A casual approach to numerical modeling

Spring-Mass-Damper-System - part 3.

by George Lungu

*A tutorial about the implementation of a **dynamic** S-M-D model in Excel 2003*

<excelunusual.com>

Advantages of a dynamic model:

1. Small size spreadsheet since there are just a few formulas (written for only one or a small group of time steps)
2. The model can be run as an infinite loop
3. The model can be paused at any time and restarted from exactly the same exact time step any time later

Drawbacks of the dynamic model:

There are no real drawbacks to a dynamic model. Dynamic models have slightly more VBA code and can sometimes can be a bit slower than the static ones but there are ways to overcome this by writing mixed models which are dynamic models nonetheless, but having 10-1000 lines of static calculation.

These mixed models are in average as fast as their static counterparts and have all the advantages of the dynamic models.

A theoretical review:

Since it is very important to understand the principles of numerical simulation, let's again look at the principles behind modeling the spring-mass-damper system. This was done in the first part of the presentation already.

Definitions used in this model:

The mechanics studied in the high school physics class is very simple. In that class the movement of a body is either uniform or uniformly accelerated. By definition, the velocity (speed) is the first derivative of position with respect to time and the acceleration is the first derivative of the speed with respect to time or the second derivative of position with respect to time.

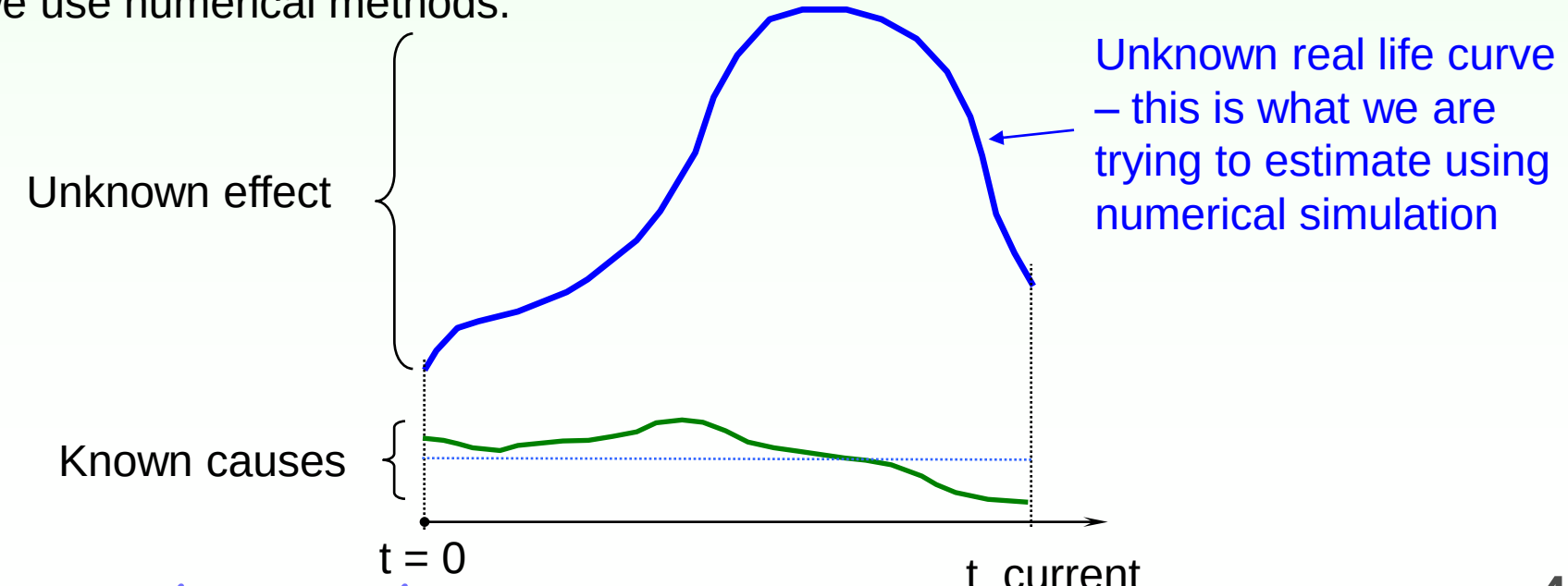
Velocity formula: $v = \frac{dx}{dt}$

Acceleration formula: $a = \frac{dv}{dt} = \frac{d^2x}{dt^2}$

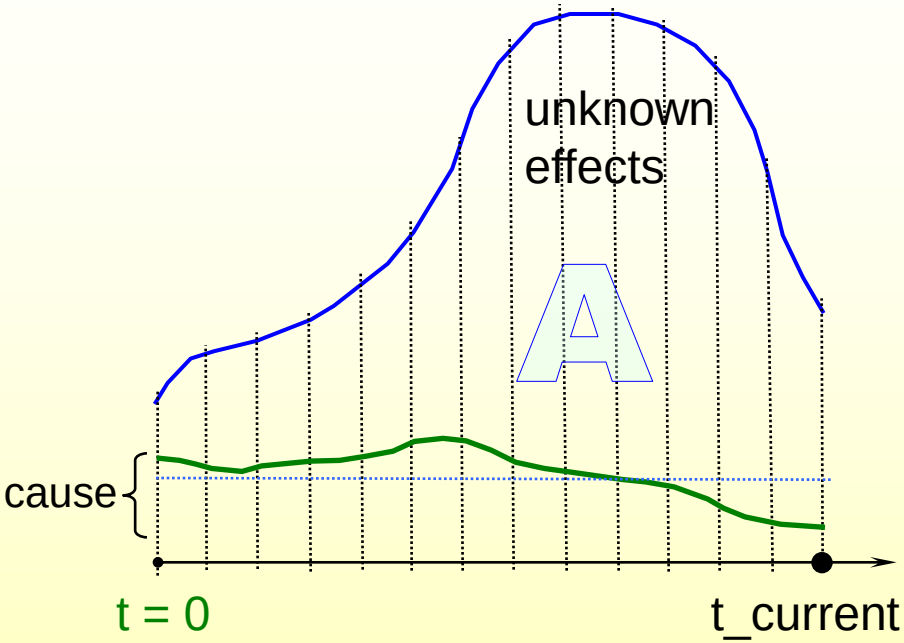
We can also derive the acceleration from Newton's second law: $a = \frac{F}{m}$

In general, elementary formulas are not very useful in a real life situations, since we almost never encounter a body traveling at a constant speed or with a perfectly constant acceleration over long periods of time. The formulas serve two purposes, firstly they have educational value and secondly they can be applied with good precision in real life situations during very small intervals of time. The question is, how good a precision we are talking about? And the answer is - any precision, the shorter the time interval, the better the precision.

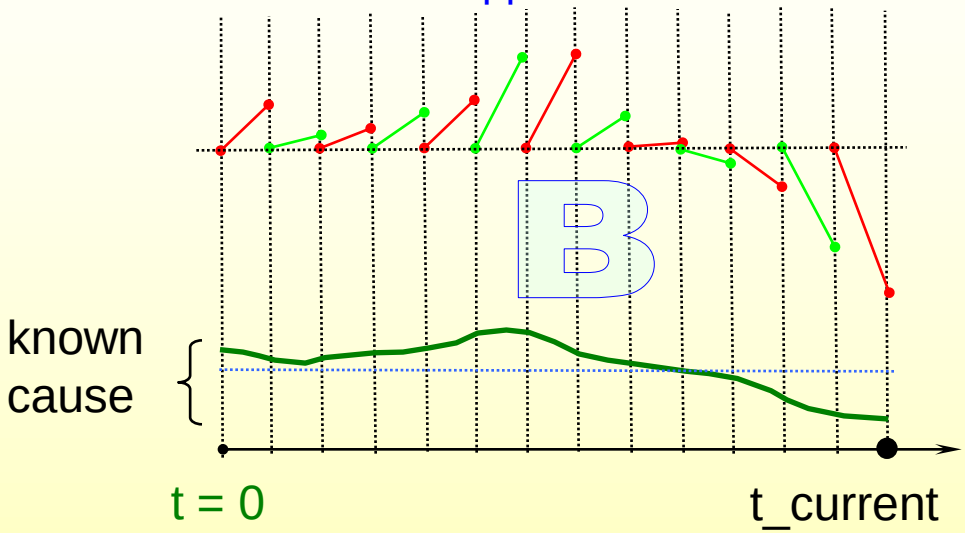
The picture below shows the evolution of a general natural phenomenon in time (blue). A certain cause (the green curve below) produce an effect (the upper blue curve). Most of the times calculating the effect at moment "t" is either impossible or very difficult unless we use numerical methods.



