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

EGR 53L Fall 2008

Test III

Rebecca A. Simmons & Michael R. Gustafson II

Name (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 Judicial 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 Judicial Board and, if found responsible for academic dishonesty or academic contempt, fail the class.

Signature:_____

Problem I: [15 pts.] Basic Derivatives and Integrals

Assume you have taken a set of eight data points:

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
y	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8

Using the method specified (or, if unspecified, the most precise method discussed in class),
(Note - if evenly spaced, $\Delta x = x_{i+1} - x_i$ for any i)

(a) Assuming the points are evenly spaced, show the equation to calculate $\left.\frac{dy}{dx}\right\|_{x=x_5}$	$\frac{y_6 - y_4}{2\ \Delta x}$
(b) Assuming the points are evenly spaced, show the equation to calculate $\left.\frac{dy}{dx}\right\|_{x=x_8}$	$\frac{y_6 - 4y_7 + 3y_8}{2\ \Delta x}$
(c) Assuming the points are evenly spaced, show the equation to calculate $\left.\frac{d^2y}{dx^2}\right\|_{x=x_1}$	$\frac{y_3 - 2y_2 + y_1}{(\Delta x)^2}$
(d) Assuming the points are evenly spaced, show the equation to calculate $\left.\frac{d^2y}{dx^2}\right\|_{x=x_6}$	$\frac{y_7 - 2y_6 + y_5}{(\Delta x)^2}$
(e) Assuming the points are evenly spaced, show the equation to calculate $\int_{x_1}^{x_8} y(\chi)\ d\chi$	$\frac{\Delta x}{3}(y_1 + 4y_2 + 2y_3 + 4y_4 + y_5) + \frac{3\Delta x}{8}(y_5 + 3y_6 + 3y_7 + y_8)$
(f) Show the equation to calculate $\left.\frac{dy}{dx}\right\|_{x=x_4}$ using two-point forward differences	$\frac{y_5 - y_4}{x_5 - x_4}$
(g) Show the equation to calculate $\left.\frac{dy}{dx}\right\|_{x=x_4}$ using two-point backward differences	$\frac{y_4 - y_3}{x_4 - x_3}$

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

Problem II: [20 pts.] An Engineering Career In Modeling

Gerald Recktenwald's book **Numerical Methods with MATLAB: Implementation and Application** has a problem with data from a book by Eckert and Drake (**Analysis of Heat and Mass Transfer**) that relates the surface tension of water (σ) to the temperature of the water (T). A part of this table is shown below:

T (°C)	0	10	20	30	40	50
σ (mN/m)	78.2	74.8	73.4	71.6	70.2	68.5

Your job is to use MATLAB with three different models to estimate the surface tension when T is 25 °C by determining the best set of coefficients for each model then calculating the estimate with those coefficients. The models to be considered are:

$$\text{quadratic: } \sigma = f + gT + hT^2 \tag{1}$$

$$\text{exponential: } \sigma = me^{nT} \tag{2}$$

$$\text{bi-exponential: } \sigma = qe^{-(T/30)} + re^{-(T/50)} + s \tag{3}$$

Write the entire MATLAB code that you could use to determine the coefficients f, g, h, m, n, q, r, and s as well as the estimate for $\sigma(T)$ when T=25 °C for each model. When your code is done, there must be variables that are clearly named as each of the different coefficients (i.e. a variable named `f`, a variable named `g`, etc) as well as three estimates with variable names that clearly indicate which model was used to calculate the estimate. For the exponential, you *must* use the method that gives you the highest coefficient of determination. For the bi-exponential, you *must* use linear algebra.

```
%% Initial work
clear; format short e
T =     [ 0   10   20   30   40   50  ]'
sigma = [78.2 74.8 73.4 71.6 70.2 68.5]';

%% Get coefficients and estimates for each model
% Quadratic
QCoefs = polyfit(T, sigma, 2)
f = QCoefs(3), g = QCoefs(2), h = QCoefs(1)
QEst = polyval(QCoefs, 25)
% Exponential - use nonlinear to get highest r^2
ExpEqn = @(coefs, T) coefs(1)*exp(coefs(2)*T)
fSSR = @(coefs, T, sigma) sum((sigma - ExpEqn(coefs, T)).^2)
ExpCoefs = fminsearch(@(coefs) fSSR(coefs, T, sigma), [80, -1])
m = ExpCoefs(1), n = ExpCoefs(2)
ExpEst = ExpEqn(ExpCoefs, 25)
% Bi-exponential
BiEqn = @(coefs, T) coefs(1)*exp(-T/30) + coefs(2)*exp(-T/50) + coefs(3)*T.^0
A     =                     [exp(-T/30)            exp(-T/50)            T.^0]
BiCoefs = A\sigma
q = BiCoefs(1), r = BiCoefs(2), s = BiCoefs(3)
BiEst = BiEqn(BiCoefs, 25)
```

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

Problem III: [20 pts.] Hot Or Not?

