Bil 108

Introduction to the Scientific and Engineering Computing with MATLAB

Lecture 6

F. Aylin Konuklar   Lale T. Ergene

Overview

- Preliminary considerations and bracketing
- Fixed Point Iteration
- Bisection
- Newton’s Method
- The Secant Method
- Hybrid Methods: the built in **fzero** function

Root finding

- Nonlinear equations can be written as

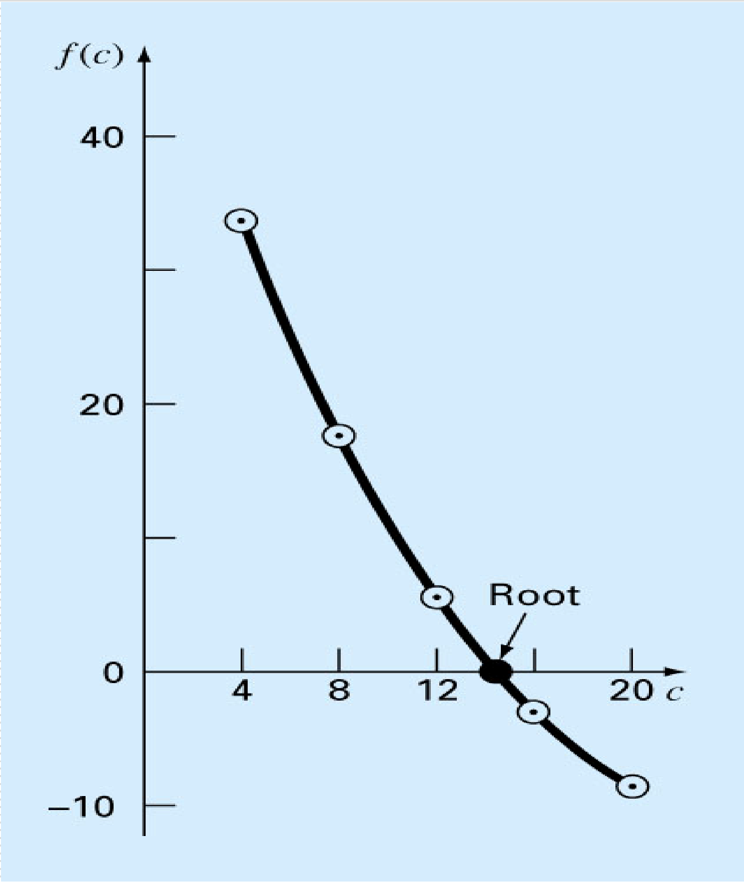
$$f(x) = 0$$

- For example, if $xe^x = 0$

Then we call $f(x) = xe^x$

Where $f(x) = 0$

Graphically



Root finding(continued)

- Finding the roots of a nonlinear equation is equivalent to finding the values of x for which $f(x)$ is zero

