VBA Macros for Solving Problems in Water Chemistry

Chad Jafvert
Purdue University

This handout and Sample Programs are available at:
http://bridge.ecn.purdue.edu/~jafvert/

Outline

- I. Basics
  - A. Creating VBA Subroutines in Excel 2007
  - B. Common Code
  - C. Creating Functions
- II. Sample Programs
  - A. Solving Equilibrium chemistry problems with Newton-Raphson Iterations (Reading, Writing)
  - B. pK_a Diagrams – Buffer Design (arithmetic, and Linking VBA-Calculations to Figures)
  - C. Chemical and Reactor Kinetics (Euler's Method)
  - D. 1-D Diffusion (Central Difference Formula)
  - E. Integrating the Area under a Concentration Profile (Simpson's Rule)

References:

1. The Excel toolbar 'help' button
2. A bunch of sophomores and juniors at Purdue
3. The usual suspects on my bookshelf (water chemistry texts)

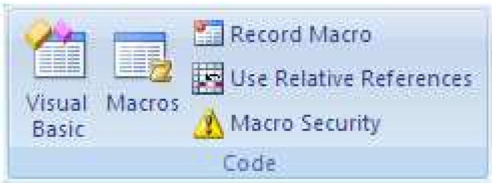
I. Basics

A. Creating VBA Subroutines (In Excel 2007)

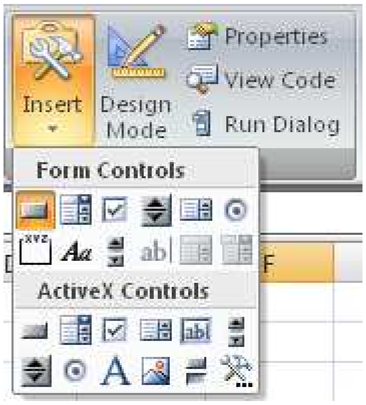
1. If the Developer tab is not available, do the following to display it:

   i) Click the Microsoft Office Button , and then click Excel Options.
   ii) In the Popular category, under Top options for working with Excel, select the Show Developer tab in the Ribbon check box, and then click OK.

2. To set the security level temporarily to enable all macros, do the following:
   i) On the Developer tab, in the Code group, click Macro Security.
   ii) Under **Macro Settings**, click **Enable all macros (not recommended, potentially dangerous code can run)**, and then click **OK**. (To help prevent potentially dangerous code from running, it is recommend that you return to any one of the settings that disable all macros after you finish working with macros.)



3. On the **Developer** tab, in the **Code** group, click **Macros**.
4. Type in a name for the Macro and click Create.
5. A Modules will be created that has a first line: Sub Name(), and a last line: End Sub.
6. Between these lines, type VBA code.
6. To run the macro from the module window, press Run, Run Sub/UserForm
7. When you run a Macro and get an error, you will need to press Run, Reset.
8. Notice that the Start Bar will have tabs for Excel and for the Macro, so that you can toggle between them.
9. To run a Macro from a worksheet, add a button to the worksheet. On the **Developer** tab, in the **Controls** group, click on Insert, and then on the button . Immediately click on the worksheet where you want to place the control button. A window will open for you to assign a Macro to the button. Each time the button is clicked, the Macro will run.

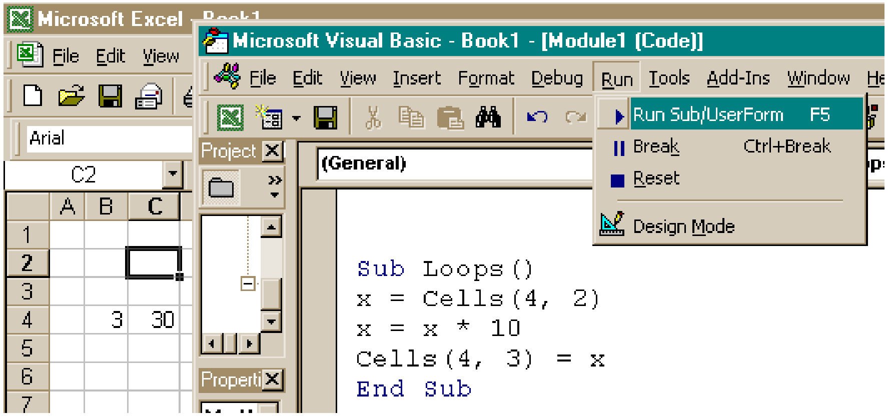


10. When you save and exit the spreadsheet, the associated Macros are saved.
11. The next time you open the spreadsheet, you can immediately run the macro from the button, or edit the macro by clicking on the **Visual Basic** icon of the **Code** group on the **Developer** Tab.

