Macros and VBA in Excel: Part 2

American Society for Engineering Education

November 11th 2013

Download this File

- umich.edu/~pruchser
  - ASEE Example Spreadsheet
  - Once open in Excel
    - Enable Editing

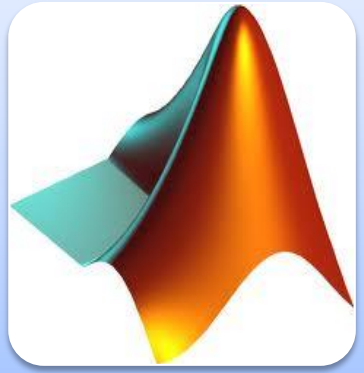
Intro

- Philipp Ruchser
- Grad Student on Exchange (EECS, IOE)
- Experience in Excel & VBA trough an internship in a financial consulting & software company
- Feel free to stop me at any point to ask any questions
- Other people walking around can also help with any problems you may run into

Excel vs. Matlab

Excel

- Great for visualizing data/processes
- Useful for designing a functional prototype
- Ubiquitous - companies have it!



Matlab

- Better for large data sets
- More built in math functions (matrix inversion/ decomposition, Laplace transforms, differential eqns., etc)

Overview

- What is VBA?
- Basic language introduction
- Simulating π using Excel & VBA
- (more) advanced hints
- Survey

What is VBA?

- Visual Basic for Applications
- Based on Visual Basic, it extends the functionality and flexibility of MS Office programs by combining
  - VB functionality
  - Host program (Excel, Word, …) functionality
- Write customized functions and procedures, create a GUI-like Excel-Sheet, control random numbers, …

Language characteristics

- Simple, intuitive
- Well-documented online
- Variable type “variant” (do not even have to be declared)
- Similar control structures to other programming languages (-> Google, MSDN)
- This simplicity comes with the downside of VBA being fault-prone and slow

Miscellaneous

- If you have a problem or question, I highly encourage you to try first to find your answer with Google (MSDN, Stackoverflow)
- VBA induced changes to an Excel Sheet CANNOT be undone with Ctrl-Z, be mindful of this

First VBA function: n!

- n! = 1*2*3*…*(n-1)*n

```
factorial = 1
while n > 1
     factorial := factorial * n
     n := n – 1
end while
return factorial
```

First VBA function: n!

- n! = 1*2*3*…*(n-1)*n

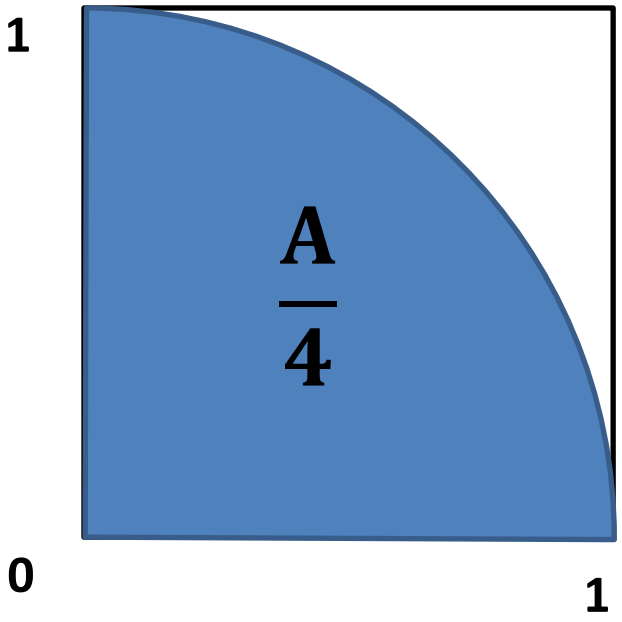
```
Function VBAFactorial(n)

    result = 1

    Do While n > 1
        result = result * n
        n = n - 1
    Loop

    VBAFactorial = result

End Function
```

Simulating π

- Monte Carlo Simulations exploit the Law of Large Numbers
- By repeatedly simulating under similar conditions, the obtained, average result approaches the expected value
- Thus, need random numbers
- VBA very useful when dealing with random numbers in Excel **(control when they update)**

Simulating π

- $A = \pi\, r^2$
- $\pi = A$ in a unit circle with r = 1
- $\frac{\pi}{4} = \frac{A}{4}$ in a **quarter** unit circle with r = 1

