AMS526: Numerical Analysis I (Numerical Linear Algebra for Computational and Data Sciences)

Lecture 15: Reduction to Hessenberg and Tridiagonal Forms; Rayleigh Quotient Iteration

Xiangmin Jiao

Stony Brook University

Outline

"Obvious" Algorithms

- Most obvious method is to find roots of characteristic polynomial $p_A(\lambda)$, but it is very ill-conditioned
- Another idea is power iteration, using fact that

$$\frac{x}{\|x\|}, \frac{Ax}{\|Ax\|}, \frac{A^2x}{\|A^2x\|}, \frac{A^3x}{\|A^3x\|}, \dots$$

  converge to an eigenvector corresponding to the largest eigenvalue of A in absolute value, but it may converge very slowly

"Obvious" Algorithms

- Most obvious method is to find roots of characteristic polynomial $p_A(\lambda)$, but it is very ill-conditioned
- Another idea is power iteration, using fact that

$$\frac{x}{\|x\|}, \frac{Ax}{\|Ax\|}, \frac{A^2x}{\|A^2x\|}, \frac{A^3x}{\|A^3x\|}, \ldots$$

  converge to an eigenvector corresponding to the largest eigenvalue of A in absolute value, but it may converge very slowly

- Instead, compute a eigenvalue-revealing factorization, such as Schur factorization

$$A = QTQ^*$$

  by introducing zeros, using algorithms similar to QR factorization

A Fundamental Difficulty

- However, eigenvalue-revealing factorization cannot be done in finite number of steps:

  Any eigenvalue solver must be iterative

- To see this, consider a general polynomial of degree n

$$p(z) = z^n + a_{n-1}z^{n-1} + \cdots + a_1 z + a_0$$

  There is no closed-form expression for roots for $n > 4$:
  In general, the roots of polynomial equations higher than fourth degree cannot be written in terms of a finite number of operations (Abel, 1842)

A Fundamental Difficulty Cont'd

- However, the roots of p_A are the eigenvalues of the *companion matrix*

$$A = \begin{bmatrix} 0 & & & & -a_0 \\ 1 & 0 & & & -a_1 \\ & 1 & \ddots & & \vdots \\ & & \ddots & 0 & -a_{n-2} \\ & & & 1 & -a_{n-1} \end{bmatrix}$$

- Therefore, in general, we cannot find the eigenvalues of a matrix in a finite number of steps
- In practice, however, there are algorithms that converge to desired precision in a few iterations

Schur Factorization and Diagonalization

- Most eigenvalue algorithms compute Schur factorization $A = QTQ^*$ by transforming A with similarity transformations

$$\underbrace{Q_j^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_j}_{Q},$$

  where Q_i are unitary matrices, which converge to T as $j \to \infty$
- Note: Real matrices might need complex Schur forms and eigenvalues
- Question: For Hermitian A, what matrix will the sequence converge to?

Schur Factorization and Diagonalization

- Most eigenvalue algorithms compute Schur factorization $A = QTQ^*$ by transforming A with similarity transformations

$$\underbrace{Q_j^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_j}_{Q},$$

  where Q_i are unitary matrices, which converge to T as $j \to \infty$
- Note: Real matrices might need complex Schur forms and eigenvalues
- Question: For Hermitian A, what matrix will the sequence converge to?

Outline

Two Phases of Eigenvalue Computations

- General A: First convert to *upper-Hessenberg* form, then to upper triangular

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{A \neq A^*} \xrightarrow{\text{Phase 1}} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{upper-Hessenberg}} \xrightarrow{\text{Phase 2}} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}}_{\text{triangular}}$$

- Hermitian A: First convert to *tridiagonal* form, then to diagonal

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{A = A^*} \xrightarrow{\text{Phase 1}} \underbrace{\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{tridiagonal}} \xrightarrow{\text{Phase 2}} \underbrace{\begin{bmatrix} \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix}}_{\text{diagonal}}$$

- In general, phase 1 is direct and requires $O(n^3)$ flops, and phase 2 is iterative and requires $O(n)$ iterations, and $O(n^3)$ flops for non-Hermitian matrices and $O(n^2)$ flops for Hermitian matrices

Introducing Zeros by Similarity Transformations

- First attempt: Compute Schur factorization $A = QTQ^*$ by applying Householder reflectors from both left and right

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1^* \cdot} \underset{Q_1^* A}{\begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}} \xrightarrow{\cdot Q_1} \underset{Q_1^* A Q_1}{\begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}}$$

- Unfortunately, the right multiplication destroys the zeros introduced by Q_1^*
- This would not work because of Abel's theorem
- However, the subdiagonal entries typically decrease in magnitude

The Hessenberg Form

- Second attempt: try to compute upper Hessenberg matrix H similar to A:

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{A} \xrightarrow{Q_1^* \cdot} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & x \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{Q_1^* A} \xrightarrow{\cdot Q_1} \underbrace{\begin{bmatrix} \times & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \times & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{Q_1^* A Q_1}$$

