Dynamic Macros for Scientific Modeling part #1

by George Lungu

<excelunusual.com>

The static time dimension:

Excel is generally used as a static tool. As an Excel companion, Visual Basic for Applications (VBA) is an event-driven programming language which allows processes automation and the creation of user defined functions. My first experience with VBA was in creation of simple control interfaces (e.g. buttons) or running through simple calculation loops in order to achieve a certain result.

Beginners can use a ***"static time dimension"*** which means some parameters of a model are calculated in cells situated along a horizontal line. The time however will advance vertically. In this case we obtain a fixed size table of values in which time is located on one of the columns. The information can be presented on a scatter plot. If we can figure out a way to chart one subset of this table and vary the position of the subset in real time we can "animate our model" which means we can give the illusion of movement on a chart.

How can we do this?

By using the spreadsheet function "Offset()" and varying the argument controlling the offset value through a VBA macro.

A basic oscillation model:

Let’s try to create the model of a fixed frequency oscillation. We’ll generate a 2 x 1000 table containing an increasing time series on the first column and a sinusoidal time function on the second column:

Insert a worksheet and name it “DynamicMacros”

Create the following labels:

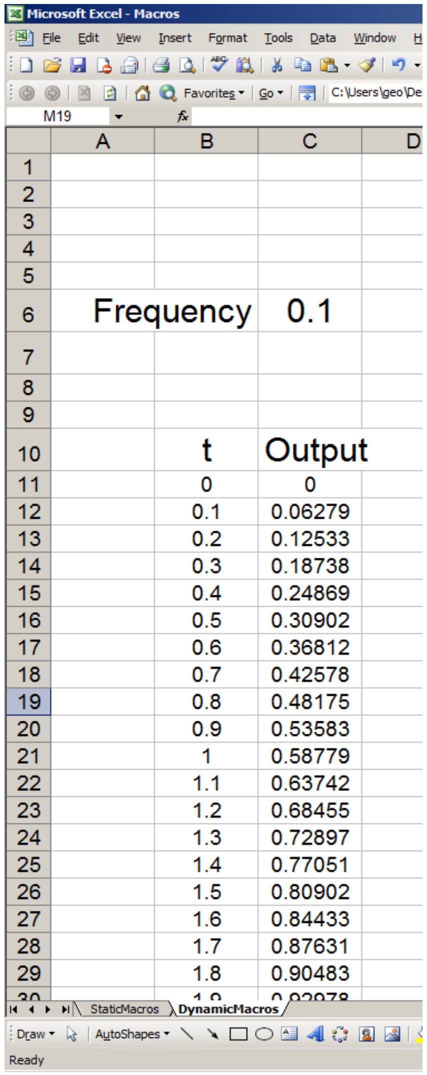
- Cell B6: Frequency
- Cell B10: Time
- Cell C10: Output

Choose some number for the frequency:

- Cell C6: =0.1

Generate the sinusoidal function:

