Ryan Jaipersaud

Final Project Report

ChE 352

May 11th, 2017

Hours Spent: 60 hr

Ryan Jaipersaud
ChE 352
Final Project Part I

Memorandum

TO: Mr. Sean F. Askebom

FROM: Ryan Jaipersaud

DATE: May 11, 2017

SUBJECT: Fitting vapor pressure to an explicit form

A numerical method was needed to produce reliable results for vapor pressure over a wide range of temperature values without the need for finding the vapor pressure at each temperature through experimentation which is time consuming and costly. The vapor pressure of four species in bars was fitted using the cftool window in matlab. Polyfit was initially used but, it returned negative pressures at temperatures approaching 170 K for some species as well as larger relative errors compared to cftool. Lagrange polynomials produced large relative errors for temperatures that were not used to construct the polynomial and were time consuming to derive. Cftool uses regression to fit the data to an explicit equation which was faster and gave lower error than the two methods previously mentioned. The equation used was the Antoine equation as shown below.

$$lnP = A - \frac{B}{T + C}$$

The natural log of vapor pressure for each species at given temperatures from Perry's Handbook was calculated and given to the cftool. From there the cftool used regression to solve for parameters A, B and C such that error is minimized. R^2 equaled 1 for all four fits. The equations for species A (ethane), C (vinyl chloride monomer), D (isobutene), and E (1-butene) are shown below.

[VaporPressure] = exp(9.568-(1706/(T-6.064))), for species A between 170 and 305.4 K

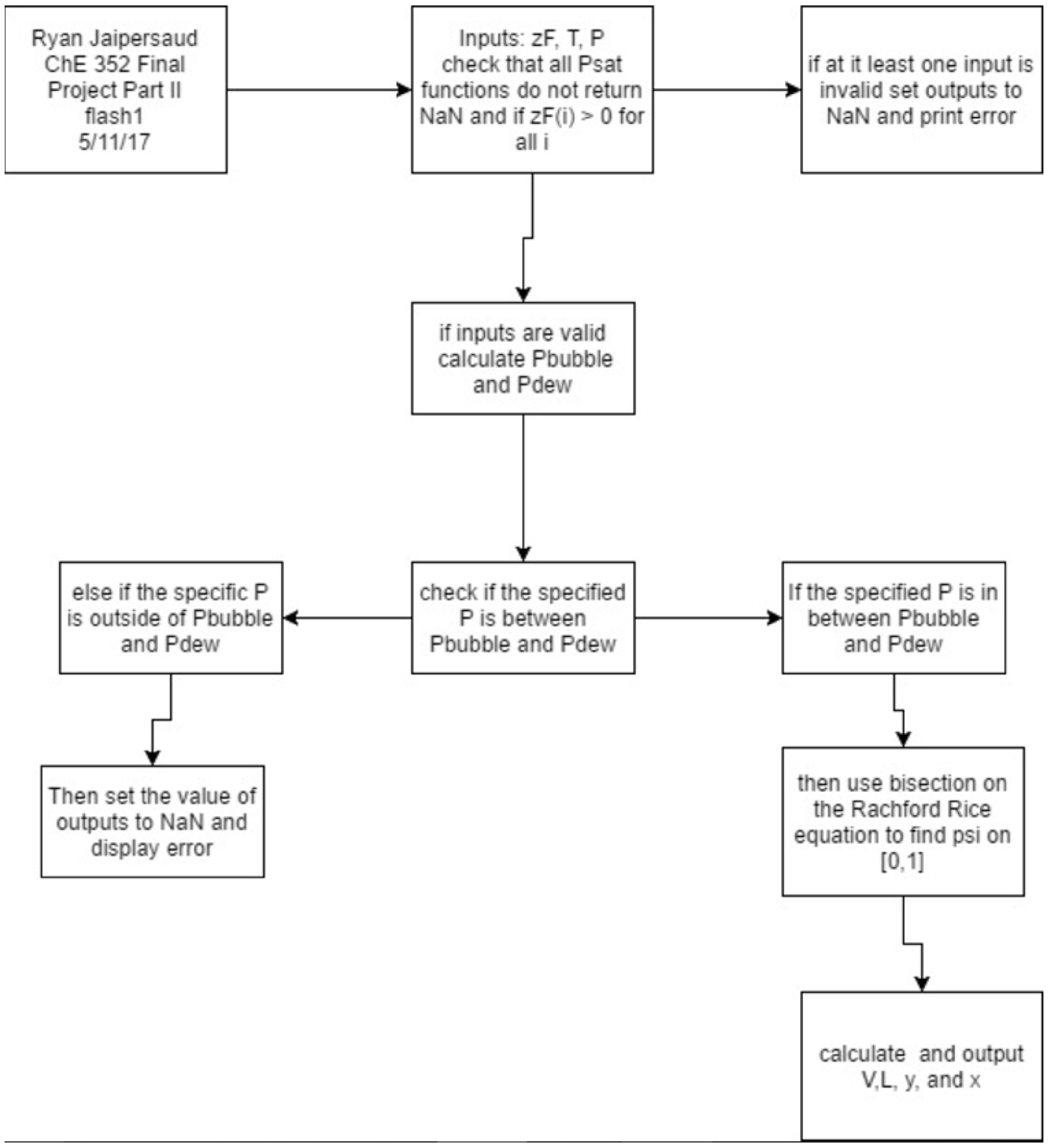
[VaporPressure] = exp(9.001 -(1946/(T-43.15))), for species C between 170 and 432 K

