CS 322: Prelim 2 Review Solutions

1. Gaussian elimination with pivoting is used to solve a 2-by-2 system $Ax = b$ on a computer with machine precision 10^{-16}. Suppose

$$A = \begin{bmatrix} .780 & .563 \\ .913 & .659 \end{bmatrix}$$

The inverse of this matrix is

$$A^{-1} = \begin{bmatrix} 659000 & -563000 \\ -913000 & 780000 \end{bmatrix}$$

It is known that the exact solution is given by

$$x = \begin{bmatrix} 1.234567890123456 \\ .0000123456789012 \end{bmatrix}.$$

Underline the digits in x_1 and x_2 that can probably agree with the corresponding digits in the computed solution. Explain the heuristic assumptions used to answer the question.

Solution

The 1-norm condition is about 10^6 since $\| A \|_1 = 1.572$ and $\| A^{-1} \|_1 = 1572000$. So from the heuristic

$$\frac{\| \hat{x} - x \|_1}{\| x \|_1} \approx \text{EPS } \kappa_1(A) \approx 10^{-10}$$

we see that $|\hat{x}_1 - x_1| + |\hat{x}_2 - x_2| \approx 10^{-10}$. Beyond the ninth or tenth decimal place things cannot be trusted in either $\hat{x}_1$ or $\hat{x}_2$.

2. Suppose (x_1, y_1) and (x_2, y_2) are distinct and that $g(x, y)$ is a continuous function. We want to use `fzero` to find t_* such that if

$$\begin{aligned} x_* &= x_1 + t_*(x_2 - x_1) \\ y_* &= y_1 + t_*(y_2 - y_1) \end{aligned}$$

then

$$g(x_*, y_*) = (g(x_1, y_1) + g(x_2, y_2))/2$$

Give an implementation of the function that you would pass to `fzero` and the initial bracketing interval that you would use. Explain why your bracketing interval includes a solution to the problem. You may assume that an implementation `g(x,y)` of the function g is available. Assume also that the points (x_1, y_1) and (x_2, y_2) are represented with the 2-vectors `x` and `y`.

Solution

```
function y = f(t,ave,x,y)
% t a scalar and ave = (g(x(1),y(1))+ g(x(2),y(2))/2
f(t) = g(x(1) + t(x(2)-x(1)),y(1)+t(y(2)-y(1))) - ave
```

[0,1] is a bracketing interval because $f(0) = g(x_1, y_1) - g(x_2, y_2)$ and $f(1) = g(x_2, y_2) - g(x_1, x_2)$. Hence, $f(0)f(1) \leq 0$.

3. The function

```
  function v = fitPlane(x,y,z)
% x,y,z are column n-vectors
% v is a unit column 3-vector with the property that
%
%    |[x(1) y(1) z(1)]'*v|^2 + ... + |[x(n) y(n) z(n)]'*v|^2
%
% is minimized.
```

returns the unit normal of the plane through the origin that best fits the data in the least squares sense. (Recall that $|x_i v_1 + y_i v_2 + z_i v_3|$ is the distance from the point (x_i, y_i, z_i) to the plane.) Implement a function

```
function v = fitGeneralPlane(x,y,z,x0,y0,z0)
```

that returns the unit normal of the plane though (x_0, y_0, z_0) that best fits the data in the least squares sense. Assume that x0, y0, and z0 are scalars. Hint: Does the unit normal change if we translate the data set?

Solution

$|(x_i - x_0)v_1 + (y_i - y_0)v_2 + (z_i - z_0)v_3|$ is the distance from the point (x_i, y_i, z_i) to the plane that passes through (x_0, y_0, z_0) and has unit normal v. It follows that we want to choose v so that sum of the squares of these quantities is minimized. So just translate all the data points so that (x_0, y_0, z_0) corresponds to the origin.

```
function v = fitGeneralPlane(x,y,z,x0,y0,z0)
v = fitPlane(x-x0,y-y0,z-z0);
```

4. One way of fitting the data $(t_1, f_1), \ldots, (t_m, f_m)$ with a polynomial $p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1}$ is to minimize

$$\phi(a) \;=\; \sum_{i=1}^{m}(p(t_i - f_i)^2 \;+\; \mu^2 \sum_{i=1}^{m} p''(t_i)^2$$

where μ is a scalar. The larger the value of μ the less nonlinear will be the optimum fitting polynomial.

