Lesson 07

Suppose we do not have the commands "Solve" or "FindRoot" and we want to find roots, what can we do?
There are several methods to do so...

http://numericalmethods.eng.usf.edu/

http://www.damtp.cam.ac.uk/lab/people/sd/lectures/nummeth98/roots.htm

http://home.scarlet.be/~ping1339/root.htm

http://www.efunda.com/math/num_rootfinding/num_rootfinding.cfm

http://en.wikipedia.org/wiki/Root-finding_algorithm
http://en.wikipedia.org/wiki/Newton's_method

Bisection method

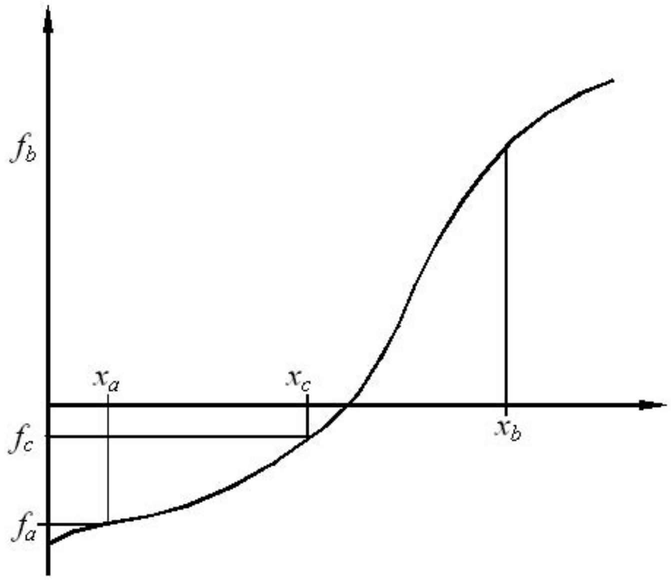
This is one of the best, most effective methods for finding the REAL zeros of a CONTINUOUS function. The bisection method begins with an interval function (x1,x2) in which the function changes sign, so that
f(x1) f(x2)<0

We then pick the midpoint x3 (bisect the interval) and evaluate the function there.

If f(x1) f(x3)<0, then there is a sign change in (x1,x3) and we repeat the procedure for this interval.
If f(x1) f(x3)>0, then there is a sign change in (x3,x2) and we repeat the procedure for this interval.
If f(x1)f(x3)=0, then x3 is the root.

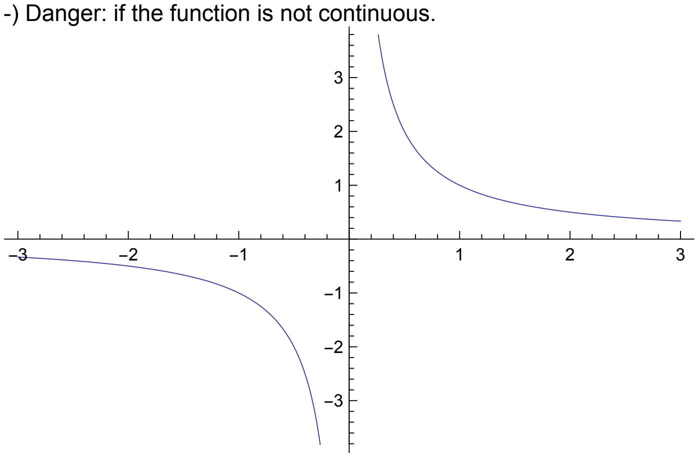
This procedure is repeated several times until we get a satisfactory approximation to the actual zero.

```
Import["Fig_Bisection.jpg"]
```



esentation of the bisection method showing two initial guesses (x_a and x_b bracketting the root).

-) In practice it rarely uses more than 20 steps.
-) Danger: if the function is not continuous.



-) We need to search for an interval where there is a sign change. If we use a small step, we may spend a long time looking for the interval. If we use a large step, we run the risk of steppun over more than one zero.

