Lec 1 - MAT 226A - Bob Guy - 9/27

- ☒ Dahlquist book has everything + depth

Numerical computations require that you rethink & reformulate problems.

Ex: Compute eigvals of a matrix $A \in \mathbb{C}^{n \times n}$

1. compute characteristic polynomial
   $p(\lambda) = \det(A - \lambda I)$
2. Find the roots of $p(\lambda)$ to get the eigenvalues.

Horrible algorithm to implement numerically!

1. Calc $\det(A)$ -> standard way is expansion by minors
   - Let C_n = cost for $n \times n$ matrix
     $C_n = n \cdot C_{n-1}$, $C_1 = 3$ flops => $C_n = 3 \cdot n!$ flops
     => $C_n = \mathcal{O}(n!)$

Reformulate: $A = LU$ where L, U are triangular
$\mathcal{O}(n^3)$ work to compute LU factorization
then $\det(A) = \det(L)\det(U)$ ($\mathcal{O}(n)$ work) => $\mathcal{O}(n^4)$ total work

2. Find roots of high-deg poly $p(\lambda)$ -> poorly conditioned (very sensitive to round-off errors)
   - L> use eigval solver

- Numerically generate $A = QTQ^*$ (Schur decomposition) where Q unitary, T triangular $\in$ eigvals are diag. elmts. Same eigvals as A.

Floating point #5

Supp. I can store 4 decimal digits, which ones to store?

Naive: Pick first four left of decimal point

Can store 0, smallest = 1, largest = 9999

Pick first four to the right: $0.xxxx$

from 0.0001 to 0.9999

To expand range, use scientific notation

$$x = r \cdot 10^n$$

r: 3 digits here

n: 1 digit here, can store #s on 10 orders of magnitude

$$\frac{1}{10} \le r < 1 \text{ or } r = 0$$

(Most) computers store numbers in binary:

$$x = \pm q \times 2^m$$

q: $0.\_\_\_\_\_$ w/ $q = 0$ or $\frac{1}{2} \le q < 1$

know leading bit is 1

Instead $x = \pm (1+q) \times 2^m$

$\pm$: 1 bit for sign

q: bits for mantissa

m: bits for exponent/characteristic

IEEE standard: 32-bit precision

1	8	23
sign	exponent	mantissa

64-bit double precision:

1	11	52
sign	exponent	mantissa

Characteristic double precision ranges from $c = 0$ to $c = \sum_{j=0}^{10} 2^j = 2^{11} - 1 = 2047$

Then $m = c - 1023 \Rightarrow$

$$x = \pm (1+q) \times 2^{c-1023}$$

Lec 2 - MAT226A - 9/29

Double precision: $x = \pm(1+q)\times 2^{c-1023}$

Don't actually use $m=1024$ or $m=1023 \to$ NaN, underflow flags!

Smallest positive # representable: $c=1,\ q=0$

$$\Rightarrow x = 2^{-1022} \approx 2\times 10^{-308}$$

largest positive # representable: $q = 1/2+1/4+1/8+\cdots \approx 1$ $\quad x = 2^{1024} \approx 2\times 10^{308}$

Same # of #'s btwn 2^{-1022} and 2^{-1021} as btwn 2^{1022} and 2^{1023}

but relative size of the gaps is the same.

Store 0 as $c=0,\ q=0$.

How much precision in decimals?

$\log_{10} 2^{52} \approx 15.7$, $\log_{10} 2^{53} \approx 15.95$

$\Rightarrow$ 15-16 decimal digits of accuracy.

What is the error of representing real #s as floating point #s?

$x \in \mathbb{R}$, $x^-, x^+ \in \mathbb{F}$ closest floats to x.

$fl(x)$ is the floating point representation of x

$$|fl(x) - x| \le |x^+ - x^-|$$ or if we're rounding, $|fl(x) - x| \le \frac{1}{2}|x^+ - x^-|$

Assuming same m, $|x^+ - x^-| = 0.0\ldots1_k \times 2^m = 2^{m-k}$ (k-bit mantissa)

Relative error: $\frac{|fl(x)-x|}{|x|} \le \frac{2^{m-k}}{|x|} = \frac{2^{m-k}}{1.b_1b_2\cdots b_k b_{k+1}\ldots \times 2^m} \le 2^{-k}$

(independent of size of x, only depends on # bits > 1)

★ Relative error in repng real # as floating point #
depends on # bits in the mantissa.

For double precision, $k=52 \Rightarrow 2^{-52} \approx 2.22\times10^{-16}$
call this → $\varepsilon_{machine}$ ← relative spacing btwn floating point #s

Experiment: $fl(1+10^{-16}) = 1$; if $\varepsilon_{machine} > 10^{-16}$

Thm: $\forall x \in \mathbb{R}$ in some range ($\pm2\times10^{-308}$ to $\pm2\times10^{308}$), $\exists x' \in \mathbb{F}$
s.t. $|x-x'| \le \varepsilon_{machine}|x|$

There exists an ε w/ $|\varepsilon| < \varepsilon_{machine}$
s.t. $fl(x) = x(1+\varepsilon)$

Thm: Let $\circledcirc$ represent $\oplus, \ominus, \otimes, \oslash$ fl pt. operation
$\forall x, y \in \mathbb{F}$, $\exists \varepsilon$ with $|\varepsilon| < \varepsilon_{machine}$ s.t.

$$x \circledcirc y = x \cdot y(1+\varepsilon).$$

(· : real operation)

Condition # Ex 3) Polynomial Eval.

$p(x) = \sum_{k=0}^{n} a_k x^k$ → How sensitive to coefficient perturbations?

$$\frac{\partial p}{\partial a_i} = x^i$$

Then $\kappa(a_i) = \left|\frac{\partial p}{\partial a_i}\right| \frac{|a_i|}{|p(x)|} = \frac{a_i x^i}{|p(x)|}$

one way to measure condition # wrt. all coeffs

$$\kappa = \frac{\sum_{k=0}^{n} |a_k x^k|}{\left|\sum_{k=0}^{n} a_k x^k\right|}$$ → poorly-conditioned near roots

↳ look at abs. cond. #!

$\hat{\Sigma} - \hat{\Sigma}\,|x^k|$

Lec 3 - MAT226A - 10/2

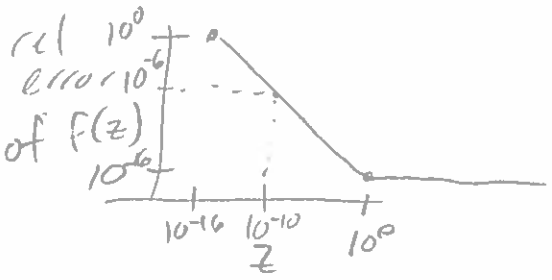
Need to be careful subtracting nearly equal numbers!

Ex: Evaluate the function $f(z) = e^z - 1$ for z small

$$f(z) = 1 + z + \frac{z^2}{2!} + \dots \qquad -1 = z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots$$

$\Rightarrow$ For z below ε_{mach}, $fl(e^z) = 1$

So Rel error $\approx \frac{|0 - z|}{|z|} = 1$



What to do? Reformulate the function

1) Use Taylor series to replace f $\rightarrow$ only good for small z.

2) Do some algebra: $f(z) = e^{z/2}(e^{z/2} - e^{-z/2})$

$= 2e^{z/2} \sinh(z/2)$

Conditioning & Stability

- Conditioning is related to how sensitive a problem is to perturbation.
- Stability is related to the error in an algorithm to solve the problem.

Condition # of a problem: Let $f: X \to Y$ where X, Y normed vector spaces. Want to know how changes in input affect changes in output.

Ex: Solve $A\underline{x} = \underline{b}$ where $A \in \mathbb{C}^{n \times n}$, $\underline{b} \in \mathbb{C}^n$. Input is $(A, \underline{b})$, output $\underline{x}$

So Given $x \in X$, let δx be a perturbation of x.

Change in output $\delta f = f(x + \delta x) - f(x)$

Given $x \in X$, δx a perturbation of x

change in output $\delta f = f(x+\delta x) - f(x)$

The condition #, $\kappa(x)$ (dep on x) satisfies

$$\frac{\|\delta f\|}{\|f\|} \leq \kappa(x) \frac{\|\delta x\|}{\|x\|}$$

($\|f\|$: norm of $f(x)$ in y.)

Def: The relative condition # is

$$\kappa(x) = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \left(\frac{\|\delta f\|}{\|f\|} \Big/ \frac{\|\delta x\|}{\|x\|} \right)$$

Def The absolute condition # is (useful when $\|f\| = 0$)

$$\hat{\kappa}(x) = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \frac{\|\delta f\|}{\|\delta x\|}$$

If f is differentiable, $\hat{\kappa}(x) = \|J\|$

where $J_{ij} = \frac{\partial f_i}{\partial x_j}$ and $\|\cdot\|$ is induced norm by norms on X and y.

Fact: If f is differentiable, relative condition # is

$$\kappa(x) = \|J\| / \left(\frac{\|f\|}{\|x\|} \right) = \|J\| \cdot \frac{\|x\|}{\|f\|}$$

($\|f\|$: norm of $f(x)$ in y)

Ex: 1) $f(x) = \sqrt{x} \Rightarrow f'(x) = \frac{1}{2} x^{-1/2} \Rightarrow \kappa(x) = \frac{1}{2\sqrt{x}} \cdot \frac{x}{\sqrt{x}} = \frac{1}{2}$ ← well-cond-ed

2) $f(x_1, x_2) = x_1 - x_2 \Rightarrow J = [1 \ -1]$ → use $\|\cdot\|_2$ as induced norm

$\Rightarrow \|J\|_2 = \sqrt{2}$, $\|x\|_2 = \sqrt{x_1^2 + x_2^2}$

$\Rightarrow \kappa(x) = \sqrt{2} \cdot \frac{\sqrt{x_1^2 + x_2^2}}{|x_1 - x_2|}$ ⇒ subtraction of nearly eq-al #s is poorly-conditioned!

Ex 3)

Lec 4 - MAT 226 A - 10/4

- Bob's OH: 4:15-5:30 Weds.

Condition # for $f: X \to Y$: $\frac{\|\delta f\|}{\|f\|} \leq \kappa(x) \cdot \frac{\|\delta x\|}{\|x\|}$

Ex: Polynomial evaluation

$p(x) = \sum_{k=0}^{n} a_k x^k$ ← look at condition # w.r.t. perturbations in the coefficients a_k

$\kappa(x) = \frac{\sum_{k=0}^{n} |a_k x^k|}{|\sum_{k=0}^{n} a_k x^k|}$ ← blows up near roots of $p(x)$

e.g. $p(x) = (x-2)^9 = \sum_{k=0}^{9} a_k x^k$

- At $x=2$, $\kappa = \infty$, but $\hat{\kappa} = \sum_{k=0}^{9} |a_k x^k|$

$\Rightarrow \hat{\kappa}(2) \approx 262,144 \approx 3 \times 10^5$

Notice that $\hat{\kappa}\,\varepsilon_{mach} \approx 10^{-11}$ $\Rightarrow$ same magnitude of "jiggles" in Bob's MATLAB solver example

Rule of thumb: rel. error $\leq$ (cond #)($\varepsilon_{machine}$) (best case)

$\Rightarrow$ Abs. cond. # is approx. # of digits we lose.

Stability & Accuracy: $f: X \to Y$ problem

Hope $\tilde{f}$ approx f! $\hat{f}: X \to Y$ algorithm

- Relative accuracy of algorithm: $\frac{\|f(x) - \hat{f}(x)\|}{\|f(x)\|} = E_{f,\hat{f}}$

Hope that $E_{f,\hat{f}} = O(\varepsilon_{machine})$

Recall: $\psi(t) = O(\phi(t))$ if $\exists c$ s.t. $|\psi(t)| \le c\,\phi(t)$; in some assumed limit, usually $t \to 0$ or $t \to \infty$

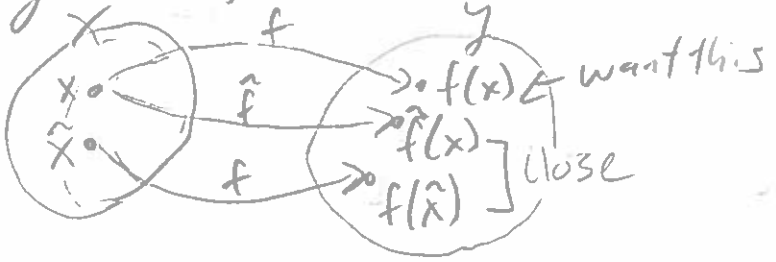
Unfortunately, $E_{f,\hat{f}} = O(\varepsilon_{machine})$ is too much to ask.

Would like a statement about bounding effects of round-off error still!

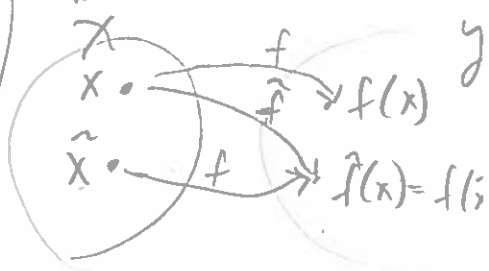
Stability: An algorithm $\hat{f}$ is stable if $\forall x \in X \;\exists \tilde{x} \in X$ s.t.

$$\frac{\|\hat{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = O(\varepsilon_{mach}) \;\&\; \frac{\|x - \tilde{x}\|}{\|x\|} = O(\varepsilon_{mach})$$

A stable algorithm gives nearly the right answer to nearly the right question.



backward stable



A stronger stability is **backward stability**: $\hat{f}$ is backward stable if

$\forall x \in X,\; \exists \tilde{x} \in X$ s.t. $\hat{f}(x) = f(\tilde{x})$ and $\frac{\|x - \tilde{x}\|}{\|x\|} = O(\varepsilon_{mach})$

↳ Backward stable alg gives the exact answer to nearly the right question.

(E.g. $\hat{f}(x) = fl(x)$ is backward stable since $fl(x) = \tilde{x}$ within ε_{mach}!)

Thm: For a backward-stable alg, $\frac{\|f(x) - \hat{f}(x)\|}{\|f(x)\|} = O(\kappa(x)\,\varepsilon_{mach})$

Pf: $\frac{\|f(x) - \hat{f}(x)\|}{\|f(x)\|} \overset{\text{backward stable}}{=} \frac{\|f(x) - f(\tilde{x})\|}{\|f(x)\|} \overset{\text{def of } \kappa}{\le} \kappa(x)\frac{\|x - \tilde{x}\|}{\|x\|} \overset{\text{backward stable}}{=} O(\kappa(x)\,\varepsilon_{mach})$ ☐

Fact: Floating point arithmetic is backward stable!

Ex: $f(x,y) = x + y$, $fl(x) = x(1+\varepsilon_1)$, $fl(y) = y(1+\varepsilon_2)$ ($\varepsilon_1, \varepsilon_2, \varepsilon_3 < \varepsilon_{mach}$)

$$\Rightarrow \hat{f}(x,y) = fl(x) \oplus fl(y) = \big(x(1+\varepsilon_1) + y(1+\varepsilon_2)\big)(1+\varepsilon_3)$$
$$= x(1+\varepsilon_1)(1+\varepsilon_3) + y(1+\varepsilon_2)(1+\varepsilon_3) = \hat{x} + \hat{y}$$

where $\hat{x} = x(1+\varepsilon_1)(1+\varepsilon_3)$, $\hat{y} = y(1+\varepsilon_2)(1+\varepsilon_3)$ ↗

Lec 5 - MAT226A - 10/6

- For a backward-stable algorithm $\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} \leq K(x)\,\varepsilon_{mach}$
  $\Rightarrow$ If $K = 10^r$, expect to lose r digits of acc.

- Last time, showed $\oplus$ is backward-stable: $\hat{f}(x,y) = fl(x) \oplus fl(y)$ $\quad < \varepsilon_{mach}$
  $= \underbrace{x(1+\varepsilon_1)(1+\varepsilon_3)}_{\tilde{x}} + \underbrace{y(1+\varepsilon_2)(1+\varepsilon_3)}_{\tilde{y}}$
  $\hookrightarrow \hat{f}(x,y) = f(\tilde{x}, \tilde{y})$.

- Given $f(x) = x+1$, $\hat{f}(x) = fl(x) \oplus 1$ is not backward-stable.
  $= x(1+\varepsilon_1) \oplus 1 = (x(1+\varepsilon_1) + 1)(1+\varepsilon_2)$
  $= \underbrace{x(1+\varepsilon_1)(1+\varepsilon_2) + \varepsilon_2}_{\tilde{x}} + 1$

  Then $\hat{f}(x) = f(\tilde{x})$ w/ $\tilde{x} = x(1+\varepsilon_1)(1+\varepsilon_2) + \varepsilon_2$
  but $\frac{|\tilde{x} - x|}{|x|} = \frac{|x + \varepsilon_1 x + \varepsilon_2 x + \varepsilon_1\varepsilon_2 x + \varepsilon_2 - x|}{|x|} = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2 + \frac{\varepsilon_2}{|x|} \neq \mathcal{O}(\varepsilon_{mach})$

- Now it's at least stable! $f(x) = fl(x) \oplus 1$
  $\Rightarrow$ Stable, not backward $= (\underbrace{x(1+\varepsilon_1)}_{\tilde{x}} + 1)(1+\varepsilon_2) = f(\tilde{x})(1+\varepsilon_2)$

An unstable algorithm: given square matrix, find λ's by finding roots of char poly.

Ex: $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow p(x) = x^2 - 2x + 1$ What if this is perturbed?
shift by ε:
$\Rightarrow$ if $\varepsilon = \varepsilon_{mach} = \mathcal{O}(10^{-16})$, roots shift by $\mathcal{O}(10^{-8})$.