- We examine several methods of finding the roots for a general function $f(x)$

Successive Substitution

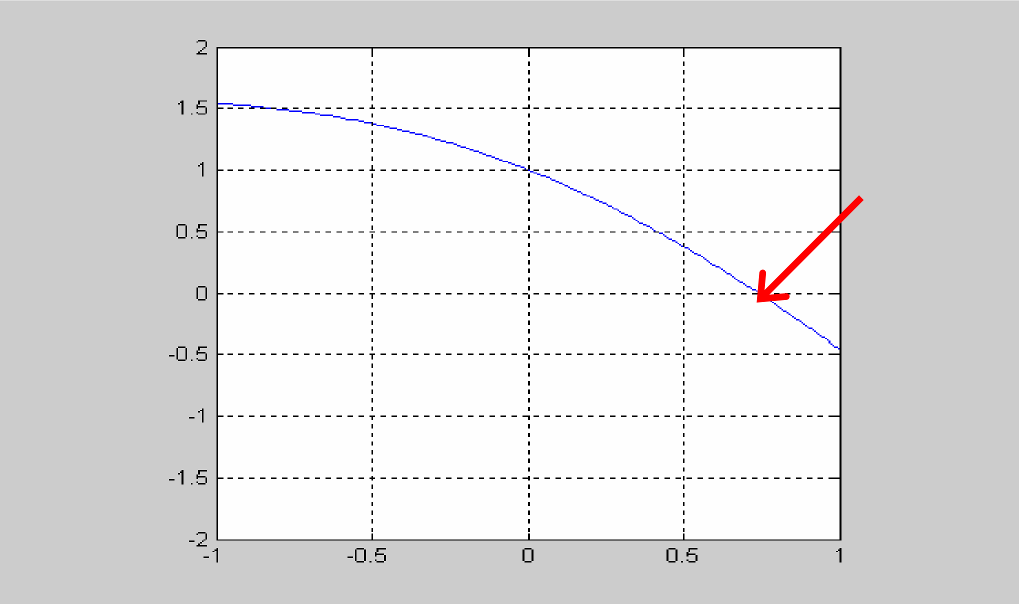
- A fundamental principle in computer science is *iteration*. As the name suggests, a process is repeated until an answer is achieved. Iterative techniques are used to find roots of equations, solutions of linear and nonlinear systems of equations, and solutions of differential equations.

- A rule or function for computing successive terms is needed, together with a starting value . Then a sequence of values is obtained using the iterative rule $p_{k+1}=g(p_k)$

Roots of f(x) = 0

Any function of one variable can be put in the form f(x) = 0.
Example:
To find the x that satisfies
cos(x) = x
Find the zero crossing of
f(x) = cos(x) − x = 0

```
%script file
x=linspace(-1,1);
y=cos(x)-x;
plot(x,y);
axis([min(x),max(x) -2 2]);
grid;
```



General Considerations

- Is this a special function that will be evaluated often?
- How much precision is needed?
- How fast and robust must the method be?

- **There is no single root-finding method that is best for all situations.**

The basic strategy for root-finding procedure

1. ***Plot the function.***

   The plot provides an initial guess, and an indication of potential problems.

2. ***Select an initial guess.***

3. ***Iteratively refine the initial guess with a root finding algorithm.***

   If x_k is the estimate to the root on the k^{th} iteration, then the iterations converge

Root-Finding Methods

- ***BRACKETING METHODS***
  1. **Bisection (Interval Halving)**
  2. **False Position (Regula Falsi)**

These methods are applied *after* initial guesses at the root(s) are identified with bracketing (or guesswork).

- ***OPEN METHODS***
  1. **Fixed point iteration**
  2. **Newton's method (Newton-Raphson)**
  3. **Secant method**

These methods can involve one or more initial guesses, but there is no need for them to bracket the root.

Bisection Method (Interval Halving)

1. find an interval in which the root is known to occur (such an interval is said to bracket the root) because the function has opposite signs at the ends of the interval

2. divide the interval into two equal subintervals

3. determine which subinterval brackets the root and continue with this subinterval

