ECE 595, Section 10
Numerical Simulations
Lecture 8: Eigenvalues

Prof. Peter Bermel

January 25, 2013

Outline

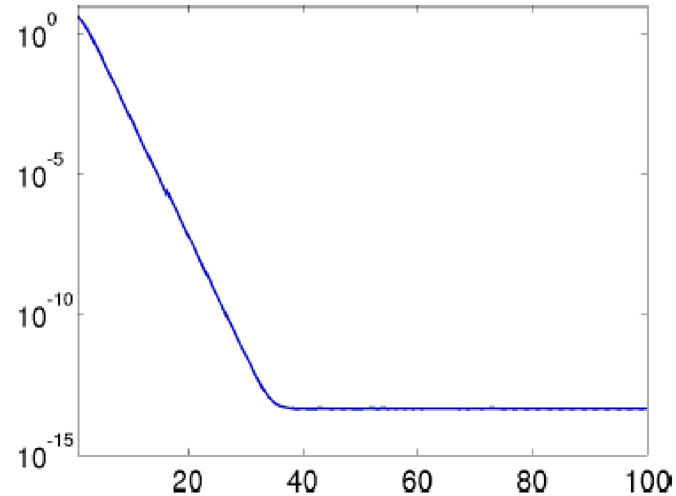
- Recap from Wednesday
- Eigenproblem Solution Techniques
  - Power Methods
  - Inverse Iteration
  - Atomic Transformations
  - Factorization Methods

Recap from Wednesday

- Optimization Methods
  - Brent's Method
  - Golden Section Search
  - Downhill Simplex
  - Conjugate gradient methods
  - Multiple level, single linkage (MLSL)
- Eigenproblems
  - Overview
  - Basic definitions

Power Method

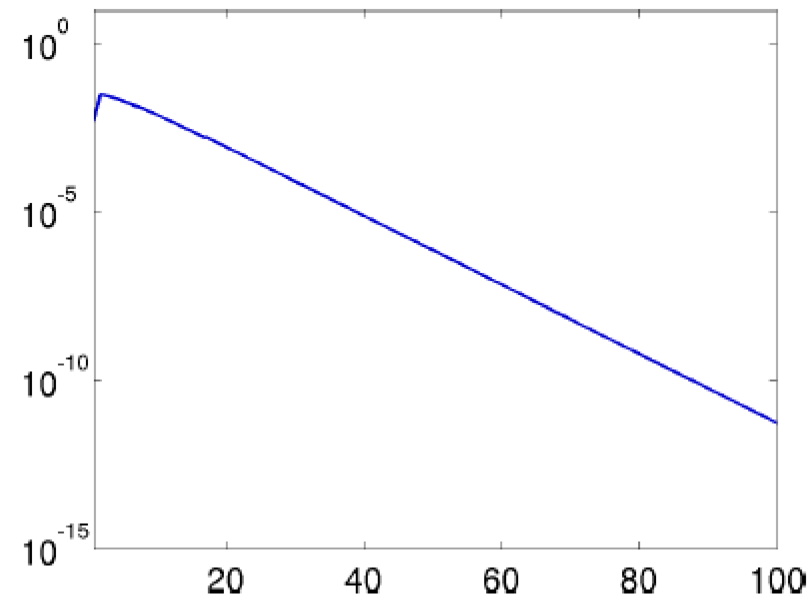
- Algorithm:
  - Initially guess dominant eigenvector v_o
  - Let $v_{k+1} = Av_k$ (optionally: normalize each step)
  - Dominant eigenvalue $\lambda_1 = \frac{{v_k}^T A v_k}{{v_k}^T v_k}$
- To find other eigenvalues:
  - Inverse power method
  - Shifted inverse power method



Error vs. iteration number

Inverse Iteration

- Formalizes concept of converging on a target eigenvalue & eigenvector
- Algorithm:
  - Start with approximate eigenvalue τ and random unit vector b_o
  - Let $x_k = (\boldsymbol{A} - \tau\mathbf{1})^{-1} b_{k-1}$
  - Let $b_k = \frac{x_k}{|x_k|}$
  - Repeat until tolerance reached
  - Our eigenvalue is given by $\lambda = {b_k}^T \cdot \boldsymbol{A} \cdot b_k$



Error vs. iteration number

Inverse Iteration Challenges

- For unlucky b_o, convergence too slow
- For multiple close roots, can only find one eigenvector
- For non-symmetric real matrices, can't find complex conjugate pairs

