Get Started with MotionGenesis – Short

F=ma www.MotionGenesis.com	To download and install a demo version of the program (PC/Macintosh), go to http://www.MotionGenesis.com Click on the **Download** Software button.

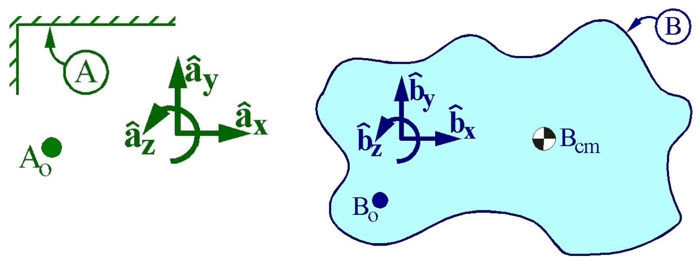
Math

Browse to the MotionGenesis folder and double-click on: **MotionGenesisStartHere**
On line (1), type: `sum = 2 + 2`
To try symbolic manipulation, type: `fred = 3*sin(t)^2 + 2*cos(t)^2`
To evaluate fred at t=pi/3, type: `test = Evaluate( fred, t = pi/3 )`
To convert units from inches to cm, type: `inchToCm = ConvertUnits(inch,cm)`
To find the roots of the quadratic equation, type:

```
Constant a, b, c
Variable x
Roots = GetQuadraticRoots( a*x^2 + b*x + c, x )
```

To save input to the text file FirstDemo.txt, type: `Save FirstDemo.txt`
To save input and output to file FirstDemo.html, type: `Save FirstDemo.html`
For general help and/or a list of commands, type: `Help`
For help with a command (e.g., Solve), type: `Help SOLVE`
To exit the program, type `Quit`

Vectors



1. To create right-handed orthogonal unit vectors Ax>, Ay>, Az> fixed in a RigidFrame A, type:

`RigidFrame A`

2. To define a vector v> in terms of Ax>, Ay>, Az>, type: `v> = 2*Ax> + 3*Ay> + 4*Az>`
   Similarly, one can define a vector w> with: `w> = 6*Ax> + 7*Ay> + 8*Az>`
3. To multiply the vector v> by 5, type: `vFive> = 5 * v>`
4. To add vectors v> and w>, type: `addVW> = v> + w>`
5. To dot-multiply v> with w>, type: `dotVW = Dot( v>, w> )`
6. To cross-multiply v> with w>, type: `crossVW> = Cross( v>, w> )`
7. To find the magnitude of v>, type: `magV = GetMagnitude( v> )`
8. To find the magnitude-squared of v>, type : `vSquared = GetMagnitudeSquared( v> )`
9. To find the unit vector in the direction of v>, type: `unitV> = GetUnitVector( v> )`
10. To find the angle between v> and w>, type: `theta = GetAngleBetweenVectors(v>, w>)`
11. To save input (for subsequent re-use), type **Save VectorSampleCommands.txt**
12. To save input and output, type: **Save VectorSampleCommands.html**
13. To quit the program, type: **Quit.**

Solving linear algebraic equations

$$\begin{aligned} 2*x + 3*y &= \sin(t) \\ 4*x + t*y &= \cos(t) \end{aligned}$$

To symbolically solve the previous set of linear equations for x and y, type

```
Variable  x, y
Zero[1] = 2*x +  3*y  -  sin(t)
Zero[2] = 4*x +  5*y  -  cos(t)
Solve( Zero, x, y )
```

To save input (for subsequent re-use), type **Save SolveLinearEqn.txt**
To save input and output, type: **Save SolveLinearEqn.html**

Solving one nonlinear algebraic equation

$$x^2 - \cos(x) = 0$$

To numerically solve the previous nonlinear equation for x , type:

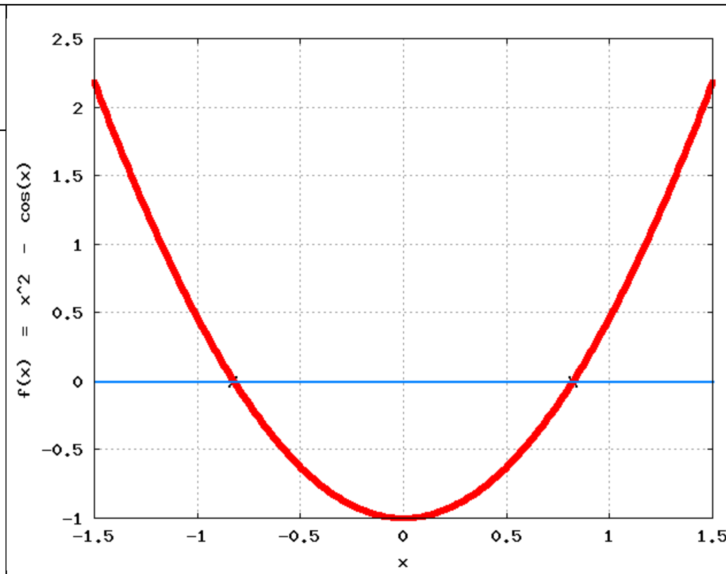
```
Variable  x
Solve( x^2 - cos(x),  x = 0.2 )
```

Nonlinear equations may have multiple solutions.

