Final Project Report

Christopher Chan

ChE 352 - Process Simulation and Mathematical techniques for ChEs

5/15/2021

Hours Spent on Part II: 30

Hours Spent on Part III: 0

The flash1 subroutine uses the Rachford-Rice equation with feed molar component flow (zF), flash temperature (T), and flash pressure (P) to determine the liquid (X) and vapor (y) fractions, and the total liquid (L) and vapor (V) molar flow rates.

When the flash1 subroutine is called with zF, T, and P, the inputs are passed through several conditional statements. These conditional statements ensure that the inputs are physically possible. The first conditional statements check if the input temperature is between 170K and 432K, which are the temperatures from the lowest temperature of available vapor pressure data to the species with the highest critical point. Therefore, the subroutine will print the string, “the input temperature is not valid”, and return nan for input temperatures outside of this range. The second conditional statements check whether the individual zF of each species is positive. Negative zF is not physically possible because there cannot be negative flow entering the flash. This conditional statement indirectly examines whether the flow is zero; flow cannot be zero because there will be no material entering the flash. Therefore, the subroutine will print the string, “the input zF is not valid” and return nan if the species feed low is negative or zero. The third conditional statement verifies that the input flash pressure is positive. A pressure of 0 bar implies that there is a vacuum, which implies that there is no material in the flash tank. Hence, the subroutine will print the string, “the input P is not valid” and return nan if the flash pressure is negative or zero. After checking that the zF, T, and P inputs are valid, the feed fraction is determined by dividing the feed molar component flow by the total feed molar flow. The feed fraction is placed into an array of length 4 called z and will be used to determine the dew and bubble point pressure. Another condition that needs to be satisfied to produce physically meaningful data would be that the input pressure is in vapor-liquid equilibrium (VLE). For the condition to be met, the input pressure needs to be lower than the bubble point pressure and higher than the dew point pressure at the flash temperature. The bubble point and dew point pressure was determined using species saturation pressure at the flash temperature, which was determined using Lagrange polynomials, and scipy.optimize.newton for finding the root of equation **1** and **2**. These equations were added from **Ref 1**. Other root-finding methods could have been used to find the root, but the scipy function is used for convenience. The scipy function uses either Newton’s, secant, or Hailey’s method to determine the root of function.

$$\boldsymbol{Eqn\ 1.} \sum_{i=1}^{4} \frac{P}{P_i^{sat}} * z_i - 1 = 0, \quad \boldsymbol{Eqn\ 2.} \sum_{i=1}^{4} \frac{P_i^{sat}}{P} * z_i - 1 = 0, \quad \boldsymbol{Eqn\ 3.} \sum_{i=1}^{4} \frac{(K_i - 1) * z_i}{1 + \psi(K_i - 1)} = 0$$

The subroutine prints the string “input pressure is not in vapor-liquid equilibrium” and either the string “the pressure is above the bubble point” or “the pressure is below the dew point”. This condition has to be satisfied because a flash condition suited for one phase will not separate the solution into vapor and liquid components.

After checking that the flash pressure is in VLE, an array, called K, with a length of 4 is created with K_i, where i = 1, 2, 3, and 4. The elements of array K will be extensively used for the Rachford-Rice (RR) equation (Eqn. **3**), which is discussed later. Two arrays with a length of 4, called X and y, are generated for the vapor and liquid fractions, respectively. Furthermore, the ratio of total liquid molar flow over total feed molar flow rate (F), called psi, is determined by finding the root of the rewritten RR equation. The RR equation was rewritten to a polynomial form for ease of root determination because there are no discontinuities that can break the root-finding method (Eqn. **4**). Similar to finding the bubble and dew point pressure, scipy.optimize.newton is used to determine the psi from the RR equation. The V is determined by multiplying psi and F and the L is determined by subtracting V from F.

