1) Floating point arithmetic

Floating point representation

$$x = \pm S \times 2^E$$

$$1 \le S < 2$$

S – binary representation $(b_0 . b_1 b_2 \cdots)_2$, where $b_0 = 1$

precision: # bits in significand (including hidden bit)

$x = \pm (1 . b_1 \cdots b_{p-1})_2 \times 2^E$ has precision p.

IEEE FP standard

Subnormals

- Can represent numbers smaller than $2^{E_{min}} = 2^{-126}$ by a 00000000 - exponent bitstring

Rounding

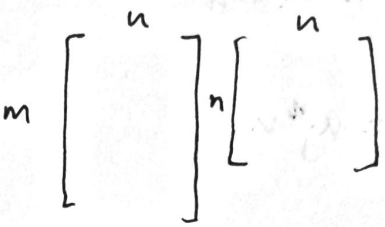
If x is in the normalized range, then

$$\text{round}(x) = x(1+\delta)$$

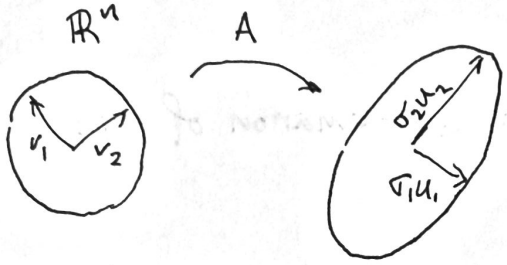
where $|\delta| < \varepsilon = 2^{-(p-1)}$

SVD

- $A: m \times n$ maps $A: \mathbb{R}^n \to \mathbb{R}^m$



- Data applications: often have $m >> n$
- Key idea: Image of unit sphere is a hyperellipse



- ~~choose weights~~ Let $u_1, \dots, u_n$ be the principal axes of the ellipse (weights $\sigma_1, \dots, \sigma_n$). Note dim range $A \leq n$ so this is the most we can have
- select preimages $v_1, \dots, v_n$

$$\rightarrow \quad AV = \hat{U}\hat{\Sigma}$$

Full SVD

- Pad $\hat{U}$ to be a full $m \times m$ matrix
- Then pad $\hat{\Sigma}$ with 0s to make dimensions work.

Eigenvalue decomposition

If A is diagonalizable, $A = V \Lambda V^{-1}$

Matrix properties

- rank $(A) = r$ = number of nonzero singular values $\sigma_1, \dots, \sigma_r$
- $\|A\|_2 = \sigma_1$
- Singular values are square roots of eigenvalues of A^*A.

Low-rank approximation

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^*$$

Theorem:

Define $A_\nu = \sum_{j=1}^{\nu} \sigma_j u_j v_j^*$

Then A_ν is the best rank ν approximation of A:

$$\|A - A_\nu\| = \min_{\text{rank}(B)=\nu} \|A - B\|$$

with $\|A - A_\nu\|_2 = \sigma_{\nu+1}$

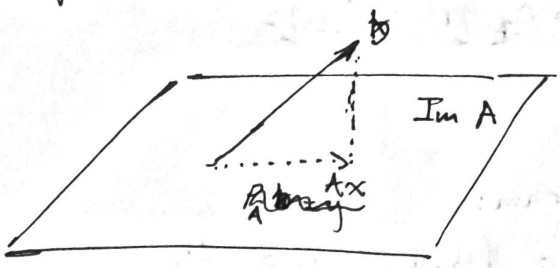
Proof: suppose there is rank ν B such that

$$\|A - B\|_2 < \|A - A_\nu\|_2 = \sigma_{\nu+1}$$

3 QR Factorization

Projectors

- $P_q = qq^*$ projects a vector $\vec{x}$ onto subspace $\vec{q}$ (where $\|\vec{q}\|=1$)
- $P_{\perp a} = I - \frac{aa^*}{a^*a}$
- for $A = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix}$,



- consider picture:
- Note $\min \|Ax-b\|_2 \iff A^TAx = A^Tb$

$$\Rightarrow \quad x = (A^TA)^{-1}A^Tb$$

so that $Ax = P_A b = A(A^TA)^{-1}A^Tb$

QR Factorization

- Given $A \in m \times n$, want $Q^*Q = I$ so that Q spans the same space as A:

$$\text{span}(q_1, \dots, q_j) = \text{span}(a_1, \dots, a_j) \quad \text{where } a_j$$

