CS 6210: Matrix Computations

Derek Lim

Fall 2019

Instructor: David Bindel

Course Description: Stable and efficient algorithms for linear equations, least squares, and eigenvalue problems. Direct and iterative methods are considered. Julia and/or MATLAB are used extensively.

Textbooks: Golub and Van Loan, *Matrix Computations* and Demmel, *Applied Numerical Linear Algebra*

Webpage: `https://www.cs.cornell.edu/courses/cs6210/2019fa/index.html`

Lecture 1: Introduction (8/30/19)

Consider the following two algorithms to compute a matrix-vector product.

```
function y = matvec_row(A, x)
  y = zeros(m,1);
  for i = 1:m
    for j = 1:n
      y(i) = y(i) + A(i,j)*x(j)
    end
  end
```

```
function y = matvec_col(A, x)
  y = zeros(m,1);
  for j = 1:n
    for i = 1:m
      y(i) = y(i) + A(i,j)*x(j)
    end
  end
```

These two algorithms for matrix-vector multiplication may take different durations of time. The layout in Fortran, MATLAB, and Julia (column-major order) leads to the second algorithm, which processes column by column, to be preferred. Memory basics:

- 1D ordering
- Cache architecture

  - Temporal: access small sets of data and do lots of work before moving on
  - Spatial: access elements in order

Basic Linear Algebra Subroutines (BLAS)

- Level 1: $O(n)$ work on $O(n)$ data
  - e.g. dot products, adding/ scaling vectors
  - typically working with each element once, so temporal locality will not help much but spatial does
- Level 2: $O(n^2)$ work on $O(n^2)$ data
  - e.g. matrix-vector product
  - again, does not really make good use of temporal locality
- Level 3: $O(n^3)$ work on $O(n^2)$ data
  - e.g matrix multiplication
  - makes use of temporal locality

When doing more complex operations, we will often break matrices into blocks to make use of Level 3 BLAS and temporal locality.

```
function C = naive_matrix_multiply(A, B)
  [m,p] = size(A):
  [p,n] = size(B);
  C = zeros(m,n);
  for i in 1:m
    for j in 1:n
      for k in 1:p
        C(i,j) = C(i,j) + A(i,k)*B(k,j)
      end
    end
  end
```

Different permutations of the indices lead to different ways to view the operations:

$(i,j)(k)$ — $c_{i,j} = A(i,:) \cdot B(:,j)$ gives dot products of rows of A with columns of B

$(k)(i,j)$ — $C = \sum_k A(:,k)B(k,:)$ is a sum of outer products of columns of A with rows of B

Instead of going elementwise, we can break the matrices into blocks.

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \quad A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Then $C_{ij} = \sum_{k=1}^{2} A_{:,k} B_{k,:}$. Locality kicks in here and improves performance. The smaller blocks could make use of the cache.

Lecture 2: Types of Matrices (9/4)

Problem du jour: *Argue that the set of unit upper triangular $n \times n$ matrices forms a group with operation given by the standard matrix product.*

Each element of the group is invertible since each element has determinant 1. The product of two unit uper triangular matrices is unit upper triangular. The identity is unit upper triangular. All other properties follow from the properties of the matrix product.

In matrix computations/ numerical linear algebra, the choice of basis is an often important consideration when studying linear maps. Some properties such as symmetry, skew symmetry, and nonsingularity do not depend on the basis. Types of matrices that we will consider are

- General dense matrix
- Diagonal (basis dependent)
- Triangular (basis dependent)
  - Strictly triangular means 0 on the diagonal
- Permutation matrix: $P \in \{0,1\}^{n \times n}$, one 1 per row/ column
- Tridiagonal matrix: $b_{ij} = 0$ for $|i-j| > 1$
- Banded matrix: $b_{ij} = 0$ for $|i-j| > \beta$ where β is the **bandwith**
  - Can also have a lower and upper bandwith
- Hessenberg

Consider the problem of computing a matrix vector product $y = Dx$ where D is diagonal.

```
D = diag(d);
y = D*x; % O(n^2) time and space

% instead, for O(n) time and no additional space,
y = d .* x;
```

A **general sparse** matrix has "most" entries as zero, so we store only the nonzeros explicitly (we are leaving the term "most" up for interpretation). For instance, for the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 4 \end{bmatrix}$$

we could store it in the usual MATLAB/ Julia/ Python/ FORTRAN **column major order**:

$$1000 \quad 0210 \quad 0031 \quad 0014$$

Compressed sparse column (CSC) format is like the column major format, except we store only the nonzero entries along with their row indices, and column pointers. Column pointer j stores the

entry	1	2	1	3	1	4
row idx	1	2	3	3	4	4
column pointer	1	2	4	6	7	

index k such that nonzero entry k (in top to bottom, left to right order) is the topmost nonzero entry in column j.

A **data sparse** matrix is one that requires far fewer than n^2 parameters to describe it. Examples include:

- Low rank matrices
  - For instance, a rank 1 matrix $A \in \mathbb{C}^{n\times n}$ can be written $A = uv^T$ where $u, v \in \mathbb{C}^n$
- Toeplitz matrices
  - Of the form:
$$\begin{bmatrix} r_1 & r_2 & r_3 & \dots & r_n \\ c_2 & r_1 & r_2 & \dots & r_n \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ c_n & c_{n-1} & c_{n-2} & \dots & r_1 \end{bmatrix}$$
  - These are not generally low rank
  - There is an $O(n \log n)$ multiply by FFTs

An example of making use of the data sparsity of low rank matrices can again be found in matrix-vector multiplication:

```
A = u*v';
y = A*x; % Bad way to multiply, O(n^2) time and space

% Better O(n) time method:
y = u*(v'*x);
```

Consider the $n \times n$ matrices, which form an n^2 dimensional vector space. Also, consider the map $X \mapsto AX$, where $A \in \mathbb{C}^{n\times n}$. This need not be an $O(n^4)$ operation because of the matrix structure that we have.

For a matrix A, we define $\text{vec}(A)$ as the vector formed by listing the elements of the matrix in column major order.

The **Kronecker product** of two matrix $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{p\times q}$ is defined as

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \dots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \dots & a_{m,n}B \end{bmatrix} \in \mathbb{C}^{mp\times nq}$$

Lecture 3: Linear Algebra Review (9/6)

Problem du jour: *How much does it cost to add two sparse matrices (in compressed sparse column form)? What about adding $O(n)$ sparse matrices, each with a constant number of nonzeros (nnz)?*

$O(nnz_1 \cdot nnz_2)$ because of reindexing.

Linear Algebra Review

We typically denote abstract vector spaces as $\mathcal{V}$,$\mathcal{U}$,$\mathcal{W}$ (generally over $\mathbb{R}$ or $\mathbb{C}$). Some examples that we will use include:

- $\mathbb{R}^n$, $\mathbb{C}^n$
- $\mathcal{P}_d = \{\text{polynomials in one variable of degree at most } d\}$
- $L(\mathcal{V},\mathcal{W}) = \{\text{linear maps } \mathcal{V} \to \mathcal{W}\}$
- $\mathcal{V}^* = \{\text{linear functions } \mathcal{V} \to F\} = L(\mathcal{V}, F)$ for the field F associated with $\mathcal{V}$
- $C(\Omega) = \{\text{continuous functions } \Omega \to \mathbb{R}\}$

To illustrate the concept of bases, we take the example $\mathcal{P}_d$. An example of a basis is the power basis, which is $[1, x, x^2, \ldots, x^d]$. Another basis is given by the Chebyshev polynomials $[T_0(x), T_1(x), \ldots, T_d(x)]$, which can be defined recursively as

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{j+1}(x) &= 2xT_j(x) - T_{j-1}(x) \end{aligned}$$

The matrix $A = \begin{bmatrix} 1 & x & x^2 \end{bmatrix}$ represents a map $\mathbb{R}^3 \to \mathcal{P}_2$. For instance,

$$\begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = 1 + 2x + x^2$$

Let $\mathcal{V},\mathcal{W}$ vector spaces, and $\mathcal{A} : \mathcal{V} \to \mathcal{W}$ a linear map. Say $B_\mathcal{V}$ and $B_\mathcal{W}$ are bases for $\mathcal{V}$ and $\mathcal{W}$, respectively. $A = B_\mathcal{W}^{-1} \mathcal{A} B_\mathcal{V}$ is the matrix representation of $\mathcal{A}$ with respect to these choices of bases.

A norm is a map $\|\cdot\| : \mathcal{V} \to \mathbb{R}$ such that

- $\|\alpha v\| = |\alpha| \|v\|$ for $\alpha \in \mathbb{C}$
- $\|v\| \geq 0$, $\|v\| = 0 \iff v = 0$
- $\|v + w\| \leq \|v\| + \|w\|$

Examples in $\mathbb{R}^n$:

- $\|u\|_p = \left(\sum_{j=1}^n |v_j|^p \right)^{1/p}$ for $p >= 1$
- $\|v\|_\infty = \max_j |v_j|$

Consider norms over $\mathcal{P}_d$

- $\|p\|_{L^2[0,1]} = \sqrt{\int_0^1 |p(x)|^2 \, dx}$
- $\|p\|_{L^2[0,1]} = \int_0^1 |p(x)| \, dx$
- $\|p\|_{L^2[0,1]} = \sup_{x\in[0,1]} |p(x)|$

An inner product $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ (or $\mathbb{C}$) has properties

- $\langle \alpha v, w \rangle = \alpha \langle v, w \rangle$
- $\langle v_1 + v_2, w \rangle = \langle v_1, w \rangle + \langle v_2, w \rangle$
- $\langle v, w \rangle = \overline{\langle w, v \rangle}$
- $\langle v, v \rangle \geq 0$, is $0 \iff v = 0$

every inner product has an associated norm $\|v\| = \sqrt{\langle v, v \rangle}$.
Here are three (or four) possible meanings of a matrix. A can represent

- mapping from a space to itself
- mapping between two spaces
- mapping from two vectors to $\mathbb{R}$ (or $\mathbb{C}$)
  - bilinear form: $(x, y) \mapsto y^T A x$
  - sequilinear form: $(x, y) \mapsto y^* A x$
- mapping $\mathcal{V} \to \mathbb{R}$ that is pure quadratic
  - $x \mapsto x^T A x$

Lecture 4: Canonical Forms, Norms (9/9)

Problem du jour: *Prove the Cauchy-Schwarz inequality* $|\langle x, y \rangle| \leq \|x\|_2 \|y\|_2$ *without making use of a specific basis/ structure of* $\mathbb{R}^n$.

Proof.

$$\begin{aligned}
\|x+y\|_2^2 &\leq (\|x\|_2 + \|y\|_2)^2 \\
&\leq \|x\|_2^2 + \|y\|_2^2 + 2\|x\|_2\|y\|_2 \\
\|x+y\|_2^2 &= \langle x+y, x+y \rangle \\
&= \langle x, x \rangle + \langle y, y \rangle + 2\langle x, y \rangle \\
&= \|x\|_2^2 + \|y\|_2^2 + 2\langle x, y \rangle \\
\|x-y\|_2^2 &= \|x\|_2^2 + \|y\|_2^2 - 2\langle x, y \rangle \quad \text{likewise} \\
2|\langle x, y \rangle| &\leq 2\|x\|_2\|y\|_2
\end{aligned}$$

□

	General choice of basis	Orthonormal basis (bases)
Linear map $L : \mathcal{V} \to \mathcal{W}$	Rank/ nullity	Singular value decomposition
Linear operator $L : \mathcal{V} \to \mathcal{V}$	Jordan form	Schur form
Quadratic form $L : \mathcal{V} \to \mathbb{C}$	Sylvester intertia	Symmetric eigendecomposition

Table 1: Canonical forms

These canonical forms have the following shapes:

- Rank/ nullity: $\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$, $A = W^{-1}\mathcal{A}V$
- Singular value decomposition: $\begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix}$ $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$, $\Sigma = U^* A V$
- Jordan form: $\begin{bmatrix} J_{\lambda_1} & & \\ & \ddots & \\ & & J_{\lambda_k} \end{bmatrix}$, $J_\lambda = \begin{bmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \lambda & 1 \\ & & & \lambda \end{bmatrix}$
- Schur form: $T = \begin{bmatrix} t_{11} & t_{12} & & & t_{1n} \\ & \ddots & \ddots & & \\ & & t_{n-1,n-1} & t_{n-1,n} & \\ & & & t_{nn} & \end{bmatrix}$
- Sylvester inertia: $\begin{bmatrix} I & & \\ & 0 & \\ & & -I \end{bmatrix}$
- Symmetric eigenproblem: $\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$

The rank/ nullity, Jordan form, and Sylvester inertia are discontinuous with respect to pertubations to the linear operator. Thus, they are not typically useful to directly compute on a physical computer, as numerical error and noise in data could change these forms significantly.

Norms

Now, consider a space of linear maps $\mathcal{L}(\mathcal{V}, \mathcal{W})$. A norm $\|\cdot\|$ over this space is **consistent** if $\|AB\| \leq \|A\|\|B\|$.

Example 0.1 (Frobenius norm)**.**

$$\|A\|_F^2 = \sum_i \sum_j |a_{i,j}|^2$$

This is a consistent inner-product norm, induced by the inner-product.

$$\langle A, B\rangle = \sum_i \sum_j a_{ij}\bar{b}_{ij} = tr(B^*A)$$

The Frobenius norm is easy to compute, but may not always give us nice bounds.

Let $\mathcal{V}, \mathcal{W}$ be normed vector spaces. Then the **operator norm/ induced norm** $\|\cdot\|_{\mathcal{V},\mathcal{W}}$ is

$$\|A\|_{\mathcal{V},\mathcal{W}} = \sup_{v\neq 0} \frac{\|Av\|_{\mathcal{W}}}{\|v\|_{\mathcal{V}}}$$

this is a consistent norm.

Example 0.2.

- $\|A\|_1 = \sup_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1} = \max_j \sum_i |a_{ij}|$
- $\|A\|_\infty = \sup_{x\neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_i \sum_j |a_{ij}|$
- $\|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_1$

  To see this, consider $\varphi(x) = \sup_{x\neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2}$.

$$\varphi(x) = \frac{x^T A^T A x}{x^T x}$$
$$\ldots \text{calculus elided}$$
$$A^T A x = \frac{x^T A^T A x}{x^T x}$$

In this last derivation we use variational notation. $\delta[f(X)] = \frac{d}{ds}\,|_{s=0}\, f(X + s\delta X) = \frac{\partial}{\partial[\delta X]} f(X)$.

As an example of the utility of variational notation, say we want to compute $\delta[A^{-1}]$.

$$\begin{aligned}
\delta[A^{-1}A] &= \delta[I] = 0 \\
\delta[A^{-1}A] &= \delta[A^{-1}]A + A^{-1}[\delta A] \quad \text{product rule} \\
\delta[A^{-1}] &= -A^{-1}[\delta A]A^{-1}
\end{aligned}$$

Lecture 5: (9/11)

Problem du jour: *Suppose $Q(s)$ satisfies $Q^TQ = I$ and is differentiable with respect to s. Show $\frac{d}{ds}Q(s) = QS$ for some $S = -S^T$.*

Proof.

$$\begin{aligned}
\Big(\frac{dQ}{ds}\Big)^T Q + Q^T\Big(\frac{dQ}{ds}\Big) &= 0 \\
Q^T\Big(\frac{d}{ds}Q\Big) &= -\Big(\frac{d}{ds}Q\Big)^T Q \\
Q^T\Big(\frac{d}{ds}Q\Big) &= -\Big[Q^T\Big(\frac{d}{ds}Q\Big)\Big]^T \\
\frac{d}{ds}Q &= QS \quad \text{since } Q \text{ is orthogonal}
\end{aligned}$$

where $S = Q^T\Big(\frac{d}{ds}Q\Big)$. In variational notation, $\delta Q = QS$. □

Resuming from last time, we can use Lagrange multipliers to determine the form of the operator 2-norm.

$$\begin{aligned}
\|A\|_2^T &= \sup_{\|x\|_2^2=1} \|Ax\|_2^2 \\
\mathcal{L}(x,\lambda) &= \|Ax\|^2 - \lambda(\|x\| - 1) \\
\delta[\|Ax\|^2] &= \delta[x^T A^T A x] \\
&= 2\delta x^T A^T A x \\
\delta[\|x\|^2 - 1] &= \delta[x^T x - 1] \\
&= 2\delta x^T x \\
\delta\mathcal{L} &= 2\delta x^T[A^T A x - \lambda x] + \delta\lambda(\|x\|^2 - 1)
\end{aligned}$$

Note that we end up with an eigenvalue problem $A^T Ax = \lambda x$. Let $A = U\Sigma V^T$ be an svd, then $A^T A = V\Sigma^2 V^T$. Since V is orthogonal, $A^T A$ has the same eigenvalues as Σ^2. Thus, the solutions correspond to the squared singular values. Hence, $\|A\|_2 = \sigma_1(A)$.

The 2-norm is useful because it has many useful properties, but it is more difficult to compute than the 1-norm or ∞-norm.

Remark 1 (Orthogonal invariance)**.** As an example of a useful identity, it holds that the euclidean norm is orthogonally invariant, meaning that $\|Qx\|_2 = \|x\|_2$. This is because $\|Qx\|_2^2 = x^T Q^T Q x = x^T x = \|x\|_2^2$.

For the Frobenius norm, it holds that

$$\|A\|_F = \|QA\|_F = \Big\|AQ'\Big\|_F \quad \text{for any } Q, Q' \text{ orthogonal of proper dimension}$$

The operator 2-norm is also orthogonally invariant.

$$\begin{aligned}
\|QAU\|_2 &= \sup_{\|x\|_2=1} \|QAUx\|_2 \\
&= \sup_{\|x\|_2=1} \|AUx\|_2 \quad \text{orthogonal invariance of euclidean norm} \\
&= \sup_{\|z\|_2=1} \|Az\|_2 \\
&= \|A\|_2
\end{aligned}$$

Thus, another way to derive the form of $\|A\|_2$ is to note that for an svd $A = U\Sigma V^T$,

$$\left\|U\Sigma V^T\right\|_2 = \|\Sigma\|_2 = \sup_{\sum z_j^2 = 1} \sigma_j z_j = \sigma_1$$

We can also compute the 2-norm of A^{-1} painlessly. For $A = U\Sigma V^T$, we have $A^{-1} = V\Sigma^{-1}U^T$. Thus, $\left\|A^{-1}\right\|_2 = 1/\sigma_n$.

Recall if $|z| < 1$, then $\sum_{j=0}^{\infty} z^j = \frac{1}{1-z}$. Also, $\sum_{j=0}^{n} z^j = \frac{1-z^{j+1}}{1-z}$.

Analogously, $\sum_{j=0}^{\infty} A^j = (I-A)^{-1}$ if $\rho(A) < 1$. A weaker condition is for $\|A\| < 1$ for some consistent norm $\|\cdot\|$.

$$\begin{aligned}\left\|(A+E)^{-1} - A^{-1}\right\| &= \left\|(A^{-1} - A^{-1}EA^{-1}O(\|E\|^2)) - A^{-1}\right\| \\ &= \left\|A^{-1}EA^{-1}\right\| + O(\|E\|^2)\end{aligned}$$

Another way to do the same analysis is as follows:

$$\begin{aligned}(A+E)^{-1} &= (A(I + A^{-1}E))^{-1} \\ &= (I + A^{-1}E)^{-1}A^{-1} \\ \left\|(I + A^{-1}E)^{-1}A^{-1} - A^{-1}\right\| &= \left\|\sum_{j=0}^{\infty}(-A^{-1}E)^j A^{-1} - A^{-1}\right\| \quad \text{if } \left\|A^{-1}E\right\| < 1\end{aligned}$$

Using that

$$\left\|\sum X^j\right\| = \left\|(I-X)^{-1}\right\| \leq \sum \|X\|^j = \frac{1}{1-\|X\|}$$

we have an expression for the sensitivity of an inverse to perturbations.

