```
% Set the tolerance value
tol = 0.00001;
soltol = 0.0001;
% Enter lower and upper limits
low = input('Enter the lower limit:  ');
high = input('Enter the higher limit:  ');

% Repeat caluclations until interval is less than 2*tolerance
while high-low > 2*tol
    mid = (high+low)/2;
    % Evaluate function at lower limit and midpoint
    fl = fun1(low);
    fm = fun1(mid);
    % Multiply values: if product > 0, signs are same. Keep upper limit
    % If products <= 0, signs are opposite. Keep lower limit
    prod = fl*fm;
    if prod > 0
        low = mid;
    else
        high = mid;
    end
end
% Root = midpoint of interval. Print out root
root = (high+low)/2;
fprintf('\n\n Root found: %f\n\n',root)

% Check to see if converges value is a root
if abs(fm) < soltol
    root = (high+low)/2;
    fprintf('\n\n Root found: %f\n\n',root)
else
    fprintf('\n\n No root found. \n\n')
end
```

```
Error using input
Cannot call INPUT from EVALC.

Error in bisection_method_example (line 5)
low = input('Enter the lower limit:  ');
```

Published with MATLAB® R2022b

```
function y = fun1(x)

y = exp(x) - 15*x - 10;
```

```
Not enough input arguments.

Error in fun1 (line 3)
y = exp(x) - 15*x - 10;
```

Published with MATLAB® R2022b

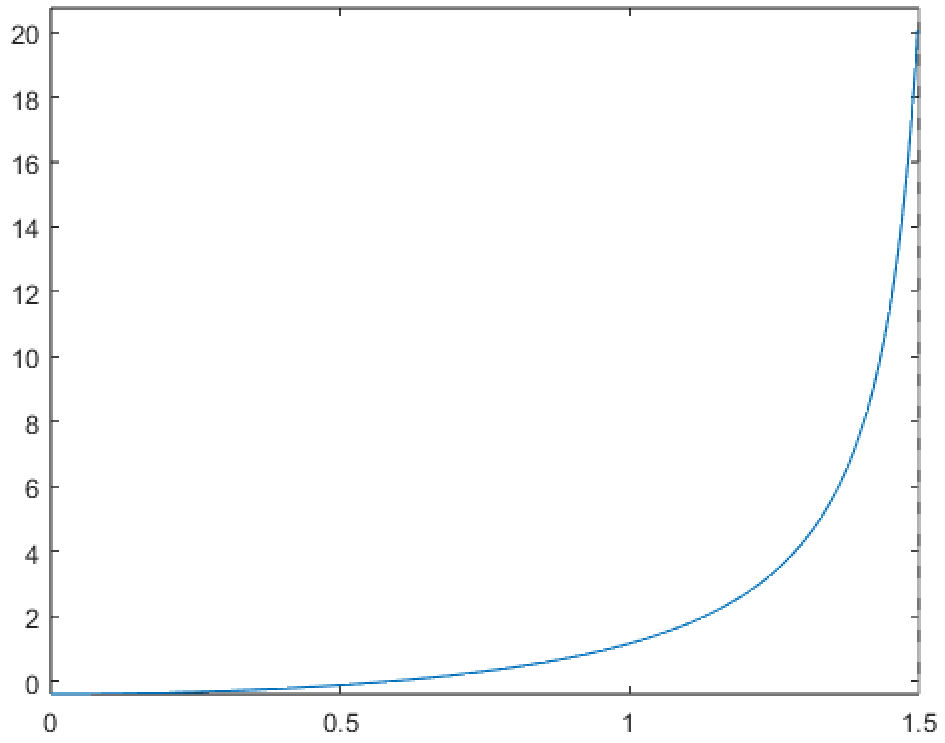
CHE 1411L - Week 9 Lab Assignment

Table of Contents

Example 6.1

```
ex6_1(4)

fplot(@ex6_1, [0 1.5])

r = fzero(@ex6_1, .5)

ex6_1(r)
```

```
ans =

    4.2313

Warning: Function behaves unexpectedly on array inputs. To improve
 performance,
properly vectorize your function to return an output with the same size and
shape as the input arguments.

r =

    0.5932


ans =

  -5.5511e-17
```



