# 

**Mathematical Methods of Engineering**

**Lecture Note 3**

**Nonlinear Equations in One Variable**

**3.1 Introduction**

Many physical and engineering problems can be governed by nonlinear equation. These equations involve circular function, hyperbolic function, exponential and other transcendental functions and their combinations. The solution(values of *x)* of the nonlinear equation is challenging. In fact, majority of equations cannot be solved analytically and so we have to solve them numerically. In this chapter, we shall consider some of the important approximate methods in finding the roots of the equations in one variable.

**3.2 Number of Real Roots**

**3.2.1 Number of Real Roots by Graphical Method**

A polynomial equation of degree *n* has exactly *n* roots. Some of them are real and others are complex. For a non-polynomial equation , there is no such rule of finding the number of roots. Geometrically, if the graph of crosses the *x*-axis at , then is a real root of . Now we shall consider graphically to find the number of real roots and its location.

Rewrite the equation as . At the point of intersection (say) of the graphs

and

that is

and hence,is a real root of the equation.

Thus the number of intersections of the two graphs will be the number of real roots.

N.B.

**Fundamental theorem of algebra:** (Theorem of equations proved by Carl Friedrich Gauss in 1799)

It states that, **“Every polynomial equation of degree n with complex number coefficients has n roots, or solutions, in the complex numbers”**.

**Example 3.1**

**(a)** Find the number of real root(s) of by graphical method.

(b) Find the number of complex roots, if any.

(c) Use MATLAB function “roots” to find all the roots including complex roots.

|  |  |
| --- | --- |
| There is one point of intersection here in the plot. So the number of real root is 1. |  |

(b) It is a polynomial equation of degree three and hence the total number of roots is three. So, number of complex roots

= Total number of roots - Number of real roots .

(c) p=[1 0 4 -3] % entry of cubic polynomial

p = 1 0 4 -3

>> Roots = roots(p)

Roots =

-0.3368 + 2.0833i

-0.3368 - 2.0833i

0.6736 + 0.0000i

**Example 3.2:** Find the number of real root(s) of by graphical method.

|  |  |
| --- | --- |
| There is one point of intersection here in the plot. So the number of real root is 1. |  |

**Example 3.3:** Find the number of real root(s) of by graphical method.

|  |  |
| --- | --- |
| There are three points of intersection here in the plot. So the number of real roots are 3. |  |

**3.2.2 Location of Roots**

To locate the roots of , first study the graph of as shown below. If we can find two values of *x*, one for which is positive, and one for which is negative, then the curve must have crossed the *x*-axis and so must have passed through a root of the equation .

|  |  |
| --- | --- |
| In general, if is continuous in and and are opposite in signs i.e.,, then there exists odd number of real roots (at least one root) of in .  But the only exception where it does not work is when curve touches the *x*-axis. For this case, the existence of a root can be determined by the sign of in the interval containing the root and it will satisfy the condition. | ***x***  ***y***  **O**  **−**  **−**  **−**  **+**  **+**  **Fig 4.1** |

**Example 3.4**  The equation has two real roots. For each root, find an interval where it lies.

Consider the values of function for different values of *x*:

|  |  |
| --- | --- |
| *x* | *f*(*x*) |
| -2 | 1.59 |
| -1 | 0.26 |
| 0 | -1 |
| 1 | -1 |
| 2 | 5.39 |

From the above table, we see that , a root lies in

and , a root lies in

**3.3 Techniques to find real roots**

**3.3.1 Method of Bisection**

Let be continuous in and , then there exists a real root of in . In this method we assume the mid-point is the approximation to the root.

If , we conclude that is a root of . If and

(i) if , the root is in or

**y**

**O**

**α**

****

***x***

***b***

***a***

**Fig 4.2**Bisection Method

(ii) if , the root is in .

By designating the new interval of root as ,

we can calculate the next iterate by the formula

, *n* = 1, 2, 3, . . .

Repeat the process until , where is the specified accuracy.