Error

Say we want to compute x and get $\hat{x}$. The **absolute error** is $\|\hat{x} - x\|$. The **relative error** is $\frac{\|\hat{x}-x\|}{\|x\|}$. These are **forward error** measures.

Backward error measures the error by comparing the computed solution to the exact solution of a related problem. An example of backward error is an expression like $(A+E)\hat{x} = b$ for some small E.

Lecture 6: (9/13)

Problem du jour: *Show* $y^T Ax = tr(Axy^T)$.

Proof. $y^T Ax = tr(y^T Ax) = tr(Axy^T)$ □

Say we wish to compute $y = f(x)$. The **absolute backward error** when we compute $\hat{y} = f(\hat{x})$ is $|\hat{x} - x|$. The **relative backward error** is $\frac{|\hat{x}-x|}{|x|}$. The **condition number** relates backward to forward error.

$$\frac{\hat{y} - y}{y} \leq \kappa_{f(x)} \frac{\hat{x} - x}{x}$$

$$\kappa_{f(x)} = \lim_{x \to 0} \sup_{\|\hat{x}-x\| \leq \epsilon} \frac{\left|(f(\hat{x}) - f(x))/f(x)\right|}{|(\hat{x} - x)/x|}$$

Think of the condition number as follows (which holds when f is real and differentiable):

$$\begin{aligned} \kappa_{f(x)} &= \frac{\left|f'(x)(\hat{x} - x)/f(x)\right|}{|\hat{x} - x|/|x|} \\ &= \frac{|f'(x)||x|}{|f(x)|} \end{aligned}$$

The following are common sources of error in a computation:

- Error from measurements/ input
- Stochastic error (e.g. in Monte Carlo)
- Error due to termination of iterations
- Error due to floating point

Say we want $y = Ax$, and we compute $\hat{y} = (A + E)\epsilon$. Then the absolute backward error is $\|E\|$. The relative backward error is $\frac{\|E\|}{\|A\|}$. The absolute forward error is $\|Ex\| \leq \|E\|\|x\|$. The relative forward error is $\frac{\|Ex\|}{\|y\|} \leq \frac{\|E\|\|x\|}{\|y\|}$. We want an expression $\frac{\|\hat{y}-y\|}{\|y\|} \leq \kappa \frac{\|E\|}{\|A\|}$. We have that

$$\begin{aligned} \frac{\|\hat{y} - y\|}{\|y\|} &\leq \Big(\frac{\|A\|\|x\|}{\|y\|}\Big)\frac{\|E\|}{\|A\|} \\ &= \Big(\frac{\|A\|\|x\|}{\|Ax\|}\Big)\frac{\|E\|}{\|A\|} \end{aligned}$$

after some algebra, $\frac{\|A\|\|x\|}{\|Ax\|} = \|A\|\left\|A^{-1}\right\|$. To see this, note

$$\frac{\|A\|\left\|A^{-1}y\right\|}{\|y\|} \leq \|A\|\left\|A^{-1}\right\|$$

note that we have been assuming A invertible, since otherwise the definition of backward error does not apply.

Since we have for the spectral norm that $\inf \frac{\|Ax\|_2}{\|x\|_2} = \sigma_n(A)$, we have that $\|Ax\|_2 \geq \sigma_n(A)\|x\|$, so that $\frac{\|A\|\|x\|}{\|Ax\|} \leq \frac{\sigma_1(A)}{\sigma_n(A)}$.

Now, we consider floating point errors, using computation of a dot product as an example. We begin with the basic algorithm.

```
d=0
for i = 1:n
  d = d + x(i) * y(i)
end
```

The computer introduces error in the floating point representation. Due to rounding, we get:

$fl(a+b) = (a+b)(1+\delta)$. $|\delta| \leq \epsilon_{\text{machine}}$.

$fl(a*b) = (a*b)(1+\delta)$.

In the dot product, say we have $fl(x_i * y_i) = x_i y_i(1+\delta_i) = \hat{x}_i y_i$, where $\hat{x}_i = (1+\delta_i)$.

Thus, we have that $d_i = (d_{i-1} + x_i y_i(1+\delta_i))(1+\delta_i')$ (defining $d_0 = 0$), so that

$$\begin{aligned}
d_0 &= 0 \\
d_1 &= (0 + x_1 y_1(1+\delta_1))(1+\delta_1') \\
&= 0 + x_1 y_1(1+\delta_1)(1+\delta_1') \\
&\approx x_1 y_1(1+\delta_1+\delta_1') \quad \text{since } \delta_1\delta_2 \text{ very small} \\
d_2 &= (x_1 y_1(1+\delta_1+\delta_1') + x_2 y_2(1+\delta_2))(1+\delta_2') \\
&\approx (x_1 y_1)(1+\delta_1+\delta_1'+\delta_2') + x_2 y_2(1+\delta_2+\delta_2') \vdots \qquad \vdots \\
d_n &\approx x_1 y_1(1+\tilde{\delta}_1) + x_2 y_2(1+\tilde{\delta}_2) + \dots
\end{aligned}$$

where $\tilde{\delta}_1 \leq n\epsilon_{\text{mach}}$, $\tilde{\delta}_2 \leq (n-1)\epsilon_{\text{mach}}$ and so on.

Now, if we look at the backward error, say we have $d_n = \tilde{x}^T y$, where $\frac{|\tilde{x}_i - x_i|}{|x_i|} \leq n\epsilon_{\text{mach}}$, so that $\|\tilde{x} - x\|_\infty \leq n\epsilon_{\text{mach}}\|x\|_\infty$.

If we instead look at $d = y^T x$, where $\tilde{d} = y^T\tilde{x}$, then what we get depends on $\frac{|y^T x|}{\|y\|\|x\|}$.

Lecture 7: (9/16)

Binary floating point (IEEE style)

A normalized floating point number is of the form

$$(-1)^s(1.b_1 b_2 \dots b_d)_2 \cdot 2^E$$

Arithmetic on these numbers is the correct result properly rounded (if there is a tie, round so that the last digit is zero i.e. round to even). This is true for $+, -, \times, \div, \sqrt{}$. This means that

$$fl(a \circ b) = (a \circ b)(1+\delta) \quad \delta \leq \epsilon_{\text{machine}}$$

Note that we cannot represent zero just with a normalized floating point number. There are certain **exceptional representations**.

- Subnormal/ denormalized numbers are smaller than the smallest normalized numbers, and are of the form
$$(-1)^s(0.b_1 \dots b_d) \cdot 2^{E_{\min}}$$
Note that 0 is a subnormal number, and there is a positive 0 and a negative 0.

- Infinities: $\pm\infty$
- Not a Number (`NaN`)

With the exceptional representations, all floating point operations can return a value in the floating point system.

Cancellation

If a and b have opposite signs, and are within a factor of 2, then $fl(a+b) = a+b$ (no roundoff). However, we wish to analyze relative error. Consider $\hat{A} - \hat{B} = a(1+\delta_a) - b(1+\delta_b)$, where both inputs have some error. Then the relative error is of the form

$$\frac{\left|\hat{a} - \hat{b} - (a-b)\right|}{|a-b|} = \frac{|\delta_a a - \delta_b b|}{|a-b|}$$

Error is high when the δ have different sign, such as when the error is approximately of the following form

$$\frac{\delta|a+b|}{|a-b|}$$

Subtraction does not introduce any new error, but it does reveal the error already present in the inputs.

Consider the problem of finding the roots of $1 - 2bz + z^2$. These are of the form

$$\xi_\pm = b \pm \sqrt{b^2 - 1}$$

Note that the computation of ξ_- by this formula may face cancellation error. Note that $\xi_- = \frac{1}{\xi_+}$ since the last coefficient is 1. This formula is better for computing ξ_- when the quadratic formula faces cancellation error.

Unstable recurrences

Consider the problem of computing

$$E_n = \int_0^1 x^n e^{x-1}\, dx$$

One way of doing this is through a recurrence obtained by integration by parts.

$$E_0 = 1 - \frac{1}{e}$$
$$E_n = 1 - nE_{n-1} \quad n \geq 1$$

Suppose that there is error $\hat{E}_0 = E_0(1+\delta)$, since we cannot exactly represent e in the machine. Also, we then have $\hat{E}_n = 1 - n\hat{E}_{n-1}$.

$$d_0 = \hat{E}_0 - E_0 = \delta E_0$$
$$\hat{E}_n = 1 - n\hat{E}_{n-1}$$
$$E_n = 1 - nE_{n-1}$$
$$d_n = -nd_{n-1}$$

Thus, we see that the errors are of the form $\delta_n = (n!)\delta E_0$. Note that if we instead start from large n and compute the recurrence backwards, then the initial error of representation is supressed by an $n!$ factor instead of amplified.

Two other issues of floating point arithmetic are:

- Undetected underflow (note that overflow is generally easy to detect). This means that the $1+\delta$ model no longer holds, since the relative error of representation can be quite high (especially as we approach 0).
- `NaN`s and branches.

```
1       if x <= 0
2         disp("<=0")
3       else if x > 0
4         disp(">0")
5       else
6         disp('uh-oh');
7       end
```

Running this script causes "uh-oh" to display.

Example 0.3 (Geometric predicates)**.** We finally present a more positive example. An example of a geometric predicate is whether two points are on the same side of a line or on different sides. Say we have points C, D and two points that determine a line A, B. One way to solve this is by looking at

$$\det[B-A, D-A]$$
$$\det[B-A, C-A]$$

If these two determinants have the same sign, then predict that the points are on the same side. Otherwise, predict that they are on different signs.

Assume that all points are in $[1,2]^2$. The bad news is that we can compute the signs incorrectly if the usual formula is carried out in the input precision. The good news is that if the inputs are single precision and the intermediates are double precision, then the signs will be correct.

For a sketch of why this is, note that all of the entries of this matrix will we computed exactly.

$$\begin{bmatrix} x_B - x_A & x_D - xA \\ y_B - y_A & y_D - y_A \end{bmatrix}$$

Moreover, all products will be exactly computed since the double precision intermediates have enough space to hold the exact result of the multiplication of the single precision numbers.

Lecture 8: September 18

Problem du jour: $|z| < 1$, $S_n = \sum_{j=1}^{n} z^j = \frac{1-z^{n+1}}{1-z} \to \frac{1}{1-z}$.

Consider $\hat{S}_n$ computed by the recurrence

$$\hat{S}_0 = 0$$
$$\hat{S}_{k+1} = (z\hat{S}_k + 1)(1+\delta_k), \quad |\delta_k| < \epsilon$$

What can we say about the relative error?

Gaussian elimination/ LU factorization

For a matrix A, we seek a decomposition

$$PA = LU$$

where P is a permutation matrix, L is unit lower triangular, and U is upper triangular. We go through such a procedure to decompose an example matrix.

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 3 & 6 & 10 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & -6 & -11 \end{bmatrix}$$

Note that

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

commute, and their product is $\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$. Thus, the operations are independent, and the order that they are done does not matter.

A general **Gauss transformation** is of the form $I - \tau e_k^T$ where $\tau_j = 0$ for $j \leq k$. Continuing the above process for one more step, we get to

$$\begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus, we have $M_2 M_1 A = U$, where M_1 is the first transformation and M_2 is the second. We then have $M_2 M_1 Ax = Ux = M_2 M_1 b$.

Upper triangular linear systems are nice because they are easy to solve by backward substitution. Moreover, note that if we have a block upper triangular linear system, then block backward substitution can be used. Thus, for a linear system of the form

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

we can do block Gaussian elimination,

$$\begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix}$$

$S = D - CA^{-1}B$ is the **Schur complement**. Now, we can use block backward substitution to solve

$$\begin{bmatrix} A & B \\ 0 & S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g - CA^{-1}f \end{bmatrix}$$

```
% Solve Ax=b
% triangular reduction
for j = 1:n-1
  tau = A(j+1:n, j) / A(j,j);
  A(j+1:n, j:n) = A(j+1:n, j:n) - tau * A(j, j:n);
  b(j+1:n) -= tau*b(j);
end

% back substitution
for i = n:-1:1
  x(i) = (b(i) - A(i, i+1:n)*x(i+1:n))/A(i,i)
end
```

The number of operations for the triangular reduction is approximately $\sum_{j=1}^{n-1} j^2 \approx \int_0^n x^2\,dx = \frac{1}{3}n^3$.

```
% Solve Ax=b, version 2
% Stores intermediate multipliers tau in A's lower triangle
for j = 1:n-1
  A(j+1:n,j) = A(j+1:n,j)/A(j,j)
  A(j+1:n,j+1:n) -= A(j+1:n,j)*A(j,j+1:n)
end

for j = 1:n-1 % implicitly solve with L
  b(j+1:n) = b(j+1:n) - A(j+1:n) * b(j)
end
for i = n:-1:1
  %backsub
end
```

Thus, for solving multiple linear systems with the same A but different right hand side, we only dod the $O(n^3)$ decomposition once.

Note that our decomposition is $M_{n-1}\cdots M_1 A = U$ where the M_k are unit lower triangular. Since such matrices form a group, we have that $M_{n-1}\cdots M_1$ is unit lower triangular, and hence $A = LU = M_1^{-1}\cdots M_{n-1}^{-1}U$, where L is again unit lower triangular due to the group property. L has entries which are the multipliers τ. Thus, the loop in the algorithm above that transforms b before backsubstitution is an implicit solve with L.

Lastly, a Gauss transformation is geometrically a shear transformation in a particular coordinate direction. These do not change volume and thus can be used to compute volumes of parallelipipeds.

Lecture 9: September 20

We still consider a decomposition $PA = LU$. Given a decomposition $A = LU$, to solve a linear system $Ax = b$, we see that $LUx = b$, so first we solve $Ly = b$ and then $Ux = y$. These two solves are $O(n^2)$ once we have the decomposition.

Consider the block submatrix structures

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \qquad U = \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$

We if $A = LU$, we have that

$$LU = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{22}U_{22} + L_{21}U_{12} \end{bmatrix}$$

- $L_{11}U_{11} = A_{11}$ we can compute by LU factorization
- $L_{11}U_{12} = A_{12}$, so $U_{12} = L_{11}^{-1}A_{12}$ give linear equations that can be solved
- $L_{21}U_{11} = A_{21}$, so $L_{21} = A_{21}U_{11}^{-1}$
- $L_{22}U_{22} + L_{21}U_{12} = A_{22}$, so we compute the LU factorization $L_{22}U_{22} = A_{22} - L_{21}U_{12}$. $S = A_{22} - L_{21}U_{12}$ is a Schur complement.

  Note that $S = A_{22} - A_{21}U_{11}^{-1}L_{11}^{-1}A_{12} = A_{22} - A_{21}A_{11}^{-1}A_{12}$.

Say we want the trailing submatrix of A^{-1}. So we want to solve for Y in

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

After Gaussian elimination, we have

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

So in backward substitution, we solve $SY = I$. Thus, we have another way to think about the Schur complement: it is an inverse of a submatrix of A^{-1}.

Using blocked LU factorization can be efficient by taking advantage of $L3$ BLAS-based routines.

Proposition 1. In decomposing $A = LU$, Gaussian elimination computes

$$A + E = \hat{L}\hat{U}$$

where $\|E\| \leq n\epsilon \left\|\hat{L}\right\| \left\|\hat{U}\right\|$. Where $\hat{L}$ and $\hat{U}$ are exactly computed factors.

Note that this gives a weak backward error bound if $\left\|\hat{L}\right\|$ and $\left\|\hat{U}\right\|$ can blow up in size. For instance, this can happen when we have small diagonal elements appearing in the process, since then the divisions in Gauss transformation steps are by small numbers.

Here we have rough argument for the proposition.

$$\hat{u}_{jk} = fl(a_{jk} - \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik})$$

Now, we use our previous dot product analysis

$$\begin{aligned}\hat{u}_{jk} &= a_{jk}(1+\delta_0) - (\sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik})(1+\tilde{\delta}+\delta_0) \\ a_{jk} &= \frac{1}{1+\delta_0}\Big[\hat{u}_{jk} - (\sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik})(1+\tilde{\delta}+\delta_0)\Big] \\ &= \hat{u}_{jk} - \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik} + e_{jk}\end{aligned}$$

In partial pivoting, at each step, we permute rows such that all multipliers are ≤ 1 in magnitude. This controls the magnitude of the entries of L. Then we have

$$\begin{aligned}\frac{\|E\|}{\|A\|} &\leq \frac{n\epsilon\left\|\hat{L}\right\|\left\|\hat{U}\right\|}{\|A\|} \\ &\leq \frac{n^2\epsilon\left\|\hat{U}\right\|}{\|A\|}\end{aligned}$$

Where $\left\|\hat{U}\right\|/\|A\|$ is called the **pivot growth factor**. Note that we can also consider a decomposition $PAQ = LU$, in which we use row or column pivoting (rook pivoting). We can also decompose $PAQ = LU$ where we pivot so that the highest-magnitude element in the remaining submatrix on the diagonal. Other methods include communication-avoiding variants, such as tournament pivoting. These work well for parallel setups. CALUTP = communication-avoiding LU with tournament pivoting.

Lecture 10: September 23

Problem du jour:

Solving $Ax = b$ by Gaussian elimination with partial pivoting gives $(A+E)\hat{x} = b$ with $\frac{\|E\|}{\|A\|} \leq 3n^2\epsilon\frac{\|U\|}{\|A\|}$. The factor of 3 comes from the errors of the 3 steps of decomposition, forward substitution, and backward substitution.