- Write

$$\begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_n \\ | & & | \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{1n} \\ & \ddots & \\ & & r_{nn} \end{bmatrix} = \hat{Q}\,\hat{R}$$

Then $a_1 = r_{11} q_1$

$a_2 = r_{12} q_1 + r_{22} q_2$

$\vdots$

$a_n = r_{1n} q_1 + \dots + r_{nn} q_n$ gives what we want

where $r_{jj} > 0$

- Full QR: pad $\hat{Q}$ with extra columns

$$\underset{m\times n}{\begin{bmatrix} A \end{bmatrix}} = \underset{m\times m}{\begin{bmatrix} Q \end{bmatrix}} \underset{m\times n}{\begin{bmatrix} R \\ \hline \end{bmatrix}}$$

where $Q = \begin{bmatrix} \hat{Q} & \tilde{Q} \end{bmatrix}$, $R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}$.

Gram-Schmidt

- Given $a_1, \dots, a_n$, construct $q_1, \dots, q_n$

Have: $a_1 = r_{11} q_1$

$a_2 = r_{12} q_1 + r_{22} q_2$

$\vdots$

- Algorithm:
  - $r_{11} = \|a_1\|, \quad q_1 = \frac{a_1}{\|a_1\|}$
  - Want q_2 to be orthogonal to q_1. Note $v_2 = a_2 - (a_2^* q_1) q_1$ is perpendicular to q_1

    choose $r_{12} = (a_2^* q_1)$

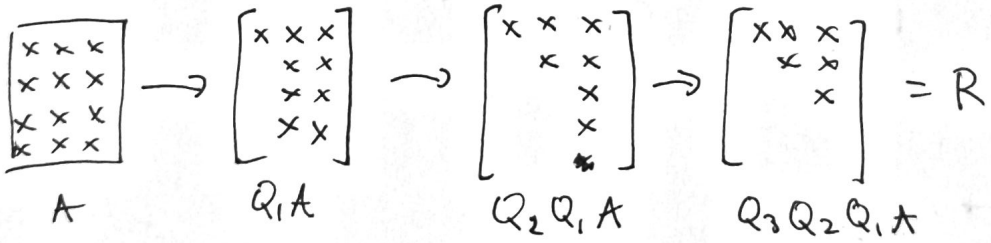
    Then $q_2 = \frac{v_2}{\|v_2\|}$ ...

Householder Algorithm

- "Orthogonal triangularization":

$$Q_n \cdots Q_1 A = R$$

- Choose Q_i such that $Q_i \cdots Q_1 A$ has 0s below diagonal in i^{th} column:



- Choosing Q_k:

$$Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & F \end{bmatrix}$$

(F: unitary)

- I_{k-1} leaves first $k-1$ columns unchanged
- F is a householder reflector induces 0s in k^{th} column.

Householder algorithm

- Choose Q_k s.t. $Q_k \cdots Q_1 A$ has 0s below diagonal in i^{th} column

$$\begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix} \longrightarrow \begin{bmatrix} * & * \\ 0 & * \\ 0 & * \end{bmatrix} \longrightarrow \begin{bmatrix} X & * \\ 0 & * \\ 0 & 0 \end{bmatrix} = R$$

A $\qquad$ $Q_1 A$ $\qquad$ $Q_2 Q_1 A$

- Choose

$$Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & F \end{bmatrix}$$

  - I_{k-1} leaves the first $k-1$ rows unchanged
  - F is chosen to create 0s in the k^{th} column
  - F should do this:

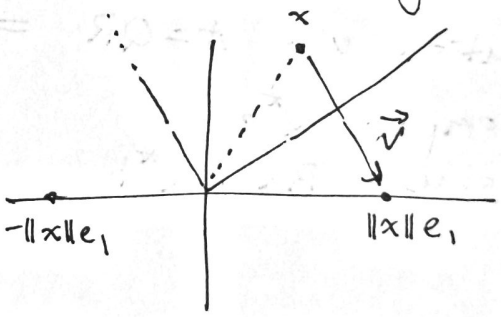
$$x = \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} \longrightarrow Fx = \begin{bmatrix} \|x\| \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  - Eliminates column
  - Orthogonal matrix so Q_k is orthogonal

