Duke University
Edmund T. Pratt, Jr. School of Engineering

EGR 53L Fall 2009

Test III

Rebecca A. Simmons & Michael R. Gustafson II

Name (please print):_____

NET ID (please print):_____

In keeping with the Community Standard, I have neither provided nor received any assistance on this test. I understand if it is later determined that I gave or received assistance, I will be brought before the Undergraduate Conduct Board and, if found responsible for academic dishonesty or academic contempt, fail the class. I also understand that I am not allowed to speak to anyone except the instructor about any aspect of this test until the instructor announces it is allowed. I understand if it is later determined that I did speak to another person about the test before the instructor said it was allowed, I will be brought before the Undergraduate Conduct Board and, if found responsible for academic dishonesty or academic contempt, fail the class.

Signature:_____

Notes

- You will be turning in each problem in a separate pile. Most of these problems will require working on separate pieces of paper - **Make sure that you do not put work for more than any one problem on any one piece of paper.** For this test, you will be turning in four different sets of work. Again, **Please do not work on multiple problems on the same sheet of paper. Also - please do not put work for one problem on the back of another problem.**
- Be sure your name *and* NET ID show up on *every page* of the test. If you are including work on extra sheets of paper, put your name and NET ID on each and be sure to staple them to the appropriate problem. **Problems without names will incur at least a 25% penalty for the problem**.
- This first page should have your name, NET ID, and signature on it. **It should be stapled on top of and turned in with your submission for Problem I.**
- **You will not need nor can you use a calculator on this test.** For "hand calculations" you will instead show the set-up of the calculation you need to perform but will not reduce it.
- You will be asked to write several lines of code on this test. **Make sure what you write is MATLAB code and not mathematics**. Also be clear if you are writing .* or ./ or .^ versus just * or / or ^ for those operations where there is a distinction.

Name (please print):
Community Standard (print ACPUB ID):

Problem I: [15 pts.] Water Water Everywhere!

Pre-script - for each of the hand-calculations, show the calculations you would need to perform by substituting in the relevant numbers but do *not* simplify. For instance, if you were asked to find the length of the third side of a triangle that had side-angle-side measurements of 1, 80°, and 2, you could write:

$$\text{length} = \sqrt{(1)^2 + (2)^2 - 2(1)(2)\cos(80^\circ)}$$

Part of the Smart Home is a set of two 1000-gallon water storage systems that can collect rainwater to be used for non-potable applications such as watering the yard.

(1) (By Hand) One mechanism to determine the amount of fluid in the tank is to measure the flow rate of water into the tank from the rainwater collection system and the flow rate of water out of the tank through the irrigation system. Given the following table of net flow rate (Q) information - where positive values denote more water coming into the tank:

t, hr	0	2	4	6	8	10
Q, gal/hr	0	11	15	30	-3	-7

(a) Assuming the tank started with 500 gallons at midnight (hour 0), using the most accurate method we discussed in lab, show the calculation you would use to find out how much water is in the tank at 10 AM.

(b) Show the processes to find an equation in the appropriate interval that would be used to estimate the flow rate Q at 3:30 AM using quadratic interpolation. For this interpolation, use the three points in the data set closest to 3:30 AM. Label your estimate $\hat{Q}_Q(3.5)$. Again, you do not need to simplify the equation.

(2) (By Hand) Another mechanism for determining the amount of fluid in the tank would be to measure the fluid pressure at the bottom of the tank and convert the pressure first to a height and then to a volume. Given the following table for volume measurements in the tank - taken on a different day from the examples above:

t, hr	0	2	4	6	8	10
V, gal	600	610	580	560	570	590

(a) Using three point differences, show the calculation you would perform to estimate the rates of change of the tank volume at midnight and at 6 AM.

(b) Using three point differences, show the calculation you would perform to estimate the second derivatives of the tank volume at 4 AM and at 10 AM.

(3) (Programming) Using the fluid volume data from part **(2)** above, show all the MATLAB code needed to calculate an array of estimates for the volume every half-hour between midnight and 10 AM using linear interpolation, polynomial interpolation, and cubic splines. In your code, call the estimate variables `VLIN`, `VP`, and `VCS` respectively. In the case of the cubic spline, assume you have no good information about what the slopes might be at the endpoints. Assume the `clear` command has just been run.

