Sci Comp II: Study Guide

(I) ODE / PDE Formulations

(I) Finite Differences: $U_{xx} = f$ $\qquad u'' \approx \dfrac{U_{k+1} - 2U_k + U_{k-1}}{h^2}$ $\qquad \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \end{bmatrix}\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$

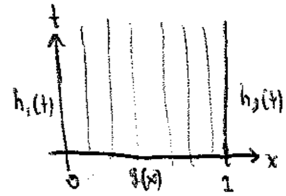
$A \quad x \quad = \quad b$

- eigenvalues of A: $-4\sin^2\left(\frac{k\pi}{2n}\right)n^2$

Note: Crank-Nicholson → trapezoid in time, centered diff in space

(II) Method of Lines: ① discretize in space $\qquad \frac{\partial u}{\partial t} = U_{xx} + f(x,t)$

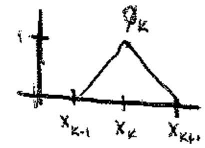
② left w/ sys of ODEs in time



$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{pmatrix} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \end{bmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{pmatrix} + \begin{pmatrix} f_1(t) + \frac{1}{h^2}h_1(t) \\ f_2(t) \\ \vdots \\ f_{n-1}(t) + \frac{1}{h^2}h_2(t) \end{pmatrix}$$

$$\frac{d\vec{U}}{dt} = A\vec{U} + \vec{F}$$ now use RK4, etc.

(III) Finite Elements: $u'' = f \qquad u(0) = u(1) = 0 \qquad$ Assume $u(x) = \sum_{k=0}^{N} u_k \phi_k$



- **Weak Form:** $\int_0^1 (u'' - f)\, v\, dx = 0$

$$\Rightarrow \int_0^1 u''v\, dx = \int_0^1 fv\, dx$$

$$\underbrace{vu'\Big|_0^1}_{0} - \int_0^1 u'v'\,dx = \int_0^1 fv \quad ; \quad \text{let } v = \phi_i$$

$$\Rightarrow -\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \end{bmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix}$$ same as finite difference

- **Energy Minimization:** $\min\ \mathcal{J}(v(x)) = \frac{1}{2}\int_0^1 \left[v'^2 + 2fv\right] dx$ where $v \in H_0^{(n)}$ ← n piecewise lines, 0 BCs

$$\mathcal{J}(u + \varepsilon v) = \frac{1}{2}\int_0^1 \left([u' + \varepsilon v']^2 + 2f[u + \varepsilon v]\right) dx$$

$$= \frac{1}{2}\int_0^1 (u'^2 + 2fu)\,dx + \frac{\varepsilon}{2}\underbrace{\int_0^1 (2u'v' + 2fv)\,dx}_{\text{need this to go to zero}} + \frac{1}{2}\varepsilon^2\int_0^1 v'^2\,dx$$

hence equivalent to weak form

- **Example:** $-u''k - u'k' + b(x)u = \frac{1}{2}f$

Weak Form: $\int_0^1 \left(-u''k - u'k' + bu - \frac{1}{2}f\right) v\, dx$

$$= \int_0^1 \left(ku'v' + buv - \frac{1}{2}fv\right) dv\,,$$

$$\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} u_j \left(\int_0^1 k(x)\,\phi_i'\phi_j'\,dx\right) u_i + \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} u_j \left(\int_0^1 b\,\phi_i\phi_j\,dx\right) u_i = \sum_{i=1}^{N-1} u_i \int_0^1 \phi_i f\,dx$$

$$J(u) = u^T A u + u^T B u - u^T F$$

$$\frac{\partial J(u)}{\partial u} = (A+B)u - F = 0 \therefore \quad (A+B)u = F$$

(IV) Integral Equations: eg $u'' = f \quad u(0)=u(1)=0 \qquad u(x) = \int_0^1 G(x,t)\, f(t)\, dt \qquad G = \begin{cases} x(t-1) & x \le t \\ t(x-1) & x \ge t \end{cases}$

$$u(x) = (x-1)\int_0^x t\, f(t)\, dt + x\int_x^1 (t-1) f(t)\, dt$$

$$u'(x) = \int_0^x t\, f(t)\,dt + \cancel{x(x-1)f(x)} + \int_x^1 (t-1) f(t)\,dt - \cancel{x(x-1)f(x)}$$