Transformation Methods

- General concept of similarity transformations: $A \to Z^{-1}AZ$
  - Atomic transformations: construct each Z explicitly
  - Factorization methods: QR and QL methods
- Keep iterating atomic transformations:
  - Until off-diagonal elements are small: then use Z matrix to read off eigenvectors
  - Otherwise: use factorization approach

Jacobi Transformations

- Atomic transformation:

$$\mathbf{P}_{pq} = \begin{bmatrix} 1 & & & & & & \\ & \cdots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \cdots & \\ & & & & & & 1 \end{bmatrix}$$

- Generalizes rotation matrix: $\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$

$$\mathbf{A}' = \mathbf{P}_{pq}^{T} \cdot \mathbf{A} \cdot \mathbf{P}_{pq} \qquad S' = S - 2|a_{pq}|^2$$

Householder Transformation

- Basic approach discussed previously

$$\mathbf{P} = \mathbf{1} - 2\mathbf{w} \cdot \mathbf{w}^T \qquad \mathbf{A}' = \mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P} = \left[\begin{array}{c|cccc} a_{11} & k & 0 & \cdots & 0 \\ \hline k & & & & \\ 0 & & & & \\ \vdots & & & \text{irrelevant} & \\ 0 & & & & \end{array}\right]$$

- Keys to this strategy:
  - Construct vector *w* to eliminate most elements: $w = [0, a_{21} - \alpha, a_{31}, \cdots, a_{N1}]^T$
  - Where $\alpha = -\operatorname{sgn} a_{21} \sqrt{\sum_{j=2}^{N} \left(a_{j1}\right)^2}$
  - Iterate recursively to tridiagonal form and solve: $R \cdot x = \lambda \prod_i {Q_i}^T x$

Factorization in Eigenproblems

- Most common approach known as QR method

  $\mathbf{A} = \mathbf{Q} \cdot \mathbf{R}$ $\quad\quad$ $\mathbf{A}' = \mathbf{R} \cdot \mathbf{Q}$ $\quad\quad$ $\mathbf{A}' = \mathbf{Q}^T \cdot \mathbf{A} \cdot \mathbf{Q}$

- Can also do the same with $\mathbf{A} = \mathbf{Q} \cdot \mathbf{L}$
- QL algorithm:
  – Use Householder algorithm to construct Q_k
  – Factorize: $A_k = Q_k L_k$
  – Rearrange: $A_{k+1} = L_k Q_k = {Q_k}^T A_k Q_k$

QL Algorithm + Implicit Shifts

- Convergence for off-diagonal elements $a_{ij}{}^{(s)} \sim \left(\frac{\lambda_i}{\lambda_j}\right)^s$
- Can be accelerated by shifting $A_k \rightarrow A_k - \beta 1$
- Convergence now goes as $\tilde{a}_{ij}{}^{(s)} \sim \left(\frac{\lambda_i - \beta}{\lambda_j - \beta}\right)^s$

Asymmetric Matrices

- Generally much more sensitive to numerical (round-off) errors
- Balancing with diagonal matrices can relieve this imbalance
- Reduction to Hessenberg form:
  - Series of Householder matrices
  - Gaussian elimination with pivoting

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix}$$

Basic Linear Algebra Subprograms (BLAS)

- Extremely early software package (1979, written originally in FORTRAN)
- Consists of 3 levels:
  – Vector transformations: $\vec{v} \to \vec{v} + \alpha\vec{w}$
  – Matrix-vector operations: $\vec{v} \to \alpha\vec{v} + \beta\overleftrightarrow{A} \cdot \vec{w}$
  – Matrix-matrix operations: $\overleftrightarrow{A} \to \alpha\overleftrightarrow{A} + \beta\overleftrightarrow{B} \cdot \overleftrightarrow{C}$
- Tremendous number of implementations and variations now available

Linear Algebra Package (LAPACK)

- Builds on BLAS to implement many of the linear algebra techniques we discussed in class
  - Linear programming/least squares
  - Matrix decompositions/factorizations
  - Eigenvalues
- Designed in 1992 to deal with special cases efficiently

Matrix type	**full**	**banded**	**packed**	**tridiag**	**generalized problem**
general	ge	gb		gt	gg
symmetric	sy	sb	sp	st	
Hermitian	he	hb	hp		
SPD / HPD	po	pb	pp	pt	
triangular	tr	tb	tp		tg
upper Hessenberg	hs				hg
trapezoidal	tz				
orthogonal	or		op		
unitary	un		up		
diagonal	di				
bidiagonal	bd				

Next Class

- Is on Monday, Jan. 28
- Will discuss numerical tools for simulating eigenproblems further