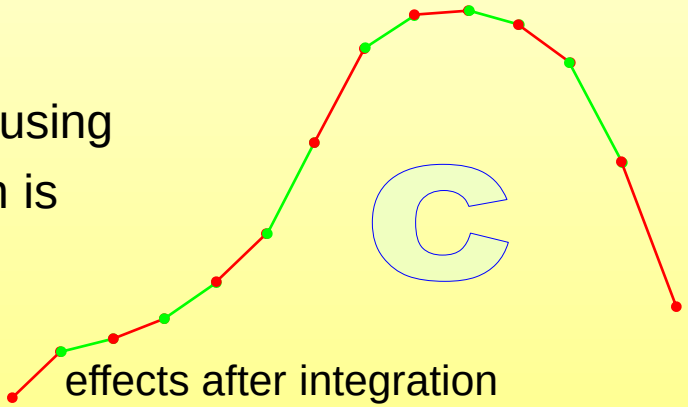
If we divide the modeling time interval in N elementary intervals (A), we can calculate the effects on each interval using the cause during that time interval (B) . If the intervals are very short, we can use extremely simple laws to calculate the effects.



elementary effects (on very small intervals) calculated by using linearization and simple numerical method approximations

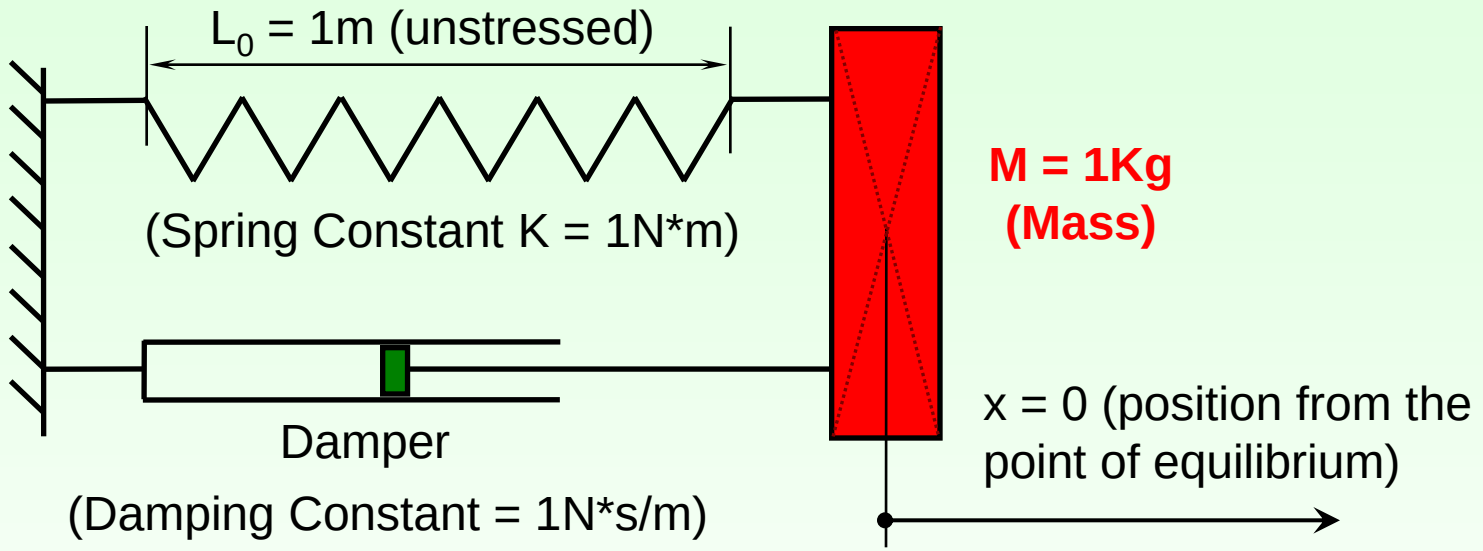


We can approximately calculate the global effect using the algebraic sum of the elementary effects which is nothing else than numerical integration (C) =>



effects after integration
(algebraic summation of effects)

Let's review our particular system:



There are a total of 3 forces acting on mass M:

1. Elastic force : $F_e = - K * x$ (the elastic force is proportional and opposite to the spring deformation)

2. Damping force : $F_d = - DC * v$ (the force of friction is proportional and opposite to the speed – the proportionality factor is the damping coefficient)

3. Inertia : $F_i = - M * a$

Because this is a one-dimensional problem we dropped the vector notations. We assume that positive vectors face to the right while negative ones face to the left.

As a parenthesis, I like to express the damping (friction) force function of the damping ratio rather than damping coefficient. This way the damping time is only related to the resonance period of the system.

The formula for the damping ratio is:

$$DR = \frac{DC}{2 \cdot \sqrt{m \cdot k}}$$

The damping force becomes:

$$F_{damping} = -2 \cdot v \cdot DR \cdot \sqrt{m \cdot k}$$

We can apply Newton's second law of dynamics to calculate the acceleration:

$$m \cdot a = F_{elastic} + F_{damping}$$

This way the acceleration becomes:

$$a = -\frac{k \cdot x + 2 \cdot DR \cdot v \cdot \sqrt{m \cdot k}}{m}$$

There will be only three formulas in our spreadsheet for "a", "v" and "x". These quantities will be calculated for each time step using data from the previous time step.