B. Common Code (note: From this point on, the windows shown here are from an older version of Excel, however the code will be the same in Excel 2007)

The program 'Introduction to VBA.exl' in the Appendix lists many common VBA language conventions, including: (i) how to dimension variables and arrays, (ii) how to write For . . . Next loops, (iii) how to read from and write to the spreadsheet, and (iv) how to format common mathematical expressions, like logarithms and exponents.

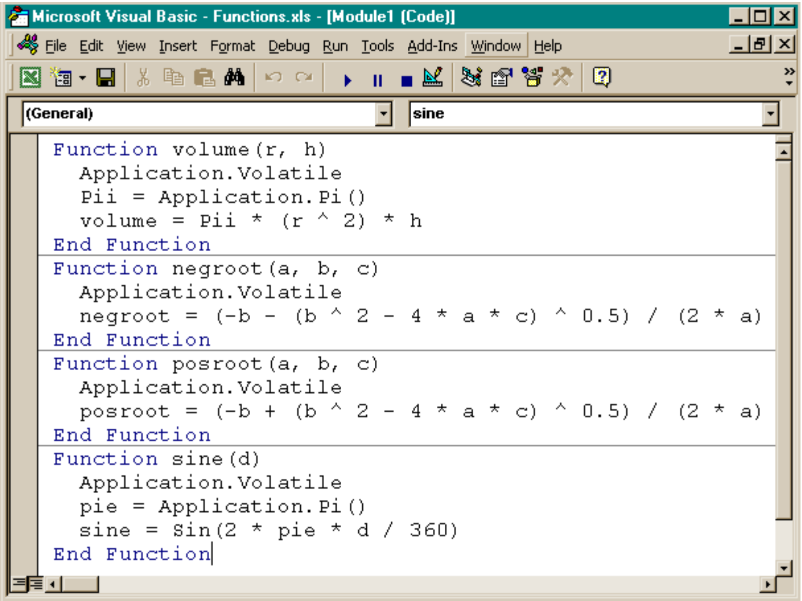
The example code on the right reads cell B4 multiples its value time 10, and writes this product to cell C4. To run this subroutine, click 'Run', move the curser to "Sub/UserForm' and click. To run the program directly from the spreadsheet, add a control button as described above.



```
Sub Loops()
x = Cells(4, 2)
x = x * 10
Cells(4, 3) = x
End Sub
```

C. Creating Functions

All Excel built-in functions are accessed by clicking the fx icon, or by directly typing them in the cells. In addition to these functions packaged with Excel, user-defined functions are created by the same procedure as VBA subroutines. To create a function, simple replace the word 'Sub' in the first line with Function. As with built-in Excel function, all variables must be included in the argument list on the first line. Here are 4 simple functions that (i) calculate the volume of a cylinder, (ii & iii) return the negative and positive roots to the quadratic equation, and (iv) return the sine of an angle whose units are degrees.


Microsoft Visual Basic - Functions.xls - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
(General)
sine

```
Function volume(r, h)
  Application.Volatile
  Pii = Application.Pi()
  volume = Pii * (r ^ 2) * h
End Function
Function negroot(a, b, c)
  Application.Volatile
  negroot = (-b - (b ^ 2 - 4 * a * c) ^ 0.5) / (2 * a)
End Function
Function posroot(a, b, c)
  Application.Volatile
  posroot = (-b + (b ^ 2 - 4 * a * c) ^ 0.5) / (2 * a)
End Function
Function sine(d)
  Application.Volatile
  pie = Application.Pi()
  sine = Sin(2 * pie * d / 360)
End Function
```

To create more than one function or subroutine within the same module, simple type the first and last lines of the new function or subroutine under the previous one, and add the code lines between these. Excel will add horizontal lines between any subroutines or functions making it easier to find the beginning and ending lines of code when editing. These functions can be used on the spreadsheet like any built-in Excel function. For example, with the last function enabled, typing '= sine(45)' in a cell will return the sine of 45° in this cell (= 0.707). Because Excel built-in trigonometric functions operate on angles in units of radians, typing '= sin(0.25·π)' in another cell returns the same value (recall 360° = 2·π radians. Arguments may be numbers or cell references.

