VB Programming Tips
AGEC 637 - Summer 2013

1. **Making your program run faster.** As your program grows, you will find that it slows down. At some point you’ll want to stop Excel from doing expensive work while your program is running. You can do this by putting the command
   at the beginning of your program, then at the end of the program adding
   Application.Calculation = xlCalculationAutomatic
   Finally, if you are interacting with your spreadsheet a lot, you may want to tell the program to not update the screen while during your program:
   Application.ScreenUpdating = False

2. **Reading in a Matrix.** Suppose you’ve got a range in your spread sheet called “X,” and this contains a matrix that you want to use in your analysis. You can read this directly into your code without explicitly defining the size of the VB array as follows
   ```vba
   ' ------------------------------------------------- --------------------------------------
   ' Dimension the array and variables indicating the size of the matrix
   ' ------------------------------------------------- --------------------------------------
   Dim xMatrix(), RowsX As Integer, ColsX As Integer
   ' ---------------------------------------------------------------------------------------
   ' Read in the matrix, then count the number of rows & columns
   ' ------------------------------------------------- --------------------------------------
   xMatrix = Range("X")
   RowsX = UBound(xMatrix, 1)
   ColsX = UBound(xMatrix, 2)
   ```

3. **Clearing A Range.** In the code below I clear a range for the V* and z* arrays and insert the necessary names at the correct places so that they don’t write over each other. The only named cell required in the spreadsheet is one called “OutputRange,” below which there is no data. This could be put into the InitializeValues subroutine.
   ```vba
   ' ------------------------------------------------- ------------------------------
   ' Clear the 100 by 100 array below the range "OutputRange"
   ' ------------------------------------------------- ------------------------------
   With Range("OutputRange")
       .OffSet(1, 0).Range("OutputRange").OffSet(100, 100).ClearContents
   End With
   ' ------------------------------------------------- ------------------------------
   ' Name the Vstar array in the spreadsheet, write its title and the initial values
   ' ------------------------------------------------- ------------------------------
   .Offset(2, 0).Name = "ValueFunction"  ' names the range
   .Offset(-1, 0) = "Value Function"      ' writes the title
   ' ------------------------------------------------- ------------------------------
   ' Name the Zstar array in the spreadsheet and write the title, leaving 3 lines
   ' after the value function array before writing the zstar output
   ' ------------------------------------------------- ------------------------------
   .Offset(nx + 3, 0).Name = "Zstar"      ' names the range
   .Offset(-1, 0) = "Zstar"              ' writes the title
4. **Renaming a Module.** If you want to give your module a better name than, “Module1” you can do so by viewing the “properties window” (commands: View Properties Window, or just F4). You can then click on the name of the module, and simply type the new name there.

5. **Using a Timer.** VB has a built in time function that can be used to store the run time of a program. If you put the following 2 lines at the start of the program

   ```vbnet
   Dim TimerVariable
   TimerVariable = Timer
   ```

   And then this line at the end

   ```vbnet
   TimerVariable = Timer – TimerVariable
   ```

   The TimerVariable will contain the number of seconds. This could be saved in your spreadsheet.

6. **Quickly moving between subs.** With the ctrl key held down, the up and down arrows will jump to the next or previous program unit (i.e., sub or declaration section).

7. **A stochastic simulation.** Suppose that you have a state transition that is stochastic, the variable, x, can take on values $i_x = 1$ to $n_x$, and these values are stored in the variable $x_{grid}$. The probability of ending up in each of these $n_x$ values is defined by elements of the Markov transition matrix, $MTM$ which is an $n_x$ by $n_x$ matrix. A random path over $n_t$ periods can be simulated using the following lines of code:

   ```vbnet
   ix = 3  ' An initial value for the index must be set
   For it = 1 To nt
   ' Store the value of the current x in spreadsheet
   Cells(it, 1) = it
   Cells(it, 2) = xgrid(ix)
   ' Generate a random #
   ' Rnd must not be dimensioned as a variable. It is a special VB function
   ' that generates a uniformly distributed random # between zero and 1.
   eps = Rnd
   ' Figure out which cell was chosen
   cdf = 0#
   For i = 1 To nx
   cdf = cdf + mtm(ix, i)
   If cdf > eps Then
   ix = i         ' update the value for ix
   Exit For
   End If
   Next i
   Next it
   ```

8. **Using the Solver from VB**

   Excel (with the necessary add ins) actually has a hill-climbing algorithm which, though slow, does a pretty good job. Automating your code to use it, however, is a bit tricky (it took me years to figure this out).
a) First you have to activate references to solver code. In VB use the Tools menu, click on References. Find “Solver” in the list and click the associated check box.
b) Here are some example uses of some solver commands:
   ' This resets the solver's settings
   SolverReset
   ' To set the optimization criterion. In this case we're maximizing the \( i^{th} \) element
   ' of the range \( v_{lhs} \) by changing the \( i^{th} \) element of \( z_{star} \)
   SolverOk SetCell:=Range("v_{lhs}"),Cells(ix, 1), MaxMinVal:=1, ValueOf:="0", __
   ByChange:=Range("z_{star}"),Cells(ix, 1)
   ' To add constraints use the following. Relation = 1 is \( \leq \); =2 is \( = \); 3 is \( \geq \)
   SolverAdd CellRef:=Range("z_{star}"),Cells(ix, 1), Relation:=1, FormulaText:="z_{max}"
   SolverAdd CellRef:=Range("z_{star}"),Cells(ix, 1), Relation:=3, FormulaText:="z_{min}"
   ' To minimize the MaxMinVal would be set to 2 and to get as close as possible to a particular
   ' number you would set it at 3, at which point the ValueOf statement would become active.
   ' When everything is set, use the command
   SolverSolve
   ' if you want to be prompted to accept or reject the solver's results use
   SolverSolve UserFinish:=True

9. Turning off Alerts
   If your VB code closes a file, Excel will ask you if you really want to do this. To turn
   off this alert, use the code:
   Application.DisplayAlerts = False
   Then, to turn the alerts back on use
   Application.DisplayAlerts = True

10. Running the random number generator with a seed
    If you are doing any kind of simulation modeling, it is frequently helpful to be able to
draw the same series of random numbers. To do this, you need to grab the same
"seed" each time you start your series of random draws. To do this, you simply put a
negative number into the parentheses in the random number function. For example
\( \text{Rnd}(-19867386) \) will always yield 0.531279 and then the first three times you call
\( \text{Rnd}() \), it will always yield 0.925858, 0.293331, 0.974786.