The liquid and vapor fractions of each species are calculated using reexpressed RR equations (Eqn. **5** and **6**). These equations are placed in a for loop and iterated for each species. The X, y, V, and L are compiled into an array is returned. The subroutine ends.

$$\boldsymbol{Eqn\ 4.}\ (K_1 - 1) * z_1 * (1 + \psi(K_2 - 1)) * (1 + \psi(K_3 - 1)) * (1 + \psi(K_4 - 1)) + (K_2 - 1) * z_2 * (1 + \psi(K_1 - 1)) * (1 + \psi(K_3 - 1)) * (1 + \psi(K_4 - 1)) + (K_3 - 1) * z_1 * (1 + \psi(K_1 - 1)) * (1 + \psi(K_2 - 1)) * (1 + \psi(K_4 - 1)) + (K_4 - 1) * z_1 * (1 + \psi(K_1 - 1)) * (1 + \psi(K_2 - 1)) * (1 + \psi(K_3 - 1))\ where\ K_i = \sum_i^4 \frac{P_i^{sat}(T)}{P}$$

$$\boldsymbol{Eqn\ 5.}\ x_i = \frac{z_i}{1 + \psi(K_i - 1)}, \qquad \boldsymbol{Eqn\ 6.}\ y_i = \frac{z_i * K_i}{1 + \psi(K_i - 1)}$$

The flash2 subroutine uses the simple line search variant of the steepest descent method zF, T, and total liquid molar flow rate (L), to x, y, V, and P.

When the flash2 subroutine is called with zF, T, and L, the inputs are passed through several conditional statements. The first and second conditional statements are the same as flash1. The total feed molar flow rate was determined by summating the feed molar flow rates of the species. The third conditional statement verifies that the input flash L is positive and less than the F. A negative L is not physically possible because there cannot be negative flow exiting the flash tank. Furthermore, the L cannot be greater than F because no reaction occurs; therefore, F = L + V. By this logic, this conditional statement would prevent the output of negative V as that is also not physically possible.

After checking that the zF, T, and L are valid inputs, an array called x, is initialized with initial guesses for the 5 unknowns: x1, x2, x3, x4, and P. These will be renamed x[0] to x[4] based on the described order (Figure **1**). The V is calculated by subtracting F by L because these values are available. The bubble point and dew point pressure was determined using the method mentioned in the flash1 manual. The initial guess for the liquid fractions is 0.25. The initial guess of the pressure is the average of the bubble and dew point pressures at the flash temperature because the subroutine does not converge efficiently. The average pressure is a better initial guess than a randomly selected flash pressure. By this logic, the and L and V are half of F.

For the iterative portion of the steepest descent method, the for loop iterates 20000 times unless the loop stops for conditions mentioned later. A tolerance of 1e-8 is defined for the stop condition and the step size exponent, j, is 0. During each iteration, the 5 functions are evaluated with the current guess, x. The 5 functions are a combination of mole balances and component balances (Figure **1**). The non-linear functions prompt the use objective function that is the sum of the function squared. A subroutine called g is called and the subroutine takes in the current guess and returns f1^2 + f2^2 + f3^2 + f4^2 + f5^2. Another subroutine called gradf takes in the current guess and values from 5 functions and returns the gradient of the objective function.

The gradient is determined by summating the derivative of the objective function (Figure 1). The search direction (d) is the negative of the gradient, and the step size is the 0.5 to the power of j. For the simple line search, a conditional statement examines whether the objective function at the next guess is greater than the current guess. This is important because for the objective function to minimize to 0. Therefore, the j = j + 1 and the step size decreases if the statement is true. The lower step size decreases the magnitude of the t*d term, which decreases the value of the objective function. If the statement is false, another statement is carried out to determine if the t * norm of d / norm of x is less than the tolerance. If the relative error condition is satisfied, the two arrays with a length of 4, called X and y, are generated for the vapor and liquid fractions, respectively. The X array is filled with the first 4 elements of the current guess array. The elements of the y array are filled using Raoult's law (Figure 1). Raoult's law determines the vapor fraction using the liquid fraction, $P_i^{sat}(T)$, and the pressure at the current guess. The flash pressure is set as x[4]. An array, called a list, is filled with the x, y, V, and P and the subroutine returns the list and ends. If the maximum iteration has been reached and the condition that t * lin.norm(d)/lin.norm(x) is less than the tolerance is not satisfied, the code prints the string, "Max iteration has been reached" and returns the list. The flash2 subroutine ends.

