MC3010: Differential Equations & Numerical Methods

Lecturers:
Dr. Padmanathan Kathirgamanathan & Mrs. Saranja Kiriparan

Department of Interdisciplinary Studies, Faculty of Engineering, University of Jaffna, Sri Lanka

NUMERICAL COMPUTATION

Lecture – 02: Solutions of Non-Linear Equations (Roots of Equations)

Department of Interdisciplinary Studies, Faculty of Engineering, University of Jaffna, Sri Lanka

Introduction

- Objective is to find a solution of ***F(x) = 0***

  Where *F* is a polynomial or a transcendental function, given explicitly.

- Exact solutions are not possible for most equations.

- A number x ± e, (e > 0) is an approximate solution of the equation if there is a solution in the interval [x-e,x+e]. e is the maximum possible error in the approximate solution.

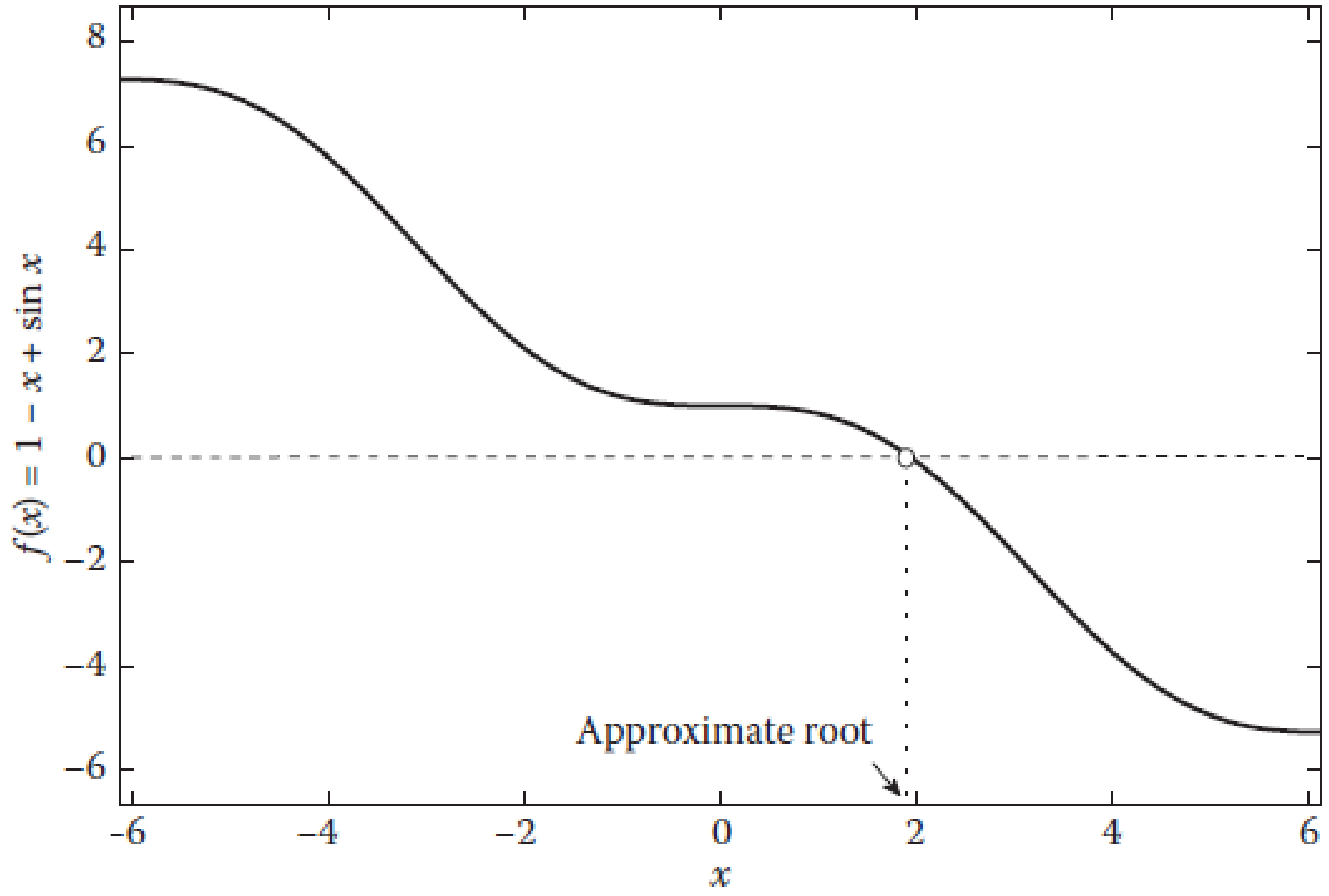
- With unlimited resources, it is possible to find an approximate solution with arbitrarily small e.

Solution for Single variable Equations

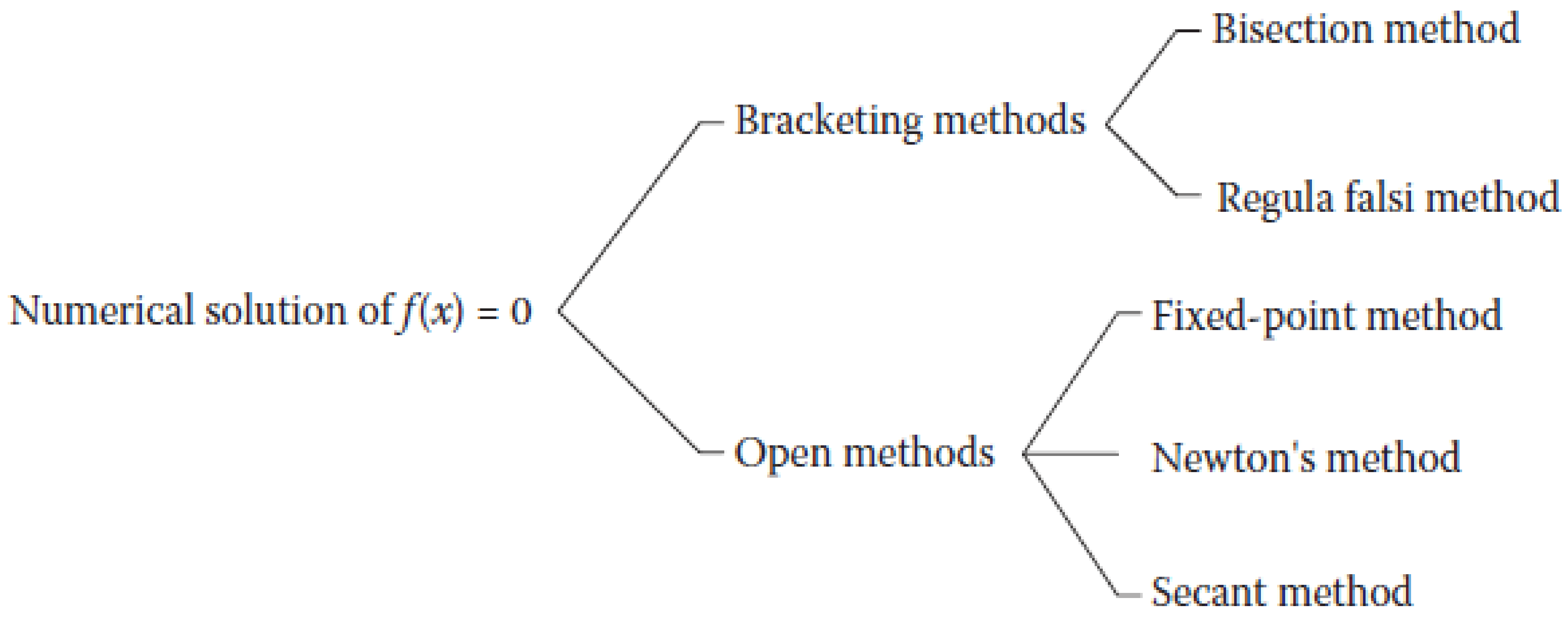
The focus of this lecture is Numerical Solution of equations in the general form

$$f(x) = 0$$

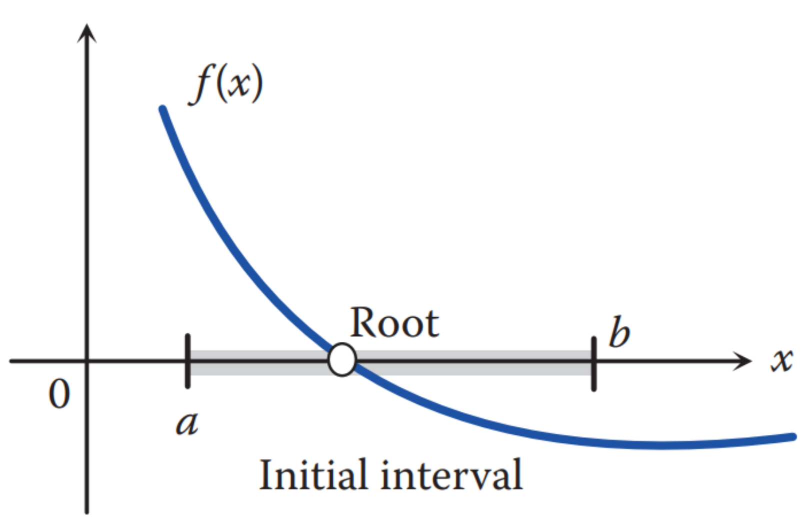
Graphically, a solution, or a root of above equation refer to the point of intersection of $f(x)$ and the x-axis. Therefore, depending on the nature of the curve of $f(x)$ in relation to the x-axis, the equation may have a unique solution, multiple solutions, or no solution. A root of an equation can sometimes be determined analytically resulting in an exact solution.

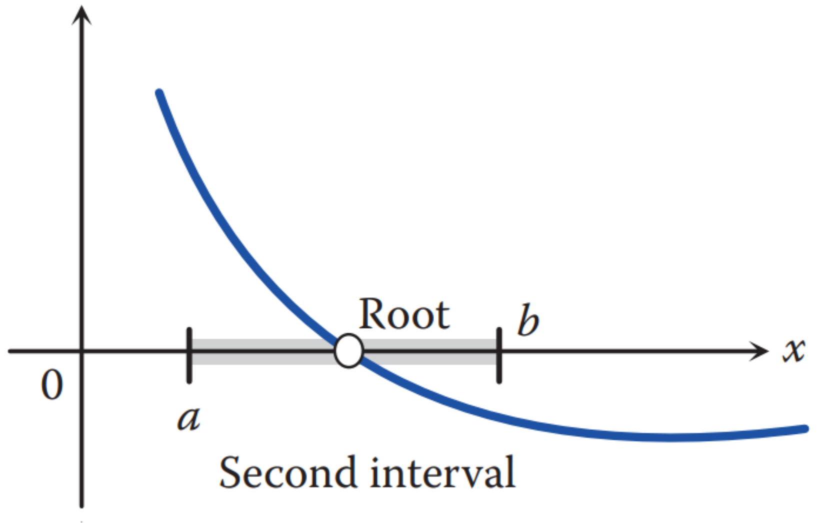

$f(x) = 1 - x + \sin x$
x
Approximate root
8
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

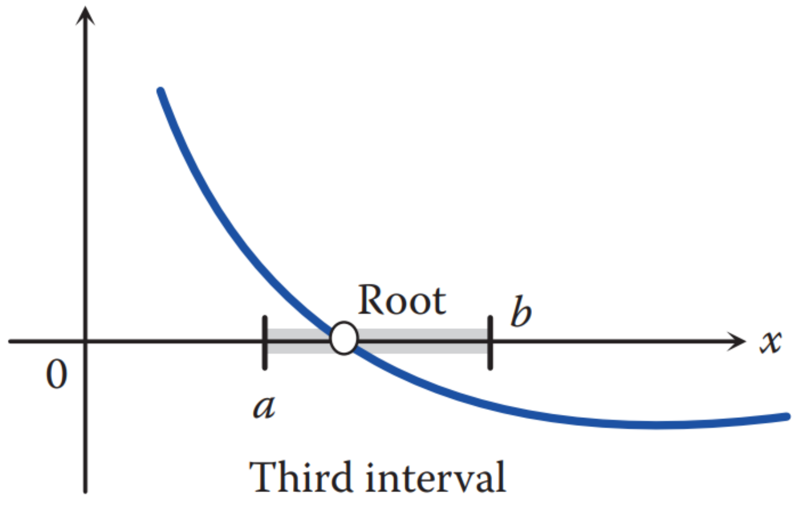
Classification of methods to solve an Equation of one variable



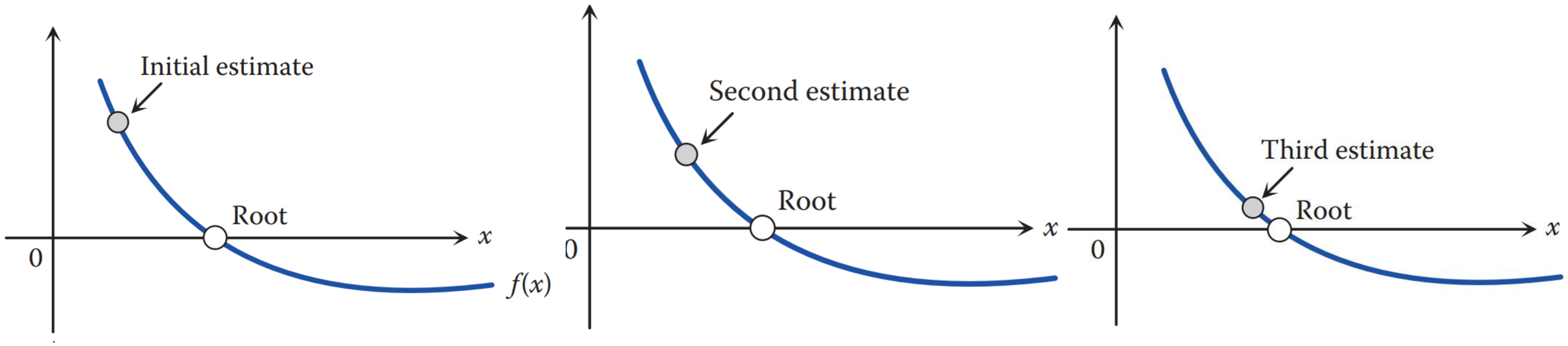
Philosophy of Bracketing methods







Philosophy of Open Methods



1. Bracketing Methods

Bracketing methods involve finding a bracket or an interval containing the root.

The root is assumed to be located within this interval, and the algorithm proceeds to narrow down the interval until the root is found.