Conditioning of a linear system $A\underline{x} = \underline{b}$ $\quad A \in \mathbb{C}^{n\times n}$

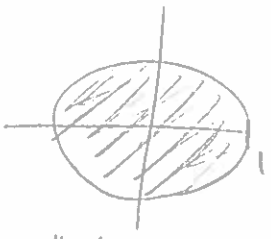
Vector norm: $\underline{x} \in \mathbb{C}^n$ $\quad \|\underline{x}\|_2 = \left(\sum_{j=1}^{n} |x_j|^2\right)^{1/2} = \sqrt{\underline{x}^*\underline{x}} = \sqrt{\langle \underline{x}, \underline{x}\rangle}$ $\quad \underline{x}, \underline{b} \in \mathbb{C}^n$
($\underline{x}$ col vector $n\times 1$, $\underline{x}^*$ is row vector $1\times n$ of complex conjugate transpose)

A norm, $\|\cdot\|$, is a fn. $\mathbb{C}^n \to \mathbb{R}$ s.t. 1) $\|\underline{x}\| \geq 0$ & $\|\underline{x}\| = 0$ iff $\underline{x} = \underline{0}$,
2) $\|\alpha\underline{x}\| = |\alpha|\|x\|$ $\forall \alpha \in \mathbb{C}$, 3) $\|\underline{x} + \underline{y}\| \leq \|x\| + \|y\|$ $\forall \underline{x}, \underline{y} \in \mathbb{C}^n$

Norms we'll use in this class: $\|\underline{x}\|_2 = \left(\sum_{j=1}^{n} |x_j|^2\right)^{1/2}$

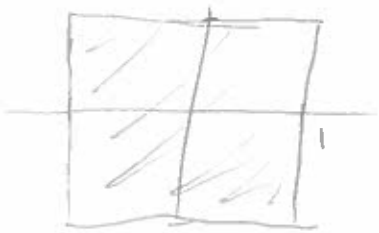
$\|\underline{x}\|_\infty = \max_{1 \le j \le n} |x_j|$, $\|\underline{x}\|_1 = \sum_{j=1}^{n} |x_j|$, $\|\underline{x}\|_p = \left(\sum_{j=1}^{n} |x_j|^p\right)^{1/p}$ $1 \le p < \infty$

Unit ball in ℓ_2:



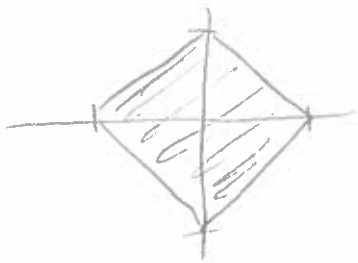
$\|x\|_2 \le 1$

in ℓ_∞:



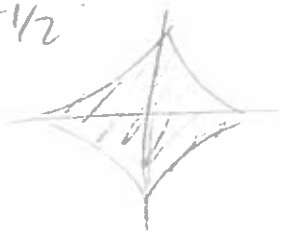
$\|x\|_\infty \le 1$

in ℓ_1



$\|x\|_1 \le 1$

in $\ell_{1/2}$:



in ℓ_0:

Lec 6 - MAT 226A - 10/9

Matrix norms: $A \in \mathbb{C}^{m \times n}$

- Could use a norm on $\mathbb{C}^{mn}$: Frobenius norm: $\|A\|_F = \left(\sum_{i,j} |a_{ij}|^2\right)^{1/2}$
  - just a measure of the 'stuff' in matrix, no structure!
  $= \mathrm{Tr}(A^*A)^{1/2}$

- An $m \times n$ matrix defines a linear function $\mathbb{C}^n \to \mathbb{C}^m$ by
$$f(\underline{x}) = A\underline{x}$$

Induced matrix norms reflect the size of the function instead of size of entries

Let's measure the size of the "stretch" → look at max stretch

Def: Let $A \in \mathbb{C}^{m \times n}$, $\|\cdot\|_{(n)}$ a norm on $\mathbb{C}^n$, $\|\cdot\|_{(m)}$ norm on $\mathbb{C}^m$, then

$$\|A\|_{(m,n)} = \sup_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \frac{\|A\underline{x}\|_{(m)}}{\|\underline{x}\|_{(n)}} = \max \|A\underline{x}\|_{(m)}$$

$\uparrow \|\underline{x}\|_{(n)} = 1$

unit ball is compact, so sup = max

Generally, pick the same norms on $\mathbb{C}^n$ and $\mathbb{C}^m$, & notate w/ that.

e.g. $\|A\|_1 = \max_{1 \le j \le n} \|\underline{a}_j\|_1$, $\|A\|_\infty = \max_{1 \le i \le m} \|\underline{a}_i^*\|_1$

$\underline{a}_j$: j^{th} col of A → max col. sum

$\underline{a}_i^*$: i^{th} row of A → max row sum

e.g. $\|A\|_2 = \sqrt{\text{largest mag. eigenvalue of } A^*A} = \sqrt{\rho(A^*A)} = \max \text{ sing. value } \sigma_1$

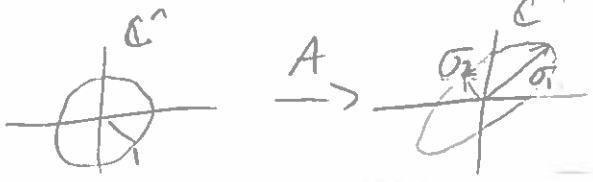


image of unit ℓ_2-ball is ellipse under A, radii of the semi-axes are singular values

Pf of $\|A\|_1 = \max_{1 \le j \le n} \|\underline{a}_j\|_1$: $\|A\underline{x}\|_1 = \left\|\sum_{j=1}^n \underline{a}_j x_j\right\|_1 \le \sum_{j=1}^n \|\underline{a}_j\|_1 |x_j|$

Choose j s.t. $\|\underline{a}_j\|_1 = \max_{1 \le j \le n} \|\underline{a}_j\|_1$. $\le \max_{1 \le j \le n} \|\underline{a}_j\|_1 \cdot \sum_{j=1}^n |x_j|$

Then choose $\underline{x} = \underline{e}_j$, so $\|A\underline{x}\|_1 = \left\|\sum_{j=1}^n \underline{a}_j x_j\right\|_1$ $\le \max_{1 \le j \le n} \|\underline{a}_j\|_1$

$= \|\underline{a}_j\|_1 = \max \|\underline{a}_j\|_1$. ∎

2-norm

Recall: a matrix Q is called unitary (orthog. if real) if

$$Q^*Q = I, \text{ i.e., } Q^* = Q^{-1}$$

each i,j of Q^*Q is an inner product $\langle q_i, q_j \rangle$

$Q^*Q = I \Rightarrow q_i^* q_j = \delta_{ij}$ $\Rightarrow$ cols. are orthonormal

$\Rightarrow \|Qx\|_2 = \|x\|_2 \Rightarrow \|Q\|_2 = 1.$

(since $\|Qx\|_2^2 = x^*Q^*Qx = x^*x = \|x\|_2^2$)

Recall: If $A^* = A$, then A has real eigvals & orthog. eigvectors
($\uparrow$ hermitian (symmetric if real))

& we can write $A = Q^* \Lambda Q$

Let Λ be diagonal, $\|\Lambda x\|_2 = \left(\sum_{j=1}^{n} |\lambda_{jj} x_j|^2 \right)^{1/2} \le \max_{1 \le j \le n} |\lambda_{jj}| \cdot \|x\|_2$

& get equality using $x = e_j$ w/ j corresp. to max $\nearrow$.

$$\Rightarrow \|\Lambda\|_2 = \max_j |\lambda_{jj}|.$$

Claim: $\|A\|_2 = (\rho(A^*A))^{1/2}$ for $A \in \mathbb{C}^{m \times n}$

$$\|Ax\|_2 = \left(x^* \underbrace{A^*A}_{\text{Hermitian}} x \right)^{1/2} = \left(x^* \underbrace{Q^* \Lambda Q}_{= A^*A} x \right)^{1/2} = \left(y^* \Lambda y \right)^{1/2}$$

where $y = Qx$

note $\|y\|_2 = \|x\|_2$

$$= \|\Lambda y\|_2 \le \|\Lambda\|_2 \|y\|_2$$

OH: W 4:30-5:30
R: 3-4

Lec 7 - MAT 226A - 10/11

Claim: $\|A\|_2 = \sqrt{\rho(A^*A)} = \sigma_1$

Pf: $\|Ax\|_2 = (x^*A^*Ax)^{1/2} = (y^*\Lambda y)^{1/2}$

where $y = Qx$, $A^*A = Q^*\Lambda Q$

$\Rightarrow \|y\|_2 = \|x\|_2$ since Q unitary

$\Rightarrow \|Ax\|_2 = \|\Lambda^{1/2} y\|_2 \le \|\Lambda^{1/2}\|_2 \cdot \|y\|_2 = \|x\|_2 \cdot \max_i \lambda_i^{1/2}$

$= \|x\|_2 \cdot \sqrt{\rho(A^*A)}$

$\Rightarrow \|Ax\|_2 \le \sigma_1 \cdot \|x\|_2$ $= \|x\|_2 \cdot \sigma_1$

Choose x to be the first singular vector / eigenvector of A^*A corr. to $\max_i \lambda_i$

then $\|Ax\|_2 = \sigma_1 \|x\|_2$.

Hence $\|A\|_2 = \sigma_1 = \sqrt{\rho(A^*A)}$. □

From the def. of induced norm, $\|Ax\| \le \|A\| \|x\|$,

we can show that $\|AB\| \le \|A\| \|B\|$

Pf: $\|ABx\| \le \|A\| \cdot \|Bx\| \le \|A\| \|B\| \cdot \|x\|$

$\Rightarrow \dfrac{\|ABx\|}{\|x\|} \le \|A\| \cdot \|B\|$ ← sup over x & done. □

Condition # of a matrix:

Fix $A \in \mathbb{C}^{m \times n}$, condition # of Ax from perturbations to x.

$$\kappa(x) = \lim_{\delta \to 0} \sup_{\|\delta x\| \le \delta} \frac{\|A(x+\delta x) - Ax\|}{\|Ax\|} \cdot \frac{\|x\|}{\|\delta x\|}$$

$$= \sup \frac{\|A\delta x\|}{\|\delta x\|} \cdot \frac{\|x\|}{\|Ax\|} = \|A\| \cdot \frac{\|x\|}{\|Ax\|}$$

Note: Solving $Ax = b$ s.t. perturbations in b
$\Rightarrow x = A^{-1}b \Rightarrow \chi(b) \leq \|A^{-1}\| \cdot \|A\|$
same thing!

Special case 1: Consider A square & invertible

$$\frac{\|x\|}{\|Ax\|} = \frac{\|A^{-1}Ax\|}{\|Ax\|} \leq \frac{\|A^{-1}\| \cdot \|Ax\|}{\|Ax\|} = \|A^{-1}\|$$

$$\Rightarrow \chi(x) \leq \underbrace{\|A\| \cdot \|A^{-1}\|}$$

Def: condition # of the matrix: $\chi(A) = \|A\| \cdot \|A^{-1}\|$

Now consider the condition # of solving $Ax = b$ s.t. perturbations in A.

$$(A + \delta A)(x + \delta x) = b$$
$$Ax + (\delta A)x + A(\delta x) + \text{h.o.t.} = b$$
$$\delta x = -A^{-1}(\delta A)x$$
$$\Rightarrow \|\delta x\| = \|A^{-1}(\delta A)x\| \leq \|A^{-1}\| \|\delta A\| \|x\| \Rightarrow \frac{\frac{\|\delta x\|}{\|x\|}}{\frac{\|\delta A\|}{\|A\|}} \leq \|A^{-1}\| \cdot \|A\|$$

take limsup

$$\Rightarrow \chi(A) \leq \|A^{-1}\| \cdot \|A\|$$

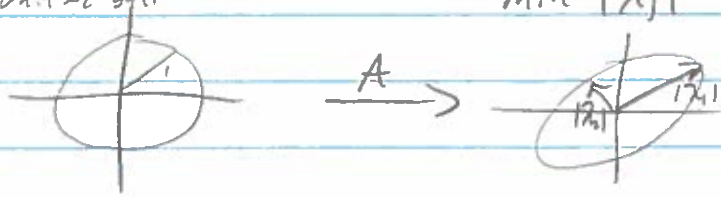
↑ equality is attained $\Rightarrow$ $\chi(A) = \|A\| \|A^{-1}\|$ ← cond. # of matrix A

condition # of solving $Ax = b$ s.t. perturb in A

Special case 2: Supp. A Hermitian, i.e. $A^* = A$.

$$\|A\|_2 = \rho(A), \quad \|A^{-1}\| = \rho(A^{-1}) = 1/\min_j |\lambda_j|$$ ← eigs of A.

$$\Rightarrow \chi(A) = \frac{\max |\lambda_j|}{\min |\lambda_j|}$$ ← can be ∞ if some $\lambda_j = 0$ (i.e. A not-invertible)



← $\chi(A) = \frac{\text{longest axis}}{\text{shortest axis}} = \text{aspect ratio}$

Then eigendecomp. of x is sum of projection onto axes

Poor conditioning $\Rightarrow$ axes on diff. scales $\Rightarrow$ addition of diff. scales $\Rightarrow$ loss of precision

Lec 7 cont'd 10/11 MATH226A

For non-square matrix A, in 2-norm

$$\gamma_1(A) = \frac{\sigma_1}{\sigma_m} \leftarrow \text{largest singular value}$$

smallest singular value

Same idea: x is sum of projection onto right-singular vectors, so if largest σ & smallest σ on diff scales, we lose precision in our reconstruction of x from singular modes!

Consider solving $Ax = b$, $A \in \mathbb{C}^{n \times n}$, A invertible, with no other information on special structure, use Gaussian elimination (Gaussian elimination = row-reduction)

$$A = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} \xrightarrow{\text{multiply } L_1} \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix} \xrightarrow{L_2} \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{pmatrix} = U$$

$$\Rightarrow \underbrace{L_2 L_1}_{\text{both are lower-triangular}} A = U \quad \text{(upper triangular)} \rightarrow \text{generating triangular factorization}$$

$$A = (L_2 L_1)^{-1} U = \underbrace{L_2^{-1} L_1^{-1}}_{\text{still lower triangular}} U = LU$$

(L: lower triangular, U: upper triangular)

MAT226A-Lec 8 - 10/13

- Gaussian elimination is effectively triangular factorization

$$A = LU = \text{lower}\Delta \cdot \text{upper}\Delta$$

Ex:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{pmatrix}$$

L_1 $\qquad$ A $\qquad$ $L_1 A$

To eliminate a_{21}, second row of L_1: $-2R_1 + R_2$

To eliminate a_{31}, third row of L_1: $-4R_1 + R_3$

To elim. a_{41}, fourth row of L_1: $-3R_1 + R_4$

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

L_2 $\qquad$ $L_1 A$ $\qquad$ $L_2(L_1 A)$

to eliminate $(L_1 A)_{32}$, third row of L_2: $-3R_2 + R_3$

to eliminate $(L_1 A)_{42}$, fourth row of L_2: $-4R_2 + R_4$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 4 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

L_3 $\qquad$ $L_2 L_1 A$ $\qquad$ U

$$\Rightarrow L_3 L_2 L_1 A = U \Rightarrow L = (L_3 L_2 L_1)^{-1} = L_1^{-1} L_2^{-1} L_3^{-1}$$

- computation is trivial

Compute inverses on the fly!

$$L_1^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{pmatrix}$$ superdrivial

---

Product also trivial

$$L_1^{-1} L_2^{-1} L_3^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \end{pmatrix}$$ & just throwing in entries into L as we go.

In general, $L_k = \begin{pmatrix} 1 & & & \\ & 1 & & 0 \\ & & \ddots & \\ & & -\ell_{k+1,k} & \\ & 0 & \vdots & 1 \\ & & -\ell_{n,k} & \end{pmatrix}$ w/ $\ell_{jk} = \frac{x_{jk}}{x_{kk}}$ ← entry in matrix at k^{th} step of elim

$k < j \leq n$

$$\underline{\ell}_k = \begin{pmatrix} 0 \\ \vdots \\ \ell_{k+1,k} \\ \\ \ell_{n,k} \end{pmatrix} \Rightarrow L_k = I - \underline{\ell}_k e_k^T$$

Show inverse in general: $(I - \underline{\ell}_k e_k^T)(I + \underline{\ell}_k e_k^T)$

$= I - \underline{\ell}_k e_k^T + \underline{\ell}_k e_k^T - \underline{\ell}_k (e_k^T \underline{\ell}_k) e_k^T$

↓ k^{th} elem of $\ell_k = 0$.

$= I$.

Hence $L_k^{-1} = (I + \underline{\ell}_k e_k^T)$ //

General product: $L_k^{-1} L_{k+1}^{-1} = (I + \underline{\ell}_k e_k^T)(I + \underline{\ell}_{k+1} e_{k+1}^T)$

$= I + \underline{\ell}_k e_k^T + \underline{\ell}_{k+1} e_{k+1}^T + \underline{\ell}_k (e_k^T \underline{\ell}_{k+1}) e_{k+1}^T$

↑ k^{th} elem of ℓ_{k+1} is also 0

$= I + \underline{\ell}_k e_k^T + \underline{\ell}_{k+1} e_{k+1}^T$

Lec 6 cont'd

Simple LU factorization algorithm (to be modified by Monday)

Given $A \in \mathbb{C}^{n\times n}$, compute $L+U$ for $A=LU$.

Init: $U=A$, $L=I$

Iterate: for $k=1$ to $n-1$ (loop over columns)

  for $j=k+1$ to n

    $l_{jk} = u_{jk}/u_{kk}$ ← note issue in trouble here if $u_{k,k}=0$

    $u_{j,k:n} = u_{j,k:n} - l_{j,k}\cdot u_{k,k:n}$ ← new j^{th} row is lin. comb. of k^{th} & j^{th} row

  end

end

Operation Count - count floating-point operations - flops

To compute l_j's in col $l_{:k}$, do $n-k$ divisions

Then changing rows need $2(n-k)^2$ flops (mult & addition)

$$\text{total flops} = \sum_{k=1}^{n-1} 2(n-k)^2 + (n-k) \approx \frac{2}{3}n^3 + \text{l.o.t.}$$

Recall: $\sum_{k=1}^{n-1} 2k^2 + k \Rightarrow \sum_{k=1}^{n} k^2 = \frac{1}{3}n^3 + \text{l.o.t.}$

$\Rightarrow$ total flops $= \mathcal{O}(n^3)$. (where $A \in \mathbb{C}^{n\times n}$)

Lec 9 - MAT226A - 10/16

The algorithm last lecture is unstable.

Ex: $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ can't be LU factorized w/ GE algorithm

but reordering rows yields a LU factorization

$$\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}_{P}\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{L}\underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{U}$$

Perturb this example: $A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix}$ ← GE will work now

$L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}$ $U = \begin{pmatrix} 10^{-20} & 1 \\ 0 & 1-10^{20} \end{pmatrix}$ but actually we'd get $\hat{U} = \begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix}$

$$\Rightarrow L\hat{U} = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}\begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix} = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 0 \end{pmatrix} = \hat{A}$$

↑ massive error here!

Supp. $\underline{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, solve $A\underline{x} = \underline{b}$

actual soln $\underline{x} \approx \begin{pmatrix} -1 \\ 1 \end{pmatrix}$, but numerical soln $\hat{x} \approx \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ ← 100% error in max-norm

why?
↳ $K(A) \approx 2.6$, but $K(L) \approx K(U) \approx 10^{40}$ awful!

Try w/ permuting rows!

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}}_{K(L)\approx 1}\underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1-10^{20} \end{pmatrix}}_{K(U)\approx 2.6}$$

No explosive growth in this case!

Lec 9 cont'd — 10/16

- This idea of swapping rows to keep condition #s low is called **Pivoting**.

Gaussian elimination w/ Pivoting:

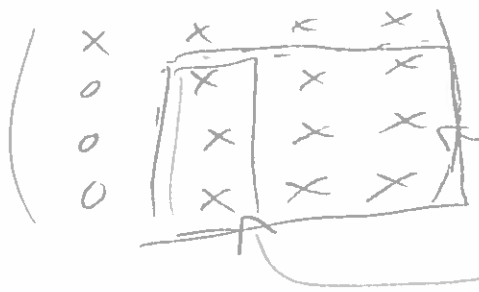
$$\begin{pmatrix} x & x & x & x \\ 0 & x_{kk} & x & x \\ 0 & x_{ik} & x & x \\ 0 & x & x & x \end{pmatrix} \xrightarrow{\text{usual}} \begin{pmatrix} x & x & x & x \\ 0 & x_{kk} & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix}$$

↓ how about

$$\begin{pmatrix} x & x & x & x \\ 0 & 0 & x & x \\ 0 & x_{ik} & x & x \\ 0 & 0 & x & x \end{pmatrix}$$

☆ don't have to go down the line, can look at submatrix & decide which rows/cols to eliminate

- Want to avoid dividing by tiny numbers



☆ Full-pivoting: pick the largest magnitude entry in submatrix for the "pivot"

☆ Partial-pivoting: pick biggest entry in column for the "pivot"

Note: Full-pivoting is more stable, but much more expensive!

To find pivot, Compares n^2 entries n times $\Rightarrow \mathcal{O}(n^3)$ work

Partial pivoting compares n entries n times to find pivots $\Rightarrow \mathcal{O}(n^2)$ work.

Algorithm: · Pick the largest entry in current column (diagonal & below)
· Swap rows to move the pivot to the diagonal.

$$\begin{pmatrix} x & x & x & x \\ 0 & x_{kk} & x_{k\cdot} & x_{k\cdot} \\ 0 & x & x & x \\ 0 & x_{ik} & x_{i\cdot} & x_{i\cdot} \end{pmatrix} \xrightarrow{\text{row swap}} \begin{pmatrix} x & x & x & x \\ 0 & x_{ik} & x_{i\cdot} & x_{i\cdot} \\ 0 & x & x & x \\ 0 & x_{kk} & x_{k\cdot} & x_{k\cdot} \end{pmatrix}$$

∈ row swap represented by mult. by a permutation matrix.
↳ Identity matrix except w/ permutation we want

Permutation matrix ex: Swap 2nd & 4th rows:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$ ← note P is symmetric & unitary

$P = P^T = P^{-1}$. ↳ only true for single-swap

Is this still LU factorization? Looks like we've permuted ↳ so much we lost its struct.

$$L_n P_n \cdots L_1 P_1 A = U$$

can show $PA = LU$ where $P = P_n \cdots P_1$

↖ what is this then?