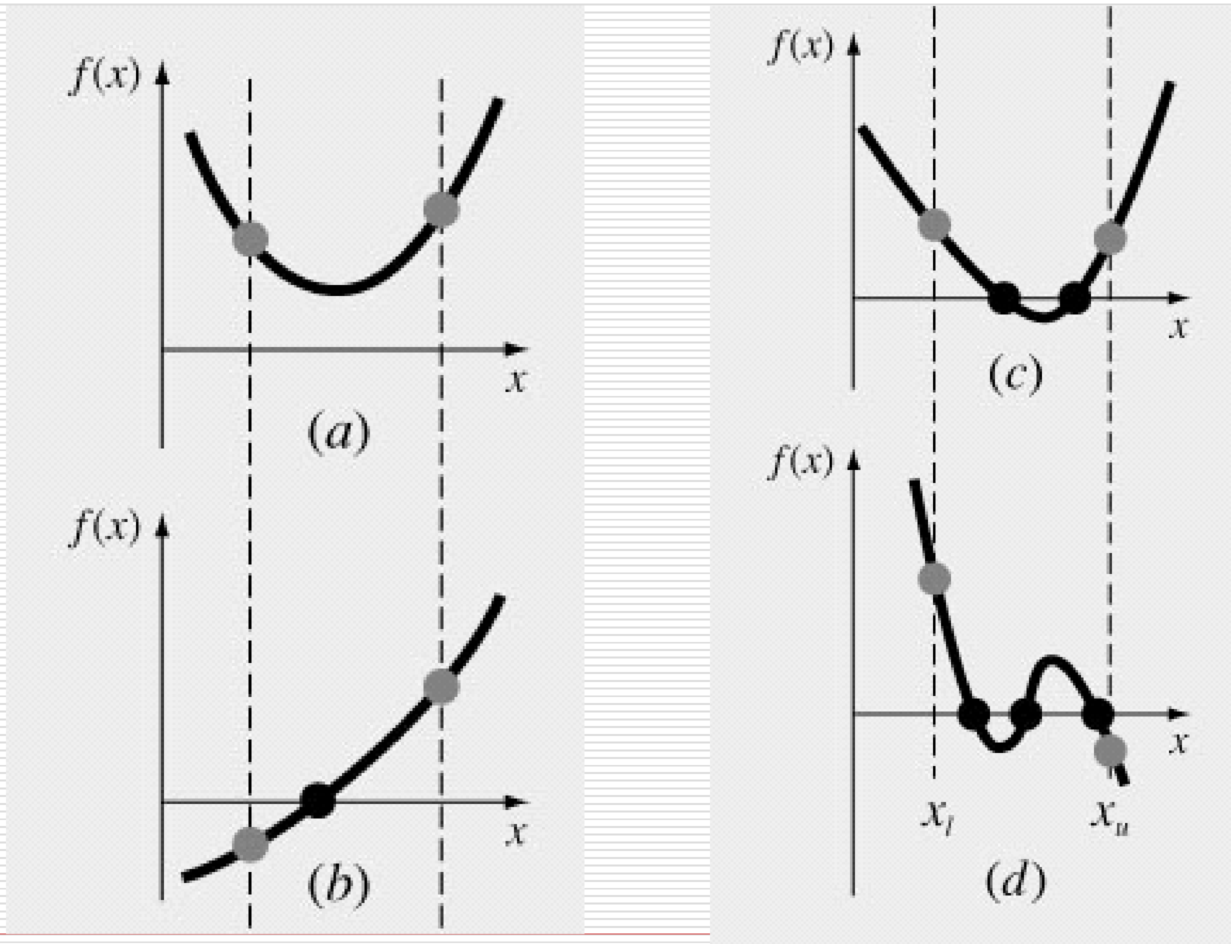
Bisection (interval halving) method

- Simplest method
- Most robust method
- need two initial guesses x_l and x_u which bracket the root
- At least one root exists between x_l and x_u