- The zeros introduced by $Q_1^* A$ were not destroyed this time!
- Continue with remaining columns would result in Hessenberg form:

$$\xrightarrow{Q_2^* \cdot} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{Q_2^* Q_1^* A Q_1} \xrightarrow{\cdot Q_2} \underbrace{\begin{bmatrix} \times & \times & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \times & \times & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \times & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{Q_2^* Q_1^* A Q_1 Q_2} \cdots$$

The Hessenberg Form

- After $n-2$ steps, we obtain the Hessenberg form:

$$\underbrace{Q_{n-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{n-2}}_{Q} = H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

- For Hermitian matrix A, H is Hermitian and hence is tridiagonal

Householder Reduction to Hessenberg

Householder Reduction to Hessenberg Form

for $k = 1$ **to** $n - 2$

$$
\begin{aligned}
x &= A_{k+1:n,k} \\
v_k &= \text{sign}(x_1)\|x\|_2 e_1 + x \\
v_k &= v_k / \|v_k\|_2 \\
A_{k+1:n,k:n} &= A_{k+1:n,k:n} - 2v_k(v_k^* A_{k+1:n,k:n}) \\
A_{1:n,k+1:n} &= A_{1:n,k+1:n} - 2(A_{1:n,k+1:n} v_k) v_k^*
\end{aligned}
$$

- Note: Q is never formed explicitly.
- Operation count

$$\sim \sum_{k=1}^{n-2} 4(n-k)^2 + 4n(n-k) \sim 4n^3/3 + 4n^3 - 4n^3/2 = 10n^3/3$$

Reduction to Tridiagonal Form

- If A is Hermitian, then

$$\underbrace{Q_{n-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{n-2}}_{Q} = H = \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \ddots & \ddots & \ddots & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

- For Hermitian A, operation count would be same as Householder QR: $4n^3/3$
  - First, taking advantage of sparsity, cost of applying right reflectors is also $4(n-k)^k$ instead of $4n(n-k)$, so cost is

$$\sim \sum_{k=1}^{n-2} 8(n-k)^2 \sim 8n^3/3$$

  - Second, taking advantage of symmetry, cost is reduced by 50% to $4n^3/3$

Stability of Hessenberg Reduction

Theorem

Householder reduction to Hessenberg form is backward stable, in that

$$\tilde{Q}\tilde{H}\tilde{Q}^* = A + \delta A, \qquad \frac{\|\delta A\|}{\|A\|} = O(\epsilon_{machine})$$

for some $\delta A \in \mathbb{C}^{n\times n}$

Note: Similar to Householder QR, $\tilde{Q}$ is exactly unitary based on some $\tilde{v}_k$

Outline

Solving Eigenvalue Problems

- All eigenvalue solvers must be iterative
- Iterative algorithms have multiple facets:
  1. Basic idea behind the algorithms
  2. Convergence and techniques to speed-up convergence
  3. Efficiency of implementation
  4. Termination criteria
- We will focus on first two aspects

Simplification: Real Symmetric Matrices

- We will consider eigenvalue problems for real symmetric matrices, i.e. $A = A^T \in \mathbb{R}^{n \times n}$, and $Ax = \lambda x$ for $x \in \mathbb{R}^n$
  - Note: $x^* = x^T$, and $\|x\| = \sqrt{x^T x}$
- A has real eigenvalues λ_1,λ_2, ..., λ_n and orthonormal eigenvectors q_1, q_2, ..., q_n, where $\|q_j\| = 1$
- Eigenvalues are often also ordered in a particular way (e.g., ordered from large to small in magnitude)
- In addition, we focus on symmetric tridiagonal form
  - Why? Because phase 1 of two-phase algorithm reduces matrix into tridiagonal form

Rayleigh Quotient

- The Rayleigh quotient of $x \in \mathbb{R}^n$ is the scalar

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

- For an eigenvector x, its Rayleigh quotient is $r(x) = x^T \lambda x / x^T x = \lambda$, the corresponding eigenvalue of x
- For general x, $r(x) = \alpha$ that minimizes $\|Ax - \alpha x\|_2$.
- x is eigenvector of $A \Longleftrightarrow \nabla r(x) = \frac{2}{x^T x}(Ax - r(x)x) = 0$ with $x \neq 0$
- $r(x)$ is smooth and $\nabla r(q_j) = 0$ for any j, and therefore is quadratically accurate:

$$r(x) - r(q_J) = O(\|x - q_J\|^2) \text{ as } x \to q_J \text{ for some } J$$

Power Iteration

- Simple power iteration for largest eigenvalue

Algorithm: Power Iteration
$v^{(0)}$ =some unit-length vector
for $k = 1, 2, \ldots$
$$w = Av^{(k-1)}$$
$$v^{(k)} = w/\|w\|$$
$$\lambda^{(k)} = r(v^{(k)}) = (v^{(k)})^T A v^{(k)}$$

