Table of Contents

Assignemnt - 4

Newton and Secant

```
% Name         : Niraj Kumar Singh
% Roll Number : AM25M807
% Department   : Applied Mechanics
```

Part 1 (Preprocessing)

```
clc; clear; close all;

% --------- Secant Method ----------
x0 = 10; x1 = 9; tol = 1e-6;
f = @(x) (x-1).*(exp(x-1)-1);

rel_err = inf; k = 0;
it_s = []; err_s = [];
```

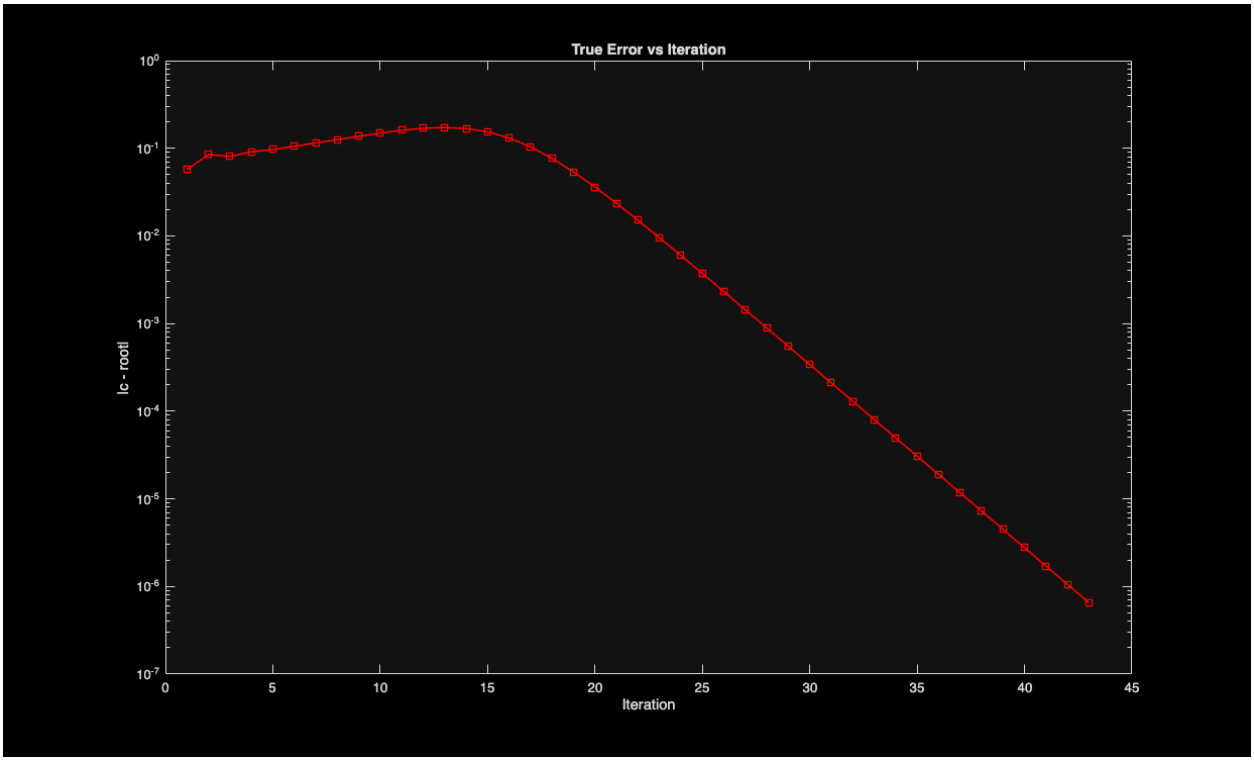
Part 2 (Processing)

```
old_err = 0;

fprintf('++++++ SECANT METHOD ++++++\n\n')
while rel_err > tol
    x2 = x1 - f(x1)*(x1 - x0)/(f(x1) - f(x0)) ;
    rel_err = abs((x2 - x1)/x2);
    x0 = x1; x1 = x2;
    k = k + 1;
    it_s(k)  = k;
    err_s(k) = rel_err;
    fprintf('Loop %d  |  c = %.6f\n', k, x2);
end

semilogy(it_s, err_s, 'r-s','LineWidth',1.2); hold on
xlabel('Iteration'); ylabel('|c - root|')
title('True Error vs Iteration')

R = arrayfun(@(j) log(err_s(j)/err_s(j-1))/ ...
                  log(err_s(j-1)/err_s(j-2)), 3:numel(err_s));
fprintf('\nEstimated order (Secant): %.4f\n\n', mean(R));
```

```
% --------- Newton Method ----------
x0 = 10; tol = 1e-6;
syms x
df = matlabFunction(diff((x-1)*(exp(x-1)-1)));