II. Sample Programs

A. Solving Equilibrium chemistry problems with Newton-Raphson Iterations (Reading, Writing with a VBA Macro)

The recipe for this problem is taken from "Principles and Applications of Aquatic Chemistry" by François M. M. Morel and Janet G. Hering (Wiley and Sons, NY, 1993). The problem and tableau solution are found on pp. 60-63 of the text. Here, the general solution is developed allowing for (i) easy re-adjustment of initial component concentrations, (ii) exact solution without the need of assumptions, and (iii) activity coefficient correction, even in the absence of swamping electrolyte.

The Recipe:

Add to pure water:	$[CaCO_3]_T$ = 10^{-3} M	$[CO_2]_T$ = 1.1 x 10^{-3} M
	$[HA]_T$ = 4.0 x 10^{-4} M	$[NaCl]_T$ = 10^{-2} M

Morel and Hering assume:

(i) $[NaCl]_T$ has no effect accept for ionic strength correction.

(ii) The amount of HA added has no effect on pH (our solution will be valid for any value of $[HA]_T$

Mass Action Equations:

(assume activity coefficients are equal for same valence ions, calculate with Davies' eq)

$$K_w^{'} = \frac{K_w}{\gamma_1 \cdot \gamma_1} = [H^+]\cdot[OH^-] = 10^{-14} \qquad K_a^{'} = \frac{K_a}{\gamma_1 \cdot \gamma_1} = \frac{[H^+]\cdot[A^-]}{[HA]} = 10^{-4.3} \qquad (1\ \&\ 2)$$

$$K_{a,1}^{'} = \frac{K_{a,1}}{\gamma_1 \cdot \gamma_1} = \frac{[H^+]\cdot[HCO_3^-]}{[H_2CO_3^*]} = 10^{-6.3} \qquad K_{a,2}^{'} = \frac{K_{a,2}}{\gamma_2} = \frac{[H^+]\cdot[CO_3^{2-}]}{[HCO_3^-]} = 10^{-10.3} \qquad (3\ \&\ 4)$$

Mass Balance Equations:

On Charge: $$[H^+] + 2\cdot[Ca^{2+}] = [OH^-] + [HCO_3^-] + 2\cdot[CO_3^{2-}] + [A^-] \qquad (5)$$

On Carbonates: $$C_{T,CO_3^-} = [CO_2]_{T,added} + [CaCO_3]_{T,added} = [HCO_3^*] + [HCO_3^-] + [CO_3^{2-}] \qquad (6)$$

On Acid: $$C_{T,A^-} = [HA]_{T,added} = [HA] + [A^-] \qquad (7)$$

On Calcium: $$C_{T,Ca^{2+}} = [CaCO_3]_{T,added} = [Ca^{2+}] \quad \textit{(an identity)} \qquad (8)$$

Mass balances on Ca^{2+}, Na^{+}, and Cl^{-} are identities & need not be considered further. It is always convenient to select as components, species that contain no other component(s). HA, for example, contains both H^{+} and A^{-}. Following this suggestion, CO_3^{2-}, H^{+}, and A^{-} are selected. Substituting the mass action equations (eqs 1-4) into the remaining mass balance equations (5-7), leaving only constants and the three component species as variables, reduces the problem to a list of 3 equations with 3 unknown, where the 3 unknowns are the 'component species' that we have selected:

$$f_1(H^+, CO_3^{2-}, A^-) \;=\; 0 \;=\; \frac{K_w^{'}}{[H^+]} \;+\; \frac{[H^+][CO_3^{2-}]}{K_{a,2}^{'}} \;+\; 2\cdot[CO_3^{2-}] \;+\; [A^-] \;-\; [H^+] \;-\; 2\cdot C_{T,Ca^{2+}}$$

$$f_2(H^+, CO_3^{2-}, A^-) \;=\; 0 \;=\; \frac{[H^+]^2[CO_3^{2-}]}{K_{a,1}^{'}\cdot K_{a,2}^{'}} \;+\; \frac{[H^+][CO_3^{2-}]}{K_{a,2}^{'}} \;+\; [CO_3^{2-}] \;-\; C_{T,CO_3^{2-}}$$

$$f_3(H^+, CO_3^{2-}, A^-) \;=\; 0 \;=\; \frac{[H^+][A^-]}{K_a^{'}} \;+\; [A^-] \;-\; C_{T,A^-}$$

Find the roots with the Newton-Raphson method (general algorithm for 4 x 4):

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{i+1} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{i} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix}_{i}^{-1} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}_{i}$$

where the subscripts i and i+1 specify where current and next iteration guess values are applied or calculated, respectively. In this example, $x_1 = H^+$, $x_2 = CO_3^{2-}$, and $x_3 = A^-$.

This Spreadsheet that performs these calculations is posted at:

And is named: "Case 3 p60 in Morel & Hering.xls".