**MATLAB Code:**

function [a,b,c]=bisectionMethod()

a=1;b=2;tol=0.01;

while abs((b-a))>tol

c=(a+b)/2;

if f(a)\*f(c)<0

b=c;

else

a=c;

end

end

end

function f=f(x)

f=x\*x-3;

end

**Example 3.5**  Find the root of using bisection method with accuracy in the interval [1, 2].

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *A* | *b* | *f(a)* | *f(b)* | *c = (a + b)/2* | *f(c)* | *f(a)\*f(c)* | *Update* | *b − a* |
| 1.0 | 2.0 | -2.0 | 1.0 | 1.5 | -0.75 | > 0 | a = c | 0.5 |
| 1.5 | 2.0 | -0.75 | 1.0 | 1.75 | 0.062 | < 0 | b = c | 0.25 |
| 1.5 | 1.75 | -0.75 | 0.0625 | 1.625 | -0.359 | > 0 | a = c | 0.125 |
| 1.625 | 1.75 | -0.3594 | 0.0625 | 1.6875 | -0.1523 | > 0 | a = c | 0.0625 |
| 1.6875 | 1.75 | -0.1523 | 0.0625 | 1.7188 | -0.0457 | > 0 | a = c | 0.0313 |
| 1.7188 | 1.75 | -0.0457 | 0.0625 | 1.7344 | 0.0081 | < 0 | b = c | 0.0156 |
| 1.71988 | 1.7344 | -0.0457 | 0.0081 | 1.7266 | -0.0189 | > 0 | a = c | 0.0078 |

**3.3.2 Order of Convergence**

There are different iterative methods in finding the roots of an equation. To compare the methods order of convergence is used.

Let be the error in the *n*th iteration for a root α of, then

If

(constant)

then the order of convergence of the sequence is *R*.

In special case,

If , the convergence is called linear.

If , the convergence is called quadratic.

If , the convergence is super linear.

**3.3.3 The Secant Method**

In **secant** method two values of *x* near the root is used and the root is approximated by the *x*-intercept of the secant line (chord) joining the two points. The straight line through the two points and is

On the *x*-axis and let , then





**α**

**y**

**O**

***x***

***x*3**

***x*2**

***x*1**

**Fig.4.3** The Secant Method

Solving for ,

The estimated value will be closer to the root than either of the two initial points. We continue the process to get better approximation of the root by using the last two computed points using the iteration formula

,

Here it is not necessary that the interval should contain the root i.e. . In selecting, and care should be taken so that is closer to the root than to get rapid convergence. This can be achieved by selecting and such that

Secant Method in MATLAB

a=input('Enter function:','s');

f=inline(a)

x(1)=input('Enter first point of guess interval: ');

x(2)=input('Enter second point of guess interval: ');

n=input('Enter allowed Error in calculation: ');

iteration=0;

fori=3:100

x(i) = x(i-1) - (f(x(i-1)))\*((x(i-1) - x(i-2))/(f(x(i-1)) - f(x(i-2))));

iteration=iteration+1;

if abs((x(i)-x(i-1))/x(i))\*100<n

root=x(i)

iteration=iteration

break

end

end

**Example 3.6:** Find the root of using secant method initiating with with tolerancy/accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| n |  |  |  |
| 1 | -0.5136 | 0.1522 | 0.4136 |
| 2 | -0.6100 | 0.0457 | 0.0964 |
| 3 | -0.6514 | 0.0065 | 0.0414 |
| 4 | -0.6582 | 0.0013 | 0.0068 |
| 5 | -0.6598 | 0.0006 | 0.0016 |
| 6 | 0.6595 | 0.0002 | 0.0003 |

# 3.3.4 Newton-Raphson Method

The procedure known as **Newton’s method** is also called **Newton-Raphson method**. In this method, the root of the equation is approximated by the *x*-intercept of the tangent line through a guess value near the root.

The equation of the tangent through is

On the *x*-axis and let , then

Solving for ,

The process can be repeated with the new estimate of *x* until we reach the required degree of accuracy.

In general, the iterative formula for the process can be expressed as

Or for a simple root the convergence of the Newton-Raphson method is of order two.

