Non-Linear Algebraic Equations

(Single variable and multi-variable systems, Roots of polynomials)

- Bisection method
- Newton-Raphson method
- Secant method
- Computer-based solutions
  - NEQNF subroutine (Fortran IMSL subroutine that Solves a system of nonlinear equations using a modified Powell hybrid algorithm and a finite-difference approximation to the Jacobian)
  - MATLAB solution of system of non-linear equations (fzero, roots functions)

Solution of Nonlinear Equations

(Root finding Problems)

- Definitions
- Classification of methods
  - Analytical solutions
  - Graphical methods
  - Numerical methods
    - « Bracketing methods
    - « Open methods
- Convergence Notations

Solution of Nonlinear Equations

Many problems in Science and Engineering are expressed as

Given a continuous function $f(x)$,
find the value r such that $f(r) = 0$

These problems are called root finding problems

Roots of Equations

A number *r* that satisfies an equation is called a root of the equation.

The equation $x^4 - 3x^3 - 7x^2 + 15x = -18$
has four roots $-2, 3, 3$ *and* -1

The equation has two simple roots (-1 and -2)
and a repeated root (3) with multiplicity $= 2$

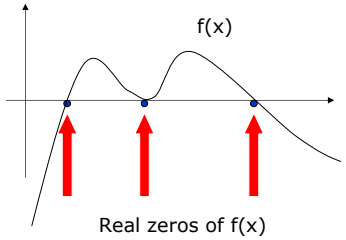
Zeros of a function

Let *f(x)* be a real-valued function of a real variable. Any number *r* for which *f(r)=0* is called a zero of the function.

Examples:

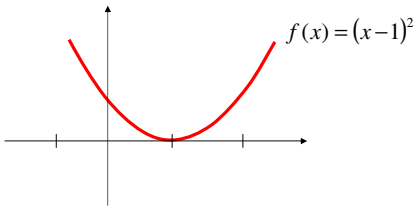
2 and 3 are zeros of the function f(x) = (x-2)(x-3)

Graphical Interpretation of zeros



- The real zeros of a function f(x) are the values of x at which the graph of the function crosses (or touches) the x-axis.

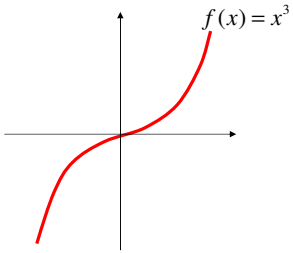
Multiple zeros



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

has double zeros (zero with muliplicity = 2) at x = 1

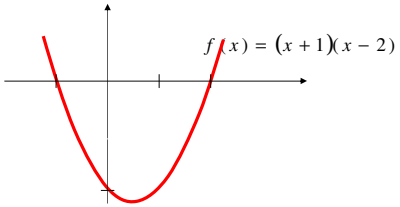
Multiple zeros



$f(x) = x^3$

has a zero with muliplicit y = 3 at x = 0

Simple zeros



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

has two simple zeros (one at $x=2$ and one at $x=-1$)

Facts

- Any nth order polynomial has exactly n zeros (counting real and complex zeros with their multiplicities).
- Any polynomial with an odd order has at least one real zero.
- If a function has a zero at ***x=r*** with multiplicity ***m*** then the function and its first ***(m-1)*** derivatives are zero at ***x=r*** *and the* ***m***th *derivative at* ***r*** *is not zero.*

Roots of Equations & Zeros of function

Given the equation

$$x^4-3x^3-7x^2+15x=-18$$

Move all terms to one side of the equation

$$x^4-3x^3-7x^2+15x+18=0$$

Define $f(x)$ *as*

$$f(x)=x^4-3x^3-7x^2+15x+18$$

The zeros of $f(x)$ are the same as the roots of the equation

(Which are -2, 3, 3 and -1)

Roots of Equations with Matlab

- The ROOTS function allows you to calculate the roots of a polynomial function.
- to find the roots of the following function: $f(x)=x^3-5x^2-x+2$
- MATLAB Command:
  - First, the polynomial function is represented by a row vector containing the coefficients. For the given example, this can be done as follows:

    >> *f* = [1 -5 -1 2]

  - Next, type the following command to obtain the roots of the equation

    >> *x* = roots (*f*)

Roots of Equations with Matlab

- *fzero* function finds zeros of the function of a single-variable
- MATLAB Command

  >> x = fzero(***fun***,x0)

  - tries to find a zero of an function defined by *fun* near x0, if x0 is a scalar.
  - ***fun*** is an M-file function
  - The value x returned by fzero is near a point where fun changes sign, or NaN if the search fails. In this case, the search terminates when the search interval is expanded until an Inf, NaN, or complex value is found.

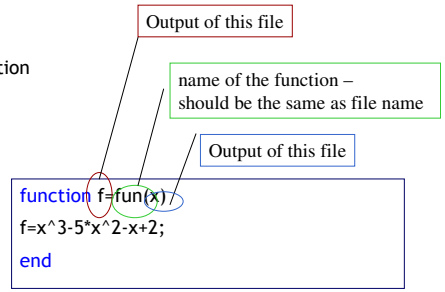
NaN: 'not a number'. It represents a 'no solution' for the function.

