the setting of optimization where $u(x) = \nabla F(x)$, we have $DF(x) = \nabla^2 u(x)$, so Newton's method reads as

$$x^{(k+1)} = x^{(k)} - \nabla^2 u(x^{(k)})^{-1}\nabla u(x^{(k)}).$$

  - In the scalar case $n = 1$, this reads as

$$x^{(k+1)} = x^{(k)} - \frac{u'(x^{(k)})}{u''(x^{(k)})}.$$

- In practice we usually want to avoid literally inverting the matrix $DF(x^{(k)})$, so we instead think of the process of computing $y = DF(x^{(k)})^{-1}F(x^{(k)})$ as solving the linear system

$$DF(x^{(k)})y = F(x^{(k)})$$

  for y, where $x^{(k)}$ is fixed.

- This yields the pseudocode:

```
– Given $x^{(0)}$, etc.
– Set $x \leftarrow x^{(0)}$.
– For $i = 1, \dots, m$ (or until convergence)
    ∗ Set $A \leftarrow DF(x)$, $b \leftarrow F(x)$.
    ∗ Set $y$ to be the solution of $Ay = b$.
    ∗ Set $x \leftarrow x - y$.
```

44.2 Orders of convergence

- We want to discuss the convergence of Newton's method, but we take a brief digression to define the notion of the order of convergence of an iterative method.
- **Definition:** Consider an iterative method that, given some initial guess $x^{(0)}$, yields a sequence $\{x^{(k)}\}$ with $\lim_{k\to\infty} x^{(k)} = x^{(\star)}$. If there exists a positive integer q and $M > 0$ such that

$$\frac{\|x^{(k+1)} - x^{(\star)}\|}{\|x^{(k)} - x^{(\star)}\|^q} \leq M$$

  holds for all k, then the ***order of convergence*** of the method is q, and the corresponding ***rate of convergence*** is M.

  - Alternatively, we can rewrite the key inequality as

$$\|x^{(k+1)} - x^{(\star)}\| \leq M \|x^{(k)} - x^{(\star)}\|^q. \quad (*)$$

- The case $q = 1$ is called ***linear convergence*** (we have already seen this in the context of fixed-point iteration).
  - Need $M < 1$ for the inequality to guarantee convergence, no matter how close the initial guess $x^{(0)}$ is to $x^{(\star)}$. Hence $M < 1$ is often an implicit requirement in the definition if $q = 1$.
- The case $q > 1$ is called ***superlinear convergence***. The case $q = 2$ is called ***quadratic convergence***.
  - In fact, we do not need $M < 1$ for the inequality to guarantee convergence. As long as the initial guess is close enough, the inequality itself will guarantee convergence.
  - In practice an iterative method may converge slowly at first, until it reaches a zone of superlinear convergence. If one is interested only in the asymptotic convergence behavior, it is therefore reasonable to assume that the initial guess is sufficiently close to the true solution.
- The following theorem explains why the inequality $(*)$ implies 'extremely fast local convergence' when $q > 1$.

Theorem. *Suppose that an iterative method satisfies $(*)$ for $q > 1$. If $\|x^{(0)} - x^{(\star)}\| \leq \frac{1}{M^{1/(q-1)}}$, then the method converges, and moreover there exist $\alpha, C > 0$ such that*

$$\|x^{(k)} - x^{(\star)}\| \leq C e^{-\alpha q^k}.$$

Remark. In fact we can take $C = M^{-1/(q-1)}$, and $\alpha = -\log\left[M^{1/(q-1)} \|x^{(0)} - x^{(\star)}\|\right]$.

- **Proof:**
  - Define the error at the k-th iteration to be $E_k := \|x^{(k)} - x^{(\star)}\|$.
  - Then taking the log of the $(*)$ equation yields

$$\log E_{k+1} \leq b + q \log E_k,$$

    where $b := \log M$.