[VaporPressure] = exp(9.186 -(2128/(T-29.53))), for species D between 170 and 408 K

[VaporPressure] = exp(9.331 -(2190/(T-32.18))), for species E between 170 and 418.6 K

The vapor pressure equations for each species returned lower relative errors over the temperature used to construct each fit than those of polyfit. The equations for species A, C, D and E can be accessed by making functions calls to Psat1(T), Psat2(T), Psat3(T), and Psat4(T), respectively. If the input temperature is not within the ranges specified above the functions will return NaN. The number of points that went into constructing each fit were 8, 13, 12, and 11, respectively. The relative error at each point was calculated. The average percent relative error over the number of points to make each curve was calculated to be 0.624, 0.776, 0.827 and 0.68 %, respectively.

Ryan Jaipersaud
ChE 352
Final Project Part II
flash1

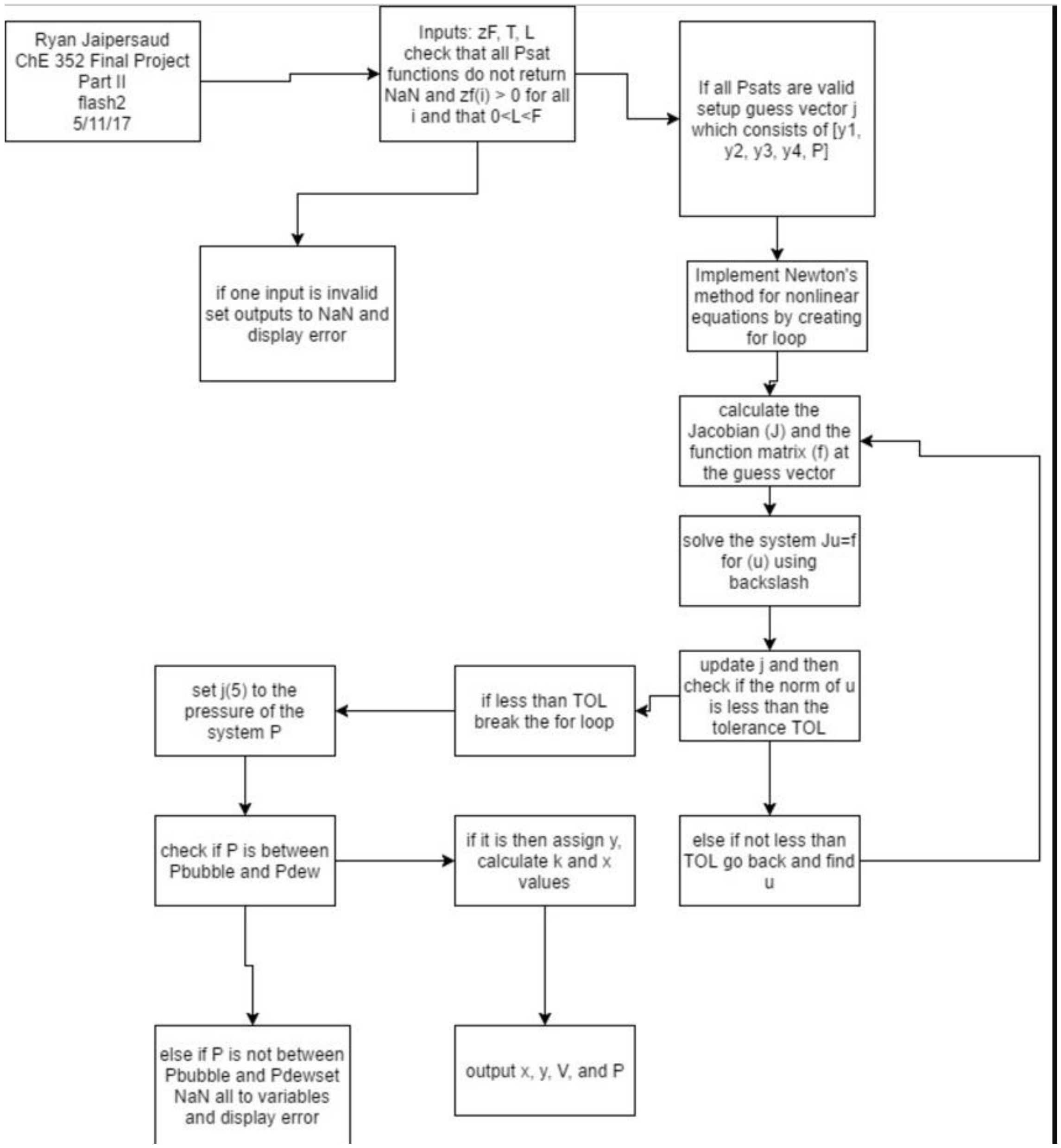

Ryan Jaipersaud ChE 352 Final Project Part II flash1 5/11/17
Inputs: zF, T, P check that all Psat functions do not return NaN and if zF(i) > 0 for all i
if at it least one input is invalid set outputs to NaN and print error
if inputs are valid calculate Pbubble and Pdew
check if the specified P is between Pbubble and Pdew
else if the specific P is outside of Pbubble and Pdew
If the specified P is in between Pbubble and Pdew
Then set the value of outputs to NaN and display error
then use bisection on the Rachford Rice equation to find psi on [0,1]
calculate and output V,L, y, and x

Ryan Jaipersaud
ChE 352
Final Project Part II
flash1

When implementing flash1, the user must input the feed, operating temperature and pressure of the flash unit. The program needs to check if a feed exists and that each element in zF is positive which it does by taking the summation of zF and by directly checking each element. To make sure that all vapor pressures are valid function calls to Psat# are made. Should the specified temperature not be within the bounds for each species the functions will return NaN. The “isnan()” function takes in an arguments that returns 1 if NaN is passed in and this is used to make sure all Psat functions are valid and don’t fall through the code which would give error messages.

To determine whether or not the conditions specified lead to flashing the bubble point and dew point pressures referred to as Pbubble and Pdew, respectively, are calculated. A check is made to make sure the pressure input (P) falls in between these values. If (P) is between Pbubble and Pdew then the code will proceed to build the Rachford Rice equation [1]. This is an equation of one unknown, psi, and therefore bisection is used to determine psi. The bounds of psi are fixed from 0 to 1 since psi is defined as vapor fraction. The code then calculates x, y, V, and L. If P is not between Pbubble and Pdew an error message will be displayed and all outputs are set to NaN. The code outputs x, y, V, and L.