rel_err = inf; k = 0;
it_n = []; err_n = [];

fprintf('++++++ NEWTON METHOD ++++++\n\n')
while rel_err > tol
    x1 = x0 - f(x0)/df(x0);
    rel_err = abs((x1 - x0)/x1);
    x0 = x1;
    k = k + 1;
    it_n(k)  = k;
    err_n(k) = rel_err;
    fprintf('Loop %d  |  c = %.6f\n', k, x1);
end
```

```
++++++ SECANT METHOD ++++++

Loop 1  |  c = 8.514260
Loop 2  |  c = 7.849599
Loop 3  |  c = 7.263247
Loop 4  |  c = 6.657108
Loop 5  |  c = 6.070337
Loop 6  |  c = 5.490465
Loop 7  |  c = 4.925553
Loop 8  |  c = 4.377068
Loop 9  |  c = 3.850557
Loop 10  |  c = 3.352014
Loop 11  |  c = 2.889406
Loop 12  |  c = 2.471689
Loop 13  |  c = 2.107762
Loop 14  |  c = 1.804321
Loop 15  |  c = 1.563650
Loop 16  |  c = 1.382442
Loop 17  |  c = 1.252555
Loop 18  |  c = 1.163307
Loop 19  |  c = 1.103994
Loop 20  |  c = 1.065527
Loop 21  |  c = 1.040999
Loop 22  |  c = 1.025536
Loop 23  |  c = 1.015859
Loop 24  |  c = 1.009831
Loop 25  |  c = 1.006087
Loop 26  |  c = 1.003766
Loop 27  |  c = 1.002329
Loop 28  |  c = 1.001440
Loop 29  |  c = 1.000890
Loop 30  |  c = 1.000550
Loop 31  |  c = 1.000340
Loop 32  |  c = 1.000210
```