if

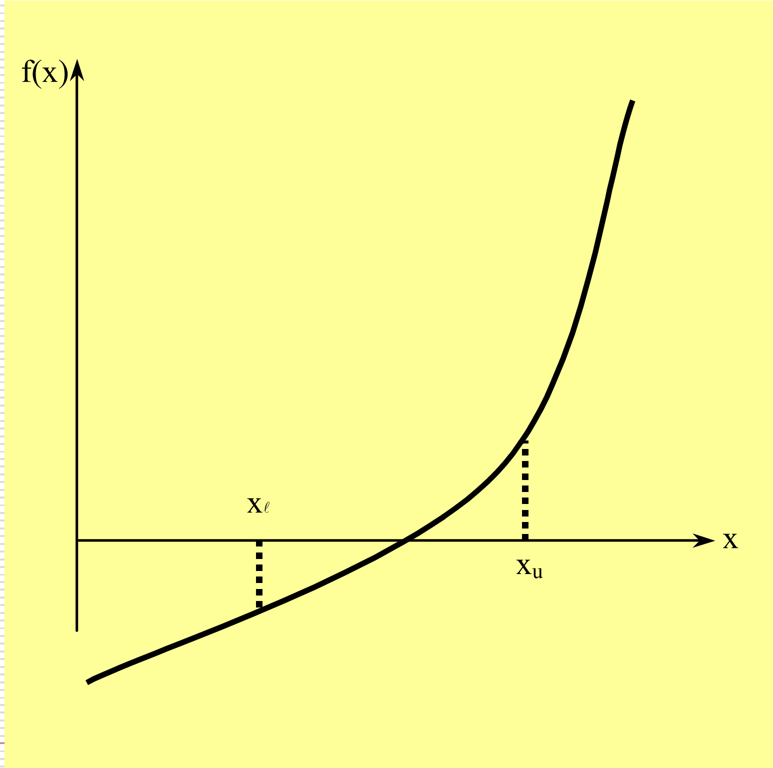
$$f(x_l)f(x_u) < 0$$

Illustration of $f(x_l)f(x_u) < 0$



Step 1

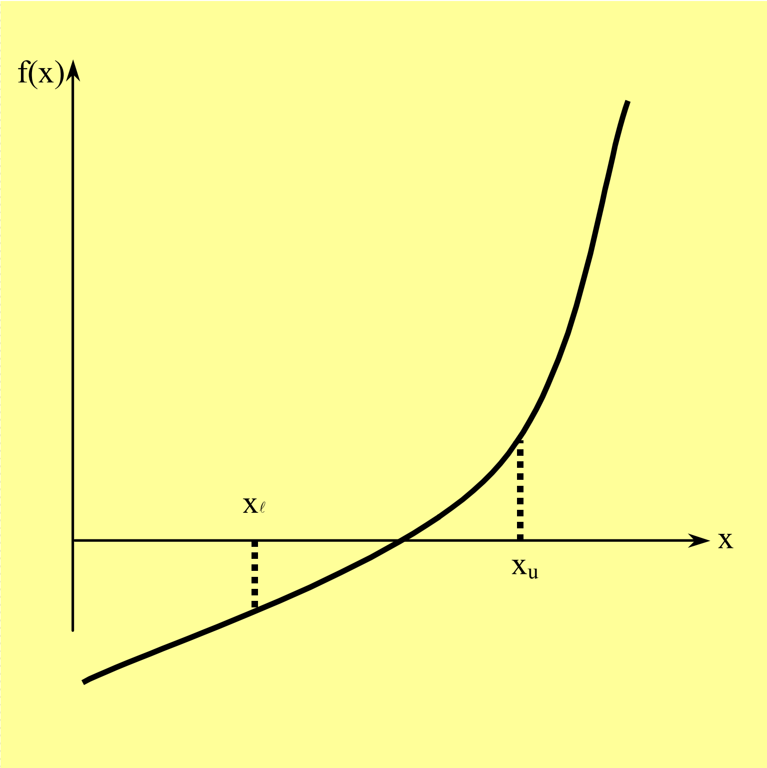
- Choose x_λ and x_u as two guesses for the root such that $f(x_\lambda)f(x_u) < 0$, or in other words, $f(x)$ changes sign between x_λ and x_u.