How do we commute these P's & L's?

eg. $L_3 \underbrace{P_3 L_2 (P_3^{-1} P_3)}_{L_2'} P_2 L_1 P_1 A$

$$= L_3 L_2' \underbrace{P_3 P_2 L_1 (P_2^{-1} P_3^{-1} P_3 P_2)}_{L_1'} P_1 A$$

$$= \underbrace{L_3 L_2' L_1'}_{L^{-1}} \underbrace{(P_3 P_2 P_1)}_{P} A = L^{-1} P A$$

P_3 permutes l_2 from 3rd row & below → shifts entries, same structure

☆ L_2' has same structure as L_2

$$P_3 L_2 P_3^{-1} = P_3 (I - l_2 e_2^T) P_3^T = I - P_3 l_2 e_2^T P_3^T = I - \overbrace{P_3 l_2}\, e_2^T$$

P_3 permutes e_2 from 3rd row & below → all zero!

Partial-Pivot GE algorithm Init $U=A$, $L=I$, $P=I$

for $k = 1$ to $n-1$ (over cols)
  select $\max_{i \geq k} |u_{ik}|$ & swap i & k rows in U, L, P
  for $j = k+1$ to n (over rows)
    $l_{jk} = u_{jk} / u_{kk}$
    $u_{j,k:n} = u_{j,k:n} - l_{jk} \cdot u_{k,k:n}$
  end

stable in practice

→ produces $PA = LU$.

Lec 10 - 10/18 - MAT226A

GE w/ pivoting - small correction

- Step w/ picking pivot → swap row in i & k in U, L, P
  - ↳ cannot do this in L → don't want to mess w/ 1's on diag.

k^{th} step: L:

instead, just swap i^{th} & k^{th} rows in L for cols 1 : k-1

Stability of GE w/ pivoting

- Recall w/o pivoting, the entries of L and U can get arb. large
  - ↳ unstable, solns can grow
- W/ pivoting, we certainly avoid large entries in L, so what about U?

Thm: Let A non-singular and $A = LU$ be computed w/o pivoting. If the algorithm completes (w/ no pivots), then the computed $\hat{L}, \hat{U}$ satisfy

$$\hat{L}\hat{U} = A + \delta A, \text{ where } \frac{\|\delta A\|}{\|L\|\,\|U\|} \leq \Theta(\varepsilon_{machine})$$

☆ Looks like def of backwards stability, but don't have $\|A\|$ in denom!

We know $\|A\| \leq \|L\|\|U\|$ but this direction doesn't help...

With pivoting, L has 1's on diag. and $|\#'s| \leq 1$ below diag.

- so $\|L\|_\infty \leq n \Rightarrow \|L\| = \Theta(1)$

☆ norm of L is nice w/ pivoting (the entire point of pivoting was to keep the norm bounded)

=> mult. by $\|L\|$ will be absorbed into $\Theta(\varepsilon_{mach})$

What about $\|U\|$? Define $\rho = \frac{\max_{i,j} |u_{ij}|}{\max_{i,j} |a_{ij}|}$ as growth factor

Then $\|U\| \leq \rho \|A\|$, so $\|U\| = \mathcal{O}(\rho \|A\|)$, hence

$$\hat{L}\hat{U} = \hat{\rho} A + \delta A$$

$$\Rightarrow \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\rho \, \varepsilon_{mach})$$

In particular, if $\rho = \mathcal{O}(1)$, then GE w/ partial pivoting is backward stable.

Ex (worst case scenario of ρ):

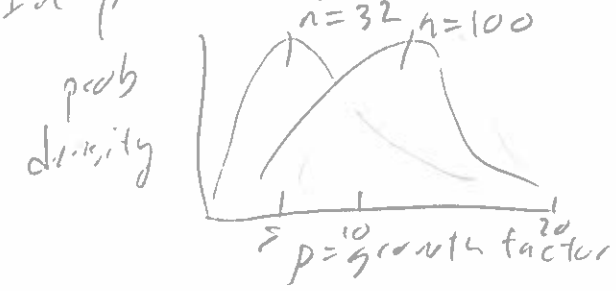
$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & -1 & 1 & 2 \\ 0 & -1 & -1 & 2 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & -1 & 4 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 8 \end{pmatrix}$$

$\Rightarrow$ for a matrix of this structure of size $n \times n$, $\rho \leq 2^{n-1}$

This still gives backwards stability! $\frac{\|\delta A\|}{\|A\|} = \mathcal{O}(2^{n-1} \cdot \varepsilon_{mach})$

Ex: $n = 100$, 2^{99} is theoretical bound $\approx 6 \times 10^{29}$ $\Rightarrow$ gigantic & useless!

In practice, growth factor is much smaller



prob. of getting huge ρ is small, pretty much zero.

See HW 2 #3c

Lec 10 cont'd 10/18

- Consider special case examples w/ more info on A.

Case: Tridiagonal matrix A → only nonzero on diag, sub, & superdiag.

- Assume no pivoting
- When we do GE on tridiag, don't introduce any nonzero where there were zeros before
- structure becomes:

- can represent as just 2 vectors ⇒ $O(n)$ work.

Case: (Hermitian/Symm) Positive Definite matrices

$A^* = A$, $x^*Ax > 0$ $\forall x \neq 0$ ⇒ all eigvals of A are real & positive

Cholesky factorization $A = R^*R$.

Ex: $A = \begin{pmatrix} 1 & w^* \\ w & K \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ w & I \end{pmatrix}\begin{pmatrix} 1 & w^* \\ 0 & K-ww^* \end{pmatrix}$

$= \begin{pmatrix} 1 & 0 \\ w & I \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & K-ww^* \end{pmatrix}\begin{pmatrix} 1 & w^* \\ 0 & I \end{pmatrix}$

R_1^* ↑ R_1 if this submatrix is symm posdef, then we can just repeat this

to generate series $R_1, R_2, R_3, \ldots$ & get factorization $A = R^*R$

Lec 11 - 10/20 - MAT226A

Continue example, what if $a_{11} \neq 1$?

$$A = \begin{pmatrix} a_{11} & w^* \\ w & K \end{pmatrix}$$

Let $\alpha = \sqrt{a_{11}}$ (possible since $e_1^* A e_1 = a_{11} > 0$)

Then $A = \begin{pmatrix} a_{11} & w^* \\ w & K \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha & w^* \\ \frac{w}{\alpha} & K \end{pmatrix}}_{R_1^*} \begin{pmatrix} 1 & 0 \\ 0 & K - \frac{ww^*}{a_{11}} \end{pmatrix} R_1$

Repeat on matrix $K - \frac{ww^*}{a_{11}}$, get $A = R_1^* R_2^* \cdots R_n^* R_n R_{n-1} \cdots R_1$

easily show R_i's combine naturally $\Rightarrow A = R^* R$.

This is the Cholesky factorization.

Exploits symmetry, so work $\sim \frac{1}{3} n^3$ (2x as fast as LU fact.)

☆ Cholesky doesn't need pivoting!

Last time, thm: $\hat{L}\hat{U} = A + \delta A$, $\frac{\|\delta A\|}{\|L\| \|U\|} = O(\varepsilon_{mach})$

For chol, $\hat{R}^* \hat{R} = A + \delta A$, $\frac{\|\delta A\|}{\|R\| \|R^*\|} = O(\varepsilon_{mach})$

$\|R^*\|_2 = \|R\|_2 = \sqrt{\rho(R^* R)} = \sqrt{\rho(A)} = \|A\|_2^{1/2}$ since A is symmetric.

$\Rightarrow \frac{\|\delta A\|}{\|A\|_2} = O(\varepsilon_{mach})$ $\Rightarrow$ Cholesky factorization is backwards stable. ☆

We talked about direct methods, but not iterative methods!

226B

Lec 11 cont'd - MAT226A - 10/20

Nonlinear Equations

$f: D \to \mathbb{R}^n$ solve $\underline{f}(\underline{x}) = 0$

$\therefore D \subset \mathbb{R}^n$

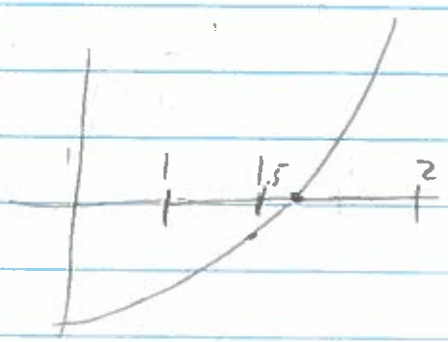
equiv. to
$$f_1(x_1, \ldots, x_n) = 0$$
$$\vdots$$
$$f_n(x_1, \ldots, x_n) = 0$$

} solve n nonlinear equations for n unknowns

- For nonlinear eqns, cannot find a general algorithm to get the solution in a finite # of steps.
  - ↳ Know that there's no formula for the roots of polynomials of deg ≥ 5.
- Use algorithms which give an approximate solution to a desired accuracy.

Ex. Want $\sqrt{3} = 1.?$ Could guess & check $(1.a)^2 > 3$?
If so, decrease a, else increase a

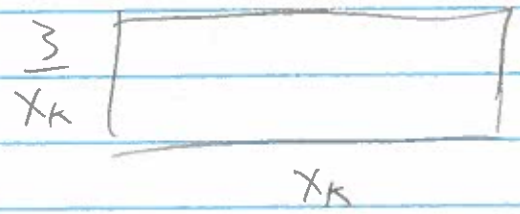
More systematically:
$f(x) = x^2 - 3$
have soln in $(1,2)$
try $x = 1.5$, $f(1.5) < 0$
so soln in $(1.5, 2)$



↳ Bisection Method
- ↳ great for scalar problems, guaranteed convergence (to desired accuracy)
- ↳ each step cuts error in half
- ↳ not generalizable to higher dims.

How fast does bisection method converge? Each step halves error, so 3-4 steps per digit of accuracy. → linear convergence

Geometric soln to finding $\sqrt{3}$: Babylonian algorithm



Check if $\frac{3}{x_k}$ & x_k equal, if not, one is too high, the other low

so next guess $\frac{1}{2}\left(x_k + \frac{3}{x_k}\right) = x_{k+1}$

& so on

$f(x) = x^2 - 3$

$x_0 = 2$

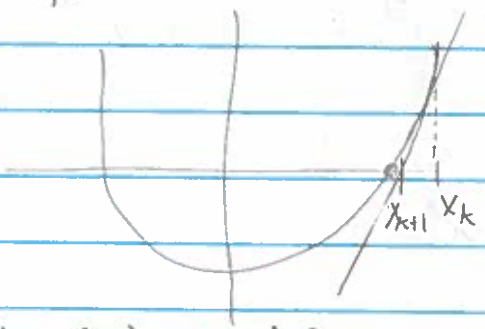
k	$f(x_k)$	$\frac{x_k - \sqrt{3}}{\sqrt{3}}$
0	1	1.55×10^{-1}
1	6.25×10^{-2}	1.04×10^{-2}
2	3.19×10^{-4}	5.31×10^{-5}
3	8.47×10^{-9}	1.41×10^{-9}
4	4.44×10^{-16}	$< 10^{-16}$

super-linear convergence

Lec12 - MAT226A - 10/23

Newton's Method

Def: Use linear approximation to find the root of $f(x)$



$$f(x) \approx L_k(x) = f(x_k) + f'(x_k)(x - x_k)$$

Solve $L_k(x_{k+1}) = 0$

$$\Rightarrow f(x_k) + f'(x_k)(x_{k+1} - x_k)$$

b) Newton's Method: $$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Ex: $f(x) = x^2 - 3$, $f'(x) = 2x$

$$\Rightarrow x_{k+1} = x_k - \frac{x_k^2 - 3}{2x_k}$$

$$= \frac{1}{2}\left(2x_k - x_k + \frac{3}{x_k}\right)$$

same as geometric algorithm! $\rightarrow$ $$x_{k+1} = \frac{1}{2}\left(x_k + \frac{3}{x_k}\right)$$

How fast does $e_k = x_k - \sqrt{3} \to 0$?

$$x_{k+1} - \sqrt{3} = -\sqrt{3} + \frac{1}{2}x_k + \frac{3}{2x_k} = \frac{x_k^2 - 2\sqrt{3}x_k + 3}{2x_k}$$

$$x_{k+1} - \sqrt{3} = (x_k - \sqrt{3})^2 / 2x_k$$

$$e_{k+1} = e_k^2 / 2x_k$$

If x_k is "close" to $\sqrt{3}$, $e_{k+1} \leq C e_k^2$

Newton's Method for Systems of eqns.

$\underline{F}(\underline{x})$. $\underline{F}: \mathbb{R}^n \to \mathbb{R}^n$

Solve $\underline{F}(\underline{x}) = \underline{0}$. Guess an initial soln, linearize $\underline{F}$ near guess.

$$\underline{F}(\underline{x}) \approx \underline{L}_k(\underline{x}) = \underline{F}(\underline{x}_k) + \underline{J}(\underline{x}_k)(\underline{x} - \underline{x}_k)$$

$\underline{J}$ → $J_{ij} = \dfrac{\partial \underline{F}_i}{\partial \underline{x}_j}$

Find $\underline{L}_k(\underline{x}_{k+1}) = \underline{0}$

$$\Rightarrow \underline{F}(\underline{x}_k) + \underline{J}(\underline{x}_k)(\underline{x}_{k+1} - \underline{x}_k) = \underline{0}$$

Define $\underline{\delta}_k = \underline{x}_{k+1} - \underline{x}_k \Rightarrow \underline{x}_{k+1} = \underline{x}_k + \underline{\delta}_k$

1. Solve $\underline{J}(\underline{x}_k)\underline{\delta}_k = -\underline{F}(\underline{x}_k)$
2. Update $\underline{x}_{k+1} = \underline{\delta}_k + \underline{x}_k$

How accurate is the linear approximation?

Supp. f is C^2. The approx $L(y) = f(x) + f'(x)(y-x)$ ignores quadratic terms. Taylor's thm says $\exists\, \xi$ btwn x & y s.t.

$$f(y) = L(y) + R(y)$$

where $R(y) = \dfrac{f''(\xi)}{2}(y-x)^2$.

Thus $\left| f(y) - f(x) - f'(x)(y-x) \right| = \left| \dfrac{f''(\xi)}{2}(y-x)^2 \right| \le \dfrac{\gamma}{2}(y-x)^2$

where $\gamma = \max |f''|$

For this result, don't need $f \in C^2$, only need f' to be Lipschitz cont's. w/ Lipschitz constant γ.

Recall: f is Lipschitz cont's iff $\exists\, \gamma$ s.t. $\forall x, y \in D$

$$|f(x) - f(y)| \le \gamma |x - y|$$

(strants w/ constant γ on D.

Lec 12 cont'd 10/23

If g is C^1, then g is Lipschitz cont's with constant $\gamma = \max|g'|$.

(Newton's Method) Convergence Thm: $f: D \to \mathbb{R}$ for an open interval D
f' is Lipschitz on D w/ constant γ.
(or $|f''| \le \gamma$ on D)

Suppose $|f'(x)| \ge \rho > 0$.

If $f(x^*) = 0$ for some $x^* \in D$, then there is some η s.t. if $|x_0 - x^*| < \eta$,

then $x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)} \longrightarrow x^*$

and $|x_{k+1} - x^*| \le \dfrac{\gamma}{2\rho} |x_k - x^*|^2$

Lec 13 - MAT226A - 10/25

- Flexible beam in low-Reynolds # fluid

$$\rho \frac{\partial y(x,t)}{\partial t} = -K \frac{\partial^4 y}{\partial x^4} + f(\,)?$$

↑ discretize space x → approx $\frac{\partial^4 y}{\partial x^4}$ → get system of ODEs

$$\frac{d\underline{y}}{dt} = A\underline{y} + \underline{f}$$

$$\frac{\underline{y}^{n+1} - \underline{y}^n}{\Delta t} = A\underline{y}^n + \underline{f}^n$$ ← requires tiny Δt. stiff ODE.

Use Backward-Euler:

$$\frac{\underline{y}^{n+1} - \underline{y}^n}{\Delta t} = A\underline{y}^{n+1} + \underline{f}^{n+1}$$ → solve system for $\underline{y}^{n+1}$ at each step

- large amplitude. $\rho \frac{\partial \underline{x}}{\partial t} = \underline{F}(\underline{x}) + g$ ← nonlinear function

Implicit method → Backward-Euler: $\frac{\underline{x}^{n+1} - \underline{x}^n}{\Delta t} = \underline{F}(\underline{x}^{n+1}) + g^{n+1}$

Stick all on one side & root find for $\underline{x}^{n+1}$

↳ use previous time step as initial guess in root find.

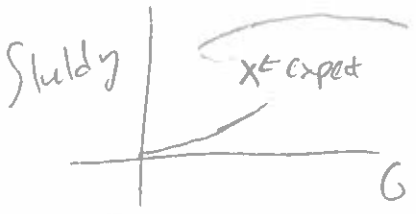
$$\rho u_{yy} - \theta u - G = 0$$
$$u(0,t) = u(1,t) = 0$$

flow → { { { { { → $u(x,t)$

↑ chemical fibers have reaction →

$$\theta_t = 1 - e^{|u|}\theta$$

Alternate PDE & ODE solves →



← steady state $\rho u_{yy} - e^{-|u|}u - G = 0$

- run time, backward to get unstable branch
  ↳ doesn't work on saddle!

Instead, do root-find → guess midpoint btwn stable states & other two to find unstable root! Find one, perturb G & use root as guess
↳ "continuation"

Convergence of Newton's Method Pf:

We have that $|f'(x)| \geq \rho > 0$ & $|f''(x)| < \gamma$ ← Lipschitz continuous or f' / const.

Statement: If we start close enough, quadratic convergence.

Let $\tau \in (0,1)$, $\hat{\eta}$ = radius of largest open interval around root x^* in D

Show by induction that: $|x_{n+1} - x^*| \leq \tau |x_n - x^*| < \eta$ (convergence)

(η s.t. $|x_0 - x^*| < \eta \Rightarrow x_n \to x^*$)

Let $\eta = \min\left\{\hat{\eta}, \tau\left(\frac{2\rho}{\gamma}\right)\right\}$.

Suppose $|x_0 - x^*| < \eta$.

Then $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \Rightarrow x_1 - x^* = x_0 - x^* - \frac{f(x_0) - f(x^*)}{f'(x_0)}$

$$= \frac{|f'(x_0)(x_0 - x^*) - f(x_0) + f(x^*)|}{f'(x_0)}$$

($f(x^*) = 0$, $-L_0(x^*)$)

Last time, $|f(x^*) - L_0(x^*)| \leq \frac{\gamma}{2}(x_0 - x^*)^2$

$$\Rightarrow |x_1 - x^*| \leq \frac{1}{|f'(x_0)|} \frac{\gamma}{2}(x_0 - x^*)^2 \leq \frac{\gamma}{2\rho}(x_0 - x^*)^2$$

Have $|x_0 - x^*| < \eta \leq \tau\left(\frac{2\rho}{\gamma}\right) \Rightarrow \frac{\gamma}{2\rho}|x_0 - x^*| < \tau$

$$\Rightarrow |x_1 - x^*| < \tau |x_0 - x^*|. \quad \text{// base case}$$

Then assume for x_n, show x_{n+1} (same thing).

Hence convergence. □

Lec 13 cont'd 10/25

If γ is small ($|f''| < \gamma$), η gets big → larger starting radius for convergence

e.g. $\gamma = 0$, $|f''| = 0$ everywhere ⇒ linearization is great approx

Newton's method on linear problem converges in 1 step indep. of starting value!

$$f(x) = Ax - b$$

$$J = A \Rightarrow x_1 = x_0 - A^{-1}(Ax_0 - b) = A^{-1}b \text{ (exact root)}$$

Imagine $f(x) \pm Ax - b + h.o.t.$

$\Rightarrow \frac{\gamma}{\rho}$ measures linear/quadratic term ratio.

This convergence thm. gives local convergence.

