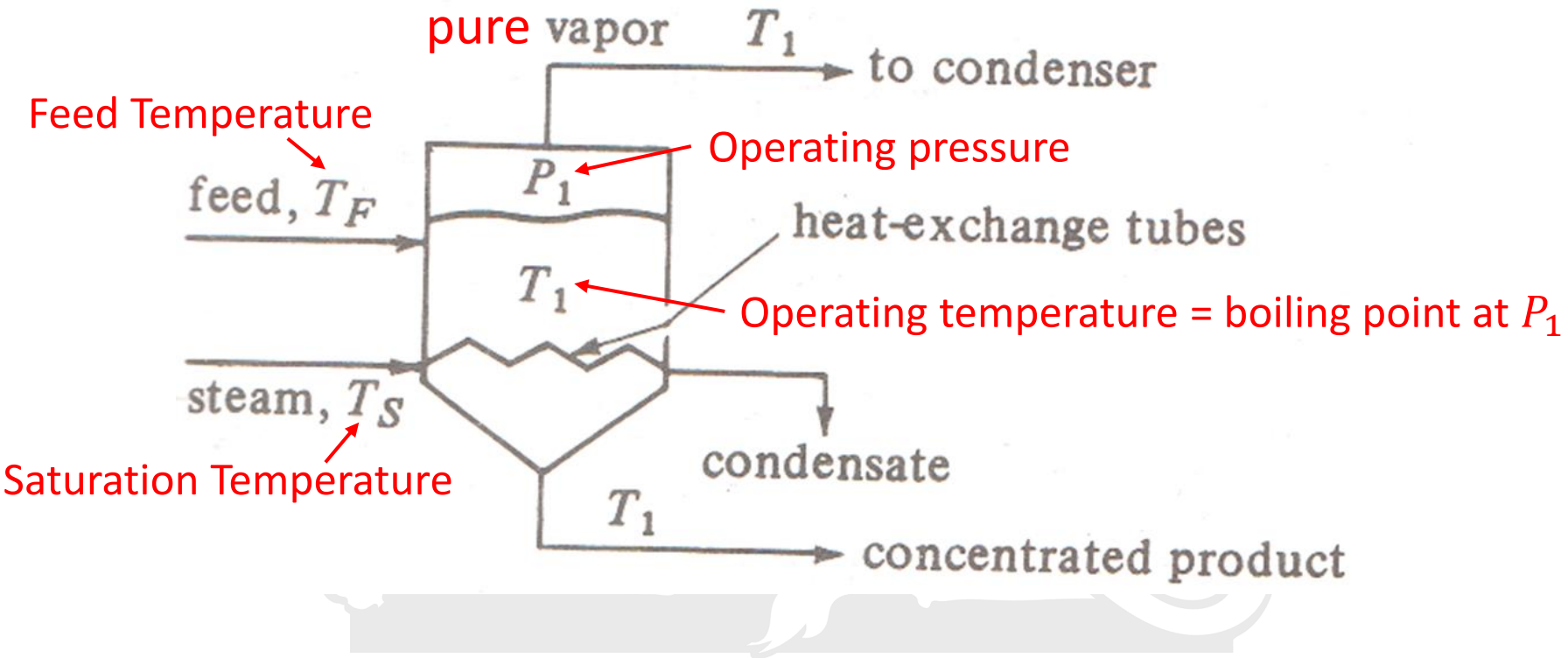
Engineering Analysis & Process Modeling

Dr. Prateek Kumar Jha

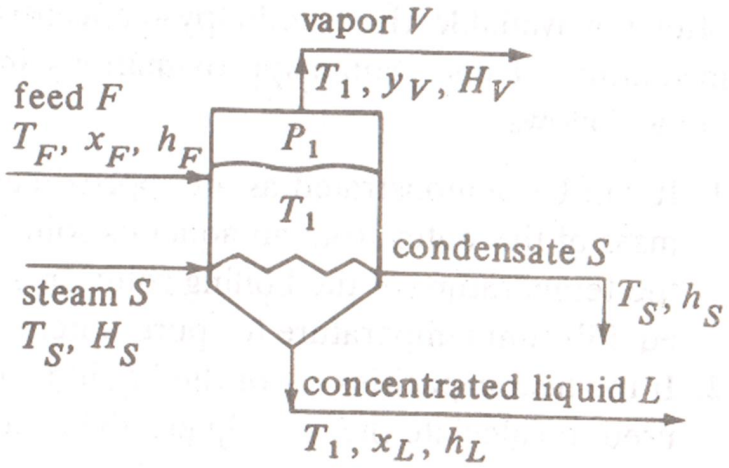
Assistant Professor, Department of Chemical Engineering

prateekjha.iitr@gmail.com

Single-Effect Evaporator



Single-Effect Evaporator: Variable Specification



Stream	Mass flow rate	Temp.	Enthalpy per unit mass	Solute mass fraction
Feed	F	T_F	h_F	x_F
Concentrated liquid	L	T_1	h_L	x_L
Vapor	V	T_1	H_V	$y_V \approx 0$
Steam	S	T_S	H_S	0
Condensate	S	T_S	h_S	0

