CS 322: Practice Final Exam Solution

1. Assume that P, A, q, and T are given column n-vectors that represent “location data” for n separate comets. In particular,

\[ x_i(t) = \frac{P_i - A_i}{2} + \frac{P_i + A_i}{2} \cos \left( \frac{2\pi}{T_i} t + q_i \right) \]

\[ y_i(t) = \sqrt{P_i A_i} \sin \left( \frac{2\pi}{T_i} t + q_i \right) \]

specifies the location of the i-th comet at time t. You may assume that 0 < P_i ≤ A_i and T_i > 0 for i = 1:n. Write a MATLAB script that plots in a single window each of these elliptical orbits. Do not worry about color, axis scaling, labelling, etc. Each plot must be based upon 200 points to ensure a smooth rendition of the orbit. Your script should be vectorized and flop efficient.

15 points
The period T and phase delay q have NOTHING to do with the shape of the orbit. That depends only on the P and A values:

tau = linspace(0,2*pi,200);
c = cos(tau);
s = sin(tau);
hold on
for i=1:length(T)
    x = (P(i)-A(i))/2 + (P(i)+A(i))/2*c;
    y = sqrt(P(i)*A(i))*s;
    plot(x,y)
end
hold off

-10 have plot outside the loop so only one orbit
-5 for linspace(0,T(i),200) inside the loop
-5 for linspace(0,2*pi,200) outside the loop
-2 for linspace(0,max(T)/2*pi,200 outside the loop
-2 for another plotting loop
-5 for trying to use P and A without loops

2. Assume that you are given an n-by-n nonsingular matrix A and an n-by-4 matrix B. Write a MATLAB script that determines a column 4-vector d so that if

\[ p(t) = d_1 + d_2 t + d_3 t^2 + d_4 t^3 \]

then \( p(i) = c_{ii} \) for \( i = 1:n \) where \( C = B^T A^{-1} B \). Make effective use of the \( \backslash \) operator.

20 points
\[ X = A \backslash B; \]
\[ \text{rhs} = \text{zeros}(4,1); \]
\[ \text{for } i = 1:4 \]
\[ \quad \text{rhs}(i) = B(:,i) \ast X(:,i); \]
\[ \text{end} \]
\[ V = [1 1 1 1; 1 2 4 8; 1 3 9 27; 1 4 16 64]; \]
\[ d = V \backslash \text{rhs}; \]

12 points for getting the \text{rhs} (the diagonal of \text{C}) and 8 points for solving \text{Vd} = \text{rhs}.
-5 if you compute all of \text{C}
-7 if you use \text{inv} because \text{inv(\text{A})} \ast \text{z} is about 2-3 times as expensive as \text{A} \backslash \text{z}. And you were asked to use \text{backslash}.

OK to use \text{LU}. Can also get \text{X} \text{vis } X(:,j) = A \backslash B(:,j), \quad j = 1:n.

3. Assume that \text{a}, \text{b}, and \text{T} are given and that \text{F} is an implementation of a function \text{F(x)} that has period \text{T}.
The command
\[ \text{numI} = \text{quad('F', a, b, 0.000001)} \]
assigns to \text{numI} an estimate of
\[ I = \int_a^b F(x)dx \]
that (usually) has absolute error less than 0.000001. Write a \text{MATLAB} script that does the same thing more efficiently assuming that \text{b} = \text{a} + (10/3)\text{T}. You may assume that \text{F}-evaluations are very expensive and that \text{F}
is uniformly behaved from \text{a} to \text{a} + \text{T}. Make sure your solution has an absolute error less than 0.000001.

15 points:
\[ \text{tol} = .000001/7; \]
\[ \text{I1} = \text{quad('F', a, a+T/3, tol)}; \]
\[ \text{I2} = \text{quad('F', a+T/3, a+T, tol)}; \]
\[ \text{I} = 4 \ast \text{I1} + 3 \ast \text{I2}; \]

12 points for this:
\[ \text{tol} = .000001/4; \]
\[ \text{I1} = \text{quad('F', a, a+T, tol)}; \]
\[ \text{I2} = \text{quad('F', a+T/3, a+T, tol)}; \]
\[ \text{I} = 3 \ast \text{I1} + 1 \ast \text{I2}; \]

For either of these, 5 points was for the correct \text{tol} adjustment.

4. Assume that \text{z} and \text{f} are given column 6-vectors with
\[ 1 < z_1 < z_2 < 2 < z_3 < z_4 < 3 < z_5 < z_6 < 4. \]

We wish to determine a column 4-vector \text{y} so that if the continuous piecewise linear function \text{L} is defined by
\[ L(t) = y_k + (t - i)(y_{k+1} - y_k), \quad i \leq t \leq i + 1 \]
then
\[ \phi(y_1, y_2, y_3, y_4) = \sum_{i=1}^{4} (L(z_i) - f_i)^2 \]
is minimized. Write a MATLAB script that determines \( y \) by solving a least squares problem of the form 
\[
\| Ay - f \|_2
\]
where \( A \) is an \( 6 \)-by-\( 4 \) matrix. Make effective use of the \( \backslash \) operator. Do NOT use \texttt{fmins}. Do not worry about vectorization.