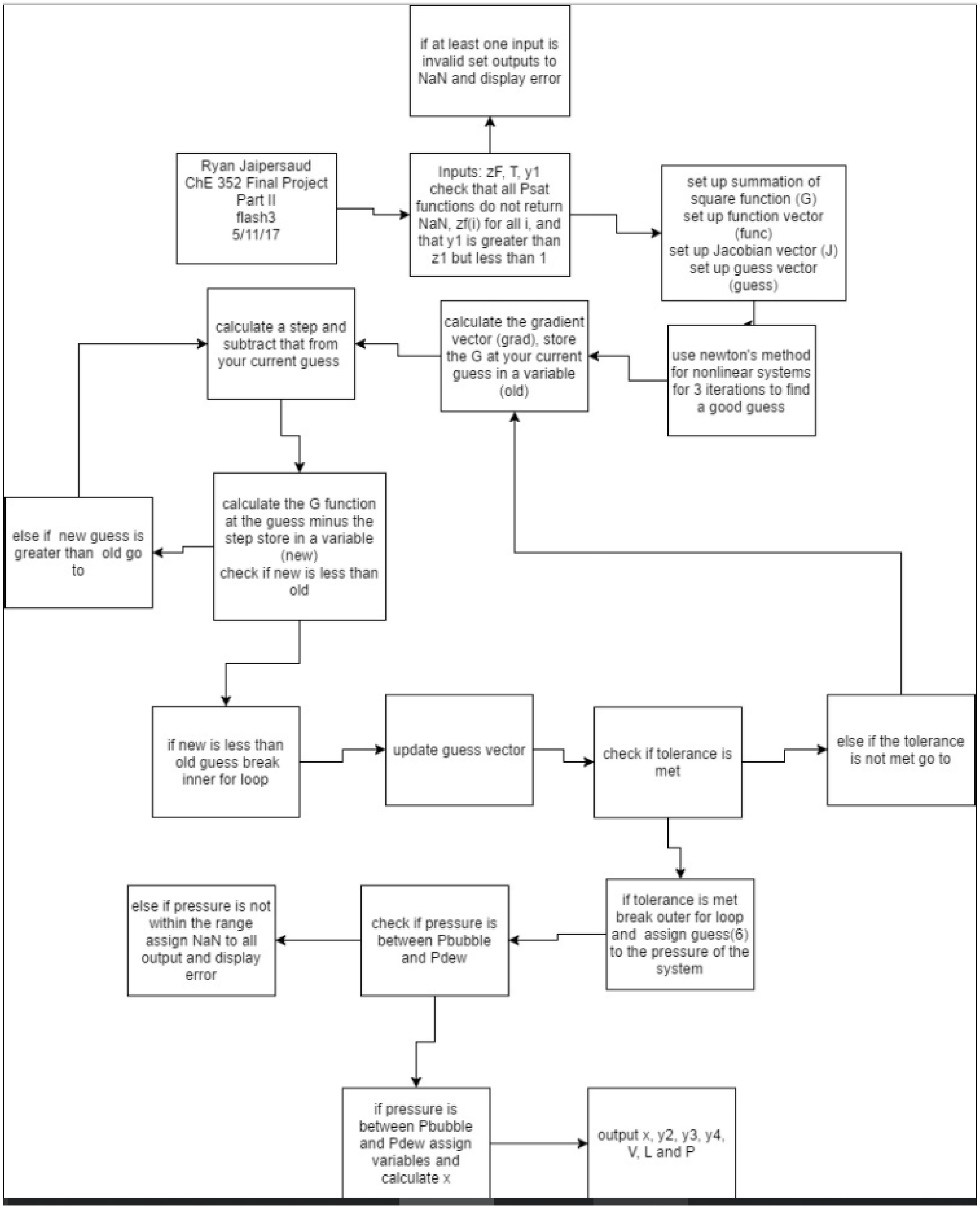
Ryan Jaipersaud
ChE 352
Final Project Part II
flash2


Ryan Jaipersaud ChE 352 Final Project Part II flash2 5/11/17
Inputs: zF, T, L check that all Psat functions do not return NaN and zf(i) > 0 for all i and that 0<L<F
If all Psats are valid setup guess vector j which consists of [y1, y2, y3, y4, P]
if one input is invalid set outputs to NaN and display error
Implement Newton's method for nonlinear equations by creating for loop
calculate the Jacobian (J) and the function matrix (f) at the guess vector
solve the system Ju=f for (u) using backslash
update j and then check if the norm of u is less than the tolerance TOL
if less than TOL break the for loop
set j(5) to the pressure of the system P
check if P is between Pbubble and Pdew
if it is then assign y, calculate k and x values
else if not less than TOL go back and find u
else if P is not between Pbubble and Pdewset NaN all to variables and display error
output x, y, V, and P

Ryan Jaipersaud
ChE 352
Final Project Part II
flash2

When implementing flash2, the user must input the feed, operating temperature and liquid exit stream of the flash unit. Flash2 starts off the same way as flash1 making the same checks for feed and Psat# functions. But, now it also checks to see if L is between 0 and F. V can be calculated without the need to do anything. This method uses Newton's method for nonlinear equations to solve the system of equations. In order to do this a guess vector must be set up which consists of, j = [y1,y2,y3,y4,P]. As a result the Jacobian matrix (J) for the problem is a 5 x 5 and the function vector (f) is 5 x 1. The system Ju=f is solved repeatedly for (u) using the backslash operator until the two norm of (u) is less than the 10^{-11}. The guess vector is repeatedly updated by subtracting (u) from the original guess. Once the tolerance for (u) is met the code breaks out of the for loop and assigns j(5) to the pressure of the system. The code then checks if the pressure is between the bubble and dew point pressures. If it is then y is assigned appropriate values and k and x values are calculated. If it is not between the points then set all outputs to NaN. The code outputs x, y, V, P.