$$u''(x) = x f(x) - (x-1)f(x) = f(x) \quad ✓$$

(II) Solving $Ax=b$

(I) Gaussian Elimination: $\sim \frac{2}{3}N^3$

not always stable but fast

- $A = LU$ — $\Theta(N^3)$

$Ax = b$

$LUx = b \rightarrow Ly = b \sim \Theta(N^2)$

then solve $Ux = y \sim \Theta(N^2)$

- Condition Number: $K_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$

- Error Analysis: $(A+\delta A)(x+\delta x) = b + \delta b$

$$\delta A x + (A+\delta A)\delta x = \delta b$$

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

$$\frac{\|\delta x\|}{\|x\|} \le \underbrace{\|A\|\|A^{-1}\|}_{\text{condition \#}} \left(\frac{\|\delta b\|}{\|A\|} + \frac{\|\delta A x\|}{\|x\|\|A\|}\right)$$

- Banded Matrix: $\Theta(2N \cdot B_U B_L)$ — upper/lower bandwidths

- Theorem: If A is symmetric positive definite, no need to pivot

- Cholesky: $\Theta(\frac{1}{3}N^3) \quad A = LL^T \leftarrow$ stable/fast

(II) QR: $\Theta(\frac{4}{3}N^3)$

- Gram Schmidt:

for $j=1,\dots,n$

$a_j = v_j$

for $i=1,\dots,j-1$

$r_{ij} = q_i^T v_j$

$a_j = a_j - r_{ij} q_i$

end

$r_{jj} = \|a_j\|^2$

$q_j = a_j / r_{jj}$

$\downarrow$

"triangular orthogonalization"

- Solving $Ax=b$:

$Ax = b$

$QRx = b \quad \sim N^3$

$Rx = Q^* b \quad \sim N^2$

- Note: $\|A - QR\| \longleftarrow$ will be small for Gram schmidt, modified Gram Schmidt, and Householder

$\|Q^*Q - I\| \longleftarrow$ will be large for GS and MGS, but small for Householder

- Householder: "orthogonal triangularization" $\quad A \to Q_1 A \to Q_2 Q_1 A \qquad Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}$; I is $(k-1)\times(k-1)$

- Note: If λ is eigenvalue of A, it is also an eigenvalue of QAQ^* $\qquad F = I - \frac{2vv^*}{v^*v}$

(III) SVD: $A = U\Sigma V^*$; U, V^* are orthogonal matrices $\quad$ * low rank application *

1) Eigenvectors:

$$A^TA = V\Sigma^2 V^T$$

$$\therefore (A^TA)V = V\Sigma^2$$

σ_k^2 is eigenvalue of A^TA and $\vec{v}_k$ is eigenvector of A^TA

$$AA^T = U\Sigma^2 U^T$$

$$AA^TU = U\Sigma^2$$

2) $Ax=b \longrightarrow U\Sigma V^* x = b$

$$\Sigma V^* x = U^* b$$

$$x = V\Sigma^{-1}U^* b$$

(IV) Least Squares: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{C}^m$ find $x \in \mathbb{C}^n$ such that $\|Ax-b\|_2$ is minimized

(1) $Ax = b$

$A^TAx = A^T \rightarrow x = (A^TA)^{-1}A^T$; ie: $A^T(Ax-b) = A^Tr = 0$

(I) Cholesky: $A^*A = R^*R$ $\quad$ * A is full rank * $\qquad$ * speed $\sim \frac{1}{3}N^3$

R - upper Δ

$$A^*Ax = R^*Rx = A^*b$$

$$R^*y = A^*b$$

$$Rx = y$$

(II) **QR**: $A = \hat{Q}\hat{R}$

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

* great for stability w/ Householder *
* A can't be rank deficient *

(III) **SVD**: $A = \hat{U}\hat{\Sigma}V^*$ w/ $P_b = UU^* b$

$$Ax = b$$
$$\hat{U}\hat{\Sigma}V^*x = UU^* b$$
$$\Sigma w = U^* b$$
$$V^* x = w \rightarrow x = Vw$$

* great for rank deficient *

* Every matrix has a SVD *

(V) Eigenvalues:

(I) Assume real symmetric matrix for QR iteration

- In general, finding eigenvals/vectors of a general matrix is ill-conditioned

(II) **Power Method**: Given $\vec{x}$,

for $n = 0, 1, \ldots$

$$v_n = \frac{\vec{x}}{\|\vec{x}\|}$$
$$v_{n+1} = \frac{A\vec{x}}{\|Ax\|}$$
$$v_{n+2} = \frac{A^2 x}{\|A^2 x\|}$$
$\vdots$