Although the subroutine works, the rate of convergence is very slow. This prompts for further optimization of the subroutine. For example, different line search or step size.

$$
\begin{aligned}
f_1 &: P - P_1^{sat}(T) - P_2^{sat}(T) - P_3^{sat}(T) - P_4^{sat}(T) = 0\\
f_2 &: z_1 F P - P_1^{sat}(T)\, x_1 V - y_1 P L = 0\\
f_3 &: z_2 F P - P_2^{sat}(T)\, x_2 V - y_2 P L = 0\\
f_4 &: z_3 F P - P_3^{sat}(T)\, x_3 V - y_3 P L = 0\\
f_5 &: z_4 F P - P_4^{sat}(T)\, x_4 V - y_4 P L = 0\\
f &= f_1^2 + f_2^2 + f_3^2 + f_4^2 + f_5^2
\end{aligned}
$$

$$\text{grad } f(x) = \nabla f_1^2(x)\ \nabla f_2^2(x)\ \nabla f_3^2(x) + \nabla f_4^2(x) + \nabla f_5^2(x) \qquad y_i = x_i \frac{P_i^{sat}(T)}{P}$$

Figure 1. 5 functions and the objective function (left). Gradient of the objective function (middle). Raoult's Law reexpressed to solve vapor fraction

The flash3 subroutine uses Newton’s method for solving systems of non-linear equations to determine the liquid (X) and vapor (y) fractions, and the total liquid (L) and vapor (V) molar flow rates.

