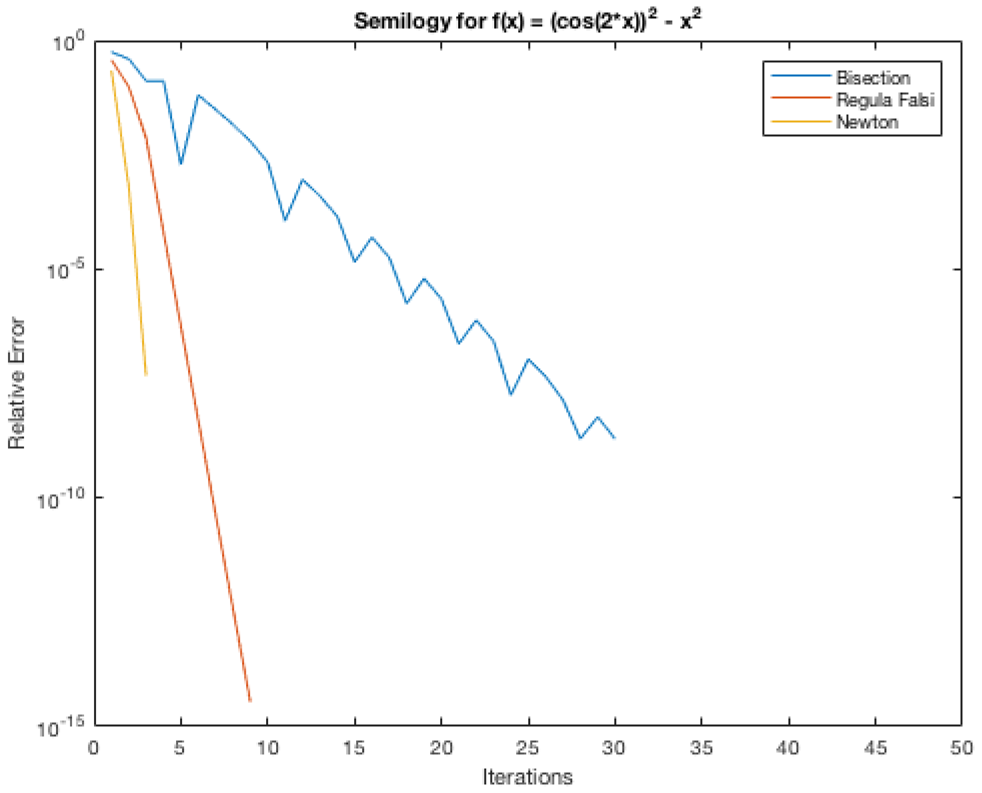
Matlab Homework 7 - Tu Tran

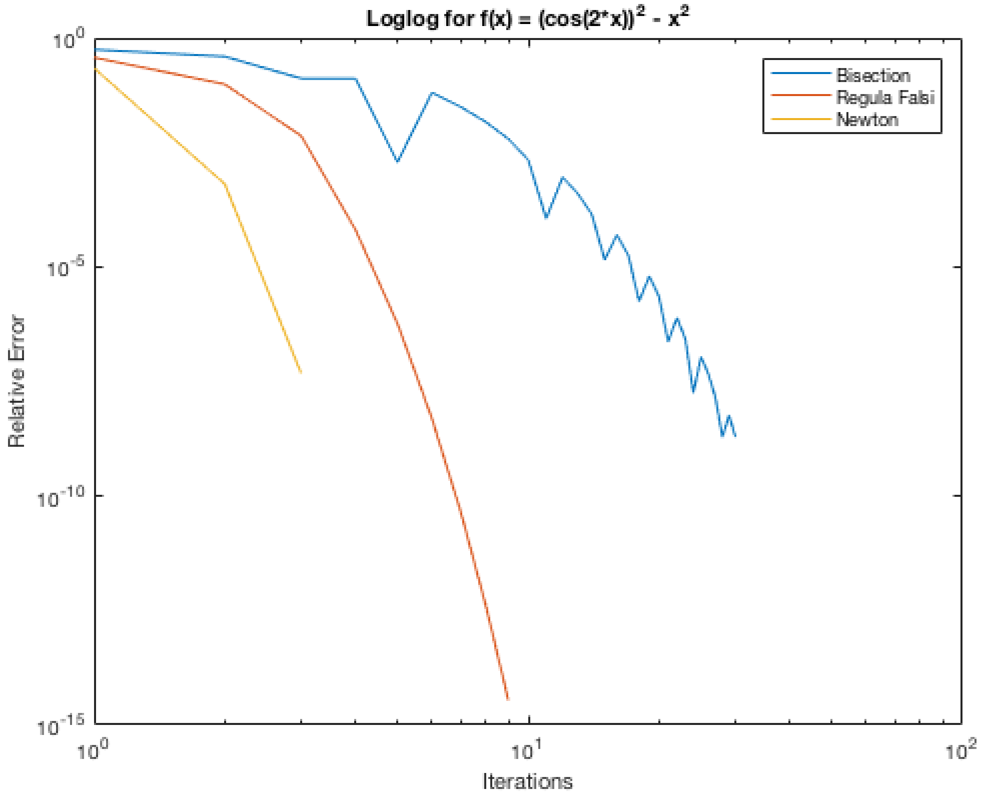
```
clc;
clear all;
format long;

% PART (a)
% The code for three methods is in the files newton.m
% bisection.m and regula_falsi.m

% PART (b)
f = @(x) (cos(2*x))^2 - x^2;
fprime = @(x) -2*sin(4*x) - 2*x;
bound = 10e-10;
a = 0; b = 1.5;
x0 = 3/4;

fprintf('Solutions for f(x) = (cos(2*x))^2 - x^2\n');
fprintf('calculate using the Bisection method');
x1 = bisection(f, bound, 10000, a, b)
fprintf('calculate using the Regula Falsi method');
x2 = regula_falsi(f, bound, 10000, a, b)
fprintf('calculate using the Newton method');
x3 = newton(f, fprime, bound, x0, 10000)

n=1:50;

errors1 = zeros(length(n));
errors2 = zeros(length(n));
errors3 = zeros(length(n));

for i=1:length(n)
    % calculate using the Bisection method
    est1 = bisection(f, bound, i, a, b);
    % calculate using the Regula Falsi method
    est2 = regula_falsi(f, bound, i, a, b);
    % calculate using the Newton's method
    est3 = newton(f, fprime, bound, x0, i);

    errors1(i) = abs(f(x1) - f(est1));
    errors2(i) = abs(f(x2) - f(est2));
    errors3(i) = abs(f(x3) - f(est3));
end

figure();
semilogy(n, errors1, n, errors2, n, errors3);
title('Semilogy for f(x) = (cos(2*x))^2 - x^2');
xlabel('Iterations');
ylabel('Relative Error');
legend('Bisection', 'Regula Falsi', 'Newton');

figure();
```