  - F is a reflection:

$$F = \left(I - 2\frac{vv^*}{v^*v}\right)$$

$$\vec{v} = \|x\| e_1 - x$$



  - For stability purposes we can project onto $-\|x\| e_1$ as well (so $\vec{v}$ isn't a very small vector)

We choose

$$\vec{v} = -\operatorname{sgn}(x_1)\|x\| e_1 - x$$

- Operation count:

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

Least squares problems

- Solution of $Ax = b$ by QR:
  - * $QRx = b$
  - * Solve $Qy = b$ for y
  - * Solve $Rx = y$ for x
- The normal equations:

$$A^*Ax = A^*b \iff x \text{ minimizes } \|Ax - b\|_2$$

- Solution via Cholesky ~~$A^*A = R^*R$~~ $A^*A = R^*R \Rightarrow R^*Rx = A^*b$
  1) Compute A^*b
  2) Solve $R^*y = A^*b$ by substitution (forward)
  3) Solve $Rx = y$ by substitution (backward)
- Solution via $A = QR \Rightarrow Q^*QRx = Q^*b$
  1) Compute Q^*b
  2) Solve $Rx = Q^*b$

4 Conditioning and Stability

Condition of a problem

"problem": $f : X \to Y$
(data, solutions)

- X and Y norm-equipped
- Typically f is continuous.
- A well-conditioned problem is one where small changes in x result in small changes in $f(x)$.

Condition number

- Absolute condition number:

$$\hat{\kappa} = \lim_{\delta \to 0} \sup_{\|\delta x\| \le \delta} \frac{\|\delta f\|}{\|\delta x\|} = \|J(x)\|, \text{ if } f \text{ is differentiable.}$$

  - Large $\hat{\kappa}$ = poorly conditioned
  - small $\hat{\kappa}$ = well-conditioned.

- Relative condition number: care about relative changes:

$$\kappa = \lim_{\delta \to 0} \sup_{\|\delta x\| \le \delta} \frac{\|\delta f\|/\|f\|}{\|\delta x\|/\|x\|}$$

$$= \frac{\|J(x)\|}{\|f(x)\|/\|x\|} \quad \text{if } f \text{ is differentiable.}$$

- Condition number of a matrix

$$\text{cond}(A) = \frac{\|A\|\|x\|}{\|Ax\|} \overset{\text{ex}}{\le} \|A\|\|A^{-1}\| =: \kappa(A).$$

- Condition of a system $Ax = b$
  - Perturbation in δA causes perturbation in solution δx

$$\Rightarrow (A+\delta A)(x+\delta x) \cong Ax + (\delta A)x + A(\delta x) = b$$

$$\Rightarrow \|\delta x\| = \|A^{-1}\delta A x\| \le \|A^{-1}\|\|\delta A\|\|x\|$$

$$\to \frac{\|\delta x\|/\|x\|}{\|\delta A\|/\|A\|} \le \|A^{-1}\|\|A\|$$

Stability

- While conditioning is a characteristic of the problem itself, stability refers to the algorithm used to solve the problem.
- Algorithm: $\tilde{f}: X \to \tilde{Y}$ used to approximate $f: X \to Y$

~~Implementation~~

- Would like

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

But this might be too much too ask for, because we need to round input data, so we will settle for

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

where $\frac{\|x - \tilde{x}\|}{\|x\|} = \mathcal{O}(\varepsilon_{mach})$. **Stable algorithm.**

"Nearly the right answer to nearly the right question."

Backward Stability

- Stronger version of stability that often occurs:

$$\tilde{f}(x) = f(\tilde{x}) \quad \text{for} \quad \frac{\|x - \tilde{x}\|}{\|x\|} = \mathcal{O}(\varepsilon_{mach}).$$

"Exactly the right answer to nearly the right question."

Accuracy of a backward stable algorithm:

Theorem: Let $\tilde{f}$ be a backward stable algorithm for f.

