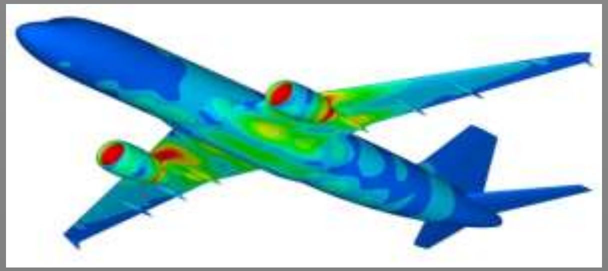
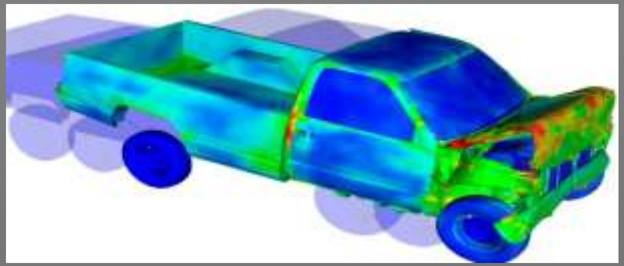
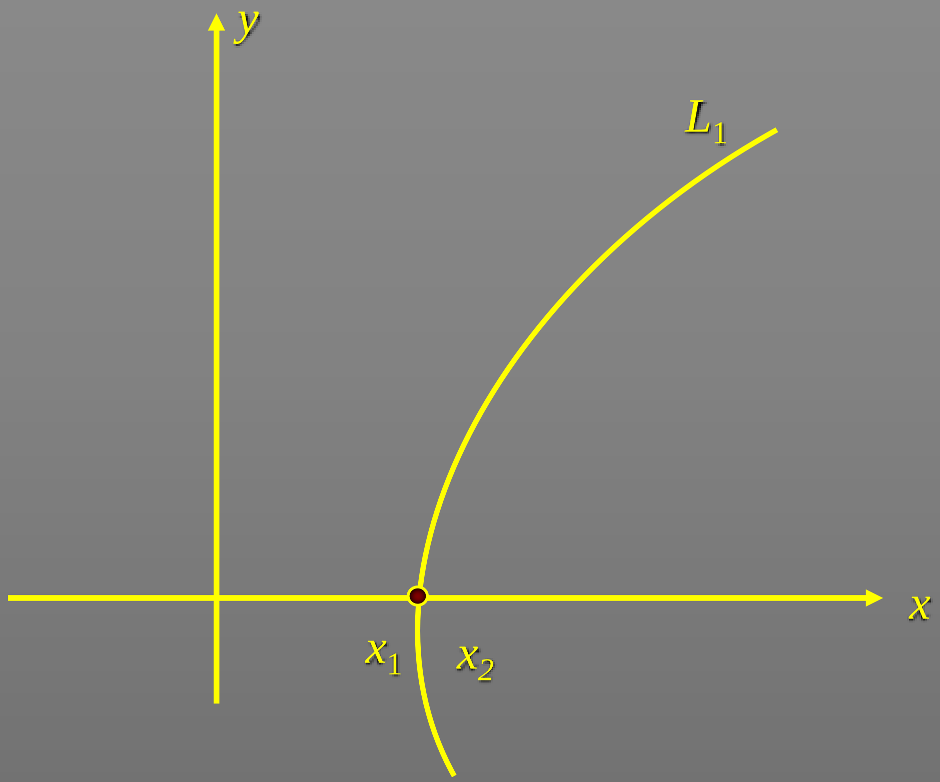
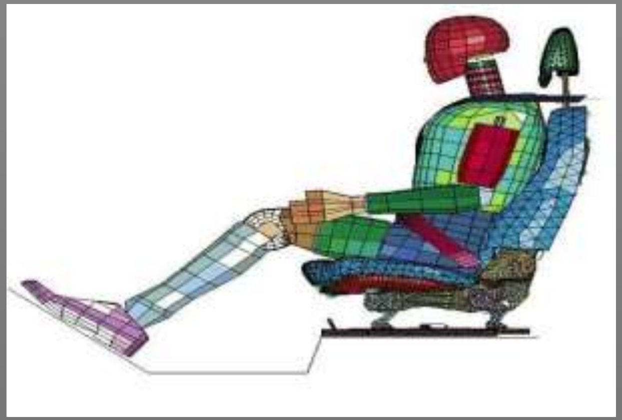
Non-linear Equations: Roots Finding









Dr. Himanshu Pathak
himanshu@iitmandi.ac.in

Roots/Zeros of Equations

- Recall that a second order polynomial may be written in the general form

$$ax^2 + bx + c = 0$$

  where a, b, c, are real numbers.