We now discuss how backward error and conditioning bounds forward error. Suppose the exact solution is $Ax = b$. Using linearized perturbation analysis,

$$\begin{aligned}\delta A x + A\delta x &= \delta b \\ \delta x &= A^{-1}(\delta b - \delta A x)\end{aligned}$$

we want to control δx, so we want $\frac{\|\delta x\|}{\|x\|}$ in terms of $\frac{\|\delta A\|}{\|A\|}$ and $\frac{\|\delta b\|}{\|b\|}$.

$$\begin{aligned}\frac{\|\delta x\|}{\|x\|} &= \frac{\left\|A^{-1}\delta b - A^{-1}\delta A x\right\|}{\|x\|} \\ &\leq \frac{\left\|A^{-1}\delta b\right\| + \left\|A^{-1}\delta A x\right\|}{\|x\|}\end{aligned}$$

we bound the first term as such

$$\begin{aligned}\frac{\left\|A^{-1}\delta b\right\|}{\|x\|} &\leq \frac{\left\|A^{-1}\right\|\|\delta b\|}{\|x\|} \\ &\leq \frac{\left\|A^{-1}\right\|\|\delta b\|}{\|b\|/\|A\|} \\ &= \underbrace{\left\|A^{-1}\right\|\|A\|}_{\kappa(A)}\frac{\|\delta b\|}{\|b\|}\end{aligned}$$

where we use that

$$\|Ax\| = \|b\| \implies \|A\|\|x\| \geq \|b\| \implies \|x\| \geq \frac{\|b\|}{\|A\|}$$

also, we bound the other term by

$$\begin{aligned}\frac{\left\|A^{-1}\delta Ax\right\|}{\|x\|} &\leq \frac{\left\|A^{-1}\right\|\|\delta A\|\|x\|}{\|x\|} \\ &= \left\|A^{-1}\right\|\|\delta A\| \\ &= \kappa(A)\frac{\|\delta A\|}{\|A\|}\end{aligned}$$

Now, suppose we have a true solution $Ax = b$, and we have an approximation $\hat{A}\hat{x} = \hat{b}$. Then we have that

$$\begin{aligned}\hat{A}\hat{x} - Ax &= \hat{b} - b \\ \hat{A}\hat{x} - A\hat{x} + A\hat{x} - Ax &= \hat{b} - b \\ \underbrace{(\hat{A} - A)}_{E}\hat{x} + A(\hat{x} - x) &= \hat{b} - b \\ \hat{x} - x &= A^{-1}(\hat{b} - b - E\hat{x})\end{aligned}$$

note the similarity to the above expressions from linearized pertubation analysis. With some algebra, we get that

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \kappa(A)\left(\frac{\|E\|}{\|A\|}\cdot\frac{\|\hat{x}\|}{\|x\|} + \frac{\left\|\hat{b} - b\right\|}{\|b\|}\right)$$

assuming that $\left\|A^{-1}E\right\| < 1$, we consider the relationship between $(A + E)\hat{x} = b$ and $Ax = b$.

$$\begin{aligned}(A + E)\hat{x} &= Ax \\ (I + A^{-1}E)\hat{x} &= x \\ \hat{x} &= (I + A^{-1}E)^{-1}x\end{aligned}$$

so that by summing the Neumann series we have that

$$\begin{aligned}(I + A^{-1}E) &= \left\| \sum_{j=0}^{\infty} (-A^{-1}E)^j \right\| \\ &\leq \sum_{j=0}^{\infty} \left\| A^{-1}E \right\|^j \\ &= \frac{1}{1 - \left\| A^{-1}E \right\|} \\ \frac{\|\hat{x}\|}{\|x\|} &\leq \frac{1}{1 - \left\| A^{-1}E \right\|}\end{aligned}$$

then we have that

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{\|E\|}{\|A\|}} \left(\frac{\|E\|}{\|A\|} + \frac{\left\| \hat{b} - b \right\|}{\|b\|} \right)$$

Iterative refinement

Suppose that we compute $\hat{L}\hat{U} = A + E = \hat{A}$, where E is modest (so not necessarily very well controlled). Consider the following iteration

$$\begin{aligned} x_0 = A^{-1} \qquad & \left(= \hat{U}^{-1}(\hat{L}^{-1}b) \right) \\ Ax_0 = b - r_0 \qquad & r_0 = b - Ax_0 \\ A\underbrace{(x_0 - x)}_{u_0} = -r_0 \quad & \text{subtracting } Ax = b \\ x = x_0 + A^{-1}r_0 & \\ x_1 = x_0 + \hat{A}^{-1}r_0 & \end{aligned}$$

We have the iteration

$$\begin{aligned} x_0 &= \hat{A}^{-1}b \\ x_{k+1} &= x_k + \hat{A}^{-1}(b - Ax_k) \end{aligned}$$

Thus, we have

$$\begin{aligned} x &= x + \hat{A}^{-1}(b - Ax) \\ e_{k+1} &= e_k + \hat{A}^{-1}Ae_k \\ &= (I - \hat{A}^{-1}A) \\ &= \hat{A}^{-1}(\hat{A} - A)e_k \\ &= (\hat{A}^{-1}E)e_k \end{aligned}$$

so that the errors satisfy

$$\|e_{k+1}\| \leq \left\| \hat{A}^{-1}E \right\| \|e_k\| \leq \left\| \hat{A}^{-1}E \right\|^k \|E_0\|$$

so the errors go to zero if $\left\| \hat{A}^{-1}E \right\| < 1$.

Lecture 11: September 25

Problem du jour: Suppose we have a finite dimensional vector space with a norm such that

$$\|x\| \leq \| \, |x| \, \|$$

for any vector x. Show that

$$|x| \leq |y| \text{ elementwise } \implies \| \, |x| \, \| \leq \| \, |y| \, \|$$

Let X be a normed vector space, and X^* its dual space, the set of all linear maps $X \to \mathbb{R}$. We define the dual norm on elements $x^* \in X^*$ as

$$\|x^*\|_* = \sup_{z \in X, \|z\|=1} |x^* z|$$

Recall that for X an inner product space with Euclidean norm, $l^* \in X^*$ satisfies $l^* x = \langle y, x \rangle$. $\|l^*\|_* = \|y\|$. l_∞ and l_1 are dual to each other. Also, for $1/p + 1/q = 1$, we have that l_p and l_q are dual to each other.

Suppose $\|x\| \leq \| \, |x| \, \|$ for any x. What is a u such that $u^*|x| = \|x\|$ and $\|u^*\| = 1$. We have that $u^*|x| = \sum_{i=1}^n u_i |x_i|$.

If $u_i < 0$, then consider y such that $y_j = |x|_j$ for $j \neq i$, and $y_i = -|x|_i$. Note $|y| = |x|$. Then we have

$$\begin{aligned} u^* y > u^* |x| &= \| |x| \| \\ |u^* y| &\leq \|u^*\| \|y\| \end{aligned}$$

so we have that u has no negative entries. Now, we have that

$$\begin{aligned} \| |x| \| = u^* |x| &\leq u^* |y| \\ &\leq \|u^*\| \| |y| \| \end{aligned}$$

Condition Estimation

Now, we consider, how do we estimate $\left\| A^{-1} \right\|$ given $PA = LU$ in less than $O(n^3)$? We wish to find

$$\begin{aligned} \left\| A^{-1} \right\|_1 &= \max_{\|x\|_1 = 1} \left\| A^{-1} x \right\|_1 \\ &= \max_{\xi \in \{\pm 1\}^n} \max_{\|x\|_1 = 1} \xi^T A^{-1} x \end{aligned}$$

note also that $\{\pm 1\}^n = \{\xi \mid \|\xi\|_\infty = 1\}$. To optimize, we consider

$$\xi^T A^{-1} (x + \delta x) = \xi^T A^{-1} x + \xi^T A^{-1} \delta x$$

this can be done in $O(n^2)$ time. In MATLAB and Julia, `condest` is using a routine based on this condition estimate.

Consider again $\hat{A}\hat{x} = b$, $\hat{A} = A + E$, where $|E| \leq \varphi(n,\epsilon)|A|$. Instead of using norms to study the error, we consider componentwise errors. We can derive scale invariant condition numbers, so that for instance differences in units of measurement do not affect the condition number.

$$\begin{aligned} Ax &= b \\ A\delta x + \delta A x &= 0 \\ \delta x &= -A^{-1}\delta A x \\ |\delta x| &= \left|A^{-1}\right||\delta A||x| \\ &\leq \left|A^{-1}\right||A|\,\varphi(n,\epsilon) \end{aligned}$$

assuming we have a norm with $\|x\| \leq \||x|\|$, we have

$$\|\delta x\| \leq \left\| \left|A^{-1}\right| \, |A| \; \varphi(n,\epsilon) \right\|$$

We define

$$\kappa_{\text{rel}}(A) = \left\| \left|A^{-1}\right| \, |A| \right\|$$

let us consider the effect of scaling A

$$\begin{aligned} \kappa_{\text{rel}}(A) &= \left\| |DA|^{-1}|DA| \right\| \\ &= \left\| \left|A^{-1}D^{-1}\right||DA| \right\| \\ &= \left\| \left|A^{-1}\right|\left|D^{-1}\right||D||A| \right\| \\ &= \left\| \left|A^{-1}\right||A| \right\| = \kappa_{\text{rel}}(A) \end{aligned}$$

so this condition number is invariant under scaling.

Now, consider the residual $r = b - A\hat{x}$. Then we have

$$\begin{aligned} A\hat{x} &= b - r \\ \frac{\|\hat{x} - x\|}{\|x\|} &\leq \kappa(A)\frac{\|r\|}{\|b\|} \quad \text{previous computation} \end{aligned}$$

turning to a matrix error, we wish to find

$$\min \|E\| \;\; \text{s.t. } (A+E)\hat{x} = b$$

observe that

$$\begin{aligned} A\hat{x} &= b - r \\ \left(A + r\frac{\hat{x}^T}{\|\hat{x}\|_2^2}\right)\hat{x} &= b \end{aligned}$$

this is a rank one perturbation of A in the direction that matters.

Symmetric factorization

Let $A = A^T$ for some real matrix A. We associate the quadratic form $\varphi(x) = x^T Ax$. Principal curvatures are eigenvalues, or we have up, down, flat directions. If Z is nonsingular, then $Z^T AZ$ and A have the same inertia.

The standard symmetric variant of LU is $PAP^T = LDL^T$, where L is unit lower triangular and D is diagonal. Note that D has the same inertia as A. If A is positive definite, then we have $A = LL^T$, where L is lower triangular. This is called the **Cholesky factorization**.

The algorithm for Cholesky factorization is very similar to LU factorization, except we need not pivot for stability.

Lecture 12: September 27

The algorithm for Cholesky factorization is much like Gaussian elimination. Suppose A is symmetric positive definite, and break it into blocks.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{12}^T & A_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & L_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21}^T \\ 0 & L_{22} \end{bmatrix}$$

Then we have that

$$\begin{aligned} a_{11} &= l_{11}^2 \\ a_{12}^T &= l_{21} l_{11} \\ A_{22} &= L_{22} L_{22}^T + l_{21} l_{21}^T \end{aligned}$$

note that $a_{11} \neq 0$ due to A being spd, so $l_{11} \neq 0$. Then l_{21} is determined, (there is no division by zero). Finally to find L_{22}, we recurse, and find the smaller Cholesky factorization

$$L_{22} L_{22}^T = A_{22} - l_{21} l_{21}^T$$

an argument can be made for why $A_{22} - l_{21} l_{21}^T$ is spd, so this is possible. Note that by properties of spd matrices, no pivoting is needed. The Cholesky factor L is often analogous to the square root of real numbers in certain applications.

Diagonal Dominance

We call a matrix A **strictly diagonally dominant** if

$$|a_{jj}| > \sum_{i \neq j} |a_{ij}| \qquad j = 1, \ldots, n$$

Decompose A into $D + F$, where D is a diagonal matrix containing the diagonal of A, and F is the off diagonal. Then A is strictly diagonally dominant if and only if

$$\left\| FD^{-1} \right\|_1 < 1$$

Then we can write $A = (I + FD^{-1})D$. This can be a useful decomposition, as D is easy to solve with. The Schur complement is also diagonally dominant, so Gaussian elimination can be done without pivoting.

Tridiagonal Matrices

Consider A a symmetric positive definite tridiagonal matrix, with diagonal given by $(\alpha_1, \ldots, \alpha_n)$ and $(\beta_1, \ldots, \beta_{n-1})$. Applying an iteration of Cholesky factorization, we have the lower right block S, where

$$S = \begin{bmatrix} \alpha_2 - \beta_1^2/\alpha_1 & \beta_2 & \\ \beta_2 & \alpha_3 & \ddots \\ \ddots & \ddots & \ddots \end{bmatrix}$$

Each iteration takes constant time, so Cholesky factorization takes linear time for such matrices. Tridiagonal matrices are a special case of banded matrices.

Sherman-Morrison-Woodbury

Suppose we have the linear system

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

Blocked Gaussian elimination gives a Schur complement of $S = D - CA^{-1}B$. We have $Sy = g - CA^{-1}f$ and $Ax = f - By$.

Consider a similar problem

$$(A + UW^T)x = f$$

where A is updated by the (low rank) matrix UW^T. Define $y = W^T x$. Then we can rewrite the system

$$\begin{bmatrix} A & U \\ W^{-1} & -I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

Then we have

$$x = (A^{-1} - A^{-1}U(I + W^T A^{-1}U)^{-1}W^T A^{-1})f$$

so we can rewrite

$$(A + UW^T)^{-1} = A^{-1} - A^{-1}U(I + W^T A^{-1}U)^{-1}W^T A^{-1}$$

this is the **Sherman-Morrison-Woodbury** formula. Note if we already have A^{-1}, then to update A^{-1}, the only inverse that needs to be computed is of $I + W^T A^{-1}U$, which is a small matrix if UW^T is low rank.

Vandermonde matrices

Suppose we have a degree $n-1$ polynomial $p(x) = \sum_{j=1}^{n} c_j x^{j-1}$, and we have sample points (x_i, f_i). We wish to find the c_j that allow p to interpolate the points. If we define $V_{ij} = x_i^{j-1}$, then $Vc = f$ is our system, where $c = (c_1, \ldots, c_n)$. This system is quite ill-conditioned, so other methods (of which there are an abundance) should be used for polynomial interpolation.

Circulant matrices

A **circulant matrix** is one of the form

$$\begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ a_n & a_1 & a_2 & \dots & a_{n-1} \\ a_{n-1} & a_n & a_1 & \dots & a_{n-2} \\ \vdots & & \ddots & \ddots & \\ a_1 & & & \dots & a_1 \end{bmatrix}$$

such a matrix is determined by the n elements $(a_1, \dots, a_n)$. The linear system $Cx = y$ can be solved very quickly. C is diagonalized by the Fourier transform. The F be the Fourier transform matrix. Then

$$\begin{aligned} F\Lambda F^{-1}x &= y \\ x &= F\lambda^{-1}F^{-1}y \end{aligned}$$

so the system can be solved in $O(n \log n)$ time.

Toeplitz and Hankel matrices also have fast linear solve methods due to their structure.

Lecture 13: 9/30

Problem du jour: Argue that if $Z \sim N(0, I)$, then $R^T Z \sim N(0, C)$, where $C = R^T R$ is a Cholesky factorization.

Recall if $X \sim N(0, C)$, then

$$p(x) \propto \exp\Big(-\frac{1}{2}x^T C^{-1} x\Big)$$

If $C = R^T R$, $C^{-1} = R^{-1}R^{-T}$. Then for $z = R^{-T}x$, so that $x = R^T z$, we have

$$\begin{aligned} p(x) &\propto \exp\Big(-\frac{1}{2}(x^T R^{-1})(R^{-T}x)\Big) \\ &= \exp\Big(-\frac{1}{2}z^T I z\Big) \end{aligned}$$

Sparse LU factorization

Naive LU factorization on a sparse matrix may destroy the sparsity pattern. For the example from our homework, the matrix

$$\begin{bmatrix} \times & & & \times \\ & \times & & \times \\ & & \ddots & \vdots \\ \times & \times & \dots & \times \end{bmatrix} = \begin{bmatrix} \times & & & \\ & \times & & \\ & & \ddots & \\ \times & \times & \dots & \times \end{bmatrix} \begin{bmatrix} \times & & & \times \\ & \times & & \times \\ & & \ddots & \vdots \\ & & & \times \end{bmatrix}$$

has sparse L and U factors, whereas

$$\begin{bmatrix} \times & \times & \dots & \times \\ \times & \times & & \\ \vdots & & \ddots & \\ \times & & \times & \end{bmatrix}$$

has potentially full fill.

In Gaussian elimination, we want to reduce filling and maintain stability. General LU is of the form:

$$PAQ = LU$$

where P is a permutation matrix chosen for stability, and Q is a permutation matrix chosen to reduce fill.

In this lecture, we focus on Cholesky factorization, since then we need not consider pivoting for stability. We define **fill** as nonzeros in L and U that are zero in A. We can think about fill by taking our matrix and associating it with a graph. Consider $A \in \mathbb{R}^{n \times n}$ as a graph on n nodes, where we define

$$V = \{1, 2, \dots, n\} \qquad E = \{(i,j) \mid a_{ij} \neq 0\}$$

Eliminating by node i causes all neighbors of i to be connected in a clique in the Schur complement graph.

$$\begin{bmatrix} \times & & \times & & & & & \\ & \times & \times & & & & & \\ \times & \times & \times & & & & & \times \\ & & & \times & & \times & & \\ & & & & \times & \times & & \\ & & & \times & \times & \times & \times & \\ & & \times & & & \times & \times & \end{bmatrix}$$

The graph of this matrix is a tree. For any tree, we can do Gaussian elimination without fill. We do this by eliminating by the leaves first. For general sparse matrices with cycles, we cannot eliminate fill. To do Gaussian elimination on matrices with general graphs, we approximately treat the graph as a tree.

For instance, for a square mesh graph, we use **nested dissection**. The general idea is to:

- Find a small vertex separator such that paths between one side and the other must go through the separator
- Recurse on the separated subgraphs

We can look at connectivity at the block level. Then we do not have fill that falls outside of the block pattern. On an $n \times n$ mesh, with $O(n^2) = O(N)$ unknowns, then unstructured (dense) Gaussian elimination is $O(N^3) = O(n^6)$. With a rough analysis for nested dissection, eliminating the final Schur complement that has n nodes that are densely connected together contributes $O(n^3)$. We also have two Schur complements at approximately half the size, $\approx n^2/2$ nodes. This takes $2O((n/2)^3) \approx c2(n/2)^3 = (c/4)n^3$ time. The next step takes $c4(n/2)^3 = (c/2)n^3$. Thus, the total cost is about $O(n^3) = O(N^{3/2})$ due to the geometric sum structure. This is becase forming each Schur complement is (asymptotically) less expensive than eliminating it.

On a $3D$ mesh, of size $n \times n \times n$, $N = n^3$, then dense solves take $O(N^3) = O(n^9)$. The top level separator (an $n \times n \times 1$ slice) has n^2 degrees of freedom. Gaussian elimination on this separator Schur complement takes $O((n^2)^3) = O(n^6) = O(N^2)$. Again this determines the asymptotic cost. Thus, people often use additional structure of the Schur complements (depending on the problem) or other methods for $3D$ meshes.

Lecture 14: 10/2

Least squares

Say we have observations $T_1, \ldots, T_m$ with corresponding independent variables $C_1, \ldots, C_m$, and we wish to fit a function $T \approx \alpha C + \beta$. We can frame this as solving the least-squares problem

$$\min_{\alpha,\beta} \sum_{i=1}^{m} \Big(T_i - (\alpha C_i + \beta) \Big)^2$$

equivalently, in matrix form,

$$\min_{\alpha,\beta} \left\| \underbrace{\begin{bmatrix} C_1 & 1 \\ \vdots & \vdots \\ C_m & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} T_1 \\ \vdots \\ T_m \end{bmatrix}}_{b} \right\|_2^2$$

so our abstract problem is for $A \in \mathbb{R}^{m \times n}$, $m > n$,

$$\min_x \frac{1}{2} \|Ax - b\|^2$$

let $\varphi(x) = \frac{1}{2}\|Ax - b\|^2$. Then we have

$$\begin{aligned} \varphi(x) &= \frac{1}{2}(Ax - b)^T (Ax - b) \\ &= \frac{1}{2} x^T A^T A x - x^T A^T b + \frac{1}{2} b^T b \\ \delta\varphi(x) &= \frac{1}{2}\Big(\delta x^T A^T A x + x^T A^T A \delta x - 2\delta x^T A^T b \Big) \quad \text{only perturbing } x \\ &= \delta x^T (A^T A x - A^T b) \end{aligned}$$

setting the derivative equal to zero, we have

$$A^T A x = A^T b$$

in different notation,

$$\begin{aligned} \varphi(x) &= \frac{1}{2} r^T r \\ \delta\varphi(x) &= \delta r^T r \\ &= \delta x^T A^T r \end{aligned}$$

because $\delta r = A\delta x$. Setting this to zero, we want r so that $A^T r = 0$. Thus, these equations are called the normal equations, since finding the r where this quantity is zero is finding an r that is orthogonal to A^T. The normal equations are

$$\min \frac{1}{2}\|Ax-b\|^T = A^T Ax = A^T b \implies x = (A^T A)^{-1} A^T b$$

we define $A^\dagger = (A^T A)^{-1} A^T$. This is the **Moore-Penrose pseudoinverse**. Note that it is in fact the inverse of A for nonsingular A.

Definition 0.1. A **pseudoinverse** of a full column rank $A \in \mathbb{R}^{m\times n}$ where $m > n$ is a $B \in \mathbb{R}^{n\times m}$ such that $BA = I$.

The Moore-Penrose pseudoinverse is an example of a pseudoinverse.

$\Pi = AB$ where B is a pseudoinverse satisfies

$$\Pi^2 = ABAB = AB = \Pi$$

so that it is a projector. For the Moore-Penrose pseudoinverse, we have a projector $\Pi = AA^\dagger$. In our least-squares problem, $Ax = \Pi b$. Also, $b - Ax = (I - \Pi)b$.