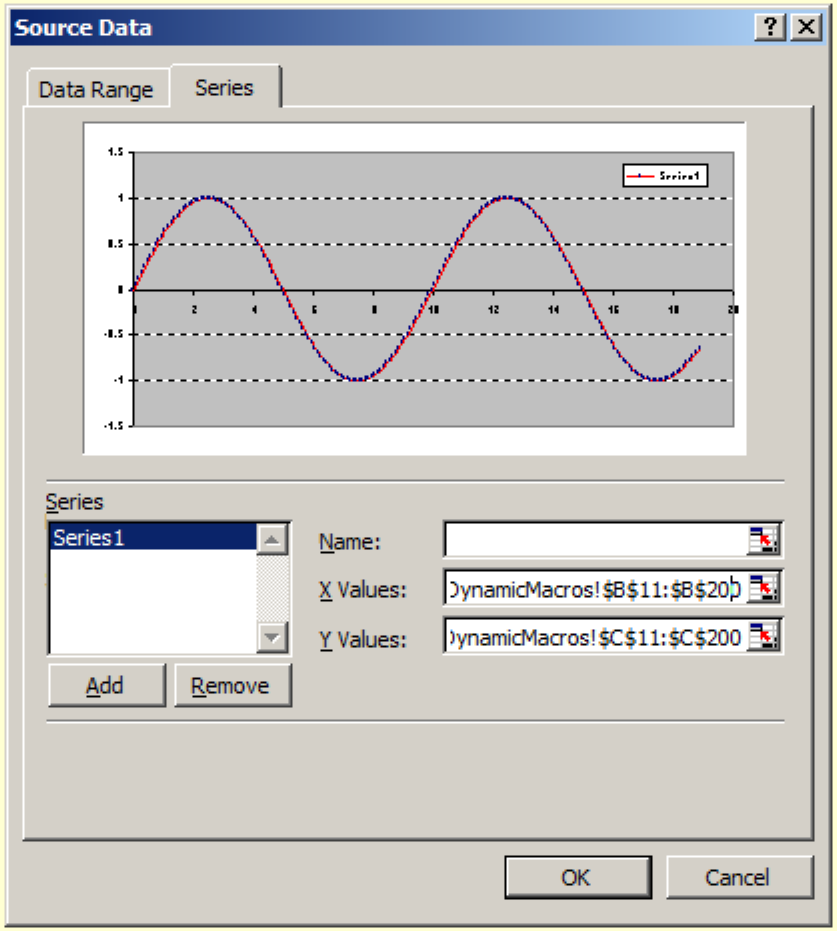
- Cell B11: =0
- Cell C11: =SIN(2*PI()*C$6*B11)
- Cell B12: =B11+0.1 (the time step is chosen 0.1 second)
- Cell C12: =SIN(2*PI()*C$6*B12)
- Copy {B12:C12} down to {B1010:C1010}


Microsoft Excel - Macros
File Edit View Insert Format Tools Data Window H
Favorites Go C:\Users\geo\De
M19
StaticMacros
DynamicMacros
Draw AutoShapes
Ready

	A	B	C	D
1				
2				
3				
4				
5				
6		Frequency	0.1	
7				
8				
9				
10		t	Output	
11		0	0	
12		0.1	0.06279	
13		0.2	0.12533	
14		0.3	0.18738	
15		0.4	0.24869	
16		0.5	0.30902	
17		0.6	0.36812	
18		0.7	0.42578	
19		0.8	0.48175	
20		0.9	0.53583	
21		1	0.58779	
22		1.1	0.63742	
23		1.2	0.68455	
24		1.3	0.72897	
25		1.4	0.77051	
26		1.5	0.80902	
27		1.6	0.84433	
28		1.7	0.87631	
29		1.8	0.90483	

Plot a small part of the sinusoidal function:

Create a 2D scatter plot of the {B11:C200} range for visual verification:

