University of Waterloo
Department of Computer Science

CS370 Midterm Examination: Fall 2006

Thursday, November 2, 2006 Instructors: Y. Li
Duration = 2 hours

Name .... Student ID ....

Section: 8:30 11:30

The aids allowed are:

- Printed Course Notes
- Lecture notes
- Hand Calculators

There are 7 questions - do all 7.

Question	Mark	Max	Init.
1		8	
2		6	
3		5	
4		8	
5		6	
6		8	
7		9	
Total		50	

1. (8 marks) Answer the following questions and briefly explain. Each of the four questions is worth 2 marks. **Note that you only obtain 2 marks if the answer is correct and the explanation is clear**.

(a) **True or False**. If A is a n-by-n nonsingular matrix, then $\text{cond}(A) = \text{cond}(A^{-1})$. **Briefly explain**.

Solution: Statement is true. Recall that:

$$\begin{aligned} \text{cond}(A) &= \|A\|\|A^{-1}\| \\ \text{cond}(A^{-1}) &= \|A^{-1}\|\|(A^{-1})^{-1}\| \\ &= \|A^{-1}\|\|A\| \quad \text{since } (A^{-1})^{-1} = A \end{aligned}$$

Since multiplication is commutative, we have that $\text{cond}(A) = \text{cond}(A^{-1})$.

(b) **True or False**. In solving a nonsingular system of linear equations, Gaussian elimination with partial pivoting usually yields a small residual even if the matrix is ill-conditioned. **Briefly explain.**

Solution: Statement is true. The computed solution $\hat{x}$ from Gaussian elimination with partial pivoting satisfies

$$(A + \delta A)\hat{x} = b, \qquad \text{where } \|\delta A\| \leq \rho\|A\|\epsilon_{\text{mach}}$$

and ρ is rarely greater than 10 and ϵ_{mach} is the machine epsilon. Thus the residual $r = A\hat{x} - b$ satisfies

$$\|r\| \leq \|\delta A\| \cdot \|\hat{x}\| \leq \rho \cdot \|A\| \cdot \|\hat{x}\| \cdot \epsilon_{\text{mach}}$$

or

$$\frac{\|r\|}{\|\hat{x}\|\|A\|} \leq \rho \cdot \epsilon_{\text{mach}}$$

Hence the statement is true.

(c) It is known that google pagerank is the unique solution to

$$Ax = x, \qquad \sum_{i=1}^{n} x_i = 1, \qquad \text{where } A \text{ is a google matrix.}$$

True or False. The google PageRank can mathematically be determined by solving the least squares problem

$$\begin{pmatrix} A - I \\ e^T \end{pmatrix} x = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Here e is the n-column vector of all ones and 0 is the n-vector of all zeros.

If the answer is no, **briefly explain** why not.

If yes, is computing the PageRank x by solving the least squares problem above a good computational approach when the total number of web pages is very large? **Explain**.

Solution: Statement is true. However, solving the least squares problem above (using either a normal equation approach or a QR factorization method) isn't a very good approach since it requires forming the coefficient matrix

$$\begin{pmatrix} A - I \\ e^T \end{pmatrix}$$

which is now completely dense even when the original connectivity matrix G is very sparse. Note that the google matrix A has strictly positive elements and is completely dense. We can no longer take advantage of the sparsity of the connectivity matrix G.

(d) Consider the following two Matlab computations for a power iteration for Google pagerank. Let $\hat{G} = p * G * D$.

$$(M1) \;\; (\hat{G} + e * z^T) * x, \qquad (M2) \quad \hat{G} * x + e * (z^T * x)$$

Assume that $\hat{G}x$ requires $2(1 - \text{sparsity}(G))n^2$ floating point operations (flops), where sparsity(G) is a percentage number representing the percentage of zero entries in G. Which statement below is true?

(1) Computation (M1) requires more flops than (M2).

(2) Computation (M2) requires more flops than (M1).

(3) Computation (M1) and (M2) require the same number of flops.

Explain your answer by providing the number of flops required in each case.

Solution: This question requires detailed analysis of flops for each method.

In general, (M1) and (M2) require different number of flops. Statement (1) is true when n is large.

Computation (M1) requires:

- n^2 flops for computing $R1 = e * z^T$ (assuming no exploitation of the fact that components of e are ones)
- n^2 flops for $R2 = \hat{G} + R1$
- $n * (2 * n - 1)$ flops for computing $R2 * x$

Thus (M1) requires a total of $4n^2 - n$ flops.