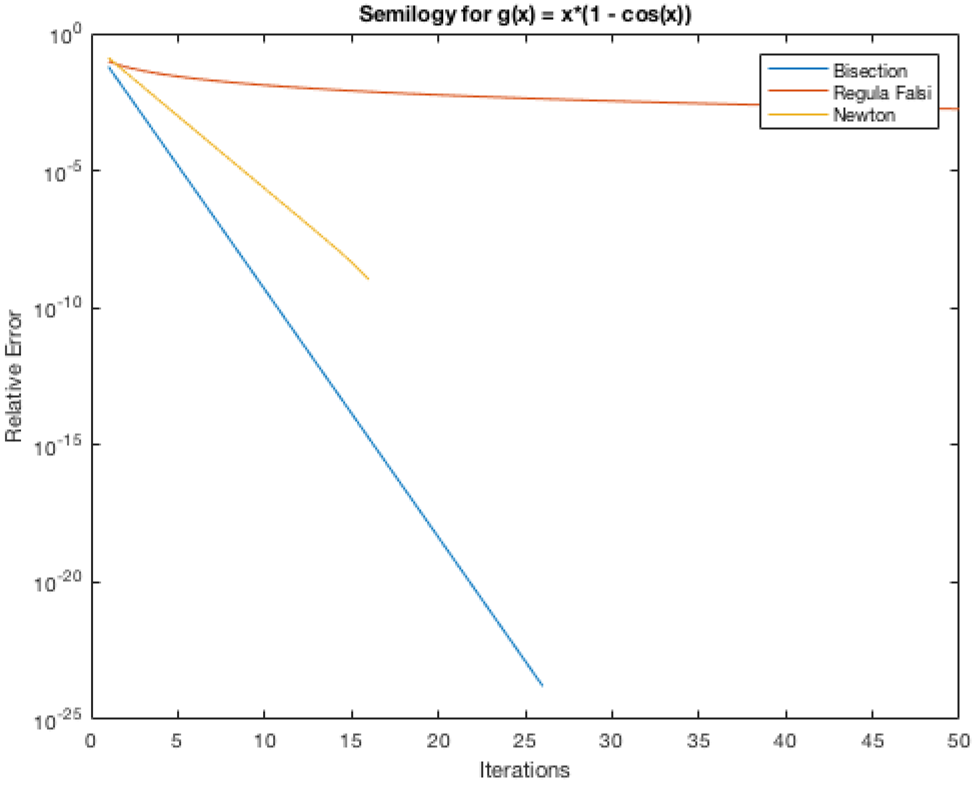
```
loglog(n, errors1, n, errors2, n, errors3);
title('Loglog for f(x) = (cos(2*x))^2 - x^2');
xlabel('Iterations');
ylabel('Relative Error');
legend('Bisection', 'Regula Falsi', 'Newton');

% Newton's method converges with quadratic rate
% Regula Falsi converges with linear rate
% Bisection also converges with linear rate but with alpha = 1/2
 (slower
% than Regula Falsi)

% PART (c)
g = @(x) x*(1 - cos(x));
gprime = @(x) 1 - cos(x) + x*sin(x);
bound = 10e-10;
a = -2; b = 1;
x0 = 1;

fprintf('Solutions for g(x) = x * (1 - cos(x) = 0\n');
fprintf('calculate using the Bisection method');
x1 = bisection(g, bound, 10000, a, b)
fprintf('calculate using the Regula Falsi method');
x2 = regula_falsi(g, bound, 10000, a, b)
fprintf('calculate using the Newton method');
x3 = newton(g, gprime, bound, x0, 10000)

n=1:50;

errors1 = zeros(length(n));
errors2 = zeros(length(n));
errors3 = zeros(length(n));

for i=1:length(n)
    % calculate using the Bisection method
    est1 = bisection(g, bound, i, a, b);
    % calculate using the Regula Falsi method
    est2 = regula_falsi(g, bound, i, a, b);
    % calculate using the Newton's method
    est3 = newton(g, gprime, bound, x0, i);

    errors1(i) = abs(g(x1) - g(est1));
    errors2(i) = abs(g(x2) - g(est2));
    errors3(i) = abs(g(x3) - g(est3));
end

figure();
semilogy(n, errors1, n, errors2, n, errors3);
title('Semilogy for g(x) = x*(1 - cos(x))');
xlabel('Iterations');
ylabel('Relative Error');