Let's review the calculations used :

Since the force is the cause of change in movement, our algorithm will calculate the current acceleration first, using the velocity and position from the previous time step since the velocity and position of the current step are not yet available:

$$a_{current} = -\frac{k \cdot x_{previous} + 2 \cdot DR \cdot v_{previous} \cdot \sqrt{m \cdot k}}{m}$$

The velocity is calculated second, using the acceleration from the current time step and the velocity from the previous time step

$$a_{current} = \frac{dv}{dt} = \frac{v_{current} - v_{previous}}{dt} \Rightarrow$$

$$v_{current} = v_{previous} + a_{current} \cdot dt$$

The position is calculated third, using the velocity from the current time step and the position from the previous time step

$$v_{current} = \frac{dx}{dt} = \frac{x_{current} - x_{previous}}{dt} \Rightarrow$$

$$x_{current} = x_{previous} + v_{current} \cdot dt$$

The outline of a dynamic model:

The dynamic model makes use of an infinite loop, which make the calculations advance in time. Instead of a large table with formulas, this time we have a small number of cells with formulas in the upper part of the table (usually calculations take place in the top most row let's call it "the current time row" or "the present") and a large number of cells with historical data (constants) under this active row.

During every loop sequence, a new set of calculations in the "current time row" occur. The input data used in this calculations is taken from the last set of historical data (previous time step). The historical data is organized like a FIFO stack: the oldest time step of historical data is deleted each cycle, the rest of historical data is shifted down one time step per loop cycle to make room for the result of the new calculation which now will represent the latest historical data on top of the stack.