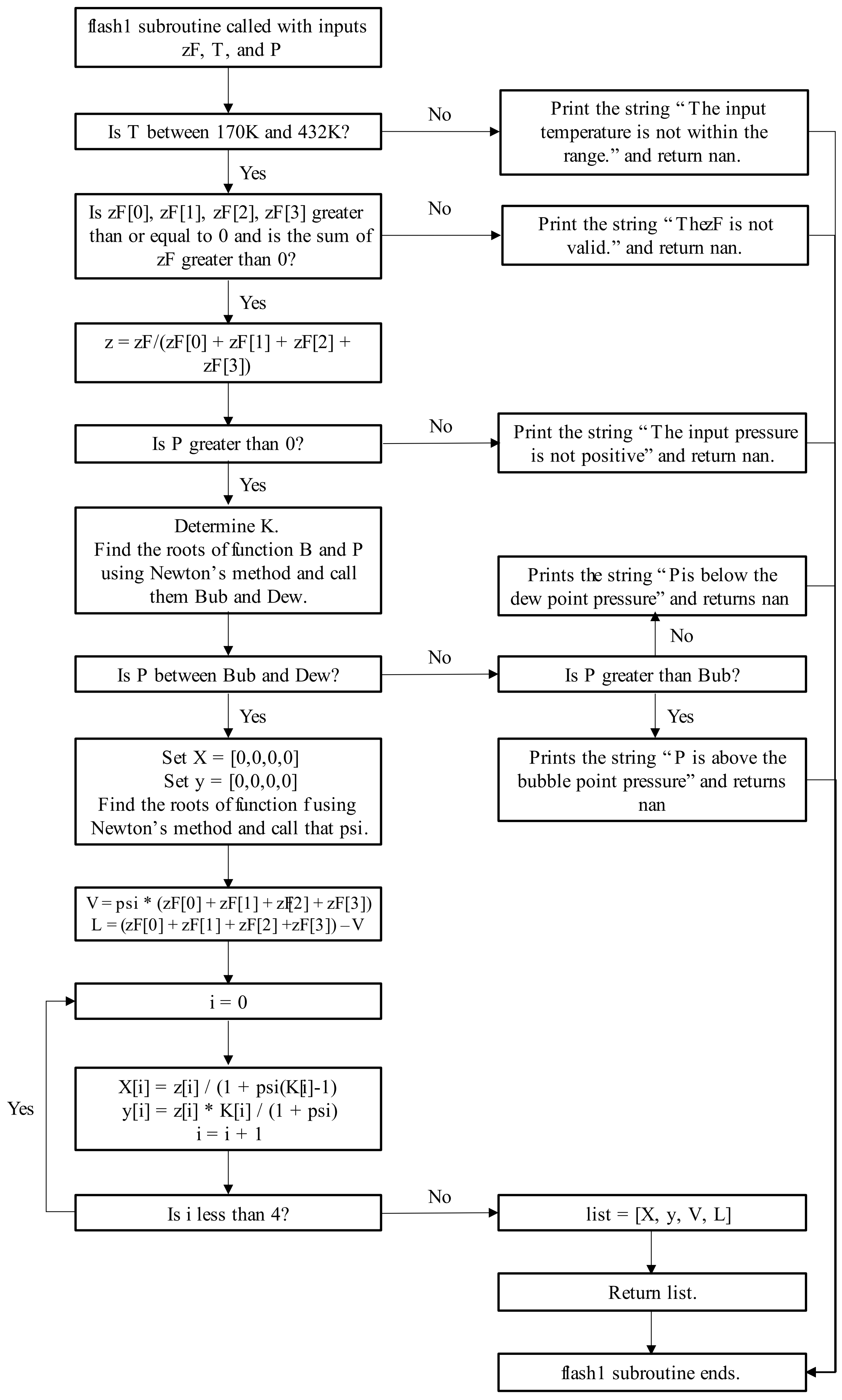
When the flash3 subroutine is called with zF, T, and vapor fraction of the lowest boiling species (y1), the inputs are passed through several conditional statements. These conditional statements ensure that the inputs are physically possible. The first and second conditional statements are the same as flash1 and flash2. The third conditional statement verifies that the y1 input is positive between 0 and 1. Vapor fraction in this study is the vapor molar flow rate of a species over the total vapor molar flow rate. A negative y1 is not physically possible because there cannot be a negative fraction of a species in a solution. By similar logic, the y1 input cannot be greater than 1 because there cannot be a higher molar flow rate of species 1 than the total flow rate. Therefore, the subroutine will print the string, “the input y1 is not valid” and return nan if the y1 is not between 0 and 1. After checking that the zF, T, and y1 are valid inputs, a search directive 10x1 matrix called b is filled with zeros, the total feed molar flow rate was determined by summating the individual feed molar flow rates, and bubble and pressure point at the flash temperature is evaluated using the newton’s method, which was described in flash1. Another 10x1 matrix called xk, is initialized with initial guesses for the 10 unknowns: x1, x2, x3, x4, y2, y3, y4, P, L, V. These will be renamed xk[0, 0] to xk[9, 0] based on the described order (Figure **2**). The initial guesses for the vapor and liquid components, excluding y1, was 0.25, the initial guess of the pressure was the average of the bubble and dew point pressures at the flash temperature, and L and V are half of F.