legend('Bisection', 'Regula Falsi', 'Newton');

figure();
```

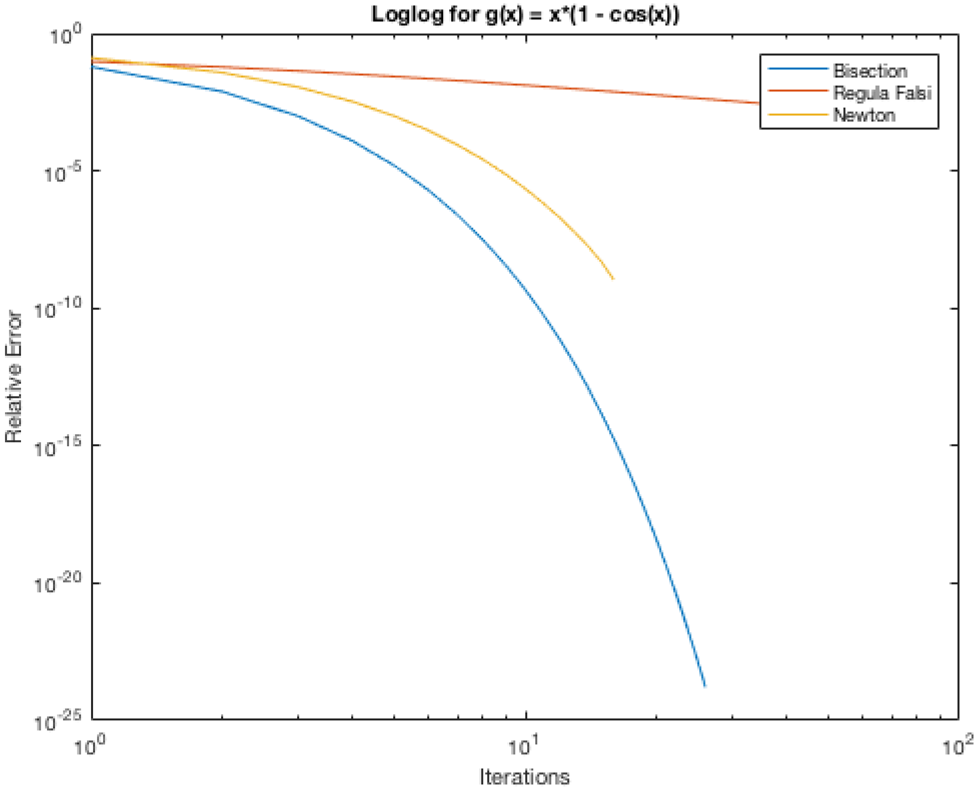
```
loglog(n, errors1, n, errors2, n, errors3);
title('Loglog for g(x) = x*(1 - cos(x))');
xlabel('Iterations');
ylabel('Relative Error');
legend('Bisection', 'Regula Falsi', 'Newton');

% The bisection method converges faster than Newton
% and Regula Falsi

% PART (d)
g = @(x) x*(1 - cos(x));
gprime = @(x) 1 - cos(x) + x*sin(x);
bound = 10e-10;
a = -2; b = 1;
x0 = 1;

fprintf('Solutions for g(x) = x * (1 - cos(x) = 0\n');
fprintf('calculate using the normal Newton method');
x1 = newton(g, gprime, bound, x0, 1000)
fprintf('calculate using the modified Newton method');
x2 = newton_modified(g, gprime, bound, x0, 1000)

n=1:50;

errors1 = zeros(length(n));
errors2 = zeros(length(n));

for i=1:length(n)
    % calculate using the Newton method
    est1 = newton(g, gprime, bound, x0, i);
    % calculate using the modified Newton method
    est2 = newton_modified(g, gprime, bound, x0, i);

    errors1(i) = abs(g(x1) - g(est1));
    errors2(i) = abs(g(x2) - g(est2));
end

figure();
semilogy(n, errors1, n, errors2);
title('Semilogy for g(x) = x*(1 - cos(x))');
xlabel('Iterations');
ylabel('Relative Error');
legend('Newton', 'Modified Newton');

figure();
loglog(n, errors1, n, errors2);
title('Loglog for g(x) = x*(1 - cos(x))');
xlabel('Iterations');
ylabel('Relative Error');
legend('Newton', 'Modified Newton');

% Modified Newton does not converge (the value stays the same
 throughout iterations)
```

