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

EGR 53L Fall 2004

Test II

Lianne Cartee
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: [10 pts.] Root Finding Using Matlab

Demonstrate your ability to find a root using Matlab by finding a root of $y(x) = x^3 + 2x + 4$ two different ways. You may assume that there is a function called `HappyQuadratic.m` that contains the following code:

```
function out=HappyQuadratic(in)
out = in.^3 + 2*in + 4;
```

A graph of the function shows that the one real root is near $x = -1$.

(1) Use Matlab's built-in command for finding the roots of polynomials. This will find all of them, including complex ones.

MyRoots = roots([1 0 2 4]);

(2) Use Matlab's built-in command for finding the zeros of functions. This will only find one of them.

OneZero = fzero('HappyQuadratic', -1);

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

Problem II: [20 pts.] Fixing Code

Make correction to the following code so that it works properly:

```
function root = bisect_with_errors(fun, a, b)
%
% This program attempts to find a root of the equation defined
% in the function fun using the bisection method.  fun accepts only a single
% x value as input.  a and b are points which bracket a root of the
% equation on the x axis.  Find the root that occurs when the difference
% between the right and left x values for the brackets is less than 10*eps
% or the value of the function at the midpoint is less than 5*eps.
% The program should iterate no more than 50 times.
% The program contains errors.  Correct the errors.  You may make
% corrections directly on the code that follows.


fa = feval(~~'~~fun~~'~~, a);

fb = feval (~~'~~fun~~'~~, b);   no quotes: fun is a string

for iterations = 1:50

    diff = b-a;

    mid = a + 0.5.*diff;

    fmid = feval(~~'~~fun~~'~~, mid);

    if ~~(abs(fa-fb)>5*eps | mid>10*eps~~)   abs(fmid)<5*eps | diff<10*eps

        root = ~~fmid;~~ mid

        return;

    end
                  == ==
    if (sign(a) ~= sign(mid))

        a = mid;
        fa = fmid;
    else

        b = mid;
        fb = fmid;
    end

end
```

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

Problem III: [15 pts.] Formatted Output

(1) Given the Matlab output below:

p	i		i	s		3	.	1	4		o	r								3	.	1	4	2
m	o	r	e		p	i		i	s				3	.	1	4	1	5						
o	r												3	.	1	4	1	5	9					

Fill in the blanks in the code as efficiently as possible - the fewer characters you use the more points you get. Use ˆ to indicate any spaces you use.

```
fprintf( 'pi^is^%4.2f^or^%11.3f\n'          , pi, pi)

fprintf( 'more pi^is^%8.4f\n'               , pi)

fprintf( 'or^%17.5f\n'                      , pi)
```

(2) Assuming that, instead of displaying to the screen you want to save the output above to a file called `myfile.dat`, write the code below that would allow you to do that.

```
fid = fopen('myfile.dat', 'w')
fprintf(fid, 'pi^is^%4.2f^or^%11.3f\n', pi, pi)
fprintf(fid, 'more pi^is^%8.4f\n', pi)
fprintf(fid, 'or^%17.5f\n', pi)
fclose(fid)
```

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

Problem IV: [20 pts.] Root Finding

(1) Assuming some function $f(x)$ and some guess x_k, write the equation that models how Newton's method determines the next guess for the root:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

(2) Demonstrate your ability to find a root using Newton's Method by finding a root of $y(x) = x^2 - x - 2$. Use a starting value of 0, a function tolerance of 0.1, an x tolerance of 0.1, and a maximum number of iterations of 6. Explain what value you found for the root and which of the three conditions above made you stop. Also indicate the total number of iterations, the final x tolerance, and the final f tolerance.

$f(x_k) = x^2 - x - 2$

$f'(x_k) = 2x - 1$

k	x_k	$f(x_k)$	$f'(x_k)$	$-f/f'$ Δx_k
1	0	−2	−1	−2
2	−2	4	−5	.8
3	−1.2	.64	−3.4	.1882
4	−1.012	.0354		

Root of $x = -1.012$ found due to f_{TOL} of $.0354 < .1$. Final x_{TOL} of $.1882$

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

Problem V: [15 pts.] Basic Loops

(1) Show the output for the following Matlab code:

```
for k=1:3
     for n=1:k-1
          fprintf('%0.0f %0.0f\n', k, n);
     end
end
```

k n
1 1:0 NOTHING
2 1:1
3 1:2

2 1
3 1
3 2

(2) Show the output for the following Matlab code:

```
k=15;
Count1=0;
while k>10
     k=k-4;
     Count1=Count1+1;
end
fprintf('k:%0.0f Count1:%0.0f', k, Count1);
```

k Count
15 0
11 1
7 2

k:7 Count1:2

(3) Show the output for the following Matlab code:

```
Count2=0;
for m=10:-10
     Count2=Count2+1;
end
fprintf('%0.0f', Count2);
```

does nothing since 10:-10 = [ ]

0

(4) Show the output for the following Matlab code:

```
Count3=0;
Count4=0;
for p=1:3:10
     Count3=Count3+1;
     Count4=Count4+p;
end
fprintf('p:%0.0f Count3:%0.0f Count4:%0.0f', p, Count3, Count4);
```

p	C3	C4
	0	0
1	1	1
4	2	5
7	3	12
10	4	22

p:10 Count3:4 Count4:22

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

Problem VI: [20 pts.] Functions and Defaults

The Nernst potential is the electrical potential generated when a semipermeable membrane separates two solutions of different concentrations and allows only one ion in the solution to flow freely. For a membrane permeable to Na^+, the Nernst potential is given by the following equation:

$$E_{Na} = \frac{-RT}{F} \ln\left(\frac{[Na_i]}{[Na_o]}\right)$$

with the following:

E_{Na}	Na^+ Nernst potential in mV
$R = 8.314$ J/(K·mol)	natural gas constant
$F = 96487$ C/mol	Faraday's constant
T	temperature (in Kelvin)
$[Na_i]$	internal concentration of Na^+ in mM
$[Na_o]$	external concentration of Na^+ in mM

The units given above all work together to produce a potential in mV. Given the above, finish the function named `Nernst` that is started below:

```
function ENA = Nernst(NAI, NAO, T)
% ENA = NERNST(NAI, NAO), for scalar values NAI and NAO, returns the Nernst
%    potential in mV given a natural gas constant of 8.314 J/(K mol),
%    Faraday's constant of 96487 C/mol, a temperature of 300 K, and
%    internal and external concentrations of Na+ in mM given by NAI and
%    NAO respectively.
%
% ENA = NERNST(NAI, NAO, T), for scalar values NAI, NAO, and T, returns the Nernst
%    potential as above only at temperature given by T in Kelvin.
```

```
if nargin < 2
    return
elseif nargin == 2
    T=300;
end

R= 8.314;
F=96487;
E=-R*T/F* log(NAI/NAO);
```