Implementing an Analytical Solution for Porous Drainage on a Sloping Bed with Uniform Rainfall in MS Excel using VBA Macro Programming

By

Macey H. Taylor

For

CE4000-020
Spring 2016

TABLE OF CONTENTS

INTRODUCTION

Childs (1971) recognized that an assumption had been incorrectly made for determinations of groundwater flow supplied by uniform rainfall over a sloping bed to transverse ditch drains. Werner (1957) and Schmid and Luthin (1964) believed that equipotential lines are vertical to the sloping bed, which they are actually perpendicular. These equipotential lines can be thought of as contour lines for the water table. So, this incorrect theory caused problems in previously derived equations. Childs (1971) modified these studies and presented a differential equation for the determination of groundwater flow, assuming streamlines were parallel to the bed slope. Childs (1971) alleged that no analytical solutions could be determined because how specific the cases for this theory were. Towner (1975) discovered that by integrating this differential equation, analytical solutions could be determined and these solutions were very similar to those of Schmid and Lutin (1964). Three analytical solutions were formed under three cases that were dependent upon bed slope, rainfall rate and hydraulic conductivity. Using these solutions for each case, the greatest vertical height to the water table mound, the location to this height and to the watershed, and the shape of the water table mound could be determined for a horizontal aquifer system. Using the integrated analytical solutions, VBA code and Newton's method, the vertical distance of the water table above the sloping bed, z, was estimated from a supplied horizontal distance measured from the upper drain, x.

PROBLEM STATEMENT

For Towner and the research within this paper, the drainage of groundwater in equilibrium with uniform rainfall flowing down a sloping bed via equidistant, parallel ditch drains routing down the bed slope and into the impermeable bed posed as an issue (1975). The vertical distance of the water table above the sloping bed needed to be estimated given a horizontal distance measured from the upper drain. From these distances, the shape and location of the water table could be determined for a horizontal aquifer system.

SKETCH

Figure 1 is a representation of a typical unconfined aquifer system with uniform rainfall infiltrating groundwater and draining over a bed slope. The bed slope, which is the very bottom layer of Figure 1 beneath the groundwater, will vary and is dependent upon the bed slope, a. Figure 2 is a cross section view of groundwater flowing between drains on a sloping bed. This figure helps better visualize a horizontal aquifer system and recognize parameters of the analytical solutions.

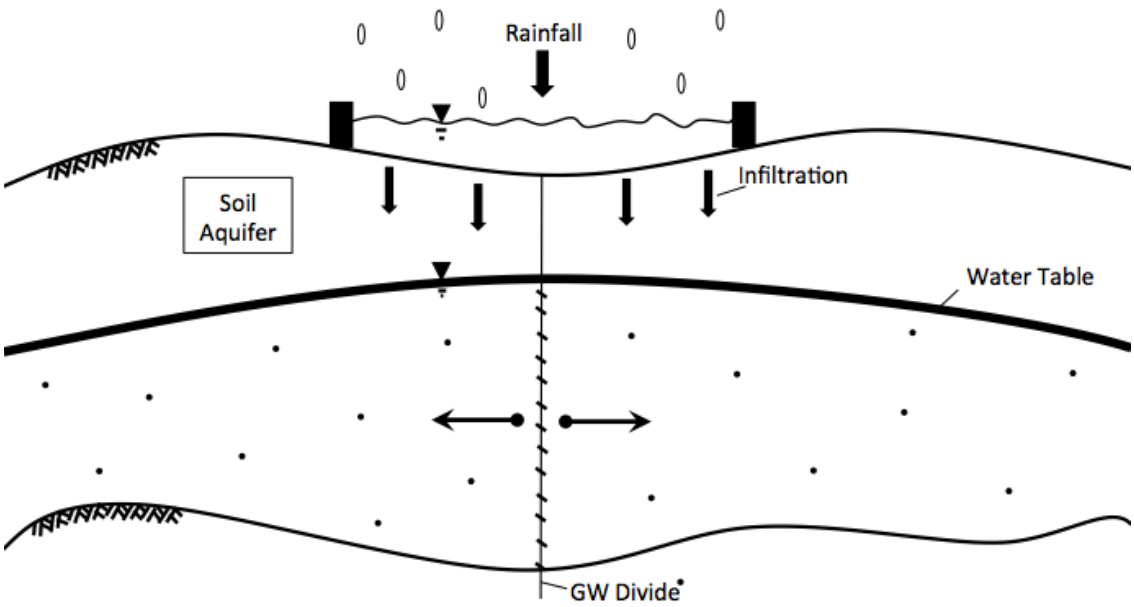

FIGURE 1. UNCONFINED AQUIFER SYSTEM

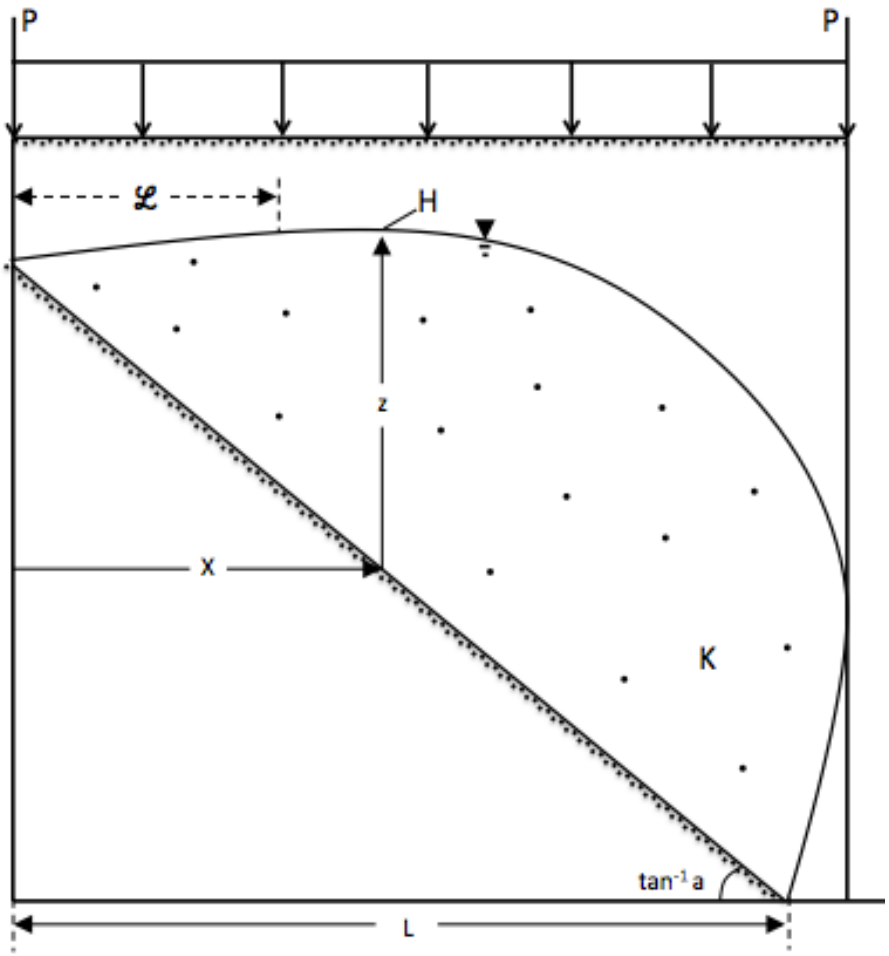

FIGURE 2. CROSS-SECTION OF THE GROUNDWATER BETWEEN DRAINS ON A SLOPING BED

KNOWN

The known parameters, excluding z, for the aquifer system are shown in Figure 2. P represents the uniform rainfall rate. H represents the highest point or mound on the water table and is the maximum value of z for said case. The horizontal distance measured from the upper drain is represented by x. The bed slope is given by a and the hydraulic conductivity is given by k. $\mathscr{L}$ represents the horizontal distance to the watershed. Using these parameters, the integrated analytical solutions for each case could be put into x as a function of z and z could be determined for any given value of x.

UNKNOWN

The main goal of this research was to manipulate each solution for each case using a supplied value of x to determine the associated value of z. Newton's method was used in order to estimate the value of z as x approached zero.

