MS6021 Scientific Computing

Lecture/Work package #3

+

Assignment #3 (20%, due by Wed of Week 9)

TOPICS:

A. **Overdetermined systems**
B. **Eigenvalues**
C. **Quadrature**

A. Overdetermined systems

- Read and work through examples:
  An Introduction to Matlab by David F. Griffiths,
  Section 16.1

- Exercise:
  It is known that the quantities z and t are related by the formula $z = k\, t^{\beta}$,
  for some unknown constants k and β.
  By writing this as $\ln z = \beta \ln t + \ln k$, implement the method of *least squares* to find the best-fit line relating $\ln z$ to $\ln t$ and hence find an approximation of the constants k and β, for the given data points
  $(t; z) = (2, 11), (4, 18), (6,16), (8, 22), (7,20).$

B. Eigenvalues: $Ax=\lambda x$

→ x is an eigenvector of A, and λ is the corresponding eigenvalue

- MatLab built-in functions:
  e=eig(A) ; also [V,D]=eig(A)
  See pp. 132-133 in [Higham & Higham, MatLab Guide]
  (read and work through examples)

eigs(A,k,'smallestabs') (for sparse matrices)
returns the k smallest magnitude eigenvalues.

--NOTE: The eig function can calculate the eigenvalues of sparse matrices that are real and symmetric. To calculate the eigenvectors of a sparse matrix, or to calculate the eigenvalues of a sparse matrix that is not real and symmetric, use the eigs function.
--USE F1 to access MatLab help and learn about this function
(read and work through examples)

→ Part I of Assignment #3

C. Quadrature = Numerical Integration

- Read on Numerical Integration:

  Natalia's MA4002 lecture notes
  http://www.staff.ul.ie/natalia/MA4002/MA42_lec_9_16.pdf
  (see Lectures 13-14)

- Write a MatLab m. function implementing the trapezoidal rule of numerical integration:
  function result = trapezoidal(f, a, b, N) , where

  f = function handle

  [a,b] = interval over which the integral is evaluated

  N = the number of equal subintervals used

  You may consult (preferably afterwards)
  Programming for Computations - MATLAB/Octave
  (Chapter 6 on quadrature+ MatLab implementation)

  NOTE: use **NO LOOPs**

- MatLab built-in functions for quadrature:

  See pp. 171-174 in [Higham & Higham, MatLab Guide]
  (read and work through examples)

→ Part II of Assignment #3

Assignement #3, 20%, by Wed of Week 9

PART I (eigenvalues)

- DEF: Let L be a differential operator combined with certain boundary conditions (BC). A function **v=v(x)** is called its **eigenfunction** if v satisfies the **BC** and **Lv=λv** for some real or complex number **λ**; the latter is called an **eigenvalue** of L.
  NOTE: For differential operators, real eigenvalues are typically described *in* ***ascending order*** **:** $\lambda_1 < \lambda_2 \le \lambda_3 \le \lambda_4 \le \cdots$

- Q1 (**1%**): CONSIDER: Lv=-v'' subject to BC: v(0)=v(1)=0. For this operator,
  - Q1.1: Find all eigenvalues and eigenfunctions of this operator analytically.
    (You may consult a textbook on ODE, or PDEs - sections on the method of separation of variables.) Hint: consider cases **λ=0, <0, >0** separately.
    Note that L has infinitely many eigenvalues!
  - Q1.2: Explain why a discrete version of the above operator L is the matrix **A** from **Ass. #1-part I**, with, if necessary, the following modifications:
    --the matrix should correspond to -v'', not v'' (i.e. note the minus);
    --the columns and rows corresponding to the homogeneous boundary conditions should be eliminated.
    NOTE that the number of eigenvalues equals the size of the matrix (i.e. it's not infinite!)

– Q2 (**1%**): For each of N=40, 80, 160, compute and plot all eigenvalues of the matrix A v the first (N-1) analytical eigenvalues. TIP: use eig

– Q3 (**1%**): Compute and plot the first 30 smallest eigenvalues of the matrix A for N=31, 60, 300, 3000, 30000 v the first 30 analytical eigenvalues. TIP: use eigs.

– Q4 (**3%**): Complete the table:

	$\lvert\lambda_{30} - \lambda^h_{30}\rvert / \lambda_{30}$
N=31	
N=60	
N=300	
N=3000	
N=30000	

where λ_{30} is the analytical eigenvalue #30, while λ^h_{30} is the computational eigenvalue #30.

– Q5: Make conclusions and recommendations on the numerical evaluation of eigenvalues of L.

- Next, consider a different operator (from **Ass. #1-part I**):

  $Lv = -v'' + a(x)\, v$, where $a(x)$ is from Ass #1-II.

  subject to BC: $v(0)=v(1)=0$

  NOTE: for this operator, it is NOT possible to find analytical eigenvalues, but we can and will compute their numerical approximations.

  – Q6 (**1%**): compute and plot the first 30 eigenvalues of the corresponding matrix A for N=31, 60, 300, 3000, 30000.

  – Q7 (**3%**): Complete the table as on the previous page, where λ^h_{30} is the numerical eigenvalue #30 for a given N, BUT λ_{30} is approximated by the numerical eigenvalue #30 for N=300000 (or the largest N your computer is able to process; here the true analytical eigenvalue #30 is no longer available).

  – Q8: Make conclusions and recommendations.

PART II (quadrature)

- Q1: Analytically evaluate the integral

$$\mathrm{I}(a) = \int_0^1 x^a (2 + 7x^2)\, dx \quad \text{for } a > -1$$

- Q2 (**5%**): Evaluate this integral numerically
  - Using your MatLab function trapezoidal (see p.4) with N=10, 100, 10000
  - Using the build-in function quad with tolerance 10^{-2}, 10^{-4}, and 10^{-8}.
  - Compare your numerical integrals I^{num} with the analytical result by completing the values $|I - I^{num}|$ in the table below.
  - What can one conclude from these results?? (You may plot $x^a(2 + 7x^2)$ for these 3 cases to understand/explain the observed behaviour.)

	a = 2.2	a = 0.5	a = -0.9
trapezoidal, N=10			
trapezoidal, N=100			
trapezoidal, N=10000			
quad, tolerance=10^{-2}			
quad, tolerance=10^{-4}			
quad, tolerance=10^{-8}			

- Next consider a more general integral

$$I_g = \int_0^1 x^a g(x)\, dx \quad \text{for } a > -1$$

NOTE: one can NOT evaluate this integral analytically for any function $g(x)$.

- Q3: Implement a tailored version of the trapezoidal rule for the above integral:

$$I_g \approx \sum_i \int_{x_{i-1}}^{x_i} x^a g^I(x)\, dx$$

where $g^I(x)$ is the linear interpolant of $g(x)$ on the interval (x_{i-1}, x_i).
(Note that for a = 1, one gets the standard trapezoidal rule.)

function result = tailored (g, alpha, N) (NO LOOPs)

- Q4 (**2.5%**): : Add and complete 3 lines in the table on the previous page using tailored, for N=10, N=100, N=10000, and **g(x)=** $2 + 7x^2$

- Q5 (**2.5%**): Employ tailored to compute approximate values of I_g for **g(x)=cos(3+x²),** a **= -0.9**, with N=10, 100, 1000, 10000, 100000.

- Q6: Comment on the results that you have obtained.