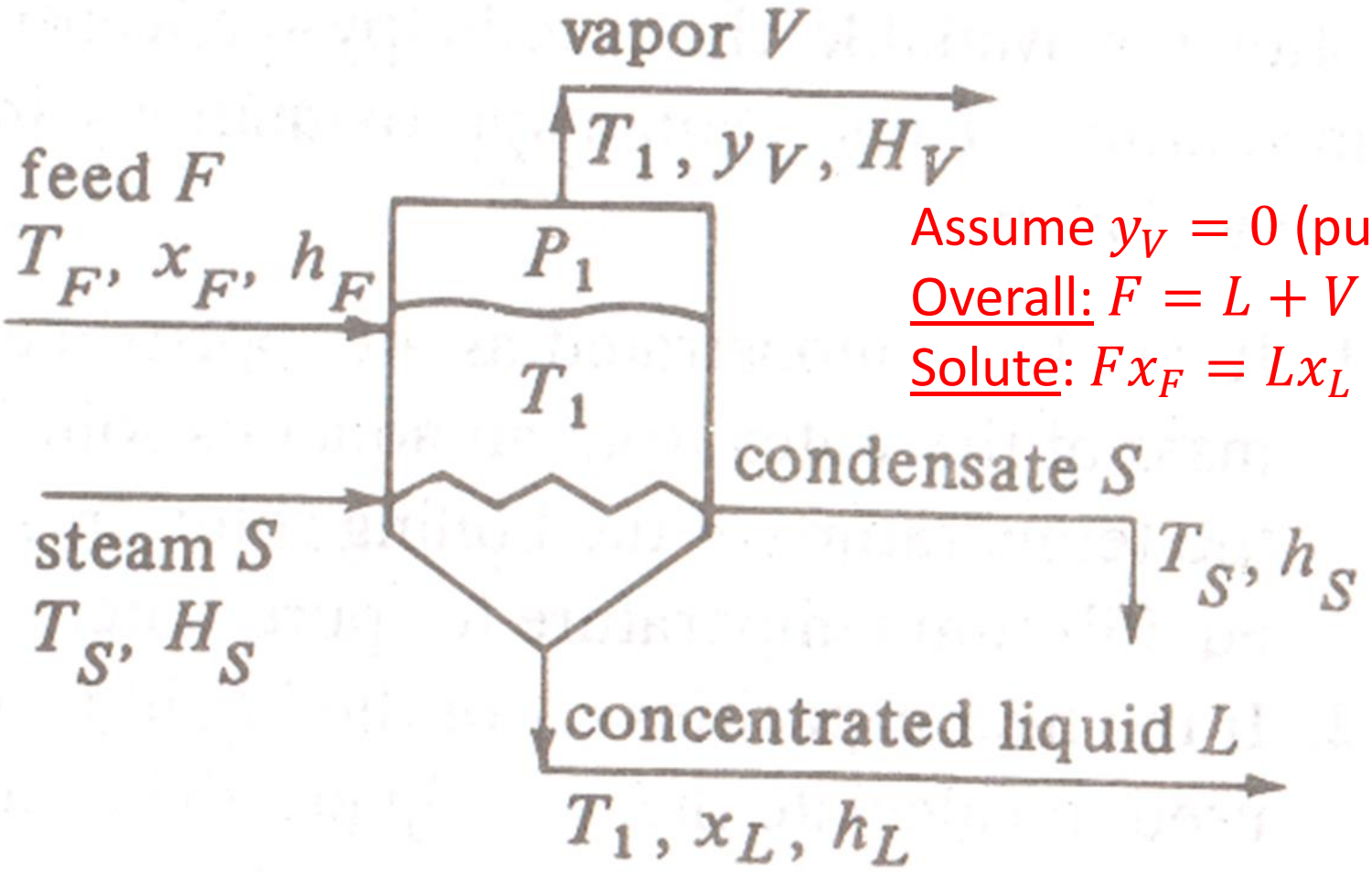
Heat transfer rate (from steam to water) is given by

