Score 38/40

TEST 1

Course Name and Section: Math 240

Student Name: **Strong A Student**

Test Due Date: April 25th

Problems Completed: 5.7, 5.18, 5.20, 6.17, 6.26, 7.16,

7.23, 7.35

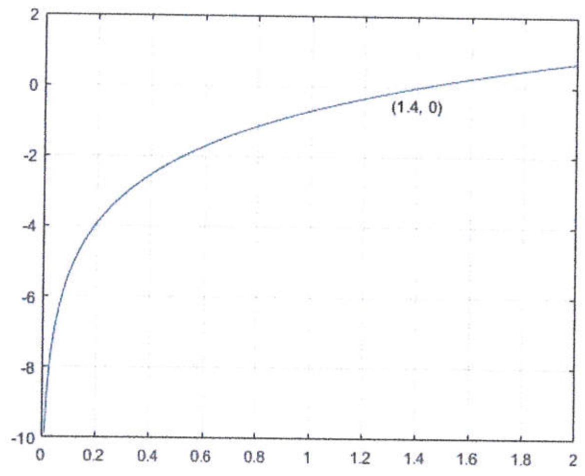
Problems Partially Completed: N/A

Problem 5.7 (a)

I was asked to determine the positive real root of $\ln(x^2) = 0.7$ graphically.

$f(x) = \ln(x^2) - 0.7$

I used MatLab to plot the function on the interval 0 to 2:



From the graph, the root seems to be around $x = 1.4$.

Problem 5.7 (b)

I was asked to determine the roots of the same equation using three iterations of the bisection method, with the initial guesses $x_l = 0.5$ *and* $x_u = 2$.

Here is the script I wrote:

```
l=.5;           %left-hand guess--f(x) is negative
r=2;            %right-hand guess--f(x) is positive

for h=1:3
    m=(l+r)/2;                      %midpoint
    if (log(m.^2)-.7)>0             %f(m) is positive
        r=m;
    else l=m;                       %f(m) is not positive
    end
end
disp(['x = ' num2str(m)])
disp(['f(x) = ' num2str(log(m.^2)-.7)])
```

The output of the script is:

```
x = 1.4375
f(x) = 0.025811
```

Problem 5.7 (c)

I was asked to find the roots of the same equation using three iterations of the false-position method, with the same initial guesses as part (b).

Here is the script:

```
l=.5;           %left-hand guess--f(x) is negative
r=2;            %right-hand guess--f(x) is positive

f1=@(x) log(x.^2)-.7;
for h=1:3
    s=r-(f1(r)*(l-r))/(f1(l)-f1(r));    %secant intersection with x-axis
    if f1(s)>0                          %f(s) is positive
        r=s;
    else l=s;                           %f(s) is not positive
    end
end
disp(['x = ' num2str(s)])
disp(['f(x) = ' num2str(f1(s))])
```

And here was the output:

```
x = 1.4484
f(x) = 0.040917
```

Problem 5.18

I must create a function file that takes a value for "Re" between 2,500 and 1,000,000 as an argument and produces an f value for the equation $\frac{1}{\sqrt{f}} = 4log_{10}(Re\sqrt{f}) - 0.4$, which represents fluid flow in pipes where f is the Fanning friction factor. The absolute error should be less than 0.000005.

This equation can be written as a function $g(f) = 4log_{10}(Re\sqrt{f}) - 0.4 - \frac{1}{\sqrt{f}}$, which I call f1 in MatLab.

Here is my function file:

```
function[m,y,t,e]=Flow(Re)

if Re<2500|Re>1000000
    error('Re must be between 2500 and 1000000')
end


f1=@(f) 4*log10(Re*f^(1/2))-0.4-1/f^1/2;
l=0;                                %left bound--f1(l) is negative
r=.5;                               %right bound--f1(r) is positive
for n=1:1000
    m=(l+r)/2;                      %midpoint
    t(n+1)=m;                       %iterations of m in vector t
    e(n+1)=f1(m);                   %iterations of f1(m) in vector e
    if f1(m)<0                      %bisects interval
        l=m;
        else r=m;
    end
    if abs(r-l)<(0.000005)&abs(f1(m))<0.000005  %checks error
        break
    end
end
y=f1(m);
```

It checks the value of Re to make sure it is between 2500 and 1,000,000, and then runs the bisection method on it, recording each iteration of the f value and the $g(f)$ value in the vectors t and e respectively. It stops when both the interval length and the $g(f)$ value is less than 0.000005, since I wasn't quite sure which one the book was asking for.