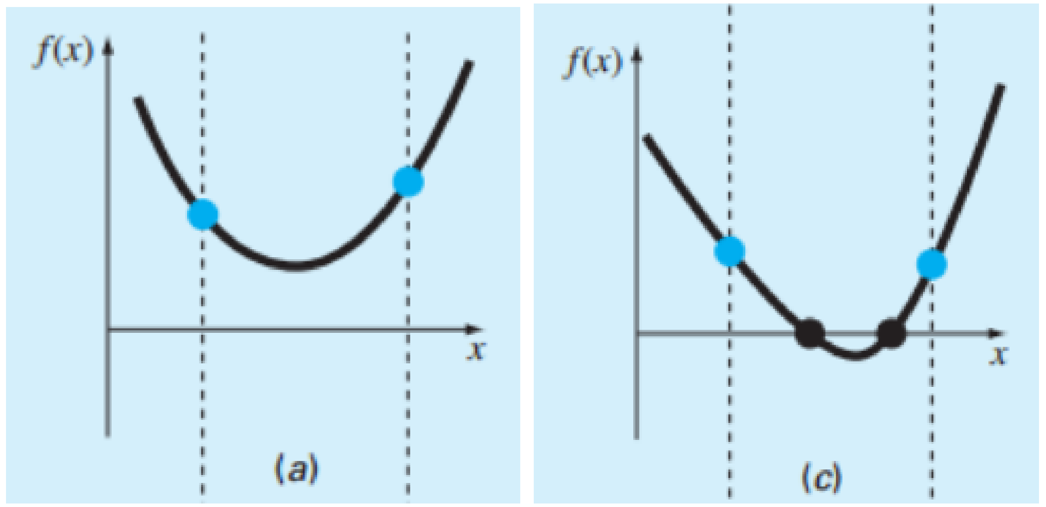
We will cover:

1. **Bisection method**
2. **False position method**

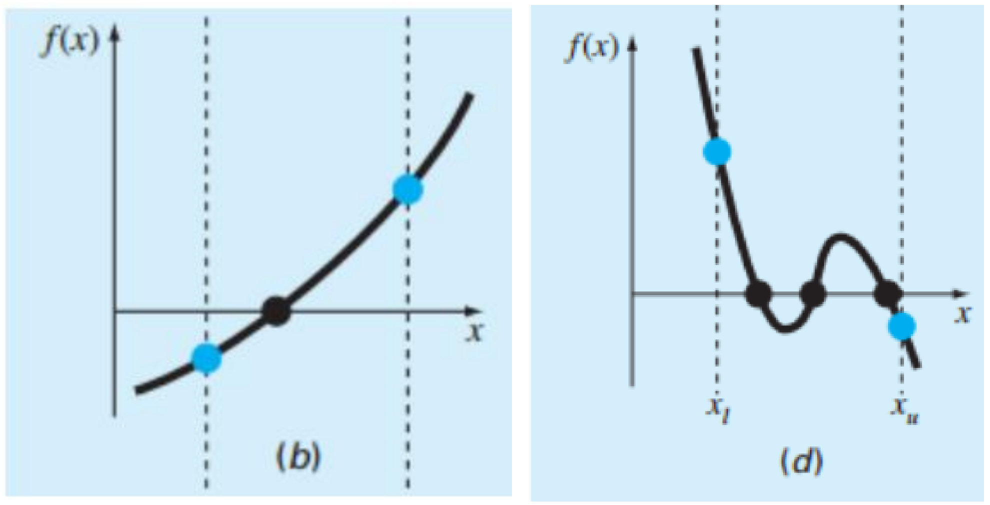
Bracketing methods are generally slower than the open methods.

Convergence to the solution

- Bracketing methods guarantee convergence to a root if the **function is continuous** and **changes sign within the interval.**
- The bracket condition means that we must know two points a, b where the function has different signs, or $f(a)f(b) < 0$.



$f(a)f(b) > 0$



$f(a)f(b) < 0$

1.1 Bisection Methods

- It is a simplest bracketing method to find a root of f(x) = 0.
- The bisection method involves first selecting two initial values of the independent variable (usually x) such that the function changes sign between the two values. These two values are referred to as the "bracketing values" or the "bracket". Then, the method repeatedly divides the interval defined by the bracketing values in half, and checks which half contains a root. The process is repeated until the desired level of accuracy is achieved.

Algorithm for the Bisection Method

Step 1:

Choose the first interval $[a, b]$ and verify the solution exists within selected interval. i.e., $f(a)$ *and* $f(b)$ have different signs such that $f(a)f(b) < 0$. The points can be determined by examining the plot of $f(x)$ versus x.

Step 2:

Calculate the first estimate of the numerical solution $x_1 = \frac{a+b}{2}$.

Step 3:

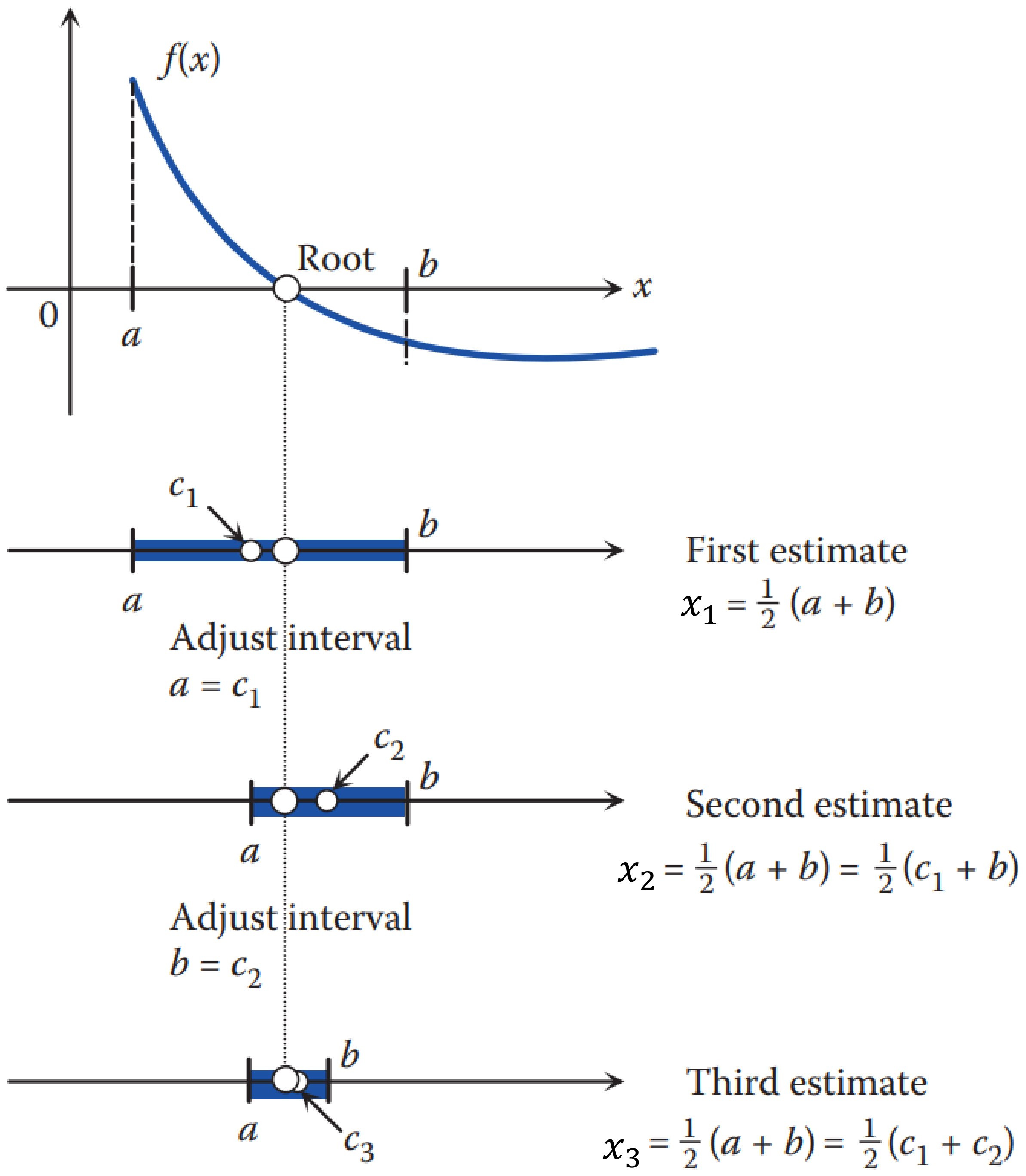
Determine whether the true solution is between a and x_1 or between x_1 and b. This is done by checking the sign of the product $f(a)f(x_1)$:

If $f(a)f(x_1) < 0$, the true solution is between a and x_1.

If $f(a)f(x_1) > 0$, the true solution is between $x_1$and b.

Step 4:

Select the subinterval that contains the true solution as the new interval $[a, b]$, and go back to Step 2 to 4 and repeat until a specified tolerance or error bound is attained.



Example - 01

Use the bisection method to find the root of the equation $f(x) = x^3 - 2x - 5$ in the interval [2, 3].
Find the root to within an absolute error tolerance of 0.01.

Solution using MATLAB or Calculator

$$f(x) = x^3 + 0x^2 - 2x - 5$$

```
>> p = [1 0 -2 -5];
>> r = roots(p)

r =

   2.0946 + 0.0000i
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i
```

Solution using Bisection

Step 1: Verify the solution exists within interval [2, 3]. i.e., $f(a)f(b) < 0$

- f(2) = -1 & f(3) = 16,
- $f(2)f(3) < 0$

So, the function does change in sign.

- $\text{Error} = \frac{b-a}{2} = \frac{3-2}{2} = 0.5 > 0.01$

Step 2: Calculate the first estimate, x_1 by: $x_1 = \frac{a+b}{2}$;

Set the interval [a, b] to [2, 3] and tolerance to 0.01.

- $x_1 = \frac{2+3}{2} = 2.5$
- $f(2.5) = 2.5^3 - 2 * 2.5 - 5 = 5.625$

Step 3: Determine whether the true solution is between $[a, x_1]$ or $[x_1, b]$

- $f(2)f(2.5) < 0$, the true solution is between 2 and 2.5.
- $\text{Error} = \frac{b-a}{2} = \frac{2.5-2}{2} = 0.25 > 0.01$

Step 4: Select the subinterval that contains the true solution as the new interval and repeat step 2 and step 3.

Set the interval to [a, b] = [2, 2.5]

- $x_2 = \frac{2+2.5}{2} = 2.25$
- $f(2.25) = 2.25^3 - 2 * 2.25 - 5 = -0.796875$
- $f(2)f(2.25) > 0$, the true solution is between 2.25 and 2.5
- $\text{Error} = \frac{b-a}{2} = \frac{2.5-2.25}{2} = 0.125 > 0.01$

Continue this process, halving the interval each time until the absolute error is less than or equal to 0.01.

$$f(x) = x^3 - 2x - 5$$

Iteration	$f(a)f(b) < 0$	Interval a	Interval b	x_n ; $x_n = \frac{a+b}{2}$	$f(x_n)$	Tolerance (0.01) ; $\frac{b-a}{2}$
1	$f(2)f(3) < 0$	2 ; f(2) = -1	3 ;f(3) = 16	2.5	5.625	0.50000
2	$f(2)f(2.5) < 0$	2	2.5	2.25	1.890625	0.25000
3	$f(2)f(2.25) < 0$	2	2.25	2.125	0.345703	0.12500
4	$f(2)f(2.125) < 0$	2	2.125	2.0625	-0.351318	0.06250
5	$f(2)f(2.0625) > 0$	2.0625	2.125	2.09375	-0.00894165	0.03125
6	$f(2.0625)f(2.09375) > 0$	2.09375	2.125	2.109375	0.166835785	0.01563
7	$f(2.09375)f(2.109375) < 0$	2.09375	2.109375	2.1015625		0.00781 < 0.01

Thus, we can conclude that the root of the equation $f(x) = x^3 - 2x - 5$ in the interval [2, 3] to within an absolute error tolerance of 0.01 is approximately x = 2.1015625.

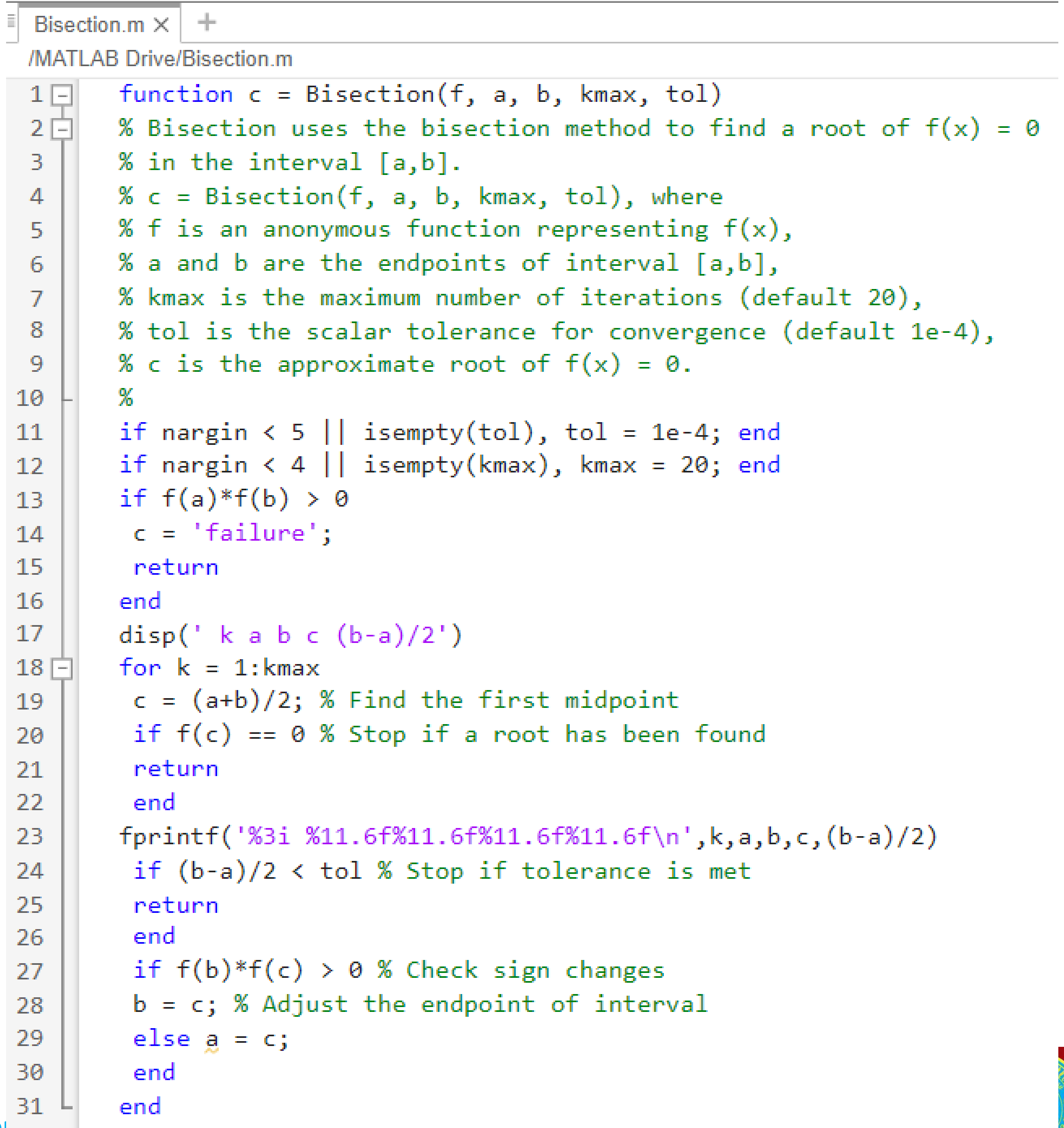
Solving f(x) = 0 using MATLAB (Bisection)