There are 2 'sheets' to this spreadsheet. On the first sheet, aptly named 'Analytical Derivatives', the partial derivatives of the Jacobian are solved analytically. The second sheet, appropriately named 'Numerical Derivatives', solves these partial derivatives numerically by calculating the slopes ($\partial f_k / \partial x_j$) over an infinitesimal change in each x_j value ($\partial x_j = x_j + x_j \cdot 10^{-8}$) and the calculated difference between each function over this range ($\partial f_k(x_j) = f_k(x_j) + f_k(x_j + x_j \cdot 10^{-8})$) while holding the 2 other x values constant.

The spreadsheet has 4 macros that are each separately linked to control buttons. One button writes the new guesses to the cells where the old guesses reside (i.e., iterates), a 2nd button performs the same task on the value of the ionic strength, and the 3rd button resets the initial guesses. To show some diversity in programming style, separate macros are written for updating the ionic strength to each worksheet. The other macros are written to be nonspecific to any one worksheet, and hence, operate only on the active worksheet.

B. pK$_a$ Diagrams – Buffer Design (arithmetic, and Linking VBA-Calculations to Figures)

Graphical solutions to acid-base equilibrium problems are ubiquitous in water chemistry textbooks. In Excel, these diagrams are easy to create. Another easy problem is calculating of amount of acid and its conjugate base to add to an aqueous solution to buffer the pH to a given value. Such calculations are performed routinely in chemical kinetic studies or in equilibrium experiments where ionic strength adjustments are necessary, and where the addition of strong acid or strong base for pH adjustment is precluded due to their direct affect on ionic strength. The simplest buffer design equation is the well-known Henderson-Hasselbalch equation:

$$\mathrm{pH} \;=\; \mathrm{pK_a} \;+\; \log\left(\frac{A^{n-1}}{HA^{n}}\right) \qquad (1)$$

where it is assumed that the acid or acid-salt (HA^n) and conjugate base (A^{n-1}) do not accept from or donate to solution or to each other any protons upon their mutual addition to water. From this relationship, the ratio of base to acid added to solution to create a solution at the designated pH value is easily calculated. The pK$_a$, is the ionic strength corrected value. If all activity coefficient corrections are considered, the pH in eq 1 is the concentration-based value, from which the activity (pH probe value) can be (back) calculated. It is easy to show that this equation is valid under most conditions by developing the general solution to the problem, independent of valency of the acid and its conjugant base, assuming that the acid and base salts contain only monovalent cations (i.e., Na$^+$, K$^+$, or Li$^+$). Here the solution is developed assuming Na$^+$ is the cation:

Define: The acid as Na$_x$HA where x = 0, 1, or 2 (i.e., H_3PO_4, NaH_2PO_4, Na_2HPO_4)
The base as Na$_{(x+1)}$A
Species: Na$^+$, HA^{-x}, A$^{-(x+1)}$

Equations: $$C_T = [HA^x] + [A^{-(x+1)}] \qquad K_a = \frac{[H^+]\cdot\gamma_{H^+}\cdot[A^{-(x+1)}]\cdot\gamma_{(x+1)}}{[HA^x]\cdot\gamma_x} \qquad (2\ \&\ 3)$$

Where: $$\gamma_{(x)} = \gamma_{HA^x} \qquad \text{and:} \qquad \gamma_{(x+1)} = \gamma_{A^{-(x+1)}} \qquad (4\ \&\ 5)$$

Let: $$K_a^{'} = K_a\cdot\frac{\gamma_x}{\gamma_{H^+}\cdot\gamma_{(x+1)}} = \frac{[H^+]\cdot[A^{-(x+1)}]}{[HA^x]} \qquad (6)$$

Combining: $$[A^{-(x+1)}] = \frac{K_a^{'}\cdot C_T}{K_a^{'}+[H^+]} \qquad \text{and:} \qquad [HA^{-x}] = \frac{H^+\cdot C_T}{K_a^{'}+[H^+]} \qquad (7\ \&\ 8)$$

If extra salt (i.e., NaCl) is added to adjust the ionic strength, $[Na^+]_{NaCl}$ = $[Cl^-]_{NaCl}$; hence, these additional ions can be ignored in the charge balance. With this in mind, the charge balance can be constructed accounting only for the Na^+ ions that result from dissolution of the buffer acid and base salts:

$$[Na^+] \quad + \quad [H^+] \quad = \quad [OH^-] \quad + \quad x \cdot [HA^{-(x+1)}] \quad + \quad (x+1) \cdot [A^{-(x+1)}] \qquad (9)$$

or: $$[Na^+] \quad + \quad [H^+] \quad = \quad [OH^-] \quad + \quad x \cdot C_T \quad + \quad [A^{-(x+1)}] \qquad (10)$$