$$q = UA\Delta T = UA(T_S - T_1)$$

Where

$U = overall\ heat\ transfer\ coefficient$

Single-Effect Evaporator: Mass Balance

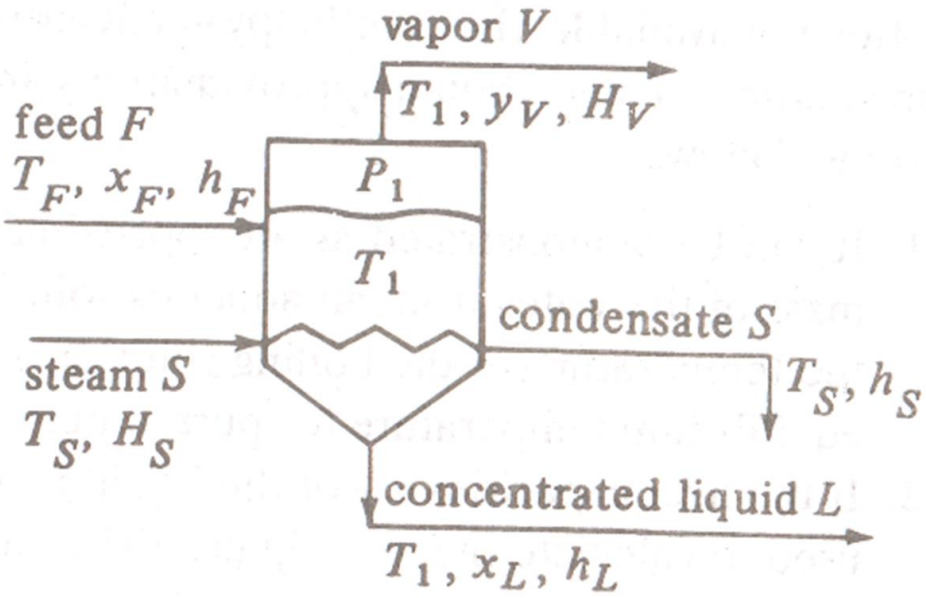


Assume $y_V = 0$ (pure vapor)

Overall: $F = L + V$

Solute: $Fx_F = Lx_L$

Single-Effect Evaporator: Energy Balance



$$\dot{In} = \dot{Out}$$

$$\Rightarrow Fh_F + SH_S = Lh_L + VH_V + Sh_S$$

Subtract $Fh_L + Sh_S$ from both sides (equivalent to setting T_1 as the reference temperature) and using $F = L + V$ we get