Roots of Equations with Matlab

- *fzero* function finds zeros of the function of a single-variable
- MATLAB Command

  >> x = fzero(***fun***,x0)

  - ***fun*** is an M-file function



Solution Methods

Several ways to solve nonlinear equations are possible.

- Analytical Solutions
  - possible for special equations only
- Graphical Solutions
  - Useful for providing initial guesses for other methods
- Numerical Solutions
  - Open methods
  - Bracketing methods

Solution Methods

Analytical Solutions

Analytical Solutions are available for special equations only.

Analytical solution of $a\,x^2 + b\,x + c = 0$

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

No analytical solution is available for $x - e^{-x} = 0$

Solution Methods
Graphical Methods

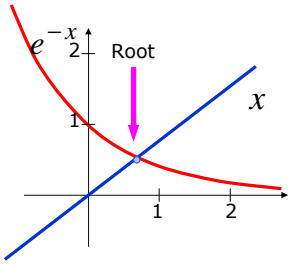
- Graphical methods are useful to provide an initial guess to be used by other methods

Solve

$x = e^{-x}$

The root $\in [0,1]$

root ≈ 0.6



Bracketing Methods

- In bracketing methods, the method starts with an interval that contains the root and a procedure is used to obtain a smaller interval containing the root.
- Examples of bracketing methods :
  - Bisection method
  - False position method

Open Methods

- In the open methods, the method starts with one or more initial guess points. In each iteration a new guess of the root is obtained.
- Open methods are usually more efficient than bracketing methods
- They may not converge to the a root.

Solution Methods

Many methods are available to solve nonlinear equations

Bisection Method
Newton's Method
Secant Method

→ These will be covered in ChE401

- False position Method
- Muller's MethodBairstow's Method
- Bairstow's Method
- Fixed point iterations
- .....

Convergence Notation

A sequence $x_1, x_2, ..., x_n, ...$ is said to **converge** to x *if* to every $\varepsilon > 0$ there exist N such that

$$|x_n - x| < \varepsilon \quad \forall n > N$$

Convergence Notation

Let $x_1, x_2, ...,$ converges to x

Linear Convergence $\dfrac{|x_{n+1} - x|}{|x_n - x|} \le C$

Quadratic Convergence $\dfrac{|x_{n+1} - x|}{|x_n - x|^2} \le C$

Convergence of order p $\dfrac{|x_{n+1} - x|}{|x_n - x|^p} \le C$

Speed of convergence

- Different methods can be caomapred in terms of their convergence rate.
- Quadratic convergence is faster than linear convergence.
- A method with convergence order q converges faster than a method with convergence order p if q>p.
- A Method of convergence order p>1 are said to have super linear convergence.

Bisection Method

- The Bisection Algorithm
- Convergence Analysis of Bisection Method

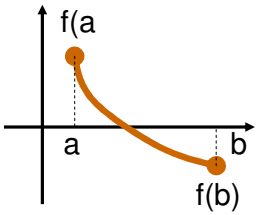
Bisection Method: Introduction

- The **Bisection method** is one of the simplest methods to find a zero of a nonlinear function.
- It is also called **interval halving** method.
- To use the Bisection method, one needs an initial interval that is known to contain a zero of the function.
- The method systematically reduces the interval. It does this by dividing the interval into two equal parts, performs a simple test and based on the result of the test half of the interval is thrown away.
- The procedure is repeated until the desired interval size is obtained.

Intermediate Value Theorem

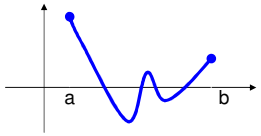
- Let f(x) be defined on the interval [a,b],
- Intermediate value theorem:

  if a function is continuous and f(a) and f(b) have different signs then the function has at least one zero in the interval [a,b]

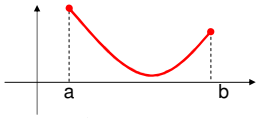


Examples

- If f(a) and f(b) have the same sign, the function may have an even number of real zeros or no real zero in the interval [a,b]
- Bisection method can not be used in these cases



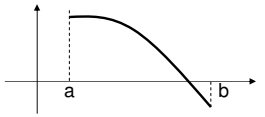
The function has four real zeros



The function has no real zeros

Two more Examples

- If f(a) and f(b) have different signs, the function has at least one real zero
- Bisection method can be used to find one of the zeros.



The function has one real zero

The function has three real zeros

Two more Examples

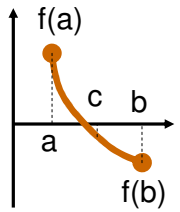
- If the function is continuous on [a,b] and f(a) and f(b) have different signs, Bisection Method obtains a new interval that is half of the current interval and the sign of the function at the end points of the interval are different.