And the Mass Balance on sodium (due to buffer species) can be constructed:

$$[Na^+] \;=\; x \cdot [Na_xHA]_{added} + (x+1) \cdot [Na_{(x+1)}A]_{added} = x \cdot C_T + [Na_{(x+1)}A]_{added} \qquad (11)$$

Combining the last 2 equations and simplifying, results in:

$$[Na_{(x+1)}A]_{added} \quad = \quad [OH^-] \quad - \quad [H^+] \quad + \quad [A^{-(x+1)}] \qquad (12)$$

Equation 12 confirms that except at extreme pH values, the acid or acid-salt (HA^n) and conjugate base (A^{n-1}) do not accept or donate any protons to solution or to each other upon their mutual addition to solution. (i.e. $[Na_{(x+1)}A]_{added} = [A^{-(x+1)}]$).

Combining eqs 7 & 12:

$$[Na_{(x+1)}A]_{added} \quad = \quad [OH^-] \quad - \quad [H^+] \quad + \quad \frac{K_a^{'} \cdot C_T}{K_a^{'} + [H^+]} \qquad (13)$$

From this equation, the amount of base to be added is calculated, and the amount of acid to be added is calculated by difference:

$$[Na_xHA]_{added} \quad = \quad C_T \quad - \quad [Na_{(x+1)}A]_{added} \qquad (14)$$

The program 'pK_a. xls' performs these calculations, making appropriate ionic strength corrections, and plots the pC-pH diagram for the buffer concentration and pK_a specified by the user. Note that the subroutine writes the calculated speciation to the second worksheet, and the 'chart' on the first worksheet graphs these values. The graph is undated any time the input parameters are changed and the program is rerun.

C. **Chemical and Reactor Kinetics** (Euler's Method)

Because VBA subroutines provide an easy way to perform iterations and to link program output to graphs, they are ideal for encoding simple numerical schemes with almost immediate display of graphical output. Additionally, is very easy (i) to display experimental data on the same figure that contains 'model' results, (ii) to evaluate squared residuals between model and data values, and (iii) to either minimize these residuals by 'eye' or with a simple grid-search method (such as the method of bisection). As an example, the problem of tetrachloroethylene transport in Lake Greifensee, Switzerland, from "Environmental Organic Chemistry" by René P. Schwarzenbach, Philip M. Gschwend & Dieter M. Imboden (Wiley and Sons, NY, 1993) is solved. The problem is presented on pp. 551-574 with figures of model simulations presented on p. 573, Figure 15.8. In the text, calculates are performed over the entire yearly cycle, however, herein the calculations are presented only for the first 90 days after the lake becomes stratified. Equations 15-30a and 15-30b on p. 569 are a pair of simultaneous 'first order linear inhomogeneous differential equations' (FOLIDE) that describe the mass balances of the chemical in a stratified lake and are reproduced here:

$$\frac{dC_E}{dt} = \frac{I_E}{V_E} + k_{g,E'} \cdot \frac{C_a}{K_H^{'}} - (k_{w,E} + k_{g,E} + k_{ex,E}) \cdot C_E + k_{ex,E} \cdot C_H \qquad (1)$$

$$\frac{dC_E}{dt} = \frac{I_E}{V_H} + k_{ex,H} \cdot C_E - k_{ex,H} \cdot C_H \qquad (2)$$

The text presents the matrix (eigen value) analytical solution to this problem in Table 15.5. The Excel spreadsheet entitled 'Stratified Lake.xls' presents the Euler's method numerical solution, to this problem assuming the model parameters presented below. These parameters similar, but not necessarily identical to those used to create Figure 15.8a in the text, as slight variation occurs between in the predictions presented in the text with those calculated with the spreadsheet. Here, Δt = 1 d is sufficiently small, however smaller time steps can be invoked by adding some 'write' counters.