ANALYTICAL SOLUTIONS

As previously stated, it was believed that no analytical solutions could be determined, but this was incorrect. *Towner* (1975) calculated analytical solutions by integrating differential equations derived by Childs (1971), which this research stemmed from various ideas presented from previous researchers. These solutions determined from integration were for the case of ditches and uniform rainfall. Towner developed three solutions for three different cases, which were dependent upon a relationship between the known values. Using these solutions, the greatest vertical height to the water table mound, the location to this height and to the watershed, and the shape of the water table mound could all be determined. Furthermore, by setting the analytical solutions for each case to zero, x became a function of z and the critical points were found that determined a corresponding z.

VBA CODING METHODOLOGY

A module is used in developer within Microsoft Word to code functions. Within these functions, there are parameter names that define variables. Table 1 shows the variables and their parameter names used and seen throughout this VBA code. The description explains what the specific parameter is used for or relates to.

TABLE 1. VBA PARAMETERS AND FUNCTION NAMES USED FOR TOWNER VARIABLES AND DESCRIPTIONS OF EACH

Description	Variable	VBA Parameter
Bed slope	a	a
Rainfall rate	p	p
Hydraulic conductivity	k	k
Horizontal distance from upper drain	x	x
Vertical distance of water table above sloping bed	z	z
Horizontal distance between the drains	L	L
Horizontal distance to the watershed	$\mathcal{L}$	curvyL
Value differentiating between case 1, 2, or 3	A	capA
Relationship between x and $\mathcal{L}$	v	vValue
Defining value for all cases	W	capW
Defining value for case 1	m	mValue
Defining value for case 1	u_0	uSub0
Defining value for case 1	u_H	uSubH
Height of the mound of water for case 1	H_1	capH1
Defining value for case 1	u	uValue
Horizontal distance to mound for case 1	X_1	capX1
Height of the mound of water for case 2	H_2	capH2
Horizontal distance to mound for case 2	X_2	capX2
Defining value for case 3	W_1	capW1
Defining value for case 3	W_2	capW2
Height of the mound of water for case 3	H_3	capH3
Horizontal distance to mound for Case 3	X_3	capX3
Defining value for case 3	C_4	capC4
Solution function for case 1		Case1
Manipulation of solution function to estimate z given x		NewtonCase1
Solution function for case 2		Case2
Manipulation of solution function to estimate z given x		NewtonCase2
Solution function for case 3		Case3
Manipulation of solution function to estimate z given x		NewtonCase3
Number of guesses for Newton's method		HowMany
Deviation from original guess for Newton's method		stepsize
How close the value is to zero for Newton's method		tolerance

In Table 2, there are a few basic VBA symbols used for certain operations in functions and code.

TABLE 2. HELPFUL VBA OPERATIONS AND SYMBOLS

Operation	Symbol
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or	<=
Equal to	=
Not equal to	<>
Natural log, ln	Log
Defines a comment describing code/function	‘comment
Defines a string value is being used	“”
Defines function and/or parameter data type	As Double (or String)

The operational math terms like “greater than” or “equal to” are very helpful to know for IF-THEN-ELSE statements. Using the symbols can help simplify code that could be in multiple functions and make them into single functions. The natural log was used quite often within this code and needed to be defined so there would be no confusion when calling the natural log function in the module. Comments can be helpful and describe what a certain function is doing or just clear something up within the code that is not obvious. A comment should at least be used before each function and more than one comment can be used. Where a comment is located in the function does not affect anything. Finally, there are two data types in code, “string” and “double”. Double defines a number and using the identifier character “#” after a number forces it to become a double data type. String defines text and words. Functions and parameters can be forced as a number or text value.

ERROR HANDLING

Error handling has been done through out the computation of each function. Some of these error handlers used notified the user when null values were put in, a parameter was zero that could not be, or when a number for a certain variable needs to be changed. Also, basic error handlers have been put used in case of general errors arising. When these errors happen, the error handlers give the user an error number and description to better diagnose the issue. The algorithm for an error handler returning an error number and description is as follows:

1. Put “On Error“ statement or IF statement for an error after function parameters and name are defined and before the function equation or IF-THEN-ELSE statements.

```
On Error GoTo ErrorHandlerParameterIssue

                    OR

If parameter..Then GoTo ErrorHandlerParameterIssue
```

2. “Exit Function” and put in error handler(s) name before “End Function”

3. Type in message wanted to be returned on error and if necessary, make the error handler return number and description. Also, a certain string or double can be set to return in the spreadsheet for the function value on error.

```
    Exit Function

ErrorHandlerParameterIssue:
    MsgBox "message" & vbCrLf & "Error" & Err.Number &
        ":" & Err.Description
    FunctionName = "string" or Number
    Exit Function

End Function
```

NEWTON'S METHOD

Netwon's method was used to estimate a value of *z* from a supplied *x* value. By finding the zeroes or critical points of the analytical solutions for each case, a value of *z* was estimated from a specific x value. Each solution was set to zero by moving all terms in the function to one side. A loop function was used to guess values of z. This was done by finding the slope at a certain point of the function, which gave a corresponding x and *z* value. The loop continued until the value of *x* was approximately zero, hence, giving the appropriate value of z. The equation defining Newton's method in terms of this case of estimating *z* is as follows:

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

In this equation, z_n is the first initial guess the loop starts at. The function or analytical solutions with the estimated *z* value is represented by $f(z_n)$ and the derivative of that function, which represents the slope of the function at that point, is represented by $f'(z_n)$. To estimate with Newton's method and use a loop, a step size was given and this was the deviation from the original guess each additional guess would follow until it reached a critical point. Also, a number of guesses were given for the count used in the loop, along with a tolerance that governed if the function value was close enough to zero. The derivative of the function in this case was found by the following equation:

$$f'(z_n) = \frac{f(constants, z_n + stepsize) - f(constants, z_n)}{stepsize}$$

Here, the constants are values like *A, a, p, k* and so on. These values do not change based off of the given *x* value and therefore, are considered constants. Using these functions and tolerance in a loop, allowed for the approximation of *z*.

EQUATION SELECTION

Towner (1975) uses and manipulates Childs' notation of a revised differential equation correction Werner (1975) and Schmid and Luthin (1964) to come up with the following relationship.

$$\int \frac{W}{W^2 - AW + (\frac{p}{k})}\, dW = -\int \frac{dv}{v}$$

This relationship was integrated from the following function, which is Equation 1 in Towner (1975):

$$\frac{dz}{dx} = \frac{akz - p(x - \mathcal{L})(1 + a^2)}{kz - p(x - \mathcal{L})\mathrm{a}}$$

In this integration, v and W are defined where:

$$v = x - \mathcal{L}$$

$$W = \frac{z}{v} - a(\frac{p}{k})$$

The following VBA script in Figures 3 and 4 represents the two functions calculating W and v.