(1) *(a)* There are 6 points, so best bet is to combine a Simpson's 1/3rd with a Simpson's 3/8ths:

$$\int_0^{10} Q(t)\,dt \approx \frac{\Delta t}{3}\left(Q(0)+4Q(2)+Q(4)\right)+\frac{3\Delta t}{8}\left(Q(4)+3Q(6)+3Q(8)+Q(10)\right)$$
$$\approx \frac{2}{3}\left(0+4(11)+(15)\right)+\frac{6}{8}\left((15)+3(20)+3(-3)+(-7)\right)$$

(b) Use values at times 2, 4, and 6 to get a quadratic. Quickest way is to use Newton polynomials:

$$\hat{Q}_Q(3.5) \approx Q(2)+\frac{Q(4)-Q(2)}{4-2}(3.5-2)+\frac{\frac{Q(6)-Q(4)}{6-4}-\frac{Q(4)-Q(2)}{4-2}}{6-2}(3.5-2)(3.5-4)$$
$$\approx (11)+\frac{(15)-(11)}{4-2}(3.5-2)+\frac{\frac{(30)-(15)}{6-4}-\frac{(15)-(11)}{4-2}}{6-2}(3.5-2)(3.5-4)$$

(2) *(a)* Must use exception at first point; can use centered at 6 AM:

$$\left.\frac{dV}{dt}\right|_0 \approx \frac{-V(4)+4V(2)-3V(0)}{2*\Delta t} \qquad \left.\frac{dV}{dt}\right|_6 \approx \frac{V(8)-V(4)}{2*\Delta t}$$
$$\left.\frac{dV}{dt}\right|_0 \approx \frac{-580+4(610)-3(600)}{2*2} \qquad \left.\frac{dV}{dt}\right|_6 \approx \frac{570-580}{2*2}$$

(b) Can use centered at 4 AM; must use exception at last point:

$$\left.\frac{d^2V}{dt^2}\right|_4 \approx \frac{V(6)-2V(4)+V(2)}{\Delta t^2} \qquad \left.\frac{d^2V}{dt^2}\right|_{10} \approx \frac{V(10)-2V(8)+V(6)}{\Delta t^2}$$
$$\left.\frac{d^2V}{dt^2}\right|_4 \approx \frac{560-2(580)+610}{2^2} \qquad \left.\frac{d^2V}{dt^2}\right|_{10} \approx \frac{590-2(570)+560}{2^2}$$

(3)

```
t       = [  0   2   4   6   8  10];
V       = [600 610 580 560 570 590];
tmodel  = 0:.5:10;

VLIN    = interp1(t, V, tmodel, 'linear');
VP      = polyval(polyfit(t, V, length(t)-1), tmodel);
VCS     = interp1(t, V, tmodel, 'spline');
VCSalt  = spline(t, V, tmodel)
```

Name (please print):
Community Standard (print ACPUB ID):

Problem II: [25 pts.] $\sqrt{4} = \cap\cap\cap$

There are several methods for communicating color information to electronic displays. One method - RGB - sends information about the amount of Red, Blue, and Green there should be in an element. Another method - YCbCr - sends the luma (Y), blue-difference (Cb), and red-difference (Cr) values. The luma is essentially the brightness of an element and can be related to the necessary RGB values for a device with:

$$Y = K_r\ (R - G) + K_b\ (B - G) + G$$

where Y is the luma value, R is the red value, G is the green value, B is the blue value, and K_r and K_b are two constants which are related to how exactly that particular device emits light. To find these constants for your computer display, you perform an experiment where you measure the luma value of your display while changing the red and blue values. You keep the green value at 0 the whole time for this experiment, meaning the equation you are actually working with is:

$$Y = K_r\ R + K_b\ B$$

For one experiment, the data set you obtained is:

R	0.0	0.0	0.0	0.5	0.5	0.5	1.0	1.0	1.0
B	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0
Y	0.00	0.04	0.07	0.11	0.14	0.18	0.21	0.25	0.28

Write code that will do the following:

(1) Create the data vectors and make a mesh plot, with contours, of Y as a function of R and B. Your R values should go on the x-axis, B values on the y-axis, and Y values on the z-axis. You do not need to include axis labels or a title, but you should be careful about making sure the proper variable changes along the proper axis.

(2) Calculate an estimate, using spline interpolation, of the Y value when R is 0.25 and B is 0.75. Call this value `Yestimate`.

(3) Calculate the coefficients K_r and K_b that would allow the model equation to best *fit* the data. Your code should store these in variables named `Kr` and `Kb`.

