Gurriday Chawla

102108211

2MEE8

**LAB # 2 : Root Finding Methods**

**1 Introduction**

This lab is concerned with several ways to compute roots (zeros) of a nonlinear equation. All the methods for computing roots are iterative in nature. We begin with a short discussion of each method and then write the algorithms, and then show some applications of methods to solve some problems. You may find it convenient to print the pdf version of this lab rather than the web page itself. This lab will take two sessions.

**2 Iterative Methods**

We consider *f*(*x*) = 0, where *f* is a continuous function. The following iterative methods will be implemented in Matlab.

**3 Bisection Method**

The first technique, based on the Intermediate Value Theorem, is called the Bisection, or Binary-search, method.

Let *f*(*x*) be a continuous function on some given interval [*a,b*] and it satisfies the condition *f*(*a*) *f*(*b*) *<* 0, then by Intermediate Value Theorem the function *f*(*x*) must have at least one root *α* in (*a,b*) with *f*(*α*) = 0. The bisection method repeatedly bisects the interval [*a,b*] by taking and then selects a subinterval in which a root must lie for further processing. It is a very simple and robust method, but it is also relatively slow. Usually [*a,b*] is chosen to contain only root *α*.

**Algorithm:** To determine a root of *f*(*x*) = 0 that is accurate within a specified tolerance value , given values *a* and *b* such that *f*(*a*) *f*(*b*) *<* 0.

Define .

If *f*(*a*) *f*(*c*) *<* 0, then set *b* = *c,* otherwise *a* = *c.*

End if.

Until (tolerance value).

Print root as *c*.

**Stopping Criteria:** Since this is an iterative method, we have used stopping criteria . We can also use must the criterion |*f*(*ck*)| very small for some cases but this can be misleading since it is possible to have |*f*(*ck*)| very small, even if *ck* is not close to the root.

The interval length after *N* iterations is . So, to obtain an accuracy of , we must have *.*

1

**Problems:**

1. **Find root of 3*x* − *ex* = 0 with an accuracy of 0*.*0001. Ans:**

a=input('Enter the intial value a:');

b=input('Enter the intial value b:');

eb=10^-4;

f=@(x) 3\*x - exp(x);

if f(a)\*f(b) > 0

fprintf("intial value is wrong");

else

c=(a+b)/2;

e=abs(f(c));

while e>eb

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

b=c;

else

a=c;

end

c=(a+b)/2;

e=abs(f(c));

end

end

fprintf("root is %d:",c)

**Output:**

bisection11

Enter the intial value a:

1

Enter the intial value b:

5

root is 1.512085e+00:

2. **Find intersection points of two curves *y* = *x* and *y* = cos*x* with an accuracy of 0*.*001.**

**Ans:**

a=input('Enter the initial value a:');

b=input('Enter the initial value b:');

eb=0.001;

f=@(x) x-cos(x);

if f(a)\*f(b)>0

fprintf("Initial value is wrong");

else

c=(a+b)/2;

2

e=abs(f(c));

while e>eb

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

b=c;

else

a=c;

end

c=(a+b)/2;

e=abs(f(c));

end

end

fprintf("root is %d:",c)

**Output:**

Bisection22

Enter the initial value a:

2

Enter the initial value b:

-2

root is 7.392578e-01

3. **An object falling vertically through the air is subjected to viscous resistance as well as to the force of gravity. Assume that an object with mass *m* is dropped from a height *s*0 and that the height of the object after *t* seconds is**

***,***

**where *g* = 32*.*17 ft/*s*2 and *k* represents the coefficient of air resistance in lb-s/ft. Suppose *s*0 = 300 ft, *m* = 0*.*25 lb, and *k* = 0*.*1 lb-s/ft. Find, to within 0*.*01 s, the time it takes this quarter pounder to hit the ground.**

**Sol:**

a=input('Enter the initial value a:');

b=input('Enter the initial value b:');

eb=0.01;

f=@(x) 300 - (0.25\*32.17\*x)/0.1 + (2.5\*2.5\*32.17\*(1-exp(-x/2.5)));

if f(a)\*f(b) >0

fprintf("Initial value is wrong");

else

3

c=(a+b)/2;

e=abs(f(c));

while e>eb

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

b=c;

else

a=c;

end

c=(a+b)/2;

e=abs(f(c));

end

end

fprintf("root is %d:",c)

**Output:**

bisection33

Enter the initial value a:

6

Enter the initial value b:

-6

root is -3.361542e+00:

**4 Fixed-Point Method**

A point *x* is a fixed point for a given function *g* if *g*(*x*) = *x.*

We can rewrite the root finding problem *f*(*x*) = 0 as as fixed-point problem *x* = *g*(*x*). Root-finding problems and fixed-point problems are equivalent classes in the following sense:

Given a root-finding problem *f*(*x*) = 0, we can define functions *g* with a fixed point at *x* in a number of ways. Conversely, if the function *g* has a fixed point at *α*, then the function defined by *f*(*x*) = *x*−*g*(*x*) has a zero at *α*.

**Algorithm:** To find a solution to *x* = *g*(*x*) given an initial approximation *x*0.

Input: Initial approximation *x*0*,* tolerance value *,* maximum number of iterations *N*. Output: Approximate solution *α* or message of failure.

Step 1: Set *i* = 1*.*

Step 2: While *i* ≤ *N* do Steps 3 to 6. Step 3:

Set *x*1 = *g*(*x*0).

4

Step 4: If then Output *x*1; (The procedure was successful.) STOP.

Step 5: Set *i* = *i* + 1.