Then $$\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(x)\|} = \mathcal{O}(\kappa(x)\varepsilon_{mach})$$

Stability of important algorithms

- QR factorization
  - Householder yields $\tilde{Q}$, $\tilde{R}$ with

$$\tilde{Q}\tilde{R} = A + \delta A \quad \text{for some } \delta A \text{ with } \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\text{Eps}) : \text{backwards stable}$$

  - Solving $Ax = b$ by QR:
    1) Form $\tilde{Q}\tilde{R}$ by Householder (Backwards stable by above).
    2) Construct $\tilde{y} \approx Q^* b$

       On a computer this solves $(\tilde{Q} + \delta Q)\tilde{y} = b$ where $\|\delta Q\| = \mathcal{O}(\text{eps})$
    3) Back solve $\tilde{R}x = \tilde{y}$

       Under FP arithmetic, get $\tilde{x}$ solving

$$(\tilde{R} + \delta R)\tilde{x} = \tilde{y}, \qquad \frac{\|\delta R\|}{\|\tilde{R}\|} = \mathcal{O}(\text{eps}).$$

This all implies that solution via QR is backwards stable, with

$$(A + \Delta A)\tilde{x} = b \quad \text{for some } \frac{\|\Delta A\|}{\|A\|} = \mathcal{O}(\text{eps}).$$

- Gaussian Elimination:
  - Without pivoting: If A has a factorization LU then

$$\tilde{L}\tilde{U} = A + \delta A, \text{ where } \frac{\|\delta A\|}{\|L\|\|U\|} = \mathcal{O}(\text{eps}).$$

    * $\|L\|$, $\|U\|$ may be arbitrarily large, so this isn't much
  - With pivoting:

$$\tilde{L}\,\tilde{U} = \tilde{P}A + \delta A \quad \text{where} \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\rho\,\varepsilon_{\text{mach}})$$

where $\rho = \dfrac{\max |u_{ij}|}{\max |a_{ij}|}$ is the growth factor.

5 Gaussian Elimination and LU

LU without Pivoting

- Transform A into U by applying lower triangular matrices $\Rightarrow$ subtract multiples of each row from subsequent rows:

$$L_{m-1} \cdots L_1 A = U$$
$$\Rightarrow A = L_1^{-1} \cdots L_{m-1}^{-1} U = LU .$$

- Example:

$$\underbrace{\begin{bmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{bmatrix}}_{A} \qquad \underbrace{\begin{bmatrix} 1 & & \\ -2 & 1 & \\ -4 & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{bmatrix}}_{L_1 A} = \left[\begin{array}{ccc} 2 & 1 & 1 \\ \hdashline 0 & 1 & 1 \\ 0 & 3 & 5 \end{array}\right]$$

- L_k is chosen such that

$$x_k = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{mk} \end{bmatrix} \xrightarrow{L_k} L_k x_k = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{kk} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- It does this by subtracting $\ell_{ik} = \frac{a_{ik}}{a_{kk}}$ times row k from row i for each row i below the diagonal $k \le i \le m$

- So L_k is given by

$$L_k = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & -\ell_{k+1,k} & \ddots \\ & & \vdots & \\ & & -\ell_{m,k} & & 1 \end{bmatrix}$$

- Turns out $L = L_1^{-1} \cdots L_k^{-1}$ is easily given in terms of the ℓ_{ij}:

$$L = \begin{bmatrix} 1 & & \\ \ell_{21} & \ddots & \\ \vdots & & \\ \ell_{m1} & \cdots & \ell_{m,m-1} & 1 \end{bmatrix} .$$

Pivoting

- Gaussian Elimination is unstable and can even outright fail:

  consider $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$.

- Solution: interchange rows of A to prevent cancellation

  $\rightsquigarrow \quad PA = LU$

Stability

- See notes on stability.
- Gaussian Elimination is back stable in practice, but ~~is unstable~~ can rack up large errors due to growth factors:

$$\tilde{L}\tilde{U} = \tilde{P}A + \delta A \qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\rho \, \text{eps})$$

where $\rho = \dfrac{\max |u_{ij}|}{\max |a_{ij}|}$ can be as large as 2^{m-1}.