Use the bisection method to find the root of the equation $f(x) = x^3 - 2x - 5$ in the interval [2, 3].
Find the root to within an absolute error tolerance of 0.01.

```
>>  f = @(x)(x^3 -2*x-5);
>> c = Bisection(f, 2, 3, [], 1e-2)
 k a b c (b-a)/2
  1     2.000000   3.000000   2.500000   0.500000
  2     2.000000   2.500000   2.250000   0.250000
  3     2.000000   2.250000   2.125000   0.125000
  4     2.000000   2.125000   2.062500   0.062500
  5     2.062500   2.125000   2.093750   0.031250
  6     2.093750   2.125000   2.109375   0.015625
  7     2.093750   2.109375   2.101562   0.007812

c =

    2.1016
```


Bisection.m ×

/MATLAB Drive/Bisection.m

```matlab
function c = Bisection(f, a, b, kmax, tol)
% Bisection uses the bisection method to find a root of f(x) = 0
% in the interval [a,b].
% c = Bisection(f, a, b, kmax, tol), where
% f is an anonymous function representing f(x),
% a and b are the endpoints of interval [a,b],
% kmax is the maximum number of iterations (default 20),
% tol is the scalar tolerance for convergence (default 1e-4),
% c is the approximate root of f(x) = 0.
%
if nargin < 5 || isempty(tol), tol = 1e-4; end
if nargin < 4 || isempty(kmax), kmax = 20; end
if f(a)*f(b) > 0
 c = 'failure';
 return
end
disp(' k a b c (b-a)/2')
for k = 1:kmax
 c = (a+b)/2; % Find the first midpoint
 if f(c) == 0 % Stop if a root has been found
 return
 end
fprintf('%3i %11.6f%11.6f%11.6f%11.6f\n',k,a,b,c,(b-a)/2)
 if (b-a)/2 < tol % Stop if tolerance is met
 return
 end
 if f(b)*f(c) > 0 % Check sign changes
 b = c; % Adjust the endpoint of interval
 else a = c;
 end
end
```

Note :-

- If the bisection method is used to approximate the root of f(x) = 0 within a prescribed tolerance ε > 0, then it can be shown that the number N of iterations needed to meet the tolerance condition satisfies;

$$N > \frac{\ln(b-a) - \ln\varepsilon}{\ln 2}$$

In Example – 01:

$f(x) = x^3 - 2x - 5$ in the interval [2, 3]. Find the root to within an absolute error tolerance of 0.01.

$$\begin{matrix} a = 2 \\ b = 3 \\ \varepsilon = 0.01 \end{matrix} \Longrightarrow N > \frac{\ln(3-2) - \ln(0.01)}{\ln 2} \Longrightarrow N > 6.64$$

Solving f(x) = 0 using Bisection (MATLAB)

Write a MATLAB program, in a script file, that determines the solution of the equation $8-4.5(x-\sin x)=0$ by using the bisection method. The solution should have a tolerance of less than 0.001 rad. Create a table that displays the values of a, b, x_{NS}, $f(x_{NS})$, and the tolerance for each iteration of the bisection process.

SOLUTION

To find the approximate location of the solution, a plot of the function $f(x)=8-4.5(x-\sin x)$ is made by using the `fplot` command of MATLAB. The plot (Fig. 3-8), shows that the solution is between $x=2$ and $x=3$. The initial interval is chosen as $a=2$ and $b=3$.

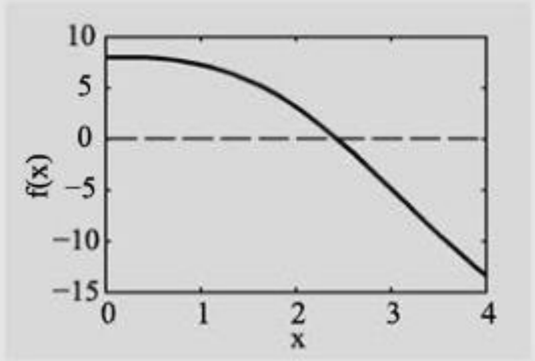


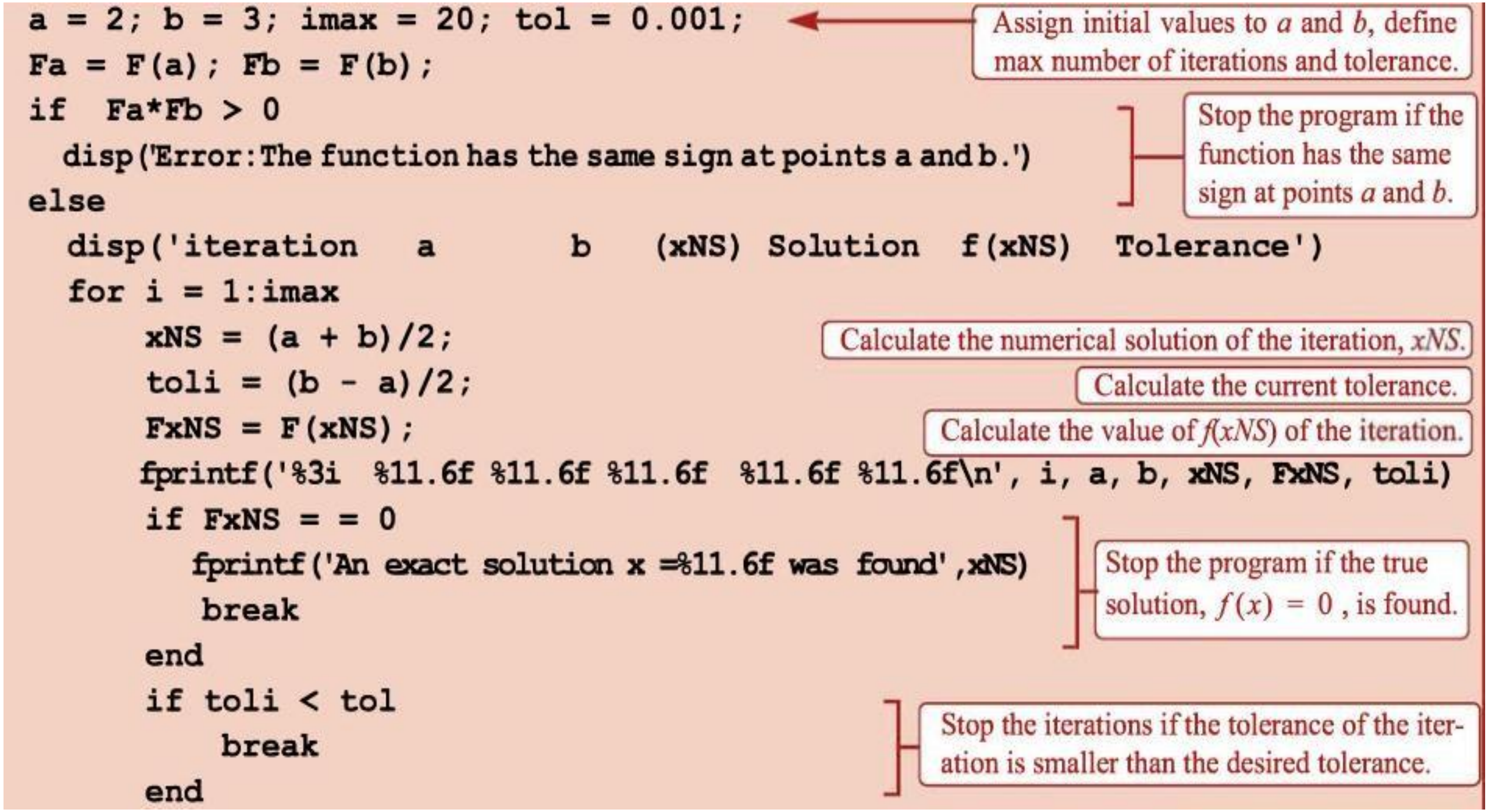
Figure 3-8: A plot of the function $f(x)=8-4.5(x-\sin x)$.

A MATLAB program that solves the problem is as follows.

Program 3-1: Script file. Bisection method.

```
clear all
F = @ (x) 8-4.5*(x-sin(x));
```

Define $f(x)$ as an anonymous function.



```
a = 2; b = 3; imax = 20; tol = 0.001;
Fa = F(a); Fb = F(b);
if  Fa*Fb > 0
  disp('Error:The function has the same sign at points a and b.')
else
   disp('iteration   a       b    (xNS) Solution  f(xNS)  Tolerance')
   for i = 1:imax
       xNS = (a + b)/2;
       toli = (b - a)/2;
       FxNS = F(xNS);
       fprintf('%3i  %11.6f %11.6f %11.6f  %11.6f %11.6f\n', i, a, b, xNS, FxNS, toli)
       if FxNS = = 0
          fprintf('An exact solution x =%11.6f was found',xNS)
          break
       end
       if toli < tol
            break
       end
```

Assign initial values to *a* and *b*, define max number of iterations and tolerance.

Stop the program if the function has the same sign at points *a* and *b*.

Calculate the numerical solution of the iteration, *xNS*.

Calculate the current tolerance.

Calculate the value of *f*(*xNS*) of the iteration.

Stop the program if the true solution, $f(x) = 0$, is found.

Stop the iterations if the tolerance of the iteration is smaller than the desired tolerance.

```
        if F(a)*FxNS < 0
            b = xNS;
        else
            a = xNS;
        end
    end
end
```

Determine whether the true solution is between a and xNS, or between xNS and b, and select a and b for the next iteration.

When the program is executed, the display in the Command Window is:

```
iteration      a            b        (xNS) Solution    f(xNS)      Tolerance
  1        2.000000     3.000000       2.500000      -0.556875     0.500000
  2        2.000000     2.500000       2.250000       1.376329     0.250000
  3        2.250000     2.500000       2.375000       0.434083     0.125000
  4        2.375000     2.500000       2.437500      -0.055709     0.062500
  5        2.375000     2.437500       2.406250       0.190661     0.031250
  6        2.406250     2.437500       2.421875       0.067838     0.015625
  7        2.421875     2.437500       2.429688       0.006154     0.007813
  8        2.429688     2.437500       2.433594      -0.024755     0.003906
  9        2.429688     2.433594       2.431641      -0.009295     0.001953
 10        2.429688     2.431641       2.430664      -0.001569     0.000977
```

The numerical solution.

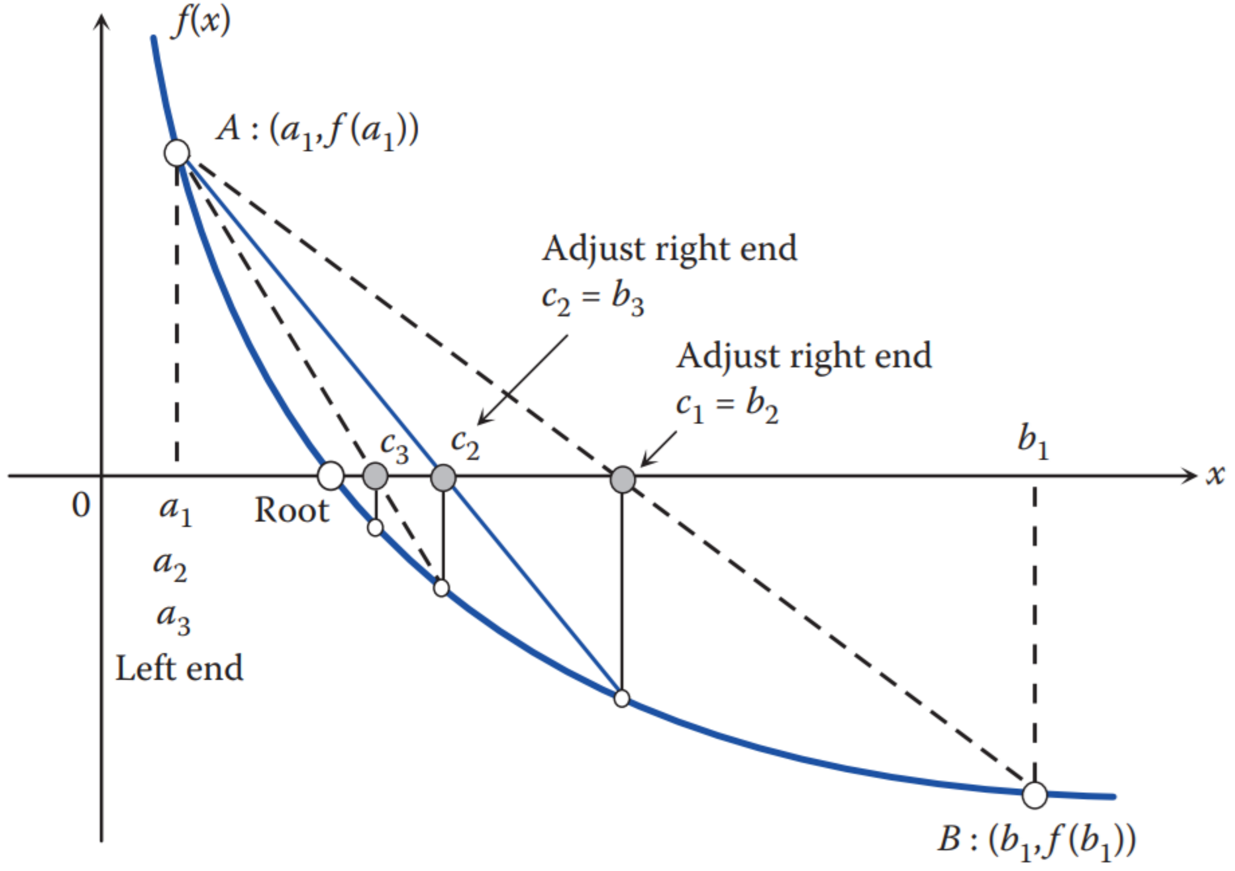
The value of the function at the numerical solution.