- Root of this equation can b directly found as:

$$\boxed{x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

- Find roots for equation $\boxed{f(x) = e^{-x} - x}$ Tedious!!!

  - Objective is to find a solution of f(x) = 0
    "f(x)" is a polynomial or a transcendental function, given explicitly.

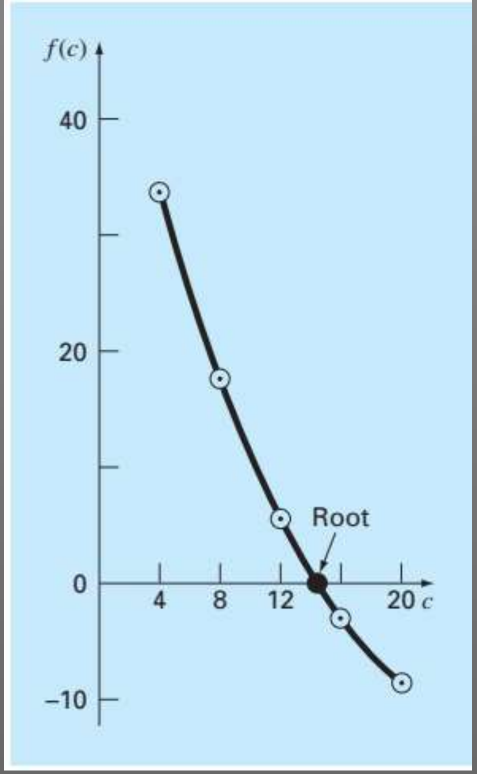
Graphical Method

- Estimate of the root of the equation f(x) = 0.
  1. Make a plot of the function and observe where it crosses the x axis.
  2. The *x* value for which f(x) =0, provides a rough approximation of the root.

Example 1: Get the root for the given equation by Graphical method using the parameters t = 10, g = 9.81, v = 40, and m = 68.1

$$f(c) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) - v$$

c	f(c)
4	34.190
8	17.712
12	6.114
16	−2.230
20	−8.368

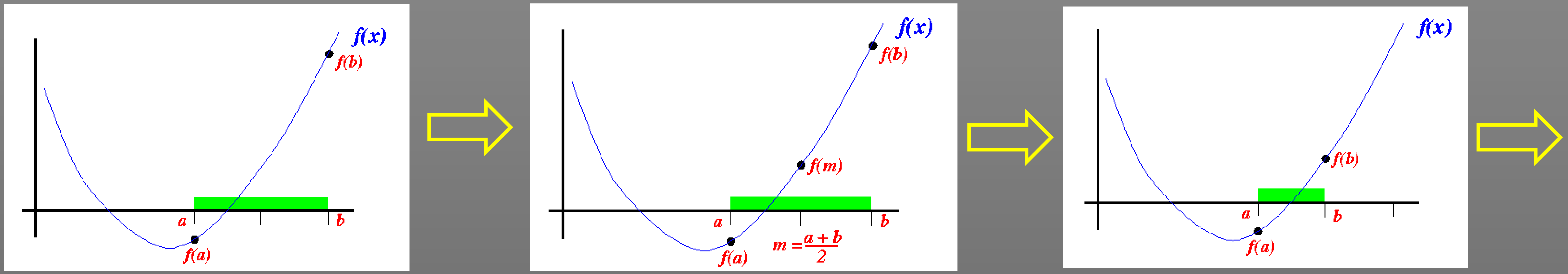


Root solution

$c \approx 14.75$

Bisection Method

- A successive approximation method that narrows down an interval that contains a root of the function f(x).
- Cuts the interval into 2 halves and check which half interval contains a root of the function



Bisection Method

Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval.

$f(x_l)f(x_u) < 0.$

Step 2: An estimate of the root x_m as $x_m = (x_l + x_u)/2$

Step 3: Make the following evaluations to determine in which subinterval the root lies:

(a) If $f(x_l)f(x_m) < 0$, the root lies in the lower subinterval.

Set $x_u = x_m$ and return to step 2.

(b) If $f(x_l)f(x_m) > 0$, the root lies in the upper subinterval.

Set $x_l = x_m$ and return to step 2.

(c) If $f(x_l)f(x_m) = 0$, the root equals x_m;

Terminate the computation.

