Table of Contents

HOMEWORK 2 KEY

```
%Due to the nature of programming, you should not expect your code to
 look
%exactly the same. However, the output should be close to identical
%(disregarding formatting). I have specifically done a number of these
%problems using varying syntax (e.g. disp(), fprintf(), and just a
 variable
%name for output) to showcase the variety of ways things can be done,
 but
%that's not to say there's no best way.
```

Problem 3.1

```
clc;clear all;format compact;close all;

R = [0.9 1.5 1.3 1.3]; %Defines R
d = [1 1.25 3.8 4.0]; %Defines d

for idx = 1:4
    fprintf('For R = %.1f and d = %.1f: ',R(idx),d(idx))
    if d(idx) > 3*R(idx) %Displays error if height is greater than
 tank
        fprintf('Overtop\n')
    elseif d(idx) > R(idx) %Calculates volume for d > R
        V = (pi*R(idx)^3)/3+pi*R(idx)^2*(d(idx)-R(idx));
        fprintf('Volume = %.2f\n',V)
    else %Calculates volume for d < R
        V = (pi*d(idx)^3)/3;
        fprintf('Volume = %.2f\n',V)
    end
end
```

```
For R = 0.9 and d = 1.0: Volume = 1.02
For R = 1.5 and d = 1.3: Volume = 2.05
For R = 1.3 and d = 3.8: Volume = 15.57
For R = 1.3 and d = 4.0: Overtop
```

Problem 3.2

```
clc;clear all;format compact;close all;

%Defining variables
P = 100000;
i = 0.05;
n = 10;

%Starting table and loop
fprintf('Year:\t Worth ($):\n')
for idx = 1:n
    F(idx) = P*(1+i)^idx; %Calculates worth
    fprintf('%3i\t\t   %.0f\n',idx,F(idx)) %Displays table values
end
```

```
Year:  Worth ($):
  1      105000
  2      110250
  3      115763
  4      121551
  5      127628
  6      134010
  7      140710
  8      147746
  9      155133
 10      162889
```

Problem 3.5

```
clc;clear all;format compact;close all;

%Declares x value to approximate/calculate
x = 0.9;

for vals = [1:8] %Loops through the algorithm using 1, 2, 3, and 8
 terms
    est = 0;
    for i = 1:vals %Loops through the equation for each term in the
 series
        est = est + (-1)^(i-1)*x^(2*i-1)/factorial(2*i-1);
        err = abs((sin(x)-est)/sin(x))*100;
    end
    %Prints results to command window
    fprintf('A series with %i terms returns %.15f (error: %.2E%
%)\n',vals,est,err)

    %disp works as below, but note how long and hard to understand
 this is.
    %This is why fprintf is really nice!!!

    %disp(['A series with ' num2str(vals,'%i') ' terms returns ' ...
```

```
    %    num2str(est,'%.4f') ' (error: ' num2str(err,'%.3f') '%)'])

end
```

```
A series with 1 terms returns 0.900000000000000 (error: 1.49E+01%)
A series with 2 terms returns 0.778500000000000 (error: 6.16E-01%)
A series with 3 terms returns 0.783420750000000 (error: 1.20E-02%)
A series with 4 terms returns 0.783325849821429 (error: 1.35E-04%)
A series with 5 terms returns 0.783326917448438 (error: 9.98E-07%)
A series with 6 terms returns 0.783326909586821 (error: 5.19E-09%)
A series with 7 terms returns 0.783326909627640 (error: 2.00E-11%)
A series with 8 terms returns 0.783326909627483 (error: 5.67E-14%)
```