Statistics and least squares

Now, we consider the statistical justification for the least-squares problem. Suppose we have samples from $N(Ax, \Sigma^2)$, where x is unknown. Given a sample b, we wish to estimate x. We want a simple statistic, so we take a linear estimator $\hat{x} = Lb$ for some linear function L. We desire this estimator to be unbiased, so

$$x = \mathbb{E}[\hat{x}] = \mathbb{E}[Lb] = L\mathbb{E}[b] = LAx$$

we wish to find a pseudoinverse that minimizes the total variance $Var[\hat{x}]$.

$$\begin{aligned} \hat{x} &= LAx \\ Var[\hat{x}] &= \mathbb{E}[\hat{x}\hat{x}^T] - \mathbb{E}[\hat{x}]\mathbb{E}[\hat{x}]^T \\ &\cdots \end{aligned}$$

To solve the normal equations $A^T Ax = A^T b$, we have different methods.

- $A^T A = R^T R$ (Cholesky). This implies that

  $$\begin{aligned} R^T Rx &= A^T b \\ Rx &= R^{-T} A^T b \\ x &= R^{-1}(R^{-T}(A^T b)) \end{aligned}$$

  we consider the intermediate factorization

  $$R^{-T} A^T = (AR^{-1})^T = Q^T$$

  note that

  $$\begin{aligned} Q^T Q &= R^{-T} A^T AR^{-1} \\ &= R^{-T} R^T RR^{-1} \\ &= I \end{aligned}$$

  so Q has orthogonal columns. This gives the economy QR factorization $A = QR$

- $A = QR$ (QR) is thus another method for solving the normal equations
- $A = U\Sigma V^T$ (SVD). Then we have $A^\dagger = V\Sigma^{-1}U^T$.

Lecture 15: 10/4

Problem du jour: Suppose A is spd with a Cholesky factorization $A = R^T R$. How does on compute δR (in terms of δA)? Here is a hint:

$$\begin{aligned} \delta A &= \delta R^T R + R^T \delta R \\ R^{-T}\delta A R^{-1} &= \underbrace{R^{-T}\delta R^T}_{\text{lower triangular}} + \underbrace{\delta R R^{-1}}_{\text{upper triangular}} \end{aligned}$$

Note that

$$\begin{aligned} \frac{1}{2}\underbrace{\|Ax-b\|^2}_{\varphi} &= \frac{1}{2}\langle Ax-b, Ax-b\rangle \\ \delta\varphi &= \langle A\delta x, Ax-b\rangle \end{aligned}$$

so to minimize $\varphi(x)$, we want $\langle A\delta x, Ax-b\rangle = 0$ for all δx.

Note that the expectation

$$\mathbb{E}[XY] = \int X(z)Y(z)\, d\mu(z)$$

satisfies the properties

$$\begin{aligned} \mathbb{E}[XY] &= \mathbb{E}[YX] \\ \mathbb{E}[\alpha XY] &= \alpha\mathbb{E}[XY] \\ \mathbb{E}[(X_1+X_2)Y] &= \mathbb{E}[X_1Y]_{\mathbb{E}}[X_2Y] \\ \mathbb{E}[X^2] &\geq 0 \\ \mathbb{E}[X^2] = 0 &\iff X = 0 \quad \text{if disallow zero variance random variables} \end{aligned}$$

so that the expectation is an inner product.

In finite dimensions, we can always choose a basis such that

$$\langle x, y\rangle_M = y^T M x \qquad M \text{ some spd matrix}$$

Consider the $L^2[-1,1]$ inner product on polynomials

$$\langle p, q\rangle_{L^2[-1,1]} = \int_{-1}^{1} p(x)q(x)\, dx$$

$$\begin{aligned} \langle \sum_{j=0}^{d} a_i x^i, \sum_{j=0}^{d} b_j x^j\rangle_{L^2} &= \sum_i \sum_j a_i b_j \langle x^i, x^j\rangle_{L^2} \\ &= b^T M a, \qquad\qquad M_{ij} = \langle x^{i-1}, x^{j-1}\rangle_{L^2} \end{aligned}$$

Gauss-Markov

Suppose we have $b \sim N(Ax, \Sigma^2)$, and we wish to find the best linear unbiased estimator (BLUE) for x. The likelihood is given by

$$C \cdot \exp\Big(-\frac{1}{2}(b-Ax)^T\Sigma^{-2}(b-Ax)\Big)$$

to maximize the likelihood, we solve

$$\min \|b-Ax\|^2_{\Sigma^{-2}}$$

so we want

$$\begin{aligned}(A\delta x)^T\Sigma^{-2}(b-Ax) &= 0\\ (A^T\Sigma^{-2}A)x &= A^T\Sigma^{-2}b\end{aligned}$$

let $A^\dagger_{\Sigma^{-2}} = (A^T\Sigma^{-2}A)^{-1}(A^T\Sigma^{-2})$. This is a pseudoinverse. If Σ is the identity, then this is the Moore-Penrose pseudoinverse as seen before.

We require that our estimator $\hat{x}$ satisfies

$$\begin{aligned}\hat{x} &= Lb && \text{linear in data}\\ \mathbb{E}[\hat{x}] &= LAx = x \quad \text{i.e. } LA = I && \text{unbiased}\end{aligned}$$

Let L be of the form $L = A^\dagger_{\Sigma^{-2}} + F$ for any F such that $FA = 0$, so that $LA = I$. Then we have that

$$Lb \sim N(x,\, L\Sigma^2L^T)$$

$$\begin{aligned}L\Sigma^{-2}L^T &= (A^\dagger_{\Sigma^{-2}} + F)\Sigma^2(A^\dagger_{\Sigma^{-2}} + F)^T\\ &= A^\dagger\Sigma^2(A^\dagger)^T + 2[A^\dagger\Sigma^2F^T]^S + F\Sigma^2F^t\end{aligned}$$

where $[B]^S$ denotes the symmetric part of B. The middle summand is

$$(A^T\Sigma^{-2}A)^{-1}A^T\Sigma^{-2}\Sigma^2F = (A^\dagger)^{-1}(FA)^T$$

thus, the minimizing choice of variance is $F = 0$ where we use the partial order $A \succeq B$ if $A - B$ is positive semidefinite. Hence, $x = A^\dagger_{\Sigma^{-2}}b$ is the BLUE.

QR Factorization

Let $A \in \mathbb{R}^{m\times n}$. The full QR factorization is of the form $A = QR$ where $Q^TQ = I$, R is upper triangular, $Q \in \mathbb{R}^{m\times m}$, and $R \in \mathbb{R}^{m\times n}$.

The economy QR factorization is of the form $A = \hat{Q}\hat{R}$, where $\hat{Q}^T\hat{Q} = I$, $\hat{R}$ is upper triangular, $\hat{Q} \in \mathbb{R}^{m\times n}$, and $\hat{R} \in \mathbb{R}^{n\times n}$.

The Gram-Schmidt process is of the form

- $a_1 = q_1r_{11}, \qquad r_{11} = \|a_1\|$
- $a_2 - q_1\underbrace{(q_1^Ta_2)}_{r12} = q_2r_2$

- $\vdots \quad \vdots$
- $a_k - \sum_{j=1}^{k-1} q_j \underbrace{q_j^T a_j}_{r_{jk}} = q_k r_k$

with this construction, we have that $a_k = \sum_{j=1}^{k} q_j r_{jk}$. Thus, we can reconstruct

$$\begin{bmatrix} a_1 & a_2 & \dots \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \dots \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots \\ & r_{22} & \dots \\ & & \ddots \end{bmatrix}$$

this gives a QR factorization. However, Gram-Schmidt is foward-unstable. The big idea to compute a QR factorization in a nice way is to think of an $LU-$like sequence of operations, but instead of Gauss transforms (shears), we use orthogonal transforms consisting of reflections (Householder) and rotations (Givens).

Lecture 16: 10/7

We present a method to compute the QR factorization using orthogonal transformations. A step of Gaussian elimination is of the form:

$$\begin{bmatrix} 1 & 0 \\ -\frac{a_{21}}{a_{11}} & I \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ 0 & S \end{bmatrix}$$

We want a reflection that maps

$$v = \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} \pm\|v\| \\ 0 \end{bmatrix}$$

Note that the only nonzero entry of the result must be $\pm\|v\|$ since orthogonal transformations preserve the 2-norm. We wish to find the plane of reflection, as defined by a vector normal to it. The normal direction is given by $v \pm \|v\| e_1$. We choose the sign that avoids cancellation errors, so choose $\text{sign}(v_1)$. Normalize this to get u, and define $H = I - 2uu^T$.

Thus, our algorithm to get the reflector is

- $\tilde{u} = v \pm \|v\| e_1$ (sign chosen as $\text{sign}(v_1)$)
- $u = \frac{\tilde{u}}{\|\tilde{u}\|}$
- $H = I - 2uu^T$

Our algorithm for Householder QR is

```
for j = 1:n
  u_j = HH(A(j:m), j)
  Store u_j
  % Apply reflection to A:
  A(j:m, j:n) = A(j:m, j:n) - 2*u_j*(u_j^T A(j:m,j:n))
end
```

The total cost is $O(mn^2)$.

Now, we consider decomposing a matrix using rotations. Thus, we want a matrix of the form

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \qquad c^2 + s^2 = 1$$

that satisfies

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

which implies that (s, c) is orthogonal to (x, y), and thus $(c, -s)$ is parallel to (x, y). Thus, we have that

$$c = \frac{x}{\sqrt{x^2 + y^2}} \qquad s = \frac{-y}{\sqrt{x^2 + y^2}}$$

Applying Givens rotations or Householder reflectors to a matrix is backward-stable. These are orthogonal transformations, the nicest types.

Consider the case of a sparse matrix A. If A is sparse, and $A^T A$ is "nice" (in which elimination does not produce much fill), then since R is also the Cholesky factor in a QR factorization, we have a sparse R. Since

$$A^T Ax = A^T b \implies Rx = R^{-T} A^T b$$

this is very nice.

Now, suppose we have a constrained optimization problem

$$\min_x \|Ax - b\|^2 \quad \text{s.t. } Cx = d$$

where $A \in \mathbb{R}^{m \times n}$ for $m > n$, $C \in \mathbb{R}^{k \times n}$ for $k < n$. We can use Lagrange multipliers. Let $\mathcal{L}(x, \lambda) = \frac{1}{2}\|Ax - b\|^2 + \lambda^T (Cx - d)$. Then we have

$$\begin{aligned} \delta \mathcal{L} &= \delta x^T A^T (Ax - b) + \delta x^T C^T \lambda + \delta \lambda (Cx - d) \\ &= \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix}^T \left(\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} - \begin{bmatrix} b \\ d \end{bmatrix} \right) \end{aligned}$$

Another way to change the problem is to consider $Cx = d$, and suppose $C^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$. Then the constraint is $R^T Q^T x = d$. Define $y = Q^T x$, so that

$$\begin{bmatrix} R_1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = d$$

Note that y_1 is contrained by this equation, and y_2 is unconstrained.

Lecture 17: 10/9

Say we have the vectors $1, x, x^2$, and want a QR decomposition, where we use the L^2 inner product.

$$\begin{bmatrix} 1 & x & x^2 \end{bmatrix} = \begin{bmatrix} L_0(x) & L_1(x) & L_2(x) \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ & r_{22} & r_{23} \\ & & r_{33} \end{bmatrix}$$

where L_0, L_1, and L_2 are orthogonal polynomials of degree at most 2. We have two ways to approach this. First, we have $A^TA = R^TR$ and $Q = AR^{-1}$. Here we have the Gram matrix and the Cholesky factor

$$A^TA = \begin{bmatrix} 2 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{2}{5} \end{bmatrix} \qquad R = \begin{bmatrix} \sqrt{2} & 0 & \frac{\sqrt{2}}{3} \\ 0 & \sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & \sqrt{\frac{8}{45}} \end{bmatrix}$$

The other method to compute the decomposition is by Gram-Schmidt. The resulting L_0, L_1, L_2 are Legendre polynomials.

Error Analysis of Least Squares

We consider the sensitivity of x and the residual $r = Ax - b$ in the solution of $\min_x \|Ax - b\|^2$. Say we have $A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $b = \begin{bmatrix} \epsilon \\ 1 \end{bmatrix}$. Then we have $x = A^\dagger b = \epsilon$. If we instead have $b = \begin{bmatrix} -\epsilon \\ 1 \end{bmatrix}$, then we have $x = A^\dagger b = -\epsilon$. Note that this is a large relative change in x compared to a small relative norm change in b.

We let θ be the angle between b and the range space of A,

$$\cos(\theta) = \frac{\|Ax\|}{\|b\|} \quad \sec(\theta) = \frac{\|b\|}{\|Ax\|} \quad \tan(\theta) = \frac{\|r\|}{\|Ax\|}$$

For general A, define the condition number

$$\kappa(A) = \|A\| \left\|A^\dagger\right\| = \frac{\sigma_1(A)}{\sigma_n(A)}$$

since we are using the 2-norm.

First, we consider sensitivity of $y = A^Tb$. We have that

$$\begin{aligned} \delta y &= \delta A^T b + A^T \delta b \\ \|\delta y\| &\le \|\delta A\|\|B\| + \|A\|\|\delta b\| \\ \frac{\|\delta y\|}{\|y\|} &\le \frac{\|\delta b\|\|b\|}{\|A^Tb\|} + \frac{\|A\|\|\delta b\|}{\|A^Tb\|} \\ &= \frac{\|A\|\|b\|}{\|A^Tb\|}\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right) \end{aligned}$$

Let $A = U\Sigma V^T$ be an economy SVD. Then we have

$$\begin{aligned} \|A\|\|b\| &\le \sigma_1(A)\|b\| \\ \left\|A^Tb\right\| &\ge \sigma_n(A)\left\|U^Tb\right\| \end{aligned}$$

$$\begin{aligned} \frac{\|A\|\|B\|}{\|A^Tb\|} &= \frac{\sigma_1(A)}{\sigma_n(A)}\frac{\|b\|}{U^Tb} \\ &= \kappa(A)\sec(\theta) \end{aligned}$$

Now, we consider the conditioning of $x = (A^TA)^{-1}A^Tb$. Then we have

$$\delta x = -(A_TA)^{-1}(\delta A^TA + A^T\delta(A^TA)^{-1}A^Tb + (A^TA)^{-1}(\delta A^Tb + A^T\delta b)$$

After some basic work, we have

$$\frac{\|\delta x\|}{\|d\|}\kappa_2(A) \leq \Big(\kappa_2(A)\tan(\theta) + \kappa(a)\Big)\frac{\|\delta A\|}{\|A\|} + \kappa(A)\sec(\theta)\frac{\|\delta b\|}{\|b\|}$$

Lecture 18: 10/11

Problem du jour: If $\mathcal{U} = R(A)$ and $\mathcal{V} = R(B)$, how can we compute

$$\max_{u\in\mathcal{U}}\min_{v\in\mathcal{V}}\angle(u,v)$$

$\angle(u,v)$ being the angle between u and v.

Say we have $U, V \in \mathbb{R}^{n\times k}$ where $U^TU = I$ and $V^TV = I$. Our problem is

$$\min_{u=Ux\in\mathcal{U}}\max_{v=Vy\in\mathcal{V}}\cos(\angle(u,v))$$

$$\min_{\substack{u=Ux\in\mathcal{U}\\ \|x\|=1}}\max_{\substack{v=Vy\in\mathcal{V}\\ \|y\|=1}} v^Tu = y^TV^TUx$$

There exists a choice of basis for spaces such that the problem is

$$\min_{\|x\|=1}\max_{\|y\|=1} y^T\Sigma x$$

where the solution happens to be $\sigma_k(v^Tu)$.

Now, consider the largest angle between $R(A)$ and $R(A+E)$. This depends on $\kappa(A)\frac{\|E\|}{\|A\|}$. Thus, we have issues when $\kappa(A) \gg 1$. This occurs with ill-posed problems, such as when we have correlated explanatory variables.

Lecture 19: 10/16

Problem du jour: Suppose $A \in \mathbb{R}^{m\times n}$, $m > n$. How can we solve

$$\min\frac{1}{2}\|x\|^2 \quad \text{s.t.} \quad A^Tx = b$$

note that the linear system is underdetermined, so there are multiple x satsifying it. Say we have a full QR decomposition $A = QR = \begin{bmatrix}Q_1 & Q_1\end{bmatrix}\begin{bmatrix}R_1\\0\end{bmatrix}$. Then our constraint is $R_1^TQ_1^Tx = b$. Assuming A has full column rank, we have $Q_1^Tx = R_1^{-T}b$. We can write

$$\begin{aligned} x &= Q_1Q_1^Tx + Q_2Q_2^Tx \\ &= Q_1R_1^{-T}b + Q_2y \\ \|x\|^2 &= \left\|R_1^{-T}b\right\|^2 + \|y\|^2 \end{aligned}$$

Thus, to minimize $\|x\|^2$, set $y = 0$, so we have the minimizing x as $x = Q_1 R_1^{-T} b$.

Let $A = U\Sigma V^T$ be a full SVD. The general Moore Penrose pseudoinverse if $A^\dagger = V\tilde{\Sigma}^{-1}U^T$, where $\tilde{\Sigma}^{-1}$ is the matrix of the same structure as Σ^T consisting of the nonzero singular values of A inverted.

Regularization ideas:

- Pivoted QR and factor selection
- Tikhonov/ ridge regression
- Truncated SVD
- l^1 and the lasso
- Regularization via iteration

Parameter choice:

- Morozov
- L-curve
- Cross-validation

Say we have $A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$ and want to pick columns of A that give a well-conditioned matrix (as linearly independent as possible). The idea for pivoted QR using Gram-Schmidt (it would be better to use Householder transformations but this method is easier to explain) is:

1. Pick a_j with largest norm and swap it with the first column
2. Compute $\tilde{a}_j = (I - q_1 q_1^T) a_j$ for each column
3. Pick $\tilde{a}_j$ with largest remaining norm, swap with the first column under consideration, and repeat

This computes a decomposition $A\Pi = QR$ for some permutation Π. Moroever, we have $|r_{ii}| \geq |r_{jj}|$ for $i < j$. From here, we can choose columns that have diagonals with magnitudes above some tolerance. MATLAB utilizes this when solving nearly singular linear systems.

The lasso is of the form:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \lambda \|x\|_1$$

The regularization term encourages sparse solutions x. However, the loss is not smooth.

Tikhonov regularization (aka ridge regression) is of the form:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2$$

Note that this is equivalent to the ordinary least squares problem

$$\min_{x \in \mathbb{R}^n} \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2^2$$

The normal equations are of the form

$$(A^TA+\lambda^2 I)x = A^Tb$$

Denote the solution $x_\lambda = A_\lambda^\dagger b = (A^A+\lambda^2 I)^{-1}b$. This estimator is not necessarily unbiased, but has lower variance in general. Say we have an economy SVD $A = U\Sigma V^T$. Then we have

$$\begin{aligned} A_\lambda^\dagger &= (A^TA+\lambda^2 I)^{-1}A^T \\ &= V(\Sigma^2+\lambda^2 I)^{-1}V^TV\Sigma U^T \\ &= V\Sigma(\Sigma^2+\lambda^2 I)^{-1}U^T \\ &= V\tilde{\Sigma}_\lambda^{-1}U^T \end{aligned}$$

so the nonzeros of $\tilde{\Sigma}_\lambda^{-1}$ are $\sigma_j^{-1} = \frac{\sigma_j}{\sigma_j^2+\lambda^2}$.

For truncated SVD, let we instead take

$$\sigma_j^{-1} = \begin{cases} \sigma_j^{-1}, & \sigma_j > \tau \\ 0, & \sigma_j \le \tau \end{cases}$$

for some tolerance τ. This means that we take a lower rank approximation to A, instead of blending the singular values with a parameter λ.

Lecture 20 : 10/18

Kernel methods

One of the reasons we may solve a least-squares problem is to fit a model to data. We may want to predict $f(a_1,\dots,a_n) \approx \sum_{j=1}^n a_j x_j$, so we minimize $\|Ax-b\|_2^2$. However, we may want to fit a nonlinear model.

Say we have data $(x_1,f(x_1)),(x_2,f(x_2)),\dots,(x_m,f(x_m))$ where $x_i \in \mathbb{R}^n$. The goal is to find a model $s(x)$ that approximates $f(x)$. The simplest model is (using statistical language) $s(x) = x^T\theta$, in which our problem is $\|X\beta - y\|$.