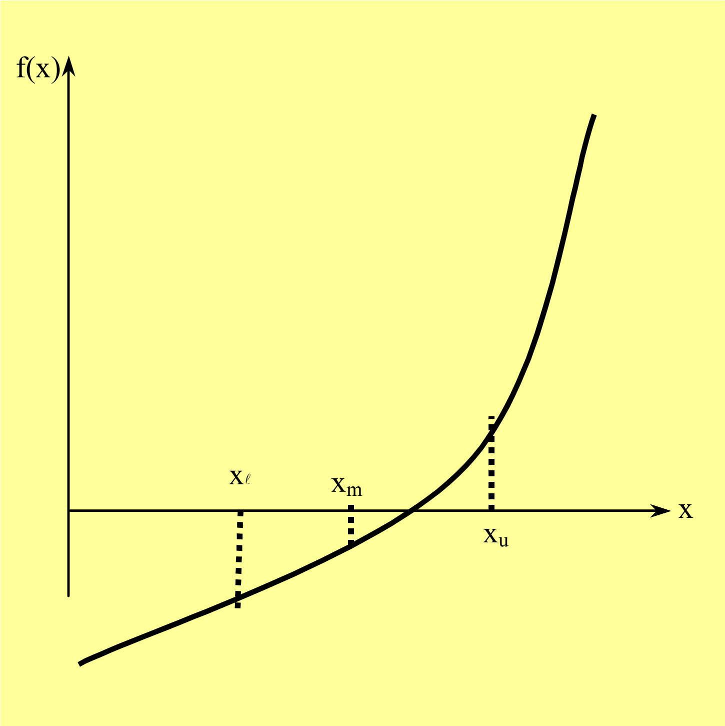
Step 2

Estimate the root, x_m of the equation f (x) = 0 as the mid-point between x_λ and x_u as

$$x_m = \frac{x_\ell + x_u}{2}$$



Step 3



Now check the following

If $f(x_\lambda)\ f(x_m) < 0$,
the root lies between x_λ and x_m;
then $x_\lambda = x_\lambda$; $x_u = x_m$.

If $f(x_\lambda\ )\ f(x_m) > 0$,
then the root lies between x_m and x_u;
then $x_\lambda = x_m$; $x_u = x_u$.

If $f(x_\lambda)\ f(x_m) = 0$;
then the root is x_m.
Stop the algorithm if this is true.

Step 4

New estimate $$x_m = \frac{x_\ell + x_u}{2}$$

Absolute Relative Approximate Error

$$\left|\in_a\right| = \left|\frac{x_m^{new} - x_m^{old}}{x_m^{new}}\right| \times 100$$

x_m^{old} = previous estimate of root

x_m^{new} = current estimate of root

Step 5

Check if absolute relative approximate error is less than prespecified tolerance or if maximum number of iterations is reached.

Yes → Stop

No → Using the new upper and lower guesses from Step 3, go to Step 2.

Bisection Algorithm

initialize: $a = \ldots, b = \ldots$

for $k = 1, 2, \ldots$

$\quad x_m = a + (b - a)/2$

$\quad$ if $\text{sign}(f(x_m)) = \text{sign}(f(x_a))$

$\quad\quad a = x_m$

$\quad$ else

$\quad\quad b = x_m$

$\quad$ end

$\quad$ if converged, stop

end

The statement eval(f) is used to evaluate the function at a given value of *x*.

EX: Apply bisection to $x - x^{1/3} - 2 = 0$

Advantages

- Always convergent
- The root bracket gets halved with each iteration - guaranteed

Drawbacks

- Slow convergence
- If one of the initial guesses is close to the root, the convergence is slower

Fixed Point Iteration

- **Fixed point iteration is a simple method.**
- To solve $f(x) = 0$
- rewrite as

$$x_{new} = g(x_{old})$$

- Algorithm Fixed Point Iteration

initialize: $x_0 = \dots$
for $k = 1, 2, \dots$
$x_k = g(x_k{-}1)$
if converged, stop
end

Example

- $f(x)=x^2-2x-3=0$ $x=-1$ and $x=3$ (imagine that we do not know the roots)

  ITERATION FNC:
  - If we start with x=4 and iterate with fixed-point algorithm
  - It appears that values are converging on the root at x=3
  - Check the other two arrangements!!!!

