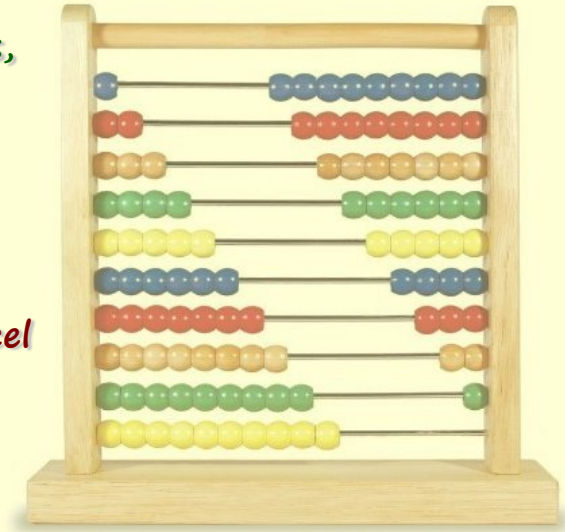
About the speed of animated models in MS Excel – part #1

by George Lungu

- This is a presentation about speed benchmarking for Excel workbooks. Though it focuses mostly on measuring the speed of loop based Excel animated models, some of the principles outlined here can be applied to any standard Excel worksheet.

My personal take on MS Excel:

- For my models I prefer to use older versions of Excel (2003), yet I believe I can bring some value to the "canonic" or "orthodox" Excel users by talking about the factors that affect spreadsheet speed.
- As an engineer I've always loathed expensive fashionable software, software controlled by command line, software that crashes and locks up, software that has no built in graphical display interface, software that has no real time parameter control during the simulation run (most are like this) and software that cannot be stopped, saved and restarted later to simulate from the same point (I haven't met any that would do that yet).
- I also hate software that enforce a slow learning curve and it has too many options. As a conclusion I hate almost any software sold as a black box, usually for a hefty price and often for an additional periodic license fee.
- Besides the all aforementioned drawbacks I see in common packages, what I hate the most is sluggish software. In my industry (electronics design) it can take a hours and sometimes a whole week to model a trivial, Mickey Mouse type of circuit. And this is done on software costing well into six digits licensing fee per person, per year.
- Excel is nothing more than a glorified abacus or a slide ruler. The animated and fully interactive models an average Joe can build in Excel (with a certain personal time effort of course) can fit very particular needs and avoid all the problems mentioned in the first paragraphs.



- I've been using spreadsheets to replace an hour of simulation time with a second of simulation time, for almost no cost (Excel) and most often with a significant but acceptable drop in precision.
- Creating animated models in Excel allows the average Joe to focus on engineering and science rather on syntax and this is because 99% or more of any model can be developed using built in worksheet formulas.

Benchmarking Excel speed – the stopwatch macro "speed_test_1":

- The speeds attained in the modern competitive swimming are well over any estimate made just a few of decades ago.
- This is in because of better coaching helped by two important tools, the stop watch and the video camera. A swimmer can now measure his performance more precisely and can also see himself with his own eyes during practice. This is much more efficient in correcting his swimming technique than listening to coaches alone.
- The first tool we need for improving speed in Excel is a stopwatch.
- Let's look at the following VBA code =>>>>>
- It is essentially an infinite loop which prints in cell B4 the value of the number of loop iterations per second.
-The VBA time function "Timer" retrieves precisely the number of seconds since midnight.
-The variable "Measurement_Start" is initialized at the start of the loop and during each loop cycle the macro will print in cell [B4] the loop iteration speed which is the number of loop cycles divided by the time passed since the start of the loop.