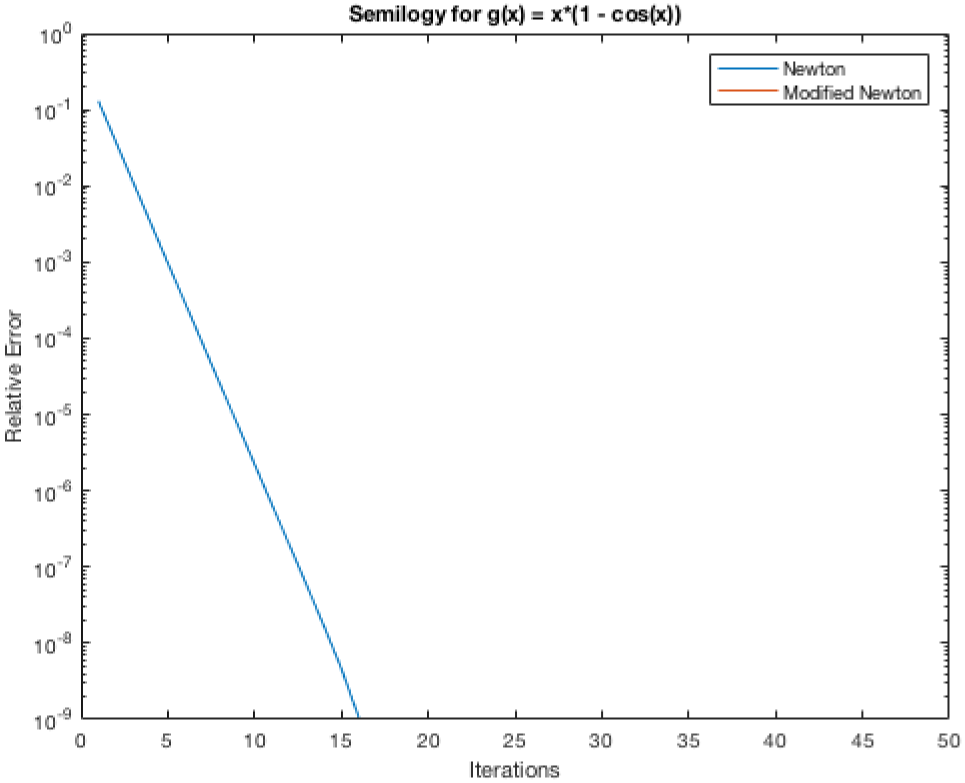
```
Solutions for f(x) = (cos(2*x))^2 - x^2
calculate using the Bisection method
x1 =

   0.514933264581487

calculate using the Regula Falsi method
x2 =

   0.514933264661129

calculate using the Newton method
x3 =

   0.514933264661129

Solutions for g(x) = x * (1 - cos(x) = 0
calculate using the Bisection method
x1 =

    -7.450580596923828e-09

calculate using the Regula Falsi method
x2 =

   0.011827770915039

calculate using the Newton method
x3 =

     9.639406144345542e-04

Solutions for g(x) = x * (1 - cos(x) = 0
calculate using the normal Newton method
x1 =

     9.639406144345542e-04

calculate using the modified Newton method
x2 =

     1.193602342586253e-05
```