6 Symmetric Positive Definite Matrices

Cholesky Decomposition

- Operates on left and right hand side of matrix during LU
- SPD Matrices $A = A^*$ $x^T A x > 0$
- Idea:
  - Write $A = \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix}$
  - Consider case $A = \begin{bmatrix} 1 & w^* \\ w & K \end{bmatrix}$
  - First LU factorization:

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

  - Exploit symmetry by factoring on right:

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

$$= R_1^* A_1 R_1$$

  - Keep proceeding along the submatrices

Complexity

- Half of LU factorization: $\frac{1}{3} m^3$ flops

Stability

- Cholesky is unconditionally stable:

$$\tilde{R}^* R = A + \delta A, \text{ where } \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(eps)$$

- Solution via Cholesky is backwards stable:
  computes $\tilde{x}$ solving $(A + \Delta A)\tilde{x} = b$ where $\frac{\|\Delta A\|}{\|A\|} = \mathcal{O}(eps)$.

Least Squares

- Since A^*A is SPD, Cholesky is the most natural choice
  - Most computationally efficient

Conjugate Gradients

- Goal. Solve $Ax = b$ where A is SPD
- Gradient descent
  - Note: $Ax = b \iff x$ minimizes $\varphi(x) = \frac{1}{2}x^T Ax - x^T b$
  - Turns problem into an optimization problems.
  - Can do iteration to minimize φ:
    $$\varphi(x_{k+1}) = x_k - \alpha_k \nabla\varphi(x_k)$$
  - But that can be extremely slow depending on the level sets.
  - Would want to choose a different search direction

Krylov Spaces

- Can define A-norm for SPD matrices: $\|x\|_A = \sqrt{x^T Ax}$
- Krylov space: $K_n = \text{span}\{b, Ab, \ldots, A^{n-1}b\}$
- Error: $e_n = (x_n - x^*)$
- Conjugate gradient: chooses $\{x_n \in K_n\}$ such that at each step, $\|e_n\|_A$ is minimized under the A-norm.

Theorem: Properties of CG iteration. For each iteration of CG,

$$\underbrace{\text{span}\{x_1, \ldots, x_n\}}_{\text{estimates}} = \underbrace{\text{span}\{r_0, \ldots, r_{n-1}\}}_{\substack{\text{residuals} \\ r_n = Ax_n - b}} = \underbrace{\text{span}\{p_0, \ldots, p_{n-1}\}}_{\substack{\text{search} \\ \text{directions}}},$$

- Furthermore, residuals are orthogonal:
  $$r_n^T r_j = 0 \quad (j < n)$$
- Search directions are A-conjugate:
  $$p_n^T A p_j = 0 \quad (j < n)$$

Complexity of CG

- Dense matrices: $O(2m^2)$ per iteration

Optimality of CG

Theorem: Let $x_1, \ldots, x_n$ be given by CG

- x_n is the unique point in $\mathcal{K}_n$ that minimizes $\|e_n\|_A$.
- Convergence is monotonic: $\|e_n\|_A \leq \|e_{n-1}\|_A$
- ~~$e_n = 0$ is achieved for some~~
- $e_n = 0$ for some $n \leq m$. (Since $\mathcal{K}_m = \mathbb{R}^m$)

However, with floating error, we'll get roundoff errors.

Polynomial Approximation

- Let $P_n = \{ n\text{-th degree polynomials with } p(0) = 1 \}$

Then $$\frac{\|e_n\|_A}{\|e_0\|_A} = \min_{p \in P_n} \frac{\|p(A)e_0\|_A}{\|e_0\|_A} \leq \min_{p \in P_n} \max_{\lambda \in eig(A)} |p(\lambda)|$$

(This is because $x_n \in \mathcal{K}_n$ so $x_n = \tilde{p}(A)b$ for some $\tilde{p}$ of degree $\leq n-1$.

- Corollary: If A has only n distinct eigenvalues, CG iteration converges in at most n steps.

(consider $p(x) = \prod_{i=1}^{n} \left(1 - \frac{x}{\lambda_i}\right)$: $|p(\lambda_i)| = 0$ for λ_i.

Rate of convergence

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^n$$

- Even if κ is large, $\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$ might be okay while $\frac{\kappa - 1}{\kappa + 1} \approx 1$.