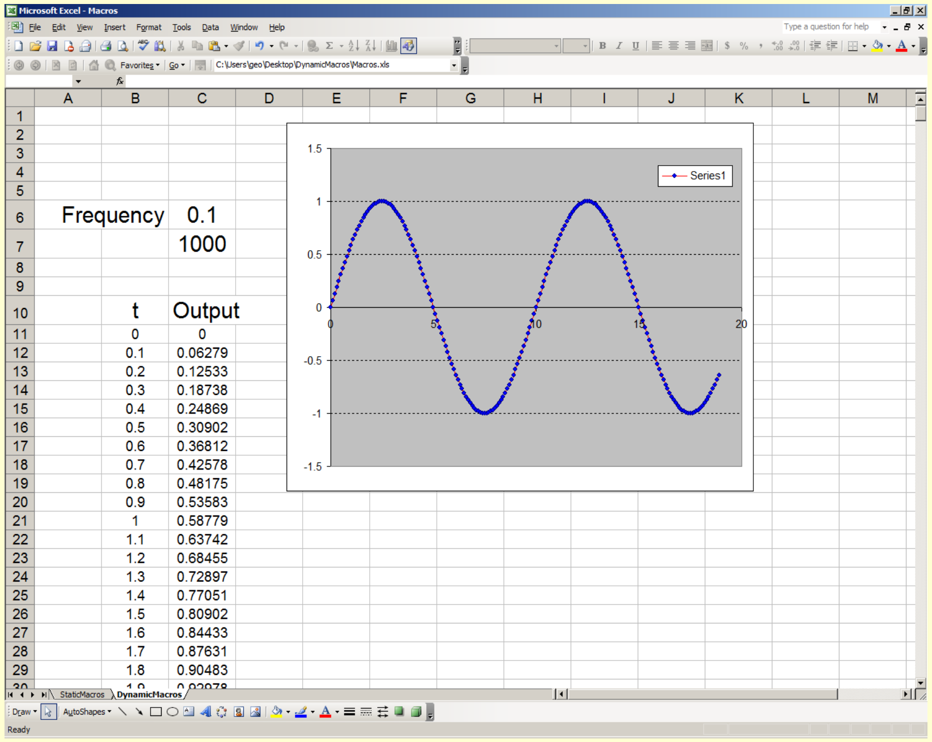


Series: Series1

Name:

X Values: DynamicMacros!$B$11:$B$200

Y Values: ynamicMacros!$C$11:$C$200


Microsoft Excel - Macros
File Edit View Insert Format Tools Data Window Help
C:\Users\geo\Desktop\DynamicMacros\Macros.xls
Series1
StaticMacros
DynamicMacros
Draw
AutoShapes
Ready

	A	B	C
6		Frequency	0.1
7			1000
10		t	Output
11		0	0
12		0.1	0.06279
13		0.2	0.12533
14		0.3	0.18738
15		0.4	0.24869
16		0.5	0.30902
17		0.6	0.36812
18		0.7	0.42578
19		0.8	0.48175
20		0.9	0.53583
21		1	0.58779
22		1.1	0.63742
23		1.2	0.68455
24		1.3	0.72897
25		1.4	0.77051
26		1.5	0.80902
27		1.6	0.84433
28		1.7	0.87631
29		1.8	0.90483
30		1.9	0.92978

A very important function, the "OFFSET()" function:

OFFSET(**reference**, **rows**, **cols**, height, width)

Reference is the reference from which you want to base the offset. Reference must refer to a cell or range of adjacent cells; otherwise, OFFSET returns the #VALUE! error value.

Rows is the number of rows, up or down, that you want the upper-left cell to refer to. Using 5 as the rows argument specifies that the upper-left cell in the reference is five rows below reference. Rows can be positive (which means below the starting reference) or negative (which means above the starting reference).

Cols is the number of columns, to the left or right, that you want the upper-left cell of the result to refer to. Using 5 as the cols argument specifies that the upper-left cell in the reference is five columns to the right of reference. Cols can be positive (which means to the right of the starting reference) or negative (which means to the left of the starting reference).

Height is the height, in number of rows, that you want the returned reference to be. Height must be a positive number.

Width is the width, in number of columns, that you want the returned reference to be. Width must be a positive number.

Use the "OFFSET()" function to isolate a certain row in the output table:

Create the following labels:

- Cell B7: Index , Cell E10: Selection

Choose an arbitrary index number:

Cell C7: =5

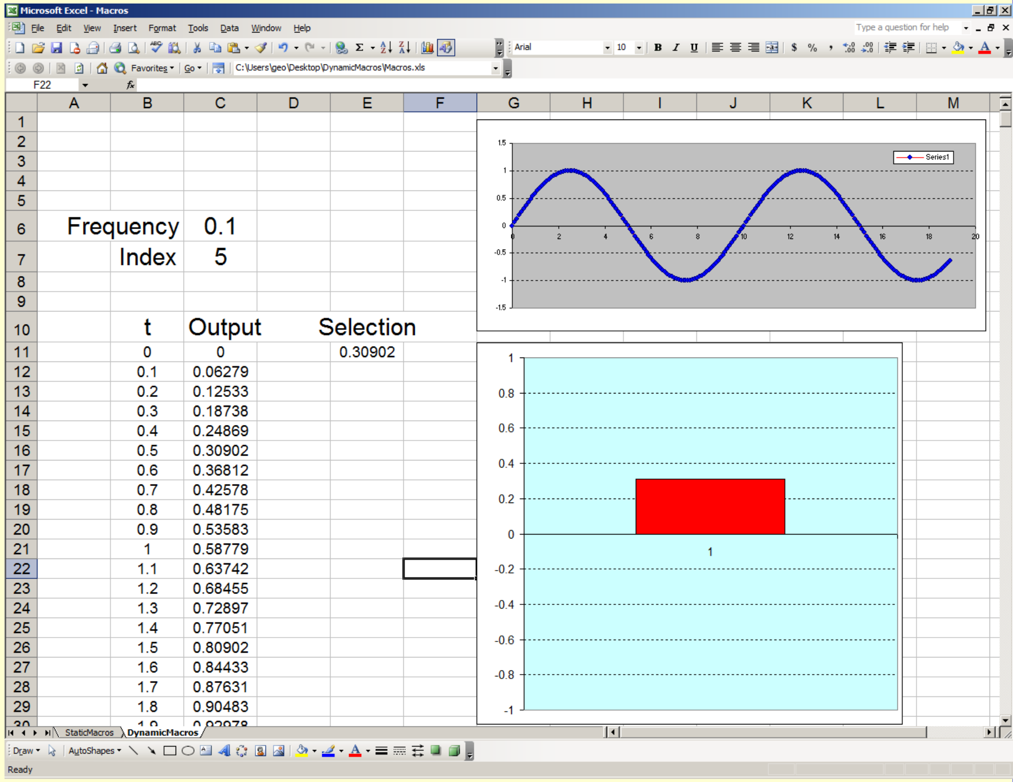
Use the "OFFSET()" function:

- Cell E11: OFFSET(C11,C7,0)

(note here that by changing the index (cell "C7"), we can select a different cell from column "C" to be displayed in cell "E11"

Create a "Column Chart" out of cell E11:

- Select cell E11 => Insert => Chart => Column => Finish
- Format the vertical axis between -1 and 1


Microsoft Excel - Macros
File Edit View Insert Format Tools Data Window Help
C:\Users\geo\Desktop\DynamicMacros\Macros.xls
F22
Series1
StaticMacros
DynamicMacros
Ready

	A	B	C	D	E
6		Frequency	0.1		
7		Index	5		
10		t	Output		Selection
11		0	0		0.30902
12		0.1	0.06279		
13		0.2	0.12533		
14		0.3	0.18738		
15		0.4	0.24869		
16		0.5	0.30902		
17		0.6	0.36812		
18		0.7	0.42578		
19		0.8	0.48175		
20		0.9	0.53583		
21		1	0.58779		
22		1.1	0.63742		
23		1.2	0.68455		
24		1.3	0.72897		
25		1.4	0.77051		
26		1.5	0.80902		
27		1.6	0.84433		
28		1.7	0.87631		
29		1.8	0.90483		
30		1.9	0.92978		