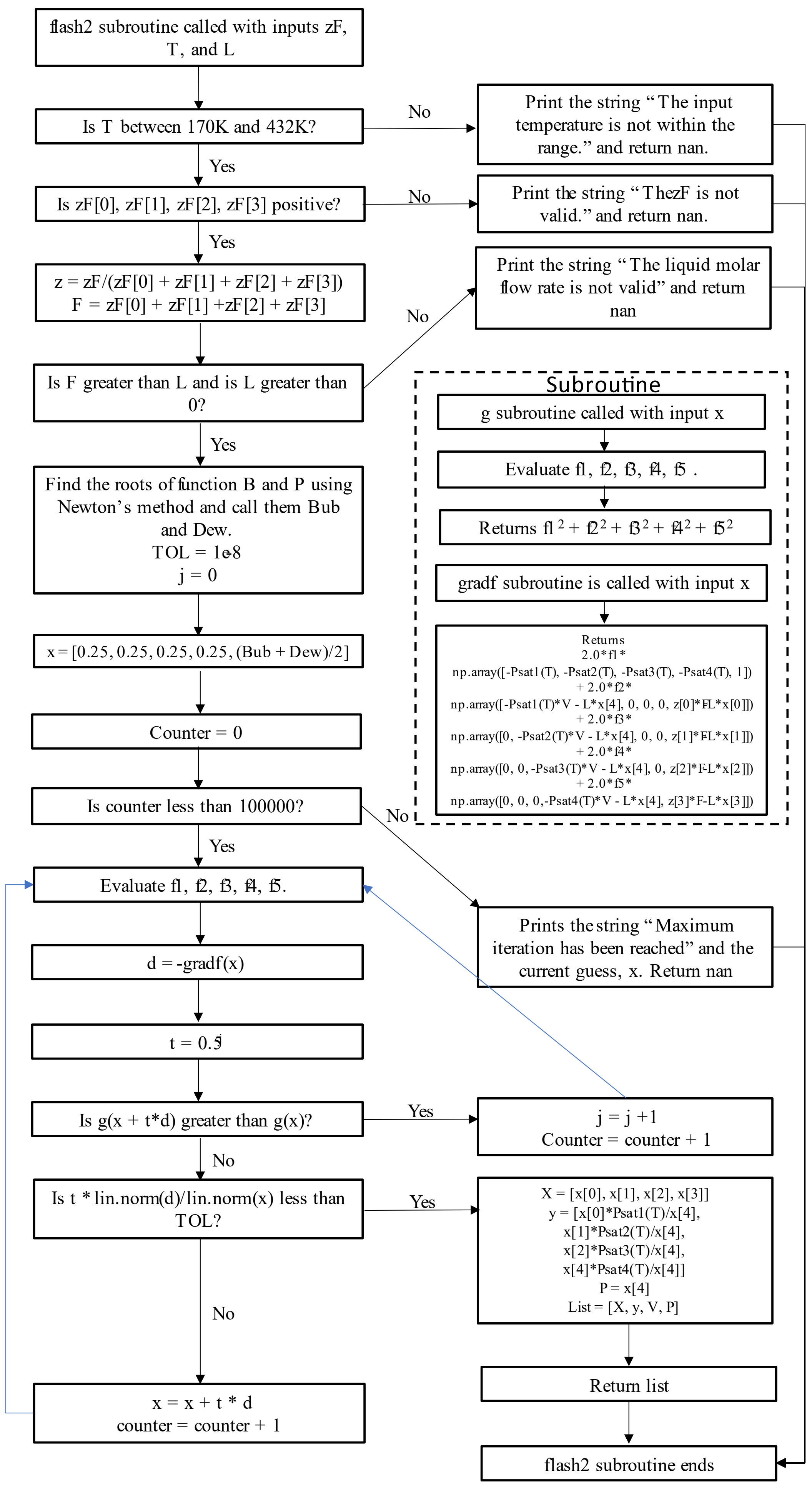
For the iterative portion of Newton’s method, a for loop is set to iterate 10000 times, unless the loop stops for conditions mentioned later. A tolerance of 1e-8 is defined for the stop condition. During each iteration, the 10 equations are evaluated with the current guess and complied into a 10x1 matrix called f. The 10 equations are a combination of mole balances, component balances, and thermodynamic relations from Raoult’s law at VLE (Figure **2**). A 10x10 Jacobian matrix is evaluated with the 10 equations and 10 unknowns using the current guess (xk). For clarity, the general Jacobian matrix used is provided in Figure 1b. The search direction matrix, b, was evaluated using NumPy's linalg function with J and f as inputs because the matrix multiplication of J and b equals f (J@b = f). A conditional statement determines whether the infinity norm of the search direction is less than the tolerance. If the statement is true, two arrays with a length of 4, called x and y, are generated for the vapor and liquid fractions, respectively. The code checks if the V, L, and P are positive, if true V, L, and P are set to xk[9, 0], xk[8,0], and xk[7, 0], respectively. Otherwise, the subroutine returns nan. This is possible even though the inputs satisfy the first three conditional statements. The x, y, V, L, P are compiled into an array called list and the list is returned; the for loop stops iterating and the subroutine ends. However, if the infinity norm of b is greater than the tolerance, the new guess, xk, becomes xk – b, and the loop repeats. If the condition that infinity norm of b is less than b is not satisfied, the subroutine prints the string, “Max iteration has been reached” and returns nan. The flash3 subroutine ends.