Preconditioning

- Changing system to find a more favorable eigenvalue structure for the system.

7 Eigenvalue Problems

Fundamentals

- Eigenvalue/eigenvector solvers must be iterative methods, because we know from Galois theory that there is no analytic solution to quintic equations, for example
- Geometric multiplicity: dimension of the span of eigenvectors corresponding to λ_j.
- Algebraic multiplicity: multiplicity of root λ_j in $p(\lambda)$.

Schur Factorization

- Any matrix can be factorized $A = QTQ^*$
  Q orthogonal
  T upper triangular, w/ ev on the diagonal
- Iterative procedure $T_j = Q_j^* \cdots Q_1^* A Q_1 \cdots Q_j \to T$ as $j \to \infty$.

Hessenberg Form

- we can save ~~time~~ computational cost by reducing it to an almost-triangular matrix before iterating:
  1) Make $A = \tilde{Q} H \tilde{Q}^*$ H has zeros below first subdiagonal
  2) Iteration to get to $A = QTQ^*$
- Create upper-Hessenberg matrix by applying Householder reflectors.
- For SPD Matrices, we get tridiagonal form.
- For SVD, we can get a bidiagonal

Power Iteration

- Suppose A has set of eigenvectors ~~$v = c_1 v_1 + \dots + c_n v_n$~~ $v_1, \dots, v_n$ with $|\lambda_1| > \dots > |\lambda_n|$.
- If $v = c_1 v_1 + \dots + c_n A v_n$

  Then $Av = c_1 \lambda_1 v_1 + \dots + c_n \lambda_n v_n$

  $$A^k v = c_1 (\lambda_1)^k v_1 + \dots + c_n (\lambda_n)^k v_n = \lambda_1^k \left[c_1 v_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k + \dots \right]$$
- Algorithm:

  $w = A v^{(k-1)}$

  $v^{(k)} = \frac{w}{\|w\|}$

  converges to leading eigenvector v_1:

  $$\| v_k^{(k)} - v_1 \| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \leftarrow \text{can be very slow.}$$

Inverse Iteration

- If we have a guess for $\lambda_j \approx \mu$, then $(A - \mu I)^{-1}$ has EVs $\frac{1}{\lambda_j - \mu}$
- Hopefully, $\left|\frac{1}{\lambda_j - \mu}\right| > \left|\frac{1}{\lambda_k - \mu}\right|$ for any other k. Then we can do power iteration to get convergence to $\frac{1}{\lambda_j - \mu}$
- Algorithm:

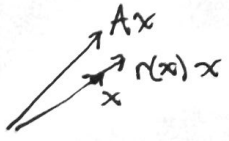
  solve $(A - \mu I) w = v^{(k-1)}$ for w

  $v^{(k)} = \frac{w}{\|w\|}$

The Rayleigh Quotient

- $r(x) = \frac{x^T A x}{x^T x}$
- If v is an eigenvector, $r(v) = \lambda$.
- In fact: $r(x) = \underset{\alpha}{\operatorname{argmin}} \|Ax - \alpha x\|$.

  $\Rightarrow$ $r(x)$ is a good eigenvalue estimate for eigenvector estimates.



Rayleigh Quotient iteration

- Put inverse iteration and power iteration together.
- Algorithm:
  - Solve $(A - \lambda^{(k-1)} I) w = v^{(k-1)}$
  - Set $v^{(k)} = \frac{w}{\|w\|}$
  - Eigenvalue estimate $\lambda^{(k)} = v^{(k)*} A v^{(k)}$.
- Convergence:
  - Rayleigh quotient iteration converges for almost everywhere $v^{(0)}$
  - CUBIC convergence!!

$$\|v^{(k-1)} - q_J\| = \mathcal{O}(\|v^{(k)} - q_J\|)$$

$$|\lambda^{(k+1)} - \lambda_J| = \mathcal{O}(|\lambda^{(k)} - \lambda_J|).$$

8 Polynomial Interpolation

- Given $\{x_0, \dots, x_n\}$ and $\{y_0, \dots, y_n\}$, find an n^{th} degree polynomial such that $p(x_i) = y_i$ for $i = 0, \dots, n$
- The interpolating polynomial exists and is unique