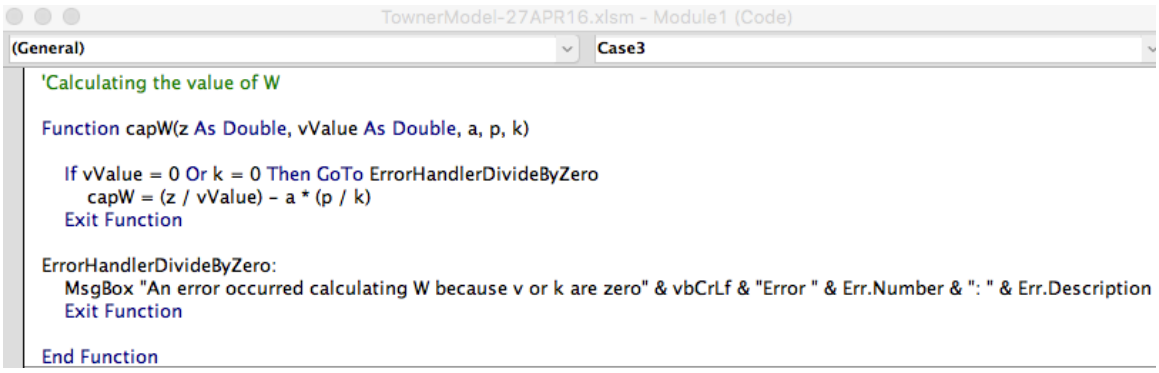

FIGURE 3. FUNCTION USED TO DETERMINE W

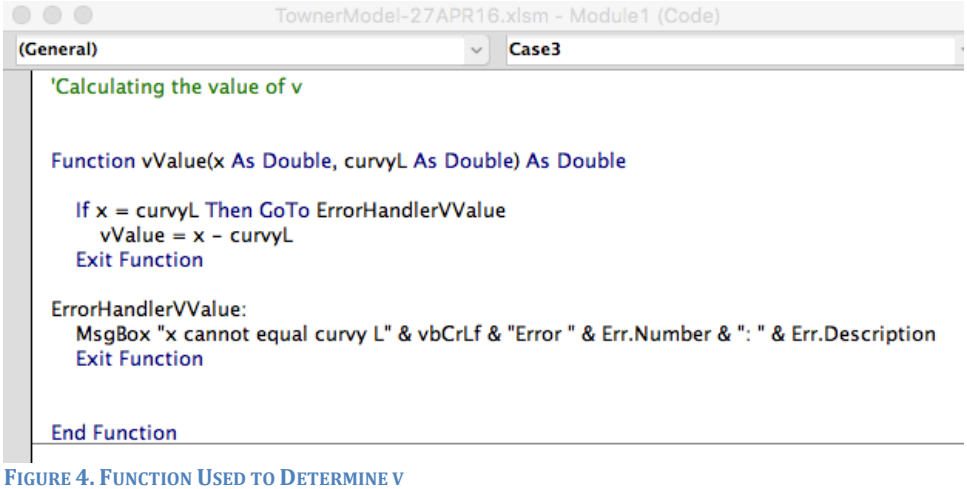

FIGURE 4. FUNCTION USED TO DETERMINE V

Towner (1975) reduces the hydraulics to three cases depending on the value of the variable A and the rainfall rate and hydraulic conductivity within the groundwater portion. After integrating Equation 1 in Towner (1975), it was seen that the

analytical solution depends on the relationship between A^2 being greater than, less than, or equal to 4*(p/k). On page 144 of Towner, A is defined as:

$$A = a[1 - \frac{p}{k}]$$

The VBA script calculates the value of A in the code fragment seen in Figure 5.

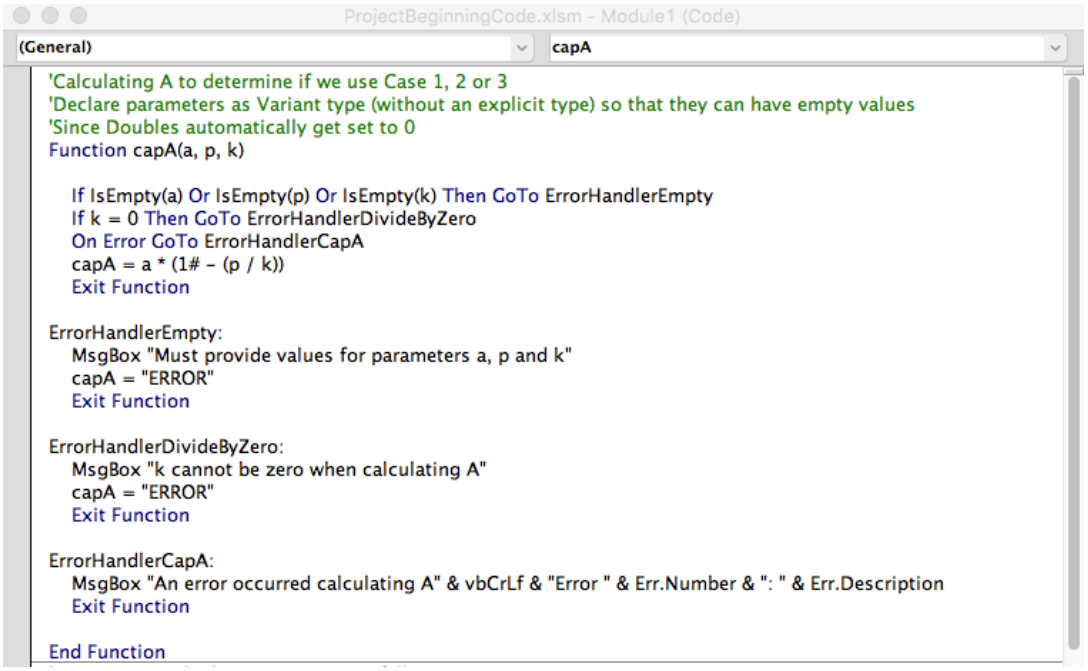

FIGURE 5. FUNCTION USED TO DETERMINE VALUE OF A

The VBA script makes the selection of the equation, or case, to use using the code fragment shown in Figure 2. Case 1 was defined by $A^2 < 4(p/k)$. Case 2 was defined by $A^2 = 4(p/k)$. Case 3 was defined by $A^2 > 4(p/k)$.

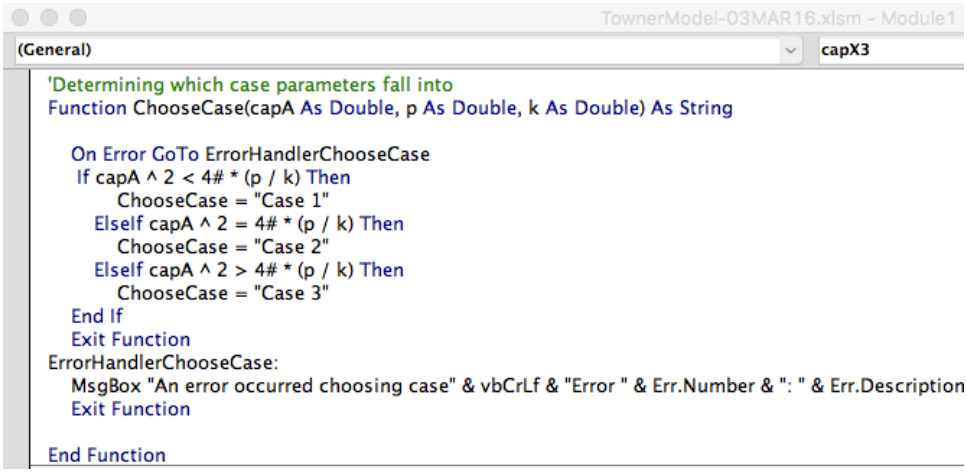

FIGURE 6. SELECTING WHICH EQUATION (CASE) TO USE BASED OFF OF CALCULATED A VALUE

In order to estimate a value of *z* for all three cases, values for *a, p,* and *k* that fell below each case were determined. Values could easily be found for Cases 1 and 3, but Case 2 was trickier. Using algebra, it was found that Case 2 solutions governed

when *a* was equal to -2.4, *p* was equal to 9, and *k* was equal to 4. Furthermore, Figure 6 shows a good example of an IF-THEN-ELSE statement and the algorithm is as follows:

1. Code If statement after function name and parameters are defined and after error handlers. Operational math terms can be used to compare numbers or parameters or strings can be used to define the if statement.

```
If Parameter1 < (>, =, <>, <=, >=) Parameter2 Then
    Function = Double or "String"
```

2. Code ElseIf statement if additional conditions can be applied

```
ElseIf Parameter1 > (>, =, <>, <=, >=) Parameter2 Then
    Function = Double or "String"
```

3. Code Else statement to define outcome if no previous conditions apply and end function with End If statement. To ensure no error, input Exit Function statement after End If.

```
Else
   Function = Double or "String"
End If
Exit Function
End Function
```

METHODS TESTING

CASE 1

Throughout each case, different variables had to found to solve the analytical solution. For Case 1, functions for *m, u, and* u_0 had to be written to determine the solutions. The value of m was calculated using the code fragment in Figure 7. Using an IF-THEN-ELSE statement, a string of "N/A" will be returned when the case does not fall under Case 1. Throughout the code fragments, IF-THEN-ELSE statements were used to define a function value as "N/A" when the case was not met. This allowed for a cleaner looking spreadsheet interface and ensured no errors would return in a cell if the function was not applicable to the specific case.