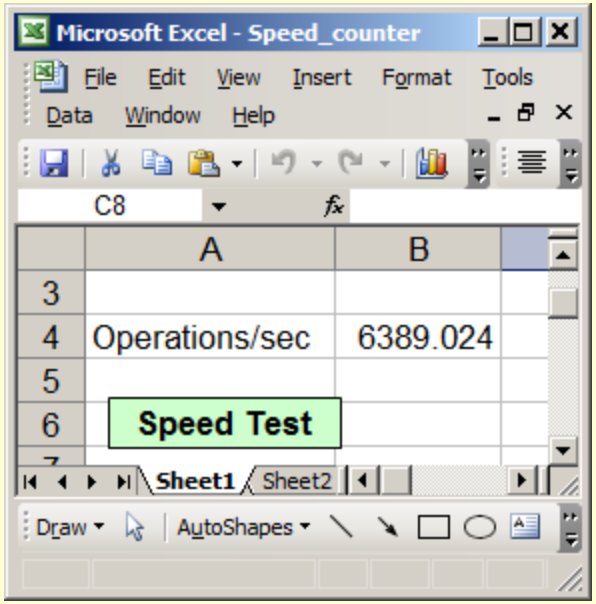
```
Dim p As Double
Dim Measurement_Start As Double

Sub speed_test_1()
p = 0
Measurement_Start = Timer
Do
[B4] = p / (Timer - Measurement_Start + 0.001)
p = p + 1
Loop
End Sub
```

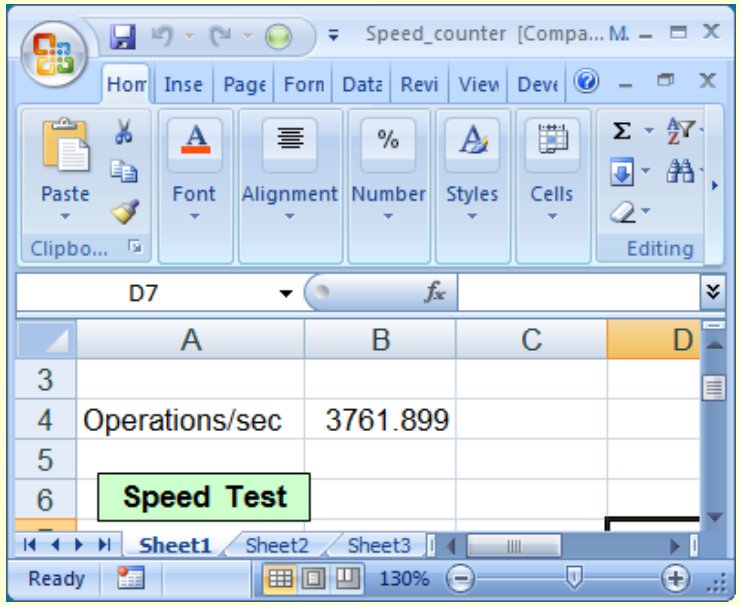
- In the formula for cell [B4] I arbitrarily added the 0.001 residue in the denominator in order to avoid the dreaded division by zero during the first iteration of the macro

Speed results using the macro "speed_test_1()" – small window:

- We used a completely blank workbook with only two cells filled [A4:B4] and one button (text box) to which we assigned the "speed_test_1()" macro. We have no formulas yet in the sheet.
- Start the macro by clicking the green button and let it run 30-60 seconds, then stop it by holding the "Esc" key down (this is an integrative measurement so the longer it runs the more precise the measurement).
- We did this measurement with a small size window to later show the speed loss associated with the window size.



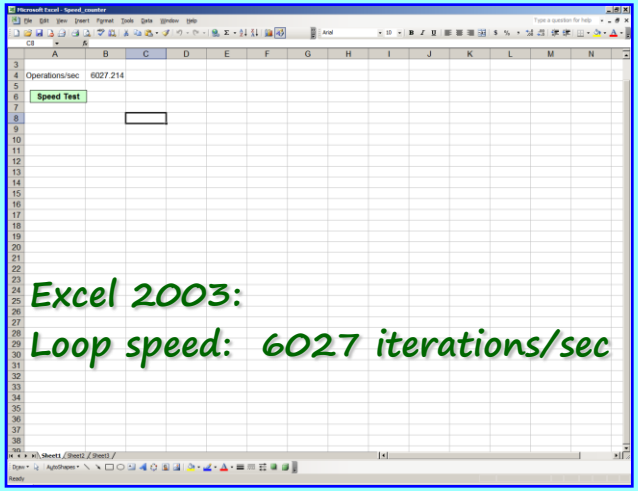
Excel 2003:
Loop speed: 6389 iterations/sec

