**Deterministic Modeling for Biosystems (BSEN 5250/6250)**

**Laboratory 1**

**Introduction to Visual Basic for Application:**

**Case Study of Diffusion**

**Introduction and Theory**

The diffusion of many molecules into a cell is characterized as a passive diffusion process. The concentration gradient of the molecule inside and outside of the cell membrane constitutes the factor driving the diffusion process. The equation governing this process is

 (1)

Where

Ci is the concentration of the molecule in the cell, moles/liter

Co is the concentration of the molecule outside the cell, moles/liter

k is the diffusion constant, 1/minute

This equation can be solved analytically by integrating the differential equation, which yields the following:

 (2)

Where

Ci(t) = concentration inside the cell at time t, moles/liter

t = time, minutes

Alternatively, Equation 1 can be solved using Euler’s method by:

 (3)

Substituting Eq. 1 in for dC/dt gives

 (4)

**Objectives of the Laboratory**

The objective of this laboratory is to develop a spreadsheet to compute the diffusion of a molecule into a cell over time. You will develop a Visual Basic for Applications (VBA) program to solve this problem using Euler’s method outlined in Equation 4.

**Procedures**

***Note: There is a file in Canvas called \Labs\VBA.DOXS that may be helpful in learning VBA programming.***

**Step 1:** All files required for this Lab are stored on Canvas under the \Files\Labs\Lab1\ Directory.

**Step 2:** A prototype spreadsheet for this lab can be found on Canvas Files in the the \Labs\Lab 1\Lab 1.xlsm. Please open the spreadsheet and copy it into computer.

**Step 3:** Study the spreadsheet carefully. It contains a table of inputs and initial conditions for this problem. It also contains the calculation of the concentration inside the cell (Ci) using the Euler method (solved within the spreadsheet cells) and analytically. The percent error column computes the error between the two methods. Change some of the inputs and watch the graph change.

**Step 4:** Column E contains a series of blank cells. This is where you write the results of your solution of Equation 4 using a VBA program that you create. You are now ready to create your VBA program.

**Step 5:** Now, let’s define a group of cells, also known as an array of cells that we can use to write model results into the spreadsheet. We will call this group of cells ***Concentration***. To define ***Concentration***, go to the menu at the top of the screen and click on the following sequence: ***Formulas >Define Names.*** Click the mouse in the top blank, and type in ***Concentration***. This defines the name of this group of cells. Click in the bottom blank, delete the default cell reference that shows up, and then use the mouse to highlight the cells on the spreadsheet you want to be referred to as ***Concentration.*** I would suggest selecting cells E14:E100. You can always modify this range in the future using the same menu.

**Step 6:** It is now time to create a visual basic program to compute the diffusion of a molecule into a cell. From the menu, select the following***: Developer >Visual Basic***. You are now in the VBA editor. On the left, you should have a blue highlight on the Sheet 1 Object. The program you type to the right will be associated with Sheet 1 of the workbook. You can rename the sheet by clicking on the sheet name in Excel and typing in a new name. From the VBA menu, type in the commands

Sub {***your program name***} ()

This will create the beginnings of your program.

**Step 7:** Now, type in the program. A description is outlined below. The actual code you should write is in bold, and an explanation of the command or code follows.

**Sub concentration()**

The Sub command defines a subroutine, or program in VBA. We are calling this subroutine “***Concentration***”, which will be defined as a Macro in Excel. The parentheses () at the end of the command means we are not passing any parameters into the subroutine. Note that when you type in the **Sub** command, the **End Sub** command will show up by default. Don’t erase this!

The DIM command is how you dimension a variable in VBA. It stands for Dimension, where you declare what a variable is and how it will be used. Double stands for a double-precision real number, integer is an integer, etc. Type in the following code to declare the variables that you will need for this program. Note that you can add comments anywhere in your program by using the apostrophe **‘** followed by your comments as shown below:

**Dim Ci As Double ' Concentration inside of cell, mol/liter**

**Dim Co As Double ' Concentration outsiede of cell, mol/liter**

**Dim k As Double ' Diffusion coefficient, 1/minute**

**Dim dt As Double ' Timestep, minutes**

**Dim C As Double ' Concentration inside cell**

**Dim row As Integer ' Row that output will be written to**

**Dim t as Double ‘ Time, minutes**

Now, you are ready to read in the variables and initial conditions from the spreadsheet required for your model. In the following code, we are setting the variables k, Ci, Co, dt and FinalTime to values that are in cells B6, B7, B8, B9 and B10 in the spreadsheet. The ***Range*** function tells VBA to go to the specified cell (ie. “B6”), and the ***Value*** command tells VBA to bind the value in the cell to the variable on the left hand side of the = sign. Type in the commands shown below.