Bisection Method

Example 1: find the root of $f(x) = x^2 - 5$

Initial Range $x_l = 0; f(x_l) = -5$

$x_u = 4; f(x_u) = 11$

Iteration 1: $x_m = (x_l + x_u)/2 = (0+4)/2 = 2$, $f(x_m) = -1$,

As $f(x_l)f(x_m) > 0$; **Set** $x_l = x_m = 2$

Iteration 2: $x_m = (x_l + x_u)/2 = (2+4)/2 = 3$, $f(x_m) = 4$,

As $f(x_l)f(x_m) < 0$; **Set** $x_u = x_m = 3$

Iteration 3: $x_m = (x_l + x_u)/2 = (2+3)/2 = 2.5$, $f(x_m) = 1.25$,

As $f(x_l)f(x_m) < 0$; **Set** $x_u = x_m = 2.5$

Iteration 4: $x_m = (x_l + x_u)/2 = (2+2.5)/2 = 2.25$, $f(x_m) = 0.0625$,

As $f(x_l)f(x_m) < 0$; **Set** $x_u = x_m = 2.25$

Iteration 5: $x_m = (x_l + x_u)/2 = (2+2.25)/2 = 2.125$, $f(x_m) = -0.484$,

As $f(x_l)f(x_m) > 0$; **Set** $x_l = x_m = 2.125$

Iteration 6: $x_m = (x_l + x_u)/2 = (2.125+2.25)/2 = 2.187$, $f(x_m) = -0.217$,

As $f(x_l)f(x_m) > 0$; **Set** $x_l = x_m = 2.187$

MATLAB© Script

MATLAB Program for Bisection Method

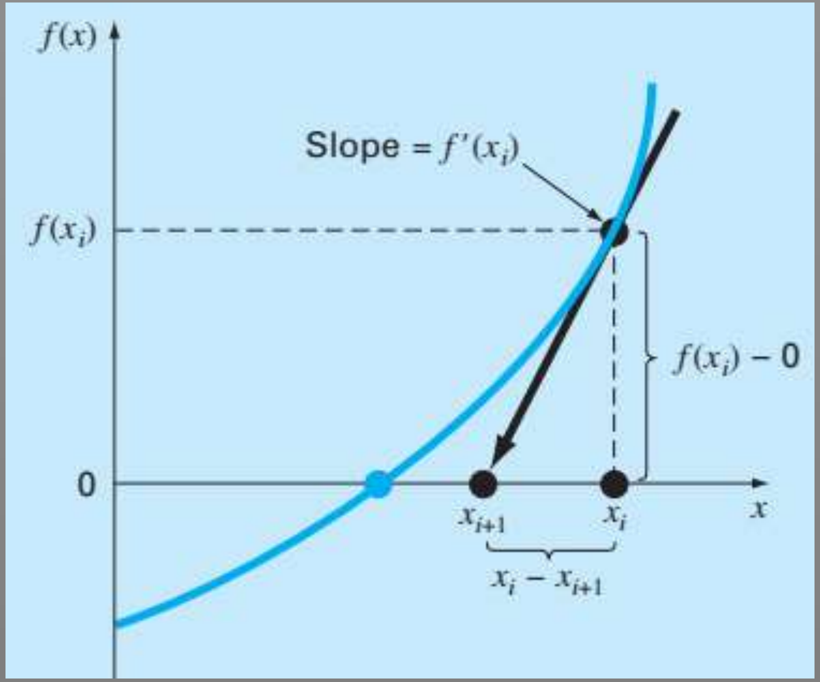
```
%Define function file
function y =bifun(x)

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