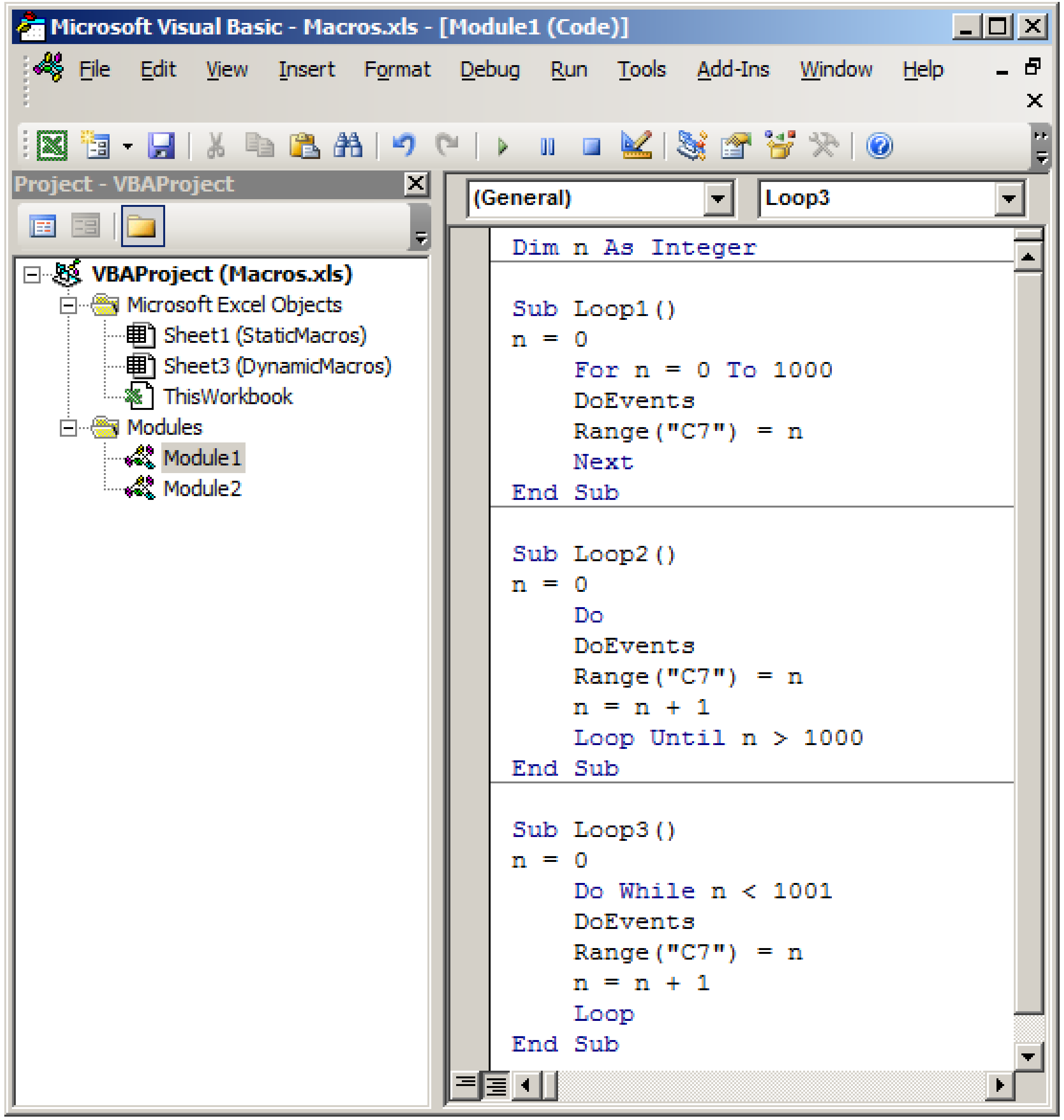
Three simple dynamic VBA macros:

There are essentially two different types of loops in VBA:

1. The "For" loop
2. The "Do" loop

Though there are different flavors of these loops depending on the way the loop is finished, we will present only three versions in this section.

- Go: Tools => Macro => Visual Basic Editor
- Select "Module1" and in the editor type in the following three macros:



```
Dim n As Integer

Sub Loop1()
n = 0
    For n = 0 To 1000
    DoEvents
    Range("C7") = n
    Next
End Sub

Sub Loop2()
n = 0
    Do
    DoEvents
    Range("C7") = n
    n = n + 1
    Loop Until n > 1000
End Sub

Sub Loop3()
n = 0
    Do While n < 1001
    DoEvents
    Range("C7") = n
    n = n + 1
    Loop
End Sub
```

The "Loop1" macro

This is a plain vanilla *"For"* loop with the addition of a *"DoEvents"* statement:

Dim n As Integer => is the declaration of the integer variable "n"

```
Sub Loop1()                 => Indicates the beginning of the macro
n = 0                       =>         initializes the variable n
        For n = 0 To 1000   =>         beginning of the "For" loop
        DoEvents            =>         DoEvents*
        Range("C7") = n     =>         instruction executed during each cycle
        Next                =>         end of the "For" loop
End Sub                     => end of the macro
```

•The "DoEvents" function surrenders execution of the macro so that the operating system can process other events.
• If the "DoEvents" statement were missing the macro would work fine but the chart would not update properly or we wouldn't be able to run other macros such as changing parameters on the spreadsheet during the time the loop is active.