**k = Range("B6").Value**

**Ci = Range("B7").Value**

**Co = Range("B8").Value**

**dt = Range("B9").Value**

**FinalTime = Range("B10").Value**

Next, let’s clear out any old values that may have been written previously in the group of cells in the spreadsheet referred to as “Concentration”. Use the Range.ClearContents as show below to do this.

**Range("Concentration").ClearContents**

You are now ready to write the initial concentration at time t=0 to the spreadsheet. In order to write values into the array of cells referred to as ***Concentration***, you will need to develop an index to tell VBA what row and, in some cases, column you want to write the value to. In this case, Concentration is only a single column of numbers. So, we only need to refer to the row in **Concentration** that we want to write to. Let’s set the row to 1, and write the initial concentration inside the cell into row 1 of ***Concentration***. This can be done by typing the following code:

**' Write the initial concentration at time t=0 into row 1 of**

**' array called "Computed" that was defined in excel**

**row = 1**

**Range("Concentration").Cells(row).Value = Ci**

Now, let’s develop a loop to solve Equation 4 over the specified time interval (FinalTime) using the user specified time step (dt). Note that in the loop below, we are incrementing the row for output to Excel, computing the new concentration at time i, and writing the concentration inside the cell back to Excel in the variable called ***Concentration.*** We are also using the For-Next looping commands in VBA. There are many other ways to construct a loop in VBA. You can use the help menu in VBA to explore these options. Also, note that we are incrementing time from dt (the initial time after 0) to the FinalTime in increments of the user input dt. The STEP function tells VBA how much to increment the variable t each cycle in the loop.

**For t = dt To FinalTime Step dt 'Main loop for solving Equation 4**

**row = row + 1 'Increment row number for output**

**Ci = Ci - k \* (Ci - Co) \* dt 'Equation 4**

**Range("Concentration").Cells(row).Value = Ci 'Write Ci to spreadsheet**

**Next t**

Finally, the End Sub command should already be at the bottom of your program code. This was inserted automatically by VBA when you created the subroutine.

**Step 8:** You can switch between the VBA program and the Excel worksheet by clicking on the ***Developer>Visual Basic*** menu item.

**Step 9:** It is now time to add a button on the spreadsheet to run the VBA program. Follow the menu sequence ***Developer > Insert***. This will pop up a toolbar with several icons on it. Select the “button” button (the square button shape) and then use the left mouse button to drag a rectangular shape anywhere on the spreadsheet. A menu will then pop up and ask you which Macro/program to assign to the button. Select the Macro you just developed, called ***Concentration.*** Select “OK” to end the creation of a button. The button you just created may be highlighted. Click anywhere on the spreadsheet to “unselect” the button. You can click on the text inside the button to change it’s name. You can also right click on the button to change fonts and other attributes.

**Step 10:** You may now run the VBA program you just created by simply clicking on the button you just created. Practice changing the time step and running the program. Notice that the Range("Analytical").ClearContents command in the program will clear out the old model results each time you run the new model. You should see the graphs changing on the screen as you change the parameters and rerun the program.

**Step 11:** Now you may go into your VBA code and run the program line by line, watching variables change as you run the program. You can do this by pressing the F8 function key. This is very useful in debugging your program. On the bottom tool bar, click the Microsoft Visual… button to go back to your program code. Click on the ***Debug*** menu and select the ***Step Into*** menu, or simply press the **F8** function key. By pressing the F8 key repeatedly, you can walk through the program one line at a time. At any point in time, you can move the cursor on top of a variable of interest, and VBA will tell you what the value of the variable is at that point in time. Practice doing this. When you are bored with this exercise, you can click on the button containing the blue square at the top of the program to reset the program and get you out of the debugging mode.

If there is a particularly troubling variable that is not being computed correctly, you can create a watch window that shows the value of that variable throughout the debugging process. Select the ***Debug*** menu and the ***Add to Watch*** menu, and type in the variable name. It will then show up at the bottom of the screen as you walk through the program. Practice doing this with the Ci variable.

**Step 12.** Upload your program to Canvas. I will log into your directory and check your results.