The last tolerance (satisfies the prescribed tolerance).

The output shows that the solution with the desired tolerance is obtained in the 10th iteration.

1.2 Regula-Falsi or False Position Method

Regula-Falsi method is used to find the root of an equation by creating a linear interpolation between two initial points and evaluating the function at the midpoint of these points.



Analytically, the procedure is illustrated as follows. The equation of the line connecting points A and B is

$$y - f(b_1) = \frac{f(b_1) - f(a_1)}{b_1 - a_1}(x - b_1)$$

To find the x-intercept, set y = 0 and solve for x = c_1:

$$c_1 = b_1 - \frac{b_1 - a_1}{f(b_1) - f(a_1)} f(b_1) \overset{\text{Simplify}}{=} \frac{a_1 f(b_1) - b_1 f(a_1)}{f(b_1) - f(a_1)}$$

Algorithm for False Position method

1. Set tolerance ϵ, max number of iterations $N_{\max}$, and find bracketing interval (a, b).
2. Compute the intersection point of a chord joining $(a, f(a))$ to $(b, f(b))$ and the x-axis:

$$x^* = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

   This is the current estimate for the true root α.

3. Retain interval over which $f(x)$ changes sign; i.e., replace either a or b with x^*.
4. If the termination criteria is satisfied, i.e $k \geq N_{max}$ or $|x_{n+1} - x_n| < \varepsilon$ then terminate, otherwise go back to 2

Example - 02

Use **Regula - Falsi method** to find the root of the equation $f(x) = x^3 - 2x - 5$ in the interval [2, 3].
Find the root to within an absolute error tolerance of 0.01.

Solution using MATLAB or Calculator

$$f(x) = x^3 + 0x^2 - 2x - 5$$

```
>> p = [1 0 -2 -5];
>> r = roots(p)

r =

   2.0946 + 0.0000i
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i
```

Solution using Regula - Falsi method

Step 1: Verify the solution exists within interval [2, 3]. i.e., $f(a)f(b) < 0$

- f(2) = -1 & f(3) = 16,
- $f(2)f(3) < 0$

So, the function does change in sign.

Step 2: Compute the intersection point of a chord, x_1, joining $(a, f(a))$ *to* $(b, f(b))$ by: $x_1 = \frac{a\,f(b) - b\,f(a)}{f(b)-f(a)}$;

Set the interval [a, b] to [2, 3] and tolerance to 0.01.

- $x_1 = \frac{2*16 - 3*(-1)}{16-(-1)} = 2.0588$
- $f(x_1) = f(2.0588) = 2.0588^3 - 2 * 2.0588 - 5 = -0.3911$

Step 3: Determine whether the true solution is between $[a, x_1]$ or $[x_1, b]$

- $f(2)f(2.0588) > 0$, the true solution is between 2.0588 and 3.

Step 4: Select the subinterval that contains the true solution as the new interval and repeat step 2 and step 3.

Set the interval to [a, b] = [2.0588, 3]

- $x_2 = \frac{2.0588*16-3*(-0.3911)}{16-(-0.39105)} = 2.0813$
- $f(2.0813) = 2.0813^3 - 2 * 2.0813 - 5 = -0.1468$
- $f(2.0813)f(3) < 0$, the true solution is between 2.0813 and 3.
- Error $= |x_{n+1} - x_n| = |2.0813 - 2.0588| = 0.023 > 0.01$

Continue this process, until the absolute error is less than or equal to 0.01.

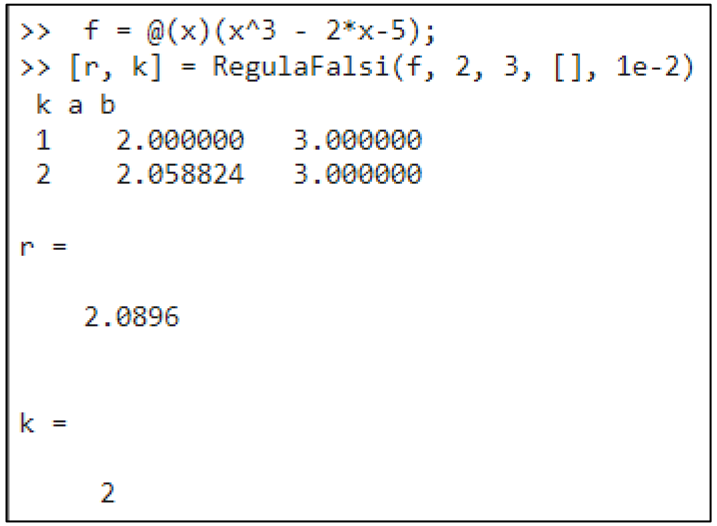
$$f(x) = x^3 - 2x - 5$$

Iteration	$f(a)f(b) < 0$	Interval a	Interval b	x_n ; $x_n = \dfrac{a\,f(b) - b\,f(a)}{f(b) - f(a)}$	$f(x_n)$	Tolerance (0.01) ; $\lvert x_{n+1} - x_n \rvert$
1	$f(2)f(3) < 0$	2 ; f(2) = -1	3 ;f(3) = 16	2.0588	-0.3911	0.5
2		2.0588				
3						
4						
5						
6						
7						

GIVE A TRY

Solving f(x) = 0 using MATLAB (Regula-Falsi)

Use the Regula - Falsi method to find the root of the equation
$f(x) = x^3 - 2x - 5$ in the interval [2, 3].
Find the root to within an absolute error tolerance of 0.01.



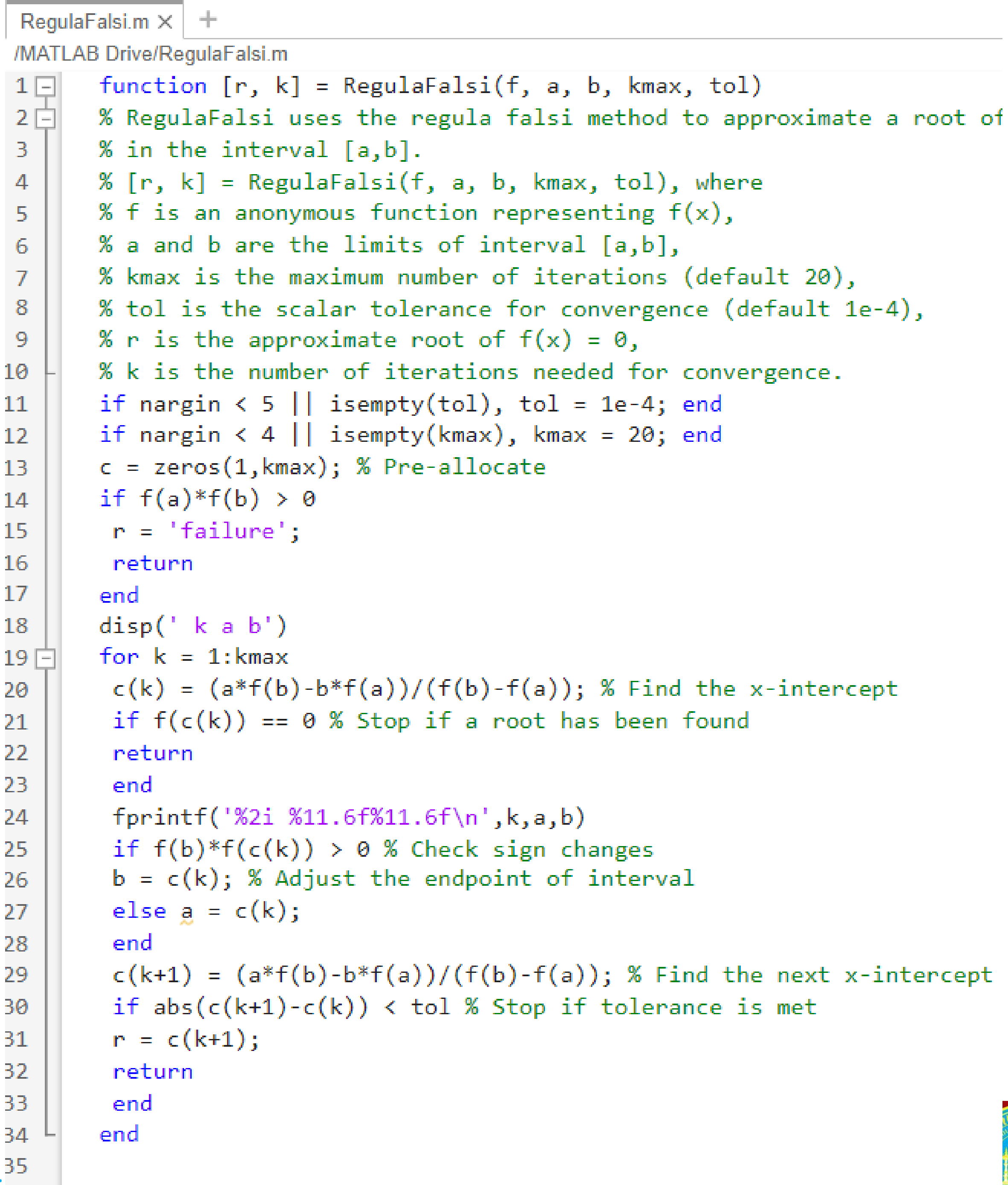
```
>>  f = @(x)(x^3 - 2*x-5);
>> [r, k] = RegulaFalsi(f, 2, 3, [], 1e-2)
 k a b
 1     2.000000   3.000000
 2     2.058824   3.000000

r =

    2.0896


k =

     2
```



RegulaFalsi.m

/MATLAB Drive/RegulaFalsi.m

```
function [r, k] = RegulaFalsi(f, a, b, kmax, tol)
% RegulaFalsi uses the regula falsi method to approximate a root of
% in the interval [a,b].
% [r, k] = RegulaFalsi(f, a, b, kmax, tol), where
% f is an anonymous function representing f(x),
% a and b are the limits of interval [a,b],
% kmax is the maximum number of iterations (default 20),
% tol is the scalar tolerance for convergence (default 1e-4),
% r is the approximate root of f(x) = 0,
% k is the number of iterations needed for convergence.
if nargin < 5 || isempty(tol), tol = 1e-4; end
if nargin < 4 || isempty(kmax), kmax = 20; end
c = zeros(1,kmax); % Pre-allocate
if f(a)*f(b) > 0
 r = 'failure';
 return
end
disp(' k a b')
for k = 1:kmax
 c(k) = (a*f(b)-b*f(a))/(f(b)-f(a)); % Find the x-intercept
 if f(c(k)) == 0 % Stop if a root has been found
 return
 end
 fprintf('%2i %11.6f%11.6f\n',k,a,b)
 if f(b)*f(c(k)) > 0 % Check sign changes
 b = c(k); % Adjust the endpoint of interval
 else a = c(k);
 end
 c(k+1) = (a*f(b)-b*f(a))/(f(b)-f(a)); % Find the next x-intercept
 if abs(c(k+1)-c(k)) < tol % Stop if tolerance is met
 r = c(k+1);
 return
 end
end
```

2. Open method

Open methods do not require a bracket or interval to be specified. These methods start with an initial guess and then use some iterative formula to generate a sequence of approximations to the root.

We will cover:

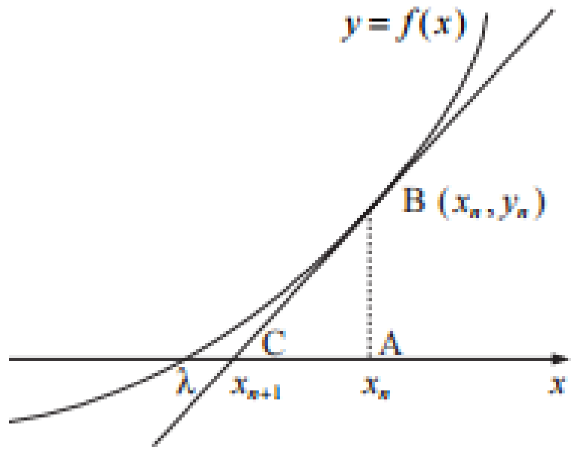
1. **Newton-Raphson method**
2. **Secant method**
3. **Fixed-point iteration method**

Open methods are generally faster than bracketing methods, but they may not converge if the initial guess is not sufficiently close to the root.