The "Loop2" macro

A "Do" loop can be an infinite type of loop. In the case of the second loop, we will use a finite flavor with an incorporated condition (n>1000) for ending it:

```
Sub Loop2()                 => Indicates the beginning of the macro
n = 0                       => initializes the variable n
    Do                      =>      beginning of the "Do" loop
    DoEvents                =>      DoEvents*
    Range("C7") = n         =>      some instruction executed during each cycle
    n = n + 1               =>      index augmentation condition
    Loop Until n > 1000     =>      end condition of the "Do" loop
End Sub                     => end of the macro
```

•The "DoEvents" function surrenders execution of the macro so that the operating system can process other events.
• If the "DoEvents" statement were missing the macro would work all right but the chart would not update properly or we wouldn't be able to run other macros such as changing parameters on the spreadsheet

The “Loop3” macro

In the case of the third loop, we will use a “Do” loop with an incorporated beginning condition (n<1001) for running it (once this condition is violated the loop stops):

```
Sub Loop3()              => Indicates the beginning of the macro
n = 0                    => initializes the variable n
    Do While n < 1001    =>       beginning of the “Do” loop with the running condition
    DoEvents             =>       DoEvents*
    Range("C7") = n      =>       some instruction executed during each cycle
    n = n + 1            =>       index augmentation condition
    Loop                 =>       end of the “Do” loop
End Sub                  => end of the macro
```

•The “DoEvents” function surrenders execution of the macro so that the operating system can process other events.
• If the “DoEvents” statement were missing the macro would work all right but the chart would not update properly or we wouldn’t be able to run other macros such as changing parameters on the spreadsheet

Let’s associate buttons to these macros and try them out

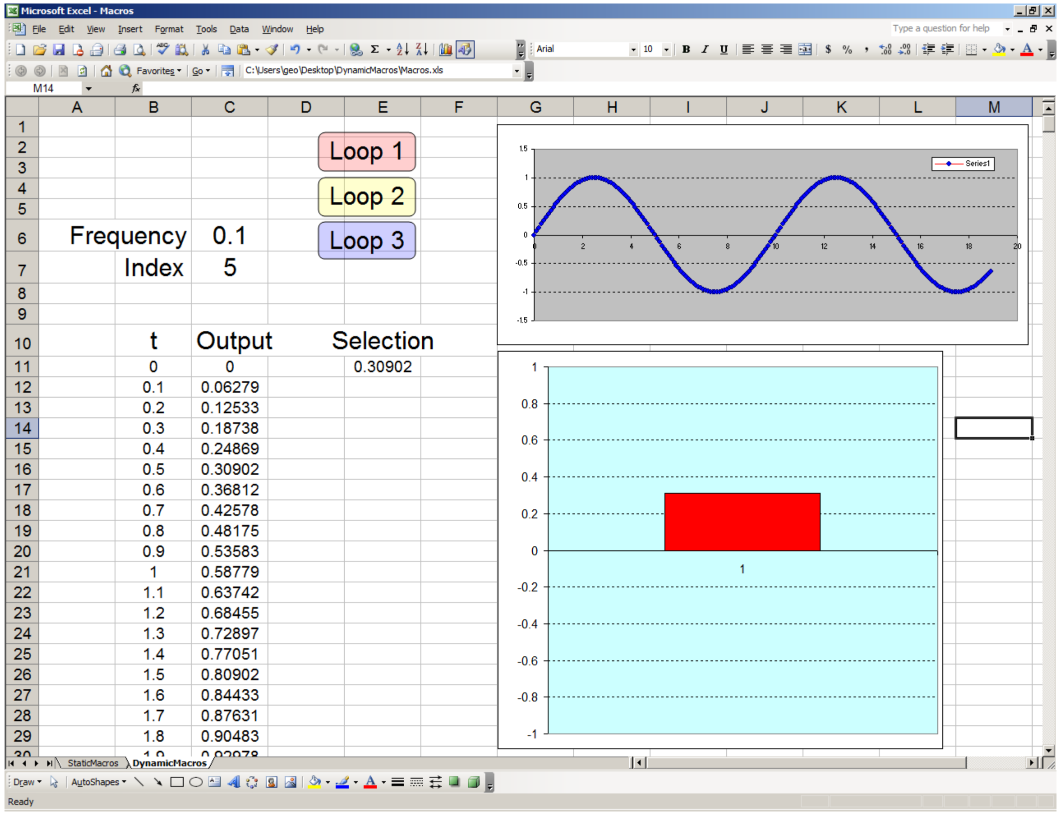
- View => Toolbars => Drawing
- (from the drawing toolbar) AutoShapes => Basic Shapes => Rounded Rectangle (you can use any other shape or a picture)
- Use basic editing to change the colors, transparency, size, etc to your taste and label each of them (write text inside each button).