- This allows us to repeat the Bisection procedure to further reduce the size of the interval.

Bisection Algorithm

Assumptions:

- f(x) is continuous on [a,b]
- f(a) f(b) < 0



Algorithm:

Loop

1. Compute the mid point c=(a+b)/2
2. Evaluate f(c)
3. If f(a) f(c) < 0 then new interval [a, c]
   If f(a) f(c) > 0 then new interval [c, b]

End loop

Bisection Method

Assumptions:

Given an interval [a,b]

f(x) is continuous on [a,b]

f(a) and f(b) have opposite signs.

These assumptions ensures the existence of at least one zero in the interval [a,b] and the bisection method can be used to obtain a smaller interval that contains the zero.

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. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

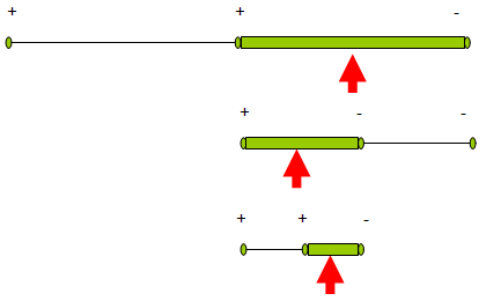
$$x_r = \frac{x_l + x_u}{2}$$

Step 3: Make the following evaluations to determine in which subinterval the root lies:
(a) If $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to step 2.
(b) If $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to step 2.
(c) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

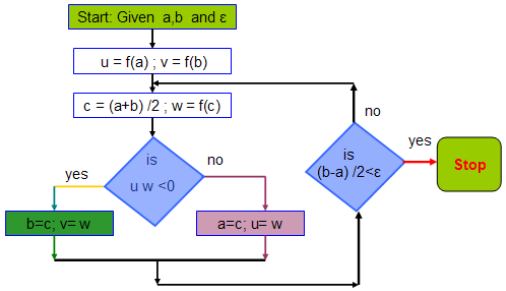
Relative error estimate: $$\varepsilon = \frac{\left|x_r^{new} - x_r^{old}\right|}{\left|x_r^{new}\right|} 100\%$$

Termination criteria: $\varepsilon < Epsilon$ OR $Max.Iteration$ is reached

Example



Flow chart of Bisection Method



Bisection Method: Matlab code

```
% Bisection Method - simple
% function f(x) = exp(-x) - x = 0   sample call:  bisection(-2, 4, 0.001,500)
function  root = bisection(xl, xu, es, imax);
if ((exp(-xl) - xl)*(exp(-xu) - xu))>0       % if guesses do not bracket, exit
    disp('no bracket')
    return
end
for i=1:1:imax
   xr=(xu+xl)/2;                      % compute the midpoint   xr
   ea = abs((xu-xl)/xl);              % approx. relative error
   test= (exp(-xl) - xl) * (exp(-xr) - xr);    % compute    f(xl)*f(xr)
   if (test < 0)    xu=xr;
   else    xl=xr;
   end
   if (test == 0) ea=0;   end
   if (ea < es)  break;   end
end
s=sprintf('\n  Root= %f    #Iterations = %d \n', xr,i); disp(s);
```


Bisection Method: iterations

- Length of the first Interval $L_o = x_u - x_l$
- After 1 iteration $L_1 = L_o/2$
- After 2 iterations $L_2 = L_o/4$

  ….. …..
- After k iterations $L_k = L_o/2^k$

- Then:

$$\frac{L_k}{x_l} \le desired\_relative\_error$$

$$\frac{L_0}{2^k} \le x_l * \varepsilon_{es}$$

$$2^k \ge \frac{L_0}{x_l * \varepsilon_{es}} \Rightarrow k \ge \log_2\left(\frac{L_0}{x_l * \varepsilon_{es}}\right)$$

Example

Can you use Bisection method to find a zero of
$f(x) = x^3 - 3x + 1$ in the interval $[0,2]$?

Answer:

$f(x)$ is continuous on $[0,2]$

$f(0) * f(2) = (1)(3) = 3 > 0$

Assumptions are not satisfied

Bisection method can not be used

Example

Can you use Bisection method to find a zero of
$f(x) = x^3 - 3x + 1$ in the interval $[0,1]$?

Answer:

$f(x)$ is continuous on $[0,1]$

$f(0) * f(2) = (1)(-1) = -1 < 0$

Assumptions are satisfied

Bisection method can be used

Best Estimate and error level

Bisection method obtains an interval that is guaranteed to contain a zero of the function

Questions:

- What is the best estimate of the zero of f(x)?
- What is the error level in the obtained estimate?

Best Estimate and error level

The best estimate of the zero of the function is the mid point of the last interval generated by the Bisection method.

$$Estimate \quad of \ the \ zero \quad r = \frac{b+a}{2}$$

$$Error \le \frac{b-a}{2}$$

Stopping Criteria

Two common stopping criteria

1. Stop after a fixed number of iterations
2. Stop when the absolute error is less than a specified value

How these criteria are related?