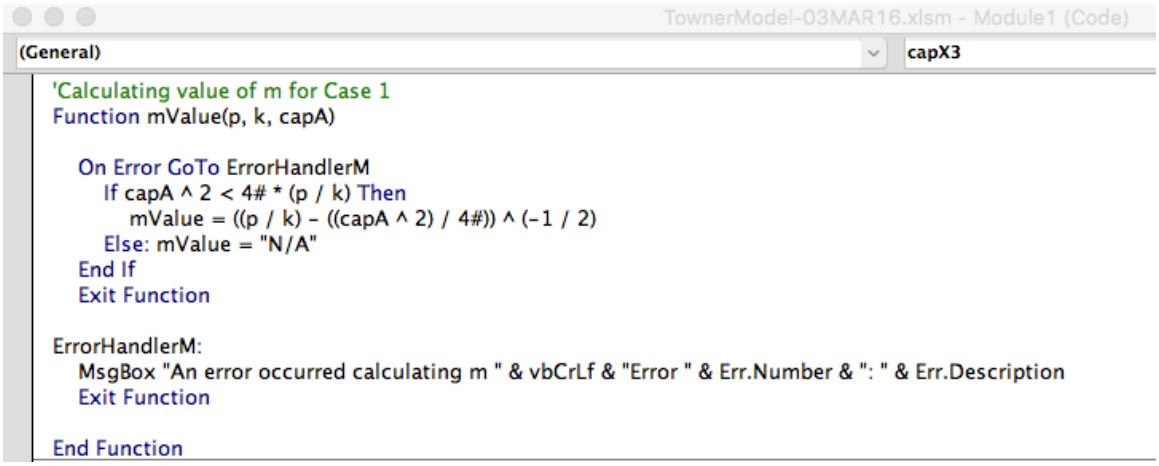

FIGURE 7. FUNCTION USED TO DETERMINE M FOR CASE 1

Figure 8 shows Equation 4 from Towner (1975) .

$$\ln\left(\frac{H}{L-\mathfrak{L}}\right)^2 = \ln\left[\left(\frac{1+a^2}{a}\right)^2\left(\frac{p}{k}\right)^2\left(\frac{u_0^{\,2}+1}{u_H^{\,2}+1}\right)\right] + Am(\tan^{-1}u_0 - \tan^{-1}u_H) \qquad (4)$$

where

$$u_0 = -m\left(\frac{p}{k}a + \frac{A}{2}\right)$$

and

$$u_H = m\left(\frac{p}{ka} - \frac{A}{2}\right)$$

FIGURE 8. EQUATION 4 IN TOWNER USED TO DETERMINE THE MOUND, H, FOR CASE 1

The first function in this figure represents the function to determine the maximum value of *z* for Case 1. To determine the point of the mound, H, a function had to be made for u_H. The entire right hand side of Towner Equation 4 is comprised of known values; that is it is a constant. So, in the code, the right hand side is evaluated and then the following algebraic and logarithmic transformations are used to compute a value for H:

$$\ln\left(\frac{H}{L-\mathcal{L}}\right)^2 = \text{constant}$$

$$\ln\left(\frac{H}{L-\mathcal{L}}\right) = constant^{\frac{1}{2}}$$

$$\frac{H}{L-\mathcal{L}} = \exp\left(constant^{\frac{1}{2}}\right)$$

$$\mathrm{H} = (\mathrm{L}-\mathcal{L})\exp\left(constant^{\frac{1}{2}}\right)$$

These operations are shown in the code fragment in Figure 11.

The values u_0 and u_H were computed using the code fragment in Figures 9 and 10. u_0 and u_H returned a value of N/A when the case did not fall under Case1.

TownerModel-27APR16.xlsm - Module1 (Code)

(General) Case3

```
'Calculating uSub0 to solve for H for case 1
Function uSub0(mValue, p, k, a, capA)

    If k = 0 Then GoTo ErrorHandlerDivideByZero
    If mValue = "N/A" Then
        uSub0 = "N/A"
    Else
        uSub0 = (-mValue) * (((p / k) * a) + (capA / 2#))
    End If
    Exit Function

ErrorHandlerDivideByZero:
    MsgBox "k cannot be zero when calculating uSub0"
    uSub0 = "ERROR"
    Exit Function

End Function
```

FIGURE 9. FUNCTION USED TO DETERMINE u_0 TO DEFINE MAXIMUM H FOR CASE 1

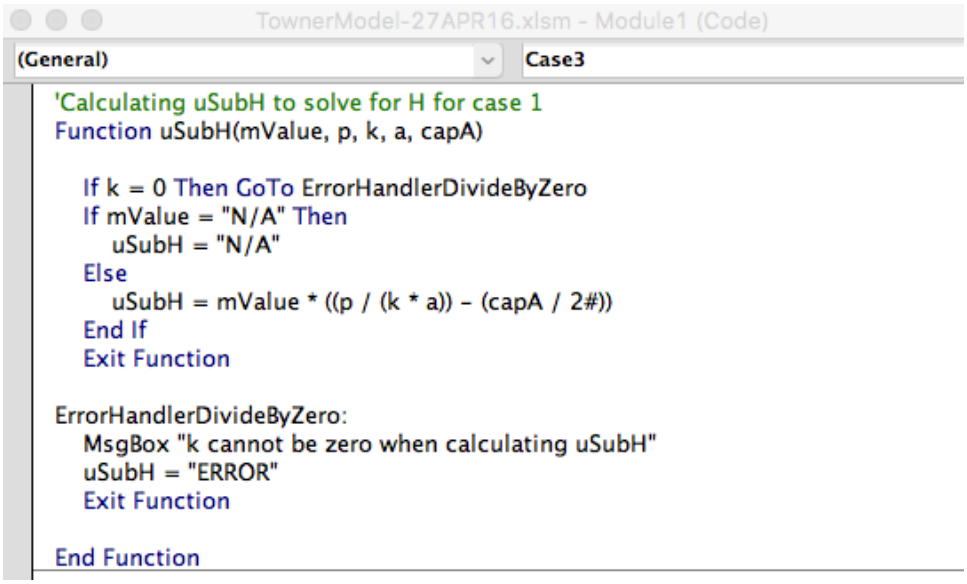

FIGURE 10. FUNCTION USED TO DETERMINE u_H TO DEFINE MAXIMUM H FOR CASE 1

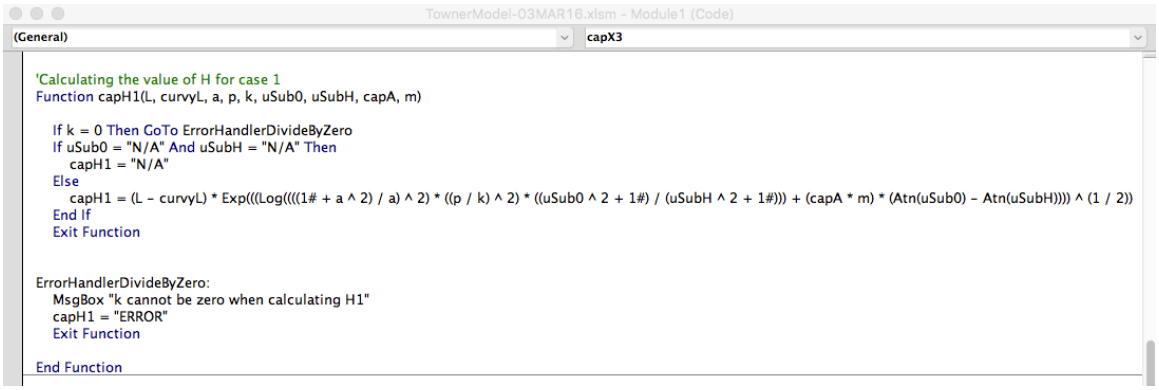

FIGURE 11. FUNCTION USED TO DETERMINE MAXIMUM H FOR CASE 3

By putting $\frac{dz}{dz} = 0$, the formula for *x*, the distance which the maximum H occurs, for Case 1 can be obtained. The maximum at which the mound occurs on the sloping bed allows for the shape of the mound to be better understood and this program. Equation 7 in Towner seen below was used to find the value of x (1975):

$$X = \mathcal{L} + \left(\frac{k}{p}\right) * \left(\frac{a}{1 + a^2}\right) * H$$

Figure 12 shows the code fragment for this function and Figure 12 shows the code fragment for the determination of *u*.