CHE 1411L - Example 6.2

```
r = fzero(@ex6_1, [0 1.5])
```

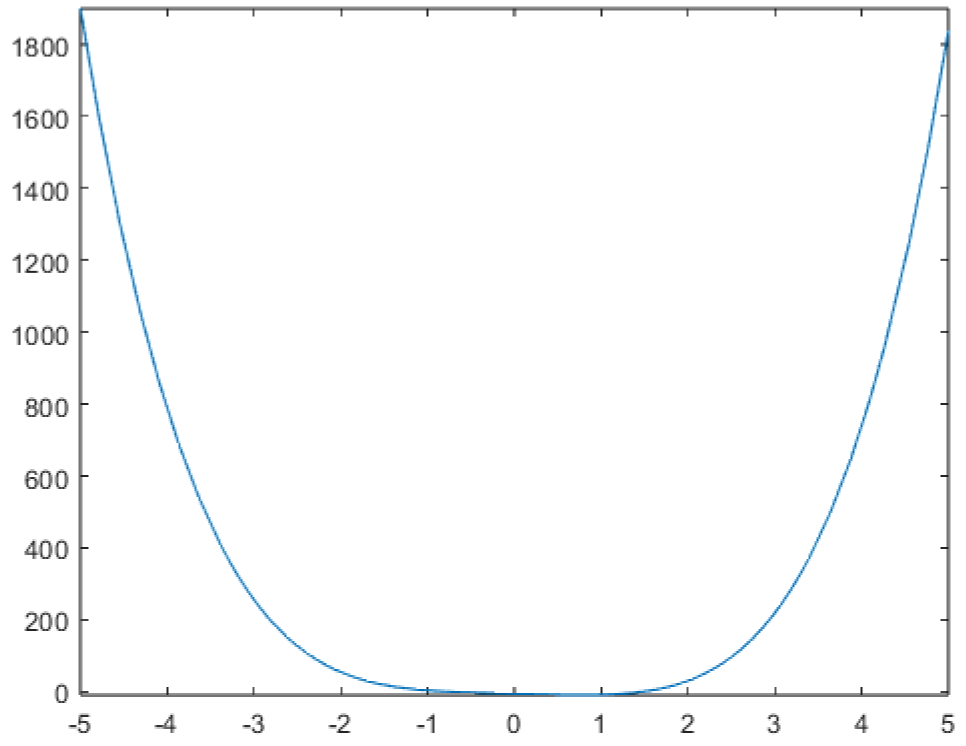
```
r =

    0.5932
```

Problem 6.5a

Consider the following function: 3x^4 - 6x = 6 a. put the funciton in standard form (ax - by = c), and plot the function using MATLAB.

```
fplot(@a65)
```

```
Warning: Function behaves unexpectedly on array inputs. To improve
 performance,
properly vectorize your function to return an output with the same size and
shape as the input arguments.
```



Problem 6.5b

Write a simple program using the Bisection Method to find any one root of the equation to within 0.01.

```
% Set the tolerance value
tol = 0.01;
% Enter lower and upper limits
% rl = input('Enter the lower limit:  ');   Hard coded as 0 for publishing.
rl = 0;
% ru = input('Enter the higher limit:  ');  Hard coded as 10 for
% publishing.
ru = 10;

% Repeat caluclations until interval is less than 2*tolerance
while ru-rl > 2*tol
    rm = (ru+rl)/2;
    % Evaluate function at lower limit and midpoint
    fl = prob6_5(rl);
    fm = prob6_5(rm);
    % Multiply values: if product > 0, signs are same. Keep upper limit
    % If products <= 0, signs are opposite. Keep lower limit
    prod = fl*fm;
    if prod > 0
        rl = rm;
    else
        ru = rm;
    end
end


% Root = midpoint of interval. Print out root
root = (ru+rl)/2;
fprintf('\n\n Root found: %f\n\n',root)
```

```


 Root found: 1.494141
```

Problem 6.5c

Write a simple MATLAB script to find the solution to the polynomial equation using the roots command.

```
x = [3 -6 -6];
roots(x)
```

```

ans =

    2.7321
   -0.7321
```

Problem 6.5d

Check your results using the Excel Goal Seek tool.

```
% Problem complete in attached Excel Workbook
```

Published with MATLAB® R2022b

```
function f = ex6_1(x)

f = x*tan(x) - 0.4;
```

```
Not enough input arguments.

Error in ex6_1 (line 3)
f = x*tan(x) - 0.4;
```

Published with MATLAB® R2022b

```
function f = prob6_5(x)

f = (3.*(x.^4)) - (6.*x) - 6;
```

```
Not enough input arguments.

Error in prob6_5 (line 3)
f = (3.*(x.^4)) - (6.*x) - 6;
```

Published with MATLAB® R2022b

	Goal Seek Problem 6.5
x	1.494547
y	0.000582
	It works!!
	:-)

```
%Example 6.3
p=[2 10 0 -144];
r=roots(p)

polyval(p,r)


r =

  -4.0000 + 2.8284i
  -4.0000 - 2.8284i
   3.0000 + 0.0000i


ans =

   1.0e-12 *

   0.0284 - 0.1563i
   0.0284 + 0.1563i
  -0.1137 + 0.0000i
```

Published with MATLAB® R2022b

```
function y = ppt6_2_p16_fun1(x)
y = 3*x^3-15*x^2-20*x+50;
```

```
Not enough input arguments.

Error in ppt6_2_p16_fun1 (line 2)
y = 3*x^3-15*x^2-20*x+50;
```

Published with MATLAB® R2022b

```
function ypr = ppt6_2_p16_fun1pr(x)
ypr = 9*x^2-30*x-20;
```

```
Not enough input arguments.

Error in ppt6_2_p16_fun1pr (line 2)
ypr = 9*x^2-30*x-20;
```

Published with MATLAB® R2022b

```
%Apply Newton's Method to find root of equation defined in function
%'ppt6_2_p16_fun1' Derivative of 'ppt6_2_p16_fun1' is function
%'ppt6_2_p16_fun1pr'

%set cpnverge tolerance
tol = 0.00001;

%Input initial guess
x = input(' Enter initial guess\n');

%Find value of function at x
f = ppt6_2_p16_fun1(x);

while abs(f) > tol
    fpr = ppt6_2_p16_fun1pr(x);
    x = x - f/fpr;
    f = ppt6_2_p16_fun1(x);
end

fprintf('\n\n Root found: %.4f\n',x)
```

```
Error using input
Cannot call INPUT from EVALC.

Error in ppt6_2_p16 (line 9)
x = input(' Enter initial guess\n');
```

Published with MATLAB® R2022b

```
fzero('fun6_6',4)
fzero('fun6_6',-3)
```

```

ans =

    4.6681


ans =

   -4.7543

```

Published with MATLAB® R2022b

```
function y = fun6_6(x)
y = (4.5+x^3*cos(x))/(7*x);
```

```
Not enough input arguments.

Error in fun6_6 (line 2)
y = (4.5+x^3*cos(x))/(7*x);
```

Published with MATLAB® R2022b