To test it, I typed into MatLab:

```
[x,y]=Flow(120000)
```

And got:

```
x = 0.0297

y = 2.5767e-06
```

Problem 5.20

I was asked to solve the equation $v = 1800 * ln\frac{160000}{160000-2600t} - 9.81t$ for t when $v = 750$, with an error of less than 1% of the true value. The equation represents the upward velocity of a rocket.

The solution to this equation corresponds to the roots of the following function, f(x)

$$f(x) = 1800 * ln\frac{160000}{160000 - 2600t} - 9.81t - 750$$

The problem tells me that the root is somewhere between $t = 10$ and $t = 50$.

I used the bisection method. Here is my script:

```
f=@(t) 1800*log(160000/(160000-2600*t))-9.82*t-750;
l=10;                                   %f(l) is negative
r=50;                                   %f(r) is positive
for n=1:1000
    m=(l+r)/2;                          %midpoint
    x(n+1)=m;                           %iterations of m in vector x
    y(n+1)=f(m);                        %iterations of f(m) in vector y
    if f(m)<0                           %bisects interval
        l=m;
    else r=m;
    end
    if abs(r-l)<.01                     %stops when error is <1%
        break
    end
end
disp(['t = ' num2str(m)])
disp(['f(t) = ' num2str(f(m))])
```

And here is my output:

```
t = 26.416
f(t) = 0.074253
```

Problem 6.17 (a)

The equation $y = \frac{T_A}{w} * \cosh\left(\frac{w}{T_A} * x\right) + y_0 - \frac{T_A}{w}$ describes the height y of a cable as a function of the distance x. I was given the values for the parameters $w = 10$ and $y_0 = 5$ and was asked to find a value for the parameter T_A such that $y = 15$ at $x = 50$.

This information leads to the function f(x):

$$f(x) = \frac{T_A}{10} * \cosh\left(\frac{500}{T_A}\right) - \frac{T_A}{10} - 10$$

Plotting the function shows me that the root is between 1,000 and 2,000. I used this as my interval. Again I used bisection. Here's the script:

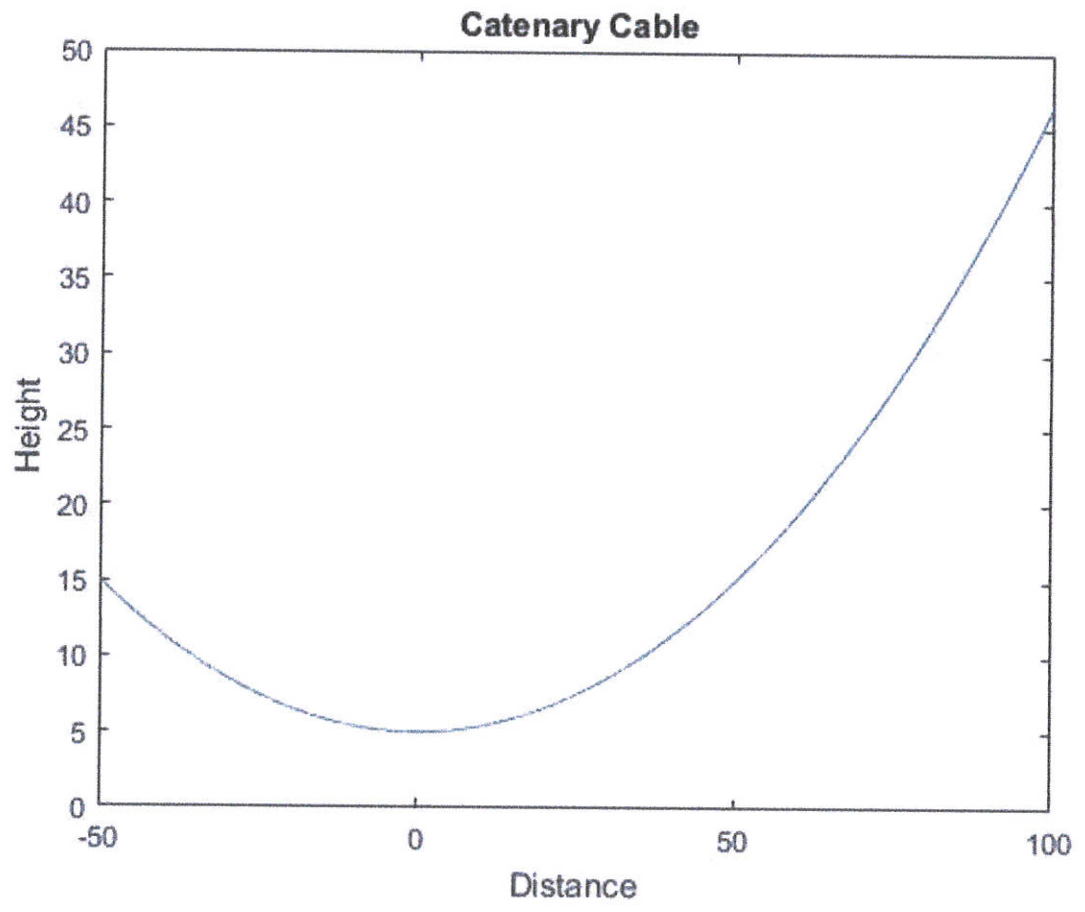
```
f=@(T) T./10.*cosh(500./T)-T./10-10;
l=1000;                                     %f(l) is positive
r=2000;                                     %f(r) is negative
for n=1:1000
    m=(l+r)/2;                              %midpoint
    x(n+1)=m;                               %iterations of m in vector x
    y(n+1)=f(m);                            %iterations of f(m) in vector y
    if f(m)>0                               %bisects interval
        l=m;
    else r=m;
    end
    if abs(r-l)<10^-8                       %stops when error <10^-8
        break
    end
end
disp(['Ta = ' num2str(m)])
disp(['f(Ta) = ' num2str(f(m))])
```

And the output:

```
Ta = 1266.3244
f(Ta) = 2.7171e-11
```

Problem 6.17 (b)

I was asked to, using the same equation and parameters from part a, plot y against x on the interval $x = -50$ to 100.



Problem 6.26

I was asked to create a foolproof function to compute the friction factor f using the parameter values Re from 4,000 to 10^7 and $\frac{\varepsilon}{D}$ from 0.00001 to 0.05 as function inputs.

The equation for the friction factor f is $0 = \frac{1}{f} + 2.0 * \log(\frac{\varepsilon}{3.7D} + \frac{2.51}{Re\sqrt{f}})$

From graphing the function for different parameter values, I determined that, within the stated parameter intervals, the root is always between $f = 0$ and $f = 1$. So I used that as my bisection interval. Here is the script:

```
function[f,g,x,y]=Friction(Re,epD)

if nargin<2                                     %checks number of arguments
    error('Please enter Re from 4000 to 10^7, and e/D from 10^-5 to 0.05')
end
while Re<4000|Re>10^7|epD<10^-5|epD>.05        %checks parameter values
    disp('Re must be from 4000 to 10^7, and e/D from 10^-5 to 0.05')
    Re=input('Enter Re from 4000 to 10^7:  ');
    epD=input('Enter an e/D from 10^-5 to 0.05:  ');
end
f1=@(f) 1./f.^(1/2)+2*log10(epD.*1/3.7+2.51./(Re*f.^(1/2)));

l=0;                                            %f1(l) is positive
r=1;                                            %f1(r) is negative
for n=1:1000
    m=(l+r)/2;                                  %midpoint
    x(n+1)=m;                                   %iterations of m to vector x
    y(n+1)=f1(m);                               %iterations of f1(m) to vector y
    if f1(m)>0                                  %bisects interval
        l=m;
    else r=m;
    end
    if abs(y(n+1))<10^-8                        %stops when f1(m) is less than 10^-8
        break
    end
end
f=m;
g=f1(m);
```