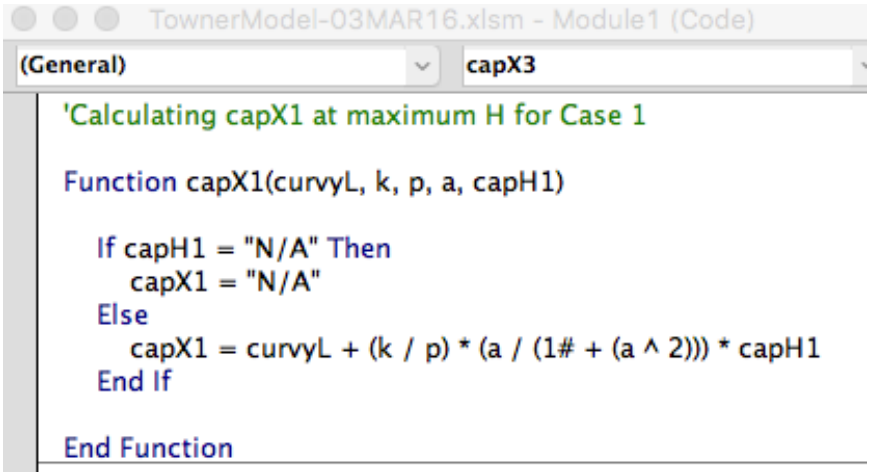

FIGURE 12. FUNCTION USED TO DETERMINE X AT MAXIMUM H FOR CASE 1

The value of u was calculated in order for a relationship of $\mathcal{L}/L$ to be found.

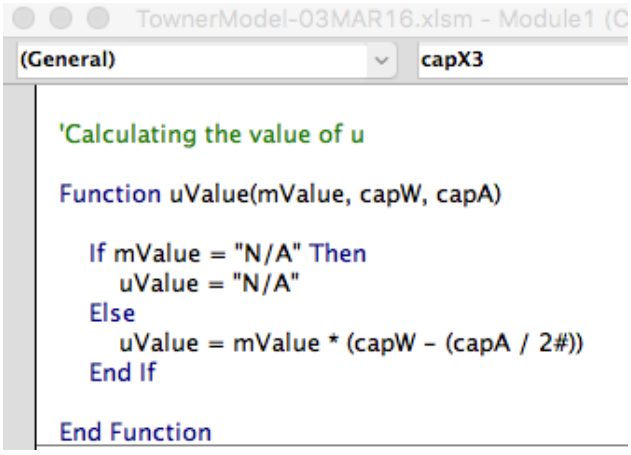

FIGURE 13. FUNCTION USED TO DETERMINE THE VALUE OF U

Figure 14 represents the analytical function for Case 1, which is given by Equation 3 in Towner (1975). All variables were moved to one side of the equation in order to be equal to zero. Thus allowing this function to be used for estimation with Newton's Method.

```
'Equation 3 in Towner, this function used to get z from supplied x under case 1
Function Case1(mValue, Zguess, x, curvyL, p, k, a, capA, capH1, vValue)

  If vValue < 0 And (capA) ^ 2 < 4 * (p / k) Then
    Case1 = Log(((mValue) ^ 2) * ((Zguess - ((x - curvyL) * ((p / k) * a + (capA / 2#)))) ^ 2) + ((x - curvyL) ^ 2)) + (capA * mValue * Atn(mValue * ((Zguess / (x - curvyL)) - ((p / k) * a) - (capA / 2#))))
      - Log(((m) ^ 2) * (((p / (k * a)) - (capA / 2#)) ^ 2) + 1) - (capA * mValue * Atn(mValue * ((p / (k * a)) - (capA / 2#)))) - Log((((k) ^ 2) * ((a) ^ 2) * ((capH1) ^ 2)) / (((p) ^ 2) * ((1 + ((a) ^ 2)) ^ 2)))
      + (capA * mValue * Pi)

  ElseIf vValue > 0 And (capA) ^ 2 < 4 * (p / k) Then
    Case1 = Log(((mValue) ^ 2) * ((Zguess - ((x - curvyL) * ((p / k) * a + (capA / 2#)))) ^ 2) + ((x - curvyL) ^ 2)) + (capA * mValue * Atn(mValue * ((Zguess / (x - curvyL)) - ((p / k) * a) - (capA / 2#))))
      - Log(((m) ^ 2) * (((p / (k * a)) - (capA / 2#)) ^ 2) + 1) - (capA * mValue * Atn(mValue * ((p / (k * a)) - (capA / 2#)))) - Log((((k) ^ 2) * ((a) ^ 2) * ((capH1) ^ 2)) / (((p) ^ 2) * ((1 + ((a) ^ 2)) ^ 2)))
  Else
    Case1 = "N/A"
  End If
End Function
```

FIGURE 14. FUNCTION OF EQUATION 3 IN TOWNER USED TO GET X AS A FUNCTION OF Z FOR CASE 1

Figure 15 shows the Newton's estimation of z for Case 1 using the function shown in Figure 14. This code fragment represents a loop function and also calls the Case1

function. It must be recognized that when calling a function, the function cannot be listed as a parameter of the function.

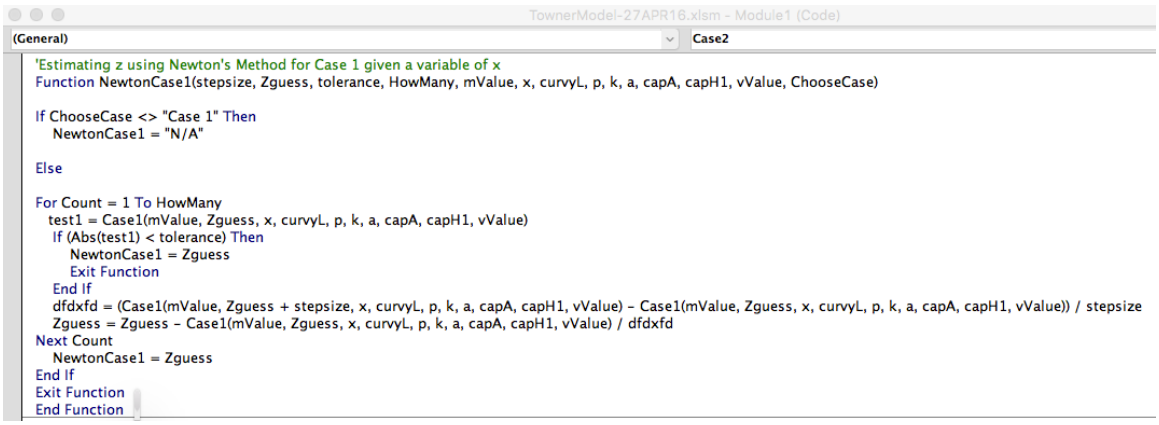

FIGURE 15. LOOP FUNCTION ESTIMATING Z WITH NEWTON'S METHOD FOR CASE 1 USING A SUPPLIED X VALUE

In order to do a loop function, and in this case Newton's method estimation, the algorithm is as follows:

1. Write the function name, which ultimately will return the estimated value of *z*, and the parameters needed for the estimation and for the analytical function.

```
Function EstimationValue(stepsize, tolerance, HowMany,
Constants, Guess)
```

2. Code the Count statement and specify test1. Make sure when calling function that parameters are in same order from original function.

```
For Count = 1 to HowMany
  test1 = Function(Constants, Guess)
```

3. Create an IF statement that checks if the absolute value of the current guess at the specified count is smaller than the tolerance.

```
If (Abs(test1) < tolerance) Then
    EstimationValue = Guess
   Exit Function
End If
```

