CSS322 Scientific Computing
Final Exam Summary

Lecture 10
Linear Least Square

♠ matrix A is full rank.

⇒ we have system of $Ax = b$
minimize $\|Ax-b\|_2$
fit some equations

♠ ① Method of Normal Equation
solve $(A^TA)x = A^Tb$
step 1) find A^TA and A^Tb
2) factorize $A^TA = LL^T$ by Cholesky's
3) $Lw = A^Tb$ ⇒ forward-sub
4) $L^Tx = w$ ⇒ backward-sub

② QR-factorization
$A = QR$
$m \times n$ ↳ Upper triangular matrix ($m \times n$)
↳ orthogonal matrix ($m \times m$)
step 1) find $A = QR$
2) compute $c = Q^Tb$
3) solve $R_1x = c_1$ // back-sub

how to find $A = QR$
↳ "House Holder Method"

QR-factorization by House Holder method.

⇒ $Ax \cong b$ (A size $m \times n$) $i = 1, \dots, n$

loop n times (# of col.)

form QR
① a = first column of A $a = \begin{bmatrix} a_1 \\ \vdots \end{bmatrix}$
② $\alpha = \begin{cases} -\|a\|_2 & ; a_1 \geq 0 \\ +\|a\|_2 & ; a_1 < 0 \end{cases}$ $e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$
③ $v = a - \alpha e_1$
④ $H_i' = I - 2\frac{vv^T}{v^Tv}$ $H_i = \begin{bmatrix} I & 0 \\ 0 & H_i' \end{bmatrix}$
⑤ find $H_i \dots H_1 A$
⑥ eliminate first row & first col. of $H_i \dots H_1 A$
⑦ $R = H_n \dots H_1 A$ / $Q = H_1 \dots H_n$ * not necessary to calculate
↳ $= \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$

form c
(×n)
⑧ find $H_i \dots H_1 b$
⑨ $H_i \dots H_1 b = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$; lock w_1 cut at i

then $c = H_n \dots H_1 b$
$\therefore c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$

⑩ solve $R_1x = c_1$ by back-sub

Lecture 11
Non-Linear Equations.

→ solve for root of equations

a.k.a. $f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

we ⊗ iterative method.

① Interval Bisection

↳ for 1 nonlinear equation on interval $[a,b]$

until interval is small enough

① find $f(a)$ & $f(b)$
② find midpoint $m = a + (b-a)/2$ & $f(m)$
③ if $\text{sign}(f(m)) \neq \text{sign}(f(a))$
→ root exists inside $[a,m]$
else if $\text{sign}(f(m)) \neq \text{sign}(f(b))$
→ root exist inside $[m,b]$

in case you forgot about partial derivatives

$$f(x_1, x_2) = x_1^2 x_2$$

$$\frac{\partial f}{\partial x_1} = 2x_1x_2 \quad ; \quad \frac{\partial f}{\partial x_2} = x_1^2$$

② Newton's Method

based on 2-terms taylor series
aka. $f(x+h) = f(x) + f'(x)h$

$x^{(0)}$ as initial guess

⊗ $x^{(k+1)} = x^{(k)} - f(x^{(k)}) / f'(x^{(k)}), \; k = 0, 1, \ldots$
⊗ if $x^{(k)}$ is a root, $x^{(k+1)} = x^{(k)}$

③ Secant Method.

$x^{(0)}, x^{(1)}$ as "2" initial guesses

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})(x^{(k)} - x^{(k-1)})}{f(x^{(k)}) - f(x^{(k-1)})}$$

⊗ System of Non-linear Equations

→ derived Jacobian matrix

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

④ Newton's Method for system of nonlinear equations

given $f(x) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

① then setup Jacobian matrix $\nabla f(x)$
$x^{(0)}$ as initial guess

② solve $h^{(k)}$
$\nabla f(x^{(k)}) h^{(k)} = -f(x^{(k)})$ by using GEPP

③ then $x^{(k+1)} = x^{(k)} + h^{(k)}$
loop until solution is accurate enough.