Ryan Jaipersaud
ChE 352
Final Project Part II
flash3


Ryan Jaipersaud
ChE 352 Final Project
Part II
flash3
5/11/17
Inputs: zF, T, y1
check that all Psat functions do not return NaN, zf(i) for all i, and that y1 is greater than z1 but less than 1
if at least one input is invalid set outputs to NaN and display error
set up summation of square function (G)
set up function vector (func)
set up Jacobian vector (J)
set up guess vector (guess)
use newton's method for nonlinear systems for 3 iterations to find a good guess
calculate the gradient vector (grad), store the G at your current guess in a variable (old)
calculate a step and subtract that from your current guess
calculate the G function at the guess minus the step store in a variable (new)
check if new is less than old
else if new guess is greater than old go to
if new is less than old guess break inner for loop
update guess vector
check if tolerance is met
else if the tolerance is not met go to
if tolerance is met break outer for loop and assign guess(6) to the pressure of the system
check if pressure is between Pbubble and Pdew
else if pressure is not within the range assign NaN to all output and display error
if pressure is between Pbubble and Pdew assign variables and calculate x
output x, y2, y3, y4, V, L and P

Ryan Jaipersaud

ChE 352

Final Project Part II

flash3

When implementing flash3, the user must input the feed, operating temperature and the light key of the flash unit. In addition to checking Psat functions and the feed, y1 must also be checked to see if it is greater than z(1) [explanation in Appendix 1]. A (guess) vector is first made consisting of guess = [y2; y3; y4; V; L; P]. Once all checks are met, a 6 x 6 Jacobian matrix function (J), a 6 x 1 function vector (func), and a summation of squares of the elements in (func) function (G) are created. Newton's method for three iterations is first used to create a decent starting guess for steepest decent. The gradient (grad) and (G) are both calculated at the current guess according to [2]. (G) at the current guess is referred to as (old). A nested for loop is then created to find a step that will decrease (G). (G) is calculated at the guess vector minus the step (new). Once (new) is less than (old) a break statement will get the user out of the nested loop and the (guess) vector will be updated by the step that decreased G. A check is then made to see if the step compared to the guess is lower than the tolerance (10^{-10}). If it is lower, a break statement will take the user out of the outer for loop. Guess(6) is assigned to the pressure and the pressure is checked to see if it falls between the bubble and dew point pressures. If the pressure is between the bubble and dew point the code will assign variables and then calculate x. The code outputs x, y2, y3, y4, V, L, and P.

Ryan Jaipersaud
ChE 352
Final Project Part III

1.) The detector must be $\boxed{\text{5.9 cm}}$ from the wall.

2.) This is a second order parabolic differential equation since the determinant of the equation is zero. The Neumann boundary condition in this problem changes the A and B matrixes which did not occur with the differential equations covered in class.

$$D_{MA}\frac{d^2C}{dx^2} = \frac{dC}{dt}$$

$$\det(X) = \det\left(\begin{vmatrix} D_{MA} & 0 \\ 0 & 0 \end{vmatrix}\right) = 0$$

3.) Three boundary conditions are needed, one for time and two for x. They are as follows: C(x,0) = 0, C(0,t) = 18 and $\frac{dC}{dx} = 0$ at the wall.

4.) Since the detector needs to be within 5.9 cm to detect a leak of 0.8 mol/m^3 after 10 seconds of methane, which is a fast diffusing gas, diffusion is deemed slow.

Ryan Jaipersaud
ChE 352
Final Project

References

1.) Seader, J; Henley, E; Roper, D *Separation Process Principles Chemical and Biochemical Operations*, 3rd edition; John Wiley & Sons: Hoboken, NJ, 2011; pp 139 – 170
2.) Faires, J; Burden, R; *Numerical Methods*, 4th edition; Cengage Learning, 2012; pp 427 – 431

Ryan Jaipersaud
ChE 352
Final Project

Appendix 1

The equation below can be derived from mass balance.

$$psi = \frac{z1 - x1}{y1 - x1}$$

Since the bounds on psi are from 0 to 1, the only time psi will equal 1 is when y1 is equal to z1. Therefore this serves as a lower bound for what y1 can be inserted as. If y1 is less than z1, psi becomes physically useless.