4. If the current guess is not smaller than the guess, Newton's method equations are used. The derivative first needs to be found for the function and then the new guess is found.

```
dfdxfd = ((Function(Constants, Guess+stepsize)-
   (Function(Constants, Guess)))/stepsize
Guess = Guess - (Function(Constants, Guess)/dfdxfd)
```

5. Finally, using the new guess, the function goes to the next count, loops back to the beginning and checks if the new guess is below the tolerance.

```
Next Count
   EstimationValue = Guess
   End If
Exit Function
End Function
```

CASE 2

If the case falls into Case 2, a new H had to be manipulated and calculated for. Equation 8 on page 145 of Towner (1975) needed to be solved for H. To do this, both sides of the equation were multiplied by L. You can see this manipulation for the H for case 2 below:

$$\frac{H}{L} = -(1+a^2) * \left(\frac{p}{k}\right) * \left(\frac{aA+2}{A-2a}\right) * \exp\left(\frac{2A*(1+a^2)}{(A-2a)*(Aa+2)}\right)$$

$$H = -L * (1+a^2) * \left(\frac{p}{k}\right) * \left(\frac{aA+2}{A-2a}\right) * \exp\left(\frac{2A*(1+a^2)}{(A-2a)*(Aa+2)}\right)$$

Now, with height of the mound found for Case 2, the location *x* can be found. The equation is the same as before for Case 1 but instead $\mathcal{L}$ is equal to zero. This results in the equation following:

$$X = \left(\frac{k}{p}\right) * \left(\frac{a}{1+a^2}\right) * H$$

The following VBA scripts in Figure 16 and 17 represent the functions to find H and *x* for Case 2.

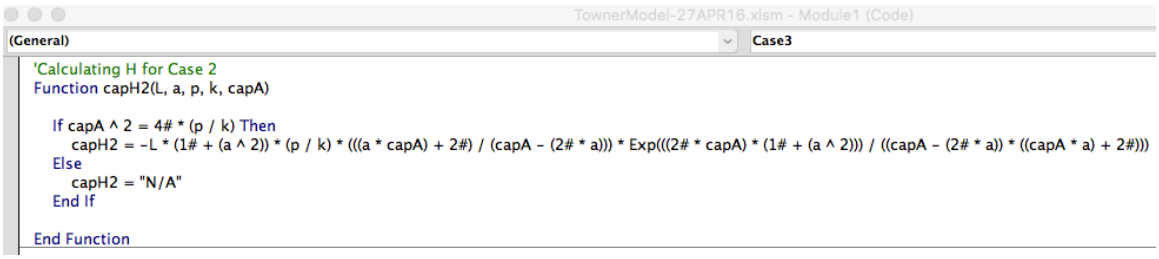

FIGURE 16. FUNCTION USED TO DETERMINE MAXIMUM H FOR CASE 2

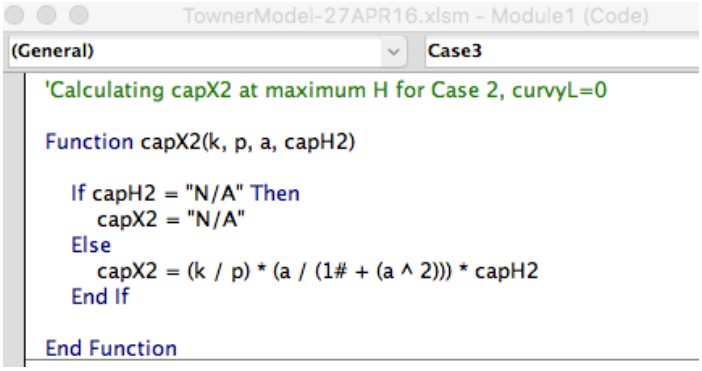

FIGURE 17. FUNCTION USED TO DETERMINE X AT MAXIMUM H FOR CASE 2

Figure 18 shows the analytical function for Case 2 represented by Equation 9 of Towner (1975). Figure 19 shows the Newton estimation of *z* for Case 2.

```
'Equation 9 in Towner, this function used to get z from supplied x under Case 2
Function Case2(capA, a, p, k, x, L, Zguess, vValue)

  If (capA) ^ 2 = 4 * (p / k) Then
     Case2 = ((Log(-(((Zguess / vValue) - (a * (p / k))) - (capA / 2#)) / (a * (p / k) + (capA / 2)))) ^ 2) - ((capA * (((Zguess / vValue) - (a * (p / k))) + (a * (p / k)))) / (a * (p / k) +
        (capA / 2#)) * (((Zguess / vValue) - (a * (p / k))) - (capA / 2#))) + ((Log(x / L)) ^ 2)
  Else
     Case2 = "N/A"
  End If
End Function
```

FIGURE 18. FUNCTION OF EQUATION 9 IN TOWNER USED TO GET X AS A FUNCTION OF Z FOR CASE 2

TownerModel-27APR16.xlsm - Module1 (Code)

(General) | Case3

```
'Estimating z using Newton's Method for Case 2 given a variable of x
Function NewtonCase2(stepsize, tolerance, HowMany, capA, a, p, k, x, L, Zguess, vValue, ChooseCase)

If ChooseCase <> "Case 2" Then
   NewtonCase2 = "N/A"

Else

For Count = 1 To HowMany
  test1 = Case2(capA, a, p, k, x, L, Zguess, vValue)
   If (Abs(test1) < tolerance) Then
      NewtonCase2 = Zguess
      Exit Function
   End If
   dfdxfd = (Case2(capA, a, p, k, x, L, Zguess + stepsize, vValue) - Case2(capA, a, p, k, x, L, Zguess, vValue)) / stepsize
   Zguess = Zguess - Case2(capA, a, p, k, x, L, Zguess, vValue) / dfdxfd
Next Count
   NewtonCase2 = Zguess
End If
Exit Function
End Function
```

FIGURE 19. LOOP FUNCTION ESTIMATING Z WITH NEWTON'S METHOD FOR CASE 2 USING A SUPPLIED X VALUE

CASE 3

The final case is where $A^2 > 4(p/k)$. First, the values of W_1 and W_2 were needed in order to complete the relationship of Equation 11 in Towner (1975):

$$\frac{W_1}{W_1 - W_2} * \ln\left(z - x\left(W_1 + a\left(\frac{p}{k}\right)\right)\right)^2 - \frac{W_2}{W_1 - W_2} * \ln\left(z - x\left(W_2 + a\left(\frac{p}{k}\right)\right)\right)^2 = C_4$$

Where,

$$W_1 = \frac{A}{2} + \left(\frac{A^2}{4} - \frac{p}{k}\right)^{1/2}$$

$$W_2 = \frac{A}{2} - \left(\frac{A^2}{4} - \frac{p}{k}\right)^{1/2}$$

$$C_4 = \frac{W_1}{W_1 - W_2}\ln\left(H\left(\frac{1}{1+a^2} - W_1\left(\frac{a}{1+a^2}\right)\left(\frac{k}{p}\right)\right)\right)^2 - \frac{W_2}{W_1 - W_2}\ln\left(H\left(\frac{1}{1+a^2} - W_2\left(\frac{a}{1+a^2}\right)\left(\frac{k}{p}\right)\right)\right)^2$$

The VBA script of W_1 and W_2 is seen in Figures 20 and 21.