- Define a sequence recursively by $a_0 = \log E_0$, and
$$a_{k+1} = b + qa_k$$
for $k = 0, 1, 2, \ldots$.
- This sequence forms an upper bound for the sequence of log-errors, i.e., $a_k \geq \log E_k$.
  * This can be checked inductively: $a_0 = \log E_0$, and if the claim holds for iteration k, then
  $$a_{k+1} = b + qa_k \geq b + q \log E_k \geq \log E_{k+1},$$
  so it holds for iteration $k+1$ as well.
- Therefore we need only bound a_k. But a_k can be computed explicitly.
- Observe $a_1 = b + qa_0$, $a_2 = b + qb + q^2 a_0$, $a_3 = b + qb + q^2 b + q^3 a_0$, and in general
$$a_k = q^k a_0 + \left(\sum_{j=0}^{k-1} q^j \right) b.$$
- The sum can be done explicitly
$$\sum_{j=0}^{k-1} q^j = \frac{q^k - 1}{q - 1} = \frac{q^k}{q-1} - \frac{1}{q-1}$$
so
$$a_k = q^k \left[a_0 + \frac{b}{q-1} \right] - \frac{b}{q-1}.$$
- We want $a_k \to -\infty$ as $k \to \infty$, because then $\log E_k \to -\infty$ as well.
- Hence we need $\alpha := -\left[a_0 + \frac{b}{q-1} \right] > 0$.
- Recall $a_0 = \log E_0$ and $b = \log M$, so
$$\alpha = -\log \left[M^{1/(q-1)} E_0 \right],$$
which matches the definition from the remark.
- Then $\log E_k \leq a_k = -\alpha q^k - \frac{b}{q-1}$.
- Taking exponentials yields $E_k \leq C e^{-\alpha q^k}$, where $C = e^{-\frac{b}{q-1}}$, which completes the proof.
- To justify the remark, observe that $C = M^{-1/(q-1)}$.

44.3 Local convergence of Newton's method

- Now we finally study the convergence of Newton's method.

Theorem. *Suppose that the coordinate functions of F are all twice continuously differentiable. Let $x^{(\star)}$ be a solution of $F(x) = 0$, and suppose that $DF(x^{(\star)})$ is invertible. Given an initial guess $x^{(0)}$ sufficiently close to $x^{(\star)}$, Newton's method converges quadratically to $x^{(\star)}$.*

Remark. In the scalar case $n = 1$, the assumption on $DF(x^{(\star)})$ amounts to saying that $F'(x^{(\star)}) \neq 0$. This is a 'nondegeneracy' assumption. In the case of *convex* optimization where $F = \nabla u$, the nondegeneracy assumption says that all the eigenvalues of $\nabla^2 u(x^{(\star)})$ are positive. (If this is not the case, then the graph is 'completely flat' in some direction around the optimizer.)

- **Proof:** For simplicity, we will only prove the ***scalar case*** $n = 1$. For a general proof, see the 'Newton's method' section of

  `https://quantumtative.github.io/math228a_lecture_notes.pdf`

  - Since $F'(x^{(\star)}) \neq 0$, there exists some $\varepsilon > 0$ and some $\delta > 0$ such that for all x in the interval $I := (x^{(\star)} - \delta, x^{(\star)} + \delta)$, we have $|F'(x)| \geq \varepsilon$. We will show in particular that if we start our iteration close enough to $x^{(\star)}$, then we will never leave this interval.
  - Since F is twice continuously differentiable, there also exists some C such that $|F''(x)| \leq C$ for all $x \in I$.
  - We will assume that $|x^{(0)} - x^{(\star)}| \leq \min\left\{\delta, \frac{1}{M}\right\}$, where $M := \frac{C}{2\varepsilon}$, so in particular $x^{(0)} \in I$.
  - Assume inductively that $x^{(k)} \in I$, and we will in particular show that $x^{(k+1)} \in I$ and that the inequality $(*)$ from above holds with $q = 2$.
    * Perform a first-order Taylor expansion of F about the base point $x^{(k)}$ and evaluate at $x^{(\star)}$ to obtain

      $$F(x^{(\star)}) = F(x^{(k)}) + F'(x^{(k)})(x^{(\star)} - x^{(k)}) + R,$$

      where

      $$R = \frac{1}{2}F''(\xi^{(k)})(x^{(\star)} - x^{(k)})^2$$

      for some $\xi^{(k)}$ between $x^{(\star)}$ and $x^{(k)}$.
    * Since $F(x^{(\star)}) = 0$, this implies

      $$0 = F(x^{(k)}) + F'(x^{(k)})(x^{(\star)} - x^{(k)}) + R,$$

      which can be rearranged to obtain

      $$\begin{aligned} \frac{-R}{F'(x^{(k)})} &= \frac{F(x^{(k)})}{F'(x^{(k)})} + (x^{(\star)} - x^{(k)}) \\ &= x^{(\star)} - \left[x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})}\right] \\ &= x^{(\star)} - x^{(k+1)}, \end{aligned}$$

      where we have used the iteration formula for Newton's method.
    * Therefore

      $$|x^{(\star)} - x^{(k+1)}| \leq \frac{|R|}{|F'(x^{(k)})|} \leq \frac{C}{2\varepsilon}|x^{(\star)} - x^{(k)}|^2,$$

      or

      $$|x^{(\star)} - x^{(k+1)}| \leq M|x^{(\star)} - x^{(k)}|^2.$$

  - This establishes the inductive hypothesis, so $(*)$ holds for all k.
  - The preceding theorem then implies quadratic convergence.

45 Further topics

- Quasi-Newton methods, Anderson acceleration, Lagrangians and duality, augmented Lagrangian method, ADMM, LASSO regression