An idea to extend this is to use a feature map $\varphi : \mathbb{R}^n \to \mathbb{R}^N$, where $N \gg n$. If $m > N$, then $\min\|\Phi\beta - y\|$ is still a least-squares problem, where the ith row of Φ is $\varphi(x_i)$, $\Phi_{ij} = (\varphi(x_i))_j$. Then our model is $s(x) = \varphi(x)^T\beta$. In other terms, we have

$$\begin{aligned} s(x) &= \varphi(x)^T\Phi^\dagger y \\ &= \varphi(x)^TG^{-1}\Phi^Ty && G = \Phi^T\Phi,\ G_{ij} = \varphi(x_i)^T\varphi(x_j) \\ &= \left(\varphi(x)^TG^{-1}\right)\left(\Phi^Ty\right) \\ &= \left(\varphi(x)^TR^{-1}\right)\left(R^{-T}\Phi^Ty\right) && G = R^TR \end{aligned}$$

Now, for $m < N$ (we can think of N as infinite if our feature space is just some Hilbert space, in

which case we use other inner products instead of the Euclidean inner product)

$$\begin{aligned} s(x) &= \varphi(x)^T \Phi (\Phi^T \Phi)^{-1} y \\ &= \varphi(x)^T \beta \qquad\qquad \beta = \Phi(\Phi^T\Phi)^{-1} y \\ &= \Big(\varphi(x)^T \Phi\Big) \underbrace{(\Phi^T\Phi)^{-1} y}_{c} \\ &= \underbrace{\Big(\varphi(x)^T \Phi\Big)(\Phi^T\Phi)^{-1}}_{d(x)^T} y \end{aligned}$$

Define a **kernel function** $k(x,x') = \varphi(x)^T \varphi(x')$. Now, note that

$$\begin{aligned} s(x) &= (\varphi(x)^T \Phi)(\Phi^T\Phi)^{-1} y \\ &= k_{xX} (K_{XX})^{-1} y \end{aligned}$$

Where $[K_{XY}]_{ij} = k(x_i, y_j)$. Note that this does not explicity use φ. This is called the **kernel trick**.

Note that Φ is of the shape

$$\Phi = \begin{bmatrix} \varphi(x_1)^T \\ \vdots \\ \varphi(x_m)^T \end{bmatrix}$$

We consider whether we can predict $\varphi(x)$ well as a linear combination of $\varphi(x_1), \ldots, \varphi(x_m)$. We solve a problem of the form

$$\min \Big\| \Phi^T d(x) - \varphi(x) \Big\|$$

Then $f(x) \approx d(x)^T y = \sum_{j=1}^m d_j(x) f(x_j) = s(x)$. Indeed, $d(x) = (\Phi^T)^\dagger \varphi(x)$.

Also, we have

$$\begin{aligned} K_{XX} c &= y \\ s(x) = k_{xX} c &= \sum_{j=1}^m k(x, x_j) c_j \end{aligned}$$

Often our kernel matrices have nice structural properties that can be taken advantage of here.

Consider the **squared exponential kernel**

$$k(x,y) = e^{-\|x-y\|^2 / 2\sigma^2}$$

Then we have that K_{XX} is typically almost low rank, meaning the eigenvalues decay (of course, K_{XX} is positive definite, so it is actually full rank). Usually, we solve $(K_{XX} + \eta^2 I)c = y$, for which there are many different justifications. Say we have a low rank decomposition $K_{XX} \approx ZZ^T$. Then the goal is to solve $(ZZ^T + \eta^2 I)c = y$.

We want something of the form $z(x)^T d$ rather than $z(x)^T (Zc)$. We observe that the equations for d are of the form (after some manipulations) (note: missing a transpose somewhere)

$$\min_d \|Zd - y\|^2 + \eta^2 \|d\|^2$$

This is a regularized least-squares problem. This least-squares problem is $n \times k$, which is much nicer to solve.

Suppose we have $(I + ZZ^T)c = y$, and consider a QR factorization

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R = \begin{bmatrix} Z \\ I \end{bmatrix}$$

compute $d = R^{-1}Q_1^T y$ and the residual is c if we put in d, which here is of the form $c = \pm(y - Zd)$.

Lecture 21: 10/21

The standard eigenvalue problem is

$$Ax = \lambda x \qquad A \in \mathbb{C}^{n\times n} \text{ or } \mathbb{R}^{n\times n}, \quad \lambda \in \mathbb{C}$$

For matrix decompositions, this is of the form

$$\begin{aligned} AV &= V\Lambda && \Lambda \text{ diagonal} \\ \text{or } AV &= VJ && J \text{ a Jordan matrix} \end{aligned}$$

Also, we have the Schur form

$$\begin{aligned} \text{Real form: } AQ &= QT && T \text{ quasi upper triangular, } Q^TQ = I,\ Q, T \in \mathbb{R}^{n\times n} \\ \text{Complex form: } AU &= UT && T \text{ quasi upper triangular, } U^*U = I,\ Q, T \in \mathbb{C}^{n\times n} \end{aligned}$$

where a quasi upper triangular matrix has 1×1 or 2×2 diagonal block.

A subspace $\mathcal{V} \subseteq \mathbb{C}^n$ is an **invariant subspace** for $A \in \mathbb{C}^{n\times n}$ if $A\mathcal{V} \subseteq \mathcal{V}$. The complex Schur form satisfies $U_{:,1:k}$ is an orthonormal basis for a k dimensional invariant subspace. This is because

$$AU_{:,1:k} = U_{:,1:k}T_{1:k,1:k}$$

A general invariant subspace basis $V \in \mathbb{C}^{n\times k}$ is of the form

$$AV = VL \qquad L \in \mathbb{C}^{k\times k}$$

Example 0.4. Consider a 2×2 matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The characteristic polynomial is

$$\begin{aligned} p(z) &= \det(A - zI) \\ &= (a - z)(d - z) - bc \\ &= z^2 - \underbrace{(a + d)}_{tr(A)} z + \underbrace{(ad - bc)}_{\det(A)} \end{aligned}$$

There are several cases, depending on $q = b^2 - 4ac = tr(A)^2 - 4\det(A)$.

- $q < 0 \implies$ 2 complex roots
- $q > 0 \implies$ 2 real roots
- $q = 0 \implies$ eigenvalue of algebraic multiplicity 2 Note that the space of matrices where this holds has codimension 1. There are two subcases for this:
  - Degenerate (geometric multiplicity 1)
  - Full geometric multiplicity $AV = V\lambda I \implies A = \lambda I$

We can consider more general eigenvalue problems, like the **generalized eigenvalue problem**

$$Ax = \lambda Mx$$

where M is often taken symmetric positive definite.

Also, we can consider a **nonlinear eigenvalue problem**

$$T(\lambda)x = 0$$

where $T : \mathbb{C} \to \mathbb{C}^{n\times n}$ is analytic.

There are also more specialized eigenvalue problems, such as the symmetric or Hermitian eigenvalue problems, in which $A = A^T$ or $A = A^*$, respectively.

Eigenvalue problems also differ based on their goals:

- All eigenvalues and eigenvectors (or the Schur form)
- A few eigenvalues and eigenvectors
- All eigenvalues, no eigenvectors

Reasons to solve eigenproblems

Eigenproblems can be used for myriad applications:

- Solving nonlinear equations
- Solving optimization problems (maybe approximately)
- Dynamics (linear, constant coefficient)
  - of numerical methods
  - of stochastic processes
  - of physical systems

Example 0.5. Consider a nonlinear equation

$$f(x) = 0 \qquad f : \mathbb{R} \to \mathbb{R} \text{ fairly smooth} \quad x \in [a, b]$$

With a good initial guess, we can run a Newton iteration. Say we do not have a good initial guess. Another method is to approximate $f(x) \approx p(x)$ where $p(x)$ is polynomial. Say we have

$p(x) = x^n - \sum_{k=0}^{n-1} c_{n-k-1} x^{n-k-1}$. Then we can find the roots of polynomial with an eigenvalue algorithm, since they are precisely the eigenvalues of the companion matrix

$$C = \begin{bmatrix} c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix}$$

Indeed, if λ is a root of p, $p(\lambda) = 0$, then with $v = (\lambda^{n-1}, \dots, \lambda, 1)$, we have $Cv = v\lambda$. MATLAB's `roots` computes the roots of a polynomial by running an eigenvalue algorithm on the companion matrix.

Example 0.6. Now, consider the problem of graph partitioning. Say we have a graph G, with labeled nodes $x_i \in \{\pm 1\}$. We want to partition the nodes and consier the quantity $\sum_{(i,j)\in E}(x_i - x_j)^2 = 4 \cdot$ number of edges cut. The graph partitioning problem is

$$\begin{aligned} &\min \frac{1}{4} x^T L x \\ &\text{s.t.} \sum_{j=1}^{n} x_j = 0 \\ &\text{s.t. } x \in \{\pm 1\}^n \end{aligned}$$

meaning we want to minimize the number of edges cut with a partition of the nodes into two equal size groups. This problem is NP-hard. We relax the problem to

$$\begin{aligned} &\min \frac{1}{4} x^T L x \\ &\text{s.t.} \sum_{j=1}^{n} x_j = 0 \\ &\text{s.t. } \|x\|^2 = n \end{aligned}$$

which is a quadratic problem with a quadratic constraint, and one additional constraint. This looks like

$$\min \frac{x^T L x}{x^T x} \qquad e^T x = 0$$

Lecture 22: Perturbation Theory (10/23)

For X invertible, we say A is similar to $X^{-1}AX$, or that A and $X^{-1}AX$ are related by a similarity transform. We make use of the following theorem about the eigenvectors and eigenvalues of certain transformed matrices

Theorem 1 (Spectral mapping theorem)**.** *If $F : \mathbb{C} \to \mathbb{C}$ is analytic on the spectrum of A, $\Lambda(A)$, then $f(A)$ has the same eigenvectors as A, and eigenvalues $f(\lambda)$ for $\lambda \in \Lambda(A)$.*

This is clear in the case where $F \in \mathbb{C}[x]$ is a polynomial. In fact, when $A = V\Lambda V^{-1}$ is diagonalizable,

$$F(A) = \sum_{j=0}^{d} c_j A^j = V F(\Lambda) V^{-1}$$

For a general analytic $F(z) = \sum_{i=0}^{\infty} c_i A^i$, the mapping is given by taking limits and is

$$F(A) = \sum_{j=0}^{\infty} c_j A^j = V F(\Lambda) V^{-1}$$

Observation 0.1.

- Eigenvalues are continuous function of matrix entries
- Eigenvectors are not continuous in general. However, they are continuous locally, away from eigenvalues of higher multiplicity. For instance, consider

  $$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \epsilon \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

  In this case, the eigenvectors are those of $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, and can be chosen arbitrarily by varying a, b, c, d, which does not vary the matrix entries much due to the ϵ scaling.

Example 0.7. Consider the matrix

$$A(\epsilon) = \begin{bmatrix} \lambda & 1 \\ \epsilon & \lambda \end{bmatrix}$$

The characteristic polynomial is $p(z) = z^2 - 2\lambda z + (\lambda^2 - \epsilon)$. The eigenvalues are thus given by $\lambda \pm \sqrt{\epsilon}$. In general, matrices close to Jordan blocks have

For an isolated eigenvalue (i.e. one of multiplicity 1), we have differentiability. Say we have λ an eigenvalue with right eigenvector v and left eigenvector w

$$\begin{aligned} Av &= v\lambda \\ w^* A &= \lambda w^* \end{aligned}$$

Then we differentiate

$$\begin{aligned} \delta A v + A \delta v &= \delta v \lambda + v \delta \lambda \\ (A - \lambda I)\delta v + (\delta A - \delta \lambda I) v &= 0 \\ w^*(A - \lambda I)\delta v + w^*(\delta A - \delta \lambda I) v &= 0 \\ w^*(\delta A - \delta \lambda I) v &= 0 \\ \delta \lambda &= \frac{w^* \delta A v}{w^* v} \end{aligned}$$

We bound the norm

$$\begin{aligned} |\delta \lambda| &\leq \frac{\|w^*\| \|\delta A\| \|v\|}{|w^* v|} \\ &= |\sec(\theta(w, v))| \|\delta A\| \end{aligned}$$

Thus, we can expect $|\delta\lambda|$ to blow up when w and v are nearly orthogonal. To sanity check this, note that for a Jordan block $J = \begin{bmatrix} \lambda & 1 \\ 0 & 1 \end{bmatrix}$, we have that $Je_1 = \lambda e_1$ and $e_2^T J = \lambda e_2^T$. Thus, in this case, the left and right eigenvectors are orthogonal, so the bound does blow up.

Gershgorin

Recall that any diagonally dominant A is invertible. If $A-\lambda I$ is diagonally dominant, then $\lambda \notin \Lambda(A)$. Thus, $\lambda \in \Lambda(A)$ implies that $A-\lambda I$ is not diagonally dominant, meaning there exists some i such that $|a_{ii}-\lambda| \leq \sum_{j\neq i}|a_{ij}|$. This implies that there exists some i such that $\lambda \in B_{\rho_i}(a_{ii})$, where $\rho_i = \sum_{j\neq i}|a_{ij}|$.

Theorem 2 (Gershgorin's Circle Theorem)**.** *Define the Gershgorin disks $\mathcal{G}_1,\ldots,\mathcal{G}_n$ by*

$$\mathcal{G}_i = \Big\{ z \in \mathbb{C} : |a_{ii}-z| \leq \sum_{j\neq i}|a_{ij}| \Big\}$$

Then

- $\Lambda(A) \subseteq \bigcup_{i=0}^{n} \mathcal{G}_i$
- *If $\mathcal{K}$ is a connected component consisting of k disks, then exactly k eigenvalues are inside $\mathcal{K}$.*

Now, write $A = D+F$, where D is the diagonal and F is the off-diagonal. Consider the curve $D+sF$. Then as s goes from 0 to 1, the Gershgorin discs grow from 0 radius to discs of radii given by off-diagonal absolute row-sums of A. This is a nice illustration for how the eigenvalues change as s varies.

—

Suppose we have $A = V\Lambda V^{-1} = V\Lambda W^*$, where $W^* = V^{-1}$. Let $\hat{A} = A+E$. The eigenvalues of $\hat{A}$ lie in Gershgorin discs of

$$V^{-1}\hat{A}V = W^*\hat{A}V = \Lambda + W^*EV$$

Thus, we can bound the difference in the eigenvalues of $\hat{A}$ by looking at the row sums of this matrix. A naive control on the eigenvalues of $\hat{A}$ is given by $\|W^*EV\|_\infty \leq \kappa(V)\|E\|_\infty$, so we know that

$$\Lambda(\hat{A}) \subseteq \bigcup_{i=1}^{n} B_{\kappa(V)\|E\|_\infty}\Big(\lambda_i(A)\Big)$$

A more refined analysis (as in the notes) gives a nicer bound

$$\Lambda(\hat{A}) \subseteq \bigcup_{i=1}^{n} B_{n\sec(\theta_i)\|E\|_2}\Big(\lambda_i(A)\Big)$$

where θ_i is the acute angle between the left and right eigenvectors for λ_i.

Lecture 23: Power iterations (10/25)

Let $A \in \mathbb{C}^{n\times n}$, with eigenvalues labeled so that $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. The eigen equations are $Ax_i = \lambda_i x_i$. We also suppose that $A = V\Lambda V^{-1}$ is diagonalizable.

A naive method of computing eigenvalues is to find roots of the characteristic polynomial $\det(A-xI)$. This is not a good method for actually computing the eigenvalues. The solutions to a high degree polynomial may change a lot with small perturbations to the coefficients.

Power method

We use that $A^k = V\Lambda^k V^{-1}$ or $A^k V = V\Lambda^k$. Take $x = V\tilde{x}$. Then we have

$$\begin{aligned} A^k x &= A^k V \tilde{x} \\ &= V\Lambda^k \tilde{x} \\ &= \sum_j v_j \lambda_j^k \tilde{x}_j \\ &= \lambda_1^k \sum_j v_j \Big(\frac{\lambda_j}{\lambda_1}\Big)^k \tilde{x}_j \end{aligned}$$

If we suppose that $|\lambda_1|$ is strictly greater than the magnitudes of the other eigenvalues, then the only summand that is not killed is at $j = 1$. Thus, we consider the following power iteration

$$x^{(k+1)} = \frac{Ax^{(k)}}{\left\|Ax^{(k)}\right\|} = \frac{A^k x^{(0)}}{\left\|A^k x^{(0)}\right\|}$$

Then under our assumptions, the iterates converge to the dominant eigenvector. Note that the convergence is linear, depending on the ratio $\frac{\lambda_2}{\lambda_1}$. To obtain λ_1, we then just apply $Ax^{(n)} \approx \lambda_1 x_1$. There are some issues with using this approach exactly:

1. We do not find $\lambda_j v_j$ for $j \neq 1$
2. What if $\frac{|\lambda_2|}{|\lambda_1|} \approx 1$?
3. What if $\tilde{x}_1 = 0$?

   This is not too terrible of an issue, since for one a random choice of vector in $\mathbb{R}^n$ almost surely has nonzero first component. Also, rounding issues on a computer could make the component nonzero and accidentally resolve this.

Choose $f(z) = \frac{1}{z-\sigma}$. Then we have $(A-\sigma I)^{-1} = f(A) = V(\Lambda - \sigma I)^{-1}V^{-1}$. Then the eigenvalues of $f(A)$ are $\frac{1}{\lambda_j - \sigma}$. Thus, the maximal eigenvalue of $f(A)$ is $\frac{1}{\lambda_j - \sigma}$ where λ_j is the eigenvalue closest in σ. Again, convergence is linear. We now consider a way to speed up the convergence.

Say we have run some iterations of the power method, and have an iterate $\hat{v}$ that is close to an eigenvector of A. Then we have $A\hat{v} - \hat{\lambda}\hat{v} \approx 0$. Applying $\hat{v}^T$ on both sides of $Av - \hat{\lambda}v = 0$, we get

$$\hat{v}^T A v - \hat{\lambda}\hat{v}^T v = 0$$

so that

$$\hat{\lambda} \approx \frac{\hat{v}^T A \hat{v}}{\hat{v}^T v}$$

Thus, at each step of our shift and invert iteration, we choose our shift to be the Rayleigh quotient ($\frac{\hat{v}^T A \hat{v}}{\hat{v}^T \hat{v}}$), an approximation of the algorithm. With this Rayleigh quotient iteration, we now have quadratic convergence.

Subspace iteration

We can in some sense extend the power method by applying A to a subspace. $\mathcal{V}_k = A^k \mathcal{V}_0$. Directly applying A to a set of vectors V just makes all iterates converge to the dominant eigenvector, so we have to make some adjustments.

Say we have an orthogonal set of p vectors $Q_k \in \mathbb{C}^{n \times n}$, and compute an economy QR decomposition

$$AQ_k = Q_{k+1} R_{k+1}$$
$$q_1^{(k+1)} r_{1,1}^{(k+1)} = A q_1^{(k)}$$

As long as $|\lambda_p| > |\lambda_{p+1}|$, we have convergence of the iterates $q_j^{(k)}$ to eigenvectors.

Lecture 24: QR Method (10/28)

Consider the shift-invert iteration, and suppose σ is close to an eigenvalue. We have the iterate $v^{(k+1)} \propto (A - \sigma I)^{-1} v^{(k)}$. Note that $A - \sigma I$ is ill-conditioned. However, it turns out that the error in the solve mostly "points in the direction of the eigenvector" so it is not terrible.

Now, we discuss subspace iteration more in depth. The iterates are

$$\mathcal{V}^{k+1} = A\mathcal{V}^k \qquad\qquad Q^{k+1} R^{k+1} = AQ^k$$

Consider the first column

$$q_1^{k+1} r_{11}^{k+1} = A q_1^k$$

Thus, the first column of the Q^k are following a power iteration. The first two columns of Q are following a two dimensional subspace iteration, and so on.