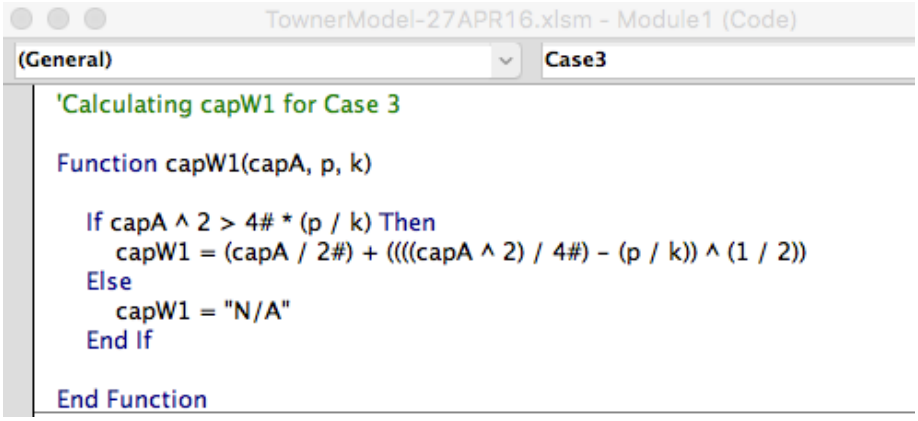

FIGURE 20. FUNCTION USED TO DETERMINE W_1 TO DEFINE C_4 FOR CASE 3

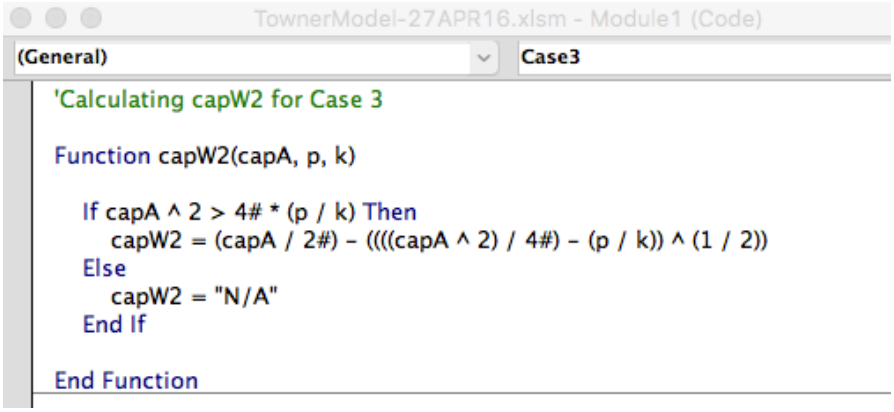

FIGURE 21. FUNCTION USED TO DETERMINE W_2 TO DEFINE C_4 FOR CASE 3

By manipulating this C_4 relationship with W_1 and W_2 and putting z equal to zero at x equal to L, a relationship for H for case 3 was determined. Below the manipulation for H for Case 3 can be seen.

$$H = L * \left(\frac{\left(\frac{1}{1+a^2}\right) - W_2\left(\frac{a}{1+a^2}\right)\left(\frac{k}{p}\right)}{a\left(\frac{p}{k}\right) + W_2} \right)^{W_2/(W_1 - W_2)} * \left(\left(\frac{\left(\frac{1}{1+a^2}\right) - W_1\left(\frac{a}{1+a^2}\right)\left(\frac{k}{p}\right)}{a\left(\frac{p}{k}\right) + W_1} \right)^{W_1/(W_1 - W_2)} \right)^{-1}$$

Again, $\mathcal{L}$ is equal to zero for Case 3 as it was for Case 2. So the same equation used to find x in case 2 was used. Figures 22 and 23 shows the code fragment used to determine H and x for Case 3.

```
'Calculating H for Case 3
Function capH3(L, a, capW2, k, p, capW1)

  If capW1 = "N/A" And capW2 = "N/A" Then
    capH3 = "N/A"
  Else
    capH3 = L * ((((1# / (1# + (a ^ 2))) - (capW2 * (k / p) * (a / (1# + (a ^ 2))))) / ((a * (p / k)) + capW2)) ^ (capW2 / (capW1 - capW2))) * (((((1# / (1# + (a ^ 2)))
        - (capW1 * (k / p) * (a / (1# + (a ^ 2))))) / ((a * (p / k)) + capW1)) ^ (capW1 / (capW1 - capW2))) ^ (-1))
  End If
End Function
```

FIGURE 22. FUNCTION USED TO DETERMINE MAXIMUM H FOR CASE 3

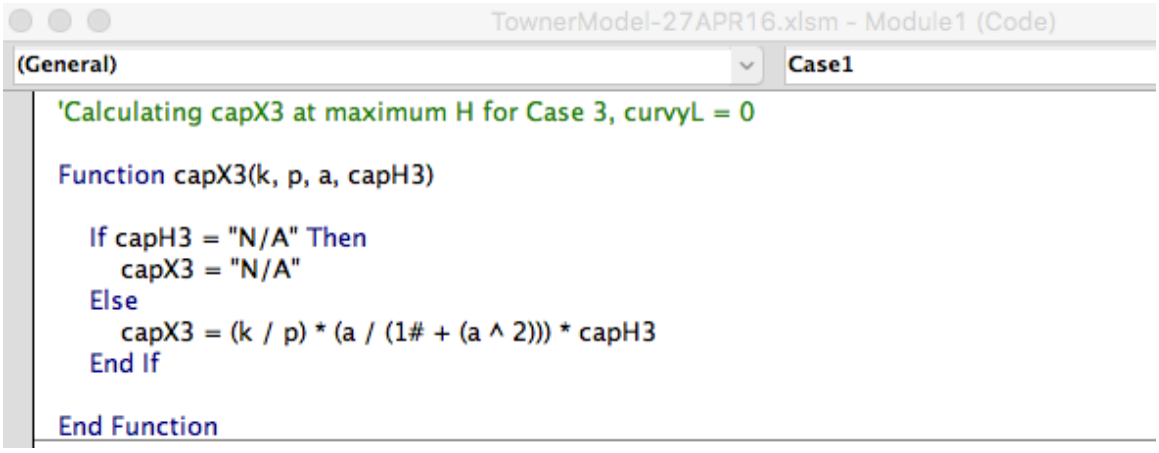

FIGURE 23. FUNCTION USED TO DETERMINE X AT MAXIMUM H FOR CASE 3

In order to use Newton's method for the analytical solution in Case 3, C_4 needed to be determined. Figure 24 shows the code fragment for the C_4 function.

```
'Calculating capC4 for case 3
Function capC4(capA, capW1, capW2, capH3, a, k, p)

  If capA ^ 2 > 4# * (p / k) Then
    capC4 = ((capW1 / (capW1 - capW2)) * (Log((capH3 * ((1 / (1 + (a ^ 2)))) - (capW1 * (a / (1 + (a ^ 2))) * (k / p)))) ^ 2)) - ((capW2 / (capW1 - capW2)) * (Log((capH3 * ((1 / (1 + (a ^ 2))))
        - (capW2 * (a / (1 + (a ^ 2))) * (k / p)))) ^ 2))
  Else
    capC4 = "N/A"
  End If
```

FIGURE 24. FUNCTION USED TO DETERMINE C_4

Figure 25 represents the analytical function for Case 3 taken from Equation 11 in Towner (1975). Figure 26 shows the loop function estimating *z* for Case 3 using Newton's method.

