Derivative Securities, Courant Institute, Fall 2006
Assignment 7, due November 7 [Updated Nov 3 to fix typos]

Important: Always check the class bboard on the blackboard site from home.nyu.edu (click on academics, then on Derivative Securities) before doing any work on the assignment. In particular, some questions may be deleted or added depending on how much material we cover in class.
Especially important: If you are new to Excel and Visual Basic, use the bboard for help. Many people in the class are true experts. Otherwise you can waste hours or days trying to figure something out from the meager documentation Microsoft and others provide. Please understand that I am not an expert so some of the information or suggestions in the files I post might be incorrect or naive. If you are an expert and think I've done something wrong, please say so in the bboard.

The web site has a file `homework7.xls` and another `homework7VBA`. The `.xls` file is a Microsoft Excel spreadsheet that includes macros `ForwardEuler` and `BlackScholesEuropeanPut`. The `homework7VBA` file has the text for these macros in ASCII, just in case you can't open the spreadsheet macros for security reasons. I promise that I did not knowingly put anything dangerous into the macros.

To edit a VBA macro (another word for program), you open Excel, then click on "tools", then "macro", then "VBA editor". If you've opened `homework7.xls`, the macros will be in `module1` and will appear if you click on `module1` somewhere in the lower left of the window. If you want to put the result of a macro calculation, say `BlackScholesEuropeanPut`, into, say, cell `C6`, you click on cell `C6`, then select "insert", then "function" from the toolbar on the top of the spreadsheet (which is not the toolbar on the top of the VBA editor). You will get a dialogue box full of functions. Click on "user defined" and scroll through the possibilities until you find `BlackScholesEuropeanPut` and select it. I did this on two versions of Excel, one on a Windows box and one on a Mac. One gave me a huge list of choices and the other gave me just the two I had defined. You'll get another dialogue box that asks you for values of the arguments. These can be either cells (preferred) or hard numbers. Enter all the values and click OK and the macro should run, with the result appearing in cell `C6`.

The macro `BlackScholesEuropeanPut` illustrates how to access the Excel function `NormSDist` fro VBA. Of course, the real Black-Scholes formula is not $N(S0)$, but you should have no trouble putting in the right thing. The keyword `Public` in front of the definition makes the function visible to Excel. Calling it `Private` would make it visible only within the VBA editor. Do this for functions you don't want the user to see. The underscore symbol, _, means "continuation", so that the VBA statement continues on the next line. Do this for readability so that all the arguments appear in a column. The `ByVal` in front of each argument means that it is passed from the spreadsheet to the function *by value* as opposed to *by reference* (which is `ByRef` in VBA). This is the way arguments are passed by default in C++, copying the value from the spreadsheet to the place where VBA stores its variables before the function

executes. It insures that if you change the value of an argument, the value in the spreadsheet cell it came from will not change. Use `ByRef` if you want to modify a cell value (usually a bad idea). This is done in C++ by passing the address of the variable to a procedure. The `As Double` means that the type of, say $S0$, will be what C++ calls `double`, a double precision floating point number. VBA does not force you to give the types of all variables, but it is a good idea to do so. For example, if `U` is an integer and `V` is a double, then `V = 3.4; U = V; V = U` will give $V = 3$.

The `Dim f(-200 To 200, 200)` in `ForwardEuler` defines an array with 401 possible j values and 200 possible n values (80,200 values in all). The `For n = periods To 2 Step -1` starts a *for loop* that executes the body of the loop for n starting at `periods` and going down by one until it reaches 2. The end of the loop body is the line `Next n`. You also can see the syntax of *if then else*.
Discussion: This is not a programming class but getting actual numbers from pricing models always involves at least a little programming. If you're new to programming, please pay attention to programming style. Creating a software system is like arranging a kitchen. Putting things in good places makes it easier to cook. You can waste lots of time rummaging around in poorly written or organized software systems, even if they in principle are correct.

1. Complete and correct the function `BlackScholesEuropeanPut` by putting in the correct Black Scholes formula. Check that it gives the same answers you got in homework 6. *Discussion*: It can be more convenient to use a macro than to put the formula into a spreadsheet directly. In future assignments, you will create a suite of functions like this one that compute greeks and implied vol. It will be easier to find them if you can look them up in a list where they are labeled by name than by clicking around in a spreadsheet looking for a cell with the formula you want. The purpose of `BlackScholesEuropeanPut` right now is to verify the correctness of the `ForwardEuler` code in part 2. A code is not finished until it has been verified in some way and *stress tested* (run on difficult problems).

2. Complete and correct the function `ForwardEuler` that values a European style put with given parameters using the trinomial tree/forward Euler method described in the book by Hull on page 425 (and nearby). Note that α, β, and γ (called a, b, and c in `ForwardEuler`) do not depend on j. Compare the output to that from the exact Black Scholes formula from part 1 as the number of periods is increased from 10 to 200. Use the parameters from Homework 6, part 2a. *Discussion*: Programming languages have different ways to express the same things. If you have experience with one, the next one is simple. Just learn how to say *for* and *if* and how to name procedures (functions, subroutines). The `ForwardEuler` code above leaves much to be desired. It hard wires in the maximum number of periods, 200. It has no error checking to protect the user from running it with 300 periods, which would make the program *crash* (fail in an uncontrolled way). I have not learned how to fix these problems in VBA, but hope to soon.

3. Create a new function `FEAP` (for Forward Euler American Put) that has the structure of `ForwardEuler` but takes into account the possibility of early exercise. Check that the answer is accurate with enough periods by checking that it is consistent. We have no formula in this case to check it against. Use the parameters from Homework 6, part 2a. *Discussion*: We were able to check that `ForwardEuler` was correct by checking against the Black Scholes formula. We hope `FEAP` because it is a small change from `ForwardEuler`. We don't know how many periods it will take to get an accurate answer, but we can guess by comparing results as we increase the number of periods and seeing when they stabilize.

4. Make a plot of the American option price similar to the European option price from Homework 6 part 2g. Make the plot accurate enough to illustrate the smooth pasting condition at the early exercise boundary. *Discussion*: It is annoying that `FEAP` returns a single number at a time. If we had thought ahead, we could have written it to give the whole curve in a single call. I hope to learn how to do that by next week, but many pricers in use work like our `FEAP`. You may notice that the computer takes a while to draw the whole curve, particularly if it has many values. Think about how much work is involved.