

## Notes on Visual Basic for Applications and Monte Carlos

### 1. Introduction

One reality of Microsoft's dominance of the PC software environment is that the world's current most popular programming language is Visual Basic (VB). VB is loosely modeled after the BASIC language that has existed for many years, but it also incorporates many of the object-oriented features of a modern programming language, such as those that are found in Java or C++. If you are familiar with any programming language, you should find VB mostly straightforward. If you are not familiar with any programming language, you may find VB a good way to start.

A dialect of VB known as VBA or Visual Basic for Applications is the scripting language in many Microsoft products, including Excel and Word. A scripting language lets you automate these programs, so that one command can invoke a whole series of commands, complete with conditionals, iterations, etc. Here we will only begin this vast topic, focusing on adding user functions to Microsoft Excel.

### 2. Excel User Functions

The standard version of Microsoft Excel is equipped with a wide range of functions and procedures. The available functions can be viewed by using the **Function...** command under the **Insert** menu. Here you can look at either the entire list of functions or sets of functions grouped by categories. If you are not sure from the name what a function does, you can click on the question mark in the lower left corner of the dialog box and can help. When you select a function, Excel then follows with a dialog box that suggests how to use it. Excel also provides a set of data analysis procedures through the **Data Analysis...** option at the bottom of the **Tools** menu. (If this choice does not appear, you may need to run the Excel installation procedure again and explicitly choose to install the Data Analysis tools.)

You can vastly increase the power of Excel by use of VBA programming. Excel provides several ways to implement VBA code, all of which start with the **Macro** command under the **Tools** menu. One option is to record a Macro, which means that you click on a start button, do a set of steps and then click on a stop button. Excel keeps track of what you do while recording the Macro and converts these steps into VBA code and records them. You then get to assign a name to this macro. When you invoke the macro, Excel repeats the steps you used in defining the macro, albeit with different data.

Here, we will take a different approach to VBA programming than defining macros. We will write explicit code to define user functions. We can then implement these user functions in our Excel spreadsheet, in the same way we use built-in Excel functions. Choose the **Macro** command under the **Tools** menu and then for the sub-command, choose **Visual Basic Editor**. You will now get a "Project Window" that will show you various components of your Excel software. This window is probably useful for something but I am not sure what. Now choose the **Module** command under the **Insert** menu. The VBA editor will open a window. This is where you write your VBA code.

As a start, try the following code:

```
Public Function cube (ByRef x As Double) As Double
    Cube=x*x*x
End Function
```

As you can probably guess, this function calculates the cube of a number and returns the calculated value as the value of the function. Now under the **File** menu, choose the option to **Close** the VBA editor or just leave the editor open and select your spreadsheet. Put a number in a cell A1 and hit **Return**. In cell B1, type "=-cube(a1)" followed by a **Return**. (*Don't type the quote marks.*) You should now see in cell B1 the cube of the number in A1. Your first VBA program has now worked.

Several of the keywords in your function could actually have been left out and VBA would have included them by default. (Unlike C++, VB is a weakly-typed language.) The keyword "Public" means that your function cube will be available throughout the Excel program. The alternative is to declare a function "Private", which means that it is only accessible in the current class module. We will declare all of our functions "Public." The keyword "ByRef" means that we want VBA to pass the argument to the function "by reference" as opposed to the alternative of passing "ByVal" meaning "by value." Passing arguments by reference has three advantages. It can be used for any kind of argument. It is faster. It allows a value calculated in the function to be passed back to the spreadsheet through an argument in addition to the value passed through the function return. The disadvantage of passing arguments by reference is that it violates the good software practice called "encapsulation" and thus results in software that is more prone to error.

The two instances of the keywords "As Double" mean that both the argument x and the value of the function cube are in the form of double-precision, floating point numbers. VBA has other types of variables, including integer, string, float, date and variant. Double is the default type and almost all of our variables will be of type double.

### 3. Monte Carlo Calculations

Monte Carlo calculations were developed during World War II and their use has vastly increased since that time because of the exponential growth in computing capacity. A Monte Carlo calculation simulates a physical process in which individual steps are only known stochastically. By stochastically, I mean that the distribution is known, but the precise answer is not known in advance.

Consider the following example: An inebriated person stands at a light post. This person then takes a series of 10 steps. Each step has an equal probability of going to the left or to the right. Each step is exactly 0.5 m in length. How far is this person from the light post after 10 steps? Try the following VBA code:

```
Public Function randomstep() As Double
    Dim step As Double
    Dim i As Integer
    step = 0
    For i = 1 To 10
        x = Rnd()
        If (x < 0.5) Then
            step = step + 1
        Else
            step = step - 1
        End If
    Next i
    randomstep = step
End Function
```

Every time you call the randomstep function, you get run another simulation of the person taking 10 random steps away from a light post. Now use the **Fill Down** command under the **Edit** menu to run the simulation 100 times. What fraction of the time is the answer 0? 1? 10? Use the **Histogram** procedure called by selecting **Data Analysis** under the **Tools** menu to make a table of your results. Then you Excel's graphing capability to turn your table into a column graph. What function best describes your histogram?

### 4. Generating Distributions

The book *Numerical Recipes* describes algorithms for calculating various distributions of random numbers for Monte Carlo programs. In the program above, we have used the built-in VBA function rnd() to calculate a uniformly distributed random number between 0 and 1. Although there are more efficient methods for some distributions, using the so-called rejection algorithm, rnd() can be used to calculate random numbers distributed in almost any way. In the rejection method, we use rnd() to choose a random number. We then calculate the probability that this number is associated with our distribution and then use rnd() again to decide whether to keep the number we just chose. The following program returns an exponentially-distributed random number.

```
Public Function exprand() As Double
    Dim x As Double
    Dim y As Double
    For i = 1 To 1000
        x = 10 * Rnd()
        If (Rnd() <= Exp(-x)) Then
            exprand = x
            GoTo a
        End If
    Next i
    exprand = -1
a: End Function
```

### 5. Lorentz Transformation Simulation

The following problem will be due **September 28**. Assume that a Z intermediate vector boson decays isotropically in its rest frame into two muons. The mass of the Z is 91 GeV. The mass of a muon is 105 MeV. In these same units, the mass of a proton is 938 MeV. Assume that the gamma of the Z in the lab frame is 10. Write a MC program in VBA or some other language to simulate the decay of the Z. Run 100 simulations. Make a histogram table and bar chart of the decay distribution in the LAB frame as a function of the angle from the direction the Z is travelling in the LAB.