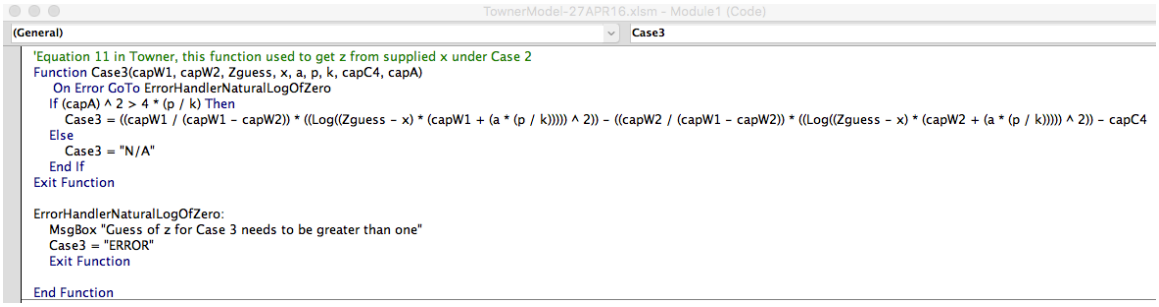

FIGURE 25. FUNCTION OF EQUATION 11 IN TOWNER USED TO GET X AS A FUNCTION OF Z FOR CASE 3

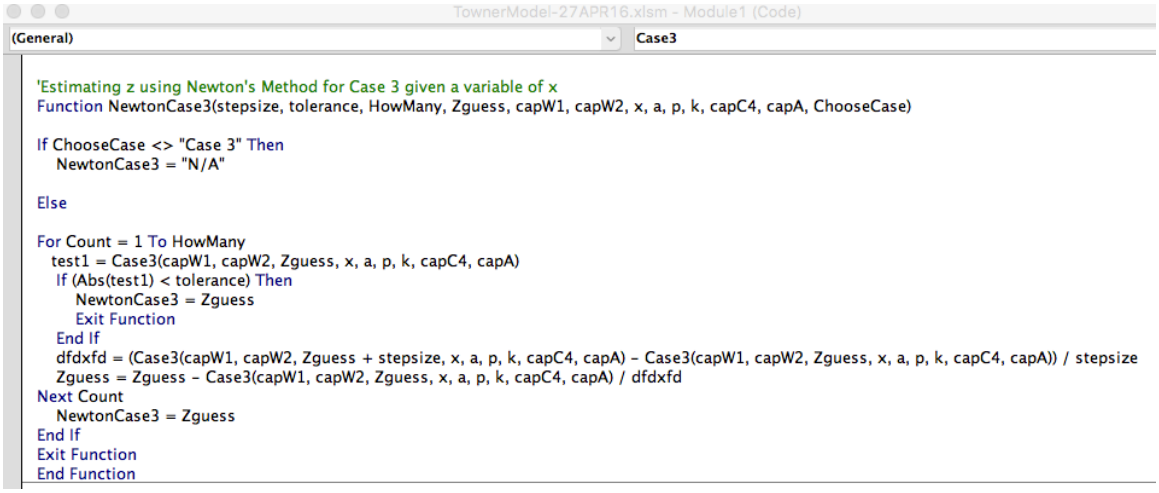

FIGURE 26. LOOP FUNCTION ESTIMATING Z WITH NEWTON'S METHOD FOR CASE 2 USING A SUPPLIED X VALUE

Lastly, Figure 27 shows the code fragment used to clean up the spreadsheet interface. This simply made the original input box for *z* equal to the estimated z value for whichever case was governing.

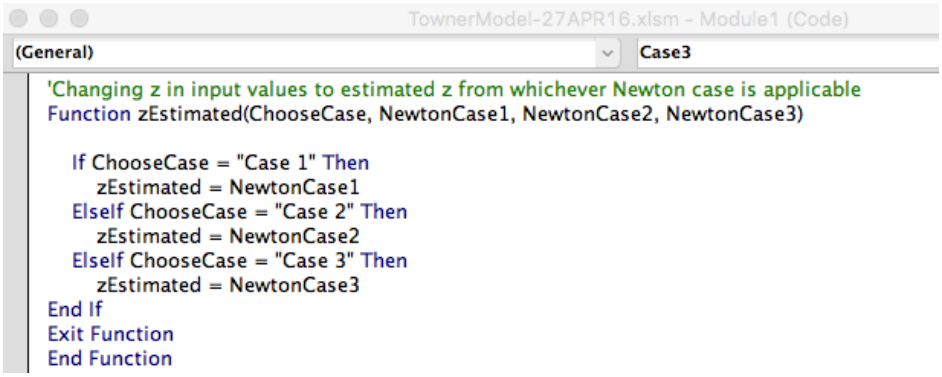

Figure 27. Changing z Input to z Estimated for Applicable Case

SPREADSHEET INTERFACE

Figures 28, 29, and 30 show the spreadsheet interface for Case 1, 2 and 3. In these spreadsheets the estimated *z* values can be seen. It should be noted that each function value is approximately zero, proving a critical point was found.

Input	Value	
a	3.00	bed slope (degrees)
p	3.00	rainfall rate (in/hr)
k	2.00	hydraulic conductivity (in/hr)
x	1.00	horizontal distance measured from upper drain
z	1.574224859	vertical distance of the water table above the sloping bed
L	1.00	horizontal distance between the drains
$\mathcal{L}$	0.50	horizontal distance to the watershed

Choose Case	Case 1

Defining Values	
A	-1.50
ν	0.5
W	-1.351550281

Case 1 Values	
m	1.03279556
u_0	-3.8729833
u_H	1.29099445
H_1	9.17368601
u	-0.6212785
X_1	2.3347372

Case 2 Values	
H_2	N/A
X_2	N/A

Case 3 Values	
W_1	N/A
W_2	N/A
H_3	N/A
X_3	N/A
C_4	N/A

Newton guessing	
Stepsize	0.001
tolerance	0.000001
how many	200

Case 1A "z" Estimation		
z Guess	z Estimated	Function Value
1	1.57422486	-1.55692E-08

Case 2 "z" Estimation		
z Guess	z Estimated	Function Value
1	N/A	N/A

Case 3 "z" Estimation		
z Guess	z Estimated	Function Value
1.01	N/A	N/A

FIGURE 28. EXCEL SPREADSHEET SHOWING CASE 1 Z ESTIMATION CORRECTLY WORKING

Input	Value	
a	-2.40	bed slope (degrees)
p	9.00	rainfall rate (in/hr)
k	4.00	hydraulic conductivity (in/hr)
x	1.00	horizontal distance measured from upper drain
z	0.000000001	vertical distance of the water table above the sloping bed
L	1.00	horizontal distance between the drains
$\mathcal{L}$	0.50	horizontal distance to the watershed

Choose Case	Case 2

Defining Values	
A	3.00
ν	0.5
W	5.400000002

Case 1 Values	
m	N/A
u_0	N/A
u_H	N/A
H_1	N/A
u	N/A
X_1	N/A

Case 2 Values	
H_2	3.730297533
X_2	-0.588607106

Case 3 Values	
W_1	N/A
W_2	N/A
H_3	N/A
X_3	N/A
C_4	N/A

Newton guessing	
Stepsize	0.001
tolerance	0.000001
how many	300

Case 1A "z" Estimation		
z Guess	z Estimated	Function Value
1	N/A	N/A

Case 2 "z" Estimation		
z Guess	z Estimated	Function Value
1.25	9.22201E-10	5.5332E-09

Case 3 "z" Estimation		
z Guess	z Estimated	Function Value
1.01	N/A	N/A

FIGURE 30. EXCEL SPREADSHEET SHOWING CASE 2 Z ESTIMATION CORRECTLY WORKING

Input	Value	
a	10.00	bed slope (degrees)
p	3.00	rainfall rate (in/hr)
k	2.00	hydraulic conductivity (in/hr)
x	1.00	horizontal distance measured from upper drain
z	1.061661213	vertical distance of the water table above the sloping bed
L	1.00	horizontal distance between the drains
$\mathcal{L}$	0.50	horizontal distance to the watershed

Choose Case	Case 3

Defining Values	
A	-5.00
ν	0.5
W	-12.87667757

Case 1 Values	
m	N/A
u_0	N/A
u_H	N/A
H_1	N/A
u	N/A
X_1	N/A

Case 2 Values	
H_2	N/A
X_2	N/A

Case 3 Values	
W_1	-0.320550528
W_2	-4.679449472
H_3	26.5825201
X_3	1.754621789
C_4	0.21856185

Newton guessing	
Stepsize	0.001
tolerance	0.000001
how many	200

Case 1A "z" Estimation		
z Guess	z Estimated	Function Value
1	N/A	N/A

Case 2 "z" Estimation		
z Guess	z Estimated	Function Value
1	N/A	N/A

Case 3 "z" Estimation		
z Guess	z Estimated	Function Value
1.01	1.061661213	4.1627E-07

FIGURE 29. EXCEL SPREADSHEET SHOWING CASE 3 Z ESTIMATION CORRECTLY WORKING

REFERENCES

Towner, G. D., Drainage of Groundwater Resting on a Sloping Bed with Uniform Rainfall, *Water Resour. Res.,,* 11.1, 144-47, 1975, Print.

Childs, E. C., Drainage of Groundwater Resting on a Sloping Bed, *Water Resour. Res.*, 7(5), 1256-1263, 1971.

Schmid, P., and J. N. Luthin, The Drainage of Sloping Lands, *J. Geophys. Res.*, 69(8), 1525-1529, 1964.

Werner, P. W., Some Problems in Non-Artesian Ground-Water Flow, *Eos Trans. AGU*, 38(4), 511-518, 1957.