⑤ Secant Updating Method / Broyden's Method.

① set Jacobian $\nabla f(x)$
and initial guess $x^{(0)}$

② $B^{(0)} = \nabla f(x^{(0)})$

③ solve $B^{(k)} h^{(k)} = -f(x^{(k)})$

④ find $B^{(k+1)} = B^{(k)} + \left(\frac{(y^{(k)} - B^{(k)} h^{(k)}) h^{(k)T}}{h^{(k)T} h^{(k)}} \right)$

looped until accurate enough.

Lecture 12
Optimization

A) One-Dimension

→ find x that minimize $f(x)$
↳ minimizer, denoted as x^*

① Successive Parabolic Interpolation

1) given $f(x)$ & 3 initial points
↳ x_0^u, x_1^v, x_2^w

2) find $f(x_0), f(x_1), f(x_2)$

3) find x_k = $\quad x_0 = u, x_1 = v, x_2 = w$

loop

$$v - \frac{1}{2}\frac{(v-u)^2(f_v - f_w) - (v-w)^2(f_v - f_u)}{(v-u)(f_v - f_w) - (v-w)(f_v - f_u)}.$$

4) next round; $x_0 = x_1$, $x_1 = x_2$, $x_2 = x_3$

② Newton's Method

① given $f(x)$ and $x^{(0)}$
② find $f'(x)$ & $f''(x)$

loop ③ find $x^{(k+1)} = x^{(k)} - \dfrac{f'(x^{(k)})}{f''(x^{(k)})}$

→ Multivariate Derivatives

given $f(x)$

then "gradient" $\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$

"Hessian Matrix" $\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$

③ Newton's Method.

① given $f(x)$ and $x^{(0)}$ as initial guess

loop ② solve $(\nabla^2 f(x^{(k)}))(h^{(k)}) = -\nabla f(x^{(k)})$ for $h^{(k)}$

③ $x^{(k+1)} = x^{(k)} + h^{(k)}$

④ BFGS
(Secant updating method.)

$x^{(0)}$ = initial guess
$B^{(0)}$ = initial Hessian approximation
For $k = 0, 1, 2, \ldots$
  Solve $B^{(k)} h^{(k)} = -\nabla f(x^{(k)})$ for $h^{(k)}$
  $x^{(k+1)} = x^{(k)} + h^{(k)}$
  $y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$
  $B^{(k+1)} = B^{(k)} + (y^{(k)}(y^{(k)})^T)/((y^{(k)})^T h^{(k)})$
  $\quad - (B^{(k)} h^{(k)} (h^{(k)})^T B^{(k)}) / ((h^{(k)})^T B^{(k)} h^{(k)})$
Endfor

Lecture 13
IVP for ODE

① EM : Euler's Method.
* not quite accurate

given $\frac{dy}{dt} = f(t,y)$, $y(0) = y_0$ (initial)

find. $y_{k+1} = y_k + h_k f(t_k, y_k)$

step size : $h_k = t_{k+1} - t_k$ ⇒ constant.

② Linear Multistep Method : LMS

→ AB2 : Adams-Bashforth Second order
* required 2 initial Points

$$y_{k+1} = y_k + \frac{3h}{2} f(t_k, y_k) - \frac{h}{2} (t_{k-1}, y_{k-1})$$

$h = t_{k+1} - t_k = t_k - t_{k-1}$; step size is constant.

→ if given only 1 initial point,
find second initial point using euler's method.