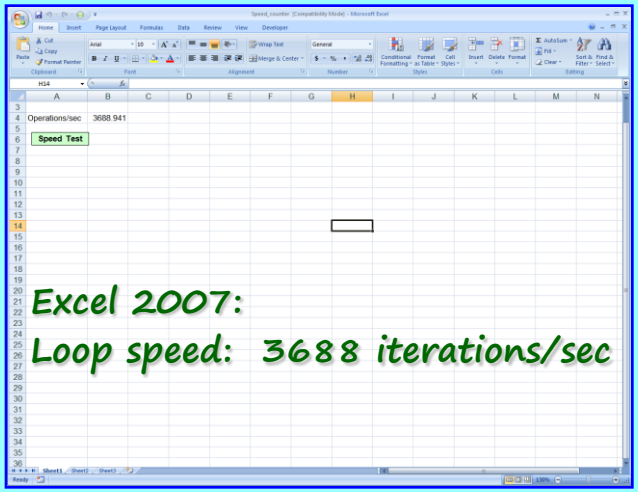


Excel 2007:
Loop speed: 3762 iterations/sec

Lets repeat the measurement with the window expanded

- In this case, due to the lack of data there is minimum speed degradation when the window is maximized.





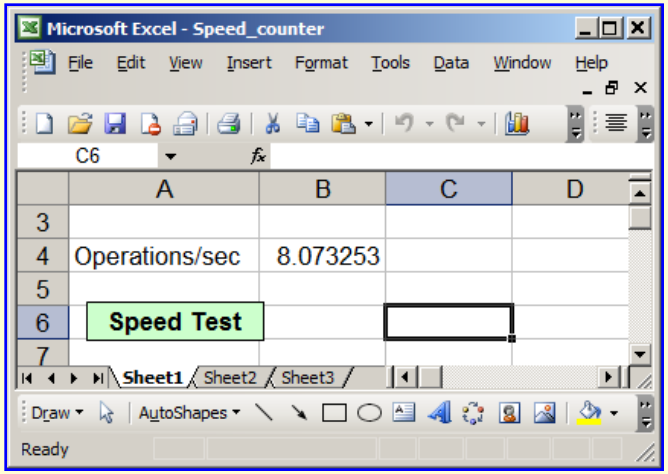
Lets create an array of formulas within the worksheet:

- In order to test the calculation speed on a large number of formulas let's introduce the following two new macros in Module1.
- The first macro will clear the worksheet of any old formulas and the second macro will create a 200 x 5000 array of random number generator formulas (1000,000 formulas), the result of this array being updated during every cycle of the "speed_test_1()" macro.

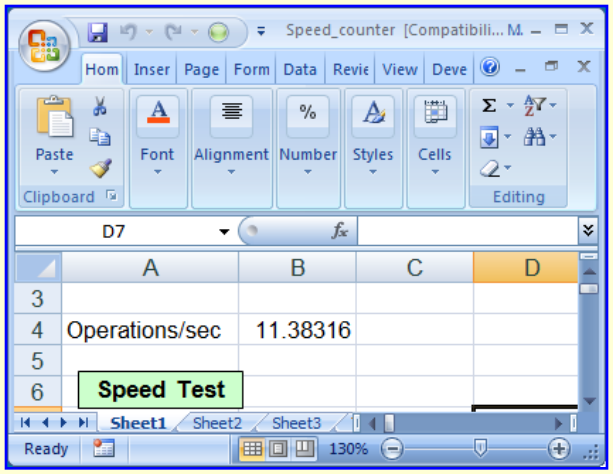
```
Sub clear_array()
[A10:IV65536].Clear
End Sub

Sub generate_array()
[A11:GR5010].Formula = "=rand()"
End Sub
```