From all the parameter values I have tested this converges after about 33 iterations. For example, when I type in

```
f=Friction(60000,0.002)
```

I get:

```
f = 0.0261
```

Problem 7.16

I was asked to create a function M-file which uses the golden-section search to locate the minimum of a user-supplied function on a user-supplied interval, and rather than using a formula for the size of the error when determining when to stop, instead it should calculate the number of iterations required to attain a desired tolerance.

Here is the script I wrote:

```
function [x,y,xi,yi]=FindMin(func,l,r,err)
    if nargin < 4
        err=.0001;
    elseif l>r
        error('Right bound must be greater than left bound')
    elseif err<=0
        error('error must be positive')
    end

    phi=(1+5^(1/2))/2;              %defines golden ratio
    d=(phi-1)*(r-l);
    g1=(l+d);
    g2=(r-d);
    intrv=r-l;                      %interval length
    base=(1/phi);
    iter=log(err/intrv)/log(base);  %calculates iterations needed
    for n=1:ceil(iter);
        xi(n+1)=(l)/2;              %iterations of x to xi vector
        yi(n+1)=func((l)/2);        %iterations of func(x) to yi vector
    if func(g1)<func(g2)            %cut interval and redefine variables
        l=g2;
        g2=g1;
        g1=l+(phi-1)*(r-l);
    else
        r=g1;
        g1=g2;
        g2=r-(phi-1)*(r-l);
    end
    end
    x=l;
    y=func(x);
```

This one was much more involved, but it's a very similar concept as the bisection problems. Before running the loop, I had to determine the number of iterations to use to get the tolerance. To find this I set up the equation $intrv * \left(\frac{1}{\varphi}\right)^t = err$ where t is the number of iterations, intrv is the length of the initial interval, and err is the maximum error. Solving for t gives $t = \frac{\ln(\frac{err}{interv})}{\ln(\frac{1}{\varphi})}$ which I used to define the variable "iter" in the script. When starting the loop, I set n to loop from 1 to "iter" rounded up to the nearest integer. The loop then uses the golden-section search to narrow in on the minimum.

The problem asked me to test the M-file on the function $f(x) = \frac{x^2}{10} - 2\sin(x)$ with the interval 0 to 4 and a maximum error of 0.0001.

I typed in the command

```
f=@(x) x^2/10-2*sin(x);
[x,y]=FindMin(f,0,4,0.0001)
```

And MatLab returned

```
x = 1.4275

y = -1.7757
```

Problem 7.23

I was asked to use the fminsearch function to determine the minimum of the multivariable function $f(x, y) = 2y^2 - 2.25xy - 1.75y + 1.5x^2$.

I first created a mesh grid and used the surf command to create a surface mesh of the function to get an idea of where the minimum might be. It seemed to be near 0, so I used that as my starting point in the fminsearch function.

The fminsearch function requires the starting point to be a single variable, but that variable can have two entries. So I had to change the format of the function somehow. I ended up creating a second function which takes in a 1x2 vector and plugs them into the first function.

I typed in

```
f=@(x,y) 2*y.^2-2.25*x*y-1.75*y+1.5*x.^2;
f2=@(t) f(t(1),t(2));
[a,b]=fminsearch(f2,[0,0])
```

And got back

```
a = 0.5676    0.7568


b = -0.6622
```

Where the entries of a are the x and y coordinates respectively, and b is the z coordinate.

Problem 7.35

I was asked to come up with an equation to calculate the total potential energy of two springs connected to two frictionless masses in a chain, with a given force of 100 N pulling on the second mass. I was given the parameter values $k_a = 20$ and $k_b = 15$ N/m and the equation for the potential energy a single spring $PE(x) = 0.5kx^2 - Fx$. I was then asked to plot the equation and locate the minimum, or equilibrium, displacements.