The program's solution of $x = 0.8241323$ depends on the starting guess which is specified by the argument $x = 0.2$.

If instead, one starts with a guess of $x = -9$, the program produces a different solution, namely $x = -0.8241323$.

The program frequently converges to a solution close to the starting guess.



To save input (for subsequent re-use), type **Save SolveNonlinearEqn1.txt**
To save input and output, type: **Save SolveNonlinearEqn1.html**

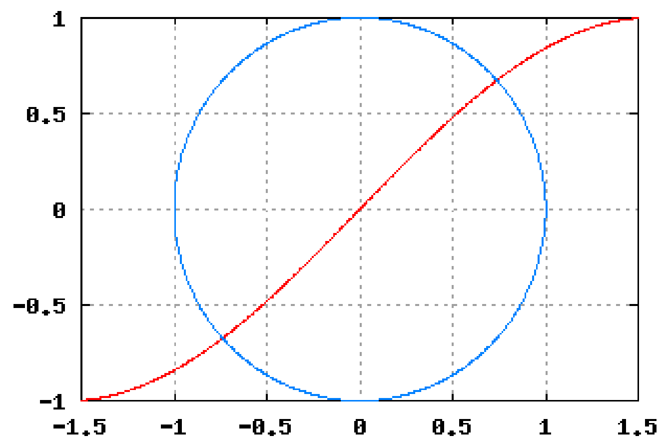
Solving sets of nonlinear algebraic equations

Equations for a circle and sine curve.

```
x^2 + y^2 = 1
y = sin(x)
```

To numerically solve the previous set of nonlinear equations for x and y, type:

```
Variable  x, y
Zero[1] = x^2 + y^2 - 1
Zero[2] = y - sin(x)
Solve( Zero,  x = 3, y = 5 )
```



These nonlinear equations have two solutions. The program's solution of x = 0.739085 and y = 0.673612 depend on the guess.
The program frequently converges to a solution close to the starting guess.

To save input (for subsequent re-use), type **Save SolveNonlinearEqn2.txt**
To save input and output, type: **Save SolveNonlinearEqn2.html**

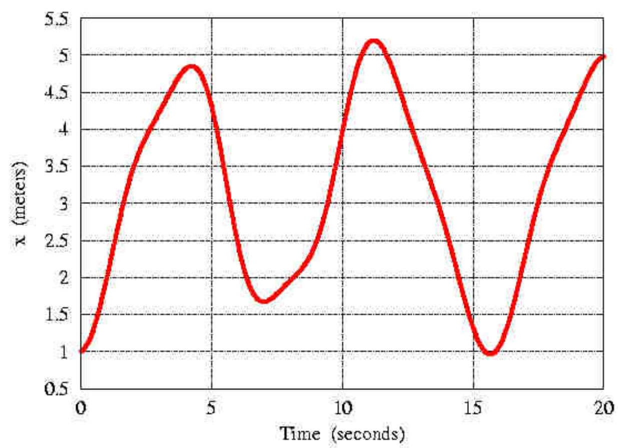
Solving ODEs (differential equations)

Solve the nonlinear ordinary differential equation

```
x'' = cos(2*t) + sin(x)
```

with the initial values x=1 m and x'=0.2 m/s,
Create a plot with t varying from 0 to 20 seconds.

Note: `t` is the independent variable time.
The prime symbol `'` denotes time-differentiation.



This plot was generated with the MotionGenesis Plot command

To numerically solve this ODE with output every 0.02 sec for the given initial values, type

```
Variable x'' = cos(2*t) + sin(x)
Input    x = 1 m,   x' = 0.2 m/s,   tFinal = 20 sec,   tStep = 0.02 sec
OutputPlot  t sec,   x m,   x' m/s
ODE() odeOutputFile          % Solves ODE (no MATLAB® required)
ODE() odeOutputFile.m        % Creates MATLAB® file that solves ODE.
```

Next: See MotionGenesisTutorial.pdf installed in your: MotionGenesis -> MGToolbox folder (after you download/install)