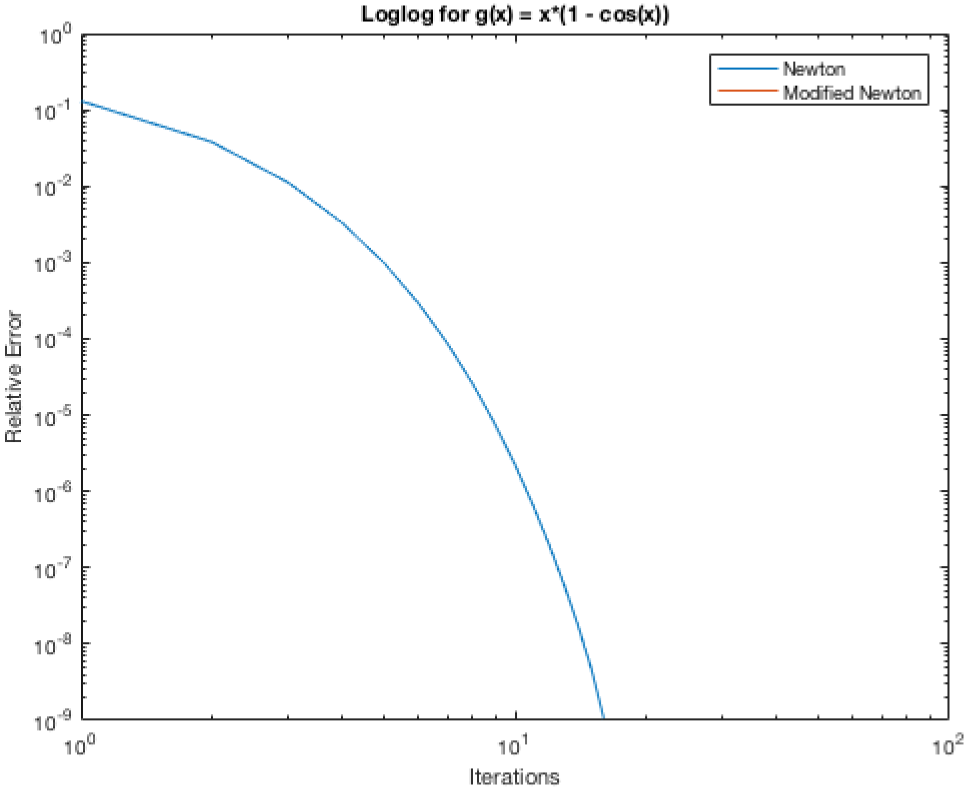

Semilogy for f(x) = (cos(2*x))^2 - x^2
Relative Error
10^0
10^{-5}
10^{-10}
10^{-15}
0
5
10
15
20
25
30
35
40
45
50
Iterations
Bisection
Regula Falsi
Newton


Loglog for f(x) = (cos(2*x))^2 - x^2
Relative Error
10^0
10^{-5}
10^{-10}
10^{-15}
10^0
10^1
10^2
Iterations
Bisection
Regula Falsi
Newton


Semilogy for g(x) = x*(1 - cos(x))
Relative Error
Iterations
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
10^{-25}
0
5
10
15
20
25
30
35
40
45
50
Bisection
Regula Falsi
Newton


Loglog for g(x) = x*(1 - cos(x))
Relative Error
Iterations
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
10^{-25}
10^{0}
10^{1}
10^{2}
Bisection
Regula Falsi
Newton


Semilogy for g(x) = x*(1 - cos(x))
Newton
Modified Newton
Relative Error
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
0
5
10
15
20
25
30
35
40
45
50
Iterations


Loglog for g(x) = x*(1 - cos(x))
Newton
Modified Newton
Relative Error
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{0}
10^{1}
10^{2}
Iterations

Published with MATLAB® R2016a