Lesson 06

We will now study how to use *Mathematica* to solve equations.

Nice site about solving polynomial equations:
http://oakroadsystems.com/math/polysol.htm

Solve: polynomial equations

***) *Mathematica* solves exactly any polynomial equation of degree less than 5**

```
Solve[x^2 + 3 x - 5 == 0, x]

Clear[roots];
roots = x /. Solve[x^4 - 2 x^3 + x + 5 == 0, x]
roots[[ 1 ]]
2. roots[[ 3 ]]

Solve[x^5 -2 x^3 + x + 5 == 0, x]
```

***) For polynomial equations of degree greater than 4, *Mathematica* can find approximate solutions**

```
NSolve[x^5 -2 x^3 + x + 5 == 0, x]
Solve[x^5 -2 x^3 + x + 5. == 0, x]
```

***) System of polynomial equations**

Where does the line y = x + 2 intersects the parabola y = 16 - x^2?

```
Plot[ {  x +2, - x^2 + 16}, {x, - 5, 4}]
Dynamic[MousePosition[“Graphics”] ]
NSolve[ { y == x +2, y == - x^2 + 16}, {x, y} ]
```

***) System of linear equations**
(linear equations are polynomial equations of degree one)

(i) The system has a unique solution
2x+y+z=7
x-4y+3z=2
3x+2y+2z=13

```
Solve[{2 x + y + z ==7, x - 4 y + 3 z ==2, 3 x + 2 y + 2 z ==13}, {x, y, z}]

Clear[A,b]
A = { { 2, 1, 1 }, {1, -4, 3}, {3, 2, 2}};
b = {7, 2, 13};
LinearSolve[A,b]
```

<u>How to solve it in a piece of paper</u>
*) *By the successive elimination of the unknowns* (called Gaussian elimination)

-) Multiply the second equation by -2 and add it to the first. Multiply the second equation by -3 and add it to the third equation. We get
9y-5z=3
x-4y+3z=2
2y-z=1
-) Multiply the last equation by -5 and add it to the first
y=2
-) Go back to the last equation above and get
z=3
-) Plug y and z in any of the equation containing x and find
x=1
*) *Matrix Inversion*
Let us call X={x, y, z}
If A has an inverse then
A.X= b
can be written as
Inverse[A].A.X= Inverse[A].b
which gives
X = Inverse[A]. b

(ii) The system has infinite solutions
3x+2y-z+w=0
x-3z=-1
-y+w=2

```
Solve[{3 x + 2 y - z + w ==0, x - 3 z == -1, -y + w ==2}, {x,y,z}]
```

```
Clear[A,b]
A = { { 3, 2, -1, 1 }, {1, 0, -3, 0}, {0, -1, 0, 1}};
b = {0,-1,2};
LinearSolve[A,b]
```

(iii) The system has no solution
2x+y+z=7
x-4y+3z=2
3x-3y+4z=13

```
Solve[{2 x + y + z ==7, x - 4 y + 3 z ==2, 3 x - 3 y +4 z ==13}, {x, y, z}]
```

```
Clear[A,b]
A = { { 2, 1, 1 }, {1, -4, 3}, {3, -3, 4}};
b = {7, 2, 13};
LinearSolve[A,b]
```

```
Det[A]
```

<u>Iterative Methods</u>
**) Jacobi Iteration*
When can rewrite A in A.X=b as
A = (D + L + U)
leading to D.X = - (L+U).X + b,
where D is a diagonal matrix and L and U are, respectively, lower and upper triangular matrices with zeros on the diagonal.

This suggests the iteration

$X_{i+1} = - D^{-1}(\mathrm{L+U}).X_i + D^{-1}.b$

[This method as presented here is seldom used, because it is hard to determine if it converges and because the Gauss-Seidel method described below, which may converges even when Jacobi does not, converges faster.]

Example:

4x - y = 2

-x + 4y - z = 6

-y + 4z = 2

```
Clear[Diag,InDiag,LowM,UpM,BB,b,x,y,z];
Diag = { {4., 0, 0}, {0, 4, 0}, {0,0,4} };
InDiag = Inverse[Diag];
LowM = { {0,0,0}, {-1.,0,0}, {0,-1.,0} };
UpM = { {0,-1.,0}, {0,0,-1.}, {0,0,0} };
BB = - InDiag.(LowM + UpM);
b = {2.,6.,2.};

Print["Solution:"];
LinearSolve[(Diag+LowM+UpM),b]
Print[];
Print[];

Print["Solution by iterations: Jacobi iteration"];
{x,y,z} = {0.,0.,0.};
Do[
{x,y,z} = BB.{x,y,z} + InDiag.b;
Print["iteration ", k];
Print[{x,y,z}];
Print[ ];
,{k,1,10}];
```

**) Gauss-Seidel Method*

Here we use

$X_{i+1} = - (D + L)^{-1}.\mathrm{U}.X_i + (D + L)^{-1}.b$

```
Clear[Diag,LowM,InDL,UpM,b,x,y,z];
Diag = { {4., 0, 0}, {0, 4, 0}, {0,0,4} };
LowM = { {0,0,0}, {-1.,0,0}, {0,-1.,0} };
InDL = Inverse[Diag + LowM];
UpM = { {0,-1.,0}, {0,0,-1.}, {0,0,0} };
b = {2.,6.,2.};

Print["Solution by iterations: Gauss-Seidel Method"];
{x,y,z} = {0.,0.,0.};
Do[
{x,y,z} = - InDL.UpM.{x,y,z} + InDL.b;
Print["iteration ", k];
Print[{x,y,z}];
```

```
Print[ ];
,{k,1,10}];
```

Transcendental equations

(Equations involving exponential, logarithmic, and trigonometric functions can only occasionally be solved with Solve or NSolve, most commonly we need FindRoot)

```
Clear[x];
Solve[ x^2 == Exp[x],  x ]
NSolve[ x^2 == Exp[x],  x]
FindRoot[ x^2 == Exp[x],  {x, 1.} ]
```

Where do the functions Sin[x] and x^2-1 cross?
A graph of the function helps selecting an initial guess
To find where they meet:
Plot[{Sin[x], x^2 -1}, {x, -Pi, Pi}]   (Or the root of the function: Plot[Sin[x] - x^2 + 1, {x, -Pi, Pi}])
The two functions intersect near x= - 1 and x=1
FindRoot[Sin[x] == x^2 - 1, {x, -1}]
FindRoot[Sin[x] == x^2 - 1, {x, 1}]

```
FindRoot[Exp[2 x] -2 Exp[ x] + 1 ==0, {x,90}]
FindRoot[Exp[2 x] -2 Exp[ x] + 1 ==0, {x,90}, MaxIterations -> 300]
```

***)**
(i) FindRoot[lhs == rhs, {x, xo}] solves the equation lhs=rhs using Newton's method with starting value xo. Newton's method fails if the derivative of the function cannot be computed.
(ii) FindRoot[lhs == rhs, {x, xo,x1}] solves the equation lhs=rhs using (a variation of) the secant method with starting values xo and x1.
The secant method is a bit slower.

```
FindRoot[ Exp[-x] == x, {x, 1} ]
FindRoot[ Exp[-x] == x, {x, 1, 2} ]
```

***) System of equations**

```
FindRoot[ { Exp[x]  + Log[y] == 2, Sin[x] + Cos[y] ==1 }, {x, 1}, {y,1} ]
```