Example:
Using the bisection method, find the root of f(x) = x^2 -2. For the search of the interval, start at x=0 and try steps = 1/2.

```
Clear[f];
f[x_]:= x^2 -2;

Print[“The actual root is “, Sqrt[2.] ];

Print[ ];
Table[{x/2,f[x/2]}, {x, 0, 5}]
```

The interval is (1,3/2)

```
Clear[x1,x2];
x1=1.;
x2=3/2.;
Do[
mid = (x1 + x2)/2.;
Print["Iteration ", k];
Print["Approximation to the root = ", mid];
If[f[x1] f[mid]  ==  0, Goto[end] ];
If[f[x1] f[mid] <  0, {x1 = x1, x2=mid}, {x1 = mid, x2=x2}];
,{k,1,10}]
Label[end];
If[f[x1] f[mid]  ==  0, Print["The root is ", mid]];
```

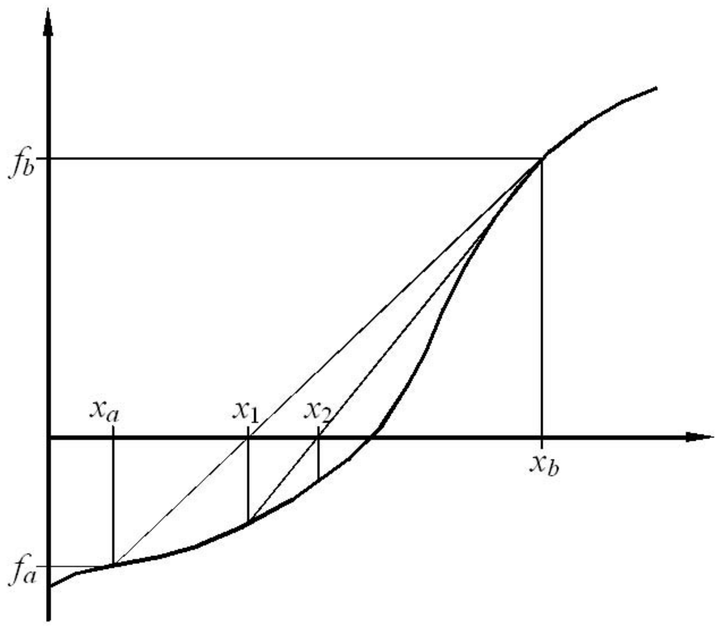
Method of false position (regula falsi)

This method attempts to do better than the slow bisection method. The idea is the following.
-) Start again with two points x1 and x2 at which the function has opposite signs [that is, we have an interval which contains the zero].
-) Instead of the midpoint, select as x3 the zero of the straight line connecting the points (x1, f(x1)) and (x2,f(x2)).

If   f(x1) f(x3)<0, then there is a sign change in (x1,x3) and we repeat the procedure for this interval.
If   f(x1) f(x3)>0, then there is a sign change in (x3,x2) and we repeat the procedure for this interval.
If   f(x1)f(x3)=0, then x3 is the root.

This procedure is repeated several times until we get a satisfactory approximation to the actual zero.

```
Import["Fig_RegulaFalsi.jpg"]
```



Root finding by the linear interpolation (regula falsi) method. The two initial gueses x_a and x_b must bracket the root.

Newton-Raphson’s method

http://www.youtube.com/watch?v=OR9DgzkB4Ag

The idea of this method is the following.