$$\begin{aligned}
f_1 &: 1 - x_1 - x_2 - x_3 - x_4 = 0 \\
f_2 &: 1 - y_1 - y_2 - y_3 - y_4 = 0 \\
f_3 &: z_1 F - y_1 V - x_1 L = 0 \\
f_4 &: z_2 F - y_2 V - x_2 L = 0 \\
f_5 &: z_3 F - y_3 V - x_3 L = 0 \\
f_6 &: z_4 F - y_4 V - x_4 L = 0 \\
f_7 &: x_1 - \frac{y_1 P}{P_1^{sat}(T)} = 0 \\
f_8 &: x_2 - \frac{y_2 P}{P_2^{sat}(T)} = 0 \\
f_9 &: x_3 - \frac{y_3 P}{P_3^{sat}(T)} = 0 \\
f_{10} &: x_4 - \frac{y_4 P}{P_4^{sat}(T)} = 0
\end{aligned}$$

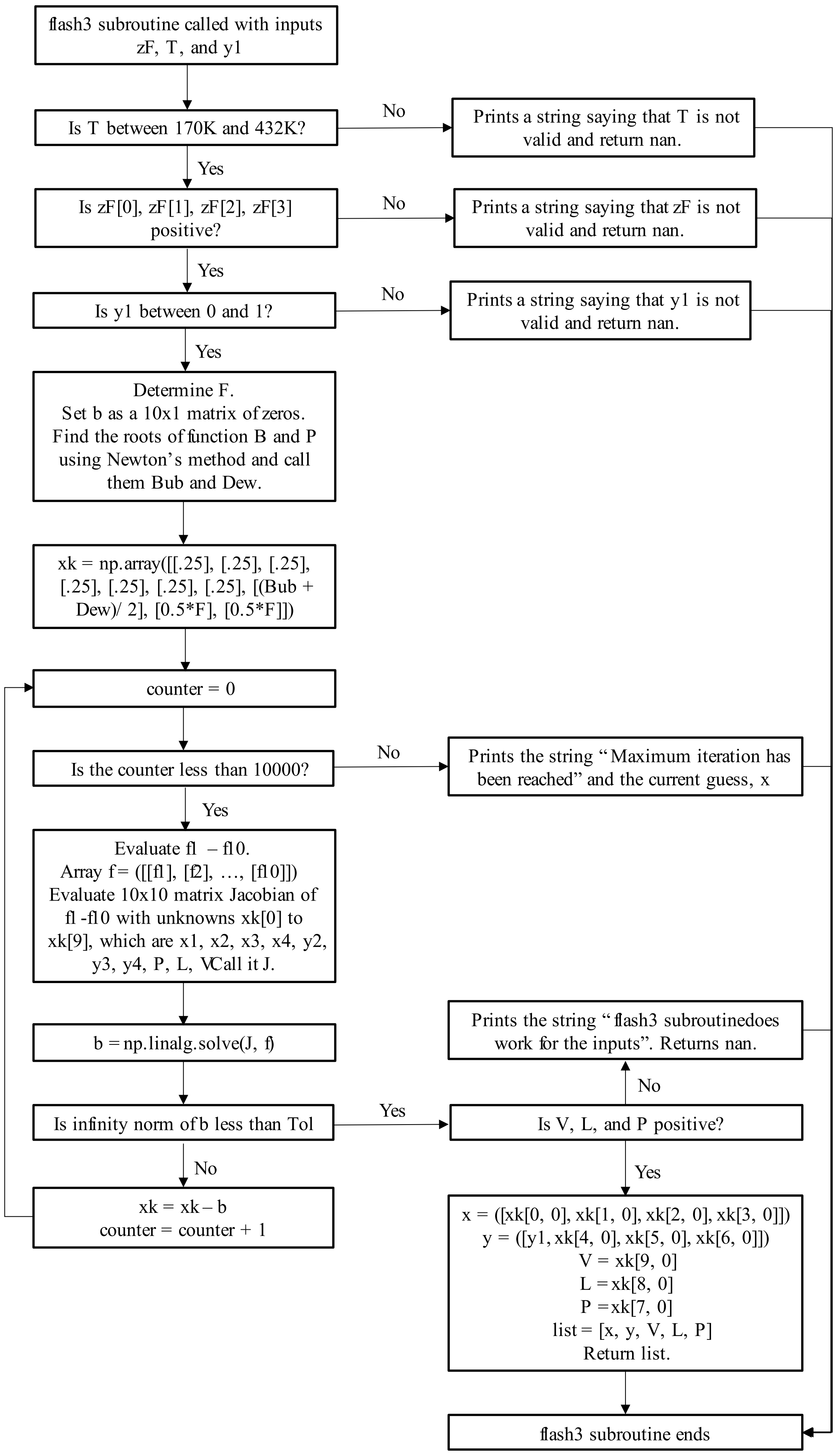
$$J(xk) = \begin{bmatrix} \frac{\partial f_1}{\partial x[0,0]} & \frac{\partial f_1}{\partial x[1,0]} & \cdots & \frac{\partial f_1}{\partial x[9,0]} \\ \frac{\partial f_2}{\partial x[0,0]} & \ddots & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial f_{10}}{\partial x[0,0]} & \cdots & & \frac{\partial f_{10}}{\partial x[9,0]} \end{bmatrix} \text{ where }
\begin{aligned}
x_1 &= x[0,0] \\
x_2 &= x[1,0] \\
x_3 &= x[2,0] \\
x_4 &= x[3,0] \\
y_2 &= x[4,0] \\
y_3 &= x[5,0] \\
y_4 &= x[6,0] \\
P &= x[9,0] \\
L &= x[8,0] \\
V &= x[9,0]
\end{aligned}$$

