CS 322: Prelim 2 Solution Guide

95-100	xxx
90-94	xxx
85-89	xxxxxxxxxxxxxx
80-84	xxxxxxxxxxxx
75-79	xxxxxxxxxxxxxxxxxxxxxxxx
70-74	xxxxxxxxxxxxxxxxxx
65-69	xxxxxxxxxxxxxxxx
60-64	xxxxxxxxxx
55-59	xxxxxxxxxxxxxxxx
50-54	xxxxxxxxxx
45-49	xxxxxxxxxx
40-44	xxxxxxxxxxxx
35-39	xxxxxxxxxx
30-34	xx
< 30	xxxxxxxxxxxx

A = [80,100], B = [55 ,75], C = [40,55]. Median around 66.

1. The MATLAB function `[L,U,P] = LU(A)` computes the factorization $PA = LU$ where L is unit lower triangular, U is upper triangular, and P is a permutation matrix. (A unit lower triangular matrix has ones along its diagonal.)

(a) (10 points) If A is unit lower triangular does it follow that U is the identity matrix? Explain.

No.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ .5 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 0 & -.5 \end{bmatrix}$$

(b) (10 points) What does it mean geometrically if a 2-by-2 linear system is poorly conditioned?

The two lines are nearly parallel.

```
Question 1a:

+10 - Anything that mentions pivoting or that PA is not lower triangular
gets full credit.

+ 5 - NO for the wrong reason (as long as it's not a completely off base
answer (like it depends on whether we compute L or U first.))

+ 2 - L = multipliers, U = parameters.

+ 0 - YES for any reason whatsoever.


Question 1b:

+10 - almost parallel lines, parallel lines, almost the same line, vectors
don't do a good job of spanning the plane, unit circle gets mapped to
long, skinny ellipse by A, intersection of the two lines changes a lot due
to sensitivity arguments

+5 - inconsistent system that cannot be represented geometrically

+3 - nearly singular/singular matrix

+2 - large condition number

+0 - not singular, small condition number, small values in A(1,1),
operations have to be performed efficiently, pivots are in bad spot for
pivoting
```

2. Complete the following function so that it performs as specified

```
  function [a,b] = SinCosFit(x,y)
% x and y are column m-vectors and m is a positive integer with 6<m.
%
% a and b are column 3-vectors with the property that if
%
%   f(z) = a(1)*sin(z) + a(2)*sin(2*z) + a(3)*sin(3*z) +
%          b(1)*cos(z) + b(2)*cos(2*z) + b(3)*cos(3*z)
%
% then (f(x(1)) - y(1))^2 + ... + (f(x(m)) - y(m))^2   is minimized.
```

You may use \. Your implementation should be vectorized.

```
m = length(x);
A = zeros(m,6);
for k=1:3
   A(:,k)   = sin(k*x);
   A(:,k+3) = cos(k*x);
end
c = A\y;
a = c(1:3); b = c(4:6);
```

```
+4  setting up sine entries of coef. matrix
+4  setting up cosine entries of coef. matrix
+4  solving linear system properly using \
+4  setting of a
+4  setting of b

Misc.

-3  if linear system numerator something other than y
-2  if sine/cosine where evaluated at (x-y) rather than x
-2  y\A instead of A\y
-2  if used A'*A stuff
-2  used more than one for-loops
```

3.(a) (10 points) Determine a 4-by-4 lower triangular matrix G so that

$$GG^T = \begin{bmatrix} 4 & 0 & 0 & 6 \\ 0 & 9 & 0 & 15 \\ 0 & 0 & 1 & 7 \\ 6 & 15 & 7 & 100 \end{bmatrix}$$

$$G = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 5 & 7 & \sqrt{17} \end{bmatrix}$$

(b) (10 points) Complete the following function so that it performs as specified

```
function z = Cholnn(A)
% A is an n-by-n symmetric positive definite matrix with the property that
% A(1:n-1,1:n-1) is diagonal.
% z = G(n,n) where G is an n-by-n lower triangular matrix such that A = G*G'.
```

Your solution should vectorized and efficient. You are not allowed to use `chol` or any other function that computes the Cholesky factorization. Note if `C` is a matrix then `diag(C)` is a row vector made up of its diagonal entries.

```
d = diag(A(1:n-1,1:n-1));
v = A(n,1:n-1)./sqrt(d);
z = sqrt(A(n,n)-v*v')
```

```
A:

+2  for each of the first three columns of G
+4  for last column of G

Misc.

-2  G(4,4) wrong but show work
-3  G(4,4) wrong because something else (i.e., G(4,3)) wrong

B:

+3  trying to involve diag
+3  correct compuation of v (or something equiv.)
+4  correct compuation of z (or something equiv.)

Misc.

-2  if z returned as a vector
```