Stopping Criteria

c_n *is* the midpoint of the interval at the n th iteration

(c_n is usually used as the estimate of the root).

r is the zero of the function

After n iterations

$$|error| = |r - c_n| \le \frac{b-a}{2^{n+1}}$$

Convergence Analysis

Given $f(x)$, a, b *and* ε

How many iterations are needed such that $|x - r| \le \varepsilon$

where r is the zero of f(x) and x is the bisection estimate (i.e. $x = c_k$)

$$n \ge \frac{\log(b-a) - \log(2\varepsilon)}{\log(2)}$$

Convergence Analysis

Given $f(x)$, a, b *and* ε

How many iterations are needed such that $|x - r| \le \varepsilon$

where r is the zero of f(x) and x is the bisection estimate (i.e. $x = c_k$)

$$n \ge \log_2\left(\frac{\text{width of initial interval}}{\text{width of desired interval}}\right) = \log_2\left(\frac{b-a}{2\varepsilon}\right)$$

Example

$a = 6,\ b = 7,\ \varepsilon = 0.0005$

How many iterations are needed such that $|x - r| \le \varepsilon$

$$n \ge \frac{\log(b-a) - \log(2\varepsilon)}{\log(2)} = \frac{\log(1) - \log(0.001)}{\log(2)} = 9.9658$$

$$\Rightarrow n \ge 10$$

Example

- Use Bisection method to find a root of the equation x = cos (x) with absolute error <0.02

  (assume the initial interval [0.5,0.9])

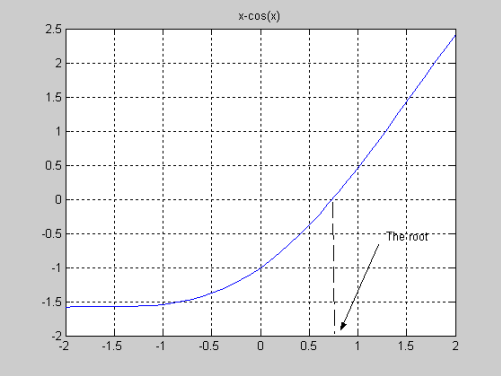
Question 1: What is *f (x)* ?

Question 2: Are the assumptions satisfied ?

Question 3: How many iterations are needed ?

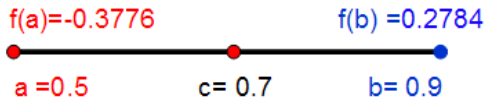
Question 4: How to compute the new estimate ?

Example

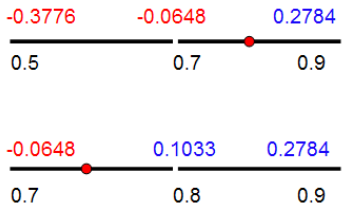


Bisection Method

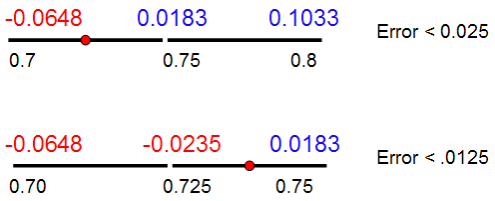
Initial Interval



Bisection Method



Bisection Method



Summary

- Initial interval containing the root [0.5,0.9]
- After 4 iterations
  - Interval containing the root [0.725 ,0.75]
  - Best estimate of the root is 0.7375
  - | Error | < 0.0125

Programming Bisection Method

```
a=.5; b=.9;
u=a-cos(a);
v= b-cos(b);
   for i=1:5
        c=(a+b)/2
        fc=c-cos(c)
        if u*fc<0
          b=c ; v=fc;
        else
           a=c; u=fc;
        end
   end
```

```
c =
    0.7000
fc =
   -0.0648
c =
    0.8000
fc =
    0.1033
c =
    0.7500
fc =
    0.0183
c =
    0.7250
fc =
   -0.0235
```

Example

Find the root of

$f(x) = x^3 - 3x + 1$ in the interval $[0,1]$

* $f(x)$ is continuous
* $f(0) = 1, \quad f(1) = -1 \Rightarrow f(a)\, f(b) < 0$
* Bisection method can be used to find the root

Example

Iteration	A	B	c= (a+b)/2	f(c)	(b-a)/2
0	0	1	0.5	-0.375	0.5
1	0	0.5	0.25	0.266	0.25
2	0.25	0.5	.375	-7.23E-3	0.125
3	0.25	0.375	0.3125	9.30E-2	0.0625
4	0.3125	0.375	0.34375	9.37E-3	0.03125

Bisection Method

Advantages

- Simple and easy to implement
- One function evaluation per iteration
- The size of the interval containing the zero is reduced by 50% after each iteration
- The number of iterations can be determined a priori
- No knowledge of the derivative is needed
- The function does not have to be differentiable

Disadvantage

- Slow to converge
- Good intermediate approximations may be discarded

Newton-Raphson Method

- Assumptions
- Interpretation
- Examples
- Convergence Analysis

Newton-Raphson Method

Given an initial guess of the root x_0 , Newton-Raphson method uses information about the function and its derivative at that point to find a better guess of the root.