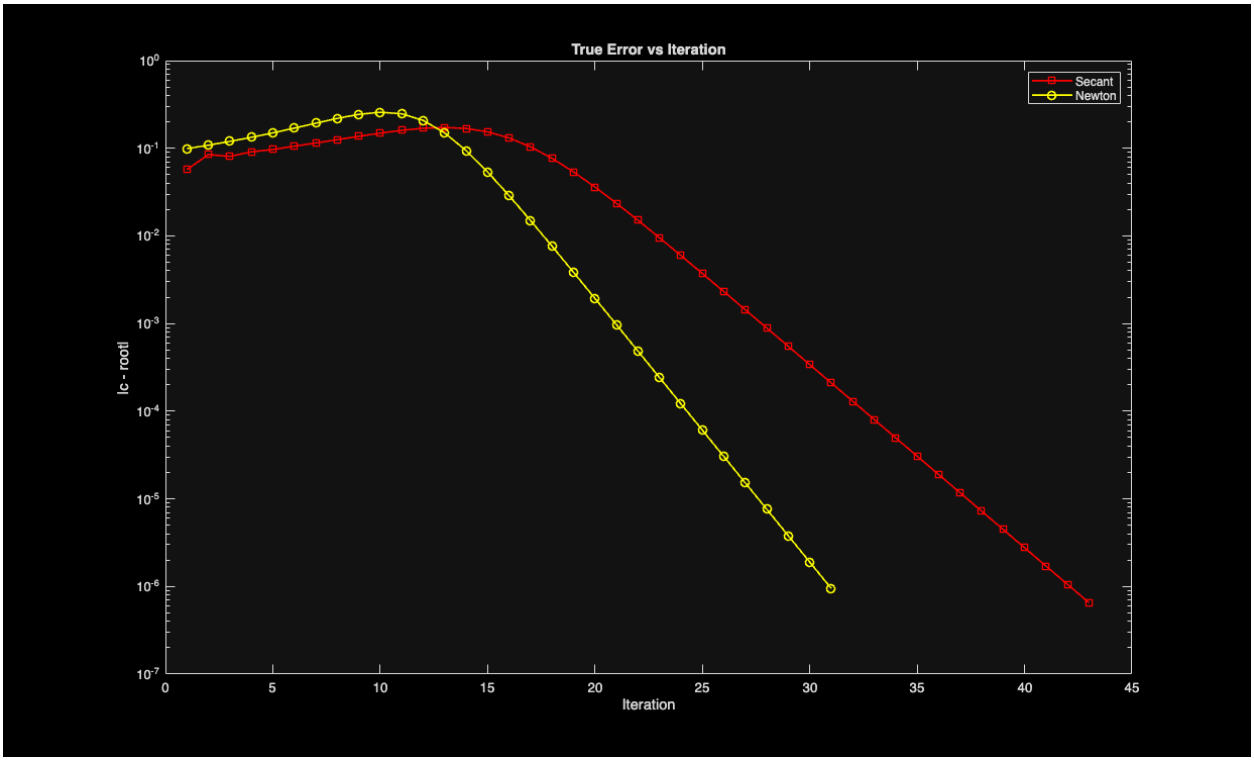
```
Loop 33  |  c = 1.000130
Loop 34  |  c = 1.000080
Loop 35  |  c = 1.000050
Loop 36  |  c = 1.000031
Loop 37  |  c = 1.000019
Loop 38  |  c = 1.000012
Loop 39  |  c = 1.000007
Loop 40  |  c = 1.000004
Loop 41  |  c = 1.000003
Loop 42  |  c = 1.000002
Loop 43  |  c = 1.000001

Estimated order (Secant): 0.9143

++++++ NEWTON METHOD ++++++

Loop 1  |  c = 9.100100
Loop 2  |  c = 8.210229
Loop 3  |  c = 7.332599
Loop 4  |  c = 6.470302
Loop 5  |  c = 5.627865
Loop 6  |  c = 4.812174
Loop 7  |  c = 4.033914
Loop 8  |  c = 3.309364
Loop 9  |  c = 2.661401
Loop 10  |  c = 2.116820
Loop 11  |  c = 1.697003
Loop 12  |  c = 1.405207
Loop 13  |  c = 1.222372
Loop 14  |  c = 1.117247
Loop 15  |  c = 1.060324
Loop 16  |  c = 1.030615
Loop 17  |  c = 1.015424
Loop 18  |  c = 1.007742
Loop 19  |  c = 1.003878
Loop 20  |  c = 1.001941
Loop 21  |  c = 1.000971
Loop 22  |  c = 1.000486
Loop 23  |  c = 1.000243
Loop 24  |  c = 1.000121
Loop 25  |  c = 1.000061
Loop 26  |  c = 1.000030
Loop 27  |  c = 1.000015
Loop 28  |  c = 1.000008
Loop 29  |  c = 1.000004
Loop 30  |  c = 1.000002
Loop 31  |  c = 1.000001
```