15 points
Since
\[
L(t) = (i + 1 - t)y_i + (t - i)y_{i+1} \quad i \leq t \leq i + 1
\]
we see that
\[
A = \text{zeros}(6, 4);
\]
\[
A(1, 1) = 2 - z(1); \quad A(1, 2) = z(1) - 1; \quad \% \text{Set } t = z(1), \ i = 1 \text{ in the above}
\]
\[
A(2, 1) = 2 - z(2); \quad A(2, 2) = z(2) - 1; \quad \% \text{Set } t = z(2), \ i = 1 \text{ in the above}
\]
\[
A(3, 2) = 3 - z(3); \quad A(3, 3) = z(3) - 2; \quad \% \text{Set } t = z(3), \ i = 2 \text{ in the above}
\]
\[
A(4, 2) = 3 - z(4); \quad A(4, 3) = z(4) - 2; \quad \% \text{Set } t = z(4), \ i = 2 \text{ in the above}
\]
\[
A(5, 3) = 4 - z(5); \quad A(5, 4) = z(5) - 3; \quad \% \text{Set } t = z(5), \ i = 3 \text{ in the above}
\]
\[
A(6, 3) = 4 - z(6); \quad A(6, 4) = z(6) - 3; \quad \% \text{Set } t = z(6), \ i = 3 \text{ in the above}
\]
y = Af;
Roughly 2 points per row.

5. The backwards Euler method for the initial value problem \( y' = f(t, y), \ y(t_0) = y_0 \) is defined by
\[
y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1})
\]
where \( t_{n+1} = t_n + h_n \). Write a MATLAB script that makes effective use of this method to produce a plot of \( y(t) \) across the interval \([0, 10]\) where
\[
y'(t) = Ay(t) + U(t) \quad y(0) = y_0
\]
Assume that \( A \) is a given, \( m \)-by-\( m \) matrix and that \( U(t) \) is an implementation of the function \( U \) that returns a column \( m \)-vector for any given scalar \( t \). Assume that the initial vector \( y_0 \) is available. The plot should be based on estimates of \( y(t) \) at \( t = \text{linspace}(0, 10, 201) \). Make effective use of \([L, U, P] = \text{lu}(C)\) that returns the factorization \( PC = LU \). Do not use \texttt{ode23} or any other MATLAB initial value problem solver.

20 points
The step \( (I - h_n A)y_{n+1} = y_n + h_n U(t_{n+1}) \)
\[
h = 10/200;
t = \text{linspace}(0, 10, 201)
\]
\[
[m, m] = \text{size}(A);
\]
\[
C = \text{eye}(m, m) - h*A;
\]
\[
[L, U, P] = \text{lu}(C)
\]
\[
z = \text{zeros}(201, 1);
z(1) = y_0^{'\star}y_0;
\]
\[
\text{for } n=1:200
\]
\[
y_{n+1} = UN\backslash P*(y_0 + h*U(t(n+1)));
z(n+1) = y_{n+1}^{'\star}y_{n+1};
y_0 = y_{n+1};
\]
\[
\text{end}
\]
\[
\text{plot}(t, z);
\]
Getting the matrix is 5 points.
Factoring it outside the loop is 4 points.
z(1) is 3 points
ynext is 5 points
Plotting and z is 3 points.

6. Assume that cubic splines \( S_x, S_y, \) and \( S_z \) interpolate the data \( (x_i, y_i, z_i), i = 1:n \) in the sense that

\[
(S_x(t_i), S_y(t_i), S_z(t_i)) = (x_i, y_i, z_i) \quad i = 1:n
\]

Assume that \( 0 = t_1 < t_2 < \cdots < t_n = 1 \). Let

\[
d(t) = |S_x(t)| + |S_y(t)| + |S_z(t)|.
\]

(a) Write a MATLAB script that computes a scalar \( t \) so that \( d(t) = (d(0) + d(1))/2 \). You must make use of the method of bisection. (Write it from scratch, do not invoke any Chapter 8 function.) The absolute error of your computed \( t \) must be less than or equal to \( \text{tol} \) where \( \text{tol} \) is a given positive scalar. Assume that \( S_x, S_y, \) and \( S_z \) are given implementations of the three splines and that the data is represented in the arrays \( x, y \) and \( z \). Recall that if \( S \) is a cubic spline and \( u \) is a scalar, then \( ppVal(S, u) \) is the value of the spline at \( u \).

(b) What can go wrong if the value of \( \text{tol} \) is too small? Explain.

15 points

d0 = abs(x(1)) + abs(y(1)) + abs(z(1));
n = length(x);
d1 = abs(x(n)) + abs(y(n)) + abs(z(n));
ave = (d0+d1)/2;

L = 0; SL = d0-ave;
R = 0; SR = d1-ave;
while R-L>2*tol
    mid = (R+L)/2;
    Smid = abs(ppVal(Sx,mid)) + abs(ppval(Sy,mid)) + abs(ppval(Sz,mid));
    if Smid*SL<0
        R=mid; SR = Smid;
    else
        L= mid; SL = Smid;
    end
end
tstar = (R+L)/2;

If \( \text{tol} \) is smaller than \( \varepsilon \) the iteration may not terminate because there will eventually be no floating point numbers in between \( L \) and \( R \).

Bisection on the wrong function -5.
Mistakes because you assume \( d \) is monotone increasing up to -8.
3 points for the tol-too-small answer
-4 if more than one function evaluation per iteration