MOCK EXAM FOR NUMERICAL PROGRAMMING I

LECTURER: C. LASSER

1. Basics: Matlab routines

Please provide short explaining sentences in the style of Matlab's help for the following built-in routines:

(1) `eps`
(2) `fft(x)`
(3) `trapz(y)`
(4) `randn`
(5) `qr(A) %A is a real mxn matrix`

Solution.

(1) distance from 1.0 to the next largest double precision number, that is, `eps = 2^(-52)`
(2) discrete Fourier transform (DFT) of a vector x computed with the fast Fourier transform (FFT) algorithm
(3) approximate integral of y via the trapezoidal method with unit spacing
(4) pseudrandom scalar drawn from the standard normal distribution
(5) QR decomposition of A, that is, an upper triangular matrix $R \in \mathbb{R}^{m\times n}$ and an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ so that $A = QR$

Date: January 8, 2015, examination time is 90 mins.

2. Basics: Numerical tools

What is …? Please provide one to two explaining sentences for each topic.

(1) cubic spline
(2) Gaussian quadrature
(3) condition number of an invertible matrix

Solution.

(1) Let $x_0, \dots, x_n \in [a, b]$ and $y_0, \dots, y_n \in \mathbb{R}$ be given. A cubic spline is a twice continuously differentiable function $s : [a, b] \to \mathbb{R}$ so that $s(x_j) = y_j$ for all $j = 0, \dots, n$ and $s\,|_{[x_j x_{,j+1}]}$ is a cubic polynomial for all $j = 0, \dots, n-1$.
(2) Gaussian quadrature approximates an weighted integral

$$I_w(f) = \int_a^b f(x)w(x)dx \approx \sum_{j=0}^{n} w_j f(x_j) = Q(f).$$

A Gaussian quadrature formula Q uses positive weights $w_0, \dots, w_n > 0$ and nodes $x_0, \dots, x_n \in [a, b]$ so that it is exact when the integrand f is a polynomial of degree $\leq 2n + 1$.
(3) Let $A \in \mathbb{R}^{n\times n}$ invertible and $\| \cdot \|$ a matrix norm on $\mathbb{R}^{n\times n}$. The condition number of A is $\kappa(A) = \|A\| \cdot \|A^{-1}\|$.

3. Understanding

Please say yes or no, and provide one short explanation for your judgement.

(1) Gradual underflow occurs as soon as we work below machine precsion.
(2) Algebraic interpolation on $n = 100$ equidistant nodes is highly accurate.
(3) The summed trapezoidal rule has some connection to spline interpolation.
(4) Back substitution for a $n \times n$ system requires $O(n^3)$ flops.
(5) The conjugate gradient method requires to solve a triangular system at each iteration step.

Solution.

(1) No. Gradual underflow concerns the subnormal numbers, which are much below machine precision.
(2) No. Algebraic interpolation on $n = 100$ equidistant nodes is ill-conditioned. The condition number grows exponentially with n.
(3) Yes. The summed trapezoidal rule computes the integral of an interpolating linear spline.
(4) No. Substitution requires $O(n^2)$ flops.
(5) No. The conjugate gradient method relies on matrix vector multiplications and inner product computations.

4. Numerical programs: reading

What do the following numerical programs compute?

(1) `n = 100; x = 1+randn(n,1); q = sum(x)/n;`

(2)
```
for i=1:n, x(i) = b(i)/L(i,i);
b(i+1:n) = b(i+1:n) - x(i)*L(i+1:n,i); end
```

(3)
```
a = A(:,1); v = a/norm(a) + sign(a(1))*eye(n,1);
H = eye(n) - 2*v*v'/norm(v)^2; B = H*A;
```

(4) `x = (A'*A)\(A'*b);`

(5)
```
D = diag(diag(A)); L = -tril(A,-1); U = triu(A,1);
for k=1:m, x = D\b + D\(L*U)*x; end
```

Solution.

(1) Monte-Carlo approximation to $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} xe^{-(x-1)^2/2}dx = 1$
(2) forward substitution for solving the lower triangular system $Lx = b$
(3) Householder elimination of the subdiagonal entries in the first row of A
(4) solution of the least squares problem min! $\|Ax-b\|_2$ via the normal equation $A^TAx = A^Tb$
(5) m steps of the Jacobi iteration for solving the linear system $Ax = b$

5. Numerical programs: writing

(1) Please provide the $(k+1)$st step of a Newton iteration.
(2) Please write a Matlab program performing n Newton iterations, starting in $x_0 = 2$ for searching a zero of $f(x) = \sin(x)$.
(3) Please explain shortly how to use the QR decomposition for solving least squares problems.
(4) Please write a Matlab program solving a least squares problem via the QR decomposition.

Solution.

(1) The $(k+1)$st step of a Newton iteration is defined as $x_{k+1} = x - f(x_k)/f'(x_k)$.
(2) `x = 2; for k = 1:n, x = x - sin(x)/cos(x); end`
(3) Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$, and $b \in \mathbb{R}^m$. We rewrite

$$\begin{aligned} \|Ax - b\|_2^2 &= \|QRx - b\|_2^2 = \|Rx - Q^T b\|_2^2 \\ &= \|R(1:n,:)x - (Q^T b)(1:n)\|_2^2 + \|(Q^T b)(n+1:m)\|_2^2 \end{aligned}$$

so that the least squares solution is the solution of the upper triangular system $R(1:n;:)x = (Q^T b)(1:n)$.
(4) `[Q,R] = qr(A); b = Q'*b; x = R(1:n,:)\b(1:n);`