2.1 Newton's Method / Newton-Raphson Method

- The diagram shows part of the curve $y = f(x)$, where the equation to be solved is $f(x) = 0$. It also shows an approximate solution x_n and the tangent to the curve at the point (x_n, y_n). If the tangent cuts the x-axis at the point $(x_{n+1}, 0)$, then it is clear that x_{n+1} is a better approximation than x_n to the true root **λ**.
- The gradient of BC is $\frac{BA}{AC}$, which is $\frac{y_n}{(x_n - x_{n+1})}$.
- But $y_n = f(x_n)$, and the gradient of the tangent is f',
  $f'(x_n) = \frac{f(x_n)}{(x_n - x_{n+1})}$
- Re-arranging this gives Newton's iteration formula

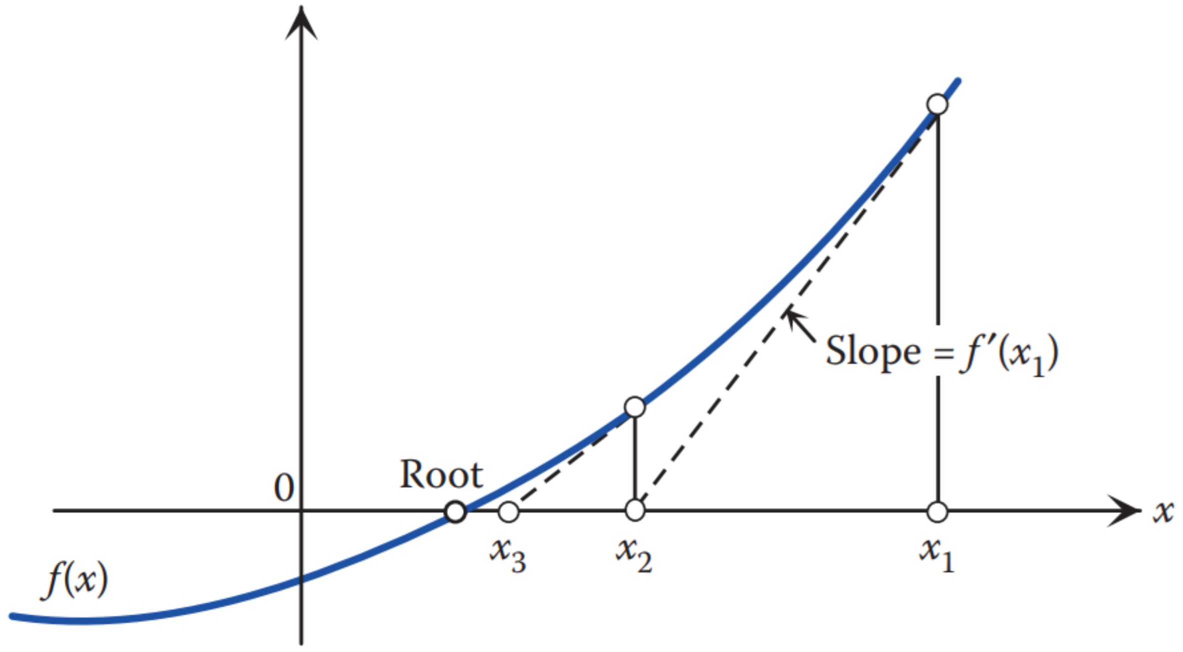
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



Newton's method for solving $f(x) = 0$

Algorithm for Newton’s Method

1. Choose a point x_1 as an initial guess of the solution.
2. For $i = 1, 2, \ldots \ldots$ until the error is smaller than a specified value i.e., $|x_{n+1} - x_n| < \varepsilon$, calculate x_{i+1} by using $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

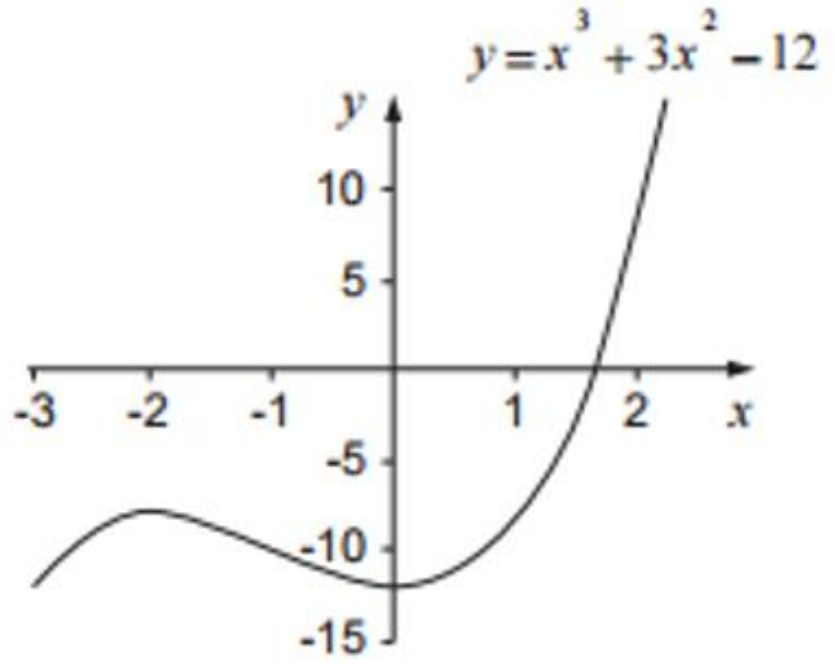


Example - 03

Solve the equation $x^3 + 3x^2 - 12 = 0$, correct to two decimal places.

Solution

Using Matlab code, fplot(@(x)x.^3+3.*x.^2-12.)



From Matlab/Calculator:

$$f(x) = x^3 + 3x^2 - 0x - 12$$

```
>> p = [1 3 0 -12];
>> r = roots(p)

r =

  -2.3064 + 1.4562i
  -2.3064 - 1.4562i
   1.6129 + 0.0000i
```

From the graph, the solution lies between 1 and 2.

So, take $x_0 = 1.5$ as first approximation.

Iteration 1:

$f(x) = x^3 + 3x^2 - 12$, so $f'(x) = 3x^2 + 6x$.

Applying Newton's formula,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$\begin{aligned} x_1 &= 1.5 - \frac{f(1.5)}{f'(1.5)} \\ &= 1.5 - \frac{-1.875}{15.75} \\ &= 1.62. \end{aligned}$$

Iteration 2:

Applying the formula again,

$$\begin{aligned} x_2 &= 1.62 - \frac{f(1.62)}{f'(1.62)} \\ &= 1.62 - \frac{0.125}{17.59} \\ &= 1.613. \end{aligned}$$

And it is not difficult now, after just two iterations, to check that $f(1.605) < 0$ and that $f(1.615) > 0$, so giving $x \approx 1.61$ correct to two decimal places.

Example - 04

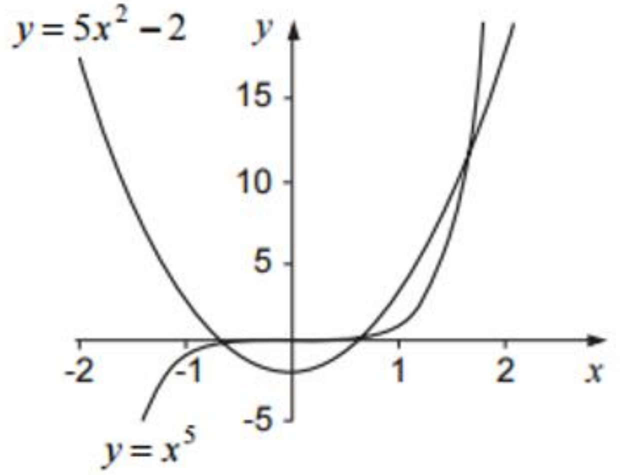
Solve the equation $x^5 = 5x^2 - 2$, correct to three decimal places

Solution

From a graph, there are solutions at $x \approx -0.6$, $x \approx 0.6$ and $x \approx 1.6$ respectively. Newton's method works only for equations in the form $f(x) = 0$, so a rearrangement gives $f(x) = x^5 - 5x^2 + 2$ and $f'(x) = 5x^4 - 10x$.

If $x_0 = -0.6$, then

$$\begin{aligned} x_1 &= -0.6 - \frac{f(-0.6)}{f'(-0.6)} \\ &= -0.6 - \frac{0.122}{6.648} \\ &= -0.618. \end{aligned}$$



If $x_1 = -0.618$, then

$$\begin{aligned} x_2 &= -0.618 - \frac{f(-0.618)}{f'(-0.186)} \\ &= -0.618 - \frac{0.00023}{6.909} \\ &= -0.618. \end{aligned}$$

$f'(-0.6185) < 0$ and $f'(-06175) > 0$, so $x \approx -0.618$ to three decimal places.

If $x_0 = 0.6$ then $x_1 = x_2 = 0.651$; and if $x_0 = 1.6$ then $x_1 = 1.619$ and $x_2 = 1.618$, which can similarly be confirmed as sufficiently accurate. Thus $x \approx -0.618, 0.651$ or 1.618 to three decimal places.

Solution of Equation using Newton’s Method

```
function Xs = NewtonRoot(Fun,FunDer,Xest,Err,imax)
% NewtonRoot finds the root of Fun = 0 near the point Xest using Newton's method.
% Input variables:
% Fun    Name of a user-defined function that calculates Fun for a given x.
% FunDer Name of a user-defined function that calculates the derivative
%        of Fun for a given x.
% Xest     Initial estimate of the solution.
% Err     Maximum error.
% imax     Maximum number of iterations
% Output variable:
% Xs      Solution
for i = 1:imax
    Xi = Xest - Fun(Xest)/FunDer(Xest);
    if abs((Xi - Xest)/Xest) < Err
        Xs = Xi;
        break
    end
    Xest = Xi;
end
```

```
if i = = imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
     Xs = ('No answer');
end
```

Example - 05

Find the solution of the equation $8 - 4.5(x - \sin x) = 0$ (the same equation as in Example 3-1) by using Newton's method in the following two ways:

(*a*) Using a nonprogrammable calculator, calculate the first two iterations on paper using six significant figures.

(*b*) Use MATLAB with the function `NewtonRoot` that is listed in Fig. 3-11. Use 0.0001 for the maximum relative error and 10 for the maximum number of iterations.

In both parts, use $x = 2$ as the initial guess of the solution.

SOLUTION

In the present problem, $f(x) = 8 - 4.5(x - \sin x)$ and $f'(x) = -4.5(1 - \cos x)$.

(*a*) To start the iterations, $f(x)$ and $f'(x)$ are substituted in Eq. (3.14):

$$x_{i+1} = x_i - \frac{8 - 4.5(x_i - \sin x_i)}{-4.5(1 - \cos x_i)} \tag{3.20}$$

In the first iteration, $i = 1$ and $x_1 = 2$, and Eq. (3.20) gives:

$$x_2 = 2 - \frac{8 - 4.5(2 - \sin(2))}{-4.5(1 - \cos(2))} = 2.48517 \tag{3.21}$$

For the second iteration, $i = 2$ and $x_2 = 2.48517$, and Eq. (3.20) gives:

$$x_3 = 2.48517 - \frac{8 - 4.5(2.48517 - \sin(2.48517))}{-4.5(1 - \cos(2.48517))} = 2.43099 \tag{3.22}$$

(*b*) To solve the equation with MATLAB using the function `NewtonRoot`, the user must create user-defined functions for $f(x)$ and $f'(x)$. The two functions, called `FunExample2` and `FunDerExample2`, are:

```
function y = FunExample2(x)
y = 8 - 4.5*(x - sin(x));
```

and

```
function y = FunDerExample2(x)
y = -4.5 + 4.5*cos(x);
```

Once the functions are created and saved, the `NewtonRoot` function can be used in the Command Window:

The user-defined functions are entered as function handles.

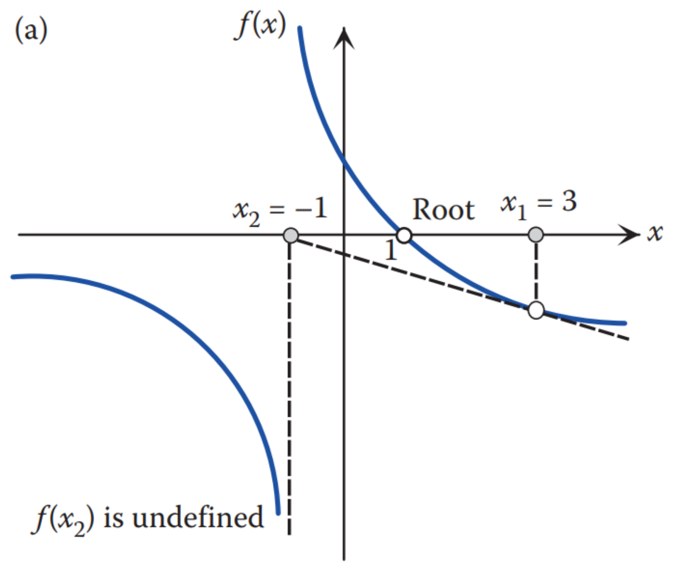
```
>> format long
>> xSolution = NewtonRoot(@FunExample2,@FunDerExample2,2,0.0001,10)
xSolution =
   2.430465741723630
```

Newton’s Method to fail!

1. A usual factor is that the initial point $\mathbf{x_1}$ is not sufficiently close to the intended root.
2. Second is that at some point in the iterations, $f'(x_n)$ may be close to or equal to zero.
3. Other scenarios, where the iteration simply halts (a) or the sequence diverges (b).

Apply Newton’s method to find the root of $2/(x + 1) = 1$.
For the initial point use $x_1 = 3$, and $x_1 = 4$.

Apply Newton's method to find the root of 2/(x + 1) = 1. For the initial point use x_1 = 3, and x_1 = 4.

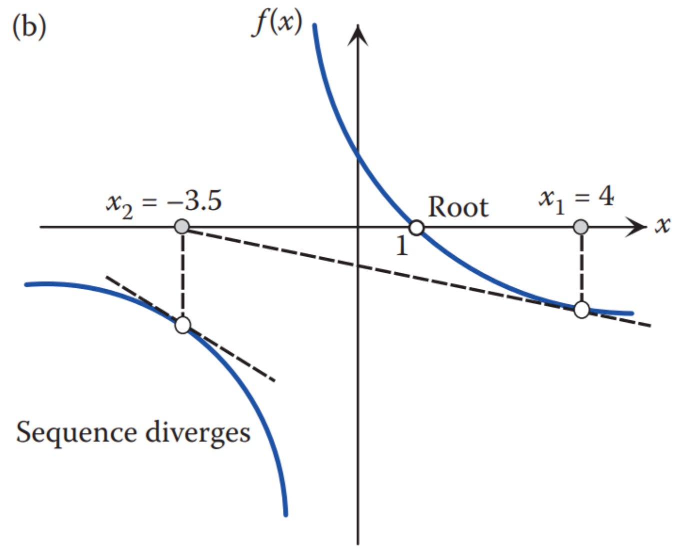


$$f(x) = \frac{2}{x+1} - 1 \text{ so that } f'(x) = -\frac{2}{(x+1)^2}.$$

1. Starting with $x_1 = 3$, we find

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 3 - \frac{-\frac{1}{2}}{-\frac{1}{8}} = -1$$

The iterations halt at this point because $f(-1)$ is undefined.