Computation (M2) requires:

- $2n - 1$ flops for computing $\alpha = z^T x$
- n flops for computing $\alpha * e$
- $2(1 - \text{sparsity}(\hat{G}))n^2$ flops for computing $\hat{G} * x$

Thus $(M2)$ requires $2(1 - \text{sparsity}(\hat{G}))n^2 + 3n - 1$ flops. The dominant cost of (M2) is $2(1 - \text{sparsity}(\hat{G}))n^2$, which is always smaller than the dominant cost $4n^2$ of (M1).

2. (6 marks) Consider the floating point number system $F(10, 5, -5, 10)$ with rounding to the nearest rule.

(a) What is the unit rounding error?

Solution: Denote the unit rounding error as E:

$$E = \frac{1}{2} \times 10^{-4} = \frac{1}{2 \times 10^4} = 5 \times 10^{-5}$$

(b) Is the number 1/3 in this FPNS? What is fl(1/3)?

Solution: The exact representation of 1/3 is not included in this FPNS since

$$\text{fl}(1/3) = 0.33333 \times 10^0$$

(c) Show that floating point add $10.000 \oplus 0.0008$ has a relative error no greater than the unit rounding error.

Solution: Carry out the floating point add:

$$\begin{aligned} 10.000 \oplus 0.0008 &= 0.10000 \times 10^2 + 0.80000 \times 10^{-3} \\ &= 0.10001 \times 10^2 \end{aligned}$$

Determine the relative error which:

$$\begin{aligned} \text{Rel. Error} &= \frac{|10.0008 - 10.001|}{10.0008} \\ &= 1.998 \times 10^{-5} < E \end{aligned}$$

Clearly, the relative error is less than the unit rounding error found in (a).

3. (5 marks) Let x be any real number. Give an expression for the solution of the following problem

$$\min_x \| \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} x - \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \|_2$$

Explain your answer.

Solution: We use the least squares fitting to solve the above problem. Note that least squares fitting can be applied since vectors are just matrices of size $m \times 1$. Therefore, we have an overdetermined system and least squares fitting can be used.

Define the following matrices:

$$A = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}; \qquad A^t = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}; \qquad B = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Using least squares fitting, we get the following equations:

$$\begin{aligned} A^t A x &= A^t B \\ m x &= \sum_{i=1}^{m} b_i \\ x &= \frac{1}{m} \sum_{i=1}^{m} b_i \end{aligned}$$

Hence, the solution to the above problem is to take the average of all the b_i.

4. (8 marks) Find numerical values $S'(0)$ and $S'(3)$ for the cubic spline below

$$S(x) = \begin{cases} S_1(x) = 1 + \alpha x + \frac{2}{3}x^2 - \frac{1}{3}x^3 & x \in [0,2] \\ S_2(x) = -7 + 13x - \frac{16}{3}x^2 + \beta x^3 & x \in [2,3] \end{cases}$$

Solution: We first need to determine the values of α and β in $S(x)$. We will do this by solving the equations obtained from the properties of a cubic spline, i.e. continuity of $S(x)$ and $S'(x)$.

- First, $S(x)$ is assumed to be continuous on $[0,3]$ which means:

$$\begin{aligned} S_1(2) &= S_2(2) \\ 1 + 2\alpha + \frac{8}{3} - \frac{8}{3} &= -7 + 26 - \frac{64}{3} + 8\beta \\ 1 + 2\alpha &= -\frac{7}{3} + 8\beta \\ \alpha &= 4\beta - \frac{5}{3} \end{aligned}$$

- The next equation is obtained based on the continuity of $S'(x)$ on $[0,3]$.

$$S'(x) = \begin{cases} S_1'(x) = \alpha + \frac{4}{3}x - x^2 & x \in [0,2] \\ S_2'(x) = 13 - \frac{32}{3}x + 3\beta x^2 & x \in [2,3] \end{cases}$$

  Considering $S_1'(2) = S_2'(2)$, we get:

$$\begin{aligned} S_1'(2) &= S_2'(2) \\ \alpha + \frac{8}{3} - 4 &= 13 - \frac{64}{3} + 12\beta \\ \alpha &= -7 + 12\beta \end{aligned}$$