A snapshot of the dynamic model worksheet layout:

t	a	v	x
		0	-0.4
10.08	-0.060209	-0.206802	-0.000691
10.08	-0.124699	-0.18874	0.06135
9.99	-0.177386	-0.15133	0.117972
9.9	-0.212783	-0.098114	0.163371
9.81	-0.227081	-0.034279	0.192805
9.72	-0.218564	0.033845	0.203089
9.63	-0.187801	0.099415	0.192935
9.54	-0.137624	0.155755	0.163111
9.45	-0.072865	0.197042	0.116384
9.36	0.000104	0.218902	0.057272
9.27	0.073996	0.218871	-0.008399
9.18	0.141342	0.196672	-0.07406

Time

Initial conditions (constants)

Current row or present - active calculations

Latest History - constants

The initial conditions idle in the blue area most of the time. During the Reset only, the "Reset" macro will copy them and paste below the active row" green. During the next cycle, the active (red) formulas will use them **once** to calculate (a,v,x) for the second time step (0+dt). After that operation, the initial condition data becomes "history" and is shifted down the stack until it reaches the end and is discarded

Moving history Stack (past) - constants

Copy the workbook "Osc_2" and rename the copy "Osc_3"

In this worksheet we will add the following features:

1. We will move away from the static large table of formula simulation into a dynamic style of simulation (loop). The table will look almost the same but now most of it is filled with historical data (constant numbers) except one row of active formulas
2. Upgrade the macros and add a reset button
3. Generate a more detailed animation of the system

The worksheet structure:

Column B contains a time series, column C will contain a series of accelera-tions, column D a series of speeds and column E the position of the oscillating mass with respect with the neutral point. There will be a total of only three formulas in the worksheet placed in row “9”:

C9: “ =-(E10*B$2+D10*B$3*2*SQRT(B$1*B$2))/B$1”

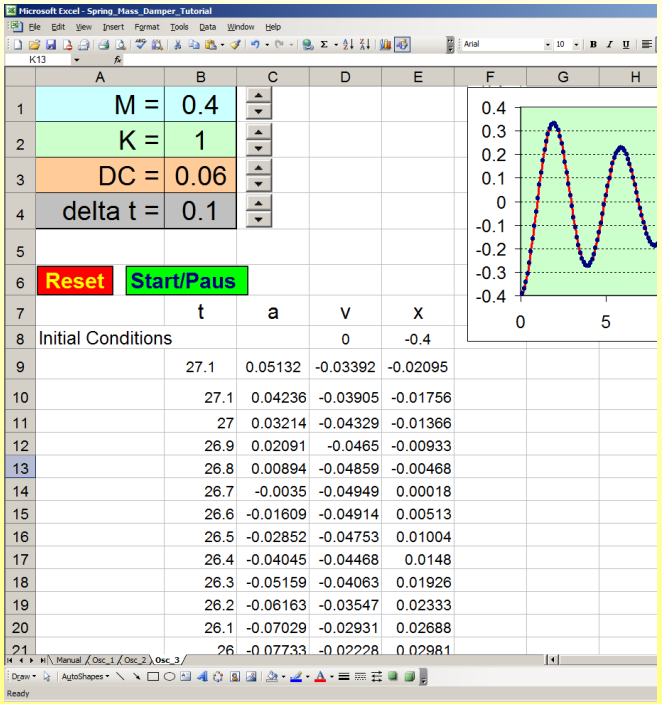
D9: “ =D10+C9*B$4”

E9: “ =E10+D9*B$4”

The position is the integral of the velocity (a first order approximation was used)

The velocity is calculated as the integral of acceleration (a first order approximation was used)

The acceleration is equal to the sum of elastic and friction forces divided by the proof mass



We used the following recursive, first order approximation of the integral of a function f(t):

$$\int_{t0}^{t} f(t)dt = \int_{t0}^{t-\Delta t} f(t)dt + f(t)\cdot\Delta t$$

Macros used in this worksheet:

Besides straightforward spinner button macros (which are not shown in the here) the following two macros were used: **Reset** and **StartStop**