***x*3**

**y**

***x***

**O**

**α**

***x*2**

***x1***

**Fig.4.4** Newton-Raphson Method

syms x;

fprintf('Newton Raphson Method\n');

Fun=input('\nType a function \n');

xi=input('\nType initial value\n');

F = sym(Fun); % reduce function to symbolic form

dF = diff(F); % evaluate derivative in symbolic form

tol=1.0e-5;

for k=1:10

Fxi = subs(F, 'x', xi); % caalculate value of the function

dFxi = subs(dF, 'x', xi); % calculaate value of derivative

xi2=xi-Fxi/dFxi; % iteration using Newton-Raphson formula

xn=vpa(xi2,3); % reduce vaues to 7 d.p.

IX=[k xn];

disp(IX)

tol2=abs(xn-xi);

if(tol2<tol)

break;

end

xi=xn;

end

**Example 3.7:** Consider the equation .

1. Perform one iteration using Newton-Raphson formula for finding its root near .
2. Write MATLAB syntax for finding the root in using MATLAB function “**fzero”.**

**Solution:**  i.

1. MATLAB code:

>> F=@(x) sin(x)+3\*x-1; % entered as function handle

>> Sol = fzero(F, [0,1])

Sol =

0.2507

# 3.3.5 Starting Value for Iteration

If the starting value is reasonably close to the root, the number of iterations needed will be less and calculation time will be saved. Use specified starting value, if stated. Otherwise, first find an interval in which a root lies and then choose as a starting value, , either

(i) one of the end-points of the interval where the magnitudes of the value of the function is small., or

(ii) guess an internal point of the interval closer to the root.

**3.3.6 Multiple Roots**

Equal (repeated) roots are known as the multiple roots. If the root α of is a repeated root, then we may write

where is bounded and .

The root α is called a *multiple root of multiplicity m*. We obtain from the above equation

,

For a multiple root, the order of convergence is reduced, but the order may be increased by modifying the methods discussed. If the multiplicity *m* of the root is known in advance, then some of the methods can be modified so that they have the same rate of convergence as that for simple roots.

**3.3.7 Modified Newton-Raphson Method**

For a multiple root the order of convergence of the Newton-Raphson formula is linear. The order can be increased by the modified formula

where *m* is the multiplicity of the root.

The order of convergence of the above is two as that of the simple root.

When the multiplicity of the root is not known in advance, we may proceed as follows;

The function

has a simple root α regardless of the multiplicity of the root of .

When the Newton-Raphson method is applied to the simple root αof , we have

or

.

**Example 3.8** The equation has a double root near .

Compute the iterative results using modified Newton-Raphson formula.

Here we may take

and

Newton-Raphson iterative formula for double root is

|  |  |  |  |
| --- | --- | --- | --- |
| N | xn | f(xn) | f'(xn) |
| 0 | 0.5 | 27 | -220.5 |
| 1 | 0.7449 | 0.70855 | 46.93183 |
| 2 | 0.7147 | 0.00013 | 0.609353 |
| 3 | 0.7143 | 1.5E-07 | 0.021 |
| 4 | 0.7143 | 1.5E-07 | 0.021 |

**3.3.8 Fixed Point Iteration Method**

A **fixed point** of a function is a real number α such that . This means α is a root of the equation . To find a root of the equation by an iterative method, first rearrange the equation into a form The function is called the iteration function. Note that there is no unique form into which the equation can be rearranged.

An iteration formula is then

If is an approximation close to a root of and is an iterative formula used to find the root of the equation near *x*0, then

(i) if , the sequence will converge to the root. In particular,

(a) if , the sequence will oscillate and converge to the root

(b) if , the sequence will converge to the root without oscillating.

(ii) if , the sequence will diverge.

(iii) if , indicates the root near to exact root.

**Example 3.9 :** The equation has a root near Using following iteration formulae, perform few iterations and comment on the results.

(a) (b) (c)

The calculation using the three iteration formulae are as follows:



The iteration function in (a) is and

With , .

So and the sequence will not converge to the root as shown in the above table.

The iteration function in (b) is and

With , .

Since is negative and , the sequence will converge to the root with oscillation as shown in the above table.

The iteration function in (c) is and

with , .

Since is positive and with small value, the sequence will converge rapidly to the root without oscillation as shown in the above table.

