1.

A. bisectionScript.m:

clc; clear all;

f = @(x)(2\*sin(x^2)-x);

a = -.01;

b = 2;

n = 1;

ep = 0.01;

x = (a + b)/2;

fprintf('n a f(a) b f(b) x f(x)\n');

while (abs(f(x)) > ep)

fprintf('%d %4.2f %4.2f %4.2f %4.2f %4.2f %4.2f\n', n,a, f(a), b, f(b), x, f(x));

x = (a + b)/2;

if(f(x)\*f(a) > 0)

a = x;

else

b = x;

end

n = n + 1;

end

root = x;

fprintf('One root is: %4.2f\n', root);

Output:

n a f(a) b f(b) x f(x)

1 -0.01 0.01 2.00 -3.51 1.00 0.68

2 1.00 0.68 2.00 -3.51 1.00 0.68

3 1.50 0.07 2.00 -3.51 1.50 0.07

4 1.50 0.07 1.75 -1.58 1.75 -1.58

5 1.50 0.07 1.62 -0.65 1.62 -0.65

6 1.50 0.07 1.56 -0.26 1.56 -0.26

7 1.50 0.07 1.53 -0.09 1.53 -0.09

One root is: 1.51

B. Finding all the roots with the fzero function:

First found the max number of roots by looking at a graph, then plugged it in.

k = [ ];

InitialGuess = -1;

currentRoots = 0;

maxRoots = 3;

while currentRoots<maxRoots

if abs(feval(f,InitialGuess))<=ep,

root=InitialGuess;

else

[root,c,e] = fzero(f,InitialGuess,ep);

if e <= 0

return

end

end

k=[k,root];

currentRoots=currentRoots+1;

InitialGuess=InitialGuess+1;

end

fprintf('By the fzero function the other roots are:');

disp(k);

fprintf('\n');

Output:

By the fzero function the other roots are: 0 0.5055 1.5115

Finding the symbolic representations of the roots:

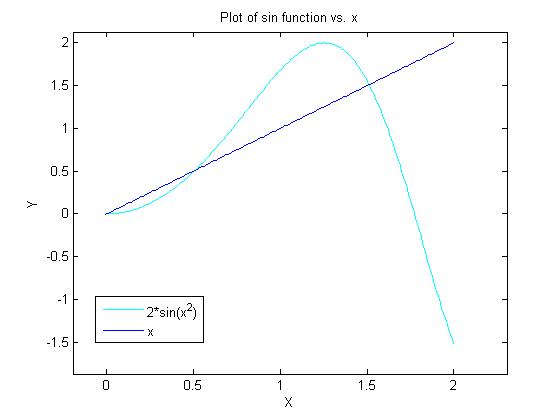
Sym(k)

Output:

0, 4552979546699371/9007199254740992, 6807383224042511/4503599627370496

This was the only answer I could get when converting the previous solutions into symbolic form. When I tried using the solve function, the only output was 0.

C. Plot of 2sin(x^2) function vs. X



I can see where the solutions are on this plot.

D.

1. My code was very easy to use; all it required the user to do is run it. If the user wanted to change a parameter, then they would do so by changing the m-file.
2. The root converged to 1.51. It took 8 iterations and the estimates of the root went directly to the root.
3. Yes, I could tell exactly which root I would find. As was stated above, the program went directly to a root, so I could tell which root I would find.

1.

A. Function bisection.m :

function [root, numIter] = bisection(f, a, b, ep)

n = 1;

x = (a + b)/2;

while (abs(f(x)) > ep)

x = (a + b)/2;

if(f(x)\*f(a) > 0)

a = x;

else

b = x;

end

n = n + 1;

end

root = x;

numIter = n;

scriptUsingBisectionFunction.m:

f = @(x) x.\*cos(3\*x);

a = -1;

b = 5;

ep = 0.01;

[root, numIter] = bisection(f, a, b, ep);

fprintf('It took %d iterations to find the root %4.2f\n', numIter, root);

Output:

It took 12 iterations to find the root 2.62

B. Finding all roots with the fzero function:

k = [ ];

InitialGuess = -1;

currentRoots = 0;

maxRoots = 7;

while currentRoots<maxRoots

if abs(feval(f,InitialGuess))<=ep,

root=InitialGuess;

else

[root,val,flag] = fzero(f,InitialGuess,ep);

if flag <= 0

return

end

end

k=[k,root];

currentRoots=currentRoots+1;

InitialGuess=InitialGuess+1;

end

fprintf('By the fzero function the other roots are:');

disp(k);

fprintf('\n');

Output:

By the fzero function the other roots are: -0.5236 0 0.5236 1.5708 2.6180 3.6652 4.7124

Finding the symbolic representation of the roots:

syms x;

fx = x\*cos(3\*x);

solve(fx,x)

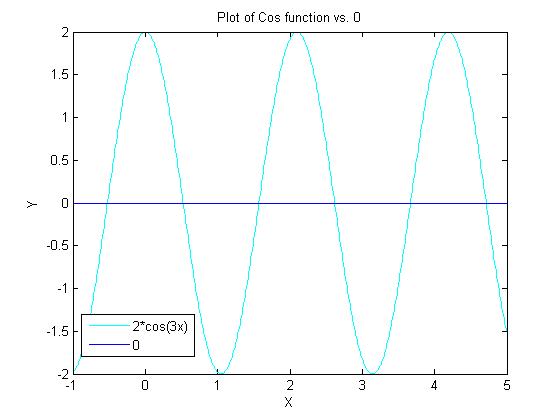
Output:

0

pi/6

When using the solve function, it only outputs two solutions.

C. Plot of 2cos(3x) function vs. 0



D.

1. My code was very easy to use; it only required running the top level script that has the function defined. If someone needed to change a value, then the top level script would just need a bit of tweaking.
2. The root converged to 2.62, and took 12 iterations. The initial estimates did not go directly to the root this time. There were many roots; I couldn’t tell which root it was converging to.
3. As stated above, I was not able to initially tell which root it was converging to.

3.

A.

clear all;

syms y w Ta x;

fy = (w/Ta) \* sqrt(1+(diff(y,x)^2)) ;

int(fy,x);

int(ans,x);

Output:

(w\*x^2)/(2\*Ta)

I don’t think this is the correct answer, because the equation above can’t be represented in Matlab correctly or at least from what I have read/learned in class. The part that can’t be represented correctly is dx/dy.

B.

Notes in class state that the Taylor series function in Matlab is used:

taylor(func, numOfTerms, aboutWhatPoint);

syms y w Ta x;

taylor((Ta/w) \* cosh(w\*x/Ta) + y0 - (Ta/w), 4, 1);

Output:

y0 + sinh(w/Ta)\*(x - 1) - Ta/w + (Ta\*cosh(w/Ta))/w + (w^2\*sinh(w/Ta)\*(x - 1)^3)/(6\*Ta^2) + (w\*cosh(w/Ta)\*(x - 1)^2)/(2\*Ta)