Figure 2. List of the 10 equations (left). General Jacobian for the flash3 subroutine (middle). Matrix element equivalents of the 10 unknowns (right)


flash1 subroutine called with inputs zF, T, and P
Is T between 170K and 432K?
No
Print the string "The input temperature is not within the range." and return nan.
Yes
Is zF[0], zF[1], zF[2], zF[3] greater than or equal to 0 and is the sum of zF greater than 0?
No
Print the string "The zF is not valid." and return nan.
Yes
z = zF/(zF[0] + zF[1] + zF[2] + zF[3])
Is P greater than 0?
No
Print the string "The input pressure is not positive" and return nan.
Yes
Determine K. Find the roots of function B and P using Newton's method and call them Bub and Dew.
Prints the string "P is below the dew point pressure" and returns nan
No
Is P between Bub and Dew?
No
Is P greater than Bub?
Yes
Yes
Set X = [0,0,0,0] Set y = [0,0,0,0] Find the roots of function f using Newton's method and call that psi.
Prints the string "P is above the bubble point pressure" and returns nan
V = psi * (zF[0] + zF[1] + zF[2] + zF[3]) L = (zF[0] + zF[1] + zF[2] +zF[3]) – V
i = 0
Yes
X[i] = z[i] / (1 + psi(K[i]-1) y[i] = z[i] * K[i] / (1 + psi) i = i + 1
Is i less than 4?
No
list = [X, y, V, L]
Return list.
flash1 subroutine ends.