```
% Bisection computation
tol = 1e-4;
xl =  0 ; % lower limit
xu = 11; % upper limit
i =1;
maxitr =1000;
while (xu – xl) > 2*tol
        xm = (xu+xl)/2;
        fl = bifun(xl);
        fm = bifun(xm);
        prod = fl*fm;
 if prod>0
        xl = xm;
else
        xu=xm;
end
if i<maxitr
Root (i) = (xl+xu)/2;
i =i+1;
else
disp('Max iteration
reached without root')
return
end
end
```

Newton Raphson Method

- A faster alternative is to use a numerical rootfinder.
- Only one Guess point is needed.
- Initial guess at the root is x_i, a tangent can be extended from the point $[x_i, f(x_i)]$.
- The point where this tangent crosses the x axis usually represents an improved estimate of the root.



$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}} \Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton Raphson Method

Step 1: Evaluate *f'(x)* symbolically and guess initial root x_i .

Step 2: Calculate new value of the root,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Step 3: Find the absolute relative approximate error as :

$$|\epsilon_a| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100$$

Step 4: Compare the absolute relative approximate error with the pre-specified tolerance $|\varepsilon_s|$

(a) If $|\varepsilon_a| > |\varepsilon_s|$, return to step 2 and calculate new value of root.

(b) If $|\varepsilon_a| < |\varepsilon_s|$, Terminate the computation.

Newton Raphson Method

Example 1: find the root of $f(x) = e^{-x} - x$

Initial Guess $x_0 = 0; f(x_0) = 1,$

$f'(x) = - e^{-x} - 1 , f'(x_0) = -2$

Iterations: $$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$x_1 = x_0 - (f(x_0) /(f'(x_0)) = 0-(-1/2) = 0.5$

$x_2 = x_1 - (f(x_1) /(f'(x_1)) = 0.5-(-0.1/1.6) = 0.566$

i	x_i	ε_t (%)
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$< 10^{-8}$

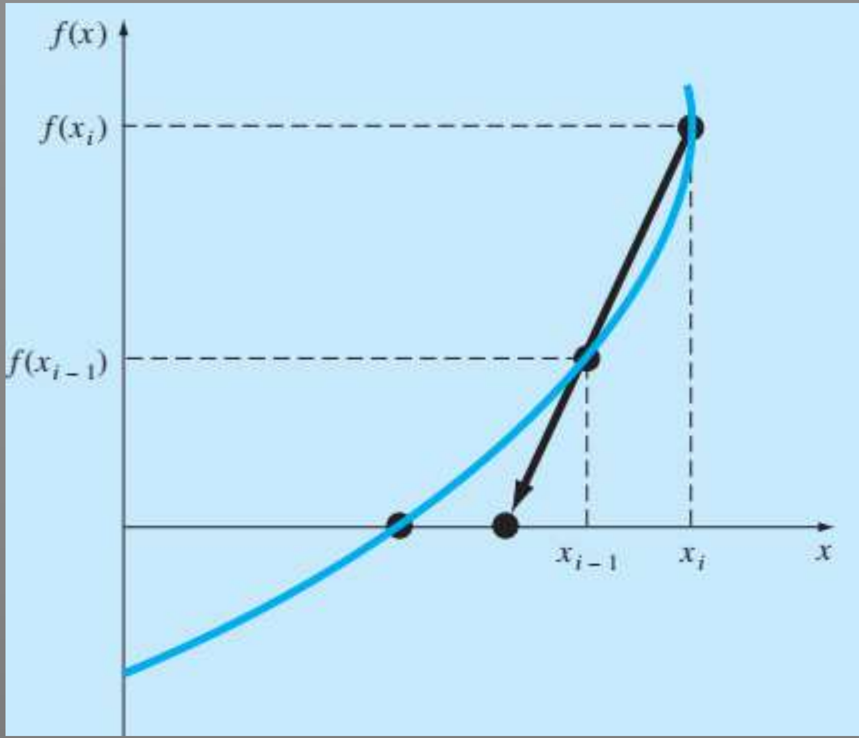
MATLAB© Script

MATLAB Program for Newton Raphson Method