* quadratic convergence *

Idea: $x = c_1\vec{e}_1 + \ldots c_n\vec{e}_n$

$$A^k x = c_1 A^k e_1 + \ldots c_n A^k \vec{e}_n = c_1\lambda_1^k \vec{e}_n + \ldots c_n \lambda_n^k \vec{e}$$

divide by λ_1

$$\frac{A^k x}{\lambda} = c_1\vec{e}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k \vec{e}_2 + \ldots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k e_n$$

Now if $|\lambda_1| > |\lambda_2| > |\lambda_3| \ldots$

$$\frac{A^k x}{\lambda_1} \approx c_1\vec{e}_1$$

(III) **Inverse Power Method**: $v_0^{(0)}$ = some v w/ $\|v\| = 1$

for $k = 1, 2, \ldots$

$$(A - \mu I)w = v^{(k-1)}$$
$$v^{(k)} = w/\|w\|$$
$$\lambda^{(k)} = (v^*)^T A v^{(k)}$$

* use Rayleigh Quotient $\mu^{(k)}$, $r(x) = \frac{x^T A x}{x^T x}$

↳ cubic convergence

*: $Ae_i = \lambda_i e_i$

$(A - cI)e_i = (\lambda_i - c)e_i$ → smallest eigenvalue in new sys is the eigenvalue closest to c.

(IV) **Overview**:

① **2 Phases**: 1) $A \rightarrow$ Hessenberg Form (or tri-diag if $A = A^*$) $\sim \theta(N^3)$

2) Hessenberg (tri-diag) → Upper triangular (diagonal if $A = A^*$) $\sim \theta(N^2)$

② **Reduction to Hessenberg** → apply Householder reflectors; $Q_k^* \cdots Q_1^* A Q_1 Q_2 \cdots Q_k$

$\sim \frac{10}{3}m^3$ ← general

$\sim \frac{4}{3}m^3$ ← Hermitian

backward stable

③ **Schur**: $A = QTQ^*$ ← Every A

if A is normal: $A^*A = AA^*$

→ **Unitary Diagonalizable**: $(A = A^*)$: $A = Q\Lambda Q^*$ ← SVD decomp

(V) Rayleigh Quotient: $r(x) = \frac{x^T A x}{x^T x}$ ∴ $r(\vec{x}) = \lambda$ iff λ is an eigenvalue associated w/ e-vec $\vec{x}$

Note: **Quadratic Convergence**: $r(\tilde{v}_i) = \underbrace{r(\vec{v}_i)}_{\lambda} + \underbrace{\nabla r(\vec{v}_i)}_{0}(\tilde{v}_i - \vec{v}_i) + \nabla^2 r(\vec{v}_i)(\tilde{v}_i - \vec{v}_i)^2 + \dots$

$\rightarrow r(\tilde{v}_i) - \lambda = \Theta(\|\tilde{v}_i - \vec{v}_i\|^2)$ ∴ $\nabla r = \frac{2}{x^T x}(Ax - r(x)x)$

(VI) QR Algorithm: procedure (stable) for computing QR factorization of $A, A^2, A^3, \dots$

- cubic convergence
- $A = A^* = A^T \in \mathbb{R}^{N \times N}$

$Q^{(0)T} A^{(0)} Q^{(0)} = A$ ← tridiagonalization step

for $k = 1, 2, \dots$

pick $\mu^{(k)} = A_{mm}^{(k-1)}$

$Q^{(k)} R^{(k)} = A^{(k-1)} - \mu^{(k)} I$ ← QR of matrix $A^{(k-1)} - \mu^{(k)} I$

$A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I$

- Simultaneous: $Q^{(0)} = I$

for $k = 1, 2, \dots$

$Z = A Q^{(k-1)}$

$Q^{(k)} R^{(k)} = Z$ ← orthonormal bases for A^k

$A^{(k)} = (Q^{(k)})^T A Q^{(k)}$ ← diag vals of $A^{(k)}$ are Rayleigh Quotients

(VI) Krylov: (I) used in low rank approximations ; $K_n = \{b, Ab, \dots, A^{n-1}b\} = \{q_1, q_2, \dots q_n\} R_n$ ← Krylov subspace

(II) Note: Krylov methods usually reduce $\Theta(N^3) \rightarrow \Theta(N^2)$

III) Arnoldi: • systematic construction of an orthonormal basis for successive Krylov subspaces

- computing a Hessenberg matrix

$A Q_n = Q_{n+1} \tilde{H}_n$

$\rightarrow A q_n = h_{n1} q_1 + h_{n2} q_2 + \dots + h_{nn} q_n + h_{n,n+1} q_{n+1}$ ← solve for q_{n+1}