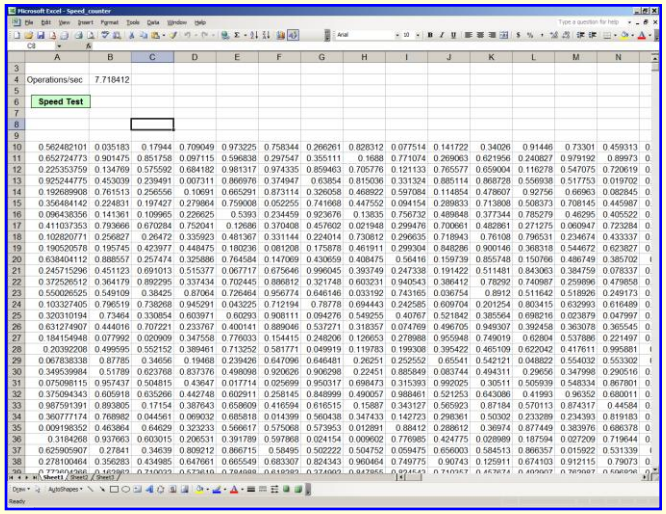
- The speed results for calculating an array of 1000,000 numbers using the "rand()" built-in function is shown below:



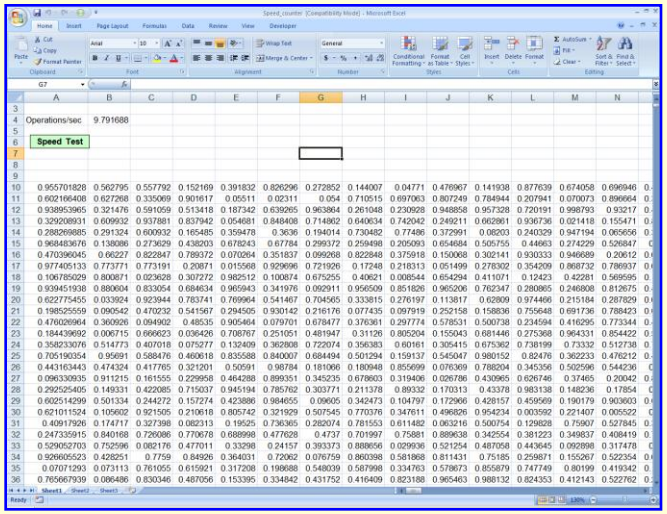
Excel 2003 – small window:
Loop speed: 8.07 iterations/sec