Assigning macros:

- Right click the “Loop 1” button => select “Assign Macro” => “Loop1”

- Right click the “Loop 2” button => select “Assign Macro” => “Loop2”

- Right click the “Loop 3” button => select “Assign Macro” => “Loop3”


Microsoft Excel - Macros
File Edit View Insert Format Tools Data Window Help
C:\Users\geo\Desktop\DynamicMacros\Macros.xls
M14
Loop 1
Loop 2
Loop 3
Series1
StaticMacros
DynamicMacros
Draw
AutoShapes
Ready

	A	B	C	D	E
6		Frequency	0.1		
7		Index	5		
10		t	Output		Selection
11		0	0		0.30902
12		0.1	0.06279		
13		0.2	0.12533		
14		0.3	0.18738		
15		0.4	0.24869		
16		0.5	0.30902		
17		0.6	0.36812		
18		0.7	0.42578		
19		0.8	0.48175		
20		0.9	0.53583		
21		1	0.58779		
22		1.1	0.63742		
23		1.2	0.68455		
24		1.3	0.72897		
25		1.4	0.77051		
26		1.5	0.80902		
27		1.6	0.84433		
28		1.7	0.87631		
29		1.8	0.90483		

Observations:

1. A macro can be assigned to almost any object in the spreadsheet : a shape, a piece of text, a graph or any type of picture.

2. In order to give a limited amount of protection to a chart we can assign the chart to the following macro:

```
Sub ChartProtect()
Range("E15").Select
End Sub
```

The way this macro operates is by selecting a spreadsheet cell when the chart is highlighted. The macro will “deflect” any attempt to change the graphed data until the macro is “de-assigned” from the chart.

Right click the first chart in the work sheet and assign the macro “ChartProtect()”, and then try to highlight the sinusoidal curve. What happens? It won’t work. You got some weak chart curve protection (against mistakes). You can still move the chart or even resize it by dragging one of its corners.

You can always un-assign any macro from a shape, button, chart or picture by right clicking it and deleting the macro in the “assign window”.

Observations (a fourth loop):

3. An alternative way to exit an infinite “Do” loop is the following:

```
Sub Loop4()
n = 0
   Do
   DoEvents
           If n > 1000 Then
           Exit Do
           Else
           Range("C7") = n
           n = n + 1
           End If
   Loop
End Sub
```

Create another button, assign to it the last macro and verify its functionality. The functionality of the “Loop4()” macro should be identical to the functionality of the previous ones.

Loop 4

The “Start/Pause” and “Reset” - extremely useful macros

Working in engineering for a long time now, I’ve been always aggravated by very expensive modeling packages which, once started, cannot be paused.

Some can be stopped but none that I know can be restarted from the same point. This sucks, especially when a simulation might take hours or days to complete and the software costs between $2K to $200K/seat/year.

Let’s introduce this nice “pausing” feature our models. We would like a Run/Pause macro which can toggle between two states, the “Run” state and the “Pause” state.

If the model is running, this macro should be able to pause it while the spreadsheet keeps at least the previous state before the “pause” command or better even more state points (say 10,000 time steps) of historical data before the pause moment (we’ll show how to do this in a future presentation).

If the model is stopped (paused) running the same macro would restart the simulation from the same point the simulation was stopped a minute, an hour, a month or a 10 years ago.

Of course we also need a “Reset” macro in case we need to start the model from the beginning.