Algorithm: b = anything ; $q_1 = b / \|b\|$

for $n = 1, 2, \dots$

$v = A q_n$

for $k = 1, 2, \dots, n-1$

$h_{nk} = q_k^* v$

$v = v - h_{nk} q_k$

end

$h_{n,n+1} = \|v\|$

$q_{n+1} = v / h_{n,n+1}$

↑ $\Theta(n^2)$

Locating eigenvalues: $x \in K_n \rightarrow x = \sum c_k A^k b = q(A) b$ ← some poly

$P(A) b = A^n b - Q_n y$ for some $y \in \mathbb{C}^n$

characteristic poly of H_n: $\min \|A^n b - Q_n y\|_2$

(IV) Lanczos Iteration : $A = A^T$ or $A = A^*$; $AQ_n = Q_{n+1}\tilde{T}_n$ Algorithm: $\beta_0 = 0,\ \alpha_0 = 0,\ q_1 = b/\|b\|$

$\mathcal{O}(n)$

for $n = 1, 2, \ldots$

$v = Aq_n$

$\alpha_n = q_n^T v$

$v = v - \beta_{n-1}q_{n-1} - \alpha_n q_n$

$\beta_{n+1} = \|v\|$

$q_{n+1} = v/\beta_{n+1}$

(V) GMRES: " approximate $x_* = A^{-1}b$ by $x_n \in K_n$ that minimizes norm of $r_n = b - Ax_n$ "

$$\min \|Ax - b\|_2$$
$$= \min \|AK_n y - b\|_2 \quad ; \ x = K_n y$$
$$= \min \|AQ_n \tilde{y} - b\|_2 \quad ; \ K_n y = Q_n \tilde{y}$$
$$= \min \|Q_{n+1}\tilde{H}_n \tilde{y} - b\|_2 \quad ; \ AQ_n = Q_{n+1}\tilde{H}_n$$
$$= \min \|\tilde{H}_n \tilde{y} - Q_{n+1}^* b\|_2$$
$$= \min \|\tilde{H}_n \tilde{y} - \|b\| \hat{e}_1\|_2$$

Final least squares problem

Algorithm: $q_1 = b/\|b\|$

for $n = 1, 2, \ldots$

[n^{th} step of Arnoldi]

$\min \|\tilde{H}_n \tilde{y} - \|b\|\hat{e}_1\|_2$

$x_n = Q_n \tilde{y}$

*each step minimizes the residual over all $x_n \in K_n$ *

(VI) Conjugate Gradients : A must be symmetric positive definite ⟵ minimizes $\|e_n\|_A = \sqrt{e_n^T A e_n}$ at each step

- Algorithm: $x_0 = 0,\ r_0 = p_0 = b$

  for $n = 1, 2, \ldots$

  $\alpha_n = \dfrac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}$ step length

  $x_{n+1} = x_n + \alpha_n p_{n-1}$ approx sol

  $r_n = r_{n-1} - \alpha_n A p_{n-1}$ residual

  $\beta_n = \dfrac{r_n^T r_n}{r_{n-1}^T r_{n-1}}$ improvement of step

  $p_n = r_n + \beta_n p_{n-1}$ search direction

Note: $K_n = \{b, Ab, \ldots, A^{n-1}b\} = \{x_0, \ldots, x_n\} = \{r_0, \ldots, r_{n-1}\}$
$= \{p_0, p_1, \ldots, p_{n-1}\}$

* residuals are orthogonal * $r_n^T r_j = 0 \quad j<n$

* search directions are A-conjugate : $p_n^T A p_j = 0 \quad j<n$

- Optimality: $x = x_n - \Delta x \in K_n \rightarrow e_n = x_* - x_n = x_* + \Delta x$

$$\|e\|_A^2 = (e_n + \Delta x)^T A (e_n + \Delta x) = e_n^T A e_n + (\Delta x)^T A (\Delta x) + \overbrace{2e_n^T A \Delta x}^{0}$$
$$= e_n^T A e_n + (\Delta x)^T A (\Delta x) \quad \text{if } \Delta x = 0 \text{ error is minimized}$$

or

$$\|e_n\|_A^2 = (x_* - x_n)^T A (x_* - x_n) = x_n^T A x_n - 2x_n^T \overbrace{A x_*}^{b} + x_*^T \overbrace{A x_*}^{b} \quad ; \ \emptyset(x) = \tfrac{1}{2}x^T A x - x^T b$$
$$= x_n^T A x_n + x_*^T b - 2x_n^T b$$
$$= 2\emptyset(x) + \text{constant}$$