4. Consider the following implementation of the method of bisection:

```
  function root = Bisection(fname,a,b,delta)
% fname is a string that names a continuous function  f(x) of a single variable.
% a and b define an interval [a,b] and f(a)f(b) < 0. delta is a non-negative real.
% root is the midpoint of an interval [alpha,beta] with the property that
% f(alpha)f(beta)<=0 and |beta-alpha| <= delta


while abs(a-b) > delta
   mid = (a+b)/2;
   if feval(fname,a)*feval(fname,mid)<0
      b  = mid;
   else
      a  = mid;
   end
end
root = (a+b)/2;
```

(a) (7pts) Explain why the loop may never terminate. **(b)** (7pts) Even if the loop terminates, the value returned in `root` may not be the midpoint of a bracketing interval with length $\leq$ `delta`. Explain. **(c)** (6 pts) Are there any other flaws in the above implementation? Explain.

(a) If a and b are adjacent floating point numbers and delta is less than their difference then the loop will never end.
(b) If $(a+b)/2$ is the only root in $[a,b]$ and $f(a) > 0$, then b never changes during the iteration and after the first step [a,b] will not be bracketing.
(c) Two f-evaluations per step. Can live with just one.

```
Question 4a:

+3 - any mention of floating point spacing

+5 - if say trouble when delta < eps

+7 - if give full argument, including if discussing when delta = 0


Question 4b:

+2 - showing how to correct the flaw but not explaining what the flaw is.

+7 - any mention of trouble when f(a)*f(b) = 0 gets full credit.


Question 4c:

+2 - Mention of inefficiency without discussing the fevals at all.

+6 - Pretty much all or nothing here.  Full credit for anything OTHER than
the flaws outlined in (a) or (b).  One flaw is the number of fevals in
the for loop.
```

5. Suppose we are given n data points $(x_1, y_1), \ldots, (x_n, y_n)$ and an additional point (h, k). We want to compute the minimum value of

$$\phi(r) = \sum_{i=1}^{n} d_i$$

where d_i is the minimum euclidean distance from (x_i, y_i) to a point on the circle $(x-h)^2 + (y-k)^2 = r^2$. Assume that we are given column n-vectors `x` and `y` that house the data points and scalars `h` and `k`.
(a) (10 pts) Write a `Matlab` script that makes effective use of

```
z = fmin('F',L,R,Options,P1,P2,..) attempts to return a value of z which is a local
  minimizer of F(z,P1,P2) in the interval L < z < R.  'F' is a string
  containing the name of the objective function to be minimized and P1, P2,..
  are its parameters.
```

and assigns to `rBest` a local minimizer of ϕ. (You may ignore `Options` in this problem.) Give a justification for the choice of the search interval endpoints `L` and `R` used by your script. **(b)** (10 pts) Give a complete implementation of the objective function that your script passes to `fmin`. Vectorize and be efficient in both parts of this problem.

```
% Objective function
function mu = f(r,rVals)
mu = sum(abs(rVals-r));

% Script
rVals = sqrt((x-h).^2 + (y-k).^2);
L = min(rVals);
R = max(rVals);
% All the data is in between the circle (x-h)^2 + (y-k)^2 = L^2 and
% the circle (x-h)^2 + (y-k)^2 = R^2.
% The optimum circle must be in between these two circles.
rBest = fmin('f',L,R,0,rVals)
```

```
Part a)
+2 correct usage of fmin
+4 correct L & R values (pretty much anything that correctly bounded rBest
was acceptable, although some things, such as a negative value for L lost
marks)
+2 justification of the values for L and R.
+2 assigning result of fmin to rBest, as per instructions

- Making many calls to fmin resulted in a one point deduction from the
"correct usage" points, even though each call to fmin may have been
syntactically correct.

Part b)
+4 correct implementation of the function
+2 no loops
+2 for general efficiency, i.e. no unneccessary work.
+2 for calculating the distances from (h,k) to all the (x_i, y_i) just
once, and passing this in as a parameter to the objective function.

- Any objective function that had the same minimizer as phi(r) was
considered correct, even if it didn't compute the same value as phi(r).
- Some people had objective functions that wouldn't allow them to take
```

advantage of passing previously calculated results in as parameters. However, this was not sufficient for them to receive those 2 points.