- Combining the two equations for α, we get:

$$\begin{aligned} 4\beta - \frac{5}{3} &= -7 + 12\beta \\ \beta &= \frac{2}{3} \end{aligned}$$

  and

$$\alpha = 4 \times \frac{2}{3} - \frac{5}{3} = 1$$

  Consequently, we find:

$$S'(x) = \begin{cases} S_1'(x) = 1 + \frac{4}{3}x - x^2 & x \in [0,2] \\ S_2'(x) = 13 - \frac{32}{3}x + 2x^2 & x \in [2,3] \end{cases}$$

which implies:

$$S'(0) = 1 \quad ; \quad S'(3) = 13 - 32 + 18 = -1$$

5. (6 marks) Assume that (x, y, z), where $x, y,\ z$ are each an m-vector, represents sample coordinates of points on a closed 3-dimensional parametric curve $(x(t), y(t), z(t))$. Note that $x_1 = x_m, y_1 = y_m, z_1 = z_m$. Using the index of the coordinates as the parameter t, write a Matlab segment to plot a smooth parametric curve (with a refining factor of 10) which interpolates the given data (x, y, z) using cubic splines. You can assume that the vectors x, y, z have been initialized for you.

Solution: Here is an example of the Matlab code segment:

```
m = length(x);

% Build parameter t
t = 1:m;

% Build splines
x_s = csape(t,x,'periodic');
y_s = csape(t,y,'periodic');
z_s = csape(t,z,'periodic');

% Refine t
t_ref = 1:0.1:m;

% Evaluate spline for refined t
x_ref = ppval(x_s,t_ref);
y_ref = ppval(y_s,t_ref);
z_ref = ppval(z_s,t_ref);

% Plot
plot3(xref, y_ref, z_ref);
```

6. (8 marks)
Find a 3-by-3 permutation matrix P and a lower triangular matrix L with unit diagonal and an upper triangular matrix U such that $PA = LU$ for

$$A = \begin{pmatrix} -2 & 8 & -28 \\ 1 & 8 & 6 \\ 4 & 8 & 24 \end{pmatrix}$$

Solution: Carry out the following row operations:

- Swap rows (1) and (3) to have largest entry in first column on top row:

$$\begin{pmatrix} 4 & 8 & 24 \\ 1 & 8 & 6 \\ -2 & 8 & -28 \end{pmatrix}$$

- We carry out the following row operation: $(3) + 1/2 * (1)$ and obtain:

$$\begin{pmatrix} 4 & 8 & 24 \\ 1 & 8 & 6 \\ 0 & 12 & -16 \end{pmatrix}$$

- Now, carry out the following row operation: $(2) - 1/4 * (1)$ and we obtain:

$$\begin{pmatrix} 4 & 8 & 24 \\ 0 & 6 & 0 \\ 0 & 12 & -16 \end{pmatrix}$$

- Next, swap rows (2) and (3):

$$\begin{pmatrix} 4 & 8 & 24 \\ 0 & 12 & -16 \\ 0 & 6 & 0 \end{pmatrix}$$

- Carry out the following row operation: $(3) - 1/2 * (2)$ and obtain:

$$\begin{pmatrix} 4 & 8 & 24 \\ 0 & 12 & -16 \\ 0 & 0 & 8 \end{pmatrix}$$

Consequently, we have:

$$U = \begin{pmatrix} 4 & 8 & 24 \\ 0 & 12 & -16 \\ 0 & 0 & 8 \end{pmatrix} \qquad P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/4 & 1/2 & 1 \end{pmatrix}$$

7. (9 marks)

Assume that A, B, C and D are given n-by-n matrices and that A is nonsingular. Assume that f, g and h are given n-by-1 vectors.

(a) Write a Matlab fragment that computes n-vectors x, y, and z so that the following equations hold:

$$\begin{aligned} Ax + \; By + \; Cz &= f \\ Ay + \; Dz &= g \\ Az &= h \end{aligned}$$

You may use the **lutx** and **forward** and **backsub** functions as in your assignment, see also appendix. **Efficiency matters**.

Solution: Here is an example of a Matlab code segment:

```
n = length(h);

[ L,U,p] = lutx(A);
w = forward(L,h(p));
z = backward(U,w);

b = g-D*z;
w_2 = forward(L,b(p));
y = backward(U,w_2);

d = f-C*z - B*y;
w_3 = forward(L,d(p));
x = backward(U,w_3);
```

(b) How many flops are required in your computation of x, y, and z? Just provide the dominant term (including an accurate coefficient).