Now, say we have iterates $Q^{k+1} R^{k+1} = AQ^k$, where this is the full QR factorization, so $Q \in \mathbb{R}^{n \times n}$. If the eigenvalues have distinct magnitudes, then $Q^k \to Q$, the orthogonal factor in the Schur form $AQ = QT$. Moreover, we have that $R^k \to T$.

There are some issues with this iteration:

- We want to focus on eigenvalues, not eigenvectors, so we want to focus on the triangular factor T
- The cost per iteration is $O(n^3)$
- It may converge slowly, depending on the difference of the magnitudes of the eigenvalues

We attempt to solve some of these problems. Our iteration is of the form (starting with $Q^0 = I$)

$$Q^1 R^1 = A$$
$$Q^2 R^2 = AQ^1$$
$$Q^2 R^2 R^1 = AQ^1 R^1 = A^2$$
$$Q^3 R^3 = AQ^2$$
$$Q^3 R^3 R^2 R^1 = AQ^2 R^2 R^1 = A^3$$
$$Q^k R^k \cdots R^1 = A^k$$

Note that this means at step k, we are implicity computing a QR factorization of A^k.

$$\begin{aligned} Q^{(k+1)}R^{(k+1)} &= AQ^k \\ Q^{(k+1)}R^{(k+1)}\cdots R^{(1)} &= A^{k+1} \end{aligned}$$

say we are trying to extract a good possible estimate of T. We select the estimate

$$\begin{aligned} A^{(k)} &= (Q^{(k)})^* A Q^{(k)} \\ A^{(k)} &= (Q^{(k)})^* Q^{(k+1)} R^{(k+1)} \\ &= \tilde{Q}^{(k+1)} R^{(k+1)} \\ A^{(k+1)} &= (Q^{(k+1)})^* A Q^{(k+1)} \\ &= (\tilde{Q}^{(k)})^* A^{(k)} \tilde{Q}^{(k)} \\ &= R^{(k)} \tilde{Q}^{(k)} \end{aligned}$$

Thus, our equivalent iteration is

$$\begin{aligned} A^{(k)} &= \tilde{Q}^{(k)} R^{(k)} \\ A^{(k+1)} &= R^{(k)} \tilde{Q}^{(k)} \end{aligned}$$

Note that the steps are still $O(n^3)$. To resolve this, we transform A into a nicer matrix to work with, by using similarity transforms so that the eigenvalues are preserved. In particular, we use unitary similarities. Applying Householder transforms on both sides can bring A to upper Hessenberg form (note that we cannot expect transformations that bring us directly to a Schur form, due to Abel-Ruffini).

Considering the iteration on a Hessenberg matrix,

$$\begin{aligned} H^{(k)} &= Q^{(k+1)} R^{(k+1)} \\ H^{(k+1)} &= R^{(k+1)} Q^{(k+1)} \quad \text{is still Hessenberg!} \end{aligned}$$

Going through the steps of Householder QR on $H^{(k)}$, the Householder transforms are just Givens rotations on the subdiagonal. Then, right multiplying $R^{(k+1)}$ by $Q^{(k+1)}$ just applies the Givens rotations to the columns, which only affects the nonzero structure of the subdiagonal of $R^{(k+1)}$, thus guaranteeing the upper Hessenberg structure of $H^{(k+1)}$. Note also that the Givens rotations only take $O(n^2)$ time per iteration!

Lecture 25: Practical QR Method (10/30)

The first column of an upper triangular matrix is special since it only contains one nonzero. Likewise, the last row contains only one nonzero. Note that in the QR iteration, we have

$$\begin{aligned} A^k &= Q^{(k)} \tilde{R}^{(k)} \\ A^{-k} &= (\tilde{R}^{(k)})^{-1} Q^{(k)^T} \\ e_n^T A^{-k} &= \underbrace{(\tilde{R}^k)^{-1}_{nn}}_{\text{a scalar}} e_n^T Q^{(k)T} \end{aligned}$$

Thus, the last column of the iterates $Q^{(k)}$ follow an inverse iteration. Recall that the first columns follow a power iteration. This motivates the shifted QR iteration.

$$\begin{aligned} Q^{(k)}R^{(k)} &= H^{(K)} - \sigma I \\ H^{(k+1)} &= R^{(k)}Q^{(k)} + \sigma I \end{aligned}$$

To check that this is an equivalent iteration, note that

$$\begin{aligned} H^{(k+1)} &= R^{(k)}Q^{(k)} + \sigma I \\ &= Q^{(k)T}(H^{(k)} - \sigma I)Q^{(k)} + \sigma I \\ &= Q^{(k)T}H^{(k)}Q^{(k)} \end{aligned}$$

This accelerates the convergence, if we choose good shifts. The simplest choice of shift is $A^{(k)}_{nn}$, which is equivalent to a Rayleigh quotient iteration in some sense. This is known as the **Wilkinson shift**. During the iteration, the subdiagonal element of the last row approaches zero, so when it becomes smaller than some tolerance, we can take the bottom right entry as an eigenvalue, then deflate and continue the algorithm on the smaller submatrix.

Note that for a real $A \in \mathbb{R}^{n\times n}$, all elements in the iterates are real, and thus we can never truly converge to an eigenvalue $\lambda \in \mathbb{C} \setminus \mathbb{R}$. However, all complex eigenvalues of real matrices do come in conjugate pairs. We can choose a complex shift to help resolve this.

To converge to an eigenvalue near σ, we iterate with $(A - \sigma I)^{-1}$. An iteration can converge to a 2 dimensional subspace associated with a conjugate pair $\lambda, \overline{\lambda}$ of maximum modulus. Consider the mapping

$$\begin{aligned} z &\mapsto \frac{1}{(z-\sigma)(z-\overline{\sigma})} \\ &= \frac{1}{z^2 - 2\Re(\sigma)z + |\sigma|^2} \end{aligned}$$

Thus, we iterate with $(A^2 - 2\Re(\sigma)A + |\sigma|^2 I)^{-1}$. This is known as the **Francis double-shift** strategy. This can be done purely with real arithmetic.

Let $U \in \mathbb{R}^{n\times 2}$ be an orthonormal basis for the eigenspace associated with a conjugate pair of eigenvalues. Consider the matrix

$$U^T A U = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

We want to choose shifts as eigenvalues of $U^T AU$. Note that we can just use the characteristic polynomial $z^2 - (\alpha + \beta)z + (\alpha\delta - \beta\gamma)$.

Note that our new iterating matrix uses A^2. This is not still upper Hessenberg, but has just one more subdiagonal of nonzeros. We have to use another trick to resolve this.

First, note that there are many Hessenberg matrices H orthogonally similar to a given matrix A. The **implicit Q theorem** says that the orthogonal Q such that $Q^T AQ = H$ is entirely determined by its first column. This means that the first column can be taken arbitrarily (of unit norm), and then the rest of the columns are determined.

The **bulge chasing** strategy applies transformation that eliminates starting from the left-most column, which may introduce zeros in columns to the right, but which are iteratively eliminated in later steps.

Lecture 26: Symmetric Eigenproblem Theory (11/1)

We first recall some useful facts for the homework. For $x = A^{-1}b$,

$$\begin{aligned}\delta x &= -A^{-1}\delta A A^{-1}b + A^{-1}\delta b \\ &= -A^{-1}\delta A x + A^{-1}\delta b\end{aligned}$$

There is a standard mapping $\mathbb{C} \to \mathbb{R}^{2\times 2}$ given by

$$a + \beta i = e^{i\theta} \mapsto \begin{bmatrix}\alpha & -\beta \\ \beta & \alpha\end{bmatrix} = \rho \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$$

Symmetric Eigenvalue Problem

The symmetric eigenvalue problem (SEP) is

$$\begin{aligned}A \in \mathbb{R}^{n\times n},\ A = A^T \\ Ax = x\lambda\end{aligned}$$

The Hermitian eigenvalue problem (HEP) is defined similarly

$$\begin{aligned}A \in \mathbb{C}^{n\times n},\ A = A^* \\ Ax = \lambda x\end{aligned}$$

There are some nice properties of the hermitian eigenvalue problem that do not hold in general:

- Eigenvalues are all real
- Complete basis of orthonormal eigenvectors
- All eigenvalues have maximal geometric multiplicity

The decomposition $A = U\Lambda U^*$ is both a Jordan and Schur form. Note also that $A = U(\Lambda S)(SU^*)$, where $S = \operatorname{diag}(s(\lambda_1), \ldots, s(\lambda_n))$, where $s(a)$ is the sign of a for $a \in \mathbb{R}$.

Recall that the Rayleigh quotient for $v \neq 0$ is

$$\rho(A, v) = \frac{v^*Av}{v^*v}$$

Let us minimize or maximize a Rayleigh quotient. We differentiate

$$\begin{aligned}\delta\Big(\frac{v^*Av}{v^*v}\Big) &= \frac{\delta(v^*Av)(v^*v) - (v^*Av)\delta(v^*v)}{(v^*v)^2} \\ &= \frac{\delta v^*\Big(2Av(v^*v) - (v^*Av)2v\Big)}{(v^*v)^2} \\ &= \frac{2\delta v^*}{\|v\|^2}\Big(Av - \rho(A,v)v\Big)\end{aligned}$$

Thus, stationary points of the Rayleigh quotient are points where $Av = \rho(A, v)v$, meaning points where v is an eigenvector with corresponding eigenvalue $\rho(A, v)$.

Thus, we consider the constrained optimization $\min / \max v^T Av \quad \text{s.t.} \|v\|^2 = 1$.

$$\mathcal{L}(v, \lambda) = v^T Av - \lambda(\|v\|^2 - 1)$$
$$\delta\mathcal{L} = 2\delta^T(Av - \lambda v) - \delta\lambda(\|v\|^2 - 1)$$

Thus, the Lagrange multiplier is the eigenvalue.

Now, consider another constraint

$$\min / \max v^T Av \quad \text{s.t.} \|v\|_M^2 = 1$$

Then we can solve

$$\mathcal{L}(v, \lambda) = \frac{1}{2}v^T Av - \frac{\lambda}{2}(v^T Mv - 1)$$
$$\delta\mathcal{L} = \delta v^T(Av - \lambda Mv) - \frac{\delta\lambda}{2}(v^T Mv - 1)$$

Thus, we now have a (symmetric) generalized eigenvalue problem. If $M = R^T R$, then $v^T Mv = v^T R^T Rv = \|Rv\|^2$. Letting $w = Rv$, we have $v = R^{-1}w$. Thus, this problem is equivalent to

$$\min / \max w^T R^{-T} AR^{-1} w \quad \text{s.t.} \|w\|^2 = 1$$

So we can convert a symmetric generalized eigenvalue problem to a standard symmetric eigenvalue problem. We do not always do this in practice.

Now, note that if $A = U\Lambda U^*$ is an eigendecomposition,

$$\begin{aligned} v^* Av &= v^* U\Lambda U^* v \\ &= (U^* v)^* \Lambda \underbrace{(U^* v)}_{\tilde{V}} \\ &= \sum_{j=1}^{n} \lambda_j \tilde{v}_{j,j}^2 \end{aligned}$$

Recall that similarity preserves eigenvalues. The natural set of transformations for quadratic forms is congruence, which is of the form $A \mapsto X^* AX, \quad X \in GL_n$. General congruence preserves the inertia of A, meaning the respective counts of positive, zero, and negative eigenvalues.

Now, if $A = U\Sigma V^*$, then

$$A^* A = V\Sigma^2 V^*$$
$$AA^* = U\Sigma^2 U^*$$

Consider the matrix

$$\begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} Av \\ A^* u \end{bmatrix} = \sigma \begin{bmatrix} u \\ v \end{bmatrix}$$

For a singular value/ vector triple (σ, u, v). This shows that the SVD can be computed by way of SEP.

Now, say we are using a shift-invert iteration on a symmetric matrix. Then we has as shift

$$\rho(A, v + \epsilon u) = \rho(A, v) + O(\epsilon^2)$$

In the nonsymmetric case, the shift is $O(\epsilon)$ close. Thus, the symmetric structure significantly benefits the convergence of the iteration.

Lecture 27: Symmetric Eigenproblem Theory Cont. (11/4)

Theorem 3 (Courant-Fisher Minimax Theorem)**.** *Let $A = A^T \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$. Then*

$$\begin{aligned}\lambda_k &= \max_{\mathcal{V}:\dim\mathcal{V}=k} \min_{v\in\mathcal{V}\setminus\{0\}} \frac{v^TAv}{v^Tv} \\ &= \min_{\mathcal{V}:\dim\mathcal{V}=n-k+1} \max_{v\in\mathcal{V}\setminus\{0\}} \frac{v^TAv}{v^Tv}\end{aligned}$$

Sketch of proof. Let $A = Q\Lambda Q^T$. Then

$$\begin{aligned}\rho(A,v) &= \frac{v^TAv}{v^Tv} \\ &= \frac{v^TQ\Lambda Q^Tv}{v^TQ^TQv} \\ &= \frac{\tilde{v}^T\Lambda\tilde{v}}{\tilde{v}^T\tilde{v}} && \tilde{v} = Qv \\ &= \frac{1}{\sum_j \tilde{v}_j^2}\Big(\sum_k \lambda_k\tilde{v}_k^2\Big) \\ &= \sum_{k=1}^n \lambda_j w_j && \sum_j w_j = 1,\ w_j \geq 0\end{aligned}$$

Thus, the Rayleigh quotient is a weighted average over the eigenvalues. We can change what the weights are. Now, for λ_k, a maximal choice of subspace is one which is spanned by the first k eigenvectors. The minimal Rayleigh quotient out of all vectors of this subspace is equal to the smallest eigenvalue in this subspace, λ_k. □

An application of Courant-Fisher is the Cauchy interlace theorem

Theorem 4. *Let A be a symmetric matrix, with blocks*

$$A = \begin{bmatrix} A_{11} & a_{12} \\ a_{21} & a_{22}\end{bmatrix}$$

Suppose $\lambda_1 \geq \ldots \geq \lambda_n$ are eigenvalues of A and $\tilde{\lambda}_1 \geq \ldots \geq \tilde{\lambda}_{n-1}$ are eigenvalues of A_{11}. Then

$$\lambda_1 \geq \tilde{\lambda}_1 \geq \lambda_2 \geq \tilde{\lambda}_2 \geq \ldots \geq \lambda_{n-1} \geq \tilde{\lambda}_{n-1} \geq \lambda_n$$

Sketch of proof.

$$\begin{aligned}\lambda_k &= \max_{\dim\mathcal{V}=k} \min_{v\in\mathcal{V}\neq\{0\}} \rho(A,v) \\ \tilde{\lambda}_k &= \max_{\substack{\dim\mathcal{V}=k \\ v_n=0}} \min_{v\in\mathcal{V}\neq\{0\}} \rho(A,v)\end{aligned}$$

So $\lambda_k \geq \tilde{\lambda}_k$ due to the extra constraint in the max of $\tilde{\lambda}_k$. Use the other direction equality in Courant-Fisher to get $\tilde{\lambda}_k \geq \lambda_{k+1}$. □

Oftentimes roots of orthogonal polynomials can be characterized as eigenproblems for nested matrices, so this theorem can be applied to show that there is interlacing of some sort.

Now, we consider perturbation results, which are much nicer than those applying to the general eigenproblem. The Weyl bound is:

$$\left|\lambda_k(A+E)-\lambda_k(A)\right| \leq \|E\|_2$$

This is because

$$\begin{aligned}\rho(A+E,v) &= \frac{v^TAv}{v^Tv}+\frac{v^TEv}{v^Tv}\\ &\leq \frac{v^TAv}{v^Tv}+|\lambda_1(E)|\\ &= \frac{v^TAv}{v^Tv}+\|E\|_2\end{aligned}$$

The Weilandt-Hoffman theorem states:

$$\sum_k \Big(\lambda_k(A+E)-\lambda_k(A)\Big)^2 \leq \|E\|_F^2$$

Note that this measures the total difference between the eigenvalues of $A+E$ and A while the Weyl bound measures the eigenvalue by eigenvalue difference.

Now, say that we have an approximate eigenvalue $\hat{v}$ with

$$A\hat{v} = \hat{v}\hat{\lambda}+r$$

Then it holds that

$$(A+E)\hat{v} = \hat{v}\hat{\lambda} \qquad E = r\hat{v}^*+\hat{v}r^*$$

Meaning that there is a symmetric perturbation that scales in size with the size of the residual for which we have an exact eigenvector to a perturbed problem.

Now, we present the Davis-Kahan $\sin\theta$ theorem.

Theorem 5. *Let $AU = U\Lambda$, where $U \in \mathbb{R}^{n\times k}$ and $\Lambda \in \mathbb{R}^{k\times k}$. This means that U spans a k-dimensional invariant subspace of A. Say also $\hat{A}\hat{U} = \hat{U}\hat{\Lambda}$. Define a residual $\hat{A}U - U\Lambda$. Then*

$$\left\|\sin\theta(U_1,\hat{U})\right\|_F \leq \frac{\|R\|_F}{\delta}$$

where δ is the gap between Λ and the rest of the eigenvalues.

The sin and angle is defined in terms of a CS decomposition. Let $Q = \begin{bmatrix}Q_1 & Q_2\end{bmatrix}$ orthogonal. We

want $\tilde{Q} = \begin{bmatrix}\tilde{Q}_1 & \tilde{Q}_2\end{bmatrix}$. We decompose

$$Q^T\tilde{Q} = \begin{bmatrix}U_1 & \\ & U_2\end{bmatrix}\begin{bmatrix}C & -S \\ S & C\end{bmatrix}\begin{bmatrix}V_1 & \\ & V_2\end{bmatrix}$$

$$C = \begin{bmatrix}\cos\theta_1 & & \\ & \ddots & \\ & & \cos\theta_k\end{bmatrix}$$

$$S = \begin{bmatrix}\sin\theta_1 & & \\ & \ddots & \\ & & \sin\theta_k\end{bmatrix}$$

This decomposes $\tilde{Q}_1^T Q_2 = U\Sigma V^T$, where $\sigma_j = \cos\theta_j$, called the jth canonical angle.

This theorem means that if A has a large gap in its eigenvalues, then regardless of any change in the eigenvectors, the span of a eigenspace of the perturbed matrix is not much different from the corresponding span of that of A.

Lecture 28: Symmetric Eigenproblem Algorithms (11/6)

First, suppose we reduce $A = A^T = QHQ^T$ where H is upper Hessenberg. Since A is symmetric, this means that $QHQ^T = A = A^T = QH^TQ^T$, meaning that $H = H^T$, so that H is in fact symmetric tridiagonal. Thus, we denote $H = T$. Let $\alpha_1, \ldots, \alpha_n$ be the diagonal elements, and $\beta_1, \ldots, \beta_{n-1}$ be the subdiagonal elements.

Note that the QR algorithm preserves tridiagonality in the iterates by using orthogonal similarities. Moreover, we can apply bulge-chasing to compute QR decompositions of a banded matrix, using Given's rotations. This costs $O(n)$ time as opposed to $O(n^2)$ time per iteration in the nonsymmetric case. Thus, the total cost per iteration of symmetric QR is $O(n)$. Moreover, we can shift with the Wilkinson shift (Rayleigh quotient iteration on last column) since all eigenvalues are real.

We consider the form of the Rayleigh quotient iteration in the symmetric case. Let v_o bet the initial guess, with error $O(\epsilon)$. Then

$$\begin{aligned}\sigma &= \rho(A, v_o) \\ &= \lambda + O(\epsilon^2)\end{aligned}$$

Thus, plugging in this shift, we get a new estimate

$$\begin{aligned}v_p &= (A - \sigma I)^{-1} v_o \\ &= v_{\text{truth}} + O(\epsilon^3)\end{aligned}$$

Indeed, the Rayleigh quotient iteration converges insanely quickly in the symmetric case.

The total cost for symmetric QR algorithms is $O(n)$ steps (due to this fast convergence), $O(n)$ cost per step, and thus $O(n^2)$ overall cost, once we have a tridiagonal matrix. However, tridiagonalizing the matrix still takes $O(n^3)$ time.