Problem 3.9

```
clc;clear all;format compact;close all;

%Declaring variables
n = [0.036 0.020 0.015 0.030 0.022];
S = [0.0001 0.0002 0.0012 0.0007 0.0003];
B = [10 8 20 25 15];
H = [2 1 1.5 3 2.6];

%The following may also be done with a loop. Note there are a lot of
 arrays
%involved here so there are many dot-operators.
U = S.^0.5./n.*(B.*H./(B+2*H)).^(2/3);

%Making the matrix
matrix = [n' S' B' H' U'];

%Displaying the table. Note it's possible to use disp, but it's
 usually not
%ideal.
disp('       n        S         B          H         U')
disp(matrix)
```

```
       n         S         B          H          U
    0.0360    0.0001   10.0000    2.0000    0.3523
    0.0200    0.0002    8.0000    1.0000    0.6094
    0.0150    0.0012   20.0000    1.5000    2.7569
    0.0300    0.0007   25.0000    3.0000    1.5894
    0.0220    0.0003   15.0000    2.6000    1.2207
```

Problem 3.12

```
clc; clear all;format compact;close all;

%Defining variables
tstart = 0;
tend = 20;
ni = 8;
```

```
%Creating array of t values
%NOTE: this also works t = linspace(tstart,tend,ni+1);
t = [tstart:(tend-tstart)/ni:tend-tstart];

%Calculating y (unmuted so output is shown)
y = 12 + 6*cos(2*pi*t/(tend-tstart))
```

```
y =
  Columns 1 through 7
   18.0000   16.2426   12.0000    7.7574    6.0000    7.7574   12.0000
  Columns 8 through 9
   16.2426   18.0000
```

Problem 3.13

```
clc; clear all;format compact;close all;

%Using a loop to go through each value of a
for a = [0 2 10 -4]
    %Predefining error as 0 in case x = 0
    err = 0;
    %Defining x as the absolute value of a
    x = abs(a);

    %Computes the root for the real part of a, and skips it if a = 0
 since
    %that would incur division by 0
    while x ~= 0
        xold=x;
        x=(x+abs(a)/x)/2;
        err=abs((x-xold)/x);
        if err <= 1e-4 %Break condition
            break
        end
    end

    %Prints header to command window
    fprintf('With a = %i\n',a)
    if a < 0 %Prints the root of a if imaginary
        fprintf('Result: %.2fi, Error: %.2E\n\n',x,err)
    else %Prints the root of a if real
        fprintf('Result: %.2f, Error: %.2E\n\n',x,err)
    end
end
```

```
With a = 0
Result: 0.00, Error: 0.00E+00

With a = 2
Result: 1.41, Error: 1.50E-06

With a = 10
Result: 3.16, Error: 5.63E-05
```

```
With a = -4
Result: 2.00i, Error: 4.65E-08
```

Problem 5.1

```
clc;clear all;format compact;close all;

%Defining variables
m = 95;
t = 9;
g = 9.81;
Es = 5; %Stopping error

%Defining function
func = @(Cd,m,g,t) sqrt(g*m/Cd)*tanh(t*sqrt(g*Cd/m))-46;

%Finding root
[root,fx,Ea,its] = bisect(func,0.2,0.5,Es,100,m,g,t);

%Displays root and error
root
Ea

root =
    0.4063
Ea =
    4.6154
```