(i) Guess an initial point x_k and trace the tangent line to the function f(x) passing through this point:

y(x) = f(x_k) +f’(x_k) (x-x_k)

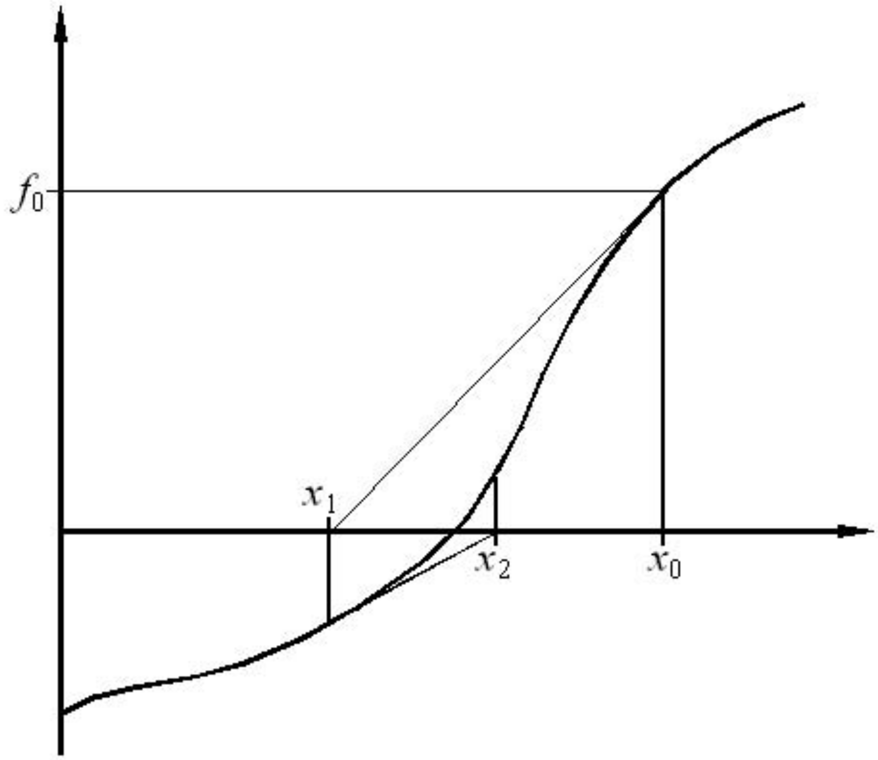
[or, you can think equivalently, that you considered the Taylor expression of the function at some point x_k and neglected quadratic and higher order terms.]

(ii) Use the zero of this tangent as the next approximation to the zero of the function, that is

$$0 = f(x_k) + f'(x_k)\,(x_{k+1} - x_k) \Longrightarrow x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

(iii) Repeat this procedure until a sufficiently good approximation to the zero of the function is found.

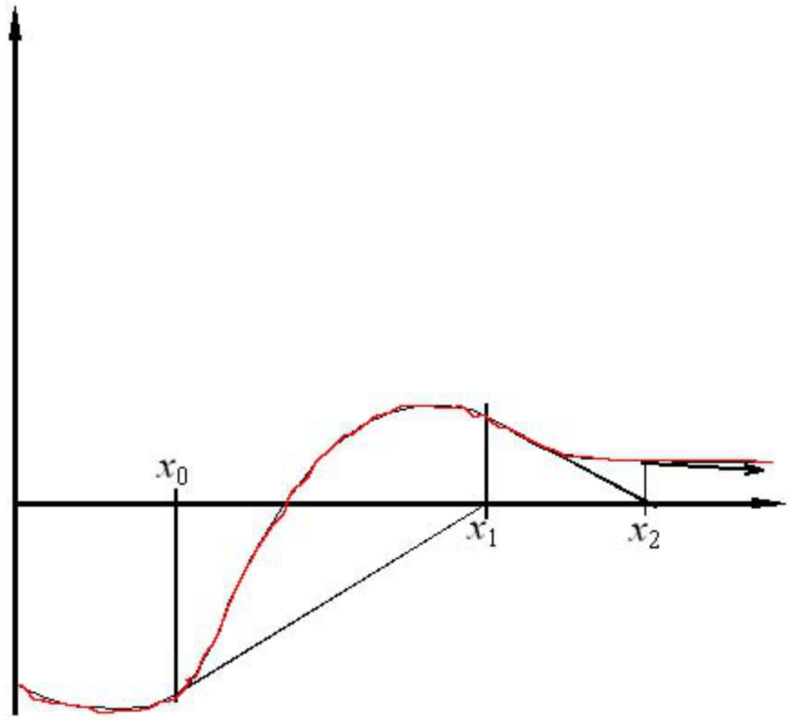
```
Import["Fig_Newton.jpg"]
```



interpretation of the Newton Raphson algorithm.