Model Parameters

$K_{w,E}$ (1 / day) =	0.0068	Epilimnetic flushing rate
$K_{g,E}$ (1 / day) =	0.0267	Epilimnetic gas exchange rate
$K_{ex,E}$ (1 / day) =	0.0075	Exchange across thermocline (epi)
$K_{ex,H}$ (1 / day) =	0.00375	Exchange across thermocline (hyp)
K_H (unitless) =	0.727	Henry's Constant for PCE
I_E (mol / day) =	0.0	Total daily input of PCE to *epilimnion*
I_H (mol / day) =	0.9	Total daily input of PCE to *hypolimnion*
Mass (mol) =	83	Total initial mass of PCE in lake upon stratification
C_a (mole / m^3) =	0.00000001	PCE concentration in the gas phase above the lake
V_E (m^3) =	50000000	Volume of the epilimnion
V_H (m^3) =	100000000	Volume of the hypolimnion

D. 1-D Diffusion (Central Difference Formula)

The 1-D diffusion equation is:

$$\frac{\partial C}{\partial t} = D_m \cdot \frac{\partial^2 C}{\partial x^2} \qquad (1)$$

where C is chemical concentration, t is time, x is distance, and D_m is the molecular diffusion coefficient. Equation 1 has several analytical solutions depending on all initial and boundary conditions. For the case of an impulse input at x = 0 into a semi-infinite media (C = 0 at x = ∞ at all times), where $C_{t=0} = 0$ at all other values of x, eq 1 has the following solution:

$$C = \frac{M}{2\sqrt{\pi \cdot D_m \cdot t}} \cdot e^{\frac{-x^2}{4 \cdot E \cdot t}} \qquad (2)$$

where M is the mass of chemical added at x = 0, with units of mass per cross-sectional area (e.g., mg / cm^2). Obviously, diffusion occurs in both directions, with the concentration profile extending further out as time is increased. To graphically display the concentration profile over a finite distance, and to avoid expressing distance as a negative value, the entire distance-scale may be offset. Defining L as the length of the media, x_{new} as the new distance locations from x_{new} = 0 to L, and x = (x_{new} – L/2). The initial impulse occurs (at x = 0) at the midpoint of the finite-length media at x_{new} = L/2.

Equation 1 can be solved also quite easily with numerical methods. The concentrations of the chemical at x_{new} = 0 and L are assumed to equal zero at all time steps. This assumption can be evaluated by examining the calculated concentration profile or by comparing the area under the curve to the initial mass. Applying the central difference formula to estimate the second derivation, the following formula is applicable at all internal nodes, n.

$$C_n^{j+1} = r \cdot (C_{n+1}^j + C_{n-1}^j) + C_n^j \cdot (1 - 2 \cdot r) \qquad (3)$$

where j and j + 1 are the current and next time steps, respectively, and r is defined by:

$$r = \frac{D_m \cdot \Delta t}{(\Delta x)^2} \qquad (4)$$

where Δt is the time step, and Δx is the distance between nodes. The Excel spreadsheet '1D Diffusion.xls' has these analytical and numerical solutions, with a VBA macro calculating the numerical solution.

E. Integrating the Area under a Concentration Profile (Simpson's Rule)

The previous example creates a nice set of data whose integral should equal the initial mass. Mass conservation can be assessed by comparing the concentration profile to the analytical solution, or by numerically integrating the area under the concentration profile. The spreadsheet '1D Diffusion.xls' contains a second macro that invokes Simpson's Rule to perform this calculation.

$$\text{Mass per unit area} \quad = \quad \frac{\Delta x}{3}\cdot\left(C_1 \quad + \quad 4\cdot\sum_{\substack{i=2 \\ \text{Step}=2}}^{n-1} C_i \quad + \quad 2\cdot\sum_{\substack{i=3 \\ \text{Step}=2}}^{n-2} C_i \quad + \quad C_n \right)$$

where the subscripts refer to the node number, that in this case equal 1 to 101.
This macro can be modified easily to integrate the area under any set of data, or by replacing the cell references in the summation equations, to integrate the area under any function, employing a very small step-size for accuracy. A control button is located on the spreadsheet to automate the calculation.