Part 3 (Post Processing / Plotting)

```
semilogy(it_n, err_n, 'y-o','LineWidth',1.2)
legend('Secant','Newton','Location','northeast')

R = arrayfun(@(j) log(err_n(j)/err_n(j-1))/ ...
                  log(err_n(j-1)/err_n(j-2)), 3:numel(err_n));
fprintf('\nEstimated order (Newton): %.4f\n', mean(R));
```

Estimated order (Newton): 1.1094



Published with MATLAB® R2025a

Table of Contents

Assignemnt - 4

Bisection and Regula Falsi

```
% Name        : Niraj Kumar Singh
% Roll Number : AM25M807
% Department  : Applied Mechanics
```

Part 1 (Preprocessing)

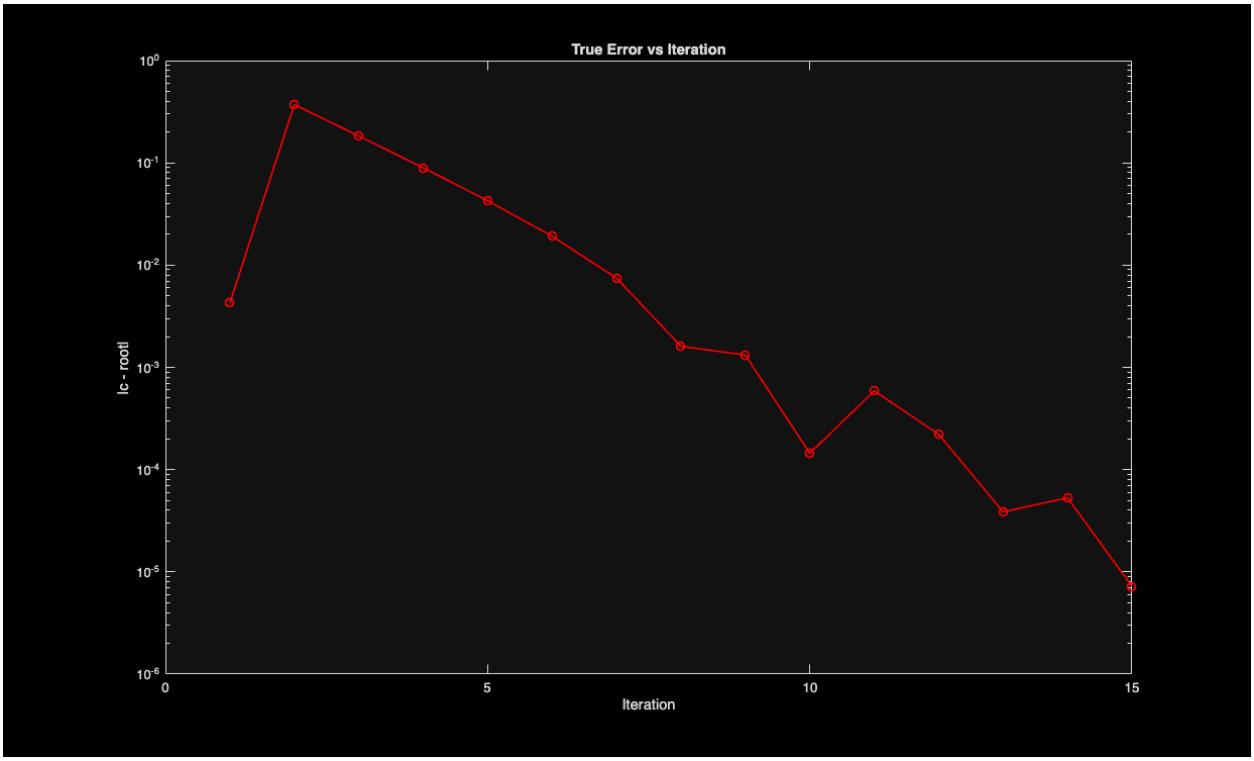
```
clc; clear; close all;
f  = @(x) sin(10*x) + cos(3*x);
tol = 1e-4;
```

Part 2 (Processing)

```
% Bisection for root near 3.74575
a = 3;  b = 6;  r1 = 3.74575;
if f(a)*f(b) > 0, error('Bad bracket'); end

k = 0;  it1 = []; err1 = [];
c = (a+b)/2;
while abs(f(c)) > tol
    if f(a)*f(c) < 0, b = c; else, a = c; end
    c = (a+b)/2;  k = k + 1;
    it1(k)  = k;
    err1(k) = abs(c - r1);
    fprintf('Bisection %2d  c=%.6f\n', k, c);
end

semilogy(it1, err1, 'r-o','LineWidth',1.2); hold on
xlabel('Iteration'); ylabel('|c - root|')
title('True Error vs Iteration')

R = arrayfun(@(j) log(err1(j)/err1(j-1))/ ...
                  log(err1(j-1)/err1(j-2)), 3:numel(err1));
fprintf('\nBisection order ≈ %.4f\n\n', mean(R));


% Regula-Falsi for root near 5.67903
a = 3;  b = 6;  r2 = 5.67903;
if f(a)*f(b) > 0, error('Bad bracket'); end
```

```
k = 0;  it2 = []; err2 = [];
c = b - f(b)*(b - a)/(f(b) - f(a));
while abs(f(c)) > tol
    if f(a)*f(c) < 0, b = c; else, a = c; end
    c = b - f(b)*(b - a)/(f(b) - f(a));
    k = k + 1;
    it2(k)  = k;
    err2(k) = abs(c - r2);
    fprintf('Regula-Falsi %2d  c=%.6f\n', k, c);
end
```