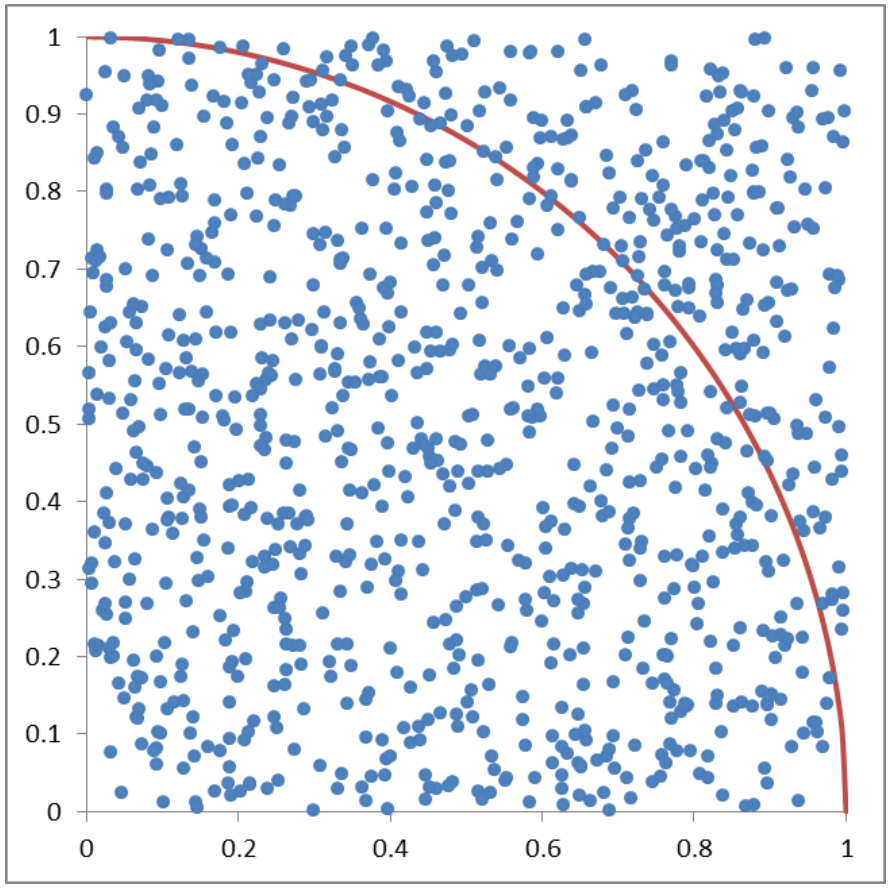


Simulating π

- We simulate uniformly and independently distributed random numbers x any y on the interval [0, 1]
- Check whether a point p = (x,y) is inside the unit circle ($x^2 + y^2 \leq 1$)
  - Yes: inCircle(p) = 1, No: inCircle(p)= 0
- Repeat this for many random points
- Estimate $\hat{A} = \hat{\pi} = \frac{\sum_p inCircle(p)}{\#\ scenarios} * 4$

Simulating π

- $P(p \in unitCircleFragment) = A/4 = \pi/4$



Simulating π – Batch mode

- Increase the quality of our estimate by repeatedly estimating π n-times and computing the average
- The standard error of our π estimate decreases (roughly) by factor $\frac{1}{\sqrt{n}}$

Simulating π – Batch mode

Pseudo-Code:

```
For i = 1 to n
    Update Random Numbers
    Copy π and i to the target sheet
Next i
Update Target Sheet
```

Necessary code fragments

- Addressing the Value of cell B1 on the Sheet “Simulation”

```
Worksheets("Simulation").Cells(2,1).Value
```

OR

```
Worksheets("Simulation").Range(“B1”).Value
```

- Addressing the Value of one/multiple cell(s) previously assigned the name “pi” on the Sheet “Simulation”

```
Worksheets("Simulation").Range(“pi”).Value
```

- Assign this value to a variable

```
pi = Worksheets("Simulation").Range(“pi”).Value
```

Simulating π

```
Sub PiBatch()

    Application.ScreenUpdating = False

    limit = Range("C2").Value

    For i = 1 To limit
        Calculate
        pi_temp = Worksheets("Simulation").Range("Pi").Value
        Worksheets("Batch").Cells(4 + i, 1).Value = i
        Worksheets("Batch").Cells(4 + i, 2).Value = pi_temp

        'Advanced extensions
        DoEvents
        Application.StatusBar = "Simulation run " & i & " of " & limit

    Next i
    Calculate
    Application.ScreenUpdating = True

End Sub
```

Advanced hints

- **`DoEvents`** (in a loop) prevents a complex VBA application from freezing Excel
- **`Application.ScreenUpdating = False`** significantly speeds up calculations by running computation in background
- **`Application.StatusBar = "String"`** allows to control the Excel status bar via VBA code
- **ESC** terminates VBA execution

Survey

- Please fill out the survey at the end to let us know what we did well and what we could have done better
- Responses are greatly appreciated, and they will help us make future sessions better
- If there is anything you would like to see in Part II of this workshop, please let us know here