- Termination condition is omitted for simplicity

Convergence of Power Iteration

- Expand initial $v^{(0)}$ in orthonormal eigenvectors q_i, and apply A^k:

$$
\begin{aligned}
v^{(0)} &= a_1 q_1 + a_2 q_2 + \cdots + a_n q_n \\
v^{(k)} &= c_k A^k v^{(0)} \\
&= c_k (a_1 \lambda_1^k q_1 + a_2 \lambda_2^k q_2 + \cdots + a_n \lambda_n^k q_n) \\
&= c_k \lambda_1^k (a_1 q_1 + a_2 (\lambda_2/\lambda_1)^k q_2 + \cdots + a_n (\lambda_n/\lambda_1)^k q_n)
\end{aligned}
$$

- If $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_m| \geq 0$ and $q_1^T v^{(0)} \neq 0$, this gives

$$
\|v^{(k)} - (\pm q_1)\| = O\left(|\lambda_2/\lambda_1|^k\right), \; |\lambda^{(k)} - \lambda_1| = O\left(|\lambda_2/\lambda_1|^{2k}\right)
$$

  where $\pm$ sign is chosen to be sign of $q_1^T v^{(k)}$
- It finds the largest eigenvalue (unless eigenvector is orthogonal to $v^{(0)}$)
- Error reduces by only a constant factor ($\approx |\lambda_2/\lambda_1|$) each step, and very slowly especially when $|\lambda_2| \approx |\lambda_1|$

Inverse Iteration

- Apply power iteration on $(A - \mu I)^{-1}$, with eigenvalues $\{(\lambda_j - \mu)^{-1}\}$
- If $\mu \approx \lambda_J$ for some J, then $(\lambda_J - \mu)^{-1}$ may be far larger than $(\lambda_j - \mu)^{-1}$, $j \neq J$, so power iteration may converge rapidly

> Algorithm: Inverse Iteration
>
> $v^{(0)}$ =some unit-length vector
>
> **for** $k = 1, 2, \ldots$
>
> Solve $(A - \mu I)w = v^{(k-1)}$ for w
>
> $v^{(k)} = w/\|w\|$
>
> $\lambda^{(k)} = r(v^{(k)}) = (v^{(k)})^T A v^{(k)}$

- Converges to eigenvector q_J if parameter μ is close to λ_J

$$\|v^{(k)} - (\pm q_J)\| = O\left(\left|\frac{\mu - \lambda_J}{\mu - \lambda_K}\right|^k\right), \; |\lambda^{(k)} - \lambda_J| = O\left(\left|\frac{\mu - \lambda_J}{\mu - \lambda_K}\right|^{2k}\right)$$

  where λ_J and λ_K are closest and second closest eigenvalues to μ
- Standard method for determining eigenvector given eigenvalue

Rayleigh Quotient Iteration

- Parameter μ is constant in inverse iteration, but convergence is better for μ close to the eigenvalue
- Improvement: At each iteration, set μ to last computed Rayleigh quotient

```
Algorithm: Rayleigh Quotient Iteration
```

$v^{(0)} =$ some unit-length vector
$\lambda^{(0)} = r(v^{(0)}) = (v^{(0)})^T A v^{(0)}$
for $k = 1, 2, \ldots$
  Solve $(A - \lambda^{(k-1)} I) w = v^{(k-1)}$ for w
  $v^{(k)} = w / \|w\|$
  $\lambda^{(k)} = r(v^{(k)}) = (v^{(k)})^T A v^{(k)}$

- Cost per iteration is linear for tridiagonal matrix

Convergence of Rayleigh Quotient Iteration

- Cubic convergence in Rayleigh quotient iteration

$$\|v^{(k+1)} - (\pm q_J)\| = O(\|v^{(k)} - (\pm q_J)\|^3)$$

and

$$|\lambda^{(k+1)} - \lambda_J| = O\left(|\lambda^{(k)} - \lambda_J|^3\right)$$

- In other words, each iteration triples number of digits of accuracy
- Proof idea: If $v^{(k)}$ is close to an eigenvector, $\|v^{(k)} - (\pm q_J)\| \leq \epsilon$, then accuracy of Rayleigh quotient estimate $\lambda^{(k)}$ is $|\lambda^{(k)} - \lambda_J| = O(\epsilon^2)$. One step of inverse iteration then gives

$$\|v^{(k+1)} - q_J\| = O(|\lambda^{(k)} - \lambda_J|\|v^{(k)} - q_J\|) = O(\epsilon^3)$$

- Rayleigh quotient is great in finding largest (or smallest) eigenvalue and its corresponding eigenvector. What if we want to find all eigenvalues?

Operation Counts

In Rayleigh quotient iteration,

- if $A \in \mathbb{R}^{n\times n}$ is full matrix, then solving $(A - \mu I)w = v^{(k-1)}$ may take $O(n^3)$ flops per step
- if $A \in \mathbb{R}^{n\times n}$ is upper Hessenberg, then each step takes $O(n^2)$ flops
- if $A \in \mathbb{R}^{n\times n}$ is tridiagonal, then each step takes $O(n)$ flops