$$x = g_1(x) = \sqrt{2x+3}$$

$$x_0 = 4$$

$$x_1 = \sqrt{11} = 3.31662,$$

$$x_2 = \sqrt{9.63325} = 3.10375,$$

$$x_3 = \sqrt{9.20750} = 3.03439,$$

$$x_4 = \sqrt{9.06877} = 3.01144,$$

$$x5 = \sqrt{9.02288} = 3.00381$$

Fixed-point Algorithm

To determine a root of f(x)=0, given a value x_1 reasonable close to the root

Reaarange the equation to an equivalent form x=g(x).

for *k* = 1, 2, *. . .*

xk = g(xk−1)

if converged,

$$\left| x_1 - x_2 \right| < tolerance\_value$$

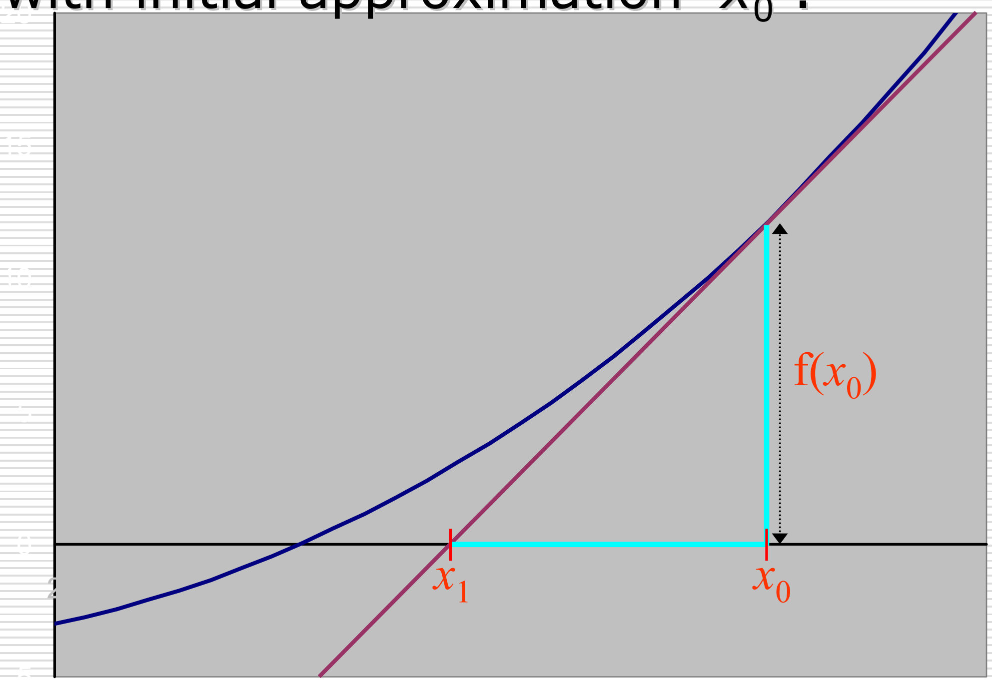
stop

Newton's (Newton-Raphson) Method

The Newton Raphson method is based on the iteration:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, 3, \ldots$$

with initial approximation x_0 .



Gradient of tangent

$$f'(x_0) = \frac{f(x_0)}{x_0 - x_1}$$

$$\Rightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Newton Raphson Method

- Step 1: Start at the point $(x_0, f(x_0))$.

- Step 2 : The intersection of the tangent of $f(x)$ at this point and the x-axis.

  $$x_1 = x_0 - f(x_0)/f'(x_0)$$

- Step 3: Examine if $f(x_1) = 0$

  or $abs(x_1 - x_0) < tolerance$,

- Step 4: If yes, solution $x_r = x_1$

  If not, $x_0 \leftarrow x_1$, repeat the iteration.

Algorithm for Newton-Raphson Method

Step 1

Evaluate $f'(x)$ symbolically

Step 2

Calculate the next estimate of the root

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

Find the absolute relative approximate error

$$|\in_a| = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \text{ x } 100$$

Algorithm for Newton-Raphson Method

Step 3

- Find if the absolute relative approximate error is greater than the pre-specified relative error tolerance.
- If so, go back to step 2, else stop the algorithm.
- Also check if the number of iterations has exceeded the maximum number of iterations.