In Chapter 7 of the Palm book, a problem has a data table of the temperature of water flowing out of a faucet after the hot tap has been turned on. The data below shows the *change* in temperature from time 0:

t (sec)	0	1	2	3	4	5	6	7	8	9	10
T (°F)	0	5.6	13.9	19.8	38.1	39.0	36.8	37.7	38.0	37.4	37.7

- By hand:

(1) Use nearest-neighbor interpolation to estimate the temperature difference when t=5.25 seconds.

$$T_{5.25} \approx T_5 = 39.0$$

(2) Use piecewise-linear interpolation to estimate the temperature difference when t=7.8 seconds.

$$T_{7.8} \approx T_7 + \frac{T_8 - T_7}{8-7}(t-7) = 37.7 + \frac{38 - 37.7}{8-7}(7.8-7) = 37.94$$

(3) Determine a quadratic interpolating equation that gives the temperature change T as a function of the time t for times between 8 sec and 10 sec. Use it to estimate the temperature difference when t=9.5 seconds.

$$\hat{T}(t)\Big|_{8\le t\le 10} = T_8 + \frac{T_9 - T_8}{9-8}(t-8) + \frac{\frac{T_{10}-T_9}{10-9} - \frac{T_9-T_8}{9-8}}{10-8}(t-8)(t-9)$$

$$\hat{T}(t) = 38.0 + \frac{37.4 - 38.0}{1}(t-8) + \frac{\frac{37.7-37.4}{1} - \frac{37.4-38.0}{1}}{2}(t-8)(t-9)$$

$$\hat{T}(t) = 38.0 + (-0.6)(t-8) + (0.45)(t-8)(t-9) = 0.45t^2 - 8.25t + 75.2$$

$$\hat{T}(9.5) = 37.438$$

- Using MATLAB:
  (a) Determine the coefficients of a quadratic interpolating polynomial that gives the temperature change T as a function of the time t for times between 0 sec and 2 sec. Call this coefficient matrix `Coefs02quad`. Use these coefficients to estimate the temperature difference when t=1.8 seconds and call this `TempAt1p8`
  (b) Also, use this interpolation to find the time at which the temperature difference is 10 °F. Call this `time10`.
  (c) Since the temperature difference starts at 0 and grows asymptotically towards a constant, a saturation-growth model might be an accurate model for the temperature difference:

$$\hat{T}(t) = \frac{T_{\max}\, t}{k + t}$$

  Write the MATLAB code you would use to find $T_{\max}$ and k using linear regression with a transformation of variables.

```
%% Initial work
clear; format short e
t = [0   1    2    3    4    5    6    7    8    9   10]';
T = [0.0 5.6 13.9 19.8 38.1 39.0 36.8 37.7 38.0 37.4 37.7]';

%% (a)
Coefs02quad = polyfit(t(1:3), T(1:3), 2)
TempAt1p8   = polyval(Coefs02quad, 1.8)

%% (b)
time10 = fzero(@(tD) polyval(Coefs02quad, tD)-10, [0 2])

%% (c)
P = polyfit(1./t(2:end), 1./T(2:end), 1)
Tmax = 1/P(2)
k    = P(1)/P(2)
```

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

Problem IV: [25 pts.] How Many Points Must An Integral Use Before We Increase The Span?

In order to save money, researchers working on signal processing want to see how fast they have to sample data in order to get accurate integrals using the Trapezoidal Rule. To do this, they model the signal they are measuring as:

$$f(t) = \cos(100t) + \cos(200t) + \cos(400t)$$

The function of the integral, taken as the integration from time 0 to time t, is therefore:

$$F(t) = \int_0^t f(\tau)\ d\tau = \frac{1}{100}\sin(100t) + \frac{1}{200}\sin(200t) + \frac{1}{400}\sin(400t)$$

They want you to write a program that does the following:

(a) Asks the user for a starting number of data points to consider. This number must be an integer 2 or greater. Your program should make sure it is both an integer and 2 or greater; should either test fail, your program must keep asking until a valid number is entered.

(b) Asks the user for a maximum possible number of points to consider. This number must be an integer larger than the number entered above. Should either test fail, your program must keep asking until a valid number is entered.

(c) Your program should then use the smallest number of points specified above and create a time base using that many points over the span of one second. It should then calculate the *analytical* value of the integral at *each* of those points using the function above as well as the *estimated* values of the integral at those same points using the Trapezoidal Rule. It should calculate a coefficient of determination to see how well the estimates match the actual values at those points.

(d) If the coefficient of determination is above 0.99, the program should state that it successfully calculated the required number of points, print out that number and the value of the coefficient of determination, and end. Otherwise, it should increase the number of points and try again. It should continue this until it has found a number of points that produces the required coefficient of determination or it exceeds the maximum number of points specified by the user. In the latter case, the program should print a message stating `Required accuracy not reached` and then end.