(4) Calculate the coefficient of determination for this model using the coefficients you calculated above. Your code should store this in a variable whose name makes sense.

In your code, put comments to show which part you are working on above. For example, when finding the estimate when R is 0.25 and B is 0.75, you might have

```
% Part (2)
```

```
clear

% Part (1)
%% Since R is the x-axis, should change by *column*
[R, B] = meshgrid(0:.5:1, 0:.5:1);
Y = [0.00 0.11 0.21; 0.04 0.14 0.25; 0.07 0.18 0.28];
meshc(R, B, Y)

% Part (2)
Yestimate = interp2(R, B, Y, 0.25, 0.75, 'spline')

% Part (3) using linear algebra
A = [R(:) B(:)];
MyCoefs = A\Y(:)
Kr = MyCoefs(1)
Kb = MyCoefs(2)

% Part (3) using Sr minimization
Yeqn = @(coefs, R, B) coefs(1)*R + coefs(2)*B;
fSSR = @(coefs, R, B, Y) sum(( Y-Yeqn(coefs, R, B) ).^2);
[MyCoefs, Sr] = fminsearch(@(MyCoefsDummy) fSSR(MyCoefsDummy, R(:), B(:), Y(:)),...
    [1 1])
Kr = MyCoefs(1)
Kb = MyCoefs(2)

% Part (4)
Yhat = Kr * R + Kb * B;
St = sum((Y(:) - mean(Y(:))).^2)
Sr = sum((Y(:) - Yhat(:)).^2)
r2 = (St - Sr) / St
```

Name (please print):
Community Standard (print ACPUB ID):

Problem III: [30 pts.] A data set in need of a tailor

A data set from an experiment is in a text file. The first column represents the independent values (x) and the second column represents the dependent values (y). Your job is to write a program that will load the data then determine the least-squares-fit coefficients for three different models. You will also take *the third model only* and use it to calculate a coefficient of determination and present its predictions graphically. Specifically, your program should:

(1) Load the data from a file called `MyData.txt` and split it up such that there is a variable called x with the independent data and a variable called y for the dependent data

(2) Calculate the best coefficients for a second-order polynomial fit:

$$\hat{y}_1(x) = A + Bx + Cx^2$$

and make sure that your code calculates values for variables named `A`, `B`, and `C`. *Note:* this is all you will be doing with this model.

(3) Calculate the best coefficients for a fit assuming

$$\hat{y}_2(x) = F + Ge^{2x} + H\left(\frac{1}{x}\right)$$

and make sure that your code calculates values for variables named `F`, `G`, and `H`. *Note:* this is all you will be doing with this model.

(4) Calculate the best coefficients for a fit assuming

$$\hat{y}_3(x) = Q\cos(Tx) + R\sin(Tx) + S$$

and make sure that your code calculates values for variables named `Q`, `R`, `S`, and `T`. Note that T shows up twice in the equation.

(5) Calculate the coefficient of determination for $\hat{y}_3(x)$ *only*. Call this variable `COD3`.

(6) Using $\hat{y}_3(x)$ *only* generate predictions for y based on 400 linearly-spaced values for x that have the same domain as the original data. Give this array a name that makes sense.

(7) Produce a graph with the original data points portrayed with black squares and the values predicted by $\hat{y}_3(x)$ *only* as a solid black line. You do not need to add axis labels, a title, or a legend to this graph.

(8) Save the graph as a black-and-white encapsulated PostScript file called `MyPlot.eps`.

In your code, put comments to show which part you are working on above. For example, when finding the coefficients for the polynomial, you might have

```
% Part (2)
```

```
% Part (1)
load MyData.txt
x = MyData(:,1);
y = MyData(:,2);

% Part (2)
MyCoefsABC = polyfit(x, y, 2);
A = MyCoefsABC(3)
B = MyCoefsABC(2)
C = MyCOefsABC(1)

% Part (3)
AA = [x.^0 exp(2*x) 1./x];
MyCoefsFGH = AA\y;
F = MyCoefsFGH(1)
G = MyCoefsFGH(2)
H = MyCoefsFGH(3)

% Part (4)
yeqn = @(coefs, x) coefs(1)*cos(coefs(4)*x)+coefs(2)*sin(coefs(4)*x)+coefs(3);
fSSR = @(coefs, x, y) sum((y - yeqn(coefs, x)).^2);
[MyCoefsQRST, Sr] = fminsearch(@(MyCoefs) fSSR(MyCoefs, x, y), [1 1 1 1]);
Q = MyCoefsQRST(1)
R = MyCoefsQRST(2)
S = MyCoefsQRST(3)
T = MyCoefsQRST(4)

% Part (5)
St   = sum((y - mean(y)).^2)
COD3 = (St - Sr) / St

% Part (6)
xmodel = linspace(min(x), max(x), 400);
ymodel = yeqn(MyCoefsQRST, xmodel);

% Part (7)
plot(x, y, 'ks', xmodel, ymodel, 'k-')

% Part (8)
print -deps MyPlot
```