③ Rk : Runge-Kutta method
a.k.a. Heun's method.

$$y_{k+1} = y_k + \frac{h}{2}(s_1 + s_2)$$

$$s_1 = f(t_k, y_k)$$

$$s_2 = f(t_k + h, y_k + h s_1)$$

④ BE : Backward Euler.

solve $y_{k+1} = y_k + h f(t_{k+1}, y_{k+1})$

⑤ Implicit Trapezoid method.

$$y_{k+1} = y_k + h\left(\frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2}\right)$$

⑥ higher level ODE

transform into equivalent first-order system

$$u' = \begin{bmatrix} u_1' \\ \vdots \\ u_k' \end{bmatrix} = \begin{bmatrix} u_2 \\ \vdots \\ u_k \\ f(t, u_1, \ldots, u_k) \end{bmatrix}$$

Lecture 14

SVD : Singular Value Decomposition

Any matrix $A_{m\times n} = U \Sigma V^T$

- U → orthogonal $n \times n$
- Σ → diagonal $m \times n$
- V → orthogonal $m \times m$

$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \\ & & & \sigma_n \end{bmatrix}$

all σ_i are unique → singular values

→ yields 2-norms of matrix

if $A = U \Sigma V^T$ then $\|A\|_2 = \sigma_1$

(largest singular value)

→ used to solve LLS Problem when matrix A is not full rank. $Ax \cong b$

the solution is $x = \sum_{\sigma_i \neq 0} \frac{u_i^T b}{\sigma_i} v_i$

u_i = ith column of U / v_i = i-th column of V

$A = U \Sigma V^T$

→ if A is invertible, then $\|A^{-1}\|_2 = 1/\sigma_n$

↳ $A^{-1} = V \Sigma^{-1} U^T$

* this is not SVD of A^{-1}

SVD of $A^{-1} = (VP)(P\Sigma^{-1}P)(PU^T)$

P = permutation matrix $\begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}$

→ condition number (2-norm)

$$\text{cond}_2(A) = \frac{\sigma_1}{\sigma_n} \quad | \quad \text{cond}_2(A^TA) = (\text{cond}_2(A))^2$$

→ rank : number of nonzero singular values (σ)

→ How to find SVD decomposition

★ Find eigenvalues & eigenvectors of A^TA & AA^T

→ eigenvectors of A^TA = columns of V

→ — " — AA^T = — " — U

- square root of eigenvalues of A^TA (or AA^T) = Singular values

Lecture 15
Eigenvalues & Eigenvectors

Characteristic Polynomials
$\det(A - \lambda I) = 0$ → solve for λ
- always in degree of n
- has exactly n eigenvalues, but not necessarily distinct.

① Power Method.
① $A \in \mathbb{R}^{n \times n}$, $x^{(0)}$
② $x^{(k+1)} = Ax^{(k)}$

if A has a unique eigenvalues w/ maximum abs. value.
then power methods converges to the eigenvector of that.
→ it becomes large / either overflow, underflow

② Normalized Power method.
① $A_{n \times n}$, $x^{(0)}$
② $\tilde{x}^{(k)} = Ax^{(k)}$
③ $f^{(k)} = \| \tilde{x}^{(k)} \|_2$
④ $x^{(k+1)} = \frac{\tilde{x}^{(k)}}{f^{(k)}}$ → normalization

→ finding eigenvalues assoc. w/ eigenvector

* Rayleigh Quotient

$$\lambda = \frac{(x^{(k)})^T A x^{(k)}}{(x^{(k)})^T x^{(k)}}$$

③ Inverse Power Method.
① $A_{n \times n}$, $x^{(0)}$
② $x^{(k+1)} = A^{-1} x^{(k)}$ → normalized by $x^{(k+1)} = x^{(k+1)} / \| x^{(k+1)} \|_2$

- form A^{-1} by GEPP! → apply power method to A^{-1}
- if A is invertible, then A & A^{-1} have the same eig. vector and if λ is eigvalue of A then $1/\lambda$ is of A^{-1}
- if A has a unique eigenvalues w/ minimum abs. value.

then inverse power methods converges to the eigenvector of that.

④ Inverse Shifted Power Method
$x^{(k+1)} = (A - \sigma I)^{-1} x^{(k)}$ // normalization
↳ choose in advance

→ if A has a unique eigenvalues w/ closest to σ
then inv. shifted power method conv. to the eigenvector of that.