Assumptions:

- f (x) is continuous and first derivative is known
- An initial guess x_0 such that f '(x_0) ≠0 is given

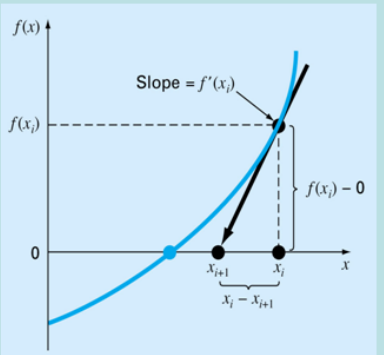
Newton-Raphson Method

- Most widely used formula for locating roots.
- Can be derived using **Taylor series** or the geometric interpretation of the slope in the figure

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

rearrange to obtain :

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



Newton's Method

Given $f(x),\ f'(x),\ x_0$

Assumption $f'(x_0) \neq 0$

for $i = 0:n$

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

end

```
C   FORTRAN PROGRAM
      F(X) = X**3-3*X**2+1
      FP(X) = 3*X**2-6*X
      X = 4
      DO 10 I = 1,5
      X = X - F(X)/FP(X)
      PRINT *, X
10    CONTINUE
      STOP
      END
```


Newton's Method

Given $f(x),\ f'(x),\ x_0$

Assumption $f'(x_0) \neq 0$

for $i = 0:n$

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

end

F.m

```
function [F] = F(X)
F = X^3 - 3*X^2 + 1
```

FP.m

```
function [FP] = FP(X)
FP = 3*X^2 - 6*X
```

```
% MATLAB PROGRAM
X = 4
for i = 1:5
    X = X - F(X)/FP(X)
end
```

Derivation of Newton's Method

Given : x_0 an initial guess of the root of $f(x) = 0$

Question : How do we obtain a better estimate?

Taylor Therorem : $f(x+h) \approx f(x) + f'(x)h$

Find h such that $f(x+h) = 0$.

$$\Rightarrow h \approx -\frac{f(x)}{f'(x)}$$

a new guess of the root $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

Example

Find a zero of the function $f(x) = x^3 - 2x^2 + x - 3$, $x_0 = 4$

$f'(x) = 3x^2 - 4x + 1$

Iteration 1 : $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 4 - \frac{33}{33} = 3$

Iteration 2 : $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 3 - \frac{9}{16} = 2.4375$

Iteration 3 : $x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 2.4375 - \frac{2.0369}{9.0742} = 2.2130$

Example

Iteration	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_{k+1} - x_k \rvert$
0	4	33	33	3	1
1	3	9	16	2.4375	0.5625
2	2.4375	2.0369	9.0742	2.2130	0.2245
3	2.2130	0.2564	6.8404	2.1756	0.0384
4	2.1756	0.0065	6.4969	2.1746	0.0010

Convergence Analysis

Theorem :

Let $f(x), f'(x)$ *and* $f''(x)$ be continuous at $x \approx r$

where $f(r) = 0$. *If* $f'(r) \neq 0$ then there exist $\delta > 0$

such that $|x_0 - r| \leq \delta \Rightarrow \frac{|x_{k+1} - r|}{|x_k - r|^2} \leq C$

$$C = \frac{1}{2} \frac{\max\limits_{|x_0 - r| \leq \delta} |f''(x)|}{\min\limits_{|x_0 - r| \leq \delta} |f'(x)|}$$

Convergence Analysis

Remarks

When the guess is close enough to a simple root of the function then Newton's method is guaranteed to converge quadratically.

Quadratic convergence means that the number of correct digits is nearly doubled at each iteration.

Problems with Newton's Method

If the initial guess of the root is far from the root the method may not converge.

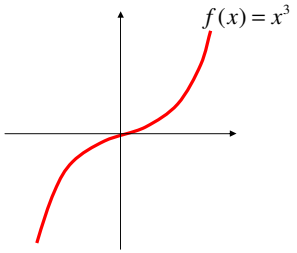
Newton's method converges linearly near multiple zeros

{ f(r) = f '(r) =0 }. In such a case modified algorithms can be used to regain the quadratic convergence.

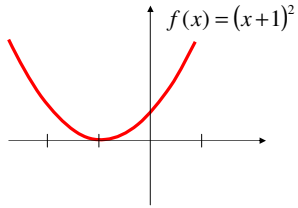
Newton's Method

- Advantages
  - More rapidly convergent than any of the methods.
  - It is quadratically convergent, that is the number of decimal places of accuracy nearly doubles at each iteration
  - Starting point can be arbitrary
- Disadvantage
  - Needs *f'(x)*.