```
clear all
clc
tol = 1e-5;
x0 =  0 ; % Guess
i =1;
maxitr =1000;
xold = x0;
syms z
fun=exp(z) +15*z -10;
fx_diff=diff(fun,z);
while i < maxitr
    fx = subs(fun,z,xold);
    Fx = vpa(fx);
    fx1 = subs(fx_diff, z, xold);
    Fx1 =vpa(fx1);
    x = xold - Fx/Fx1;
    if abs(x-xold) > tol
        xold = x;
        i = i +1;
    else
        disp('Root calculated')
        x
        return
    end
end
```

Secant Method

- A faster alternative is to use a numerical rootfinder.
- Two initial point is needed.
- No need of function derivative calculation.
- Root is predicted by extrapolating tangent of function to x-axis.
- Modified form of Newton-Raphson to avoid function derivative calculation.
- Function derivative approximated by backward finite divided difference.



$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton-Raphson

$$\Rightarrow$$

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

Secant

Secant Method

Step 1: Guess initial root x_{i-1} and x_i .

Step 2: Calculate new value of the root,

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}}$$

Step 3: Find the absolute relative approximate error as :

$$|\epsilon_a| = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100$$

Step 4: Compare the absolute relative approximate error with the pre-specified tolerance $|\varepsilon_s|$

(a) If $|\varepsilon_a| > |\varepsilon_s|$, return to step 2 and calculate new value of root.

(b) If $|\varepsilon_a| < |\varepsilon_s|$, Terminate the computation.

MATLAB© Script

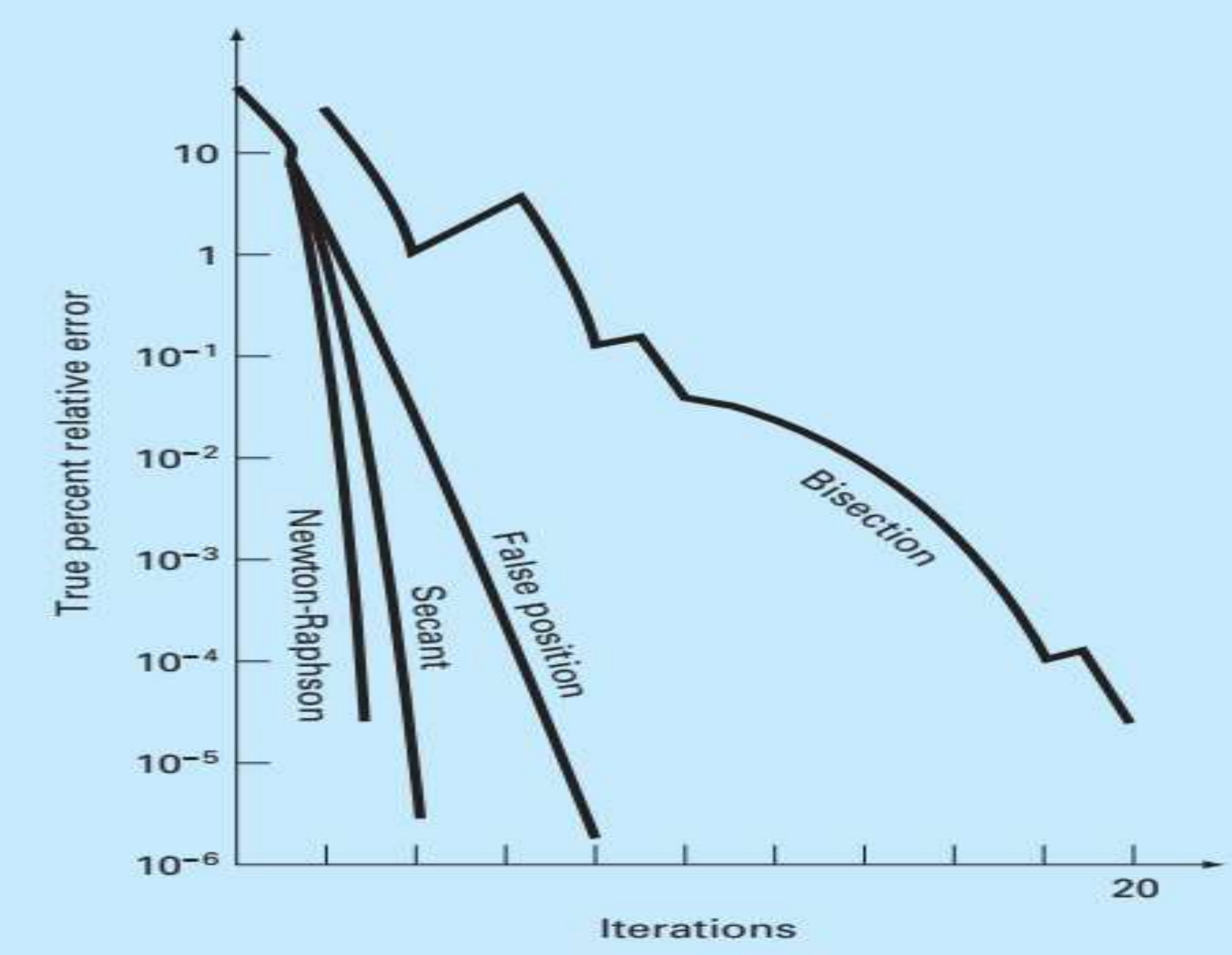
MATLAB Program for Secant Method

```
%Define function file

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

```
clear all
clc
tol = 1e-5;
x = [1 5];  % Guess
i =3;
maxitr =1000;
while i < maxitr
    f1 = secfun(x(i-2));
    f2 = secfun(x(i-1));
    fact = (f2*(x(i-2)-x(i-1)))/(f1-f2);
    x(i) = x(i-1) - fact;
  if abs(x(i)-x(i-1)) < tol
      disp('Root calculated')
      x(i)
      i
      return
   else
    i = i +1;
   end
end
```

Convergence



THANK YOU

Questions??