$$F(h_F - h_L) + S(H_S - h_S)$$

$$= L(h_L - h_L) + V(H_V - h_L)$$

$$\Rightarrow F(h_F - h_L) + S\lambda_S = V(H_V - h_L)$$

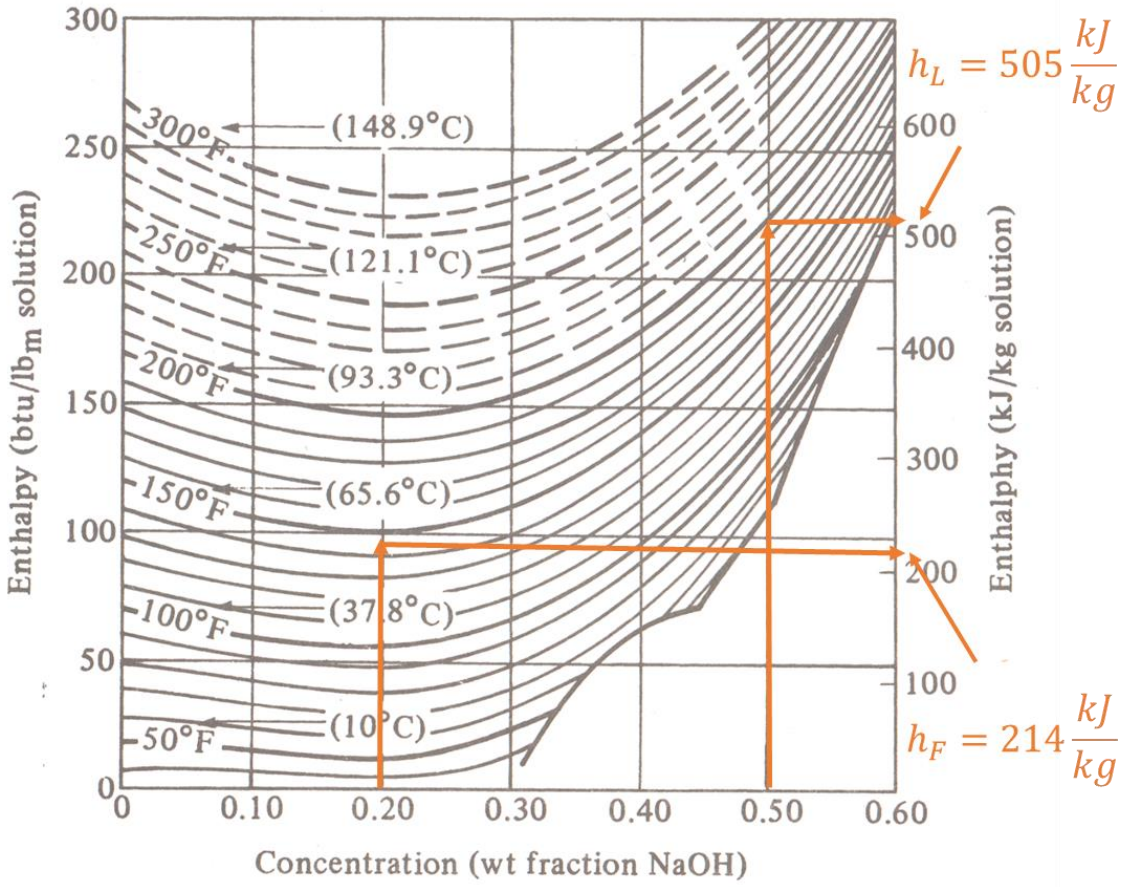
- Latent heat, λ_S is obtained from steam table at the steam pressure P_S and temperature T_S
- H_V is evaluated from steam table at P_1 and T_1
- Heat transferred to the evaporator: $q = S\lambda_S = UA(T_S - T_1)$

Enthalpy-Concentration Chart -- Example

- h_F and h_L can be obtained from the *enthalpy-concentration chart*, available for some systems, which also takes into account "*heat of solution*" (enthalpy change associated with the dissolution)
- In case the heat of solution is negligible, we can approximate $h_F - h_L$ using heat capacities.