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

```
  function a = LScubic(t,f,mu)
% t and f are column n-vectors and and n>=4.
% a is a column 4-vector with the property that if
%
%           p(x) = a(1) + a(2)*x + a(3)*x^2 + a(4)*x^3
%
% then
%      (p(t(1)) - f(1))^2 + ... + (p(t(m)) - f(m))^2 +  mu^2(p''(t(1))^2 + ... + p''(t(m))^2))
%
% is minimized.
```

Hint. Any sum of squares is the square of the 2-norm of some vector r. In this case $r = Ca - g$ where C is $2m$-by-4 and g is $2m$-by-1.

Solution

For $m = 5$ want to minimize the 2-norm of the vector

$$r = \begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \\ 1 & t_4 & t_4^2 & t_4^3 \\ 1 & t_5 & t_5^2 & t_5^3 \\ 0 & 0 & 2\mu & 6t_1\mu \\ 0 & 0 & 2\mu & 6t_2\mu \\ 0 & 0 & 2\mu & 6t_3\mu \\ 0 & 0 & 2\mu & 6t_4\mu \\ 0 & 0 & 2\mu & 6t_5\mu \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} - \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

so in general we do this:

```
m = length(f);
C = [ones(m,1) t t.^2 t.^3 ; zeros(m,1) zeros(m,1) 2*mu*ones(m,1) 6*mu*t];
g = [f ; zeros(m,1)];
a = C\g;
```

5. Suppose `A` is a given n-by-n nonsingular matrix. To compute A's LU factorization we start by adding multiples of the first row to rows 2 through n. The multiples are chosen so as to zero components $(2,1),\ldots,(n,2)$. (a) Write a MATLAB script that does this. (b) Approximately how many flops are required? (c) Explain in English how pivoting changes this step.

Solution
(a)

```
m(2:n) = A(2:n)/A(1,1);
for i =2:n
   A(i,:) = A(i,:) - m(i)*A(1,:);
end
```

(b) Each time through the loop we have a length n vector operation of the form "vector gets vector plus multiple of another vector". Each of these is about $2n$ flops so about $2n^2$ flops altogether. (c) Pivoting addresses the worry that $A(1,1)$ is small or zero. So before this script is executed we go into the A-array and swap row 1 with row q where $a(q,1)$ has the largest absolute value of any entry in `A(:,1)`.

6. Complete the following MATLAB function:

```
 function x = F(d,a,b)
% d is a column n-vector.
% a is a column 4-vector.
% b is a column n-vector.
%
% x is a column n-vector that solves Mx = b where
%
%        M = a(1)*I + a(2)*D + a(3)*D^2 + a(4)*D^{3}
%
% where I is the n-by-n identity matrix and D is the n-by-n diagonal
% matrix with D(i,i) = d(i), i=1:n.
```

Solution

Note that $M = (m_{ij})$ is diagonal and so

$$m_{ii} = a_1 + a_2 d_i + a_3 d_i^2 + a_4 d_i^3$$

Since $Mx = b$ means $x_i = b_i / m_{ii}$, $i = 1{:}n$, we obtain

```
n = length(b); x = zeros(n,1)
for i=1:n
   x(i) = b(i)/(a(1) + a(2)*d(i) + a(3)*d(i)^2 + a(4)*d(i)^3)
end

% or

x = b./(a(1) + a(2)*d + a(3)*d.^2 + a(4)*d.^3);

% or

x = b./( ((a(4)*d + a(3)).*d + a(2)).*d + a(1));
```