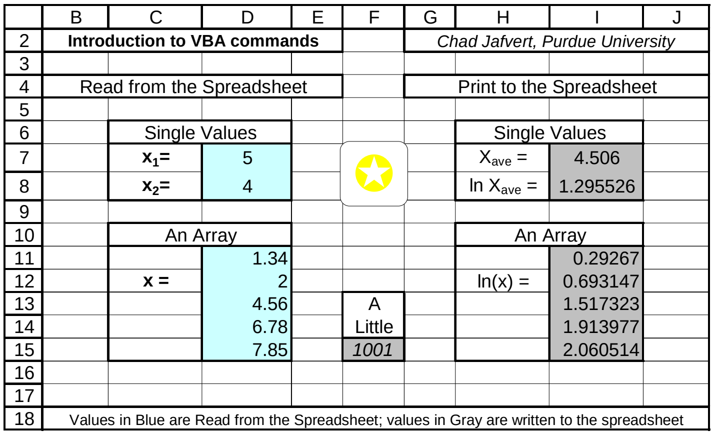
Appendices

Program 1: Introduction of VBA

Source Code:

```
Sub intro()
'  Author: C. T. Jafvert, Purdue University
'  Introduction to Visual Basic for Applications (VBA)
commands
'  At any time during program construction, you may test your
code***
'  An apostropy or REM in the 1st space signifies 'comment'
'  Blank lines are OK; The Help, index is very useful

'  To declare xx and yy as Double Precision Variables:
   Dim K1, K2, Xave, Yave, X1, Y1 As Double
'  To declare an integer
   Dim I As Integer
'  To declare double precision Arrays or Matrices:
   Dim X(1 To 20), Y(1 To 20), matri(1 To 10, 1 To 10) As
Double

'  To declare a string of miscelaneous characters:
   Dim title1 As String * 10

'  One way to initialize coefficients or data:
   Conc1 = 2#         'note: The # means real (2.0 = 2#)

'  Another way is to read it from the spreadsheet:
    ' This line reads the value in cell D6 (row 7, column 4)
    ' and defines as K1:
   K1 = Cells(7, 4)    'or formally: A1 = Cells(7, 4).Value
'  This line reads the value in cell D8 and defines as K2:
   K2 = Worksheets("sheet1").Range("D8").Value

'  To read in an array of data from the spreadsheet,
'  read the data within a For-Next loop
   For I = 1 To 5
     X(I) = Cells(10 + I, 4).Value   'first value is cells(11,4)
   Next I
'  This loop is like a DO loop in FORTRAN and can be Nested

'  Calculations can be performed on this data set.
'  Lets take the natural log of all of these numbers:
   n = 5
   For I = 1 To n Step 1  'loops can have step sizes different
that 1
     Y(I) = Log(X(I))     'log is natural log
   Next I

'  Lets find the average of both arrays
   X1 = 0: Y1 = 0    'a : separates two lines of code in the same
                  'physical line
   For I = 1 To n
     X1 = X1 + X(I):  Y1 = Y1 + Y(I)
  Next I
   Xave = X1 / n:  Yave = Y1 / n
```

Spreadsheet:



	B	C	D	E	F	G	H	I	J
2	**Introduction to VBA commands**					*Chad Jafvert, Purdue University*			
3									
4	Read from the Spreadsheet					Print to the Spreadsheet			
5									
6		Single Values					Single Values		
7		**x_1=**	5		★		X_{ave} =	4.506	
8		**x_2=**	4				ln X_{ave} =	1.295526	
9									
10		An Array					An Array		
11			1.34					0.29267	
12		**x =**	2				ln(x) =	0.693147	
13			4.56		A			1.517323	
14			6.78		Little			1.913977	
15			7.85		*1001*			2.060514	
16									
17									
18	Values in Blue are Read from the Spreadsheet; values in Gray are written to the spreadsheet								

```
'  Other common mathematical functions:(see help for other
functions)
   x10 = X1 * Y1 / X1
   Y10 = Log(Y1) / 2.302585092994   'base 10 log
   x11 = 10 ^ (X1)                  '10 to the power
   x12 = Exp(X1)                    'e to the power
'  note: VBA does not know the value of Pi:
   Pie = Application.Pi()          'so import it from Spreadsheet.
   x4a = Abs(x4)                   'to take the absolute value:
   x4a = Int(x4a)                  'to make a real number an
integer:

'  Let's print to the spreadsheet:
'  Print the value of Xave to cell I7 (I is the 9th letter):
   Cells(7, 9).Formula = Xave
'  This prints Yave to cells I8; You can print to different
sheets.
   Worksheets("sheet1").Range("I8").Value = Yave

'  Print the array Y(I) to cells I11 to I15
   For I = 1 To n
     Cells(10 + I, 9).Formula = Y(I)
   Next I

'  To loop back to a specific line, number the line:
   X1 = 1
   'And let's change the Font to Bold, Italics
   Worksheets("sheet1").Range("F15").Font.Italic = True
10 X1 = 1 + X1
   Worksheets("sheet1").Range("F15") = X1
   If X1 < 1000.5 Then GoTo 10

End Sub
```

Program 2: Solving Equilibrium chemistry problems with Newton-Raphson Iterations (Reading, Writing with a VBA Macro)