⑤ QR Iteration → convergence is slow
① $M^{(0)} = A$
② $k = 0, \ldots$
2.1) factor $M^{(k)} = Q^{(k)} R^{(k)}$ (known M^k, compute $Q^k R^k$ by factorization) QR
2.2) Define $M^{(k+1)} = R^{(k)} Q^{(k)}$ (known Q^k, R^k, compute M^{k+1})

Lecture 16
BVP for ODE

→ second-order scalar boundary value problems

$$u'' = f(t, u, u'), \quad a < t < b$$

boundary conditions: $u(a) = \alpha$, $u(b) = \beta$

① Shooting Method.

→ approx satisfy ODE from start and iterate until cond. are satisfied.

1) guess $u'(a)$
2) solve IVP
3) check the solution at $t=b$ is equal to β or not?
4) if yes, then this is a solution
   no, try different $u'(a)$ then repeat

② Finite different method

→ mesh point $t_i = a + ih$, $h = (b-a)/(n+1)$
aka step size

$$(y_{i+1} - 2y_i + y_{i-1})/h^2 = f\left(t_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right)$$

→ in calculate include boundary points w/ mesh point.

③ Collocation Method

$$u(t) \approx v(t, x) = \sum_{i=1}^{n} x_i \phi_i(t)$$

x_i: coefficient
$\phi_i(t)$: basis functions → can analytically computed derivatives
n: usually used monomial basis

④ Residual Method.

Possion equation $u''(t) = f(t)$, $a < t < b$

boundary ⇒ homogenous $u(a) = 0$, $u(b) = 0$

$$r(t,x) = v''(t,x) - f(t) = \sum_{i=1}^{n} x_i \phi_i''(t) - f(t)$$

or $\frac{\partial r}{\partial x_i} = \phi_i''(t)$

⑤ LS method.

system of linear equation $Ax = b$ where

$$a_{ij} = \int_a^b \phi_j''(t)\phi_i''(t)dt,$$

$$b_i = \int_a^b f(t)\phi_i''(t)dt.$$

⑥ Weighted Residual

$$a_{ij} = \int_a^b \phi_j''(t)w_i(t)dt,$$

$$b_i = \int_a^b f(t)w_i(t)dt.$$

⑦ Galerkin Method.

ere

$$a_{ij} = -\int_a^b \phi_j'(t)\phi_i'(t)dt,$$

$$b_i = \int_a^b f(t)\phi_i(t)dt.$$

Lecture 17 PDE

① Heat Equation

$u_t = cu_{xx}$ given initial cond. $u(0,x) = f(x)$

$0 \le x \le L$, $t \ge 0$

boundary $u(t,0) = \alpha$, $u(t,L) = \beta$

② Wave Equation $u_{tt} = cu_{xx}$

③ Laplace Equation $u_{xx} + u_{yy} = 0$

Time-Dependent Problem

① Method of lines

$$y_i'(t) = \frac{c}{(\Delta x)^2}\left(y_{i+1}(t) - 2y_i(t) + y_{i-1}(t)\right), \quad i = 1, \ldots, n,$$

② fully discrete method. → heat equation

$$u_i^{k+1} = u_i^k + c\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1}^k - 2u_i^k + u_{i-1}^k\right)$$

→ wave equation

$$u_i^{k+1} = 2u_i^k - u_i^{k-1} + c\left(\frac{\Delta t}{\Delta x}\right)^2\left(u_{i+1}^k - 2u_i^k + u_{i-1}^k\right)$$

→ Crank-Nicholson Method

$$u_i^{k+1} = u_i^k + c\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1} + u_{i+1}^k - 2u_i^k + u_{i-1}^k\right),$$

Time-independent.

① finite difference method.