The Syntax:

```
Dim n As Integer
Dim RunSim As Boolean
---------------------------------------
Sub StartStop()

    RunSim = Not (RunSim)

    Do While RunSim = True
    DoEvents
    Range("C7") = n
    n = n + 1
    Loop
End Sub
---------------------------------------
Sub reset()
RunSim = False
n = 0
Range("C7") = 0
End Sub
```

We have to declare a Boolean (logic) variable to monitor the state of the simulation. When *RunSim* = True the simulation is running, otherwise is paused.

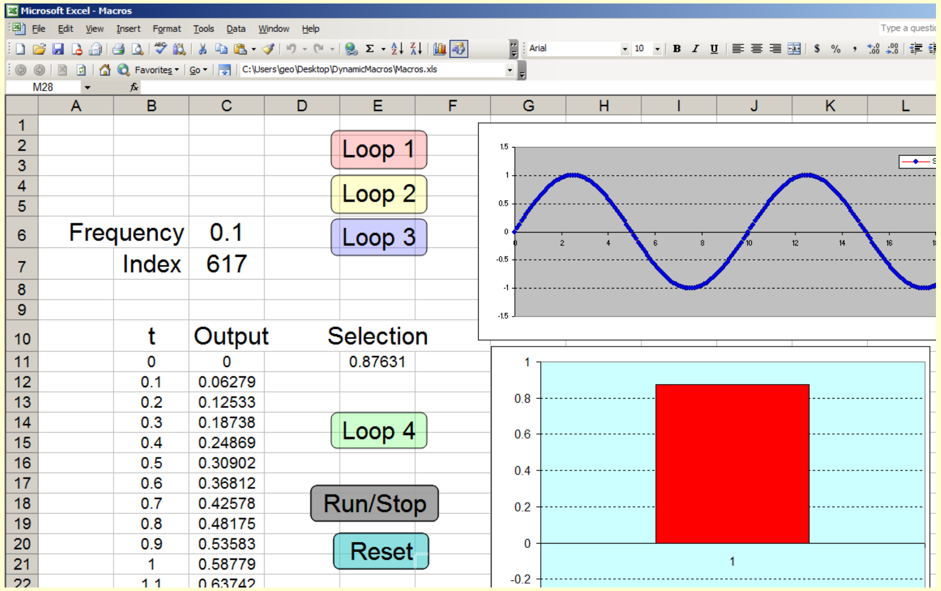
Any time this macro is run it needs to switch the RunSim variable from the existing logic state to the opposite logic state (toggle between True and False).

This is a conditional "Do" loop which would run as long as *RunSim* = True. The loop is watching every cycle and the moment this macro is triggered again, *RunSim* becomes False and the loop (hence the simulation) will stop.

And of course a "Reset" macro is needed in case the simulation needs to be started from square one. This macro turns the RunSim variable into False and resets the index and it's spreadsheet correspondent (C7).

Implementation:

- Two buttons were created from rounded rectangle drawing shapes, then the Run/Pause and Reset macros were assigned to them.
- The macros code is placed in the “Module2” section .
- The reader must check the functionality.


Microsoft Excel - Macros
File Edit View Insert Format Tools Data Window Help
C:\Users\geo\Desktop\DynamicMacros\Macros.xls
M28
Loop 1
Loop 2
Loop 3
Loop 4
Run/Stop
Reset

	A	B	C	D	E
6		Frequency	0.1		
7		Index	617		
10		t	Output		Selection
11		0	0		0.87631
12		0.1	0.06279		
13		0.2	0.12533		
14		0.3	0.18738		
15		0.4	0.24869		
16		0.5	0.30902		
17		0.6	0.36812		
18		0.7	0.42578		
19		0.8	0.48175		
20		0.9	0.53583		
21		1	0.58779		
22		1.1	0.63742		

Conclusions:

- Dynamic macros were created using two different type of loops: “For” and “Do”.
- Two different but equivalent conditional “Do” loops were explored.
- It was shown that macros can be assigned to drawing shapes, pictures, graph, etc.
- A very valuable Run/Pause macro was described and implemented

The reader must pay close attention to this presentation. In order to create science or engineering models one needs these techniques.