Name (please print):
Community Standard (print ACPUB ID):

Problem IV: [30 pts.] Presented to you by the Squirrel Nut Zippers

You and your lab partner are running an experiment where, every 100 seconds (and *exactly* every 100 seconds) you take a temperature measurement in Kelvin. For the experiment, you need to know the running integral of the temperature as well as the first and second derivatives of temperature at the most recently obtained data point. Because the data points are spaced apart, you will want to make sure both your integral and derivatives use the most accurate methods we have used in class. Note that your program will *not* know in advance how many data points you will be collecting.

Your program should ask for temperatures and, as long as the value entered is some non-negative number, print out the (most accurate) current value of the total integral as well as the most accurate version of the first and second derivative of the temperature that we covered in class and lab. *Note:* When the first temperature is entered, nothing should be reported. When the second temperature is entered, the first derivative may be calculated but the second derivative should be reported as 0. After receiving the third entry and beyond, the most accurate (i.e. three-point) versions of the derivatives should be used.

If the user inputs a negative number, the program should stop running *without* trying to calculate an integral or derivatives based on that point. When printing items, use integers for the times and the *%g* replacement code for all other numbers. An example run showing the different values your program should print out at each step is below. Note that the numbers 300, 310, 340, 320, 350, and -1 are *examples* that would be entered by the user.

```
Enter first temperature: 300

Enter the next temperature: 310

At time 100:
int(T)    = 30500 K*s
dT/dt     = 0.1 K/s
d^2T/dt^2 = 0 K/s/s

Enter the next temperature: 340

At time 200:
int(T)    = 62666.7 K*s
dT/dt     = 0.4 K/s
d^2T/dt^2 = 0.002 K/s/s

Enter the next temperature: 320

At time 300:
int(T)    = 96375 K*s
dT/dt     = -0.45 K/s
d^2T/dt^2 = -0.005 K/s/s

Enter the next temperature: 350

At time 400:
int(T)    = 128333 K*s
dT/dt     = 0.55 K/s
d^2T/dt^2 = 0.005 K/s/s

Enter the next temperature: -1
```

```
clear
dt = 100;
T(1) = input('Enter first temperature: ');
if T(1)<0 return; end;
T(2) = input('\nEnter next temperature: ');
nT = 2;
while(T(nT)>=0)
    Time = (nT-1)*dt;

    if nT==2
        My1Diff = (T(2)-T(1))/dt;
        My2Diff = 0;
        MyInt = dt/2*(T(1)+T(2));
    else
        My1Diff = (3*T(nT)-4*T(nT-1)+T(nT-2))/2/dt;
        My2Diff = (T(nT)-2*T(nT-1)+T(nT-2)) / dt^2;
        if nT==3
            MyInt = dt/3*(T(1)+4*T(2)+T(3));
        elseif nT==4
            MyInt = 3*dt/8*(T(1)+3*T(2)+3*T(3)+T(4));
        elseif floor(nT/2)~=(nT/2)
            MyInt = dt/3*(T(1)+4*sum(T(2:2:end-1))+2*sum(T(3:2:end-2))+T(end));
        else
            MyInt = dt/3*(T(1)+4*sum(T(2:2:end-4))+2*sum(T(3:2:end-5))+T(end-3))+...
                3*dt/8*(T(end-3)+3*T(end-2)+3*T(end-1)+T(end));
        end
    end

    fprintf('\nAt time %d:\n', Time)
    fprintf('int(T)    = %g K*s\n', MyInt)
    fprintf('dT/dt     = %g K/s\n', My1Diff)
    fprintf('d^2T/dt^2 = %g K/s/s\n\n', My2Diff)

    nT = nT + 1;
    T(nT) = input('Enter next temperature: ');
end
```