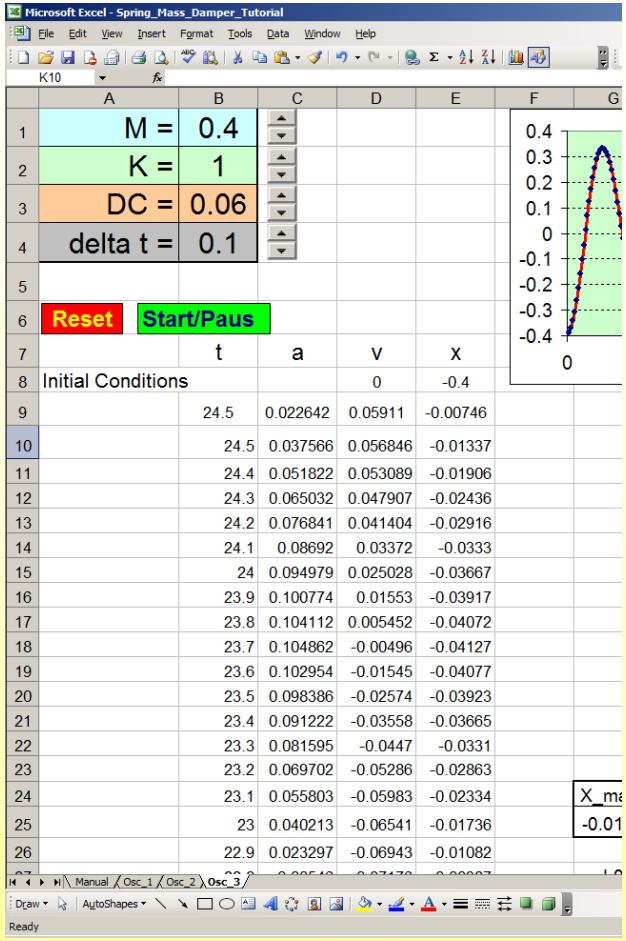
```
Sub reset()
DoEvents
Range("B9") = 0
Range("B10:E1010").Clear
Range("B10:E10") = Range("B8:E8").Value
End Sub
```

The reset macro achieves three different functions:

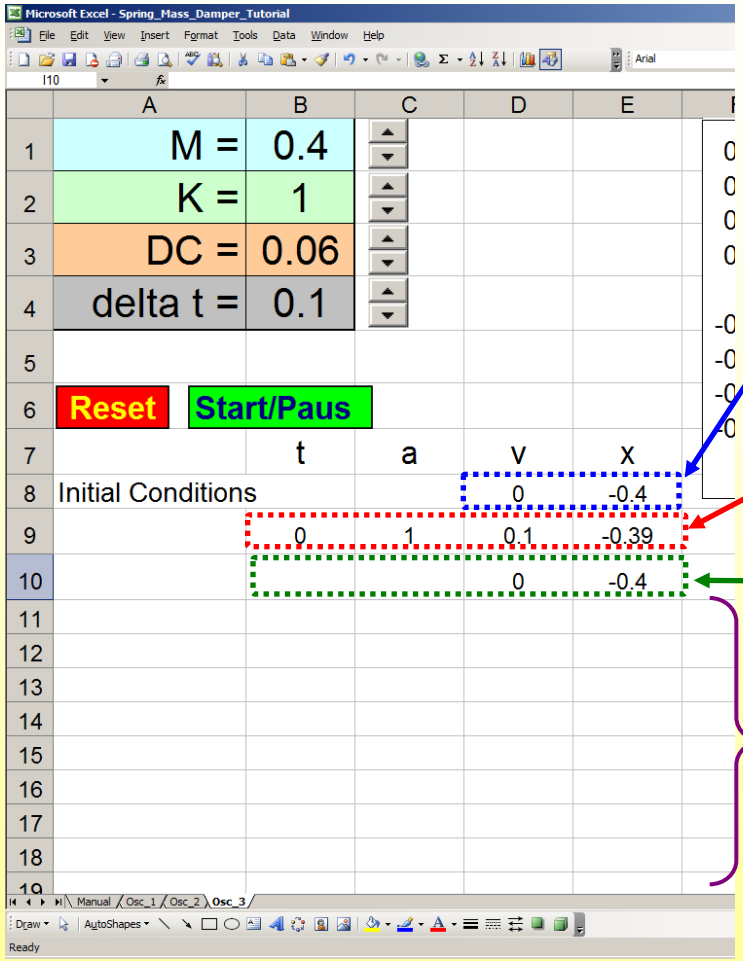
1. Resets the time in the current time row to zero
2. Clears the history (deletes everything up the current time row)
3. After the history is cleared, the macro pastes the initial conditions one row below the active row so that the active formulas can use these initial conditions to calculate the values of the first time step (acceleration, speed, position)