Since the equation for a single spring's potential energy is $PE(x) = 0.5kx^2 - Fx$, I started with the sum of two of those (one for each spring), plugged in the parameters, and made adjustments from there.

So I had $PE = 0.5 * 20x^2 - Fx + 0.5 * 15y^2 - 100y$ where x is the displacement of the first mass and y is that of the second.

The first thing that calls for adjustment is that the second spring only "notices it" when y changes relative to x. So I replaced each y with (y-x).

$$PE = 0.5 * 20x^2 - Fx + 0.5 * 15(y - x)^2 - 100(y - x)$$

Next I needed to define the force F pulling on the first spring. The only force pulling directly on the first spring is the second spring, so I used Hooke's law to calculate the force. $F = 15y$. But again, the spring only "cares" about y's displacement relative to x. So F=15*(y-x). I plugged this into the equation and got:

$$PE = 0.5 * 20x^2 - 15(y - x)x + 0.5 * 15(y - x)^2 - 100(y - x)$$

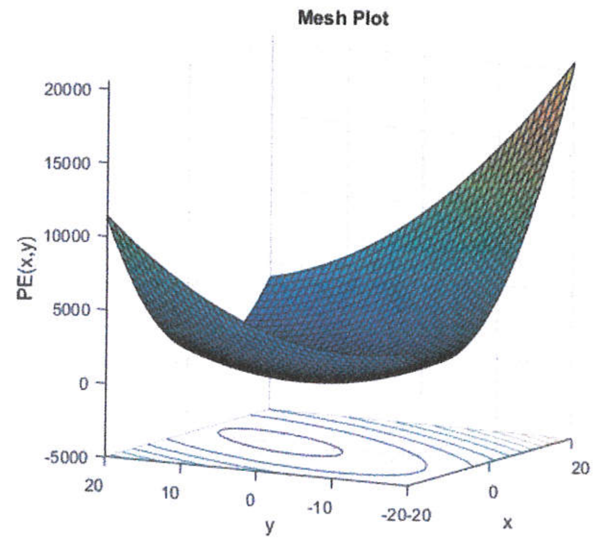
With the following script I made a mesh plot and contour plot, and used the fminsearch command to locate the x y and $PE(x,y)$ coordinates of the equilibrium point:

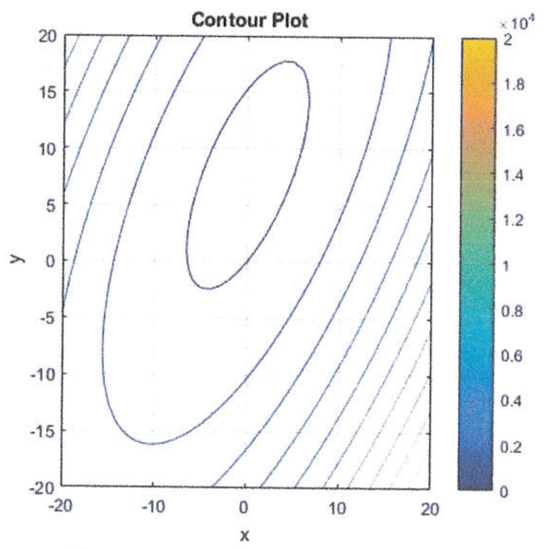
```
PE=@(x,y) 0.5*20*x.^2-15*(y-x)+0.5*15*(y-x).^2-100*(y-x);
[x,y]=meshgrid([-20:1:20]);
for r=1:41
for c=1:41
z(r,c) = PE(x(r,c), y(r,c));
end
end
subplot(1,2,1)
surfc(x,y,z)
xlabel('x')
ylabel('y')
zlabel('PE(x,y)')
title('Mesh Plot')
subplot(1,2,2)
contour(x,y,z)
xlabel('x')
ylabel('y')
title('Contour Plot')
grid on

f2=@(t) PE(t(1),t(2));
[a,b]=fminsearch(f2,[10,10])
```

This produces these values and plots:

```
a = -0.0000    7.6666
b = -440.8333
```





Where the entries of a are the x and y coordinates of the equilibrium point respectively, and b is the z coordinate.