2. Starting with $x_1 = 4$, we find

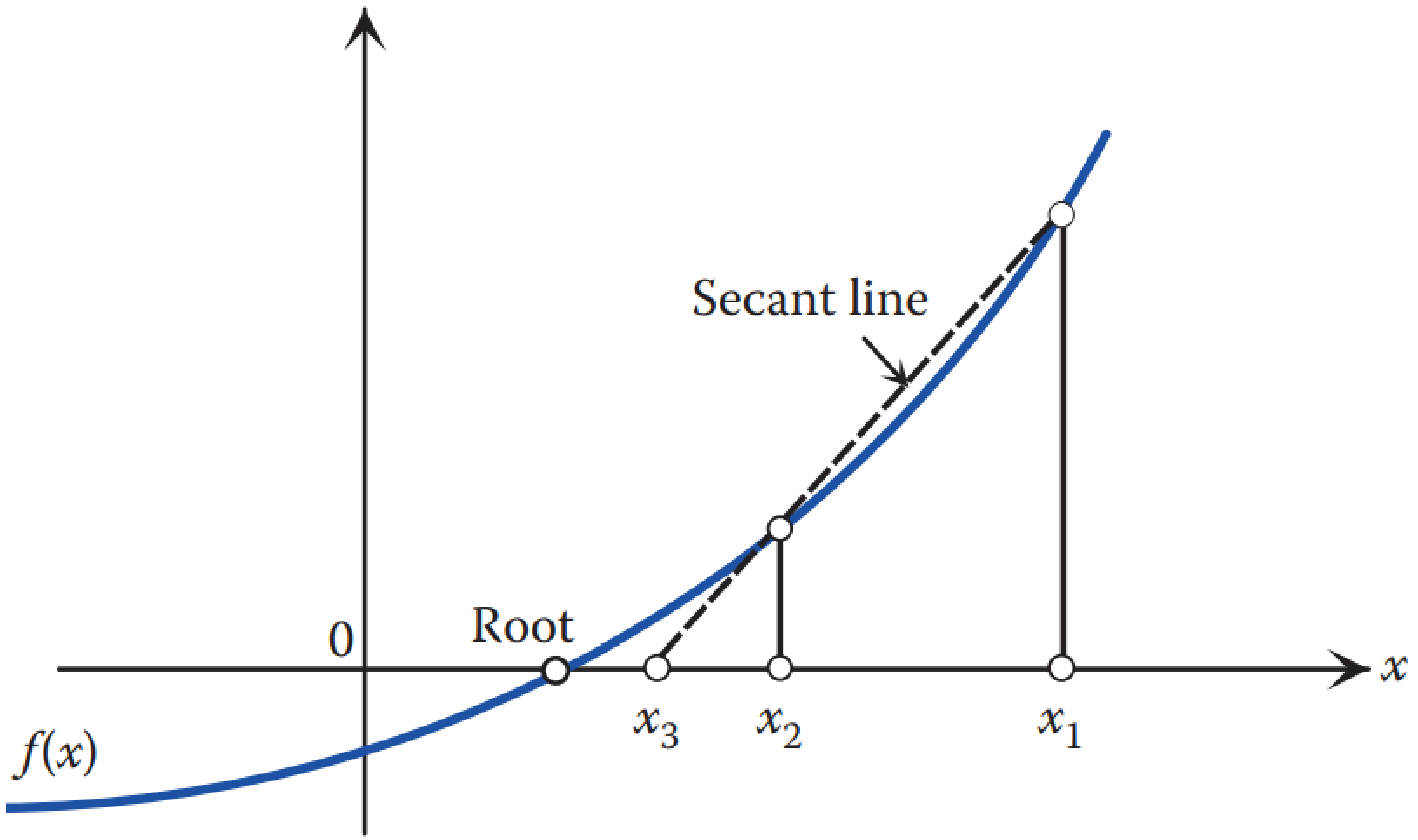
$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 4 - \frac{-\frac{3}{5}}{-\frac{2}{25}} = -3.5, \quad x_3 = -9.1250, \quad x_4 = -50.2578, \quad \ldots$$

The sequence clearly diverges.

2.2 Secant Method

It is similar to the Newton-Raphson method, but instead of using the derivative of the function, it approximates the derivative using a secant line between two points.

The secant method is generally faster than the bisection method and simpler than the Newton-Raphson method. However, it may converge more slowly or even diverge if the initial points are not chosen carefully.



$$x_{n+1} = x_n - f(x_n)\,\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\,.$$

Example - 06

Find the positive solution of $f(x) = x - 2sinx = 0$ by the Secant method, starting from $x_0 = 2, x_1 = 1.9$

Solution:

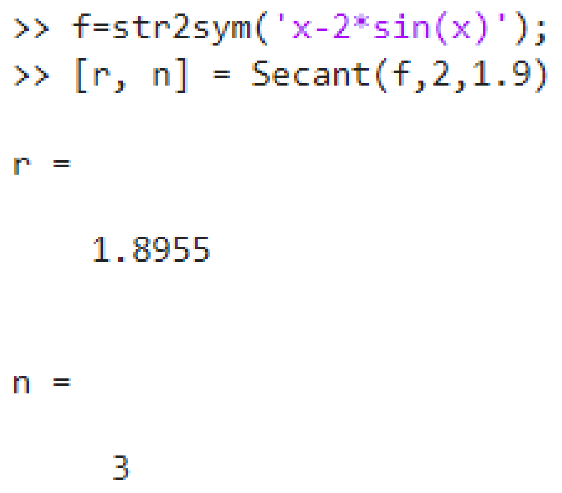
$$x_{n+1} = x_n - f(x_n)\left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right]$$

$$x_{n+1} = x_n - \frac{(x_n - 2\sin x_n)(x_n - x_{n-1})}{x_n - x_{n-1} + 2(\sin x_{n-1} - \sin x_n)} = x_n - \frac{N_n}{D_n}.$$

n	x_{n-1}	x_n	N_n	D_n	Tolerance; $x_{n+1} - x_n = -\frac{N_n}{D_n}$	x_{n+1}
1	2.000000	1.900000	-0.000740	-0.174005	-0.004253	1.895747
2	1.900000	1.895747	-0.000002	-0.006986	-0.000252	1.895494
3	1.895747	1.895494	0		0	

Solving f(x) = 0 using MATLAB (Secant Method)

Find the positive solution of $f(x) = x - 2sinx = 0$ by the Secant method, starting from $x_0 = 2, x_1 = 1.9$



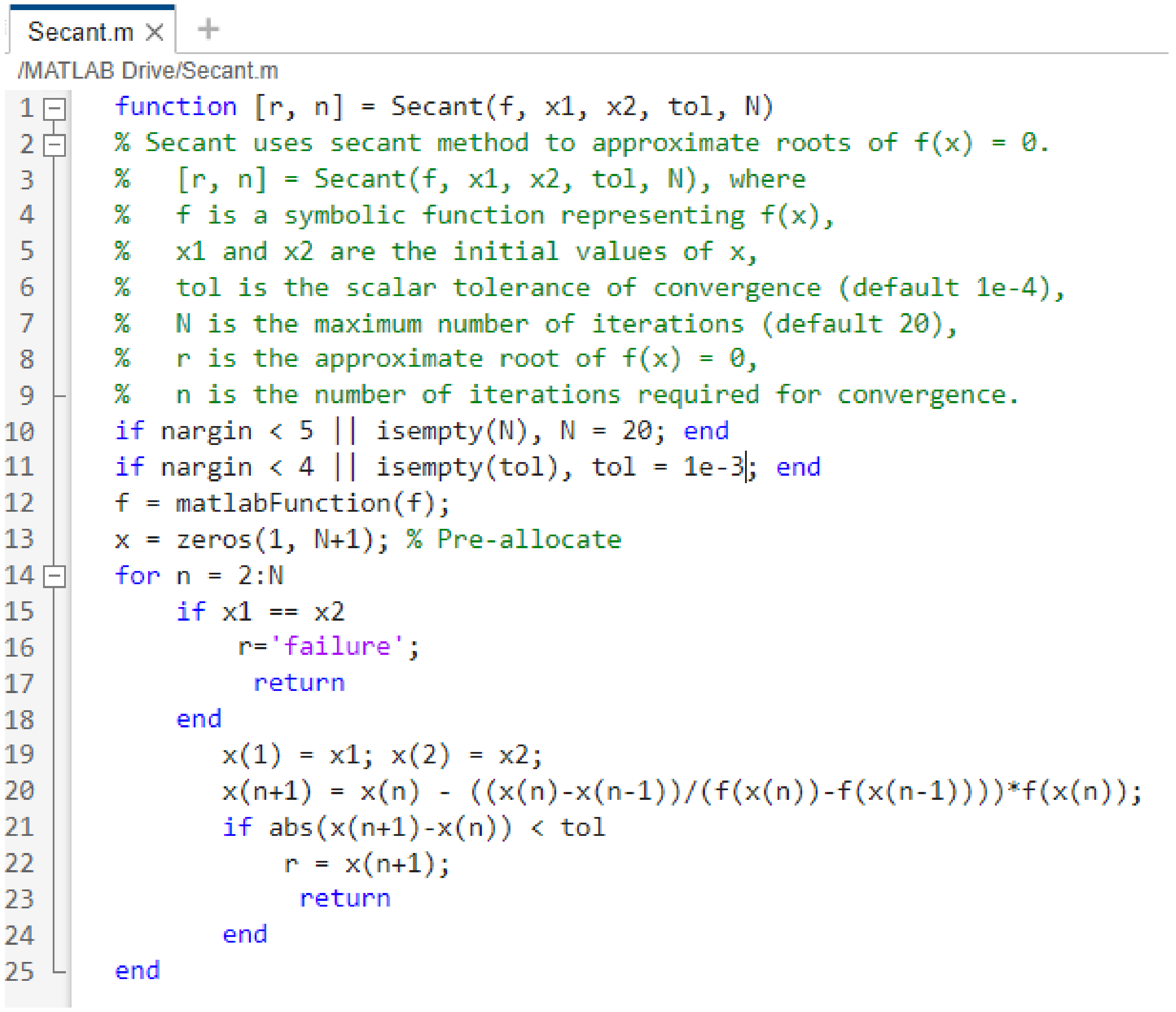
```
>> f=str2sym('x-2*sin(x)');
>> [r, n] = Secant(f,2,1.9)

r =

    1.8955


n =

     3
```



Secant.m

/MATLAB Drive/Secant.m

```
function [r, n] = Secant(f, x1, x2, tol, N)
% Secant uses secant method to approximate roots of f(x) = 0.
%   [r, n] = Secant(f, x1, x2, tol, N), where
%   f is a symbolic function representing f(x),
%   x1 and x2 are the initial values of x,
%   tol is the scalar tolerance of convergence (default 1e-4),
%   N is the maximum number of iterations (default 20),
%   r is the approximate root of f(x) = 0,
%   n is the number of iterations required for convergence.
if nargin < 5 || isempty(N), N = 20; end
if nargin < 4 || isempty(tol), tol = 1e-3; end
f = matlabFunction(f);
x = zeros(1, N+1); % Pre-allocate
for n = 2:N
    if x1 == x2
        r='failure';
         return
    end
       x(1) = x1; x(2) = x2;
       x(n+1) = x(n) - ((x(n)-x(n-1))/(f(x(n))-f(x(n-1))))*f(x(n));
       if abs(x(n+1)-x(n)) < tol
           r = x(n+1);
            return
       end
end
```

2.3 Fixed-point Method

The fixed-point method is particularly useful when other numerical methods such as Newton's method or the bisection method are not applicable.

It can be used to solve a wide range of non-linear equations, including transcendental equations, algebraic equations, and differential equations.

Fixed point method is a simple and easy-to-use numerical algorithm used to solve non-linear equations.

Algorithm for Fixed Point Method

1. Rewrite the original equation $f(x) = 0$ in the form $x = g(x)$ for a suitable $g(x)$; $g(x)$ – called as iteration function

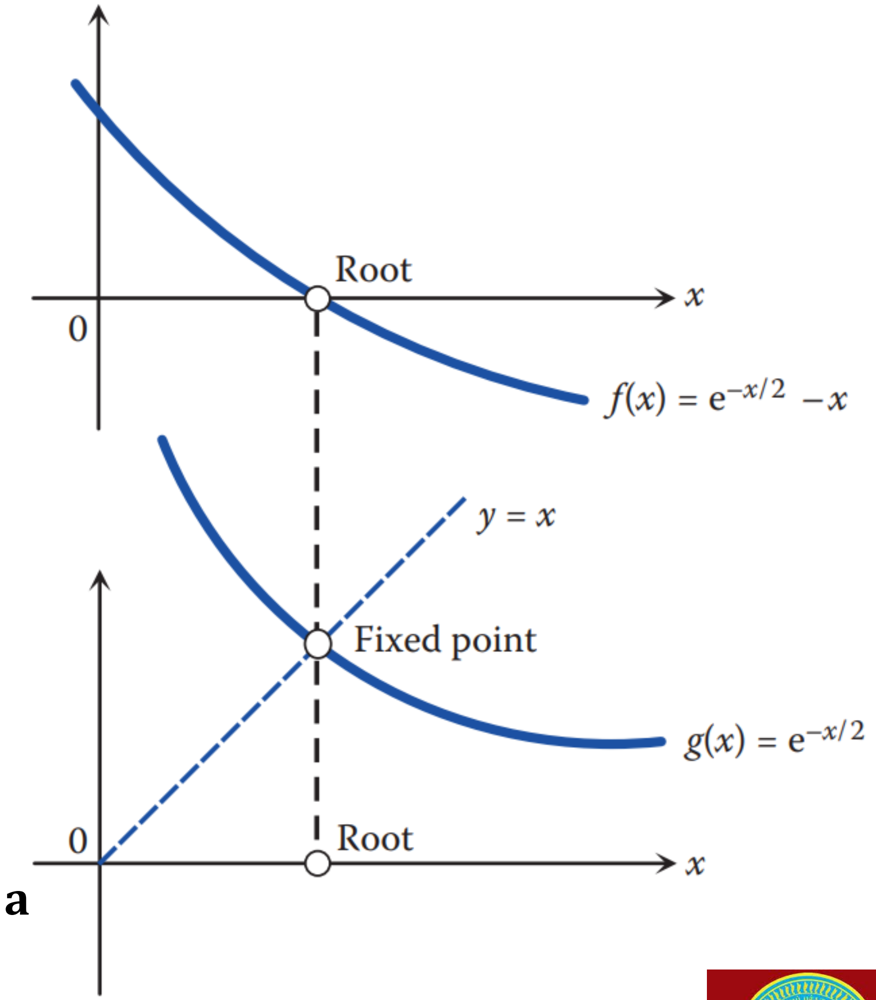
2. Choose an initial guess x_0.

3. Use the iterative formula $x_{n+1} = g(x_n)$ to generate a sequence of successive approximations to the solution.

4. Repeat step 3 until the sequence converges to the desired accuracy or until a maximum number of iterations is reached.