Lastly, we note that the symmetric QR can be used to compute an SVD. The exact method is a bit more complicated.

Jacobi iteration

A **Jacobi rotation** is like a Given's rotation but applied to both sides of a matrix.

$$J^T A J = \Lambda \qquad \text{all in } \mathbb{R}^{2\times 2}$$

$$J = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

The Jacobi iteration is of the form:

- Pass through all off diagonals and apply Jacobi rotation to $\begin{bmatrix} A_{kk} & A_{kl} \\ A_{lk} & A_{ll} \end{bmatrix}$
  - $O(1)$ to compute J and $O(n)$ to apply across all of A

When a Jacobi rotation is applied, previous off diagonals may be reintroduced, but they will be of lower magnitude. This is much easier to parallelize than the QR algorithm.

Divide and conquer

Consider the tridiagonal eigenproblem. Say we only want some subset of the eigenvalues. If we have a good estimate $\sigma \approx \lambda_1$, then Rayleigh quotient iteration converges cubically with $O(n)$ cost per iteration. The key idea is the linear solve with $A - \sigma I$. If this were spd, then we could use Cholesky. If not, then we can compute an LDL^T factorization.

$$P(A-\sigma I)^{-1}P^T = LDL^T$$

Note that $A-\sigma I$ and D are congruent. Thus, they have the same inertia, meaning that the number of positive, zero, or negative d_{ii} are preserved. Thus, we have the number of eigenvalues of A that are $> \sigma$, number that are $= \sigma$, and number that are $< \sigma$.

Thus, the bisection idea is to get sufficient information to get a set of intervals containing eigenvalues. Once the intervals are small enough, we have good eigenvalue estimates, and can run Rayleigh quotient iterations to converge more closely. Moreover, we can use the interlacing theorem to get further information in some way.

The so-called Grail code has optimal complexity for computing eigenvectors once given the eigenvalues. To compute k eigenvectors, which contain $O(nk)$ data, it takes $O(nk)$ time.

Lecture 29: (11/8)

We review the connection between the second derivative test for Lagrange multipliers and the symmetric eigenvalue problem. Consider an optimization problem with equality constraints,

$$\min \varphi(x) \quad s.t.\ g(x) = 0$$

in which $\varphi : \mathbb{R}^{\rightarrow}\mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^k$, $k < n$. We form

$$L(x, \lambda) = \varphi(x) + \lambda^T g(x)$$

Then differentiating with respect to x,

$$\delta L = [\varphi'(x) + \lambda^T g'(x)]\delta x + \delta\lambda^T g(x)$$

At a constrained stationary point where $\delta L(x^*, \lambda^*) = 0$, we look at directions consistent with $g(x) = 0$.

$$\begin{aligned} & g(x + \epsilon v) = O(\epsilon^2) \\ \text{i.e.} \quad & \underbrace{g'(x^*)}_{\in \mathbb{R}^{k\times n}} v = 0 \end{aligned}$$

Then for a constrained minimum, we need

$$\begin{aligned} & v^T L''(x^*) v >= 0, \quad v \neq 0 \\ & \text{s.t. } g'(x^*) v = 0 \end{aligned}$$

For example, consider the problem

$$\begin{aligned} \varphi(x) &= x^T A x \\ g(x) &= x^T x - 1 \end{aligned}$$

Then we compute

$$\begin{aligned} L(x, \mu) &= x^T A x - \mu(x^T x - 1) \\ \delta L &= 2\delta x^T (Ax - \mu x) - \delta\mu^T (x^T x - 1) \end{aligned}$$

Our second derivative test is

$$v^T (A - \mu I) v > 0 \quad \text{s.t. } x^T v = 0$$

Sylvester's equation

Now, moving on, consider Sylvester's equation

$$AX + XB = C$$

$$A \in \mathbb{R}^{n\times n},\ X \in \mathbb{R}^{n\times k},\ B \in \mathbb{R}^{k\times k},\ C \in \mathbb{R}^{n\times k}$$

We can rewrite this as a linear system using Kronecker products and vec, which takes a matrix and lays it out in stacking columns vertically. Note that

$$\begin{aligned} \text{vec}(AX) &= \text{vec}(\begin{bmatrix} Ax_1, \ldots, Ax_n \end{bmatrix}) \\ &= \begin{bmatrix} Ax_1 \\ \vdots \\ Ax_n \end{bmatrix} \\ &= \begin{bmatrix} A & & \\ & \ddots & \\ & & A \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= (I \otimes A)\text{vec}(X) \end{aligned}$$

thus, Sylvester's equation is

$$(I \otimes A + B^T \otimes I)\text{vec}(X) = \text{vec}(C)$$

Of course, this is a massive system takes $O((nk)^3)$ time to solve

We find that a discretization for computing Laplacians also takes this form

$$\nabla^2 f(x,y) \approx \frac{f(x+h,y) - 2f(x,y) + f(x-h,y)}{2h^2} + \frac{f(x,y+h) - 2f(x,y) + f(x,y-h)}{2h^2}$$

say we have a grid of function evaluations, where $U_{ij} = u(x_i, y_i)$. Then we have the approximation

$$(\nabla^2 u)_{ij} \approx \frac{-1}{h^2}(I \otimes T + T \otimes I)\text{vec}(U)$$

$$T = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

Now, note that if $A = \text{diag}(\alpha_1, \ldots, \alpha_n)$ and $B = \text{diag}(\beta_1, \ldots, \beta_n)$, then the solution to Sylvester's equation is $X_{ij} = \frac{c_{ij}}{\alpha_i + \beta_j}$. Thus, we would like to reduce matrices to diagonal form. Say that A and B are symmetric and we have eigendecompositions $A = Q_A \Lambda_A Q_A^T$ and $B = Q_B \Lambda_B Q_B^T$. Then our equation is

$$\begin{aligned} Q_A \Lambda_A Q_A^T X + X Q_B \Lambda_B Q_B^T &= C \\ \Lambda_A Q_A^T X Q_B + Q_A^T X Q_B \Lambda_B &= Q_A^T C Q_B \\ \Lambda_A \tilde{X} + \tilde{X} \Lambda_B &= \tilde{C} \end{aligned}$$

then the time to solve Sylvester's equation is simply the cost of the eigendecompositions $O(n^3)$, where we assume $n \geq k$.

If A or B are not symmetric, we can use a similar method with the Schur forms $A = U_A T_A U_A^*$ and $B = U_B T_B U_B^*$. Then Sylvester's equation is

$$T_A \tilde{X} + \tilde{X} T_B = \tilde{C}$$

Note that T_A and T_B are triangular, not diagonal. The Bartels-Stewart algorithm can solve this system. We look at each column one by one

$$\begin{aligned} \text{First column: } T_A \tilde{x}_1 + \tilde{x}_1 (T_B)_{11} &= \tilde{C}_1 \\ (T_A + (T_B)_{11}) \tilde{x}_1 &= \tilde{C}_1 \end{aligned}$$

Note that this is a triangular system, so we can solve it in $O(n^2)$ time. The other columns are similar, so this takes $O(n^3)$ time total.

Riccati equations

Sylvester's equation is a linear equation of matrices. Riccati's is a quadratic matrix equation

$$A^T X + X A - X B R^{-1} B^T X + Q = 0$$

This is equivalent to

$$\begin{bmatrix} I & X^T \end{bmatrix} \begin{bmatrix} Q & A \\ A^T & -BR^{-1}B \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = 0$$

$$\begin{bmatrix} Q & A \\ A^T & -BR^{-1}B \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} X \\ -I \end{bmatrix}$$

$$= \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix}}_{Z} \begin{bmatrix} I \\ X \end{bmatrix} = 0$$

So $\begin{bmatrix} I \\ X \end{bmatrix}$ is a null space basis.

Lecture 30: Iterative Methods and Model Problems (11/13)

(Monday was Veteran's day).

We are moving on to iterative methods for linear systems and iterative methods for eigenproblems. Here, sparsity (meaning data sparsity) is very important. The actual nonzero structure of the matrices are important; the methods generally do not work well with general sparse matrices. Thus, we consider performance on model problems. One important model problem that we will consider is the second-order finite difference discretization of Poisson's equation. This takes the form

$$-\frac{d^2u}{dx^2} = f \quad \text{on } (0,1), \qquad u(0) = u(1) = 0$$

we take a mesh $x_0, \ldots, x_N$ where $x_j = jh$ so that the steps are $h = 1/N$. We wish to approximate $u_j \approx u(x_j)$. Denote $f_j = f(x_j)$. Taylor expansion gives

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + O(h^4)$$

$$u(x-h) = u(x) - u'(x)h + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 - O(h^4)$$

$$u(x+h) + u(x-h) = 2u(x) + u''(x)h^2 + O(h^4)$$

so that our approximation is

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2)$$

so we have the equations

$$-u_j'' \approx \frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f_j \qquad j = 1, \ldots, N-1$$

$$-u_{j-1} + 2u_j - u_{j+1} = h^2 f_j$$

which is the linear system

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{bmatrix}$$

we denote a matrix of this tridiagonal structure of size $N \times N$ by T_N.

We consider an associated eigenproblem $(T - \lambda I)\psi = 0$. The row-wise equation is

$$-\psi_{j-1} + (2-\lambda)\psi_j - \psi_{j+1} = 0$$

so we consider the characteristic equation

$$z^2 - (2-\lambda)z + 1 = 0$$

we get two conjugate roots ξ, $\overline{\xi}$ with $|\xi| = 1$ when $\lambda \in [0,4]$. The solution is then of the form $\psi_j = \alpha\xi^j + \beta\overline{\xi}^j$, which we can solve as $\hat{\alpha}\cos(j\theta) + \hat{\beta}\sin(j\theta)$ using the polar decomposition of ξ. We can use the boundary conditions to get the exact eigensolutions.

$$\text{eigenvectors } z_j(k) = \sqrt{\frac{2}{n+1}} \sin\Big((j\pi)x_k\Big)$$
$$\text{eigenvalues } \lambda_j = 2\Big(1 - \cos\Big(\frac{\pi j}{n+1}\Big)\Big)$$

If we consider $j \ll n$, then Taylor expansion gives

$$\begin{aligned} \cos\Big(\frac{j\pi}{n+1}\Big) &\approx 1 - \frac{1}{2}\Big(\frac{j\pi}{n+1}\Big)^2 \\ &= 1 - \frac{\pi^2}{2}(jh)^2 + O((jh)^4) \end{aligned}$$

so that we have

$$\lambda_j \approx h^2(\pi j)^2$$

Now, we consider the $2D$ model problem, which is of the form

$$-\nabla^2 u = -(\partial_{xx}u + \partial_{yy}u) = f \qquad (x,y) \in (0,1)^2$$

$$u(x,y) = 0 \quad \text{for } |x| = 1 \text{ or } |y| = 1$$

Then we want approximations

$$U_{ij} \approx u(x_i, y_j)$$

Discretizing gives a Sylvester equation

$$TU + UT = h^2 F$$
$$(T \otimes I + I \otimes T)\text{vec}(U) = h^2 \text{vec}(F)$$

which is still a matrix equation of dimension $N = n^2$. However, the 3D equation is not a matrix equation:

$$(T \otimes I \otimes I + I \otimes T \otimes I + I \otimes I \otimes T)\text{vec}(U) = h^2 \text{vec}(F)$$

The nonzero pattern of $T_{N\times N}$, the matrix for the $2D$ equation, is block tridiagonal, with the diagonal blocks having tridiagonal structure and the sub/super-diagonal blocks have diagonal structure.

We have various direct methods to solve with $T_{N\times N}$:

- Dense Cholesky $O(N^3)$

  This is the slow, naive method.

- Banded Cholesky $O(N^{2.5})$

- Nested dissection $O(N^{1.5})$

  This is a pretty good method, and can make good use of caches.

  The above methods all use the nonzero structure.

- Sylvester solver $O(N^{1.5})$ factorization and $O(N)$ solves.

  This makes use of the Kronecker product structure.

- Discrete sine transforms $O(N \log N)$

There are also iterative methods (listed are amount of work to decrease error by a constant factor)

- Jacobi $O(N^2)$
- Gauss-Seidel $O(N^2)$
- Conjugate Gradients $O(N^{3/2})$
- SOR $O(N^{3/2})$
- SSOR with Chebyshev acceleration $O(N^{5/4})$
- Multigrid $O(N)$

Lecture 31: Iterative Methods (11/15)

Our goal is to solve $Ax = b$ by iteration, in which we produce $x_0, x_1, \ldots$ that converge to $x^* = A^{-1}b$. Today we will be consider stationary methods / fixed point iterations. For these, we have a function F so that $x^* = F(x^*)$. Then we consider an iteration $x_{k+1} = F(x_k)$. We analyze this by considering

$$x_{k+1} - x^* := e_{k+1} = F(x^* + e_k) - F(x^*)$$

We desire contractivity, meaning

$$\|e_{k+1}\| \leq \alpha \|e_k\|$$

for some $\alpha \in (0,1)$.

For a matrix A, we take a splitting $A = M - K$, in which M is easy for solves. Then we have

$$\begin{aligned} Ax &= b \\ (M-K)x &= b \\ Mx &= Kx + b \\ x &= M^{-1}(Kx+b) \end{aligned}$$

Then we have

- Truth: $Mx^* = Kx^* + b$
- Iteration: $Mx^{k+1} = Kx^k + b$

Taking differences, the error iteration is

$$\begin{aligned} Me^{k+1} &= Ke^k \\ e^{k+1} &= \underbrace{(M^{-1}K)}_{R} e^k \end{aligned}$$

Thus, $\rho(R) < 1$ is necessary and sufficient for convergence. Since $\rho(R)$ is bounded from above by $\|R\|$ for any operator norm, a sufficient condition is that $\|R\| < 1$. Suppose we do have a bound $\|R\| \leq \rho < 1$. Then we have

$$\begin{aligned} \left\|e^{k+1}\right\| &= \left\|Re^k\right\| \\ &\leq \rho \left\|e^k\right\| \\ &\leq \rho^k \left\|e^0\right\| \end{aligned}$$

Thus, to reduce the error by a constant factor $C < 1$,

$$\begin{aligned} \frac{\left\|e^k\right\|}{\left\|e^0\right\|} &\leq \rho^k < C \\ k \log \rho &\leq \log(C) \\ k &\geq \frac{\log(C)}{\log(\rho)} \end{aligned}$$

Now, we consider specific splittings. Consider a splitting $A = D - L - U$, where D is diagonal, $-U$ is upper trianglar, and $-L$ is lower triangular. Then we have three classic iterations

- Richardson: $M = \alpha I$
- Jacobi: $M = D$
- Gauss-Seidel: $M = D - L$

For the Richardson iteration, the fixed point equation is

$$\begin{aligned} \alpha I x &= (\alpha I - A)x + b \\ x &= \underbrace{(I - \omega A)}_{R} x + \omega b \qquad\qquad \omega = \frac{1}{\alpha} \end{aligned}$$

In the SPD case, say we have $0 < \lambda_1 \leq \ldots \leq \lambda_n$. For convergence, it is necessary that

$$\begin{aligned} 1-\omega\lambda_n(A) &> -1 \\ 2-\omega\lambda_n(A) &> 0 \\ \omega &< \frac{2}{\lambda_n(A)} \end{aligned}$$

The optimal ω has

$$\begin{aligned} 1-\eta &= 1-\omega\lambda_1(A) \\ -1+\eta &= 1-\omega\lambda_n(A) \\ \implies \omega^* &= \frac{2}{\lambda_1(A)+\lambda_n(A)} \end{aligned}$$

Because we have that

$$\begin{aligned} \rho(1-w^*A) &= 1-\omega^*\lambda_1 - |2-w^*\lambda_n| \\ 1-\frac{2\lambda_1}{\lambda_1+\lambda_n} &= \frac{\lambda_1+\lambda_n}{\lambda_1+\lambda_n} - \frac{2\lambda_1}{\lambda_1+\lambda_n} \\ &= \frac{\lambda_n-\lambda_1}{\lambda_n+\lambda_1} \\ &= \frac{\kappa(A)-1}{\kappa(A)+1} \end{aligned}$$

Recall that for our model problem, a row of our equation $T_{N\times N}u = h^2 f$ is

$$-u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} + 4u_{i,j} = h^2 f_{ij}$$

Note that the Jacobi and Richardson iterations are essentially the same, since the diagonal of $T_{N\times N}$ is the constant 4. The Jacobi iteration updates by assuming that the neighbors in the previous step are exactly correct

$$u_{ij}^{(k+1)} = \frac{1}{4}(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} + h^2 f_{ij})$$

Gauss-Seidel updates by assuming that the most recently updated neighbors are correct

$$u_{ij}^{(k+1)} = \frac{1}{4}(u_{i-1,j}^{(\text{most recent})} + \ldots + h^2 f_{ij})$$

We end with consideration of convergence rates. For Jacobi we have

$$\begin{aligned} x_{k+1} &= D^{-1}((L+U)x_k + b) \\ R &= D^{-1}(L+U) \end{aligned}$$

Note that $\|R\|_\infty < 1$ if and only if A is strictly diagonally dominant. This is because each row sum of R is the sum of the off diagonals of that row divided by the diagonal element. Thus, strict diagonal dominance of A is sufficient for convergence of Jacobi.

Note that our model problem $T_{N\times N}u = h^2 f$ is equivalent to

$$\min \underbrace{\frac{1}{2}u^T T_{N\times N}u - h^2 u^T f}_{\varphi}$$

This is because this optimization problem has a convex objective, so it achieves its minimum at the only stationary point, which occurs exactly when $T_{N\times N}u = h^2 f$ as seen by differentiating.

Thus, we can also consider solving the model problem with optimization strategies. For instance, we can consider coordinate descent on $\varphi : \mathbb{R}^m \to \mathbb{R}$. This takes the form of iterating

$$\hat{u}_j = \text{argmin}_z\, \varphi(u - e_j u_j + z e_j)$$

It can be shown that Gauss-Seidel has the exact same steps as this. This is enough to show that Gauss-Seidel does not blow up. Not too many more arguments give convergence as well.

Lecture 32: Approximation from subspaces (11/18)

(Missed this lecture)

In estimating a solution to the linear system $Ax = b$ by an approximate solution $\hat{x}$ in a subspace $\mathcal{V}$, there are several possible approaches:

- Least squares: Solve $\min_{\hat{x}\in\mathcal{V}} \|A\hat{x} - b\|_M^2$ for some M.
- Optimization: If A is spd, solve $\min_{\hat{x}\in\mathcal{V}} \phi(x) = \frac{1}{2}x^T Ax - x^T b$ over
- Galerkin: Choose $A\hat{x} - b \perp \mathcal{W}$ for a test space $\mathcal{W}$. For Bubnov-Galerkin methods, $\mathcal{W} = \mathcal{V}$. If the test space is different than the approximation space, then the method is a Petrov-Galerkin method.

Lecture 33: Krylov subspaces (11/20)

We look at our iterative methods

$$\begin{aligned} Mx_{k+1} &= Kx_k + b \\ Mx_k &= Kx_{k-1} + b \\ Md_{k+1} &= Kd_k, \qquad\qquad d_k = x_k - x_{k-1} \\ d_{k+1} &= \underbrace{M^{-1}K}_{R} d_k \end{aligned}$$

Note that we have

$$\begin{aligned} x_k &= x_0 + d_1 + \ldots + d_k \\ &= x_0 + d_1 + Rd_1 + \ldots + R^k d_1 \\ &= x_0 + \sum_{k=0}^{k} R^j d_1 \\ &= x_0 + (I - R^{j+1})(I - R)^{-1} d_1 \end{aligned}$$

Note that this is a power iteration. If $R = V\Lambda V^{-1}$, then

$$\begin{aligned} d_k &= R^k d_1 \\ \tilde{d}_k &= \Lambda^k \tilde{d}_1 \end{aligned}$$

If $A = V\Lambda V^{-1}$ is diagonalizable, then $A^{-1} = V\Lambda^{-1}V^{-1}$. If we find an interpolating polynomial $p(z)$ such that $p(\lambda_j) = \frac{1}{\lambda_j}$, then $p(A) = A^{-1}$. Thus, instead of using the basic polynomial $\sum_{j=0}^{k} R_j$ in our iteration, we can consider using different polynomials that may allow quicker convergence.