→ w/ laplace equation

$u_{xx} + u_{yy} = 0$

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0,$$

Lecture 18
Optimization

① Naive Random Search
→ pick a point from neighborhood
* may stuck at local minimizer

② Simulated Annealing (SA)
//Ref. CS5329 Lecture Optimization
* to get global min, sometimes you need to pick some larger value (worse choice)"
→ cooling schedule
→ mimics temperature of metal
"If Temp is high, the chance to go explore the neighborhood is high"

③ Particle swarm optimization (PSO)
→ instead of updating single value, we updated "population" aka set of candidate → swarm.
→ particle = each candidate solution
→ each particle tracks its p-best position
→ the best-so-far overall pos. ⇒ gbest

1. Set $k := 0$. Generate an initial population $P(0)$.
2. Evaluate $P(k)$.
3. If stopping criterion is satisfied, stop.
4. Select $M(k)$ from $P(k)$.
5. Evolve $M(k)$ to form $P(k+1)$.
6. Set $k := k+1$, go to step 2.

④ Genetic Algorithm (GA)

solve $\max_{x \in \mathbb{R}^n} f(x)$ (find global max.)

→ mimics "natural selection of charles darwin".
↳ survival of the fittest.

GA: not work directly w/ points in $\mathbb{R}^n$
→ encoding! (map $\mathbb{R}^n$ to set of strings)

fitness: objective f^n value represented by chromosome

chromosome $x \to f(x)$ = fitness

① initial population: $P(0)$

② form new generation of population by

→ 2a) Selection:

each chromosome in ppl. is selected by probability $\frac{f(x^{(k)})}{F(k)}$ ← fitness / ← sum of all fitness in ppl.

aka. more fitness, more likely to be selected.

↳ $M(k)$ is set of selected chromosomes // mating pool.

↳ 2b) Evolution / Crossover
→ take a pair of chromosome, select point, then crossover
→ add to mating pool.

↳ 2b) Evolution / Mutation
→ take each chromosome from $M(k)$ → randomly change symbol

③ Real - Number GA

→ Define Crossover & mutation operations. / x, y is parent.

option a) midpoint / average

Crossover

- offspring $z = (x+y)/2$
- b) $z_1 = (x+y)/2 + w_1$ / $z_2 = (x+y)/2 + w_2$
  - w_1 & w_2: randomly generated vector
- "convex"
  - a) $z_1 = \alpha x + (1-\alpha)y$, $z_2 = (1-\alpha)x + \alpha y$
  - b) $z_1 = \alpha x + (1-\alpha)y + w_1$, $z_2 = (1-\alpha)x + \alpha y + w_2$

  $\alpha \in [0, 1]$ (randomly)
  w_1 & w_2 random generated vectors.

Mutation

Optional) real number creep.

=) add random vector to chromosome.

b) replace chromosome w/ random convex combination

▶ Lecture 19

Iterative methods for linear system.

* lower accuracy but more efficiently

* Problem: Solve $Ax = b$ for x

① Stationary Iterative Method.

→ initial guess $x^{(0)}$

→ Iterate. $x^{(k+1)} = G x^{(k)} + c$ (G: matrix, c: vector constant)

How to find G & c? → "Splitting"

* $A = M - N$ (M nonsingular)

Take $G = M^{-1}N$, $c = M^{-1}b$

Therefore, we solve for $Mx^{(k+1)} = Nx^{(k)} + b$

② Jacobi Method.

Let matrix D → diagonal matrix w/ same diagonal entries of A

L & U → strictly lower & upper portion of A

Splitting: $M = D$, $N = -(L+U)$

* not always converge! → needs initial guess $x^{(0)}$

$$x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}}, \quad \text{for } i = 1, \ldots, n$$

(round)

③ Gauss-Seidel Method.

Splitting: $M = D + L$, $N = -U$

$$x_i^{(k+1)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}}{a_{ii}},$$

(on left; next guess; right; current guess)

④ Successive Over-Relaxation (SOR)

* not included in exam

Choose a fixed **relaxation parameter** (scalar) ω.

From $x^{(k)}$, first compute the next iterate that would be given by the Gauss-Seidel method, $x_{GS}^{(k+1)}$.

Then instead set

$$x^{(k+1)} = x^{(k)} + \omega \left(x_{GS}^{(k+1)} - x^{(k)} \right).$$