Lagrange Basis

- $L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$
- $L_k(x_i) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$
- Much more stable than Vandermonde matrix

Error bound

- Let $f \in C^n$ and p_n be the interpolating polynomial. Then $\exists\, \xi \in (a, b)$.

$$|p_n(x) - f(x)| = \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \right| |\pi_{n+1}(x)| \leq \frac{M_{n+1}}{(n+1)!} |\pi_{n+1}(x)|$$

where $\pi_{n+1}(x) = (x - x_0) \cdot \dots \cdot (x - x_n)$.

- since $\pi_{n+1}(x)$ depends

- Runge phenomenon: equally spaced points can get large errors on the fringes.

Hermite Interpolation

- Goal: Given $\{x_i\}_{i=0}^n$, $\{y_i\}_{i=0}^n$, $\{z_i\}_{i=0}^n$, want $p \in p^{2n+1}$ s.t.
  $p(x_i) = y_i$,
  $p'(x_i) = z_i$
- Idea: choose $H_k(x)$ $K_k(x)$ s.t.

~~$H_k(x_k) = 1$~~
~~$H_k(x_i) = 0 \quad k \neq i$~~
~~$K_k(x_k) =$~~

$$\begin{cases} H_k(x_i) = 1 \text{ if } k = i, \text{ else } 0 \\ H_k'(x_i) = 0 \\ K_k(x_i) = 0 \\ K_k'(x_i) = 1 \text{ if } k = i, \text{ else } 0. \end{cases}$$

- Then $p_{2n+1}(x) = \sum_{k=0}^{n} [y_k H_k(x) + z_k K_k(x)]$.

Cubic splines

- Idea: Use piecewise-cubic polynomials so that on each interval we are a cubic and at each abscissa the function values and derivatives match.

Chebyshev approximation / polynomials

- Def: Chebyshev Polynomials

$$T_n(x) = \cos(n \cos^{-1} x) \quad \text{real} \quad x \in [-1, 1], \quad n = 0, 1, \ldots$$

- Properties

1) $T_{n+1}(x) + T_{n-1}(x) = 2x T_n(x)$ Recurrence relation.

2) Zeros of T_n occur at $x_j = \cos \frac{(2j-1)\pi}{2n}$

3) Extrema of $T_n(x)$ occur at Chebyshev points $x_j = \cos\left(\frac{j\pi}{n}\right)$ $j = 0, \ldots, n$.

- Chebyshev series:

- If f is Lipschitz continuous on $[-1, 1]$, it has a unique Chebyshev series

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

The coefficients are given by $a_k = \frac{2}{\pi} \int_{-1}^{1} f(x) T_k(x) \frac{1}{\sqrt{1-x^2}} dx$.

- We can estimate f by the truncated series

$$f(x) \approx \sum_{k=0}^{n} a_k T_k(x)$$

or the Chebyshev interpolant:

$$f(x) \approx \sum_{k=0}^{n} c_k T_k(x)$$

where $f(x_i) = \sum_{k=0}^{n} c_k T_k(x_i)$ for $i = 0, \ldots, n$.

9 Numerical Integration

Newton_Cotes rules

- Approximate $\int_a^b f(x)\,dx$ by integral of interpolating polynomial on equidistant points.

$$\int_a^b p_n(x)\,dx = \sum_{k=0}^{n}\left(\int_a^b L_k(x)\,dx\right) f(x_k) = \sum_{k=0}^{n} w_k f(x_k).$$

- Trapezoid rule:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right)$$

- Simpson's rule

$$\int_a^b f(x)\,dx \simeq \frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

- Error:

$$E_n(f) = \int_a^b f(x)\,dx - \sum_{k=0}^{n} w_k f(x_k)$$

$$|E_n(f)| \le \frac{M_{n+1}}{(n+1)!}\int_a^b |\pi_{n+1}(x)|\,dx, \quad M_{n+1} = \max_{\xi\in[a,b]} f^{(n+1)}(\xi)$$

- Error is given by the pdynominal error
- Trapezoid rule: $E_1(f) = \frac{(b-a)^3}{12} M_2$
- Simpson's rule: $E_2(f) = \frac{(b-a)^4}{192} M_3$