flash2 subroutine called with inputs zF, T, and L
Is T between 170K and 432K?
No
Print the string "The input temperature is not within the range." and return nan.
Yes
Is zF[0], zF[1], zF[2], zF[3] positive?
No
Print the string "ThezF is not valid." and return nan.
Yes
z = zF/(zF[0] + zF[1] + zF[2] + zF[3])
F = zF[0] + zF[1] +zF[2] + zF[3]
No
Print the string "The liquid molar flow rate is not valid" and return nan
Is F greater than L and is L greater than 0?
Yes
Find the roots of function B and P using Newton's method and call them Bub and Dew.
TOL = 1e-8
j = 0
x = [0.25, 0.25, 0.25, 0.25, (Bub + Dew)/2]
Counter = 0
Is counter less than 100000?
No
Prints the string "Maximum iteration has been reached" and the current guess, x. Return nan
Yes
Evaluate f1, f2, f3, f4, f5.
d = -gradf(x)
t = 0.5
Is g(x + t*d) greater than g(x)?
Yes
j = j +1
Counter = counter + 1
No
Is t * lin.norm(d)/lin.norm(x) less than TOL?
Yes
X = [x[0], x[1], x[2], x[3]]
y = [x[0]*Psat1(T)/x[4],
x[1]*Psat2(T)/x[4],
x[2]*Psat3(T)/x[4],
x[4]*Psat4(T)/x[4]]
P = x[4]
List = [X, y, V, P]
No
x = x + t * d
counter = counter + 1
Return list
flash2 subroutine ends
Subroutine
g subroutine called with input x
Evaluate f1, f2, f3, f4, f5 .
Returns f1 ^{2} + f2^2 + f3^2 + f4^2 + f5^2
gradf subroutine is called with input x

```
Returns
2.0*f1*
np.array([-Psat1(T), -Psat2(T), -Psat3(T), -Psat4(T), 1])
+ 2.0*f2*
np.array([-Psat1(T)*V - L*x[4], 0, 0, 0, z[0]*F-L*x[0]])
+ 2.0*f3*
np.array([0, -Psat2(T)*V - L*x[4], 0, 0, z[1]*F-L*x[1]])
+ 2.0*f4*
np.array([0, 0, -Psat3(T)*V - L*x[4], 0, z[2]*F-L*x[2]])
+ 2.0*f5*
np.array([0, 0, 0,-Psat4(T)*V - L*x[4], z[3]*F-L*x[3]])
```


flash3 subroutine called with inputs zF, T, and y1
Is T between 170K and 432K?
No
Prints a string saying that T is not valid and return nan.
Yes
Is zF[0], zF[1], zF[2], zF[3] positive?
No
Prints a string saying that zF is not valid and return nan.
Yes
Is y1 between 0 and 1?
No
Prints a string saying that y1 is not valid and return nan.
Yes
Determine F.
Set b as a 10x1 matrix of zeros.
Find the roots of function B and P using Newton's method and call them Bub and Dew.
xk = np.array([[.25], [.25], [.25], [.25], [.25], [.25], [.25], [(Bub + Dew)/ 2], [0.5*F], [0.5*F]])
counter = 0
Is the counter less than 10000?
No
Prints the string "Maximum iteration has been reached" and the current guess, x
Yes
Evaluate f1 – f10.
Array f= ([[f1], [f2], …, [f10]])
Evaluate 10x10 matrix Jacobian of f1 -f10 with unknowns xk[0] to xk[9], which are x1, x2, x3, x4, y2, y3, y4, P, L, V. Call it J.
b = np.linalg.solve(J, f)
Is infinity norm of b less than Tol
Yes
Is V, L, and P positive?
No
Prints the string "flash3 subroutine does work for the inputs". Returns nan.
No
xk = xk – b
counter = counter + 1
Yes
x = ([xk[0, 0], xk[1, 0], xk[2, 0], xk[3, 0]])
y = ([y1, xk[4, 0], xk[5, 0], xk[6, 0]])
V = xk[9, 0]
L = xk[8, 0]
P = xk[7, 0]
list = [x, y, V, L, P]
Return list.
flash3 subroutine ends

Reference

[1] Seader, J. D.; Henley, E. J.; Roper, D. K. In *Separation process principles: chemical and biochemical operations*; John Wiley & Sons: Hoboken, NJ, 2011; pp 149–149.