- Something like bisection always converges if you have an interval where f changes sign & is cont's.
- Would like a similar statement for Newton's method (cont'd)

Ex: Root of $\tan^{-1}(x) = 0$.

Newton's method: $x_{n+1} = x_n - (1 + x_n^2)\tan^{-1}(x_n)$

k	x_k
0	1
1	-5.7×10^{-1}
2	1.2×10^{-1}
3	-1.1×10^{-3}
4	8×10^{-10}
5	$< \varepsilon_{machine}$

vs.

k	x_k
0	2
1	-3.5
2	14
3	-280

What is close enough for an initial guess?

Should try half-steps or line-search type alg to avoid overshooting

Lec 14

10/27

For $\arctan x$ $\exists\ \bar{x} \approx 1.39\ldots$

s.t. Newton's method yields a period 2 cycle

$x_k = \bar{x}, -\bar{x}, \bar{x}, \ldots$

Thus Newton's method will converge for $\arctan x$ iff $x_0 \in (-\bar{x}, \bar{x})$

This is not great! Want an algorithm that doesn't rely on x_0 for convergence!

Lec 14 - MATH226A - 10/27

Let's make Newton's method globally convergent w/ a hybrid method to combine something like bisection method w/ Newton when close.

Simple idea for scalar problems: Require $|f(x_{n+1})| < |f(x_n)|$

First: $x_+ = x_n - \dfrac{f(x_n)}{f'(x_n)}$

while $(|f(x_+)| \geq |f(x_n)|)$

$\quad x_+ = \dfrac{x_n + x_+}{2}$

end

Next: $x_{n+1} = x_+$

Does this work? Yes, but why?

$$\frac{d}{dx} f^2(x) = 2f(x)f'(x)$$

$$x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{2f(x_n)f'(x_n)}{2(f'(x_n))^2} = x_n - \beta\left(\frac{d}{dx} f^2 \Big|_{x_n}\right)$$

(β: some direction; $\beta \to 0$)

Know that for β small enough, f^2 goes down in this iteration

Extend to higher dims:

$$\|\underline{F}\|^2 = \underline{F}^T\underline{F} = \sum_k \underline{F}_k^2$$

$$\left(\nabla \|\underline{F}\|^2\right)_i = \frac{\partial}{\partial x_i} \sum_k \underline{F}_k^2 = \sum_k 2\underline{F}_k \cdot \frac{\partial \underline{F}_k}{\partial x_i} = (2\underline{J}^T\underline{F})_i$$

Newton step: $\underline{x}_{n+1} = \underline{x}_n - \underline{J}^{-1}(x_n)\underline{F}(x_n)$

$= \underline{x}_n + \underline{p}_n$

(compute $\underline{p}_n$ · $\nabla\|F\|^2 = 2(\underline{J}^T\underline{F})^T p = 2\underline{F}^T\underline{J}(-\underline{J}^{-1}\underline{F})$

$= -2\underline{F}^T\underline{F} = -2\|\underline{F}\|^2 < 0$

$\underline{p}_n$: search direction; $\nabla\|F\|^2$: uphill.

$\Rightarrow \underline{p}_n$ points in a direction of decreasing $\|\underline{F}\|$!

Line Search / Backtracking algorithm:

Idea: Suppose $x_+ = \underline{x}_n + \underline{p}_n$ w/ $\underline{p}_n = -\underline{J}^{-1}(x_n)\underline{F}(x_n)$

& supp. $\|\underline{F}(x_+)\| \geq \|\underline{F}(x_n)\|$

Search along line $\underline{x}_+(\lambda) = \underline{x}_n + \lambda \underline{p}_n$ $\lambda \in (0,1)$

so that $\|\underline{F}(\underline{x}_+(\lambda))\| < \|\underline{F}(\underline{x}_n)\|$

Pseudocode:

$p = -\underline{J}^{-1}(x_n)\underline{F}(x_n)$

$\lambda = 1$

$x_+ = x_n + \lambda p$

while $(\|F(x_+)\| \geq \|F(x_n)\|)$ ←

  $\lambda = \lambda/2$

  $x_+ = x_n + \lambda p$

end

$x_{n+1} = x_+$

make fancy by changing this to

$\|F(x_n + \lambda p)\| \geq (1 - \alpha\lambda)\|F(x_n)\|$

$\alpha \in (0,1)$

e.g. $\alpha = 10^{-4}$ in practice

guarantees $\|F\| \to 0$

not just decreasing

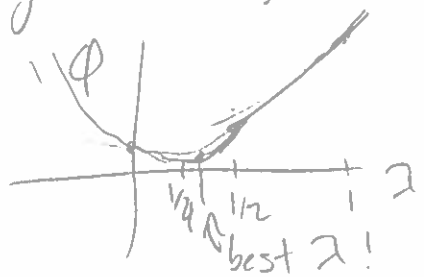
Lec 14 cont'd

In line search alg, use info from failed attempts!

Define $\phi(\lambda) = \|F(x_n + \lambda p)\|^2$

Supp. $\phi(0) < \phi(1)$

Try $\lambda = 1/2$, get $\phi(1/2) > \phi(0)$ (fails again)



maybe optimal $\lambda > 1/4$ but less than $1/2$.

Lec 15 - 10/30

Closing thoughts on Newton's method

- Terminating the iteration - want $F(x) = 0$
  - generically. stop when $\|F\|$ is small - absolutely/relatively?
    - relative to what though? $\|F(x^1)\| = 0$ can't divide! ↳ $\|F(x_n)\| < \tau_a$
    - ↳ $\|F(x_k)\| < \tau_r \|F(x_0)\|$. ← Use both in some combo
      - ↑ sensitive to initial guess
      - bad guess ⇒ early stop
      - good guess ⇒ might be too hard a bound
      - ignores scale of problem
  - Also can stop if $\|x_{n+1} - x_n\| < \tau \times \|x\|$
    - ↑ scale of problem, if don't know say $\|x_n\|$.

What if you don't have the Jacobian in Newton's method?

↳ Can use finite differences to approx. Jacobian

$$J_{ij} = \frac{F_i(x + h e_j) - F_i(x)}{h}$$

→ Typically, $h = \sqrt{\varepsilon_{mach}} \cdot \max(|x_j|, 1) \operatorname{sign}(x_j)$

Alternative: Broyden's method - each iteration update approx. Jacobian w/ current information

Jacobian-free Newton-Krylov

Krylov: class of iterative methods for linear systems that involve only matrix-vector products

- Approximately solve

$$J\delta x = -F$$

using Krylov method - e.g. gmres, conj. gradient (CG).

which requires only a routine to compute Jw.

↳ why solve $J\delta x = -F$ tightly? Why not half-ass solve it? Or tenth-ass solve it?

- Too tight a tolerance - spend a lot on linear iterations (inner) - small η
- Too loose a tolerance - spend a lot on nonlinear (outer) iterations (Newton) - large η

★ Solve $J\delta x = -F$ iteratively until $\|J\delta x + F\| \le \eta \|F\|$ ← relative tolerance

→ $\eta = 10^{-1}$ or 10^{-2}?

★ Good performance requires pre-conditioning!!! ★★★

↳ $BJ\delta x = -BF$ where B invertible

Ideally, $B \approx J^{-1}$ ⇒ system really solved to start

Lec 15 - MAT226A - 10/30

Least-Squares - overdetermined linear systems

$$A\underline{x} = \underline{b} \quad \text{w/ } A \in \mathbb{R}^{m\times n},\ m > n$$

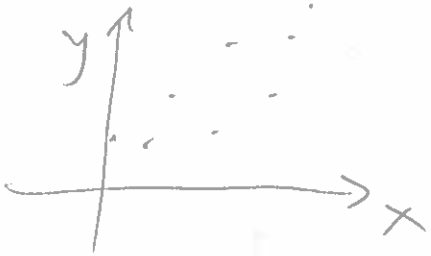
$$x \in \mathbb{R}^{n\times 1}$$

$$b \in \mathbb{R}^{m\times 1}$$

more equations than unknowns!

overdetermined - generally, no soln.

Ex: Fit data w/ a polynomial



data (x_i, y_i) $i = 1, \dots, m$

Fit w/ a poly. of deg. $n-1$

$$p(x) = c_0 + c_1 x + c_2 x^2 + \dots + c_{n-1} x^{n-1}$$

- would like: $p(x_1) = y_1$, $p(x_2) = y_2$, $\dots$, $p(x_m) = y_m$

$$\rightarrow \begin{aligned} c_0 + c_1 x_1 + c_2 x_1^2 + \dots + c_{n-1} x_1^{n-1} &= y_1 \\ &\vdots \\ c_0 + c_1 x_m + c_2 x_m^2 + \dots + c_{n-1} x_m^{n-1} &= y_m \end{aligned}$$

$$\Rightarrow \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{n-1} \end{bmatrix}}_{m\times n} \underbrace{\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix}}_{n\times 1} = \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_{m\times 1}$$

When $m > n$, expect no solution!

$Ax = b \rightarrow r = b - Ax$ ← want residual as small as possible.

$$\underbrace{\begin{pmatrix} & \\ & \end{pmatrix}}_{m\times n,\ m>n} \underbrace{\begin{pmatrix} \\ \end{pmatrix}}_{n\times 1} = \underbrace{\begin{pmatrix} \\ \\ \end{pmatrix}}_{m\times 1}$$

Least squares soln

Find $x \in \mathbb{R}^n$ to minimize

$$\|r\|_2^2 = \|b - Ax\|_2^2$$

How to solve? Try calculus:

$$\mathcal{E} = \|b - Ax\|_2^2$$

Find x to minimize $\mathcal{E}$.

$$\mathcal{E} = (b-Ax)^T(b-Ax)$$
$$= b^Tb - x^TA^Tb - b^TAx + x^TA^TAx$$

$$\frac{\partial \mathcal{E}}{\partial x_j} = -\frac{\partial}{\partial x_j}\sum_{i=1}^{n} x_i (a^T)_i b_i - \frac{\partial}{\partial x_j}\sum_{i=1}^{n} b_i a_i x_i + \frac{\partial}{\partial x_j}\sum_{i=1}^{n} x_i^2 (a^T)_i a_i$$
$$= -(a^T)_j b_j - b_j a_j + 2x_j (a^T)_j a_j$$

$$\nabla \mathcal{E} = -2A^Tb + 2A^TAx$$

lec 16

$$\begin{cases} \nabla \mathcal{E} = 0 \\ A^TAx = A^Tb \end{cases}$$

normal eqns → if normal eqns have soln x, does it yield a min of $\mathcal{E}$?

$$\nabla\nabla \mathcal{E} = 2A^TA$$

x min $\mathcal{E}$ if A^TA is pos-def.

$\Rightarrow$ $y^TA^TAy > 0$ for $y \neq 0$

$\|Ay\|_2^2 > 0$ if $\text{null}(A) = \emptyset$

need A full-rank

Lec 16 - MATL226A - 11/1

$\begin{pmatrix} & A & \\ & & \end{pmatrix} \begin{pmatrix} \\ y \\ \\ \end{pmatrix}$ = lin comb. of cols

↳ if cols of A are lin ind.

$Ay \neq 0$ for $y \neq 0$

$\Rightarrow$ A full rank $\Rightarrow$ A^*A is pos-def & hermitian

Suppose A full-rank, so $(A^TA)^{-1}$ exists

hence $x = (A^TA)^{-1}A^Tb$

is the soln to the normal eqns: $\nabla \mathcal{E} = 0$

$A^TAx = A^Tb$

hence x is the unique minimizer of $\mathcal{E}$.

$(A^TA)^{-1}A^T = A^+$ is the Moore-Penrose pseudoinverse of A.

↳ note $A^+A = I_{n\times m}$ (since $\underset{n\times m}{A^T}\underset{m\times n}{A} = n\times n$ $\Rightarrow \underset{n\times n}{(A^TA)^{-1}}\underset{n\times m}{A^*} = n\times m$)

Solving normal eqns -> Assume A full rank

$A^*A\,x = A^*b$

pos-def. & hermitian -> use Cholesky factorization to solve for $(A^*A)^{-1}$.

↳ fastest algorithm for least-squares.

Cost of solving normal eqns $\sim \frac{1}{3}n^3$ to solve $x = (A^*A)^{-1}A^*b$

cost of $\underset{n\times m}{A^*}\underset{m\times 1}{b} \sim \underbrace{2mn}_{\text{cost per element of product}}$ (number of elements of product)

cost of $A^*A \sim \underbrace{2m}_{\text{cost of element}}\underbrace{n^2}_{\text{no. of elements}} \sim 2mn^2$ (exploit symmetry) ← $m \geq n$ so most expensive step

Backslash in MATLAB for rectangular $A = m \times n$, $m > n$

$x = A \backslash b$ gives Least-squares soln

(but uses a different algorithm)

If the condition # of A is large, introduce error A^*b

& condition # of A^*A is worse than A roughly κ_A^2.

Other methods (of increasing expense & robustness) for LS:

"standard" method for Least-Squares (used in MATLAB $x = A \backslash b$)

↳ QR factorization

$A = \underbrace{Q}_{\text{orthogonal}} R$ → upper-triangular

↳ easy to invert, $\kappa(Q) = 1$ → doesn't introduce error!

Most robust way → for singular problems / nearly-singular (very poorly conditioned A)

↳ SVD: $A = U \Sigma V^*$] most expensive

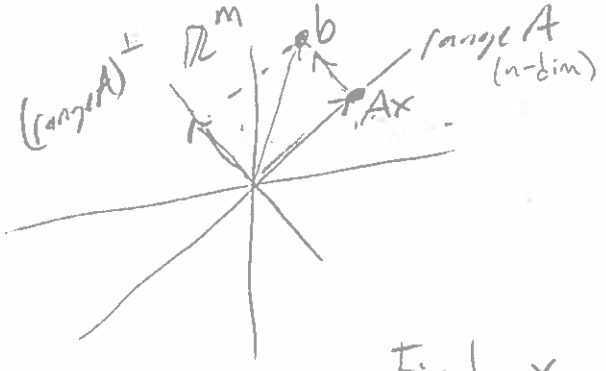
(U, V^*: orthogonal; Σ: diagonal)

For square A, work for QR $\sim \frac{4}{3} n^3$.

SVD involves using QR repeatedly, so more expensive

Lec 16 cont'd - 11/1

- Geometric view of Least Squares



A is $m \times n$, $m > n$

range of A is an n-dimensional subspace of $\mathbb{R}^m$

Generically, $b \in \mathbb{R}^m$ but $b \notin \text{range}A$.

Find x s.t. Ax is as close as possible to b.

looks like Ax should be the orthogonal projection of b onto range$A$.

Decompose $\underline{b} = \underbrace{\underline{b}_0}_{(\text{range}A)^\perp} + \underbrace{\underline{b}_r}_{\text{range}(A)}$ s.t. $\langle \underline{b}_0, \underline{b}_r \rangle = 0$

- Then $\|\underline{b} - A\underline{x}\|_2^2 = \|\underbrace{\underline{b}_0}_{\perp \text{ to range}A} + \underbrace{\underline{b}_r - A\underline{x}}_{\in \text{range}(A)}\|_2^2 = \|b_0\|_2^2 + \underbrace{\|\underline{b}_r - A\underline{x}\|_2^2}_{\text{can make } 0 \text{ for some } x}$

Claim: For x the soln to the LS problem, the residual is orthogonal to the range of A ($\underline{e} = \underline{b}_0$).

$r = b - Ax$

$\langle r, Ax \rangle = (Ax)^*(b - Ax) = x^* A^*(b - Ax) = x^*(A^*b - A^*Ax)$

For x the soln to the normal eqns, $x = (A^*A)^{-1}A^*b$

$\Rightarrow \langle r, Ax \rangle = 0$ □

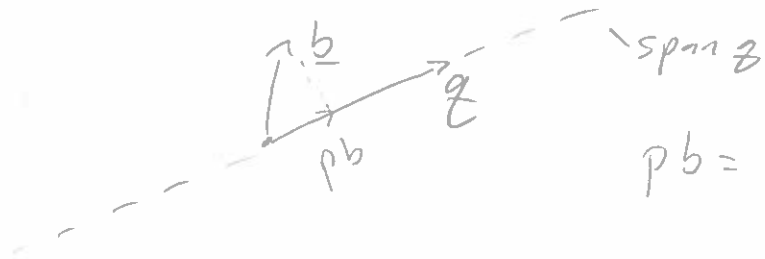
- Given $\underline{b}$, how to find its orthogonal projection onto range$A$?

You may be familiar w/ orthogonal projections onto single vector

Lec 17 - MATH 22CA - 11/3

Given $\underline{b}$, how to find orthogonal projection onto range A?

Recall: Given $\underline{b}$, projection onto vector $\underline{a}$:



$$pb = \underbrace{\frac{a^*b}{a^*a}}_{\text{scalar}}\underbrace{a}_{\text{dimen.}} = \underbrace{\frac{aa^*}{a^*a}}_{\text{matrix}} b$$

Claim: $Pb = \frac{aa^*}{a^*a} b$, $b - Pb \perp a$.

$$a^*(b - Pb) = a^*\left(b - \frac{aa^*}{a^*a}b\right) = a^*b - \left(\frac{a^*a}{a^*a}\right)a^*b = 0$$

Def: A square matrix P is a projector if $P^2 = P$.

A projector is orthogonal if $P^*(I - P) = 0$.

equivalently, if $P^* = P$.

Ex: $P = \frac{aa^*}{a^*a} \Rightarrow P^2 = \frac{a\cancel{a^*}\,\cancel{a}a^*}{\cancel{a^*a}\,\cancel{a^*a}} = \frac{aa^*}{a^*a} = P$.

$$P^* = \left(\frac{aa^*}{a^*a}\right)^* = \frac{aa^*}{a^*a} = P \Rightarrow$$ our example indeed was an orthogonal projector.

Lec 17 Cont'd - 11/3

Project into n-dim'l space spanned by the orthonormal basis $\{q_1, \dots, q_n\}$

↳ recall $\langle q_i, q_j \rangle = \delta_{ij}$

$$Pb = q_1(q_1^* b) + q_2(q_2^* b) + \dots + q_n(q_n^* b)$$

$$= \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} q_1^* b \\ q_2^* b \\ \vdots \\ q_n^* b \end{pmatrix}$$

$$= \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} — & q_1^* & — \\ — & q_2^* & — \\ & \vdots & \\ — & q_n^* & — \end{pmatrix} b$$

$$= QQ^* b$$

$\Rightarrow P = QQ^* \Rightarrow P^2 = QQ^*QQ^* = QQ^* = P.$

($Q^*Q = I$ by orthonormality of $\{q_1 \dots q_n\}$)

$P^* = (QQ^*)^* = QQ^* = P$

$\Rightarrow$ P is an orthogonal projector.

If we get an orthonormal basis for range A, how does it help solve the LS problem?

This generates a reduced QR factorization

$$A = \underset{m \times n}{\hat{Q}} \underset{n \times n}{\hat{R}}$$ upper triangular

$m \times n$ — orthonormal columns

$Ax = b \Rightarrow \hat{Q}\hat{R}x = b \rightarrow \hat{Q}^*\hat{Q}\hat{R}x = \hat{Q}^*b$

($\hat{Q}^*\hat{Q} = I$)

$\rightarrow \hat{R}x = \hat{Q}^*b$ ← linear system on range A, solve

Lec 18 · MAT226A · 11/6

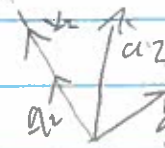
QR factorization - goal is to find orthonormal basis for range A

- Given matrix A $m \times n$, $m \geq n$, find an orthonormal basis for range A.

- range A is spanned by the cols of A: $\{\underline{a}_1, \ldots, \underline{a}_n\}$ span range A
  - ↳ want to orthogonalize this basis

Gram-Schmidt Orthogonalization: Work sequentially.