Problems with Newton's Method - Multiple Roots

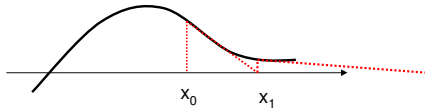


$f(x)$ has has three zeros at x = 0



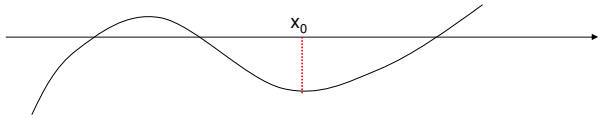
$f(x)$ has has two zeros at x = -1

Problems with Newton's Method - Runaway



The estimates of the root is going away from the root.

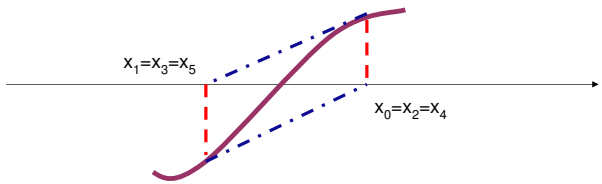
Problems with Newton's Method - Flat Spot



The value of f'(x) is zero, the algorithm fails.

If f '(x) is very small then x_1 will be very far from x_0.

Problems with Newton's Method - Cycle



The algorithm cycles between two values x_0 and x_1

Newton's Method for Systems of nonlinear Equations

$$Given: X_0 \text{ an initial guess of the root of } F(x)=0$$

$$Newton's\ Iteration$$

$$X_{k+1} = X_k - [F'(X_k)]^{-1} F(X_k)$$

$$F(X) = \begin{bmatrix} f_1(x_1, x_2, ...) \\ f_2(x_1, x_2, ...) \\ \vdots \end{bmatrix}, \quad F'(X) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \vdots \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \\ \vdots & & \end{bmatrix}$$

Example

- Solve the following system of equations

$$y + x^2 - 0.5 - x = 0$$
$$x^2 - 5xy - y = 0$$
Initial guess $x = 1,\ y = 0$

$$F = \begin{bmatrix} y + x^2 - 0.5 - x \\ x^2 - 5xy - y \end{bmatrix},\quad F' = \begin{bmatrix} 2x - 1 & 1 \\ 2x - 5y & -5x - 1 \end{bmatrix},\quad X_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Solution Using Newton's Method

Iteration 1 :

$$F = \begin{bmatrix} y + x^2 - 0.5 - x \\ x^2 - 5xy - y \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} =,\quad F' = \begin{bmatrix} 2x - 1 & 1 \\ 2x - 5y & -5x - 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & -6 \end{bmatrix}$$

$$X_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 2 & -6 \end{bmatrix}^{-1} \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.25 \\ 0.25 \end{bmatrix}$$

Iteration 2 :

$$F = \begin{bmatrix} 0.0625 \\ -0.25 \end{bmatrix} =,\quad F' = \begin{bmatrix} 1.5 & 1 \\ 1.25 & -7.25 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} 1.25 \\ 0.25 \end{bmatrix} - \begin{bmatrix} 1.5 & 1 \\ 1.25 & -7.25 \end{bmatrix}^{-1} \begin{bmatrix} 0.0625 \\ -0.25 \end{bmatrix} = \begin{bmatrix} 1.2332 \\ 0.2126 \end{bmatrix}$$

Example

- Solve the following system of equations

$$y + x^2 - 1 - x = 0$$
$$x^2 - 2y^2 - y = 0$$
Initial guess $x = 0,\ y = 0$

$$F = \begin{bmatrix} y + x^2 - 1 - x \\ x^2 - 2y^2 - y \end{bmatrix},\quad F' = \begin{bmatrix} 2x - 1 & 1 \\ 2x & -4y - 1 \end{bmatrix},\quad X_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Example

Iteration	0	1	2	3	4	5
X_k	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.6 \\ 0.2 \end{bmatrix}$	$\begin{bmatrix} -0.5287 \\ 0.1969 \end{bmatrix}$	$\begin{bmatrix} -0.5257 \\ 0.1980 \end{bmatrix}$	$\begin{bmatrix} -0.5257 \\ 0.1980 \end{bmatrix}$

Secant Method

- Secant Method
- Examples
- Convergence Analysis

Newton's Method (Review)

Assumptions: $f(x),\ f'(x),\ x_0$ *are available*, $f'(x_0) \neq 0$

Newton's Method new estimate

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

Problem:

$f'(x_i)$ is not available

or difficult to obtain analytically

Secant Method

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

if x_i *and* x_{i-1} are two initial points

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

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

Secant Method

Assumption s:

Two initial points x_i *and* x_{i-1} *such that* $f(x_i) \neq f(x_{i-1})$

New estimate (Secant Method):

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

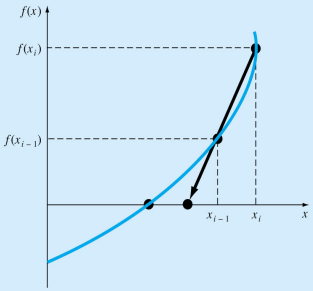
Secant Method

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

- Requires two initial estimates x_o, x_1.