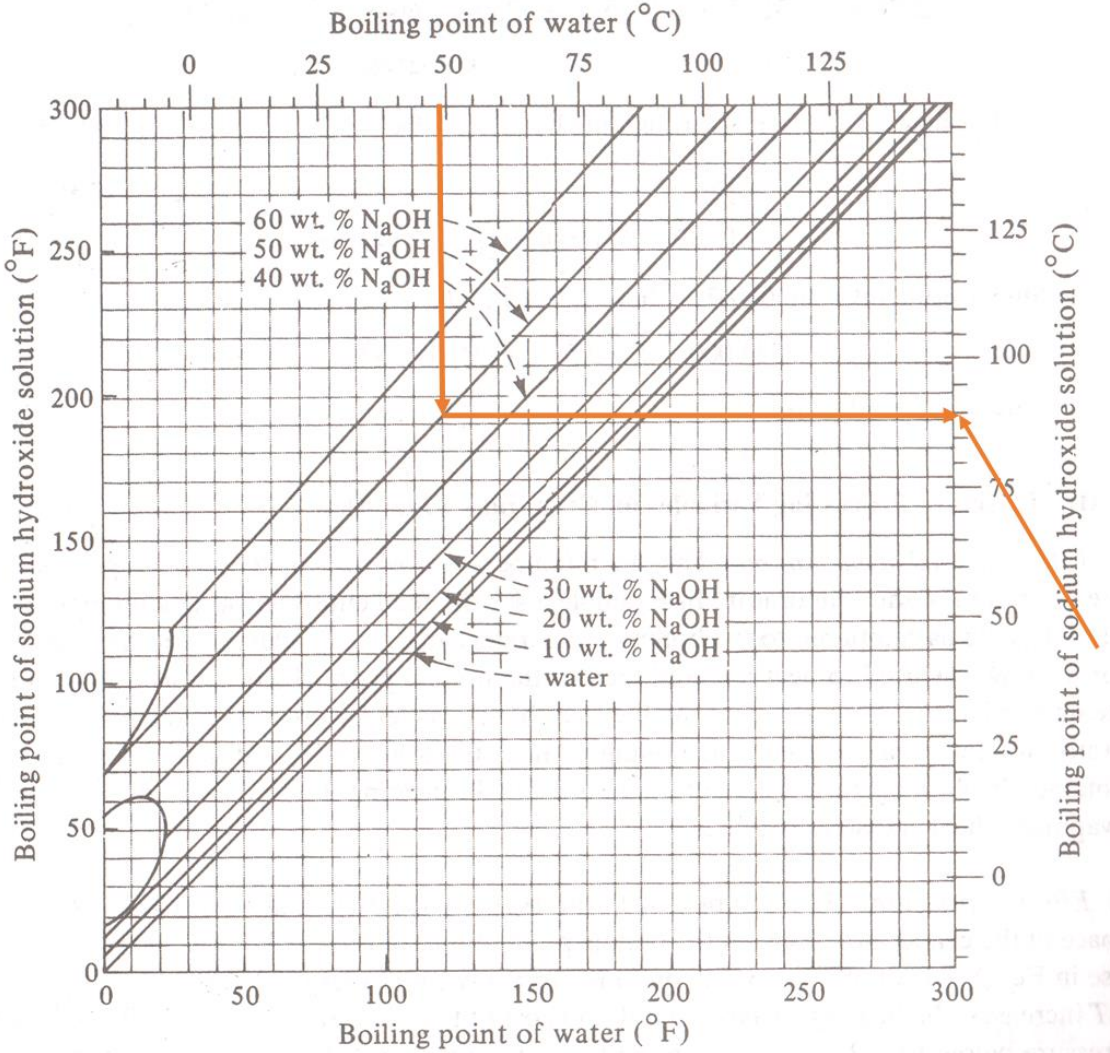
$$h_F - h_L = C_{pF}(T_F - T_1)$$

(also assuming that heat capacity is constant over the temperature range)



Boiling Point Elevation - Dühring plot

1. Compute boiling point of pure water corresponding to P_1
2. Use Dühring plot to find the boiling point of solution corresponding to the boiling point found above