Given $\{\underline{a}_1, \underline{a}_2\}$, let $\underline{q}_1 = \frac{\underline{a}_1}{\|\underline{a}_1\|_2}$ ← spans same space as $\{\underline{a}_1\}$



project off part of $\underline{a}_2$ in $\underline{q}_1$ direction, left w/ projection of $\underline{a}_2$ onto $\underline{q}_1^{\perp}$.

$$\underline{v}_2 = \underline{a}_2 - \underline{q}_1(\underline{q}_1^* \underline{a}_2)$$

$$\underline{q}_2 = \frac{\underline{v}_2}{\|\underline{v}_2\|_2}$$

$$\underbrace{\begin{pmatrix} | & | \\ \underline{a}_1 & \underline{a}_2 \\ | & | \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} | & | \\ \underline{q}_1 & \underline{q}_2 \\ | & | \end{pmatrix}}_{\hat{Q}} \underbrace{\begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}}_{\hat{R}}$$

w/ $r_{11} = \|\underline{a}_1\|_2$, $r_{12} = \underline{q}_1^* \underline{a}_2$, $r_{22} = \|\underline{a}_2 - r_{12}\underline{q}_1\|_2$

Easy to generalize - for n^{th} basis vector, subtract off $n-1$ projections

Classical Gram-Schmidt (unstable algorithm)

```
for j = 1:n
    v_j = a_j
    for i = 1:j-1
        r_ij = q_i* a_j
        v_j = v_j - r_ij q_i
    end i loop
    r_jj = ||v_j||_2
    q_j = v_j / r_jj
```

Lec 18 cont'd - 11/6

What did classical Gram-Schmidt produce?

↳ Gives the reduced QR factorization

$$\underset{m\times n}{A} = \underset{\substack{m\times n\\ \text{orthonormal cols.}}}{\hat{Q}}\ \underset{\substack{n\times n\\ \text{upper-triangular}}}{\hat{R}}$$

Could adapt this to generate full QR-factorization

$$\underset{m\times n}{A} = \underset{m\times m}{Q}\ \underset{m\times n}{\begin{pmatrix}\hat{R}\\ 0\end{pmatrix}}$$

($\hat{R}$: n rows; 0: $m-n$ rows)

$Q = \begin{pmatrix}\hat{Q} & \vdots\end{pmatrix}$ (n cols | $m-n$ cols)

orthogonal cols

extra vectors from last $m-n$ cols of Q are a basis for $(\text{range } A)^\perp = \text{null}(A^*)$.

Modified Gram-Schmidt (stable) - project off q_i from cols $j > i$ (work row-by-row in R instead of col-by-cols)

```
V = A
for i = 1:n
    r_ii = ||v_i||_2
    q_i = v_i / r_ii
    for j = i+1:n
        r_ij = q_i* v_j
        v_j = v_j - r_ij q_i
    end j loop
end i loop
```

Modified GS

1st step:

$$\begin{pmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} \frac{1}{r_{11}} & -\frac{r_{12}}{r_{11}} & \cdots & -\frac{r_{1n}}{r_{11}} \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} = \begin{pmatrix} | & | & & | \\ q_1 & v_2^{(2)} & \cdots & v_n^{(2)} \\ | & | & & | \end{pmatrix}$$

R_1

Next step:

$$R_2 = \begin{pmatrix} 1 & 0 & \cdots & \\ & \frac{1}{r_{22}} & -\frac{r_{23}}{r_{22}} & \cdots & -\frac{r_{2n}}{r_{22}} \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

$$\Rightarrow A \underbrace{R_1 R_2 \cdots R_n}_{\hat{R}^{-1}} = \hat{Q}$$

How expensive is GS?

Work of inner loop: $r_{ij} = \underline{q}_i^* \underline{v}_j$ ← inner prod of vectors length m $= 2m$ flops

↳ $4m$ flops inside

$\underline{v}_j = \underline{v}_j - r_{ij} \underline{q}_i$ $= 2m$ flops

Total work: $\sum_{i=1}^{n} \sum_{j=i+1}^{n} 4m \sim 2mn^2$

3rd method for QR: Householder method

$$Q_n \cdots Q_2 Q_1 A = R$$

GS = triangular orthogonalization - use repeated triangularR's to get Q

HH = orthogonal triangularization - use repeated with Q's to get R

★ HH will be faster & more accurate since all Q's have $\kappa = 1$

↳ Superior for LS when A has poor conditioning.

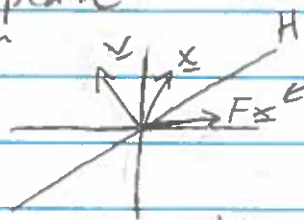
Lec 19 - MAT226A - 11/8

Gram-Schmidt: $AR_1R_2\cdots R_n = \hat{Q}$

Householder: $Q_n\cdots Q_2Q_1A = R$

The Q_k's involve Householder reflections through a hyperplane.
Define hyperplane through the origin by the normal vector $\underline{v}$.

In $\mathbb{C}^m$, the space orthogonal to $\underline{v}$ is a $m-1$ dim'l space, ie. a hyperplane



$$\underline{x} = \underbrace{P_v\underline{x}}_{\perp \text{ projection onto span}\{v\}} + \underbrace{(I-P_v)\underline{x}}_{\perp \text{ projection onto } H}$$

$$\Rightarrow F\underline{x} = -P_v\underline{x} + (I-P_v)\underline{x} = (I-2P_v)\underline{x}$$

$$\Rightarrow F = I - 2P_v$$

Recall, P_v is an orthogonal projector, so $P_v^2 = P_v$, $P_v^* = P_v$

Claim: F is orthogonal.

Pf: $$F^*F = (I-2P_v)^*(I-2P_v)$$
$$= (I - 2\underbrace{P_v^*}_{=P_v})(I-2P_v)$$
$$= I - 4P_v + 4\underbrace{P_v^2}_{=P_v}$$
$$= I$$

□

How do we use F to triangularize A?

Goal: $\begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix} \xrightarrow{Q_1} \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{pmatrix} \xrightarrow{Q_2} \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \end{pmatrix} \xrightarrow{Q_3} \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{pmatrix}$

A $\quad Q_1 A$ $\quad Q_2 Q_1 A$ $\quad Q_3 Q_2 Q_1 A = R$

Q_2 only looks at this submatrix

$\hat{R}$ in 6

In designing Q_i, only care about col i & setting subdiag entries to 0.

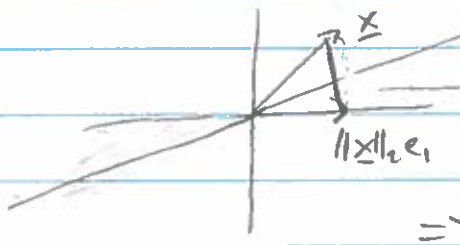
↳ Figure out Q_1, then get Q_2 consider submatrix & so on.

$$Q_k = \begin{bmatrix} I & \\ & F \end{bmatrix}$$

where
I is $k-1 \times k-1$
F is $m-(k-1) \times m-(k-1)$ reflector

Why? $\begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}\begin{bmatrix} T & B \\ 0 & C \end{bmatrix} = \begin{bmatrix} T & B \\ 0 & FC \end{bmatrix}$ ← preserves upper-triangular structure!

Goal: $Fy = F\begin{pmatrix} x \\ \vdots \\ x \end{pmatrix} = \begin{pmatrix} \pm\|y\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ ← since F orthogonal



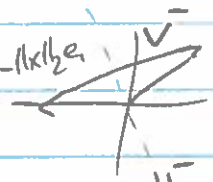
vector v connecting x & $\|x\|_2 e_1$ is $\perp$ to H

$\Rightarrow P_v = vv^*/v^*v$

$\Rightarrow v = \|x\|_2 e_1 - x$

$$F = I - 2\frac{vv^*}{v^*v}$$

Could use $v = z\|x\|_2 e_1 - x$ where $|z| = 1$.



Choice of $z = \pm 1$ matters for stability!

Lec 19 cont'd - 11/8

In Householder reflection, reflection H^+ or H^- matters for stability!
↳ v^+ ↳ v^-

Suppose $x_1 > 0$:

$v^+ = \|x\|_2 e_1 - x$ ⇝ if x_1 is close to $\|x\|_2$ → subtracting 2 nearly equal numbers! = big loss of precision.

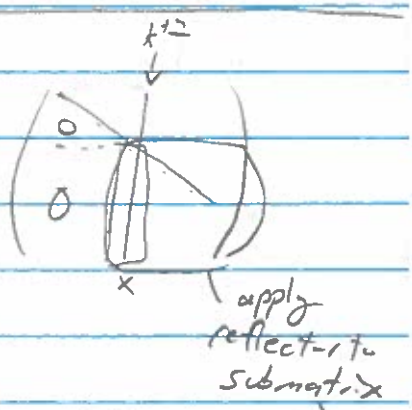
$v^- = -\|x\|_2 e_1 - x$ ↳ adding 2 nearly equal #'s = no big deal

So choose

$$v = \text{sign}(x_1)\|x\|_2 e_1 + x$$

w/ $\text{sign}(x_1) = \begin{cases} +1, & x_1 \geq 0 \\ -1, & x_1 < 0 \end{cases}$

Householder Algorithm



for $k = 1:n$ (loop over cols)

$x = A(k:m, k)$

$v_k = \text{sign}(x_1)\|x\|_2 + x$

$v_k = v_k / \|v_k\|_2$

$A(k:m, k:n) = \underbrace{A(k:m, k:n)}_{IA} - \underbrace{2 v_k (v_k^* A(k:m, k:n))}_{-2P_vA}$

end (k loop)

Work $\sim 2mn^2 - \frac{2}{3}n^3$

If $m = n$, work $\sim \frac{4}{3}n^3$ (2× cost of LU)

This method generates R.

$QRx = b$

$Rx = Q^* b$, ... use this for linear solve ... ⇒

Solve linear system $Ax = b$ w/ QR:

$$QRx = b$$

$$Rx = \underbrace{Q^* b}$$

only need this for linear solve

now Householder gives

$$\underbrace{Q_n \cdots Q_2 Q_1}_{=Q^*} A = R$$

we had the application of Q^* in algs.

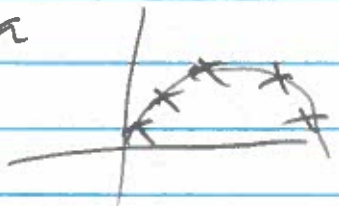
Add:

```
for k = 1:n
    b(k:m) = b(k:m) - 2 v_k (v_k^* b(k:m))
end (k loop)
```

Lec 20 - MAT226A - 11/13

Least-Squares & Orthogonal Polynomials

Intro - LS polynom fit to data



N.w. given cont's fn. f on some interval, find a polynomial to approximate f.

Let $f \in C([a,b])$

The "data" to fit is a cont's fn.

Use a degree n poly. $P_n(x) = \sum_{k=0}^{n} c_k x^k$

$r(x) = f(x) - P_n(x)$ → residual is a cont's fn.

Same approach - use 2-norm to measure residual & minimize

$$\mathcal{E} = \|r(x)\|_2^2 = \int_a^b (r(x))^2 dx$$

$$= \int_a^b \left(f(x) - P_n(x)\right)^2 dx$$

$$= \int_a^b \left(f(x) - \sum_{k=0}^{n} c_k x^k\right)^2 dx$$

Find c_k's where $\nabla_{c_k} \mathcal{E} = 0$

Find c_k's where $\nabla_{c_k} \mathcal{E} = 0$, i.e.,

$$\frac{\partial \mathcal{E}}{\partial c_j} = 0 \quad \text{for } j = 0, \ldots, n$$

$$\mathcal{E} = \int_a^b \left(f(x) - \sum_{k=0}^{n} c_k x^k \right)^2 dx$$

$$\frac{\partial \mathcal{E}}{\partial c_j} = \int_a^b 2 \left(f(x) - \sum_{k=0}^{n} c_k x^k \right) \cdot (-x^j)\, dx = 0$$

$$\int_a^b x^j \sum_{k=0}^{n} c_k x^k \, dx = \int_a^b f(x) x^j \, dx$$

$$\sum_{k=0}^{n} c_k \int_a^b x^j x^k \, dx = \int_a^b f(x) x^j \, dx$$

$$\sum_{k=0}^{n} c_k \langle x^j, x^k \rangle = \langle f(x), x^j \rangle \rightarrow n+1 \text{ of these eqns}$$

★ obviously should have chosen orthog

Do this for $j = 0, \ldots, n$ to get

$$A\underline{c} = \underline{b} \rightarrow \text{normal eqns}$$

w/ $b_j = \langle f(x), x^j \rangle$

& $A_{jk} = \langle x^j, x^k \rangle$

If $a = 0$, $b = 1$, i.e. $f \in C([0,1])$

$A_{jk} = \int_0^1 x^{j+k} \, dx$ is the Hilbert matrix

$\Rightarrow$ Very poorly conditioned!

n	$K(A)$
5	1.5×10^7
10	5.2×10^{14}

Lec 20 cont'd - 11/13

Back to discrete case $A\underline{x} = \underline{b}$

where A is $m \times n$ w/ $m \geq n$

$$A^* A \underline{x} = A^* \underline{b}$$

what if cols of A were orthogonal?

↳ A^*A is diagonal & $\chi(A^*) = 1$ so $A^*\underline{b}$ is nice!

↳ $A = QI$

$\Rightarrow A^*A\underline{x} = \underline{x} = A^*\underline{b}$. done!

Start w/ orthonormal cols. LS soln is trivial!

Don't use basis $\{1, x, x^2, \ldots, x^n\}$

for Π_n = space of polys of deg. $\leq n$.

Use an orthogonal basis!

↳ When we wrote out problem, we weren't given matrix A, we chose a basis & derived it. So choose orthogonal basis & re-derive A.

↳ $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$

$$P_n = \sum_{k=0}^{n} c_k \phi_k(x)$$

$$\xi = \int_a^b \left| f(x) - \sum_{k=0}^{n} c_k \phi_k(x) \right|^2 dx$$

$$\frac{\partial \xi}{\partial c_j} = \int_a^b 2\left(f(x) - \sum_{k=0}^{n} c_k \phi_k(x)\right)\left(-\phi_j(x)\right) dx = 0$$

$$\Rightarrow \sum_{k=0}^{n} c_k \langle \phi_j(x), \phi_k(x) \rangle = \langle f(x), \phi_j(x) \rangle$$

ϕ_i orthogonal $\Rightarrow c_j \langle \phi_j(x), \phi_j(x) \rangle = \langle f(x), \phi_j(x) \rangle$

Hence system of normal eqns becomes

$$A\underline{c} = \underline{b}$$

w/ $\underline{b}_j = \langle f(x), \phi_j(x) \rangle$

$A_{jk} = \langle \phi_j(x), \phi_k(x) \rangle$ $\int_{jk} \Rightarrow A = I$.

ϕ_j's orthogonal $\Rightarrow$ A diagonal.

$\Rightarrow$ $c_j = \langle f, \phi_j \rangle / \langle \phi_j, \phi_j \rangle$

This is the size of projection of f onto ϕ_j!

Where do we get an orthogonal basis?

↳ Gram-Schmidt works great analytically.

↳ Use GS on $\{1, x, x^2, \dots, x^n\}$ to generate orthogonal basis $\{\phi_0, \phi_1, \dots, \phi_n\}$

↳ obtains the Legendre polynomials if we chose L² inner product & interval $[-1, 1]$.

Legendre polys: $\{1, x, \frac{3}{2}x^2 - \frac{1}{2}, \frac{5}{2}x^3 - \frac{3}{2}x, P_4, \dots, P_n\}$

where $P_k = \left(\frac{2k-1}{k}\right) \times P_{k-1} - \left(\frac{k-1}{k}\right) P_{k-2}$

(If we did GS on $\{1, x, x^2\}$, we'd get $P_2 = x^2 - \frac{1}{3}$ instead

$\Rightarrow$ monic Legendre polys (same up to constant multiple.))

Lec 21 - MAT226A - 11/15

Legendre Polynomials - orthogonal in $\langle f,g\rangle = \int_{-1}^{1} f(x)g(x)dx$ w/ P_n a n^{th} degree poly.

Say $\{\phi_j(x)\}_j$ orthogonal on $(-1,1)$ → generate on (a,b)?

map $(a,b) \to (-1,1)$

$$s = \frac{2(x-a)-(b-a)}{b-a}$$

The polynomials $\{\phi_j(s(x))\}_j$ are orthogonal on (a,b).

↳ $$\int_a^b \phi_i\left(\frac{2(x-a)-(b-a)}{b-a}\right)\phi_j\left(\frac{2(x-a)-(b-a)}{b-a}\right)dx$$

Introduce $s=$ above, $ds = \frac{2}{b-a}dx$

$$\Rightarrow \int_{-1}^{1} \phi_i(s)\phi_j(s)\frac{b-a}{2}ds = 0 \quad i\neq j.$$

In HW3, solve discrete least-squares problem by sampling a function & fitting poly. to data (w/ monomial basis)
↳ get matrix A, 50×12, w/ $\chi(A)\approx 10^8$.

↳ Could have used Legendre pdys. as basis but these are not orthogonal wrt. discrete inner product (but they are nearly orthogonal, so $\chi(A_L)\approx 5$.

↳ If switched to 20^{th} degree monomial basis, have more basis vectors linearly dependent ⇒ worse conditioning $\chi(A)\approx 10^5$. Or w/ 20^{th} deg. Legendre pdys, $\chi(A_L)\approx 10$.

Other types of orthogonal polynomials — diff. domains — diff. inner product $\langle f,g\rangle = \int_a^b f(x)g(x)w(x)\,dx$; $w(x)\geq 0$

Type	Basis	Inner Product
Legendre	$P_n = \left(\frac{2k-1}{k}\right)xP_{k-1} - \left(\frac{k-1}{k}\right)P_{k-2}$	$\int_{-1}^{1} f(x)g(x)\,dx$ $(w(x)=1)$
Chebyshev	$T_k(x) = \cos(k\cos^{-1}(x))$	$\int_{-1}^{1} f(x)g(x)\underbrace{\frac{1}{\sqrt{1-x^2}}}_{w(x)}\,dx$
Trigonometric → $\{1, \cos kx, \sin kx\}$		$\int_{-\pi}^{\pi} f(x)g(x)\,dx$.
→ $\{e^{ikx}\}$		$\int_{-\pi}^{\pi} f(x)\overline{g}(x)\,dx$

★ Tchebyshev polys: orthog. in $\int_{-1}^{1} f(x)g(x)\frac{1}{\sqrt{1-x^2}}\,dx$

$T_0(x)=1,\ T_1(x)=x,\ T_2(x)=2x^2-1,\ T_3(x)=4x^3-3x$

$T_k(x) = \cos(k\cos^{-1}(x))$ ← k^{th} degree poly. ⇝ closely related to Fourier series

★ Approx. of periodic fns → f cont's w/ $f(x+p)=f(x)$

can restrict to a single period to approx. f on $[-\frac{p}{2}, p/2)$

Want $p_n(x) = \sum_{k=0}^{n} c_k\phi_k(x)$ ← use periodic $\phi_k(x)$'s & orthogonal

e.g. $\{1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots, \cos nx, \sin nx\}$ — 2π-periodic & orthogonal

Use $p_n(x) = a_0 + \sum_{k=1}^{n} a_k\cos kx + b_k\sin kx$

↑ n^{th} degree trigonometric poly.

Approx f w/ p_n by minimizing

$\|f(x)-p_n(x)\|_2^2 = \int_{-\pi}^{\pi}(f(x)-p_n(x))^2\,dx$

Lec 21 cont'd - 11/15

Approximating periodic f w/ orthogonal basis of trigonometric polynomials, get the normal eqns:

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

$$a_k = \frac{\langle f, \cos kx\rangle}{\langle \cos kx, \cos kx\rangle} = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)\,dx$$

$$b_k = \frac{\langle f, \sin kx\rangle}{\langle \sin kx, \sin kx\rangle} = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)\,dx$$

This $p_n(x)$ is a truncated Fourier series of $f(x)$.

So $\|f(x) - p_n(x)\|_2^2 \to 0$ as $n \to \infty$ for $f \in L^2$.

(uniform convergence! $\Rightarrow$ pointwise convergence).

Could use complex rep'n

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

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

to use the basis $\{e^{ikx}\}_{k=-n}^{n}$ for trig. polys instead!