Illustration for Fixed Point Method

As an example, consider $e^{-x/2} - x = 0$ and its root as shown:



Step 1: Rewrite the original equation $f(x) = 0$ in the form $x = g(x)$

$x = e^{-x/2}$; $g(x) = e^{-x/2}$ Or $x = -2\ln x$; $g(x) = -2\ln x$

Step 2: An initial guess x_0 near the fixed point

Step 3: The next point x_1 is found as $x_1 = g(x_0)$

Step 4: Repeat Step 3 until two successive points are within a prescribed distance of one another, i.e., $|x_n - x_{n-1}| < \varepsilon$

Example - 07

Set up an iteration process for the equation $f(x) = x^2 - 3x + 1 = 0$. Since we know the solutions

$$x = 1.5 \pm \sqrt{1.25}, \qquad \text{thus} \qquad 2.618034 \qquad \text{and} \qquad 0.381966,$$

we can watch the behavior of the error as the iteration proceeds.

$$\boldsymbol{f(x) = x^2 - 3x + 1 = 0}$$

Step 1: Rewrite the original equation $f(x) = 0$ in the form $x = g(x)$

- $x = \frac{1}{3}(x^2 + 1)$; $g(x) = \frac{1}{3}(x^2 + 1)$

Step 2: An initial guess x_0 near the fixed point

- $x_0 = 1.000$

Step 3: The next point x_2 is found as $x_1 = g(x_0)$

- $x_1 = g(x_0) = \frac{1}{3}(x_0^2 + 1)$

Step 4: Repeat Step 3 until two successive points are within a prescribed distance of one another, i.e., $|x_n - x_{n-1}| < \varepsilon$

- $|0.667 - 1.000| = 0.333 > \varepsilon$

n	x_{n-1}	$x_n = g(x_{n-1})$	**Tolerance;** $\lvert x_n - x_{n-1}\rvert < \varepsilon$
1	1.000	0.667	0.333
2	0.667	0.482	0.185
3	0.482	0.411	0.071
4	0.411	0.390	0.021

$$f(x) = x^2 - 3x + 1 = 0$$

If we choose $x_0 = 1$, we obtain the sequence

$$x_0 = 1.000, \quad x_1 = 0.667, \quad x_2 = 0.481, \quad x_3 = 0.411, \quad x_4 = 0.390, \cdots$$

which seems to approach the smaller solution. If we choose $x_0 = 2$, the situation is similar. If we choose $x_0 = 3$, we obtain the sequence

$$x_0 = 3.000, \quad x_1 = 3.333, \quad x_2 = 4.037, \quad x_3 = 5.766, \quad x_4 = 11.415, \cdots$$

which diverges.

Our equation may also be written (divide by x)

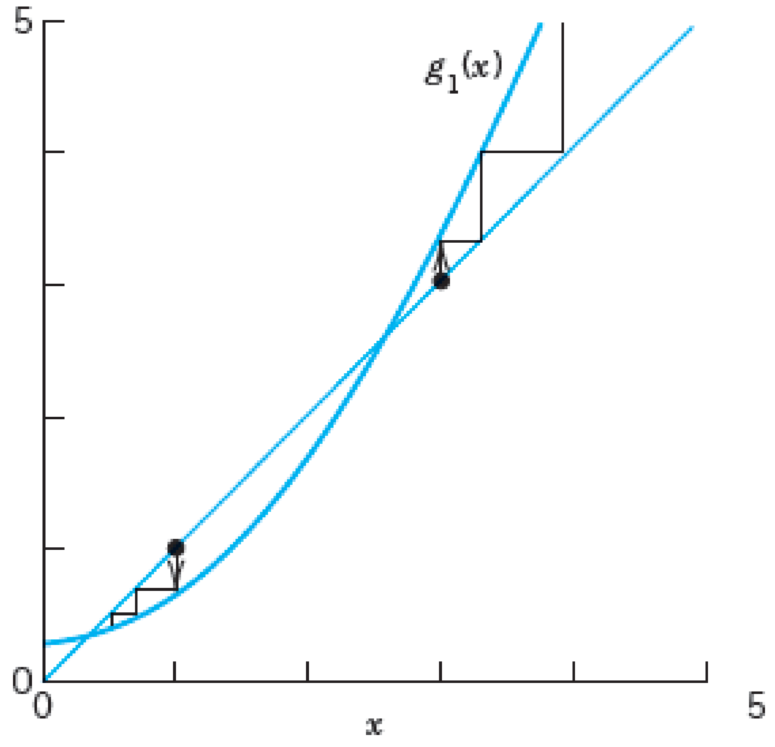
$$x = g_2(x) = 3 - \frac{1}{x}, \qquad \text{thus} \qquad x_{n+1} = 3 - \frac{1}{x_n},$$

and if we choose $x_0 = 1$, we obtain the sequence

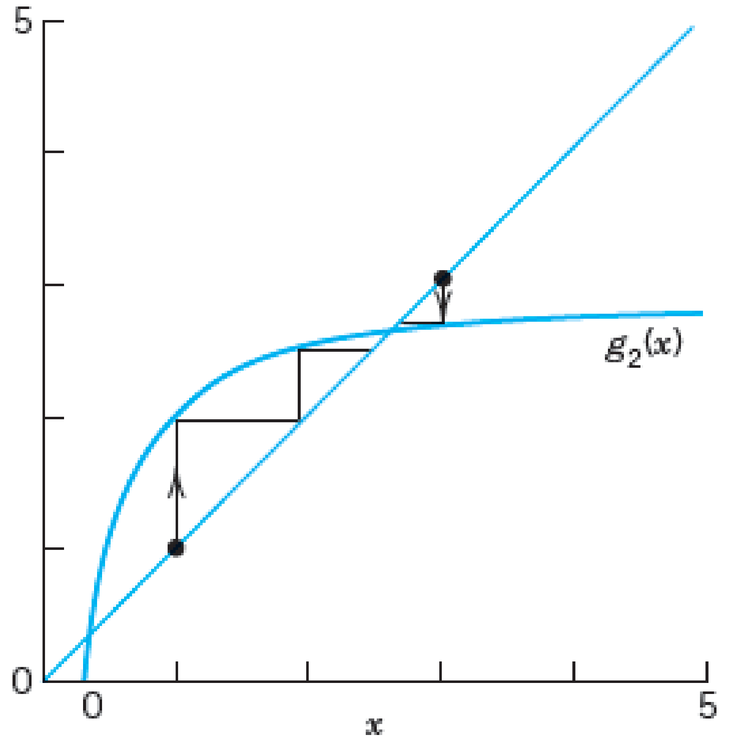
$$x_0 = 1.000, \quad x_1 = 2.000, \quad x_2 = 2.500, \quad x_3 = 2.600, \quad x_4 = 2.615, \cdots$$

which seems to approach the larger solution. Similarly, if we choose $x_0 = 3$, we obtain the sequence

$$x_0 = 3.000, \quad x_1 = 2.667, \quad x_2 = 2.625, \quad x_3 = 2.619, \quad x_4 = 2.618, \cdots.$$



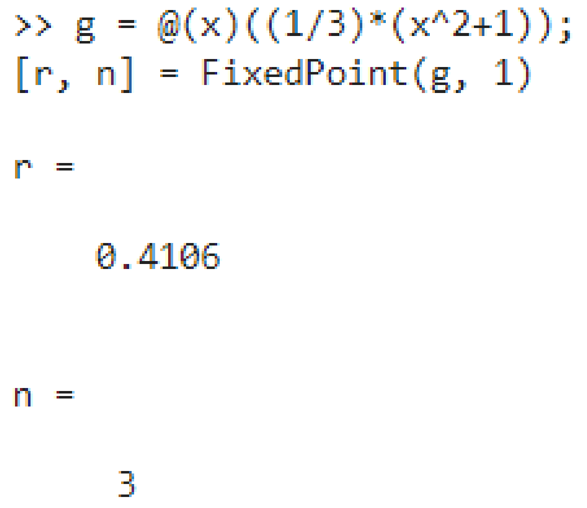
(a)



(b)

Solving f(x) = 0 using MATLAB (Fixed-Point Method)

Find the positive solution of $f(x) = x - 2sinx = 0$ by the Fixed-Point Method, starting from $x_0 = 1.0$



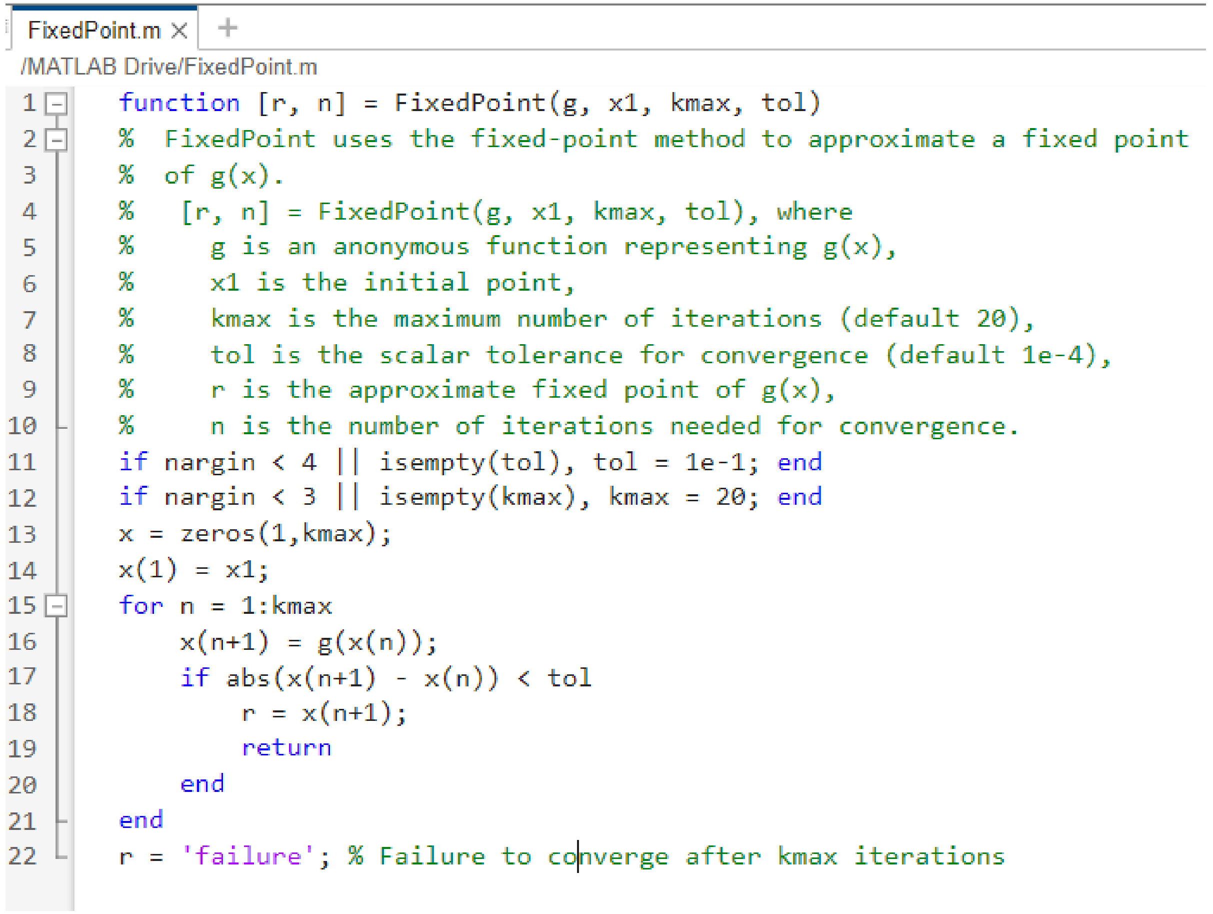
```
>> g = @(x)((1/3)*(x^2+1));
[r, n] = FixedPoint(g, 1)

r =

    0.4106


n =

     3
```



FixedPoint.m × +

/MATLAB Drive/FixedPoint.m

```
function [r, n] = FixedPoint(g, x1, kmax, tol)
% FixedPoint uses the fixed-point method to approximate a fixed point
% of g(x).
%  [r, n] = FixedPoint(g, x1, kmax, tol), where
%    g is an anonymous function representing g(x),
%    x1 is the initial point,
%    kmax is the maximum number of iterations (default 20),
%    tol is the scalar tolerance for convergence (default 1e-4),
%    r is the approximate fixed point of g(x),
%    n is the number of iterations needed for convergence.
if nargin < 4 || isempty(tol), tol = 1e-1; end
if nargin < 3 || isempty(kmax), kmax = 20; end
x = zeros(1,kmax);
x(1) = x1;
for n = 1:kmax
    x(n+1) = g(x(n));
    if abs(x(n+1) - x(n)) < tol
        r = x(n+1);
        return
    end
end
r = 'failure'; % Failure to converge after kmax iterations
```

Convergence of Fixed-Point Iteration

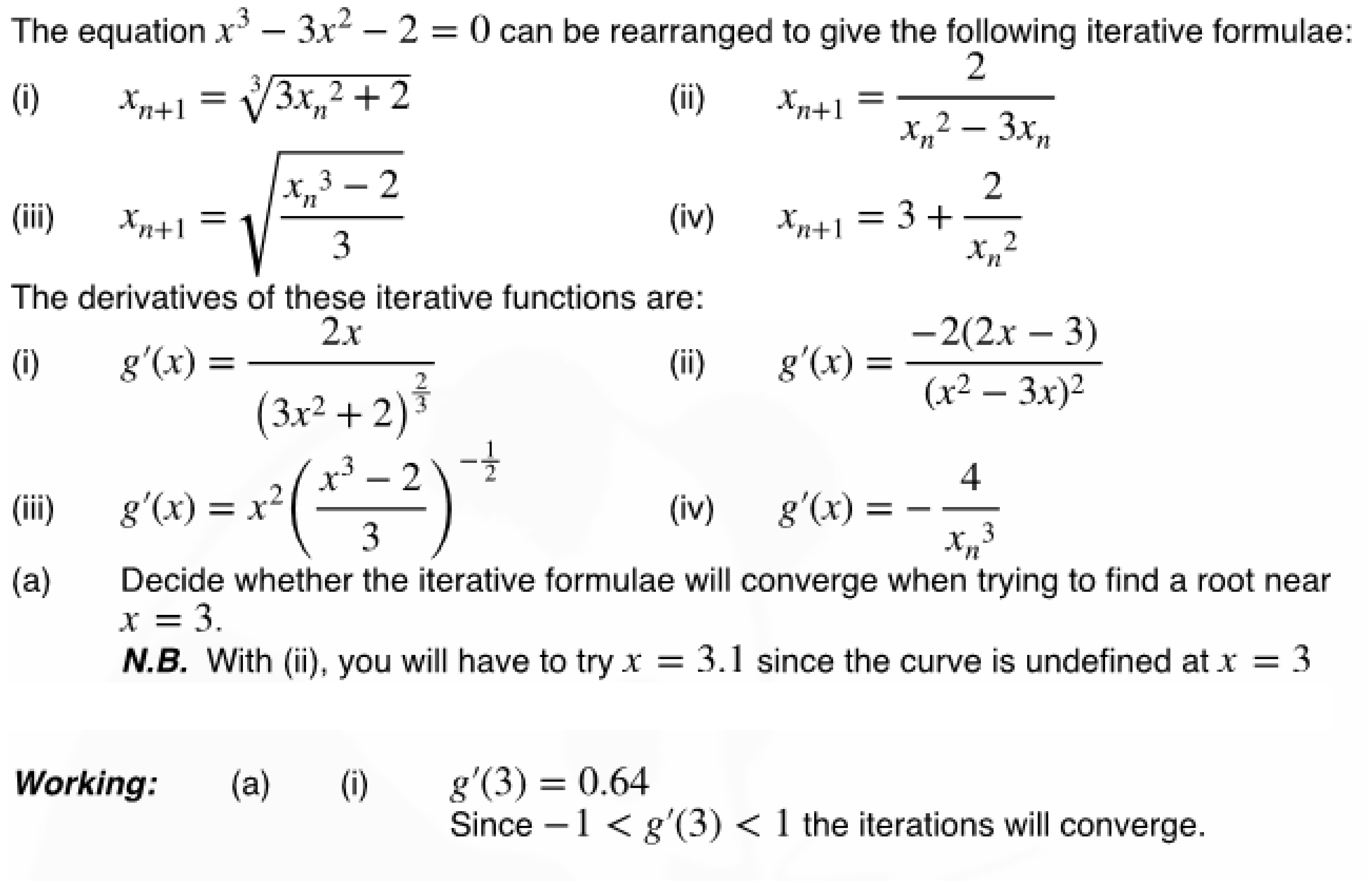
- Let r ∈I be a Fixed point of g(x).
- Assume that g(x) has a continuous derivative in interval I, and $|g'(x)| \leq K < 1$ for all x ∈I.
- Then, for any initial point x_1 ∈I, the Fixed-point iteration in Equation

$$x_{n+1} = g(x_n), \quad n = 1, 2, 3, \ \ldots, \quad x_1 = \text{initial guess}$$

generates a sequence $\{x_n\}$ that converges to r.