**Example 3.10 :** Given that .

|  |  |  |
| --- | --- | --- |
| a. | i. | Find the number of real roots of the equation . |
|  | ii. | For each root, find an interval where it lies. |
| b. | i. | Show that the equation has a root in . |
|  | ii. | Use bisection method twice to find the new smaller interval where the root lies. |
| c. |  | Apply secant method once to find the root to 2 d.p in the last interval obtained by using Bisection method. |
| d. | i. | Write down an iteration formula based on Newton-Raphson formula. |
|  | ii. | Perform one iteration starting with the value obtained in (c). |
|  | iii. | Write down MATLAB commands to execute the iteration four times. |
| e. |  | An iterative formula can be used to estimate the root of |
|  | i. | State with reason whether the iterative formula will converge to the root near . |
|  | ii. | If the iterative formula converges to the root do the iteration two times to estimate the root to 3 decimal places |
|  | iii. | Write down MATLAB commands to execute the iterations five times. |

**Solution: a.i.**

The equation i.e. can be written as

Consider the graphs of and.The two curves intersect at three points and hence the number of real roots is 3.

A picture containing wire, electronic

Description automatically generated

ii. Consider the values of *f*(*x*) for different values of *x*:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | *x* | *f(x)* | | 0 | 4 | | 1 | 0.2 | | 2 | -1.3 | | 3 | 0.9 | | 4 | -2.3 | | From the table we see that  *f*(1)*f*(2) = - 0.26 <0, a root lies in (1, 2)  *f*(2)*f*(3) = - 1.17 <0, a root lies in (2, 3)  *f*(3)*f*(4) = - 2.27 <0, a root lies in (3, 4) |

b. i. It can be seen that and

Here . Thus, a root lies in .

ii. Applying bisection method on , we have

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *A* | *b* | *f(a)* | *f(b)* | *c = (a + b)/2* | *f(c)* | *f(a)\*f(c)* | *Update* | *b − a* |
| 0.8 | 1.6 | 1.142 | -1.596 | 1.2 | -0.675 | < 0 | b = c | 0.8 |
| 0.8 | 1.2 | 1.142 | -0.675 | 1.0 | 0.16771 | > 0 | a = c | 0.4 |
| 1.0 | 1.2 | 0.16771 | -0.675 | 1.1 |  |  |  |  |

c. Given starting values for secant method is and

|  |  |  |
| --- | --- | --- |
| n |  |  |
| 1 | 1 | 0.1677 |
| 2 | 1.2 | -0.6748 |
| 3 | 1.0398 |  |

Root to 2 d.p. is .

d. i. When Newton-Raphson method applied to the equation, we have

.

Newton-Raphson iterative formula is

ii. Using the starting value from (c), we may proceed as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| *n* |  |  |  |
| 0 | 1.04 |  |  |
| 2 |  |  |  |

iii. f=@(x) 2\*cos(2\*x)+2-x;

fd=@(x) -4\*sin(2\*x)-1;

x(1)=1.04;

for n=1:4

x(n+1)=x(n)-f(x(n))/fd(x(n));

end

**Output:** >> Solution = x

1.0400 1.0367 1.0367 1.0367 1.0367

**e.** i. In this case the iterative function is given by

The iterative formula will converge to the root near 3.5 since

ii. With , the successive iterates are given below:

|  |  |  |
| --- | --- | --- |
| *n* |  |  |
| 1 | 3.5 |  |
| 2 | 3.501951 | 0.01951 |
| 3 | 3.502130 | 0.000179 |

iii. f=@(x) (2+3\*x+2\*cos(2\*x))/4;

x(1)=3.5;

for n=1:5

x(n+1)=f(x(n));

end

Output: Solution = x

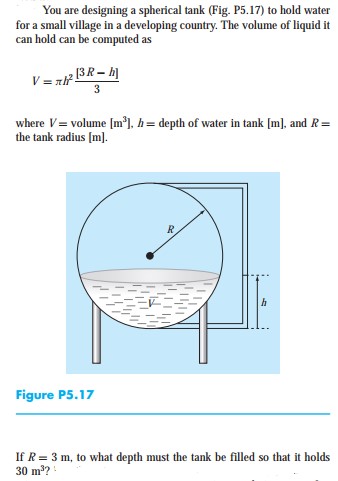
3.5000 3.5020 3.5021 3.5021 3.5021 3.5021