Example

- Use the Newton Raphson method to determine the mass of the bungee jumper with a drag coefficient of 0.25kg/m to have a velocity of 36m/s after 4s of free fall The acceleratıon of gravity is 9.81m/s^2

- Soln. The function to be evaluated and its derivative is shown below

$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t)$$

$$\frac{df(m)}{dm} = \frac{1}{2}\sqrt{\frac{g}{mc_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - \frac{g}{2m} t \sec h^2\left(\sqrt{\frac{gc_d}{m}} t\right)$$

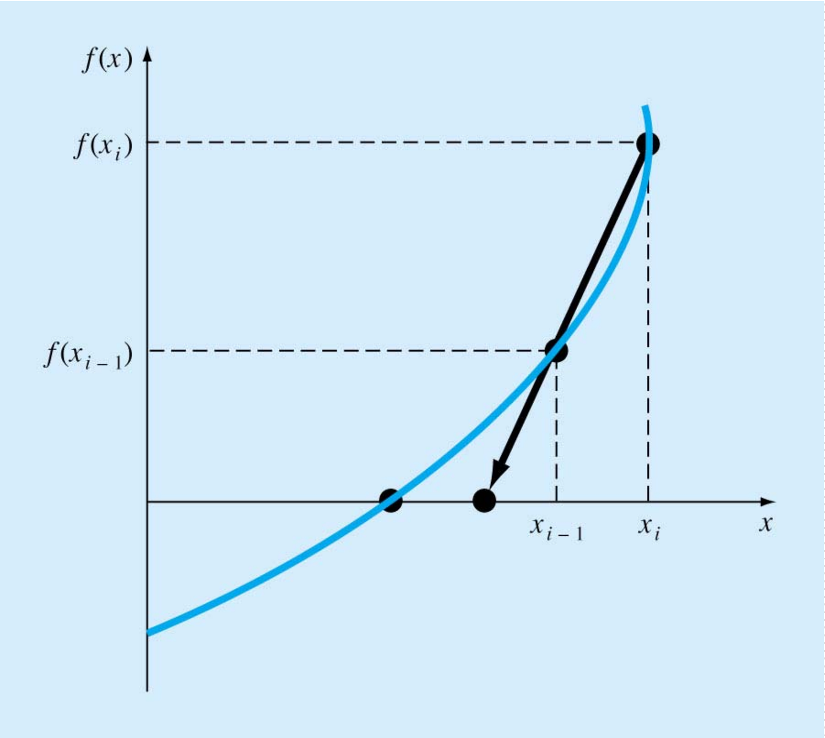
Example

- **Example :** *Find an approximation of the solution of the equation* $x^3-2x^2 - 5 = 0$ *for* x *in* [0,5] *accurate to within* 10^{-5}.

- newton1('fx2n',2,0.00001)

Secant Method

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



Secant Method

Step 1: Start at two points (x1, f(x1)), (x2,f(x2)), where x1 & x2 are close

Step 2 : Find the intersection of the line connecting the two points and the x-axis. (The line approximates the tangent)

$$x3 = x2 - f(x2)*(x2 - x1)/(f(x2) - f(x1))$$

Step 3: Examine if f(x3) = 0

Step 4: If yes, solution xr = x3,
else x1=x2 & x2=x3, repeat the iteration.

```
>>secant1('fx2n',[2,4],0.0001)
```

Summary

- Plot $f(x)$ before searching for roots
- Bracketing finds coarse interval containing roots and singularities
- Bisection is robust, but converges slowly
- Newton's Method

Requires $f(x)$ and $f'(x)$.

Iterates are not confined to initial bracket.

Converges rapidly.

- Secant Method

Uses $f(x)$ values to approximate $f(x)$.

Iterates are not confined to initial bracket.