$\{e^{ikx}\}$ are orthogonal in the inner product

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

Normal eqns for Least-squares problem

$$p_n(x) = \sum_{k=-n}^{n} c_k e^{ikx} \quad \text{where } c_k \text{ complex \& if } p_n \text{ is real, then } \overline{c_{-k}} = c_k.$$

$$c_k = \frac{\langle f, e^{ikx}\rangle}{\langle e^{ikx}, e^{ikx}\rangle} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-ikx}\,dx \leftarrow \text{Fourier transform!}$$

Lec 22 - MAT226A - 11/17

Recall: Chebyshev polys are orthogonal in

$$\langle f, g \rangle = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\, dx$$

$T_k(x) = \cos(k \cos^{-1}(x))$ ← will show this is a poly!

← analog to FS on non-periodic fns.

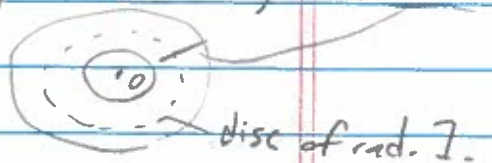
Recall: Deg. n trig approx to f which is 2π-periodic is a truncated Fourier sum

$$P_n(x) = \sum_{k=-n}^{n} c_k e^{ikx} \quad \text{where} \quad c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x) e^{-ikx}\, dx$$

If $f(x)$ is real, then $\overline{c_{-k}} = c_k$.

✡ Fourier series are related to Laurent series

$$f(z) = \sum_{k=-\infty}^{\infty} c_k z^k$$ ← If have this repn, f is analytic on annulus



evaluating Laurent series of f on circle of radius 1 → $z = e^{i\theta}$

$$\Rightarrow f(z) = f(e^{i\theta}) = \sum_{k=-\infty}^{\infty} c_k e^{ik\theta} = \text{Fourier series}$$

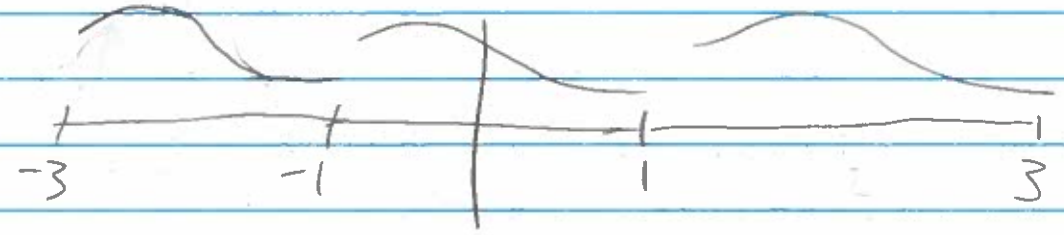
The c_k in Laurent series: $c_k = \frac{1}{2\pi i}\oint_{\gamma} \frac{f(z)}{z^{k+1}}\, dz$

for $z = e^{i\theta}$, this yields same c_k as FS.

Back to approximating f on $[-1,1]$

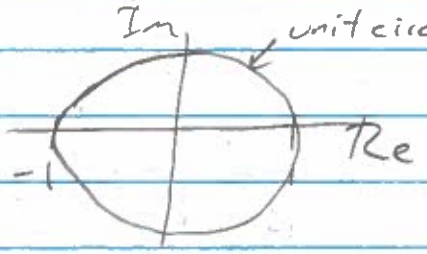
Can think of f as a periodic (but discont's) function

Lec 22 cont'd



FS converges in L^2 => not pointwise, very slow

↳ get Gibbs phenomena at discontinuities



Extend x to $\mathbb{C}$ & evaluate f on the unit circle to use Fourier series.

Change of vars: $f(x) = F(z)$

$$x = \frac{1}{2}\left(z + \frac{1}{z}\right)$$

Note if $z = e^{i\theta}$, $x = \frac{1}{2}(e^{i\theta} + e^{-i\theta}) = \cos\theta$

F has the symmetry (on the unit circle) $F(z) = F(1/z) = F(\bar{z})$

We can compute the Fourier series of $F(z)$

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} F(e^{i\theta}) e^{-ik\theta}\, d\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi} F(e^{i\theta}) e^{ik\theta}\, d\theta$$

↑ exploiting symmetry.

$$\Rightarrow \quad c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} F(z)\left(\frac{e^{ik\theta} + e^{-ik\theta}}{2}\right) d\theta$$

change back to x in integral

$$x = \cos\theta \Rightarrow \theta = \cos^{-1}(x) \qquad d\theta = \frac{dx}{\sqrt{1-x^2}}$$

$$\frac{e^{ik\theta} + e^{-ik\theta}}{2} = \cos(k\theta) = \cos\left(k\cos^{-1}(x)\right)$$

$$\Rightarrow \quad c_k = \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)\cos\left(k\cos^{-1}(x)\right)}{\sqrt{1-x^2}}\, dx, \qquad c_0 = \frac{1}{\pi}\int \cdots dx$$

$$c_k = \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)\cos(k\cos^{-1}(x))}{\sqrt{1-x^2}}\,dx \quad , \quad c_0 = \frac{1}{\pi}\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}\,dx$$

Thus the c_k's are the orthogonal projection of f onto the Chebyshev polynomials in the appropriate inner product.

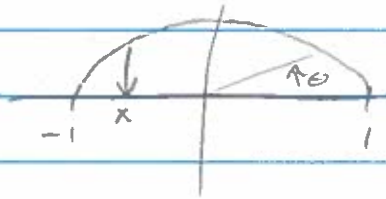
$T_k(x) = \cos(k\cos^{-1}(x))$ is the k^{th} Chebyshev poly.

Thm: If f is Lipschitz cont's on $[-1,1]$, it has a unique Chebyshev series representation

$$f(x) = \sum_{k=-\infty}^{\infty} c_k T_k(x) \quad \leftarrow \text{series converges uniformly (pointwise)}$$

$$\Rightarrow c_k = \frac{\langle f, T_k\rangle_{cw}}{\langle T_k, T_k\rangle_{cw}} \quad \text{where } \langle f,g\rangle_{cw} = \int_{-1}^{1}\frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx$$

Note: $T_k(x) = \mathrm{Re}(e^{ik\theta})$
$= \cos(k\theta)$
$= \cos(k\cos^{-1}(\theta))$



Recursion Relation for Chebyshev

$T_0(x) = \cos(0\cdot\cos^{-1}(x)) = 1$

$T_1(x) = \cos(\cos^{-1}(x)) = x$

$\theta = \cos^{-1}(x)$

$T_{k+1}(x) = \cos((k+1)\theta) = \cos(k\theta)\cos(\theta) - \sin(k\theta)\sin(\theta)$

$T_{k-1}(x) = \cos((k-1)\theta) = \cos(k\theta)\cos(\theta) + \sin(k\theta)\sin(\theta)$

$\Rightarrow T_{k+1}(x) + T_{k-1}(x) = 2x\cdot T_k(x)$

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$$
$$T_0 = 1,\ T_1 = x.$$

$\rightarrow$ Clearly, these are polys.

Lec 23 - MAT226A - 11/20

- Chebyshev polys: $T_n(x) = \cos(n \cos^{-1}(x))$

  Recursive form: $T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x)$

  $T_0(x) = 1$, $T_1(x) = x$

Properties of Chebyshev polys

- Roots & Extreme points:

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

$\cos\theta = x$, x on $[-1,1]$, θ on $[0,\pi]$

- Max/min of T_n: $n\theta = j\pi$
  - ↳ 1, -1

  $\theta = \frac{j\pi}{n}$, $j = 0, \dots, n$

  ← n fixed, degree of poly.

⇒ Extrema at $x_j = \cos\left(\frac{j\pi}{n}\right)$, $j = 0, \dots, n$

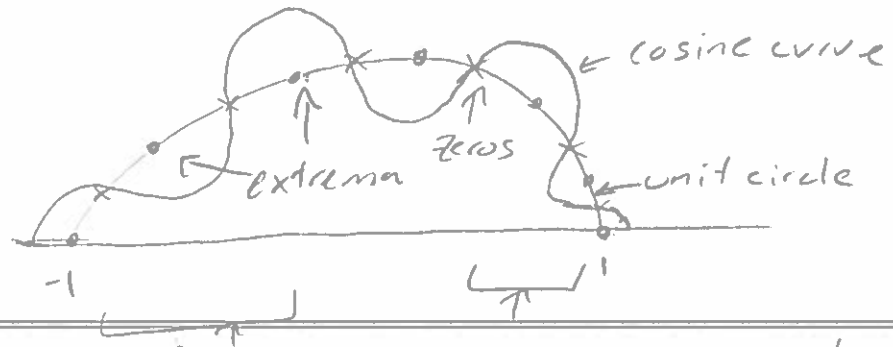
$T(x_j) = (-1)^j$

- Zeros of $T_n(x)$: $n\theta = (j - \frac{1}{2})\pi \rightarrow \theta \in [0,\pi]$ for $j = 1, \dots, n$

  ⇒ $x_j = \cos\left(\frac{(j-\frac{1}{2})\pi}{n}\right)$ ↳ not $j=0$!

⇒ n roots of $T_n(x)$ in $[-1,1]$, n+1 extrema

↳ roots & extreme points useful for interpolation ⇒ Chebyshev points



faster oscillations near edges

$T_n(x)$ equioscillates on $[-1,1]$, i.e. the oscillations vary based on the magnitude between the points.

↳ Chebyshev points clustered near the endpoints

What's the leading coefficient of $T_n(x)$?

$T_0(x) = 1$, $T_1(x) = x$

$T_2(x) = 2x(x) - 1 = 2x^2 - 1$

$T_3(x) = 2x(2x^2-1) - x = 4x^3 - 3x$

$T_4(x) = 2x(4x^3-3x) - (2x^2-1) = 8x^4 \ldots$

$\Rightarrow$ leading coeff of T_n is 2^{n-1}.

Monic polys have leading coeff 1, so monic Chebyshev are:

$\tilde{T}_0(x) = 1$, $\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x)$ $n = 1, 2, \ldots$

↳ have same roots & locations of extrema, but w/ max/min values of $(-1)^j / 2^{n-1}$

$\Rightarrow$ smaller & shrinking b/c of our rescaling.

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

$\forall$ degree n monic polys. $\hat{P}_n(x)$.

$\Rightarrow$ Chebyshev polys have the smallest extreme values of all monic polys $\Rightarrow$ smallest oscillations

Lec 23 cont'd — 11/20

Thm Let $\tilde{\Pi}_n$ be the space of monic, deg. n polys. then

$$\min_{\tilde{p} \in \tilde{\Pi}_n} \left(\max_{x \in [-1,1]} |\tilde{P}(x)| \right) = \frac{1}{2^{n-1}}$$

Pf: (By contradiction). Suppose $P_n(x) \in \tilde{\Pi}_n$ and

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

Let $Q(x) = \tilde{T}_n(x) - P_n(x)$, note Q is at most of deg. $n-1$ (since $\tilde{T}_n$, P_n monic)

Let x_j be the j^{th} extreme-point of $T_n(x)$ ($n+1$ such x_j)

$$Q(x_j) = \tilde{T}_n(x_j) - P_n(x_j) = \frac{(-1)^j}{2^{n-1}} - P_n(x_j)$$

Then $Q(x_j) \geq 0$ for j even

& $Q(x_j) \leq 0$ for j odd

$\Rightarrow$ Q must have roots in each interval (x_j, x_{j+1})

for $j = 0, \ldots, n-1$ $\Rightarrow$ n roots

but degree$(Q) \leq n-1$ $\Rightarrow$ $Q = 0$!

Hence $P_n(x) = \tilde{T}_n(x)$ □

Lec 24 - MAT226A - 11/22

Polynomial Interpolation: Given 2 points $(x_0, f(x_0)), (x_1, f(x_1))$, if $x_0 \neq x_1$, $\exists$ unique linear fn.

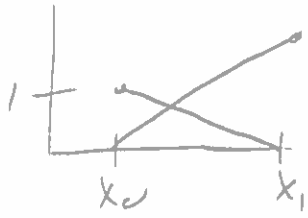
Newton form: $p(x) = f(x_0) + \frac{f(x_1)-f(x_0)}{x_1-x_0}(x-x_0)$

This is a rearrangement of point-slope form, dubbed Newton form

(could also write in Lagrange form

Lagrange Form: $p(x) = \left(\frac{x-x_1}{x_0-x_1}\right) f(x_0) + \left(\frac{x-x_0}{x_1-x_0}\right) f(x)$

which is a "weighted average" or sum of these 2 fns:



Given $x_0, \ldots, x_n$ distinct data points & values $f(x_0), \ldots, f(x_n)$

$\exists$ unique poly. of deg $\leq n$, $p(x)$, s.t. $p(x_j) = f(x_j)$ $\forall j$.

How to find $p(x)$?

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \leftarrow \text{solve for } \vec{c}$$

Don't just solve for $\vec{c}$ generically, instead

we find the determinant $\prod_{0 \leq i \leq j \leq n} (x_i - x_j)$

Lec 24 cont'd – 11/22

For $n+1$ points,

$$p(x) = \sum_{k=0}^{n} l_k(x) f_k, \text{ where } l_k(x_j) = \delta_{jk}$$

$$\& \; l_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \left[\frac{(x - x_j)}{(x_k - x_j)} \right]$$

Note $l_k(x)$ are n^{th} deg polys.

This gives a uniq-e poly!

Pf: Assume $\exists\, q$ s.t. $q(x_j) = f(x_j)$ for $j = 0, \dots, n$; $\deg(q) \leq n$

Have $R(x) = p(x) - q(x)$ of $\deg(R) \leq n$

Then $R(x_j) = 0 \;\forall j \Rightarrow n+1$ roots $\not\downarrow$ $\deg(R) \leq n$

unless $R(x) = 0 \Rightarrow q(x) = p(x)$. □

What about lower deg polys for higher # of data points?

Suppose $f \in C^{n+1}[a,b]$, $x_0, x_1, \dots, x_n \in [a,b]$

For each $x \in [a,b]$, $\exists\, \xi \in (a,b)$ s.t.

$$f(x) = p(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0) \cdots (x - x_n)$$

Remainder – don't know how big
so not as accurate

Lagrange form: $p(x) = \sum_{k=0}^{n} l_k(x) f_k$ ← numerically too much work

Newton form:

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \dots + c_n \prod_{j=0}^{n-1} (x - x_j)$$

Lec 24 cont'd

Fastest ways to find c's in Newton Form:

Divided difference (chooses a basis that triangularizes the matrix)

$$c_0 = f[x_0]$$

$$c_1 = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$c_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

Other polyforms:

Barycentric form: $l_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n-1} \frac{(x - x_j)}{(x_k - x_j)}$

$$= \left(\underbrace{\prod_{j=0}^{n} (x - x_j)}_{\Phi(x)} \right) \left(\frac{w_k}{x - x_k} \right)$$

where $w_k = \prod_{\substack{j=0 \\ j \neq k}}^{n-1} \frac{1}{(x_k - x_j)}$.

Then $l_k(x) = \Phi(x) \frac{w_k}{x - x_k} \Rightarrow p(x) = \begin{cases} \Phi(x) \sum_{k=0}^{n} \frac{w_k}{x - x_k} f_k, & x \neq x_k \\ f_k, & x = x_k \end{cases}$

Note: w_k only depends on point distribution, can precompute!

Note: If $f_k = 1 \; \forall k$, then $p(x) = 1$ (do the algebra)

Lec 25 - MAT226A - 11/27

Global Polynomial Interpolation

★ Performance of polynomial interpolation depends on point spacing

Compare interpolation to given fn. using equidistant points

$$x_k = -1 + \frac{2k}{n}, \quad k = 0, 1, \ldots, n$$

or Chebyshev points → generally better for high deg. polys

$$x_k = \cos\left(\frac{k\pi}{n}\right), \quad k = n, n-1, \ldots, 0$$

Four examples on handouts:

1) $y = |x|$ C^0, not C^1
2) $y = |x|^3$ C^2
3) $y = e^{-x}$ C^∞
4) $y = \frac{1}{25x^2+1}$ C^∞ → has poles in complex plane, so convergence is slow for Ch. & non-existent for equidist.

See Bernstein loops & contour integrals

Thm: For smooth functions, interpolation error of n^{th} deg. approx is:

$$\underbrace{\frac{f^{(n+1)}(\xi)}{(n+1)!}}_{\text{depends on fn.}} \underbrace{\prod_{k=0}^{n} (x - x_k)}_{n+1 \text{ deg monic polynomial}}$$

★ If the x_k's are the zeros of Chebyshev polys, this term is minimized

Conditioning of the interpolation problem

Sensitivity of the interpolant to the data

Consider the x_k's fixed (interpolant f-n) & wiggle the data:

$$p(x) = \sum_{k=0}^{n} f_k \ell_k(x)$$

$$\|p(x)\|_\infty = \max_{x\in[-1,1]} \left| \sum_{k=0}^{n} f_k \ell_k(x) \right|$$

$$\leq \max_{x\in[-1,1]} \sum_{k=0}^{n} |f_k| |\ell_k(x)|$$

$$\leq \underbrace{\|f\|_\infty}_{\text{depends on data}} \cdot \underbrace{\max_{x\in[-1,1]} \sum_{k=0}^{n} |\ell_k(x)|}_{\text{depends on point distribution}}$$

$\hookrightarrow (\tilde{f})_k = f_k$

Define Lebesgue constant $\Lambda = \max_{x\in[-1,1]} \sum_{k=0}^{n} |\ell_k(x)|$

Can show $\Lambda = \sup_f \frac{\|p\|_\infty}{\|f\|_\infty}$ ← ∞-norm of interpolation operator that sends conts fn. → poly. fn.

Λ depends on the points x_k → # & distribution.

Can also show that Λ is the condition # of the interpolation problem

Λ_n = Lebesgue constant for n^{th} deg. poly interpolant

Equidistant points: $\Lambda_n \sim \frac{2^{n+1}}{e \cdot n \ln(n)}$ ← awful, exponential growth

Chebyshev points: $\Lambda_n \sim \frac{2}{\pi} \ln(n)$ ← grows, but small for relevant n.