Problem 5.7b

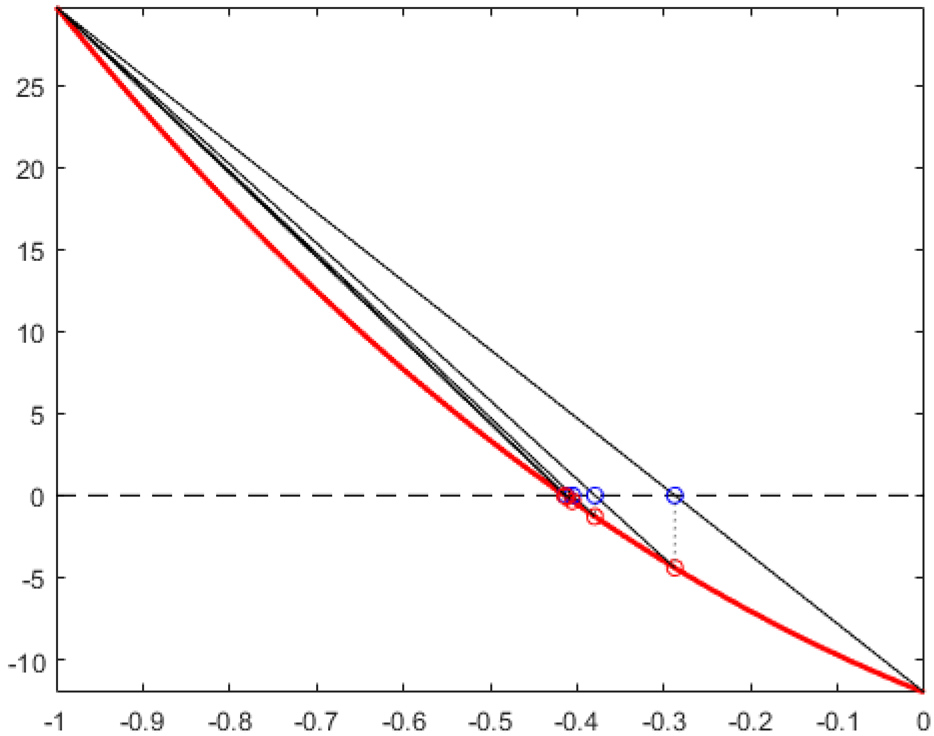
```
clc;clear all;format compact;close all;

%Defining function and search range
func = @(x) -12-21*x+18*x^2-2.75*x^3;
range=[-1,0];

%Initializing variables (prevents errors and early termination due to
 how
%the loop is formatted)
err = 1;
guess = range(1);

%Performs root finding via bisection
while err > 0.01
    guessold = guess;
    guess = mean(range); %Guesses root at middle of search area
    val = func(guess); %Evaluates function at guess
    if func(range(1))*func(guess) < 0 %checks for sign change between
 lower
        range(2)=guess;               %bound and the value at the
 middle.
```

```
    else                                  %If the sign changes, moves the
 upper
        range(1)=guess;                   %bound down, and lower if no
 change.
    end
    err = abs((guess-guessold)/guess);
end

%Displays the error and the root
disp('Root:')
disp(guess)
disp('Error:')
disp(err)

Root:
   -0.4180
Error:
    0.0093
```

Problem 5.7c

```
clc;clear all;format compact;close all;

%NOTE: This problem differs from the others in that I have created it
 not
%only to solve the problem, but to graphically show how the solution
 was
%attained. The latter is not necessary, it was only implemented for
%learning purposes. However, the given equation is not the most ideal
 for
%such a display, so another is provided below that you can simply
 uncomment
%and it will use it instead. Note that I've designed this to be as
 flexible
%as possible, so go ahead and try other equations if you wish.
%
%All plot-related lines will end with a %P, for Plot.

%Declares function and variables
func = @(x) -12-21*x+18*x.^2-2.75*x.^3; %vectorized to satisfy
 matlab's
xl = -1;                                %pickiness about functions
 like this
xu = 0;
Es = .01;

%Unmute the next line for a more visual example of the false position
 method
%func = @(x) exp(x-10)-x.^2-20; xl = 13; xu = 17;

fplot(func,[xl,xu],'r','LineWidth',2) %P
line([xl,xu],[0,0],'LineStyle','--','Color','Black') %P
axis([xl xu min(func(xu),func(xl)),max(func(xu),func(xl))]) %P
```