Converges almost as rapidly as Newton's method.

fminbnd

x = fminbnd('function',x1,x2)

x = fminbnd('function',x1,x2) returns a value of x which is a local minimizer of function(x) in the interval x1 < x < x2. function is a string containing the name of the objective function to be minimized.

```
function y = f(x)
y = x.^3-2*x-5;
```

Then invoke fmin with

```
x = fminbnd(' x.^3-2*x-5 ', 0, 2)
```

The result is

```
x =
   0.8165
```

The value of the function at the minimum is

```
y = f(x)
y =
   -6.0887
```

fzero Function

fzero is a hybrid method that combines bisection, secant and reverse quadratic interpolation

- **Syntax:**

  r = fzero(*'fun'*,*x0*)

Points where the function touches the x-axis, but does not cross it, are not considered zeros.

fzero Function

```
function y = f(x)
= x.^3-2*x-5;
To find the zero near 2
z = fzero('f',2)
z = 2.0946
```

- Requires a starting point.
- The function f must be continuous.
- Fast convergence!

Examples

Problem: Find a root of the function f(x) = sin(x) near x= .5 .

>> fzero('sin', .5)

Zero found in the interval: [-0.28, 1.9051].

ans =

1.5485e-24

Notice that sin(0) = 0 but fzero returned a value close to but not exactly equal to zero .

Problem: Find a root of the function $f(x) = x^2 - e^x$ near x= 0 .

>> fzero('x.^2-exp(x)', 0)

ans =

-0.7035

fzero Function

One form of the fzero function is:

- **fzero('function_name', x0)**

Where **function_name** is either the name of a built-in Matlab function or the name of a user defined function.

- **x0** is an initial guess for the root. If you have no idea what **x0** should be then try using the **fplot** function to plot the function.

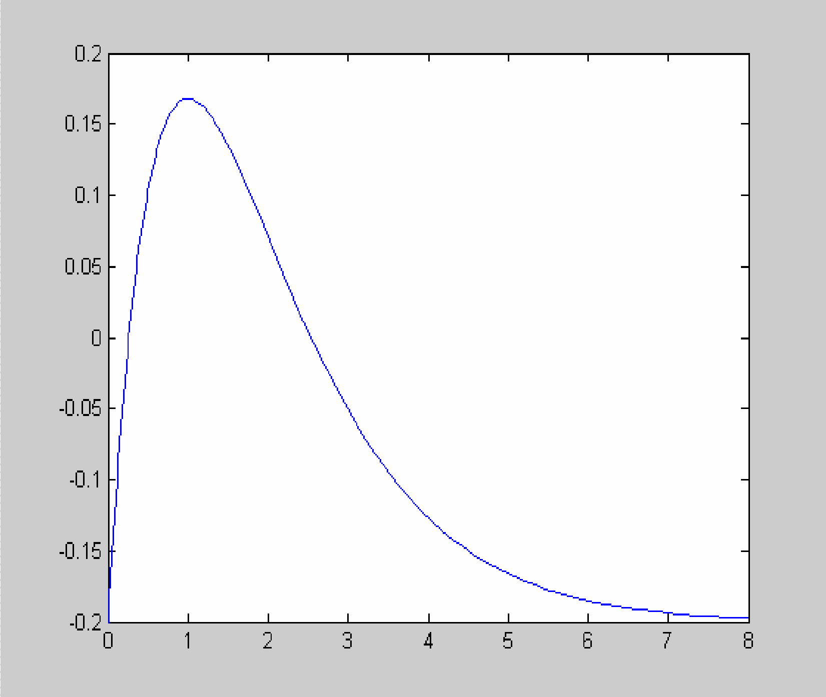
Example

Determine the solution of the equation $xe^{-x}=0.2$

$f(x)= xe^{-x}-0.2$

- Then plot the function

```
>>fplot('x*exp(-x)-0.2', [0 8])
```




```
>> x1=fzero('x*exp(-x)-0.2',0.7)

x1 =

  0.2592
>> x2=fzero('x*exp(-x)-0.2',2.8)

x2 =

  2.5426
```