Composite Integration

- Break up into subintervals of size h and use NC rules on each subinterval.
- Trapezoidal rule: $T(m) = h\left[\frac{1}{2}f(x_0) + f(x_1) + \dots + f(x_{m-1}) + \frac{1}{2}f(x_m)\right]$
- Error: $|E_1(f)| \le \frac{(b-a)}{12} h^2 M_2$
- Simpson rule: $S(2m) = \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4(x_3) + \dots\right]$

  Error $|E_2(f)| \le \frac{b-a}{180} h^4$

Gaussian Quadrature

- Instead of choosing equally spaced points like with Newton-Cotes, choose points so that some weights are zero.
- Consider Hermite Polynomial, but choose ~~weights~~ abscissas so that derivatives don't factor in.
- Results in Quad rule which is exact for polynomials of degree $2n+1$.

- Consider integral

$$I = \int_a^b w(x) f(x)\, dx$$

- Let $\{x_i\}_{i=0}^n$ be the points of interpolation for Hermite polynomial:

$$p_{2n+1}(x) = \sum_{k=0}^{n} f(x_k) H_k(x) + \sum_{k=0}^{n} f'(x_k) K_k(x)$$

$$\int w(x)\, p_{2n+1}(x)\, dx.$$

- Example weight functions:
  - $w(x) = 1$ (Inner product for Legendre basis)
  - $w(x) = \frac{1}{\sqrt{1-x^2}}$ Chebyshev weights

WLOG $w(x) = 1$:

- Integral of $p_{2n+1}(x)$:

$$\int_a^b p_{2n+1}(x)\, dx = \sum_{k=0}^{n} f(x_k) W_k + \sum_{k=0}^{n} f'(x_k) V_k$$

where $W_k = \int H_k(x)\, dx$ $V_k = \int K_k(x)\, dx$

- V_k' can be written in terms of the polynomial $\pi_{n+1}(x)$:

$$V_k = C_k \int \pi_{n+1}(x) L_k(x)\, dx \qquad C_k = \prod_{i \neq k} \frac{1}{(x_k - x_i)}$$

- Key idea: choose $x_0, \ldots, x_n$ so that $\pi_{n+1}(x)$ is orthogonal to $L_k(x)$ to all polynomials of degree $\leq n$, and thus $L_k(x)$.

- Error: $I - G_n(f) = \frac{f^{(2n+2)}(\xi)}{(2n+1)!} \int_a^b w(x)\, \pi_{n+1}^2(x)\, dx.$

10 Nonlinear Equations

Contractive Mapping Theorem

- Want to find fixed points $f(\xi) = \xi$
- Lipschitz continuous: $\| g(x) - g(y) \| \leq L \| x - y \|$
- Contraction: Lipschitz with $L < 1$.

Theorem: Contractive mapping theorem.

Suppose
- D is closed
- $g(D) \subseteq D$
- g is a contraction

Then
1) g has a unique fixed point $\xi \in D$
2) The iterative procedure $x_{k+1} = g(x_k)$ converges to ξ for all $x_0 \in D$.

Newton's Method

- Goal: find roots x such that $f(x) = 0$
- Consider
$$g(x) = x - [J_f(x)]^{-1} f(x)$$

$g(x) = x \iff f(x) = 0$ assuming $J_f(x)$ is nonsingular.

Theorem: Suppose
1) $f(\xi) = 0$ is a root
2) $N_r(\xi) \subseteq D$ for some $r > 0$
3) Jacobian is nonsingular; with $\| J(\xi)^{-1} \| \leq \beta$
4) Jacobian is Lipschitz: $\| J(x) - J(y) \| \leq \gamma \| x - y \|$ for $x, y \in N_r(\xi)$

Let $\varepsilon = \min(r, \frac{1}{2\beta\gamma})$. Then Newton's method $x_{k+1} = x_k - J(x_k)^{-1} f(x_k)$ converges quadratically for $x_0 \in N_\varepsilon(\xi_0)$:

$$\| x_{k+1} - \xi \| \leq \beta\gamma \| x_k - \xi \|^2$$