**Exercise 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **1.** |  | Given the following polynomial equations and an interval. | |
|  |  | a.  b. | c.  d. |
|  | **i.** | Find the number of real roots of the equation by graphical method. Find also the number of complex roots, if any. | |
|  | **ii.** | Apply bisection method two times in the given interval to find the new smaller interval of this root. | |
|  | **iii.** | Apply secant method to estimate the root correct to 2 d.p. in the last interval acquired by using bisection method. | |
|  | **iv.** | Write down an iteration formula based on Newton-Raphson method. | |
|  | **v.** | Perform one iteration starting using the above formula (iv.) with a suitable value in the given interval to estimate the root to 2 d.p. | |
|  | **vi.** | Write down MATLAB codes to execute the iteration four times. | |
|  | **vii.** | Write MATLAB commands “**roots**” to find all the roots including complex roots. | |
| **2.** |  | Given the following nonlinear equations. | |
|  |  | a.  b.  c.  d. | e.  f.  g.  h.  i. |
|  |  |  | |
|  | **i.** | For each functions, find a suitable interval for which the root lies. | |
|  | **ii.** | Use the bisection method two times to find the new smaller interval in one of the interval obtained in(ii). | |
|  | **iii.** | Use secant method in the interval obtained in (iii) to find the root of the equation correct to 2 decimal places. | |
|  | **iv.** | Write an iterative formula based on Newton-Raphson method and iterate 2 times with a suitable value. | |
|  | **v.** | Write MATLAB codes to execute the iterative formula five times in part (v). | |
|  | **vi.** | Use MATLAB builtin function “**fzero”** to find all the roots of each equation using an interval or a guess value. | |

|  |  |  |
| --- | --- | --- |
| **3.** |  | Given the equation . |
|  | **i.** | Apply Newton-Raphson method in the interval to estimate the root correct to 3 d.p. |
|  | **ii.** | The following iterative formulae are suggested to estimate the root of the above equation.  a. b.  State with reason which iterative formula will converge faster to the root near . |
|  | **iii.** | Use the suitable iterative formula from the above two (a) and (b) to find the root correct to 2 decimal places. |
|  | **iv.** | Write MATLAB codes to execute the iterative formula in (iii) five times. |
|  |  |  |
| **4.** |  | Given the equation |
|  | **i.** | Apply bisection method **two** times in the interval to find the new smaller interval where the root lies. |
|  | **ii.** | The following iterative formulae are suggested to estimate the root of the above equation  State with reason which iterative formula will converge faster to the root near . |
|  | **iii.** | Use the suitable iterative formula from the above two ((i) and (ii)) to find the root correct to 2 decimal places. |
|  | **iv.** | Use MATLAB built-in function “**fzero(fun,x0)”** to find all the roots of the given equation in the interval . |
|  |  |  |
| **5.**Given the equation . | | |
| i. The following iterative formulae are suggested to estimate the root of the above equation.  State with reason which iterative formula will converge faster to the root near . | | |
| ii. Use the suitable iterative formula from the above two (a) and (b) to find the root correct to 2 decimal places. | | |
| iii. Write MATLAB codes to execute the iterative formula used in (ii) five times. | | |

|  |  |  |
| --- | --- | --- |
| **6.** |  | Consider the polynomial equation |
|  | **a.** | Find the number of real roots of the equation by graphical method. Find also the number of complex roots, if any. |
|  | **b.** | Show that the equation has a root in . |
|  | **c.** | An iterative formula may be used to estimate the root. Verify whether the iterative formula will convergeto the root near. |
|  | **d.** | If the iterative formula converges to the root do the iteration two times to estimate the root to 3 decimal places. |
|  | **e.** | Write MATLAB commands “**roots(p)**” to find all the roots including complex roots. |

7.

Use N-R method to calculate the depth.

8.The volume of the liquid Vin a hollow horizontal cylinder of radius *r* and length *L* is related to the depth of the liquid *h by*

Determine *h* given