Step 6: Set *x*0 =*x*1. (Update *x*0*.*) Step 7:

Print the output and STOP.

**Problems:**

1. **Find positive root of *x*3−7*x*+2 = 0 with an accuracy of 10−3. Compute the absolute errors and plot the errors with number of iterations.**

**Ans:**

x0=input('Enter the initial guess');

n=input('number of iterations');

f=@(x) x-((x^3-7\*x+2)/(3\*x^2-7));

eb=0.001;

for i=1:n

x1=f(x0);

if abs(x0-x1)<eb

break

end

x0=x1;

end

disp(x0)

**Output:**

Enter the initial guess

1

number of iterations

10

0.2892

2. **Find smallest and second smallest positive roots of the equation tan*x* = 4*x,* with an accuracy of 10−3.**

**Ans:**

x0=input('Enter the initial guess');

n=input('Number of iteration');

eb=0.001;

5

f=@(x) (tan(x))/4;

for i=1:n

x1=f(x0);

if abs(x0-x1)<eb

break

end

x0=x1;

end

disp(x0)

Output:

Enter the initial guess

1

Number of iteration

10

4.0219e-04

3. Determine a solution accurate to within 10−2 for 2sin*πx* + *x* = 0 on [1*,*2]. Use initial guess *x*0 = 1*.*

4. The equation *f*(*x*) = *x*3 + 4*x*2 − 10 = 0 has a unique root in [1*,*2]. There are many ways to change the equation to the fixed-point form *x* = *g*(*x*) using simple algebraic manipulation. Let *g*1*,g*2*,g*3*,g*4*,g*5 are iteration functions obtained by the given function, then check which of the following iteration functions will converge to the fixed point? 

(a) *g*1(*x*) = *x* − *x*3 − 4*x*2 + 10 (b)

√

(c) *g*3(*x*) = 0*.*5 10 − *x*3

*.*

Sol.

**5 Newton’s Method**

The iterations of Newton’s method are given by



**Algorithm:** To find a solution to *f*(*x*) = 0*,* given an initial approximation *x*0.

6

Input: Initial approximation *x*0*,* tolerance value *,* maximum number of iterations *N*. Output: Approximate solution *x*1 or message of failure.

Step 1: Set *i* = 1*.*

Step 2: While *i* ≤ *N* do Steps 3 to 6.

Step 3: Set .

Step 4: If then OUTPUT *x*1; (The procedure was successful.) STOP.

Step 5: Set *i* = *i* + 1.

Step 6: Set *x*0 =*x*1. (Update *x*0*.*)

Step 7: Print the Output and Stop.

Step 8: Plot the errors with number of iterations.

**Problems:**

**√**

1. **Compute 17 with an accuracy of 0*.*005.**

**Ans**

f=@(x) x^2 - 17;

g=@(x) 2\*x;

tol=0.001;

x0=input('Enter the initial approximation');

n=input('Enter the no of iteration');

for i=1:n

x1=x0-(f(x0)/g(x0));

if abs(x0-x1)<tol

break

else

x0=x1;

end

end

disp(x1)

**Output:**

newton

Enter the initial approximation

1

Enter the no of iteration

10

4.1231

2. **Use Newton’s method to find the root of cos*x* − *xex* = 0 with an accuracy 10−3. Also plot the errors.**

7

**Ans**

f = @(x) cos(x)-x\*exp(x);

g= @(x) - exp(x) - sin(x) - x\*exp(x);

tol=0.001;

x0=input('Enter the initial approximation');

n=input('Enter the no of iteration');

for i=1:n

x1=x0-(f(x0)/g(x0));

if abs(x0-x1)<tol

break

else

x0=x1;

end

end

disp(x1)

3. Solve the equation 4*x*2 −*ex* −*e*−*x* = 0 which has two positive solutions *x*1 and *x*2. Use Newton’s method to approximate the solution to within 10−5 with the following values of *x*0:

*x*0 = −10*,*−5*,*−3*,*0*,*1*,*3*,*5*,*10*.*

Sol.

**6 Secant Method**

The Successive iterations in secant method are given by



**Algorithm:**

1. Give inputs and take two initial guesses *x*0 and *x*1.

2. Start iterations

*.*

3. If



8

then stop and print the root.

4. Repeat the iterations (step 2). Also check if the number of iterations has exceeded the maximumnumber of iterations. **Problems:**

1. **Use secant method to compute 71*/*3 with an accuracy of 0*.*01. Ans**

f=@(x) x^3-17;

x0=input('Enter the intial number x0:');

x1=input('Enter the intial number x1:');

tol=0.001;

n=input('enter no of iteration');

for i=1:n

x2=((x1-x0)/(f(x1)-f(x0)))\*f(x1);

if abs(x2-x1)<tol

break

else

x0=x1;

x1=x2;

end

end

disp(x2)

**output:**

secant

Enter the intial number x0:

1

Enter the intial number x1:

8

enter no of iteration

10

-2.5961

2. **Apply secant method to find the one positive and one negative root (nearest to zero) of the equation *ex* = cos*x* with relative error less than *<* 0*.*001%.**

**Sol.**

f=@(x) cos(x)-exp(x);

x0=input('Enter the intial number x0:');

x1=input('Enter the intial number x1:');

tol=0.001;

n=input('enter no of iteration');

for i=1:n

x2=((x1-x0)/(f(x1)-f(x0)))\*f(x1);

if abs(x2-x1)<tol

break

9

else

x0=x1;

x1=x2;

end

end

disp(x2)

**Output**

Enter the intial number x0: 10

Enter the intial number x1: 1

enter no of iteration 10

4.0879e-04

10