Macros used in this worksheet:

Let's see the Reset macro in action:



Before **"Reset"**



Initial conditions (constants)

Current row or present - active calculations

Latest History - constants

Moving history Stack (past) - constants

After **"Reset"**

The StartStop macro:

```
Sub StartStop()
RunSim = Not (RunSim)
   Do While RunSim = True And [B9] =< 31
   DoEvents
   Range("B10:E1010") = Range("B9:E1009").Value
   Range("B9") = Range("B9") + Range("B4")
   Loop
End Sub
```

Macro declaration statement

It logically "flips" the Boolean control variable RunSim so that we can use the same button to start or stop the macro

Conditional Do loop statement declaration. The loop will run until either RunSim is False or the time (in cell B9) becomes larger than 31 seconds

The macro hogs the processor, this statement allows other processes to happen during each loop cycle such as updating the chart.

Increments time in cell B9 by delta t (which is the time step from cell B4)

End of loop statement

End of macro statement

The **StartStop** macro achieves three different functions:

1. If the calculation loop is stopped it will start it and vice versa
2. If the calculation loop is running it will increment the time
3. If the calculation loop is running it will delete the earliest (bottom) entry in the history, shift the whole history stack down and write the active data values on the top of the history stack (see the **bold blue line** within the macro). This way the history stack will act like a FIFO (First-In-First-Out type of shift register).

Let's observe the StartStop macro and the conditional "Do" loop in action:

t	a	v	x
		0	-0.4
0	1	0.1	-0.39
		0	-0.4

After "Reset"

t	a	v	x
		0	-0.4
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After first loop iteration

t	a	v	x
		0	-0.4
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After second loop iteration

t	a	v	x
		0	-0.4
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After third loop iteration

t	a	v	x
		0	-0.4
0.4	0.694646	0.434065	-0.262123
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After fourth loop iteration

t	a	v	x
		0	-0.4
0.5	0.572949	0.49136	-0.212987
0.4	0.694646	0.434065	-0.262123
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After fifth loop iteration

t	a	v	x
		0	-0.4
0.6	0.439238	0.535284	-0.159459
0.5	0.572949	0.49136	-0.212987
0.4	0.694646	0.434065	-0.262123
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After sixth loop iteration

t	a	v	x
		0	-0.4
0.7	0.297084	0.564992	-0.102959
0.6	0.439238	0.535284	-0.159459
0.5	0.572949	0.49136	-0.212987
0.4	0.694646	0.434065	-0.262123
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39
		0	-0.4

After seventh loop iteration

t	a	v	x
		0	-0.4
0.8	0.150199	0.580012	-0.044958
0.7	0.297084	0.564992	-0.102959
0.6	0.439238	0.535284	-0.159459
0.5	0.572949	0.49136	-0.212987
0.4	0.694646	0.434065	-0.262123
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044
0	1	0.1	-0.39

After eighth loop iteration

t	a	v	x
		0	-0.4
0.9	0.002346	0.580247	0.013067
0.8	0.150199	0.580012	-0.044958
0.7	0.297084	0.564992	-0.102959
0.6	0.439238	0.535284	-0.159459
0.5	0.572949	0.49136	-0.212987
0.4	0.694646	0.434065	-0.262123
0.3	0.800994	0.364601	-0.30553
0.2	0.888986	0.284501	-0.34199
0.1	0.956026	0.195603	-0.37044

After ninth loop iteration

Observe the data shifting and renewal that takes place

The macro can be started at any time and restarted from the place it was turned off

The next tutorial will explain the animation used in the simple mass-spring-damper system model