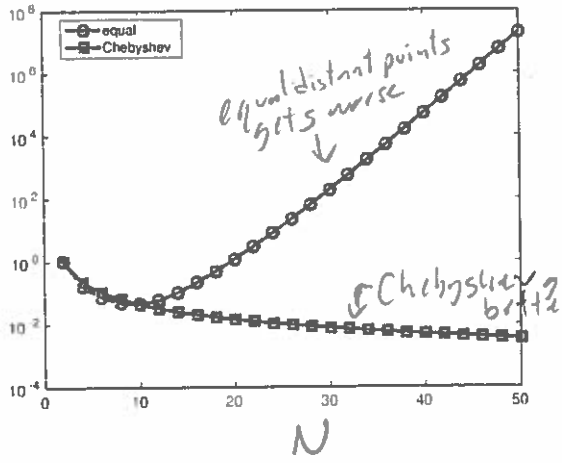
lec25 – 11/27

N = # points

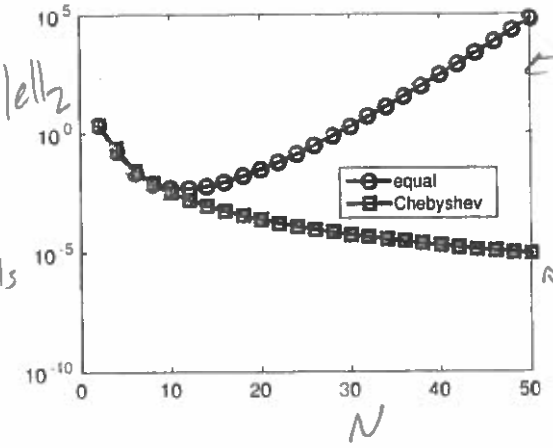
$$\|e\|_2 = \left(\sum_{n=1}^{2000} |y(x_n) - \tilde{y}(x_n)|^2 \right)^{1/2} \approx \|y - \tilde{y}\|_2$$

↑ evaluated on 2000 points numerically

1) $y = |x|$



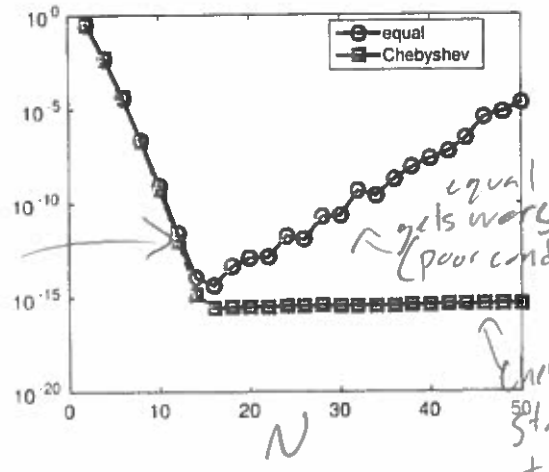
2) $y = |x|^3$



← equidistant does worse exply

↖ Chebyshev converges faster than in (1)

3) $y = \exp(-x)$

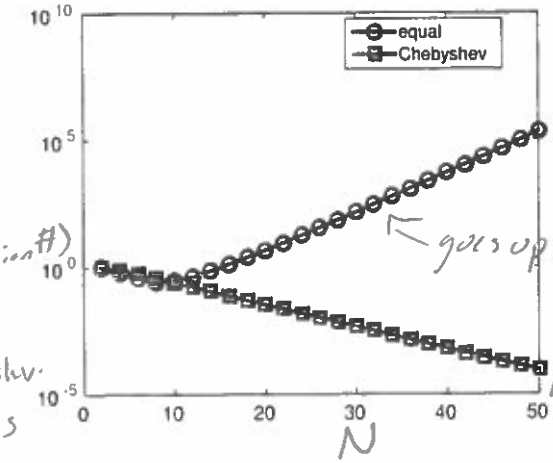


both converge to machine ε

↖ equal gets worse (poor condition #)

Chebyshev stays at machine ε

4) $y = \dfrac{1}{25x^2 + 1}$



↖ goes up from start

↖ converges

Here, both converge to ε_{mach} precision. After ε_{mach}, the equispaced points interpolation begins to grow in error, due to its poor conditioning.

On the other hand, Chebyshev spaced points interpol stays at ε_{mach}, suggesting condition # is small.

- Chebyshev is analog of Fourier series, so smoothness of fn. leads to speed of convergence of approx.

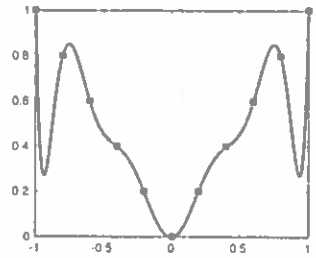
- Not true for equispaced interval

Comparing performance of poly. interpolation using different point spacings

Lec 25
11/27

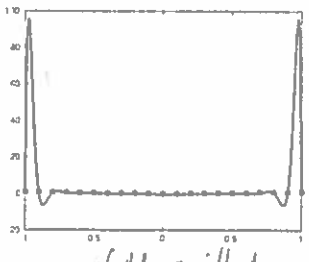
$y = |x|$

$N = 11$ Equal



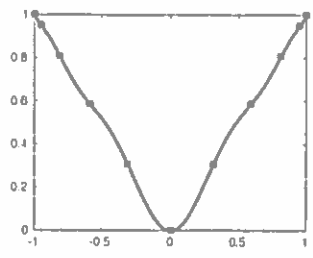
wiggles btwn points - bad!

$N = 21$ Equal



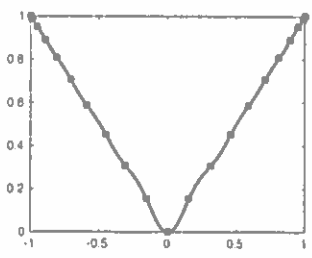
awful! oscillations orders of mag higher near bdry

$N = 11$ Chebyshev



looks okay

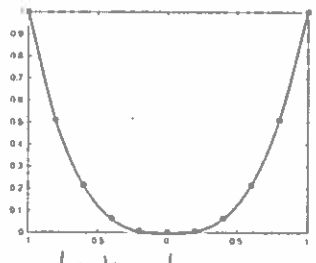
$N = 21$ Chebyshev



respectable

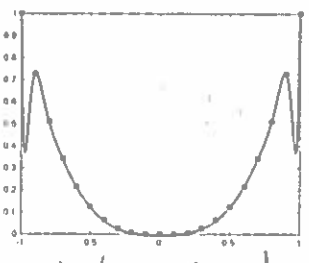
$y = |x|^3$

$N = 11$ Equal



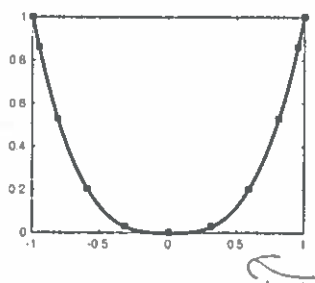
looks okay

$N = 21$ Equal

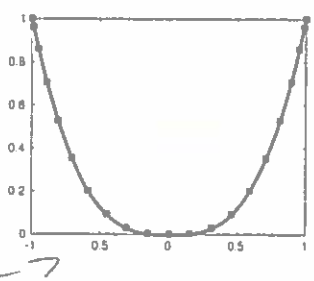


wiggles appear! bad

$N = 11$ Chebyshev



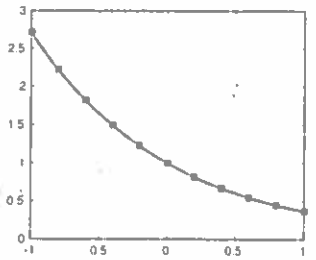
$N = 21$ Chebyshev



both look pretty good to the eye

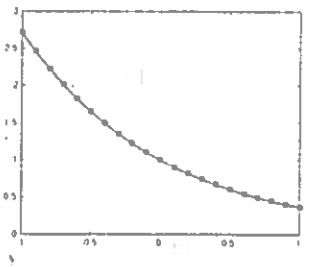
$y = \exp(-x)$

$N = 11$ Equal

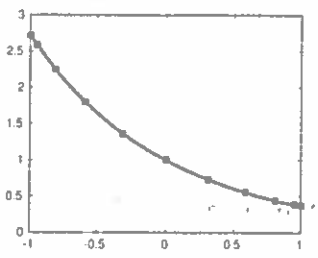


looks great!

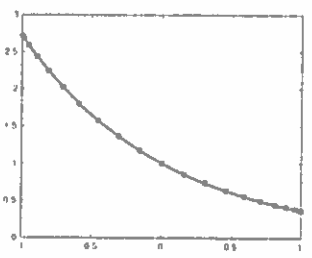
$N = 21$ Equal



$N = 11$ Chebyshev



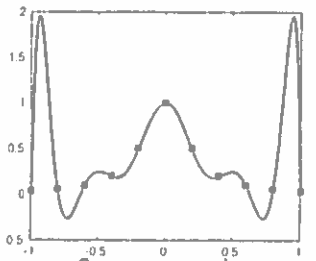
$N = 21$ Chebyshev



$$y = \frac{1}{25x^2 + 1}$$

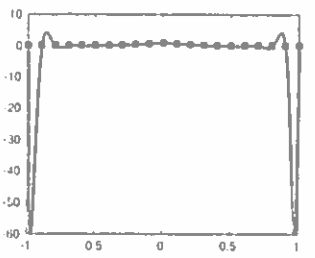
→ has poles in complex plane

$N = 11$ Equal



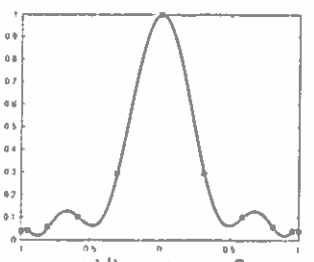
awful, wild oscillations

$N = 21$ Equal



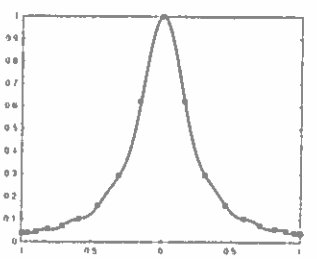
even larger errors near boundaries

$N = 11$ Chebyshev



still not perfect, wiggles exist

$N = 21$ Chebyshev



looks great!

Lec 26 - 11/29 - MAT 226A

Piecewise Polynomial Approximation

Previously, used single polynomial to approx fn/data on the entire domain

Piecewise linear interpolant

spline of order 2 →

← connect the dots spline

Generically, given $a = x_0 < x_1 < \dots < x_n = b$ locations
& data $y_0, y_1, \dots, y_n$
↳ underlying function $y_j = f(x_j)$ potentially.

$S(x) = S_j(x)$ on $x \in [x_j, x_{j+1}]$

$S_j(x) = a_j + b_j(x - x_j)$

Require S to interpolate data: $S(x_j) = y_j$
& S is cont's

We have n intervals. $2 \cdot n$ coeffficients to determine S
Interpolate data -> $n+1$ conditions to satisfy -> $2n$ conditions
Continuity at interior nodes -> $n-1$ conditions
Enough information + constraints to uniquely determine S

Linear spline (order 2) => $S_j(x) = y_j + \left(\frac{y_{j+1} - y_j}{x_{j+1} - x_j}\right)(x - x_j)$

Example of spline: Given distinct $a = x_0 < x_1 < \dots < x_n = b$,
a spline $S(x)$ of order $k \geq 1$ is
a) poly. of deg k on each interval $[x_j, x_{j+1}]$
b) for $k=1$, S is PW constant, for $k \geq 2$ S & its $k-2$ derivs are cont's.

linear is $k=2$ cubic is $k=4$

☆ PW linear spline minimizes length

History: a spline is a drafting tool - a thin, flexible wood/metal strip.

Pin some points called knots, place wood & then trace.

From elasticity, the curve will minimize strain energy

$$\mathcal{E} = \int_a^b (K(x))^2 \, dx$$

↑ curvature

If y' is small, $K \approx y'' \Rightarrow \mathcal{E} = \int_a^b (K(x))^2 dx \approx \int_a^b (y''(x))^2 dx$

Find $y(x)$ to minimize this functional
s.t. the interpolation constraints
↳ gives the natural interpolating cubic spline.

Cubic Spline Interpolants (order 4)

Given $a = x_0 < x_1 < \ldots < x_n = b$ & data $y_j = f(x_j)$.
$S(x)$ is a cubic poly. on each $[x_j, x_{j+1}]$

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Require S interpolates the data $S(x_j) = y_j$ → $n+1$ constraints
& $S \in C^2$ ⇒ S, S', S'' cont's at interior points → $3(n-1)$ constraints

$S_j(x_{j+1}) = S_{j+1}(x_{j+1})$, $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$, $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$

⇒ $4n-2$ constraints

Have n intervals & 4 coeffs on each → need $4n$ constraints → need 2 more constr.

Need boundary conditions! Natural → $S''(a) = S''(b) = 0$ (energy min.)
clamped (complete spline): $S'(a) = f'(a)$, $S'(b) = f'(b)$

Lec 26 cont'd 11/28

Cubic spline interpolants

$S_j(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3$

Require $S(x_j) = y_j$, $S_j(x_{j+1}) = S_{j+1}(x_{j+1})$, $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$
& $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$

Need 2 more constraints - choices:

Natural BC's: $S''(a) = S''(b) = 0$

Clamped/complete: $S'(a) = f'(a)$, $S'(b) = f'(b)$

Periodic BC's: $S'(a) = S'(b)$, $S''(a) = S''(b)$

Not-a-knot cond'ns: $S_0'''(x_1) = S_1'''(x_1)$ → S_0 & S_1 same
$S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1})$ → S_{n-2} & S_{n-1} same

---

Lec 27 — 12/1

Accuracy for clamped (complete) interpolating cubic splines.

Let $f \in C^4[a,b]$, let S be the unique clamped cubic spline interpolant to f through the nodes $a = x_0 < x_1 < \dots < x_n = b$, then

$$|f(x) - S(x)| \leq \frac{5}{384} \max_{x\in[a,b]} |f^{(4)}(x)| \left(\max_j (x_{j+1}-x_j)\right)^4$$

If $x_{j+1} - x_j = h \ \forall j$, then $\|f(x) - S(x)\|_\infty \leq Ch^4$

↳ not as sensitive to point spacing compared w/ poly interpolation

guaranteed to converge as we add more data points & $h \to 0$.

Lec 27 cont'd

System of eqns. to find cubic interpolating spline

$$S_j(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3$$

$$S_j'(x) = b_j + 2c_j(x-x_j) + 3d_j(x-x_j)^2$$

$$S_j''(x) = 2c_j + 6d_j(x-x_j)$$

} eqns

constraints: interpolation: $S_j(x_j) = y_j = a_j$ ★ for $j=0,\dots,n$

continuity of S: $S_j(x_{j+1}) = S_{j+1}(x_{j+1}) = a_{j+1}$

$$\Rightarrow S_j(x_{j+1}) = a_j + b_j\underbrace{(x_{j+1}-x_j)}_{h_j} + c_j h_j^2 + d_j h_j^3 = a_{j+1} \quad (1)$$

continuity of S': $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1}) = b_{j+1}$

$$\Rightarrow S_j'(x_{j+1}) = b_j + 2c_j h_j + 3d_j h_j^2 = b_{j+1} \quad (2)$$

continuity of S'': $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1}) = 2c_{j+1}$

$$\Rightarrow S_j''(x_{j+1}) = 2c_j + 6d_j h_j = 2c_{j+1} \quad (3)$$

4n x 4n system → get down to n x n

Solve (3): $d_j = \frac{c_{j+1}-c_j}{3h_j}$ ★ → plug into (1) & (2) to eliminate d_j

$$(1'): \quad a_j + b_j h_j + c_j h_j^2 + \tfrac{1}{3}(c_{j+1}-c_j)h_j^2 = a_{j+1}$$

$$(2'): \quad b_j + 2c_j h_j + (c_{j+1}-c_j)h_j = b_{j+1}$$

Use (1') to solve for b_j in terms of c's

Lec 27 cont'd 12/1

Solve for b_j from (1'):

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$$

Plug into (2'):

System: $$h_{j-1}c_{j-1} + 2(h_{j-1}+h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

$j = 1, \dots, n-1$ → need c_0 & c_n ~~define this~~

↳ need BC's

Natural BC's: $S''(a) = S''(b) = 0$

$S''(a) = S_0''(x_0) = c_0 \implies c_0 = 0$

$S''(b) = S_{n-1}''(x_n) = 2c_{n-1} + 6d_{n-1}h_{n-1} = 0$

Recall: $d_j = \frac{c_{j+1} - c_j}{3h_j}$ isn't valid for $j = n-1$ since no c_n

⇒ use BC for $d_{n-1} = \frac{-c_{n-1}}{3h_{n-1}}$ ⇒ Define $c_n = 0$, these expressions same for $j = n-1$.

⇒ We can define $c_n = 0$ & $d_j = \frac{c_{j+1} - c_j}{3h_j}$, $j = 0, \dots, n-1$

Structure of linear system for c's

$$A\underline{c} = \underline{r}$$

$$A = \begin{pmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & \end{pmatrix}$$

$$\underline{r} = \begin{pmatrix} \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \end{pmatrix}$$

note A is strictly diagonally dominant ⇒ invertible

↳ Gaussian elim. w/o pivoting

Lec 27 cont'd

Given f + data + given f' at all nodes

↳ can use Piecewise Hermite poly. interpolant

↳ only C^1, but 4^{th} order accurate & way easier to compute than cubic spline → no linear system solve.

$f(a)$ $f(b)$
$f'(a)$ $f'(b)$

→ 4 constraints can pin 4 coeffs to determine a cubic Hermite polynomial

Cubic Hermite poly is easy to evaluate w/ this data.

Poly. int. error

Approx $f(x)$ w/ poly. interpolation

poly interpolant of deg n, $n+1$ points

$p_n(x_j) = f(x_j)$ for $j = 0, \dots, n$.

Thm: $|f(x) - p_n(x)| = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \cdot \underbrace{\prod_{j=0}^{n} (x - x_j)}_{w(x)}$

For $x = x_j$, $j = 0, \dots, n$, $|f(x) - p_n(x)| = 0$.

~~F(x)~~ $F(t) = f(t) - p_n(t) - \frac{[f(x) - p_n(x)]}{w(x)} w(t)$

for $x \neq x_j$.

For $t = x_j$, $f(x_j) = p_n(x_j) = w(x_j) = 0 \Rightarrow F(x) = 0$

& for $t = x$, $F(x) = 0$

$\Rightarrow$ $n+2$ zeros in $[a, b]$

By MVT, $\exists$ 0 of F' between zeros of F.

$\Rightarrow$ F' has $n+1$ zeros

repeat until $F^{(n+1)}$ has 1 zero, ξ_x

Then $0 = F^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - 0 - \frac{[f(x) - p_n(x)]}{w(x)} (n+1)!$

$\Rightarrow f(x) - p_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} w(x)$ //

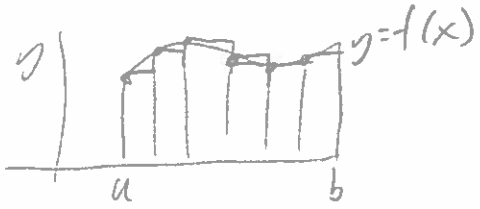
$(A^T A)^T = A^T A$

Lec 28 - MAT226A - 12/4

- **Numerical Integration** → given function $f(x)$ (not just data like interpolation)

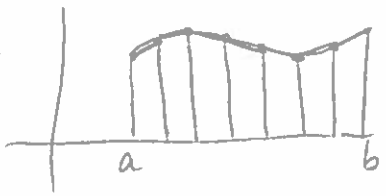
$\int_a^b f(x)dx \leftarrow$ approximate numerically

Riemann



could use Riemann sum, or do better w/ trapezoids instead of rectangles

Trapezoid



← much more accurate!
essentially integrated a linear spline

- think trapezoid rule as approximating f w/ a Piecewise linear function which interpolates f at the endpoints + knots. Then integrating the interpolant
  => Allows us to extend to higher deg approximations + estimate error

$$\int_a^b f(x)dx = \sum_k \underbrace{f(x_k)w_k}_{\text{numerical quadrature}} + \underbrace{\varepsilon}_{\text{error}}$$

↳ we will use PW polynomial, not necessarily a spline
↳ don't worry about matching derivatives at knots

We will cover:

1) Newton-Cotes quadrature - use equidistant points,
   ↳ e.g. trapezoidal rule, Simpson's rule, midpoint rule
   - approx fn. w/ interpolating poly on each interval
   - integrate

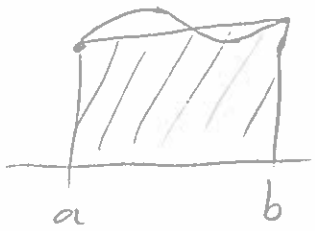
2) Gaussian quadrature - use unequally spaced points, pick quadrature points to minimize the error
   - related to orthogonal polynomials
   - use high deg. poly

3) Low order Newton-Cotes & estimate errors → refine intervals w/ more points as needed (Adaptive quadrature)

Newton-Cotes Quadratures

Trapezoidal Rule

$$\int_a^b f(x)\,dx = \frac{(b-a)}{2}\left(f(a)+f(b)\right) + \xi$$



Composite trapezoidal rule: divide $[a,b]$ into equal-length subintervals & do trapezoidal rule on each subinterval

$\rightsquigarrow x_j = a + jh$ for $j = 0, \dots, n$

w/ $h = \frac{b-a}{n}$

$$\Rightarrow \int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(x_0)+f(x_1)\right) + \frac{h}{2}\left(f(x_1)+f(x_2)\right) + \dots + \frac{h}{2}\left(f(x_{n-1})+f(x_n)\right)$$

$$= h\left(\frac{f(x_0)}{2} + \sum_{j=1}^{n-1} f(x_j) + \frac{f(x_n)}{2}\right)$$

Error analysis → look at one interval: $\int_a^b f(x)\,dx$.