Advantage: If it converges, it does so quite rapidly.
But, it may not converge depending on the function.
The derivative may also be zero.

```
Import["Fig_Newton2.jpg"]
```



Divergence of the Newton Raphson algorithm due to the presence of a turning point close to the root.

Apply Newton's method, using the starting value given:
*) y = xe^x -1   for x=0.7

```
FindRoot[x Exp[x] -1 == 0, {x, 0.7}]

Clear[xint, f];
xint = 0.7;
f[x_] := x Exp[x] - 1
df = D[f[x], x];

Do[
 xint = xint - f[xint]/(df /. x → xint);
 Print[xint];
 , {k, 1, 4}]
```

Secant method

http://www.youtube.com/watch?v=jXIUi7zxWIU
http://www.youtube.com/watch?v=qC9xnsfOd30

This method is essentially the same as Newton-Raphson's method, but

$f'(x_k)$ becomes instead $\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$

so the interation is based on

$$x_{k+1} = x_k - f(x_k)\,\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

TWO initial guesses are required, x_k and x_(k-1)

Newton-Raphson’s method to find two roots

There are no good, general methods for solving systems of more than one nonlinear equation.

Suppose the case of two dimensions, where we want to solve

$$f(x,\ y) = 0$$
$$g(x,\ y) = 0$$

An idea that some times work is to use the basis of Newton’s method.

```
Import["Newton2D.jpg"]
```

Newton-Raphson Method for Nonlinear Systems of Equations

$f(x, y) = 0$

$g(x, y) = 0$

Numerical Methods
T. Veerarajan
T. Ramachandran
(Tata Mc Graw-Hill)

→ initial guess (x_0, y_0)

if $(x_0+h,\ y_0+K)$ was the solution then

$$\begin{cases} f(x_0+h,\ y_0+K) = 0 \xrightarrow{\text{Taylor}} \underbrace{f(x_0,y_0)}_{f_0} + h \overbrace{\left.\frac{\partial f}{\partial x}\right)_{x_0,y_0}}^{f_x} + K \overbrace{\left.\frac{\partial f}{\partial y}\right)_{x_0,y_0}}^{f_y} + \dots = 0 \\ g(x_0+h,\ y_0+K) = 0 \xrightarrow{\text{Taylor}} \underbrace{g(x_0,y_0)}_{g_0} + h \underbrace{\left.\frac{\partial g}{\partial x}\right)_{x_0,y_0}}_{g_x} + K \underbrace{\left.\frac{\partial g}{\partial y}\right)_{x_0,y_0}}_{g_y} + \dots = 0 \end{cases}$$

→ To find h and K

$$\begin{cases} h = -\dfrac{D_x}{D} \\ K = -\dfrac{D_y}{D} \end{cases}$$

$$D = \begin{vmatrix} f_x & f_y \\ g_x & g_y \end{vmatrix} \qquad D_x = \begin{vmatrix} f_0 & f_y \\ g_0 & g_y \end{vmatrix} \qquad D_y = \begin{vmatrix} f_x & f_0 \\ g_x & g_0 \end{vmatrix}$$

→ get new $x_1 = x_0 + h$, $y_1 = y_0 + K$
and restart the procedure
and repeat it until the solution is obtained with a desired accuracy

Check $|f(x_1, y_1)|$, $|g(x_1, y_1)|$ if both $< \epsilon$ stop

$\uparrow$ desired accuracy

*) In a piece of paper, obtain the equations for h and k.

*) Example from the assignment:
Solve
f(x,y) = Exp[3 x] + 4 y
g(x,y) = 3 y^3 - 2 Log[x] + 7.31 x^2

Use as an initial guess xo=1 and yo=2