Here are some sample runs of the program:

```
Positive integer: -2
Integer greater than 1, PLEASE: 10
Larger positive integer: 8
Integer greater than previous integer, PLEASE: -2
Integer greater than previous integer, PLEASE: 20
Required accuracy not reached
```

```
Positive integer: 100
Larger positive integer: 500
Accuracy reached!
Coefficient of determination is 0.990068 when using 190 points
```

```
%% (a)
Nlow = input('Positive integer: ');
while Nlow<2 | (floor(Nlow)~=Nlow)
    Nlow = input('Integer greater than 1, PLEASE: ');
end
%% (b)
Nhi  = input('Larger positive integer: ');
while Nhi<Nlow | (floor(Nhi)~=Nhi)
    Nhi = input('Integer greater than previous integer, PLEASE: ');
end

%% (c) and (d)
N = Nlow;
r2 = 0;

while N<(Nhi+1) & r2<0.99
    t = linspace(0, 1, N);
    f = cos(100*t)+cos(200*t)+cos(400*t);
    Festimate = cumtrapz(t, f);
    Factual   = 1/100*sin(100*t)+1/200*sin(200*t)+1/400*sin(400*t);
    St = sum((Factual - mean(Factual)).^2);
    Sr = sum((Factual - Festimate).^2);
    r2 = (St - Sr) / St;
    N = N + 1;
end

if r2>0.99
    fprintf('Accuracy reached!\n')
    fprintf('Coefficient of determination is %f when using %d points\n', r2, N-1)
else
    fprintf('Required accuracy not reached\n')
end
```

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

Problem V: [20 pts.] Déjà Vu Again

A couple researchers take data of fluid flow in a concrete pipe given various pipe diameters and slopes and come up with the following data[1]

Diameter (D, m)	0.3	0.6	0.9	0.3	0.6	0.9	0.3	0.6	0.9
Slope (S, m/m)	0.001	0.001	0.001	0.010	0.010	0.010	0.050	0.050	0.050
Flow (Q, m^3/s)	0.04	0.24	0.69	0.13	0.82	2.38	0.31	1.95	5.66

One of their graduate students comes up with the following model for...wait...wrong test. Instead:

(a) Show the MATLAB code you would use to make an interpolated surface plot, with contours, of the flow as a function of the diameter and the slope. You should have 20 points in the D direction and 30 points in the S direction. Use ~~piecewise cubic Hermite~~ spline[2] interpolation to generate the values for the surface. You should also have axis labels, but do not need a title.

(b) The more accurate model from Test II, using powers, can be generalized to

$$Q = aD^bS^c$$

This is a nonlinear fit, but could be linearized using natural logs:

$$\ln(Q) = \ln(a) + b\ln(D) + c\ln(S)$$

Show all the MATLAB code you would use to determine a, b, and c.

(c) Show the MATLAB code to determine the coefficient of determination for this model given those coefficients.

(d) Finally, using MATLAB, you might determine that:

$$S_r = 3.6037e-03 \qquad\qquad S_t = 2.6231e+01$$

Was this model any good?

Here is a start to the codes from above:

```
clear; format short e
D =  [0.300, 0.600, 0.900; 0.300, 0.600, 0.900; 0.300, 0.600, 0.900]
S =  [0.001, 0.001, 0.001; 0.010, 0.010, 0.010; 0.050, 0.050, 0.050]
Q =  [0.040, 0.240, 0.690; 0.130, 0.820, 2.380; 0.310, 1.950, 5.660]
```

[1] Applied Numerical Methods with MATLAB, p. 233. Steven Chapra. McGraw-Hill, 2005

[2] turns out that 'pchip' is not a valid option for interp2; credit will give given on the test, of course

```
%% (a)
[Dm, Sm] = meshgrid(linspace(min(D(:)), max(D(:)), 20), ...
                    linspace(min(S(:)), max(S(:)), 30));
Qm = interp2(D, S, Q, Dm, Sm, 'spline');
surfc(Dm, Sm, Qm)
xlabel('D'), ylabel('S'), zlabel('Q')

%% (b)
%%% Nonlinear version:
Qeqn = @(coefs, D, S) coefs(1).*(D.^coefs(2)).*(S.^(coefs(3)));
fSSR = @(coefs, D, S, Q) sum(sum((Q-Qeqn(coefs, D, S)).^2))
NonlinCoefs = fminsearch(@(coefs) fSSR(coefs, D, S, Q), [1, 1, 1])
a = NonlinCoefs(1), b = NonlinCoefs(2), c = NonlinCoefs(3)

%%% Linear version:
Qeqn = @(coefs, D, S) coefs(1).*(D.^coefs(2)).*(S.^(coefs(3)));
A = [D(:).^0 log(D(:)) log(S(:))]
LinCoefs = A\log(Q(:))
a = exp(LinCoefs(1)), b = LinCoefs(2), c = LinCoefs(3)

%% (c) Depends on method used
Qhat = Qeqn([a b c], D, S) % Can be NonlinCoefs instead of [a b c]
St = sum((Q(:)-mean(Q(:))).^2)
Sr = sum((Q(:)-Qhat(:)).^2)
r2 = (St - Sr) / St
```