Source Code:

```
Sub loop1()
Rem This subroutine reads values in I30:I32
'   and writes them to B12:B14.
'   This values are the initial guesses.
For I = 1 To 100             'try I = 1 to 100
x1 = Cells(30, 9).Value
x2 = Cells(31, 9).Value
x3 = Cells(32, 9).Value
Cells(12, 2).Formula = x1
Cells(13, 2).Formula = x2
Cells(14, 2).Formula = x3
Next I
End Sub

Sub loop2()
Rem  Reads the ionic strength from L28 and writes to F5
u = Cells(28, 12).Value
Cells(5, 6).Formula = u
End Sub

Sub Init()
Rem Resets the initial guesses to 1.0e-5, only
'   on the 'Analytical Derivatives' worksheet.
Worksheets("Analytical Derivatives").Range("B12:B14").Value = 0.00001
Worksheets("Analytical Derivatives").Range("F5").Value = 0#
End Sub

Sub Init2()
Rem Resets the initial guesses to 1.0e-5, only
'   on the 'Numerical Derivatives' worksheet.
Worksheets("Numerical Derivatives").Range("B12:B14").Value = 0.00001
Worksheets("Numerical Derivatives").Range("F5").Value = 0#
End Sub
```

	A	B	C	D	E	F	G	H	I	J	K	L
2	**Case 3, pp 60-63, Morel and Hering**					**w/ Analytical Derivatives**					**ANS:**	
3	*Chad Jafvert, Purdue University*											
4						**at μ =**	**γ_1 =**	0.8963			**Calculated**	
5	**Initial Values (Recipe) (M)**			**Constants at μ = 0**		1.20E-02	**γ_1 =**	0.6453			**Species Concentrations**	
6	$(CaCO_3)_T$ =	1.00E-03		K_w =	1.00E-14	1.24E-14						
7	$(CO_2)_T$ =	1.10E-03		K_a =	5.01E-05	6.24E-05					**Identities (M)**	
8	$(HA)_T$ =	4.00E-04		$K_{a,1}$ =	5.01E-07	6.24E-07					Ca^{2+} =	1.00E-03
9	$(NaCl)_T$ =	1.00E-02		$K_{a,2}$ =	5.01187E-11	7.77E-11					Na^+ =	1.00E-02
10											Cl^- =	1.00E-02
11	**Initial Guesses (M)**			**Functions**								
12	x_1 (H^+) =	1.947E-07		f_1 =	0.000E+00						**Components (M)**	
13	x_2 (CO_3^{2-}) =	6.384E-07		f_2 =	0.000E+00						$[H^+]$ =	1.95E-07
14	x_3 (A^-) =	3.988E-04		f_3 =	0.000E+00						$[CO_3^{2-}]$ =	6.384E-07
15											$[A^-]$ =	3.988E-04
16	**The Partial Derivatives:**											
17	df1/dx1=	8.218E+03		**In Matrix Notation:**							**Other Species (M)**	
18	df1/dx2=	2.508E+03			**A**						$[OH^-]$ =	6.39E-08
19	df1/dx3 =	1.000E+00		8.218E+03	2.508E+03	1.000E+00					$[H_2CO_3]$ =	4.99E-04
20	df2/dx1 =	1.335E+04		1.335E+04	3.289E+03	0.000E+00					$[HCO_3^-]$ =	1.600E-03
21	df2/dx2 =	3.289E+03		6.391E+00	0.000E+00	1.003E+00					$[HA]$ =	1.244E-06
22	df2/dx3 =	0.000E+00										
23	df3/dx1 =	6.391E+00									**Mass Bal. (Check)**	
24	df3/dx2 =	0.000E+00									$(CO_3^{2-})_T$ =	2.100E-03
25	df3/dx3 =	1.003E+00									$(HA)_T$ =	4.000E-04
26												
27					**Function**		**Product**		**New**		**Other**	
28	**The Inverted Matrix (Jacobian)**				**Vector**		**Vector**		**Guesses**		μ =	1.20E-02
29		**A^{-1}**			**f(x1,x2,x3)**		**$A^{-1}f$**		**x_{new}**		$p\{H^+\}$ =	6.71
30	-5.083E-04	3.876E-04	5.067E-04		0.000E+00		0.000E+00		1.947E-07			
31	2.063E-03	-1.269E-03	-2.056E-03		0.000E+00		0.000E+00		6.384E-07			
32	3.238E-03	-2.469E-03	9.937E-01		0.000E+00		0.000E+00		3.988E-04			
33												

Initially, assign a constant value to the ionic strength (e.g., 0.001 M); perform the iterations until convergence is achieved; then use the calculated value for ionic strength by simply setting tll equal to cell I37 (i.e., move μ); and iterate again until convergence is achieved.