Consider the iteration $x^{k+1} = Rx^k + b$, where R is spd with $\rho(R) < 1$. Then we have

$$\begin{aligned} x^{(1)} &= Rx^0 + b \\ x^{(2)} &= R^2x^0 + Rb + b \\ x^{(3)} &= R^3x^0 + R^2b + Rb + b \end{aligned}$$

if we set $x^{(0)} = 0$, then

$$x^{(k)} = \sum_{j=0}^{k-1} R^j b \xrightarrow{k} (I - R)^{-1}b$$

Now, suppose we have more information, that the spectrum is contained in an interval $\sigma(R) \subseteq [\alpha, \beta]$, $0 < \alpha < \beta < 1$. We wish to find a polynomial $p(z)$ so that $p(A)b \approx A^{-1}b$, meaning

$$Ap(a)b - b = [Ap(A) - I]b \quad \text{is small}$$

We call $q(z) = 1 - zp(z)$, the negation of the polynomial that appears. Note $q(0) = 1$, and we want $q(\lambda_j)$ as small as possible. We can apply the equioscillation theorem, and will later discuss Chebyshev polynomial methods that work for us.

We define the kth Krylov subspace

$$\begin{aligned} \mathcal{K}_k(A, b) &= \text{span}(b, Ab, \ldots, A^{k-1}b) \\ &= \{p(A)b : p \in \mathbb{C}_{k-1}[x]\} \end{aligned}$$

The key ideas going forward are:

- Use Krylov subspaces as approximation spaces
- Use Galerkin/optimization to choose good solutions from the spaces

The best possible residual involves minimizing $q(z)$ on the spectrum, note that

$$q(z) = \frac{\det(zI - A)}{\det(-A)} \quad \text{gives 0 residual}$$

Now, we go back to the spd case, and ask what can we say if we only know the eigenvalues are contained in $[\alpha, \beta]$.

We will make use of the Chebyshev polynomials, defined as

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{k+1}(x) &= 2xT_k(x) - T_{k-1}(x) \end{aligned}$$

This is a constant coefficient recurrence, we have

$$T_k(x) = \alpha\xi_1^k + \beta\xi_2^k$$

where ξ_1, ξ_2 are the roots of $z^2 - 2xz + 1$. For x between $[-1,1]$, $T_k(x)$ oscillates and is given by $T_k(x) = \cos(k \arccos(x))$. Outside of $[-1,1]$, we have $T_k(x) = \cosh(k \cosh^{-1}(x))$, which blows up quickly. We have a bound

$$T_m(1+\epsilon) \geq \frac{1}{2}(1 + m\sqrt{2\epsilon})$$

This is connected to the condition number. On $[\alpha, \beta]$, we have

$$\begin{aligned} |q_m| &\leq \frac{2}{1 + m\sqrt{2\epsilon}} \\ \epsilon &= 2(\kappa(A) - 1)^{-1} \\ |q_m| &\leq 2\Big(1 - \frac{2m}{\sqrt{\kappa(A) - 1}}\Big) + O\Big(\frac{m^2}{\kappa(A) - 1}\Big) \end{aligned}$$

If the spectrum is not uniformly spread about $[\alpha, \beta]$, then we would want unequal oscillations about the interval.

Lecture 34: Krylov subspaces (11/22)

We consider the $\mathcal{K}_k(A, b) = \operatorname{span}(b, Ab, \ldots, A^{k-1}b) = \{p(A)b \mid p \in \mathbb{C}_{k-1}[x]\}$. Observe that

- If $p(z)$ is the minimal polynomial of A, $p(z) = \prod_j (z - \lambda_j)$, so $p(A) = 0$, and is of minimal degree, then $\mathcal{K}_{\deg p}(A, b)$ is the same as the Krylov subspace for any higher degree. In general, if q is a polynomial such that $q(A)b = 0$, then we can use Krylov subspaces of degree up to $\deg p$.

Thus, if we can find a polynomial such that $q(A)$ kills b, we can bound the largest Krylov subspace we need consider.

Today, we will consider orthonormal bases for Krylov subspaces and projections of A onto these bases.

Let us take a generic $A \in \mathbb{R}^{n \times n}$. We want an orthonormal basis with $\operatorname{span}\{q_0, \ldots, q_{k-1}\} = \mathcal{K}_k(A, b)$. Our process is

$$\begin{aligned} q_0 &= \frac{b}{\|b\|} \\ \tilde{q}_1 &= Aq_0 - q_0(q_0^T A q_0) \\ q_1 &= \frac{\tilde{q}_1}{\|\tilde{q}_1\|} \\ \vdots \quad & \quad \vdots \\ \tilde{q}_{j+1} &= Aq_j - \sum_{i=0}^{j} q_i(q_i^T A q_j) \\ q_{j+1} &= \frac{\tilde{q}_{j+1}}{\|\tilde{q}_{j+1}\|} \end{aligned}$$

We write this as

$$\tilde{q}_{j+1} = Aq_j - \sum_{i=0}^{j} h_{ij}$$
$$q_{j+1} = \frac{\tilde{q}_{j+1}}{h_{j+1,j}}$$

so we have

$$h_{j+1,j}q_{j+1} + \sum_{i=0}^{j} q_i h_{ij} = Aq_j$$
$$\sum_{i=0}^{j+1} q_i h_{ij} = Aq_j$$

Writing this in matrix form,

$$A\begin{bmatrix} q_0 & q_1 & \dots \end{bmatrix} = \begin{bmatrix} q_0 & q_1 & \dots \end{bmatrix} \begin{bmatrix} h_{00} & h_{01} & h_{02} & \dots \\ h_{10} & h_{11} & h_{12} & \dots \\ & h_{21} & h_{22} & \dots \\ & & \ddots & \end{bmatrix}$$

When we have computed the full Krylov subspace, then we have an incrementally computed Hessenberg decomposition $AQ = QH$. The iterates in between are the Arnoldi decompositions $AQ^{(k)} = Q^{(k+1)}\overline{H}^{(k)}$, where

$$Q^{(k)} = \begin{bmatrix} q_0 & \dots & q_k \end{bmatrix}$$

and $H \in \mathbb{R}^{(k+1)\times k}$ is upper Hessenberg with an extra row. This iteration is the Arnoldi algorithm.

The classical Gram-Schmidt orthogonalization step is

$$w = Aq_j$$
$$\tilde{q}_{j+1} = w - \sum_{i=0}^{j} q_i(q_i^T w)$$

on the other hand, modified Gram-Schmidt uses that $(I - \sum_{i=0}^{j} q_i q_i^T) = \prod_{i=0}^{j}(I - q_i q_i^T)$ due to orthogonality, so the iteration is

- $w \leftarrow Aq_j$
- For $i = 0, \dots, j$
  $w \geq w - q_i(q_i^T w)$

If A is symmetric, then the iteration computes

$$AQ^{(k)} = Q^{(k+1)}T = Q^{(k)}T^{(k)} + \beta_{k+1}q_{k+1}e_k^T$$

where $\beta_1, \dots, \beta_{n-1}$ are the subdiagonal entries of T. This is known as the Lanczos iteration .

Now we expand to try to make an iteration. Reading off one column of the equation is

$$\begin{aligned}(AQ)_k &= Aq_j \\ &= (QT)_j \\ &= \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \\ \beta_j q_{j+1} &= (A - \alpha_j I)q_j - \beta_{j-1}q_{j-1}\end{aligned}$$

Conjugate Gradients

Let A by symmetric. Consider the problem

$$\min \frac{1}{2}x^T Ax - x^T b$$

over a Krylov space.

$$\begin{aligned}AQ^{(k)} &= Q^{(k)}T^{(k)} + \beta_k q_{k+1}e_k^T \\ \hat{x} &= AQ^{(k)}y \\ Q^{(k)T}AQ^{(k)}y &= T^{(k)}y \\ \text{solve } T^{(k)}y &= Q^{(k)T}b \\ \hat{x} &= Q^{(k)}y\end{aligned}$$

If we have A generally nonsymmetric, and we want to solve

$$\min\|Ax - b\|$$

over $\hat{x} = Q^{(k)}y$, then

$$\min\left\|\overline{H}^{(k)}y - Q^{(k+1)}b\right\| = \min\left\|\overline{H}^{(k)}y - \|b\|\, e_1\right\|$$

gives the GMRES method.

Lecture 35: Conjugate Gradients (11/25)

Let A be spd. We will form a Krylov subspace $\mathcal{K}$ and minimize the objective

$$\min_{\mathcal{K}} \varphi(x) = \frac{1}{2}x^T Ax - x^T b$$

over the Krylov subspace. First we will look at some properties of this problem

$$\begin{aligned}\tilde{\varphi}(x) &:= \frac{1}{2}\|r\|_{A^{-1}}^2 \\ &= \frac{1}{2}\Big((Ax-b)^T A^{-1}(Ax-b)\Big) \\ &= \frac{1}{2}\Big(x^T Ax - 2x^T b + b^T A^{-1}bA^{-1}(Ax-b)\Big) \\ &= \phi(x) + \frac{1}{2}\|b\|_{A^{-1}}^2\end{aligned}$$

since the second summand does not depend on x, minimizing $\tilde{\varphi}$ is equivalent to minimizing φ.

There is another way to think about this

$$\begin{aligned} e(x) &= x - x^* \\ r(x) &= Ax - b \\ &= Ax - Ax^* \\ &= Ae \end{aligned}$$

Thus, at the solution, the error e is orthogonal to the residual r.

Now, consider

$$\begin{aligned} \hat{\varphi}(x) &= \frac{1}{2}\|e\|_A^2 \\ &= \frac{1}{2}\|x - x^*\|_A^2 \\ &= \frac{1}{2}(x - x^*)^T A(x - x^*) \\ &= \frac{1}{2}x^T Ax - x_*^T Ax + \frac{1}{2}x_*^T Ax_* \\ &= \frac{1}{2}x^T Ax - x^T b + \|x_*\|_A^2 \\ &= \tilde{\varphi}(x) \end{aligned}$$

So we have three perspectives for looking at the method of conjugate gradients. Now, recall the Lanczos iteration

$$Q^{(k)T} A Q^{(k)} = T^{(k)}$$

Consider Bubnov-Galerkin with $x^{(k)} = Q^{(k)} y^{(k)}$. The problem is of the form

$$\min_{y^{(k)}} \varphi(Q^{(k)} y^{(k)})$$

Note that

$$Q^{(k)} b = \|b\| e_1 = (\|b\|, 0, \ldots, 0)$$

Note the problems $T^{(k)} y^{(k)}$ are nested linear systems. This is because $T^{(k+1)} y^{(k+1}$ is of the form

$$\begin{bmatrix} T^{(k)} & & \\ & & \beta_k \\ & \beta_k & \alpha_{k+1} \end{bmatrix} \begin{bmatrix} y_1^{(k+1)} \\ y_2^{(k+1)} \end{bmatrix}$$

We use LU decompositions of the iterates $T^{(k+1)} = L^{(k+1)} U^{(k+1)}$.

$$\begin{bmatrix} L^{(k)} & & 0 \\ 0 & \tilde{l} & 1 \end{bmatrix} \begin{bmatrix} U^{(k+1)} & 0 \\ & \tilde{u} \\ 0 & S \end{bmatrix}$$

Then our intermediate iterates are

$$\begin{aligned} y^{(k)} &= Q^{(k)}[T^{(k)}]^{-1}\|b\|\,e)1 \\ &= \|b\|\left(\underbrace{Q^{(k)}[U^{(k)}]^{-1}}_{V^{(k)}}\underbrace{[L^{(k)}]^{-1}e_1}_{z^{(k)}}\right) \end{aligned}$$

$$L = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & l_2 & 1 & \\ & & \ddots & \ddots \end{bmatrix}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & & & \\ & u_{22} & u_{23} & & \\ & & u_{33} & u_{34} & \\ & & & \ddots & \ddots \end{bmatrix}$$

Where $z^{(k)}$ can be computed simply

$$\begin{aligned} Lz &= \|b\|\,e_1 \\ z_1 &= \|b\| \\ l_1 z_1 - z_2 = 0 &\implies z_2 = -l_1 z_1 \\ z_{k+1} &= -l_k z_k \end{aligned}$$

and $V^{(k)}$ can also be computed

$$\begin{aligned} v_1 u_{11} = q_1 &\implies v_1 = q_1/u_{11} \\ v_1 u_{12} + v_2 u_{22} = q_2 &\implies v_2 = (q_2 - v_1 u_{12})/u_{22} \end{aligned}$$

This means that the iterates can be simply updated

$$x^{(k+1)} = V^{(k+1)} z^{(k+1)} = x^{(k)} + v_{k+1} z_{k+1}$$

Lecture 36: CG and Beyond (12/4)

We continue with conjugate gradients. Consider $\varphi(x) = \frac{1}{2}x^T Ax - x^T b$. Also, we consider an optimization framework

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

where $p^{(k)}$ should be a descent direction, so $p^{(k)T}\nabla\varphi < 0$, but not too close to zero. Here, we have $\nabla\varphi = Ax - b$, which happens to be the residual. One way to choose a descent direction is by gradient descent, with $p^{(k)} = -\nabla\varphi$. This is of the form

$$x^{(k+1)} = x^{(k)} + \alpha_k(b - Ax^{(k)})$$

For $\alpha_k = \alpha$ fixed, this is the Richardson iteration. This does not work well if A is ill-conditioned. There are methods to choose α_k in a better way; we will consider changing p_k. There are two ways:

1. Scaling/ preconditioning $x^{(k+1)} = x^{(k)} + \alpha_k M^{-1}(b - Ax^{(k)})$

   There are several preconditioning methods derived from stationary methods.

2. Look at previous steps. This is what conjugate gradients does.

Consider the iteration

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

and choose $p_{k-1} \in \mathcal{K}_k(A, b)$ such that

$$\begin{aligned} & p_{k-1} \perp_A \mathcal{K}_{k-1}(A, b) \\ & \text{i.e.} \quad p_{k-1}^T A v = 0 \quad \forall v \in \mathcal{K}_{k-1}(A, b) \end{aligned}$$

Then compute an optimal step length α_k.

$$\begin{aligned} x^{(k)} &= x^{(k-1)} + \alpha_k p^{(k-1)} \\ r^{(k)} &= r^{(k-1)} - \alpha_k A p^{(k-1)} \end{aligned}$$

since $r^{(k)} \perp r^{(k-1)}$, we can then solve

$$\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$$

Beyond CG

Sadly, not all matrices are symmetric positive definite. For a general problem, we have idea 0:

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 = \frac{1}{2} x^T A^T A x - x^T A^T b + \|b\|^2$$

Thus, one possibility is CG on normal equations (CGNE). This is not used terribly often, since $A^T A$ can be quite ill-conditioned. A more commonly used method, which we call idea 1, is within MINRES (symmetric) and GMRES (nonsymmetric). GMRES starts from the form

$$A Q^{(k)} = Q^{(k+1)} \overline{H}^{(k)}$$

And solves the problem is

$$\begin{aligned} & \min_y \left\| A \underbrace{Q^{(k)} y}_{x^{(k)}} - b \right\| \\ & \equiv \min_y \left\| \overline{H}^{(k)} - \|b\| e_1 \right\|^2 \end{aligned}$$

We solve this and form the iterate $x^{(k)}$. The issue is that we have to save the Krylov basis. Thus, in practice, GMRES(k) is often used:

- While not converged

- $r = b - A\hat{x}$
- Approximate $A\Delta x = r$ with (up to) k steps of GMRES
- $\hat{x} \leftarrow \hat{x} + \Delta x$

Typically, k is chosen to be 10-20, but convergence can be weird. Indeed, it is hard to reason about this algorithm. Of course, it is often preconditioned, so that we solve $M^{-1}Ax = M^{-1}b$.

Lecture 37: Krylov methods for eigenproblems (12/6)

We will discuss solving eigenproblems with Krylov subspaces:

- Variational/ Galerkin approach to eigenproblems
- What does the Krylov subspace contain (in practice)?
- Spectral transformations and filtering

Today we will focus on symmetric eigenproblems, so $A = A^T$. Recall that for the Rayleigh quotient $\rho_A(v) = \frac{v^T A v}{v^T v}$, the nonzero stationary points of ρ_A are eigenvectors, meaning that $\delta\rho_A(v) = 0 \implies Av = \rho_A(v)v$.

The goal is to approximate eigenpairs from a subspace $\mathcal{V}$. We are looking for $v = Vy$ where V is a (orthonormal) basis for $\mathcal{V}$. Consider the Rayleigh quotient

$$\begin{aligned}\rho_A(Vy) &= \frac{y^T V^T A V y}{y^T V^T V y} \\ &= \frac{y^T V^T A V y}{y^T y} \\ &= \rho_{V^T A V}(y)\end{aligned}$$

We will find stationary points of the Rayleigh quotient over some subspace. This general method is called a **Rayleigh-Ritz procedure**. The approximations $(\theta, u) \approx (\lambda, x)$ are called **Ritz pairs**, where $V^T A V y = \theta y$, $u = Vy$.

Recall that if $u = v + e$, where v is an actual eigenvector, then $\theta = \lambda + O(\|e\|^2)$. This comes directly from the Rayleigh quotient.

A good choice of subspace to search over is a Krylov subspace $\mathcal{K}_k(A, b)$ for some random vector b. Note that $\mathcal{K}_k(A, b) = \text{span}(b, Ab, \ldots, A^{k-1}b)$. These are power iterates, so we expect good estimates for the eigenpair with maximal magnitude eigenvalue. Note that the Krylov subpaces are shift invariant, meaning that $\mathcal{K}_k(A, b) = \mathcal{K}_k(A - \sigma I, b)$ for $\sigma \in \mathbb{R}$. Thus, we also expect good estimates for the eigenpair with the minimal magnitude eigenvalue. For a polynomial $p \in \mathbb{R}_{k-1}[x]$,

$$\begin{aligned}\frac{b^T p(A)^T A p(A) b}{b^T p(A)^T p(A) b} &= \frac{b^T A p(A)^2 b}{b^T p(A)^2 p(A) b} && p(A) \text{ commutes with } A \\ &= \frac{\sum_{j=1}^n \tilde{b}_j^2 \lambda_j p(\lambda_j)^2}{\sum_{j=1}^n \tilde{b}_j p(\lambda_j)^2}\end{aligned}$$

Thus, the approximation theory here is linked to polynomials (as in Krylov methods for linear systems). This means that the spacing of the spectrum matters a lot in the approximation: clusters of eigenvalues are hard to deal with, and the extremal eigenvalues are easier to approximate. Now, let $Q^{(k)}$ be the orthonormal basis as in Lanczos,

$$Q^{(k)T} A Q^{(k)} = T^{(k)}$$

It can be seen that interior eigenvalues are expensive to compute. They need lots of steps and work from reothogonalization. One solution to this is implicit restarting, which essentially reduces the size of the Krylov subspace on some steps while maintaining eigenvectors that have already been converged to. This is the one used by `eigs` in ARPACK. Another approach is Krylov-Schur. Yet another solution is explicit filtering/ spectral transformations. There are preconditioners for eigenproblems, but they are complicated.

Spectral transformations

There are (simple) rational transformations and polynomial transformations. The simplest rational transformation is of the form $(A-\sigma I)^{-1}$. For these methods, we need only a way to multiply by this matrix, preferably with direct factorization. These converge first to the eigenvalues near σ. The Cayley transform is of the form $(A-\sigma I)^{-1}(A+\sigma I), \sigma \in \mathbb{R}^+$. There is built-in support for shift-invert and Cayley transforms in ARPACK. Polynomial methods have the advantage that they do not require linear system solves. Spectrum slicing groups eigenvalues into slices, and solves with polynomials that are focused on one group at a time.