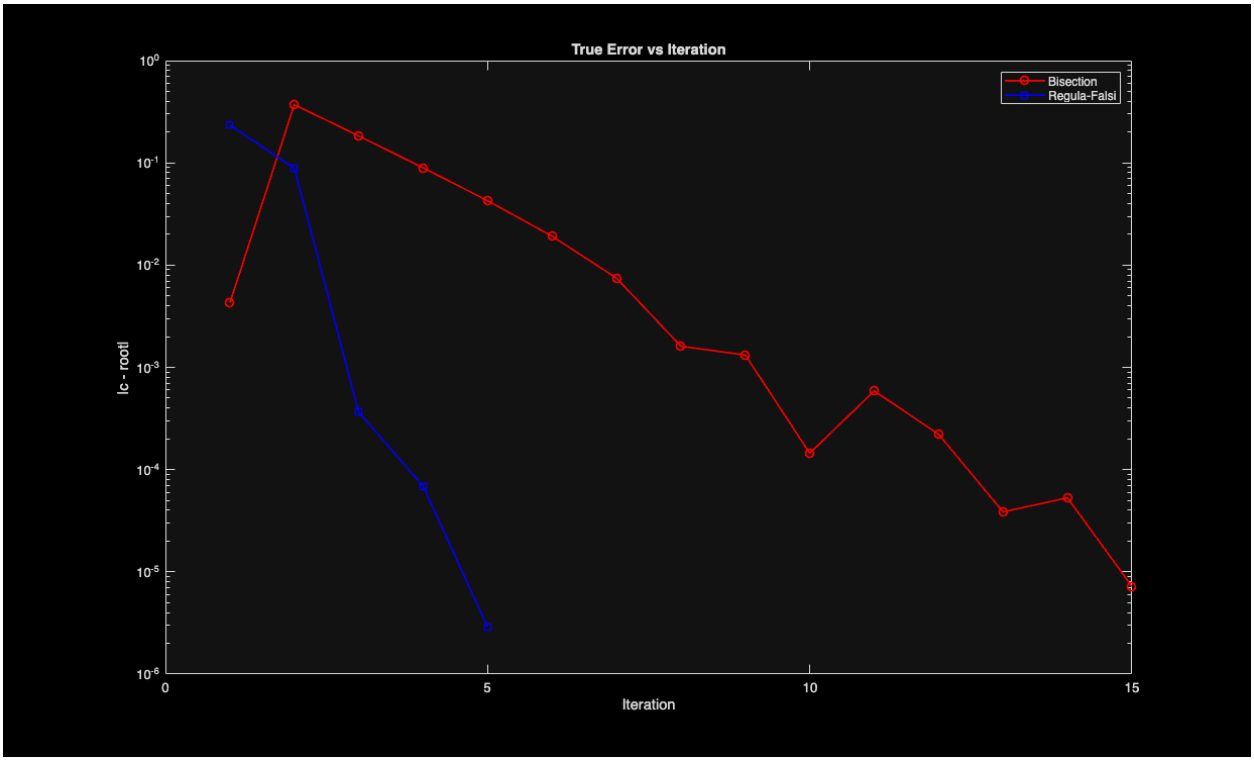
```
Bisection  1  c=3.750000
Bisection  2  c=3.375000
Bisection  3  c=3.562500
Bisection  4  c=3.656250
Bisection  5  c=3.703125
Bisection  6  c=3.726562
Bisection  7  c=3.738281
Bisection  8  c=3.744141
Bisection  9  c=3.747070
Bisection 10  c=3.745605
Bisection 11  c=3.746338
Bisection 12  c=3.745972
Bisection 13  c=3.745789
Bisection 14  c=3.745697
Bisection 15  c=3.745743

Bisection order ≈ 0.8519

Regula-Falsi  1  c=5.914017
Regula-Falsi  2  c=5.767982
Regula-Falsi  3  c=5.679394
Regula-Falsi  4  c=5.678961
Regula-Falsi  5  c=5.679033
```



Part 3 (Post Processing / Plotting)

```
semilogy(it2, err2, 'b-s','LineWidth',1.2)
legend('Bisection','Regula-Falsi','Location','northeast')

R = arrayfun(@(j) log(err2(j)/err2(j-1))/ ...
                  log(err2(j-1)/err2(j-2)), 3:numel(err2));
fprintf('\nRegula-Falsi order ≈ %.4f\n', mean(R));
```

Regula-Falsi order ≈ 2.6215



Published with MATLAB® R2025a