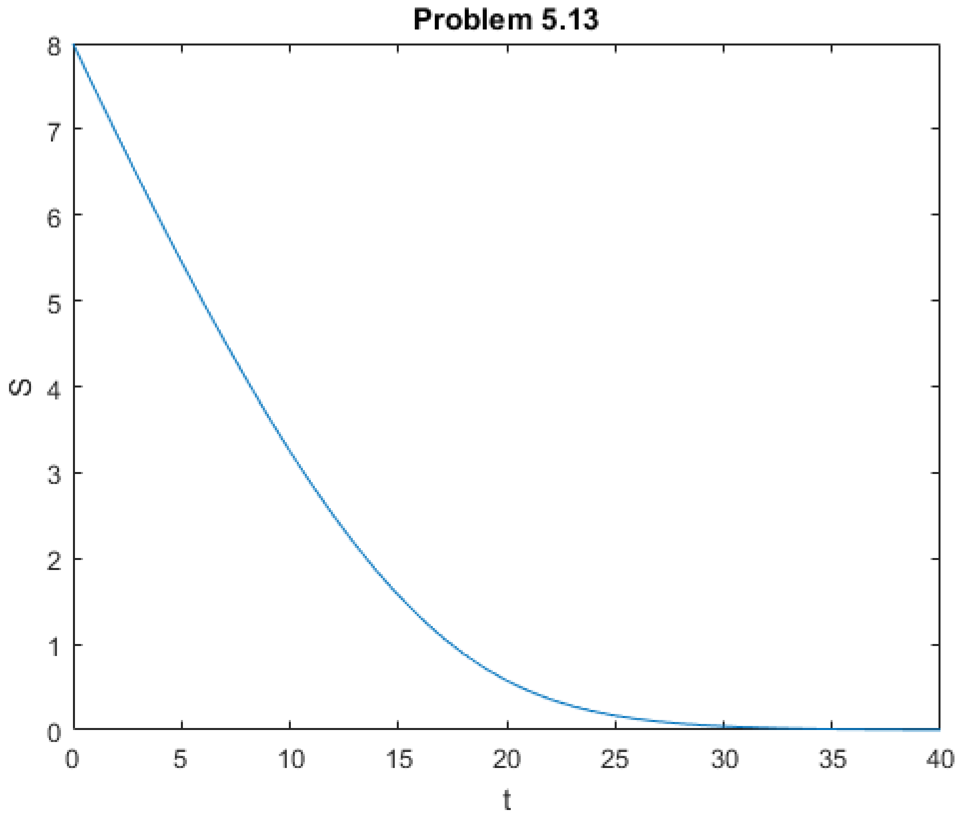
```
hold on %P

%Starts the false position method algorithm
guess = 0;
while 1
    guessold = guess;
    guess = xu-func(xu)*(xl-xu)/(func(xl)-func(xu)); %Finds x-
intercept
    val = func(guess); %Evaluates f(x) at the x-intercept between xl,
 xu
    hand(1) = line([xl,xu],[func(xl),func(xu)],'Color','Black'); %P
    hand(2) = plot(guess,0,'bo'); %P
    hand(3) = plot(guess,val,'ro'); %P
    hand(4) = line([guess,guess],[0,val], 'Color',
[0.2,0.2,0.2],'LineStyle',':'); %P
    if func(xl)*func(guess) < 0 %See prior problem for logic
 explanation
        xu=guess;
    else
        xl=guess;
    end
    pause(1) %P (leaves time to see what's happening)
    err = abs((guess-guessold)/guess);
    if err <= Es %Checks if the error is below an allowable amount
        break %Breaks loop
    end
    %delete(hand) %P, uncomment to remove the prior iteration's
 markers
end

%Displays the root and the error
fprintf('The root is %.4f with a relative error of %.2f%%
\n',guess,err*100)

The root is -0.4140 with a relative error of 0.45%
```



Problem 5.13

```
clc;clear all;format compact;close all;

%Predefine variables
S_0 = 8;
vm = 0.7;
ks = 2.5;
tstart = 0;
tend = 40;
interval = 1;

%Create t array, declare function
t = [tstart:interval:tend];
func = @(S,t) S_0-vm*t+ks*log(S_0/S)-S;

%Solves S for each increment of t using bisect.m
for idx = 1:length(t)
    S(idx) = bisect(func,0,10,0.01,100,t(idx));
end

%Plots the results
plot(t,S)
xlabel('t')
ylabel('S')
```

```
title('Problem 5.13')
```



Published with MATLAB® R2016b