If |g′(x)| < 1 near a Fixed point of g(x), convergence is guaranteed. In other words, *if in a neighborhood of a root, the curve representing g(x) is less steep than the line y = x, then the Fixed-point iteration converges to that root*. Note that this is a sufficient, and not necessary, condition for convergence.

Exercise:



The equation $x^3 - 3x^2 - 2 = 0$ can be rearranged to give the following iterative formulae:

(i) $x_{n+1} = \sqrt[3]{3x_n{}^2 + 2}$ (ii) $x_{n+1} = \dfrac{2}{x_n{}^2 - 3x_n}$

(iii) $x_{n+1} = \sqrt{\dfrac{x_n{}^3 - 2}{3}}$ (iv) $x_{n+1} = 3 + \dfrac{2}{x_n{}^2}$

The derivatives of these iterative functions are:

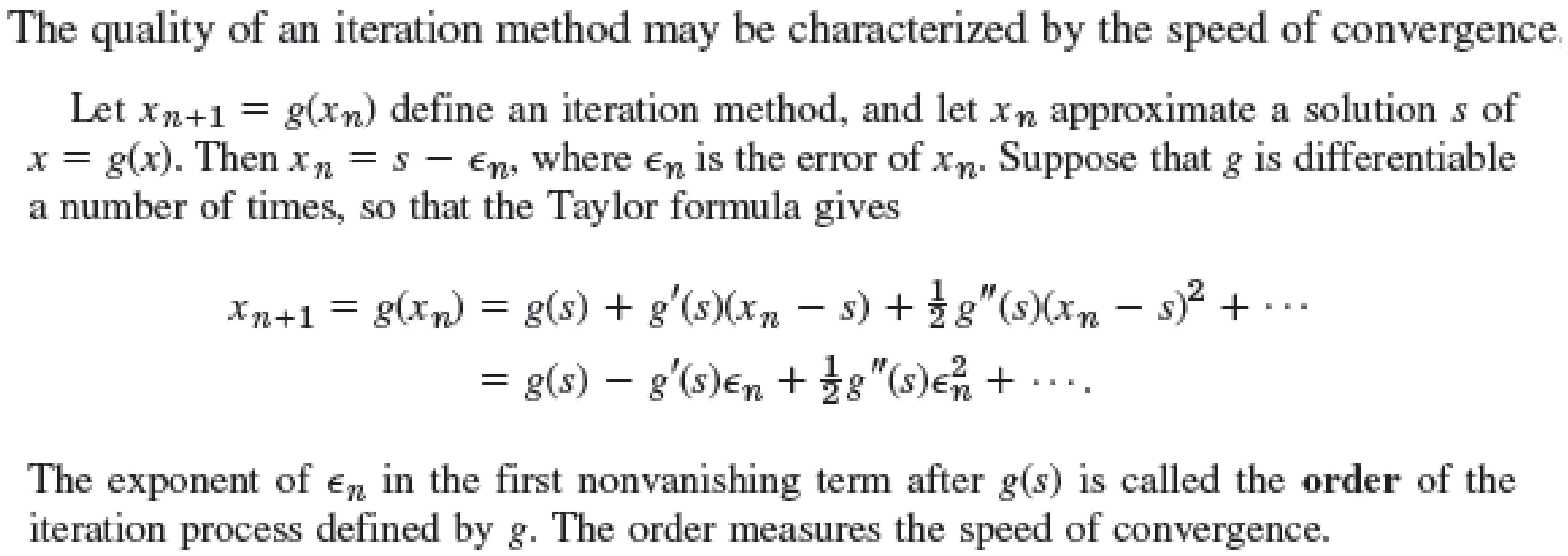
(i) $g'(x) = \dfrac{2x}{(3x^2 + 2)^{\frac{2}{3}}}$ (ii) $g'(x) = \dfrac{-2(2x - 3)}{(x^2 - 3x)^2}$

(iii) $g'(x) = x^2\left(\dfrac{x^3 - 2}{3}\right)^{-\frac{1}{2}}$ (iv) $g'(x) = -\dfrac{4}{x_n{}^3}$

(a) Decide whether the iterative formulae will converge when trying to find a root near $x = 3$.
N.B. With (ii), you will have to try $x = 3.1$ since the curve is undefined at $x = 3$

Working: (a) (i) $g'(3) = 0.64$
Since $-1 < g'(3) < 1$ the iterations will converge.

Order of an Iteration Method (Speed of Convergence)


The quality of an iteration method may be characterized by the speed of convergence.

Let $x_{n+1} = g(x_n)$ define an iteration method, and let x_n approximate a solution s of $x = g(x)$. Then $x_n = s - \epsilon_n$, where ϵ_n is the error of x_n. Suppose that g is differentiable a number of times, so that the Taylor formula gives

$$\begin{aligned} x_{n+1} = g(x_n) &= g(s) + g'(s)(x_n - s) + \tfrac{1}{2}g''(s)(x_n - s)^2 + \cdots \\ &= g(s) - g'(s)\epsilon_n + \tfrac{1}{2}g''(s)\epsilon_n^2 + \cdots . \end{aligned}$$

The exponent of ϵ_n in the first nonvanishing term after $g(s)$ is called the **order** of the iteration process defined by g. The order measures the speed of convergence.

Example Consider the convergent iteration

$$p_0 = 0.5 \quad \text{and} \quad p_{k+1} = e^{-p_k} \quad \text{for } k = 0, 1, \ldots$$

The first 10 terms are obtained by the calculations

$$\begin{aligned} p_1 &= e^{-0.500000} = 0.606531 \\ p_2 &= e^{-0.606531} = 0.545239 \\ p_3 &= e^{-0.545239} = 0.579703 \\ &\vdots \qquad\qquad \vdots \\ p_9 &= e^{-0.566409} = 0.567560 \\ p_{10} &= e^{-0.567560} = 0.566907 \end{aligned}$$

The sequence is converging, and further calculations reveal that

$$\lim_{n \to \infty} p_n = 0.567143 \ldots .$$

Thus we have found an approximation for the fixed point of the function $y = e^{-x}$.

TASK - 01

Consider $f(x) = x\cos(x) + 1 = 0$ has a root in the interval $[-2, 4]$, Set the tolerance $\varepsilon = 10^{-2}$ and maximum 20 iterations for your computations; Provide the approximations to six decimal places.

1. Find the root using **BISECTION METHOD**
2. Find the root using **REGULA FALSI METHOD**
3. Find the root using **NEWTON'S METHOD** with $x_1 = 1$.
4. State whether, the iterations using **FIXED-POINT METHOD** will converge or not.

Exercises

1. Let $f(x) = x^2 - x + 2$.
   (a) Find the Newton-Raphson formula $p_k = g(p_{k-1})$.
   (b) Start with $p_0 = -1.5$ and find p_1, p_2, and p_3.
2. Let $f(x) = x^2 - x - 3$.
   (a) Find the Newton-Raphson formula $p_k = g(p_{k-1})$.
   (b) Start with $p_0 = 1.6$ and find p_1, p_2, and p_3.
   (c) Start with $p_0 = 0.0$ and find p_1, p_2, p_3, and p_4. What do you conjecture about *this sequence?*
3. Let $f(x) = (x - 2)^4$.
   (a) Find the Newton-Raphson formula $p_k = g(p_{k-1})$.
   (b) Start with $p_0 = 2.1$ and find p_1, p_2, p_3, and p_4.
   (c) Is the sequence converging quadratically or linearly?
4. Let $f(x) = x^3 - 3x - 2$.
   (a) Find the Newton-Raphson formula $p_k = g(p_{k-1})$.
   (b) Start with $p_0 = 2.1$ and find p_1, p_2, p_3, and p_4.
   (c) Is the sequence converging quadratically or linearly?
5. Consider the function $f(x) = \cos(x)$.
   (a) Find the Newton-Raphson formula $p_k = g(p_{k-1})$.
   (b) We want to find the root $p = 3\pi/2$. Can we use $p_0 = 3$? Why?
   (c) We want to find the root $p = 3\pi/2$. Can we use $p_0 = 5$? Why?

6. Consider the function $f(x) = xe^{-x}$.

(a) Find the Newton-Raphson formula $p_k = g(p_{k-1})$.

(b) If $p_0 = 0.2$, then find p_1, p_2, p_3, and p_4. What is $\lim_{n\to\infty} p_k$?

(c) If $p_0 = 20$, then find p_1, p_2, p_3, and p_4. What is $\lim_{n\to\infty} p_k$?

(d) What is the value of $f(p_4)$ in part (c)?

7. For each function, find an interval $[a, b]$ so that $f(a)$ and $f(b)$ have opposite signs.

(a) $f(x) = e^x - 2 - x$

(b) $f(x) = \cos(x) + 1 - x$

(c) $f(x) = \ln(x) - 5 + x$

(d) $f(x) = x^2 - 10x + 23$

Determine rigorously if each function has a unique fixed point on the given interval

(a) $g(x) = 1 - x^2/4$ on $[0, 1]$

(b) $g(x) = 2^{-x}$ on $[0, 1]$

(c) $g(x) = 1/x$ on $[0.5, 5.2]$

Investigate the nature of the fixed-point iteration when

$$g(x) = -4 + 4x - \frac{1}{2}x^2.$$

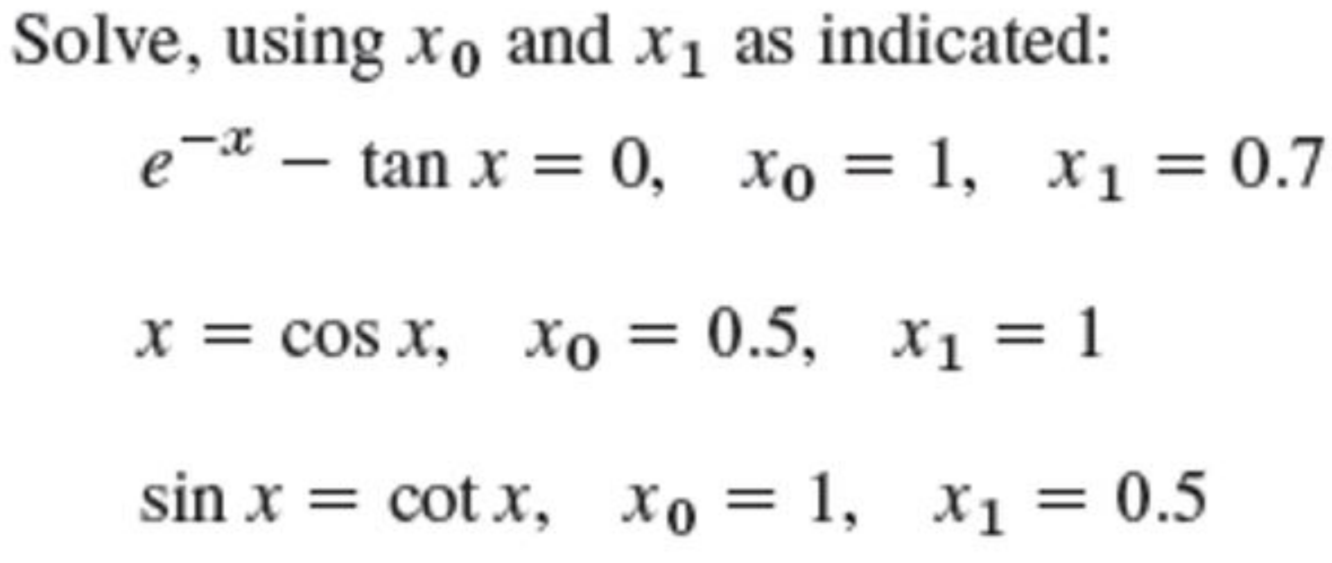
(a) Solve $g(x) = x$ and show that $P = 2$ and $P = 4$ are fixed points.

(b) Use the starting value $p_0 = 1.9$ and compute p_1, p_2, and p_3.

(c) Use the starting value $p_0 = 3.8$ and compute p_1, p_2, and p_3.

(d) Find the errors E_k and relative errors R_k for the values p_k in parts (b) and (c).

Find the positive solution of following by the Secant method,



Solve, using x_0 and x_1 as indicated:

$$e^{-x} - \tan x = 0, \quad x_0 = 1, \quad x_1 = 0.7$$

$$x = \cos x, \quad x_0 = 0.5, \quad x_1 = 1$$

$$\sin x = \cot x, \quad x_0 = 1, \quad x_1 = 0.5$$