  However, it is not a "bracketing" method.
- *The Secant Method* has the same properties as *Newton*'s method.

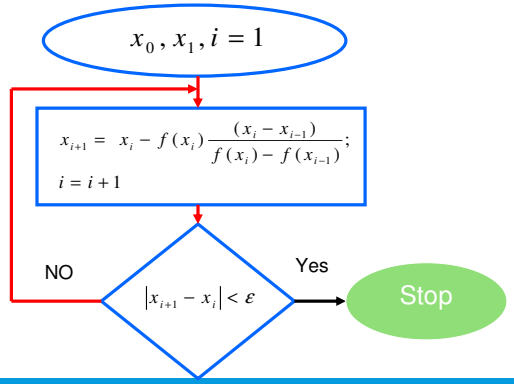
  Convergence is not guaranteed for all x_o, f(x).



Secant Method

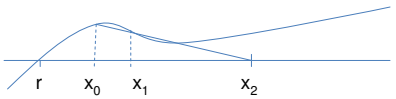
$$f(x) = x^2 - 2x + 0.5$$
$$x_0 = 0$$
$$x_1 = 1$$
$$x_{i+1} = x_i - f(x_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Secant Method



Secant Method

- Advantage:
  - Convergence is fast and
  - It does dot need derivative.
- Disadvantage:
  - The method may fail if the function is far from linear and near the root the successive iterates can fly off to points far from the root.



Modified Secant Method

In this modified Secant method only one initial guess is needed

$$f'(x_i) \approx \frac{f(x_i+\delta_i)-f(x_i)}{\delta_i}$$

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

Problem: How to select δ_i ?

If not selected properly, the method may diverge

Example

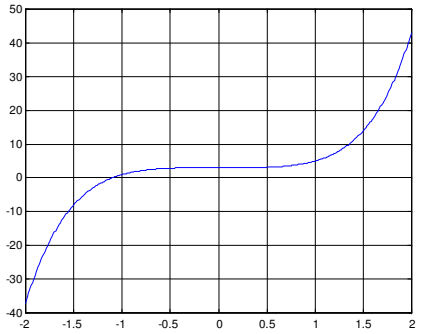
find the roots of

$f(x) = x^5 + x^3 + 3$

initial points

$x_0 = -1$ *and* $x_1 = -1.1$

with error < 0.001



Example

x(i)	f(x(i))	x(i+1)	|x(i+1)-x(i)|
-1.0000	1.0000		
-1.1000	0.0585	-1.1062	0. 0062
-1.1062	0.0102	-1.1052	0.0009
-1.1052	0.0001	-1.1052	0.0000

Convergence Analysis

- The rate of convergence of the Secant method is super linear

$$\frac{|x_{i+1} - r|}{|x_i - r|^{\alpha}} \le C, \qquad \alpha \approx 1.62$$

r : *root* x_i : estimate of the root at the i^{th} iteration

- It is better than Bisection method but not as good as Newton's method

Comparison of Root finding methods

- Advantages/disadvantages
- Examples

Summary

Bisection	Reliable, Slow One function evaluation per iteration Needs an interval [a,b] containing the root, f(a) f(b)<0 No knowledge of derivative is needed
Newton	Fast (if near the root) but may diverge Two function evaluation per iteration Needs derivative and an initial guess x_0, $f'(x_0)$ is nonzero
Secant	Fast (slower than Newton) but may diverge one function evaluation per iteration Needs two initial points guess x_0, x_1 such that $f(x_0)$- $f(x_1)$ is nonzero. No knowledge of derivative is needed

Example

Use Secant method to find the root of $f(x) = x^6 - x - 1$

Two initial points $x_0 = 1$ *and* $x_1 = 1.5$

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

Solution

k	x_k	$f(x_k)$
0	1.0000	-1.0000
1	1.5000	8.8906
2	1.0506	-0.7062
3	1.0836	-0.4645
4	1.1472	0.1321
5	1.1331	-0.0165
6	1.1347	-0.0005

Example

Use Newton's Method to find a root of

$f(x) = x^3 - x - 1$

Use the initial points $x_0 = 1$

Stop after three iterations or

if $|x_{k+1} - x_k| < 0.001$ or

if $|f(x_k)| < 0.0001$

Five iterations of the solution

k	x_k	$f(x_k)$	$f'(x_k)$	ERROR
0	1.0000	-1.0000	2.0000	
1	1.5000	0.8750	5.7500	0.1522
2	1.3478	0.1007	4.4499	0.0226
3	1.3252	0.0021	4.2685	0.0005
4	1.3247	0.0000	4.2646	0.0000
5	1.3247	0.0000	4.2646	0.0000

