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/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 \((x_1,x_2)\) in which the function changes sign, so that \(f(x_1)f(x_2)<0\)

We then pick the midpoint \(x_3\) (bisect the interval) and evaluate the function there.

If \(f(x_1)f(x_3)<0\), then there is a sign change in \((x_1,x_3)\) and we repeat the procedure for this interval.
If \(f(x_1)f(x_3)>0\), then there is a sign change in \((x_3,x_2)\) and we repeat the procedure for this interval.
If \(f(x_1)f(x_3)=0\), then \(x_3\) is the root.

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

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

(iii) Repeat this procedure until a sufficiently good approximation to the zero of the function is found.
Advantage: If it converges, it does so quite rapidly. But, it may not converge depending on the function. The derivative may also be zero.
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=jXIUi7xzWIU
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 iteration is based on

\[ x_{k+1} = x_k - \frac{f'(x_k) \left( x_k - x_{k-1} \right)}{f(x_k) - f(x_{k-1})} \]

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

\[
\begin{align*}
f(x, y) &= 0 \\
g(x, y) &= 0
\end{align*}
\]

An idea that sometimes works is to use the basis of Newton’s method.

Import["Newton2D.jpg"]
Newton-Raphson Method for Non-linear Systems of Equations

\[ f(x, y) = 0 \]
\[ g(x, y) = 0 \]

Initial guess \((x_0, y_0)\)

If \((x_0 + h, y_0 + k)\) was the solution then:

\[
\begin{cases}
    f(x_0 + h, y_0 + k) = 0 \quad \text{ substitute } f(x_0, y_0) + h \frac{df}{dx} \bigg|_{x_0, y_0} + k \frac{df}{dy} \bigg|_{x_0, y_0} + \ldots = 0 \\
    g(x_0 + h, y_0 + k) = 0 \quad \text{ substitute } g(x_0, y_0) + h \frac{dg}{dx} \bigg|_{x_0, y_0} + k \frac{dg}{dy} \bigg|_{x_0, y_0} + \ldots = 0
\end{cases}
\]

To find \(h\) and \(k\):

\[
\begin{cases}
    h = \frac{D_x}{D} \\
    k = \frac{D_y}{D}
\end{cases}
\]

\[
D = \begin{vmatrix}
    f_x & f_y \\
    g_x & g_y
\end{vmatrix}
\]

\[
D_x = \begin{vmatrix}
    f_x & f_y \\
    g_x & g_y
\end{vmatrix}
\]

\[
D_y = \begin{vmatrix}
    f_x & f_y \\
    g_x & g_y
\end{vmatrix}
\]

Get new \(x_1 = x_0 + h\), \(y_1 = y_0 + k\)

and repeat the procedure

and repeat it until the solution is obtained with a desired accuracy

Check if both \(|f(x_1, y_1)|\) and \(|g(x_1, y_1)|\) are below \(\varepsilon\) stop.

\(\varepsilon\) 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