Solution: Work for the following operations:

- LU factorization: $\frac{2}{3}n^3 + \mathcal{O}(n^2)$
- Forward and backward solve: $2n^2 + \mathcal{O}(n)$
- Matrix-vector multiply: $2n^2 - n$

In the code from part (a), the work can be calculated as follows:

$$
\begin{aligned}
\text{Nbr flops} &= (\text{LU fact.}) + 3 \times (\text{For+back solve}) + 3 \times (\text{Mat-vec mult.}) + 3 \times (\text{Sum of vectors}) \\
&= \frac{2}{3}n^3 + \mathcal{O}(n^2) + 3\left[2n^2 + \mathcal{O}(n)\right] + 3[2n^2 - n] + 3n \\
&= \frac{2}{3}n^3 + \mathcal{O}(n^2)
\end{aligned}
$$

Appendix

```
function pp = csape(x,y,conds,valconds)
%CSAPE Cubic spline interpolation with various end-conditions.
%
%   PP  = CSAPE(X,Y)
%
%   returns the cubic spline interpolant (in ppform) to the given
%   data (X,Y) using Lagrange end-conditions (see default in table below).
%
%   PP  = CSAPE(X,Y,CONDS) uses the end-conditions specified in CONDS, with
%   default values (which depend on the particular conditions).
%
%   CONDS may be a *string* whose first character matches one of the
%   following: 'complete' or 'clamped', 'not-a-knot', 'periodic',
%   'second', 'variational', with the following meanings:
%
%   'complete'    : match endslopes (as given in VALCONDS, with
%                   default as under *default*)
%   'not-a-knot'  : make spline C^3 across first and last interior
%                   break (ignoring VALCONDS if given)
%   'periodic'    : match first and second derivatives at first data
%                   point with those at last data point
%                   (ignoring VALCONDS if given)
%   'second'      : match end second derivatives (as given in VALCONDS,
%                   with default [0 0], i.e., as in variational)
%   'variational' : set end second derivatives equal to zero
%                   (ignoring VALCONDS if given)
%   The *default* : match endslopes to the slope of the cubic that
%                   matches the first four data at the respective end.

%-----------------------------------------------------------------%

function output = spline(x,y,xx)
%SPLINE Cubic spline data interpolation.
%   YY = SPLINE(X,Y,XX) uses cubic spline interpolation to find YY, the values
%   of the underlying function Y at the points in the vector XX.  The vector X
%   specifies the points at which the data Y is given.  If Y is a matrix, then
%   the data is taken to be vector-valued and interpolation is performed for
%   each column of Y and YY will be length(XX)-by-size(Y,2).
%
%   PP = SPLINE(X,Y) returns the piecewise polynomial form of the cubic spline
%   interpolant for later use with PPVAL and the spline utility UNMKPP.
%
```

```
%   Ordinarily, the not-a-knot end conditions are used. However, if Y contains
%   two more values than X has entries, then the first and last value in Y are
%   used as the endslopes for the cubic spline.  Namely:
%      f(X) = Y(:,2:end-1),   df(min(X)) = Y(:,1),   df(max(X)) = Y(:,end)

%--------------------------------------------------------------------%

function v=ppval(pp,xx)
%PPVAL  Evaluate piecewise polynomial.
%   V = PPVAL(PP,XX) returns the value at the points XX of the piecewise
%   polynomial contained in PP, as constructed by SPLINE or the spline utility
%   MKPP.
%
%   See also SPLINE, MKPP, UNMKPP.

%--------------------------------------------------------------------%

%PLOT3  Plot lines and points in 3-D space.
%   PLOT3() is a three-dimensional analogue of PLOT().
%
%   PLOT3(x,y,z), where x, y and z are three vectors of the same length,
%   plots a line in 3-space through the points whose coordinates are the
%   elements of x, y and z.

%--------------------------------------------------------------------%

function [L,U,p] = lutx(A)
%LUTX  Triangular factorization, textbook version
%   [L,U,p] = lutx(A) produces a unit lower triangular matrix L,
%   an upper triangular matrix U, and a permutation vector p,
%   so that L*U = A(p,:)

%--------------------------------------------------------------------%

function x = forward(L,x)
% FORWARD. Forward elimination.
% For lower triangular L, x = forward(L,b) solves L*x = b.

%--------------------------------------------------------------------%

function x = backsubs(U,x)
% BACKSUBS.  Back substitution.
% For upper triangular U, x = backsubs(U,b) solves U*x = b.

%--------------------------------------------------------------------%
```

Scrap paper

Scrap paper