Example

Use Newton's Method to find a root of

$f(x) = e^{-x} - x$

Use the initial points $x_0 = 1$

Stop after three iterations or

if $|x_{k+1} - x_k| < 0.001$ or

if $|f(x_k)| < 0.0001$

Example

Use Newton's Method to find a root of

$f(x) = e^{-x} - x, \qquad f'(x) = -e^{-x} - 1$

x_k	$f(x_k)$	$f'(x_k)$	$\frac{f(x_k)}{f'(x_k)}$
1.0000	- 0.6321	- 1.3679	0.4621
0.5379	0.0461	- 1.5840	- 0.0291
0.5670	0.0002	- 1.5672	- 0.0002
0.5671	0.0000	- 1.5671	- 0.0000

Example

Estimates of the root of x-cos(x)=0

0.60000000000000	initial guess
0.74401731944598	1 correct digit
0.73909047688624	4 correct digits
0.73908513332147	10 correct digits
0.73908513321516	14 correct digits

Example

In estimating the root of x-cos(x)=0

To get more than 13 correct digits

- 4 iterations of Newton (x_0=0.6)
- 43 iterations of Bisection method (initial interval [0.6, .8]
- 5 iterations of Secant method (x_0=0.6, x_1=0.8)

Computer based solution method

- NEQNF subroutine (Fortran IMSL subroutine that Solves a system of nonlinear equations using a modified Powell hybrid algorithm and a finite-difference approximation to the Jacobian)

Solve a system of nonlinear equations using a modified Powell hybrid algorithm and a finite-difference approximation to the Jacobian.

Usage

CALL NEQNF (FCN, ERRREL, N, ITMAX, XGUESS, X, FNORM)

Arguments

FCN — User-supplied SUBROUTINE to evaluate the system of equations to be solved. The usage is CALL FCN (X, F, N), where

X — The point at which the functions are evaluated. (Input)
X should not be changed by FCN.

F — The computed function values at the point X. (Output)

N — Length of X and F. (Input)

FCN must be declared EXTERNAL in the calling program.

ERRREL — Stopping criterion. (Input)
The root is accepted if the relative error between two successive approximations to this root is less than ERRREL.

Computer based solution method

- NEQNF subroutine (Fortran IMSL subroutine that Solves a system of nonlinear equations using a modified Powell hybrid algorithm and a finite-difference approximation to the Jacobian)

Solve a system of nonlinear equations using a modified Powell hybrid algorithm and a finite-difference approximation to the Jacobian.

Usage

CALL NEQNF (FCN, ERRREL, N, ITMAX, XGUESS, X, FNORM)

N — The number of equations to be solved and the number of unknowns. (Input)

ITMAX — The maximum allowable number of iterations. (Input)
The maximum number of calls to FCN is ITMAX * (N + 1). Suggested value ITMAX = 200.

XGUESS — A vector of length N. (Input)
XGUESS contains the initial estimate of the root.

X — A vector of length N. (Output)
X contains the best estimate of the root found by NEQNF.

FNORM — A scalar that has the value F(1)2 + . . . + F(N)2 at the point X. (Output)

Computer based solution method - NEQNF IMSL - Example

The following 3 × 3 system of nonlinear equations

$$f_1(x) = x_1 + e^{x_2 x_3} + (x_2 + x_3)^2 - 27 = 0$$

$$f_2(x) = e^{x_2 x_3} / x_1 + x_3^2 - 10 = 0$$

$$f_3(x) = x_3 + \sin(x_2 - 2) + x_2^2 - 7 = 0$$

is solved with the initial guess (4.0, 4.0, 4.0).

```
C                                  Declare variables
      INTEGER    ITMAX, N
      REAL       ERRREL
      PARAMETER  (N=3)
C
      INTEGER    K, NOUT
      REAL       FNORM, X(N), XGUESS(N)
      EXTERNAL   FCN, NEQNF, UMACH
C                                  Set values of initial guess
C                                  XGUESS = (  4.0  4.0  4.0 )
C
      DATA XGUESS/4.0, 4.0, 4.0/
C
      ERRREL = 0.0001
      ITMAX  = 100
```


Computer based solution method - NEQNF IMSL - Example

```
C
      CALL UMACH (2, NOUT)
C                                  Find the solution
      CALL NEQNF (FCN, ERRREL, N, ITMAX, XGUESS, X, FNORM)
C                                  Output
      WRITE (NOUT,99999) (X(K),K=1,N), FNORM
99999 FORMAT ('  The solution to the system is', /, '  X = (', 3F5.1,
     &        ')', /, '  with FNORM =', F5.4, //)
C
      END
C                                  User-defined subroutine
      SUBROUTINE FCN (X, F, N)
      INTEGER    N
      REAL       X(N). F(N)
C
      REAL       EXP, SIN
      INTRINSIC  EXP, SIN
C
      F(1) = X(1) + EXP(X(1)-1.0) + (X(2)+X(3))*(X(2)+X(3)) - 27.0
      F(2) = EXP(X(2)-2.0)/X(1) + X(3)*X(3) - 10.0
      F(3) = X(3) + SIN(X(2)-2.0) + X(2)*X(2) - 7.0
      RETURN
      END
```

Output

```
The solution to the system is
X = (  1.0  2.0  3.0)
with FNORM =.0000
```