(VII) Multigrid Methods: (I) Stationary Iterative Methods:

- work if matrix is diagonally dominant: $|a_{ii}| > \sum_j |a_{ij}|$

or

- if $|eig(B)| < 1 \rightarrow \rho(B) = \max\{eig(B)\}$

for solving $Ax = b$

Method 1: Jacobi: $\vec{x}^{n+1} = B\vec{x}^n + \vec{f}$: $B = D^{-1}(L+U)$ $\quad \vec{f} = D^{-1}b$

$A = D - L - U$

Method 2: Gauss Seidel: $\vec{x}^{n+1} = (D-L)^{-1}U\vec{x}^n + (D-L)^{-1}\vec{b}$

Method 3: SOR: $\vec{x}^{n+1} = \omega \vec{x}^{n+1}_{GS} + (1-\omega)\vec{x}^n$

using Gauss Seidel iter

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

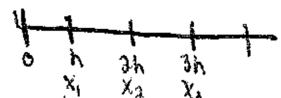
Method 4: Weighted Jacobi: $\vec{x}^{n+1} = \omega \vec{x}^{n+1}_J + (1-\omega)I\vec{x}^n$ $\quad$ * $\omega = 2/3$ is optimal for Jacobi *

(VIII) Fast Matrix Vector Product: $u(x_i) = \int_0^1 G_0(x_i,t)\rho(t)\,dt$ ← how to compute in $\Theta(N)$ time?

ex) $G_0(x,t) = \begin{cases} x(t-1) & x \le t \\ t(x-1) & x \ge t \end{cases}$ $\quad u(x_i) = \int_0^{x_i} \rho(t)\, t(x_i - 1)\,dt + \int_{x_i}^1 \rho\, x_i(t-1)\,dt$

Idea: $\underbrace{\int_0^{x_i+h} t\rho(t)\,dt}_{\phi_{i+1}} = \underbrace{\int_0^{x_i} \rho(t)\,t\,dt}_{\phi_i} + \int_{x_i}^{x_i+h} \rho(t)\,t\,dt$

→ use trapezoid rule



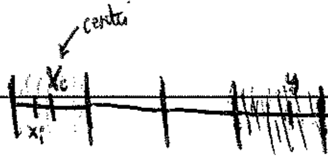
$u(x_1) = (x_1 - 1)\int_0^h t\rho(t)\,dt + \ldots \approx (h-1)\frac{h}{2}\left[h\rho(h)\right]$

$u(x_2) = \frac{(h-1)h}{2}(h\rho(h)) + (2h-1)\int_h^{2h} t\rho(t)\,dt = \frac{(h-1)h}{2}\left[h\rho(h)\right] + \frac{(2h-1)h}{2}\left[2h\rho(2h) + h\rho(h)\right]$

$$\Rightarrow \begin{pmatrix} u(x_1) \\ u(x_2) \\ \vdots \end{pmatrix} = \frac{h^2}{2}\begin{pmatrix} (h-1) & & \\ (3h-2) & 2(2h-1) & \\ \vdots & & \ddots \end{pmatrix}\begin{pmatrix} \rho(h) \\ \rho(2h) \\ \vdots \end{pmatrix}$$

$(h-1) + 2h - 1$

(IX) **Multipole:** low rank approximate



~ far field approx.

$$\phi(y) = \sum_{i=1}^{m} \frac{q_i}{y-x_i} = \sum_{i=1}^{m} \frac{q_i}{y-x_c-(x_i-x_c)}$$

$$= \sum_{i=1}^{m} \frac{q_i}{y-x_c}\left(\frac{1}{1-\frac{x_i-x_c}{y-x_c}}\right)$$

only 21 terms b/c $\left(\frac{x_i-x_c}{y-x_c}\right)^{21} < 10^{-15}$

$$\approx \sum_{i=1}^{m} \frac{q_i}{y-x_c}\Bigg[\underbrace{1 + \frac{x_i-x_c}{y-x_c} + \left(\frac{x_i-x_c}{y-x_c}\right)^2 + \dots + \left(\frac{x_i-x_c}{y-x_c}\right)^{21}}\Bigg]$$

$$M_p = \sum_{i=0}^{21\,n} q_i\,(x_i-x_c)^{1-i}$$

$$\Longrightarrow \quad \phi(y) = \sum_{j=1}^{m} \frac{M_p}{(y-x_i)^p}$$

Steps: (1) Compute M_p ~ $21n$

(2) Evaluate at y_j $j=1,2,\dots,m$ ~ $21m$