Excel 2007 – small window:
Loop speed: 11.38 iterations/sec



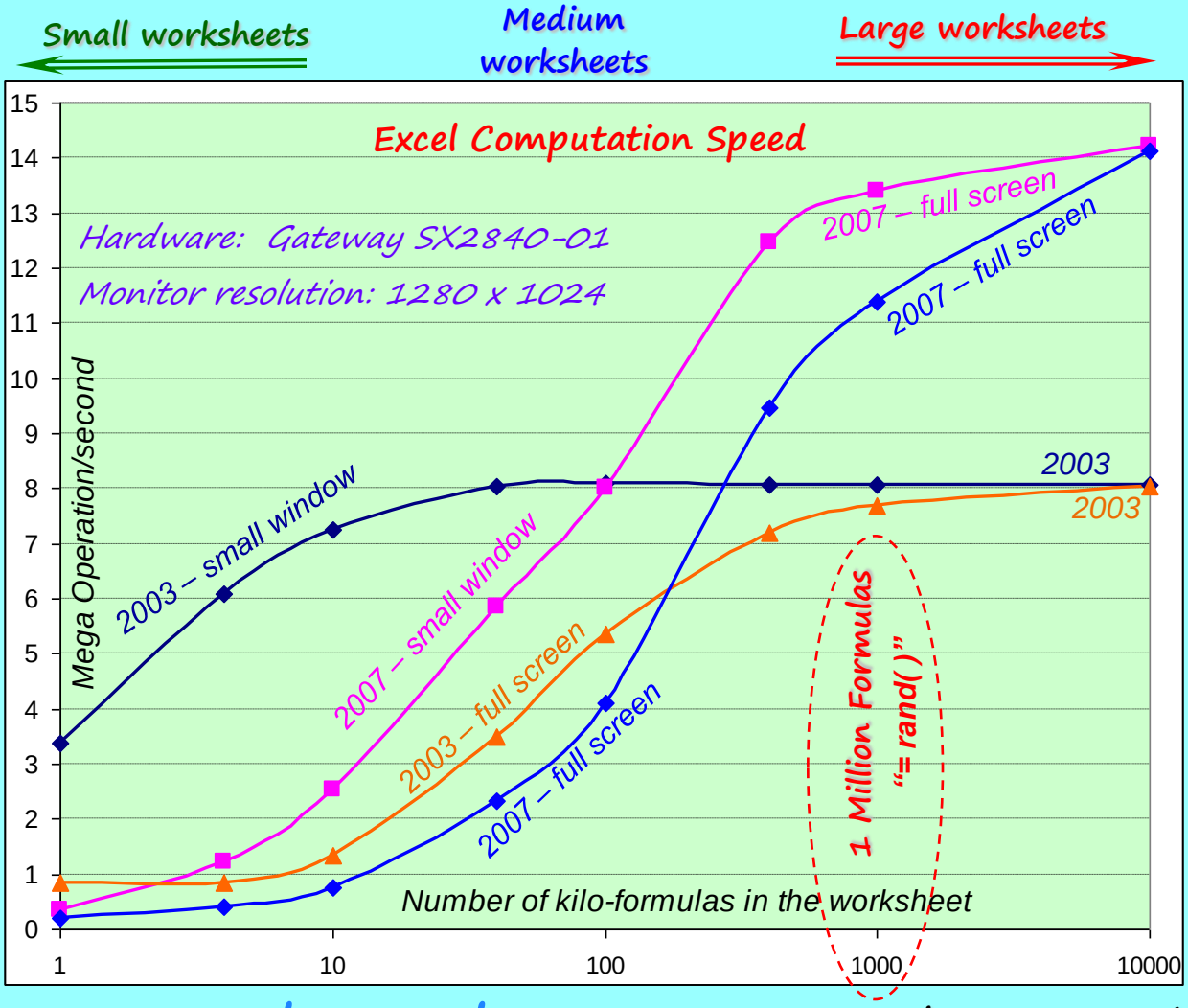
Excel 2003 – large window:
Loop speed: 7.71 iterations/sec



Excel 2007 – large window:
Loop speed: 9.79 iterations/sec

How does the speed scale with the number of formulas?

– In this section, the previous test is being run on both, a reduced size window and a full screen window for the following series of formula sizes: 1000 formulas, 10,000, 100,000, 1000,000 and 10 million formulas. To the right you can see the various macros used for generating the arrays of formulas.




```
Sub generate_array_1K()
[A10:GR65536].Clear
[A11:GR15].Formula = "=rand()"
End Sub

Sub generate_array_4k()
[A10:GR65536].Clear
[A11:GR30].Formula = "=rand()"
End Sub

Sub generate_array_10k()
[A10:GR65536].Clear
[A11:GR60].Formula = "=rand()"
End Sub

Sub generate_array_40k()
[A10:GR65536].Clear
[A11:GR210].Formula = "=rand()"
End Sub

Sub generate_array_100k()
[A10:GR65536].Clear
[A11:GR510].Formula = "=rand()"
End Sub

Sub generate_array_400k()
[A10:GR65536].Clear
[A11:GR2010].Formula = "=rand()"
End Sub

Sub generate_array_1M()
[A10:GR65536].Clear
[A11:GR5010].Formula = "=rand()"
End Sub

Sub generate_array_10M()
[A10:GR65536].Clear
[A11:GR50010].Formula = "=rand()"
End Sub
```