Approx f w/ linear fn. interpolating f at the endpoints

interpolate: $f(x) = \underbrace{\frac{(x-b)}{a-b} f(a) + \frac{(x-a)}{b-a} f(b)}_{\text{Lagrange interpolant}} + \underbrace{\frac{f''(\xi(x))}{2}(x-a)(x-b)}_{\text{error term}}$

integrate:

$$\int_a^b f(x)\,dx = f(x_0)w_0 + f(x_1)w_1 + \xi$$

$x_0 = a,\ x_1 = b$

$$w_0 = \int_a^b \frac{(x-b)}{(a-b)}\,dx = \frac{b-a}{2} \qquad w_1 = \int_a^b \frac{x-a}{b-a}\,dx = \frac{b-a}{2}$$

$$\xi = \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)\,dx = \int_a^b f[a,b,x](x-a)(x-b)\,dx$$

$f[a,b,x]$: divided difference

↳ easier to integrate

Lec 28 cont'd – 12/4

For higher deg poly approx in Newton-Cotes quadrature:

$$f(x) = P_n(x) + R(x)$$

w/ $P_n(x) = \sum_{k=0}^{n} L_k(x) f(x_k)$ w/ $L_k(x) = \dfrac{\prod_{j\neq k}(x - x_j)}{\prod_{j\neq k}(x_k - x_j)}$

& $R(x) = \prod_{k} \dfrac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_k)$

$$\Rightarrow \int_a^b f(x)\,dx = \sum_{k=0}^{n} f(x_k) w_k + \underbrace{\varepsilon}_{\leq \int_a^b R(x)\,dx}$$

w/ $w_k = \int_a^b L_k(x)\,dx$ ← only depends on points, not function!

Closed Newton-Cotes formulas – use n subintervals on $[a,b]$ including endpoints

$\begin{cases} x_j = a + jh \\ h = \frac{b-a}{n} \end{cases}$

to evaluate function & interpolate

$n=1 \Rightarrow$ Trapezoidal rule

$$\int_a^b f(x)\,dx = \frac{h}{2}(f(a) + f(b))$$

$h = \frac{b-a}{2}$

$n=2 \Rightarrow$ Simpson's rule

$$\int_a^b f(x)\,dx = \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)) = \frac{b-a}{6}(f(x_0) + 4f(x_1) + f(x_2))$$

Open Newton-Cotes : exclude endpoints from interpolant

$n=0$: Midpoint rule

$$\int_a^b f(x)\,dx = (b-a) f\left(\frac{a+b}{2}\right)$$

$n=1$: next one uses 2 interior points & a line through them

Standard & Adaptive Simpson's Rule -> MATLAB's quad command

Accuracy

Degree of precision of a quadrature formula is the largest integer m s.t. the quadrature is exact for all polynomials of degree $\leq m$ but not for some poly. of deg. $m+1$.

- Expect trapezoidal rule to have deg. of precision 1 (it does)
- Expect Simpson's rule to have deg. of precision 2, but in fact, Simpson's rule has degree of precision 3!

Will see that composite trapezoidal rule is $\mathcal{O}(n^2)$ where n = # points
& Simpson's rule is $\mathcal{O}(n^4)$!

$\Rightarrow$ Trap. rule can integrate linear polys exactly, but not quadratic

$\Rightarrow$ Simpson's rule can integrate cubic polys. exactly, but not quartic

Lec 29 - MAT 226A - 12/6

Error Analysis of Numerical Quadrature

-> Use weighted Mean Value Thm for integrals

Thm: Suppose $f \in C[a,b]$ and g is integrable on $[a,b]$ & doesn't change sign
There exists $\xi \in (a,b)$ s.t. $\int_a^b f(x)g(x)dx = f(\xi)\int_a^b g(x)dx$

Trapezoidal Rule: $\int_a^b f(x)dx = \frac{b-a}{2}(f(a)+f(b)) + \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)dx$

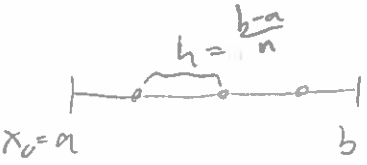
($(x-a)(x-b)$: doesn't change sign on $[a,b]$)

(integral term: error term from accuracy of interpolating polynomial)

Use weighted MVT on error term, $\exists \xi \in (a,b)$ s.t.

$$\xi = \frac{f''(\xi)}{2}\int_a^b \underbrace{(x-a)(x-b)}_{g(x)}dx = -\frac{f''(\xi)}{12}(b-a)^3 = -\frac{f''(\xi)}{12}h^3$$

Composite Trapezoidal Rule: $h = \frac{b-a}{n}$



(extend to n subintervals, $\xi_k \in [x_{k-1}, x_k]$)

$$\int_a^b f(x)dx = h\left(\frac{1}{2}f(a) + \sum_{k=1}^{n-1} f(x_k) + \frac{1}{2}f(b)\right) - \frac{h^3}{12}\sum_{k=0}^{n-1} f''(\xi_k)$$

Note: $\min_{x\in[a,b]} f''(x) \le \underbrace{\frac{1}{n}\sum_{k=0}^{n-1} f''(\xi_k)}_{\text{average}} \le \max_{x\in[a,b]} f''(x)$

Since f'' continuous, by intermediate value thm, $\exists \mu \in [a,b]$ s.t.

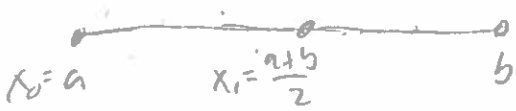
$$\frac{1}{n}\sum_{k=0}^{n-1} f''(\xi_k) = f''(\mu)$$

$$\Rightarrow \xi = -\frac{h^3}{12}\sum_{k=0}^{n-1} f''(\xi_k) = \frac{(b-a)}{12}\frac{h^2}{n}\sum_{k=0}^{n-1} f''(\xi_k) = \frac{b-a}{12}h^2 f''(\mu)$$

$\Rightarrow \xi = \mathcal{O}(h^2)$ so (comp. trap. rule is second-order accurate.

(if we double the points, get error reduction factor 4)

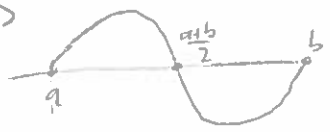
Simpson's Rule:



$$\int_a^b f(x)\,dx = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) + \int_a^b \frac{f'''(\xi(x))}{6}\cdot(x-a)(x-x_1)(x-b)$$

changes sign

$(x-a)(x-x_1)(x-b) \rightarrow$



apply MVT naively on each half-interval
get f''' with opposite signs
→ looks like 4th derivative

Idea: Integrate error term by parts to get 4th deriv & $g(x)$ poly that doesn't change sign

After: $\xi = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\mu) = -\frac{(b-a)^5}{2880} f^{(4)}(\mu)$

→ cubic polys have $f^{(4)} = 0 \Rightarrow \xi = 0$
extra degree of precision from symmetry

Composite Simpson's rule: same analysis as comp. trap rule.
add up n error terms to drop a power of h out of the error term

$$\xi = -\frac{(b-a)}{180} h^4 f^{(4)}(\mu)$$, where h is the point spacing

divide into intervals, apply Simpson's rule to triples of points

$\xi = O(h^4)$

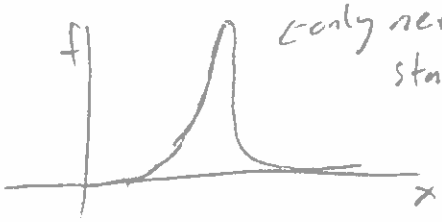
$\Rightarrow$ Comp. Simpson's rule is 4th order accurate

So if we double the points $\Rightarrow$ error goes down by factor of 16.

Lec 29 cont'd - 12/6

Adaptive Quadrature

Adaptive Simpson's: Idea: estimate the error using two diff. approx & use more points locally as needed

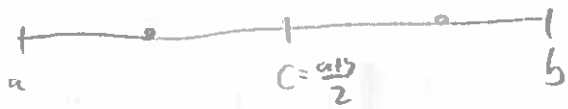
Ex: 

only need points where $f(x)$ changes sharply/bends
standard comp. Simpson's wastes effort!

$$S(a,b) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right), \quad \xi_1 = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$$

$$\int_a^b f(x)\,dx = S(a,b) + \xi_1$$

← apply 2 interval Simpson's rule



$$\int_a^b f(x)\,dx = S(a,c) + S(c,b) + \xi_2$$

where $\xi_2 = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 \frac{2}{2^5} f^{(4)}(\xi_2)$

Assume $f^{(4)}(\xi) = f^{(4)}(\xi_2)$, then $\xi_2 = \frac{1}{16}\xi_1$

$$\Rightarrow \int_a^b f(x)\,dx = S(a,b) + 16\xi_2$$

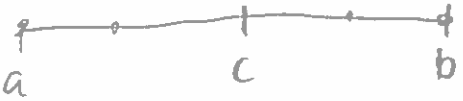
$$\int_a^b f(x)\,dx = S(a,c) + S(c,b) + \xi_2$$

↖ Subtract these equations

$$S(a,c) + S(c,b) - S(a,b) - 15\xi_2 = 0$$

$$\Rightarrow \xi_2 = \frac{S(a,c) + S(c,b) - S(a,b)}{15}$$

Idea:

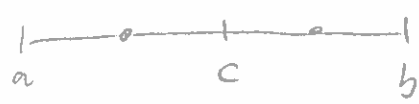


← see if error ξ_2 is small enough

← do same check by applying 2 simpson's to each interval, refine until error in each part small enough

Lec 30 - MAT226A - 12/8

Adaptive Simpson's rule



$\rightsquigarrow \int_a^b f(x)dx$

$Q_1 = S(a,b)$

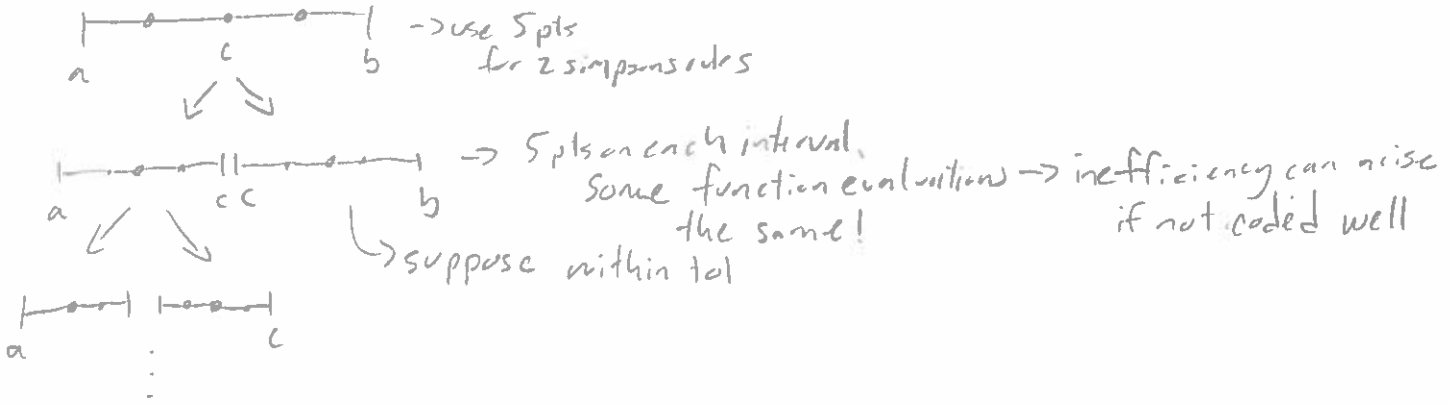
do 2 Simpson's rule approxs $Q_2 = S(a,c) + S(c,b)$

& obtained $E_2 = \frac{Q_2 - Q_1}{15}$

If $|E_2| < \epsilon$, $Q = Q_2 + \frac{1}{15}(Q_2 - Q_1)$

o/w, repeat on $[a,c]$ & $[c,b]$ separately

$$\int_a^b f(x)dx = \underbrace{\int_a^c f(x)dx}_{\text{apply same procedure w/ tol } \epsilon/2} + \underbrace{\int_c^b f(x)dx}_{\text{apply same procedure w/ tol } \epsilon/2}$$



Gaussian Quadrature - unlike Newton-Cotes, which uses equally spaced points, GQ uses different spacing points.

- Recall for high-order polynomial interpolation, using equidistant points led to wiggles/error in interpolant! Use others, e.g. Chebyshev points

- Choose point locations to get the greatest degree of precision

Ex: 2 points: $\int_a^b f(x)dx \approx f(x_1)w_1 + f(x_2)w_2$ ← 4 degrees of freedom → can get deg 3 precision

Require $\int_a^b 1\,dx = w_1 + w_2$, $\int_a^b x\,dx = x_1 w_1 + x_2 w_2$, $\int_a^b x^2 dx = x_1^2 w_1 + x_2^2 w_2$

$\int_a^b x^3 dx = x_1^3 w_1 + x_2^3 w_2$

Lec 30 cont'd

Solve system:
$$\int_a^b 1\,dx = w_1 + w_2$$
$$\int_a^b x\,dx = x_1 w_1 + x_2 w_2$$
$$\int_a^b x^2\,dx = x_1^2 w_1 + x_2^2 w_2$$
$$\int_a^b x^3\,dx = x_1^3 w_1 + x_2^3 w_2$$

} 2 point Gaussian quadrature

On $[-1,1]$, we get $w_1 = w_2 = 1$, $x_{1,2} = \pm 1/\sqrt{3}$

other GQ's tabulated for $[-1,1]$ → look them up!

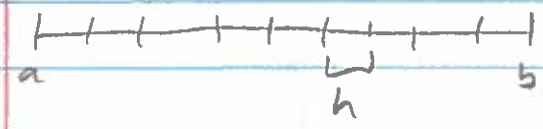
On $[a,b]$, use transformation

$$x = \left(\frac{b-a}{2}\right) z + \left(\frac{b+a}{2}\right)$$

to transform $[-1,1]$ to $[a,b]$

$$\int_a^b f(x)\,dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\left(\frac{b-a}{2}\right) z + \left(\frac{b+a}{2}\right)\right) dz$$

2 pt. GQ error: $E_2 = \frac{(b-a)^5}{4320} f^{(4)}(\xi)$ → looks like a factor of 2 more accurate than Simpson's (who cares)



→ use composite version by dividing $[a,b]$ into n subintervals of width $h = \frac{b-a}{n}$.

Obtain 2 pt. comp. GQ error

$$E = \frac{(b-a)}{4320} h^4 f^{(4)}(\xi)$$ ← factor 2 better than comp. Simpson's rule

The diff. in accuracy btwn GQ & Newton-Cotes is more striking as we increase the # of points

↳ 6Q standard is ~7 pts ⇒ How do we get the points? Table, duh but where do they come from?

The points $\{x_k\}$ for n-point Gaussian quadrature $\sum_{k=1}^{n} f(x_k) w_k$ on $[-1,1]$ are the n roots of the n^{th} degree Legendre polynomial

The weights $w_k = \int_{-1}^{1} \prod_{j \neq k} \frac{(x - x_j)}{(x_k - x_j)} dx$

The n-point Gaussian quadrature has degree of precision $2n-1$.

On the interval $[a,b]$, $\xi_n = \frac{(b-a)^{2n+1} (n!)^4 f^{(2n)}(\xi)}{(2n+1)[(2n)!]^3}$ → lots of derivatives

For composite n-pt GQ w/ interval length h, $\varepsilon = O(h^{2n})$

GQ is awesome! Needs smooth f since it requires so many derivatives
- so not perfect for non-smooth f → non-smooth f better w/ Simpson's
- vectorizes easier than adaptive Simpson's.

Why is n-pt. GQ deg. precision n?

pf: Let $p \in \Pi_{2n-1}$, then $p(x) = q(x) \phi_n(x) + r(x)$
(ϕ_n: n^{th} deg Legendre poly)

& q, r are $n-1$ deg. polys

$\Rightarrow \int p(x)dx = \int \underbrace{q(x)\phi_n(x)}_{0 \text{ from orthogonality}} + r(x)dx = \int r(x)dx$

$\Rightarrow \sum_{k=1}^{n} p(x_k) w_k = \sum_{k=1}^{n} \left(q(x_k) \cancel{\phi_n(x_k)}^{0} + r(x_k) \right) w_k = \sum_{k=1}^{n} r(x_k) w_k$

$= \int r(x)dx = \int p(x)dx.$

Final Exam

Math 226A

Fall 2017

1. Suppose you run the following MATLAB commands to form three related random matrices and random vector.

```
N=4000;         % size of the problem
r = rand(N,1); % random Nx1 vector
A = rand(N);    % form a random NxN square matrix
B = A'*A;
C = A + A';
```

You then use the built-in linear solver to solve the three systems $Ax = r$, $Bx = r$, and $Cx = r$, and you record the time to solve these systems. The results are

```
>> tic; x=A\r; toc
Elapsed time is 1.051728 seconds.

>> tic; x=B\r; toc
Elapsed time is 0.498099 seconds.

>> tic; x=C\r; toc
Elapsed time is 0.989374 seconds.
```

Comment on the results, and explain why one system is solved faster than the others.

2. The 2-point open Newton-Coats approximation to the integral

$$\int_0^1 f(x)\,dx \approx f(x_0)w_0 + f(x_1)w_1$$

uses the function values at $x_0 = 1/3$ and $x_1 = 2/3$.

(a) Derive formulas for the weights w_0 and w_1.

(b) What is the degree of precision of the 2-point open Newton-Coats approximation to the integral?

3. A square matrix Q is orthogonal if $Q^T = Q^{-1}$. Show that the condition number (in 2-norm) of an orthogonal matrix is 1.

4. In class we examined the three different matrix factorizations: LU, Cholesky, and QR. All of these factorizations can be used to solve the linear system $A\vec{x} = \vec{b}$. For each factorization comment on when it would be advantageous to use one over the other and explain why.

5. Suppose that f is a C^3 function on $\mathbb{R}$, $f(x^*) = 0$, and $f'(x^*) \neq 0$.

   (a) Will Newton's method applied to f converge to x^* for an initial guess sufficiently close to x^*?

   (b) Define $g(x) = (f(x))^2$. Then $g(x^*) = 0$. Discuss the convergence of Newton's method applied to g instead of f. Does it converge at all? If so, is the convergence faster or slower than the convergence for f?

6. In class we observed the error in using a polynomial interpolant to approximate a function sometimes grew exponentially as the number of points increased. Is the growth of error due stability or conditioning? Explain why exponential growth of the error is expected.

7. Suppose A is a full rank m by n matrix with $m > n$. The least squares solution to $A\vec{x} = \vec{b}$ can be found be either solving the normal equations $A^*A\vec{x} = \vec{b}$ or by using the QR factorization. Discuss the advantages and disadvantages of each approach. Specifically, comment on when you might use one method over the other.

8. The IEEE standard for quad precision floating-point numbers uses 128 bits: 1 for the sign bit (s), 15 for the exponent (c), and 112 for the mantissa (or fractional part) (q). Floating point numbers are represented as

$$x = (-1)^s(1+q) \times 2^{c-16383}.$$

   (a) Recall that one definition of ϵ_{mach} is the relative difference between two floating point numbers so that for $x \in \mathbb{R}$ (in an appropriate range)

$$|x - fl(x)| \leq \epsilon_{\text{mach}}|x|,$$

   where $fl(x)$ is the floating point representation of x. What is ϵ_{mach} for quad precision?

   (b) Approximately how many decimal digits of accuracy are achieved by quad precision? Note that $\log_{10}(2) \approx 0.3$. 33 or 34?

   (c) Suppose we use a backward stable algorithm applied to a problem with a condition number of $\kappa = 10^p$ to perform a calculation in quad precision. How many digits of accuracy do you expect in the computed solution?

9. Suppose you approximate $f(x) = \cosh(x)$ on $[-1, 1]$ by constructing a polynomial interpolant using the function values at the $n+1$ points $x_k = \cos(k\pi/n)$ for $k = 0, \ldots, n$. Derive a bound on the maximum error in using the interpolant to approximate the function. Hint: these are the extreme points of the Chebyshev polynomials, and in homework 4, you examined a polynomial with these roots.