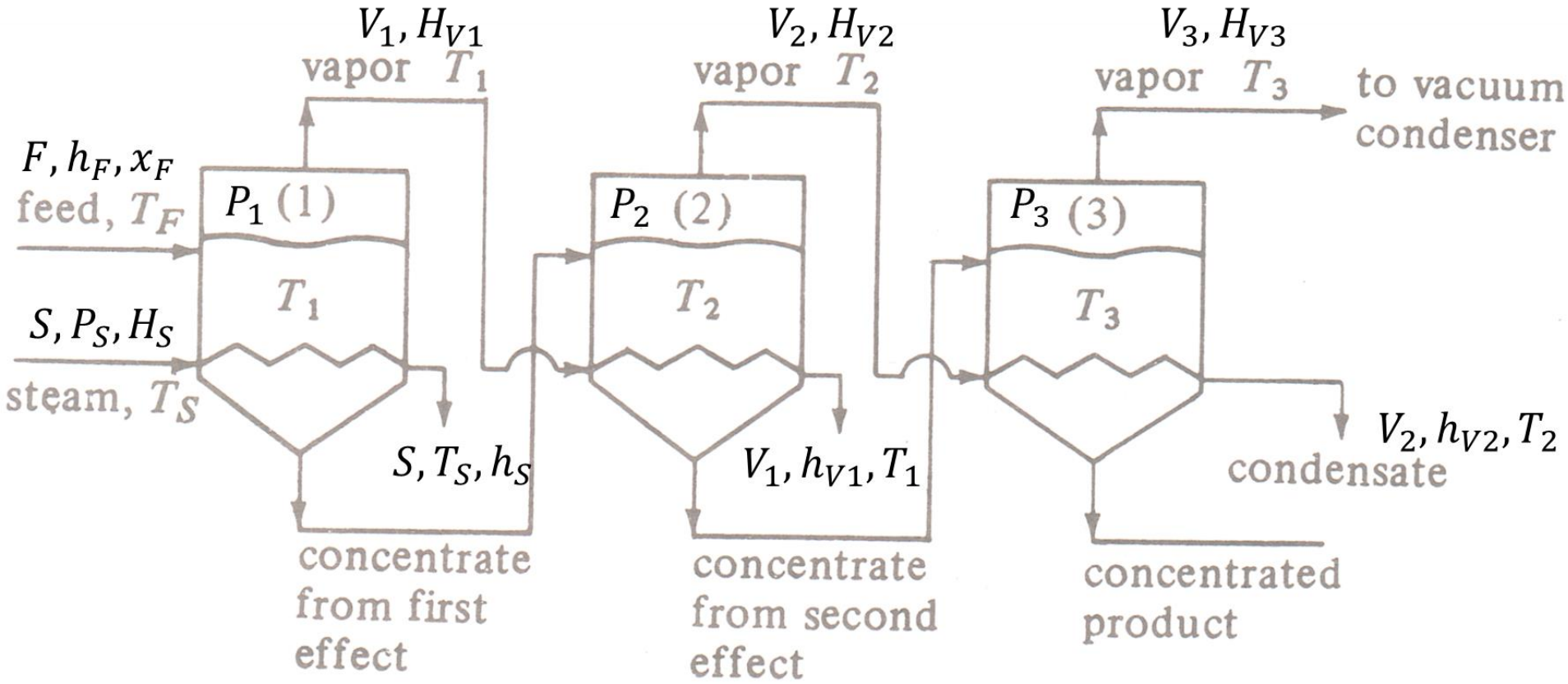


If we ignore elevation in boiling point, h_L is also the enthalpy of pure liquid at T_1

Thus, $H_V - h_L = \lambda_1$, which is the latent heat of pure water at P_1, T_1 and we get,

$$F(h_F - h_L) + S\lambda_S = V\lambda_1$$

Forward feed triple effect evaporator

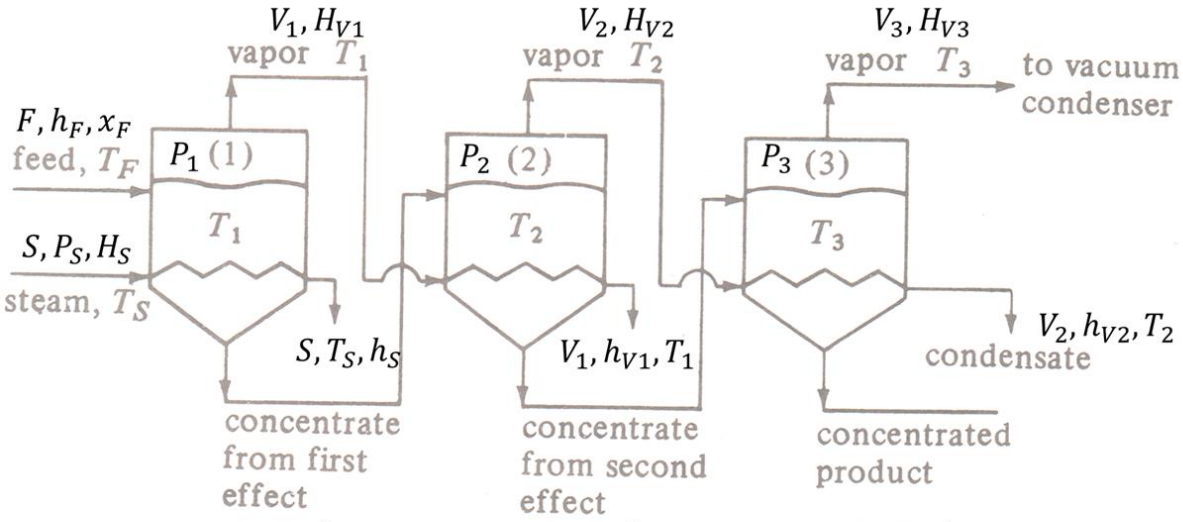


L_1, T_1, h_{L1}, x_{L1} L_2, T_2, h_{L2}, x_{L2} L_3, T_3, h_{L3}, x_{L3}

$$q_1 = U_1 A_1 (T_S - T_1) \quad q_2 = U_2 A_2 (T_1 - T_2) \quad q_3 = U_3 A_3 (T_2 - T_3)$$

Mass Balance

$V_1 = F - L_1 \quad V_2 = L_1 - L_2 \quad V_3 = L_2 - L_3$



$q_1 = U_1 A_1 (T_S - T_1) \quad q_2 = U_2 A_2 (T_1 - T_2) \quad q_3 = U_3 A_3 (T_2 - T_3)$

Overall Mass Balance

- First effect:

$$F = L_1 + V_1 \Longrightarrow V_1 = F - L_1$$

- Second effect:

$$L_1 = L_2 + V_2 \Longrightarrow V_2 = L_1 - L_2$$

- Third effect:

$$L_2 = L_3 + V_3 \Longrightarrow V_3 = L_2 - L_3$$

Solute Mass Balance

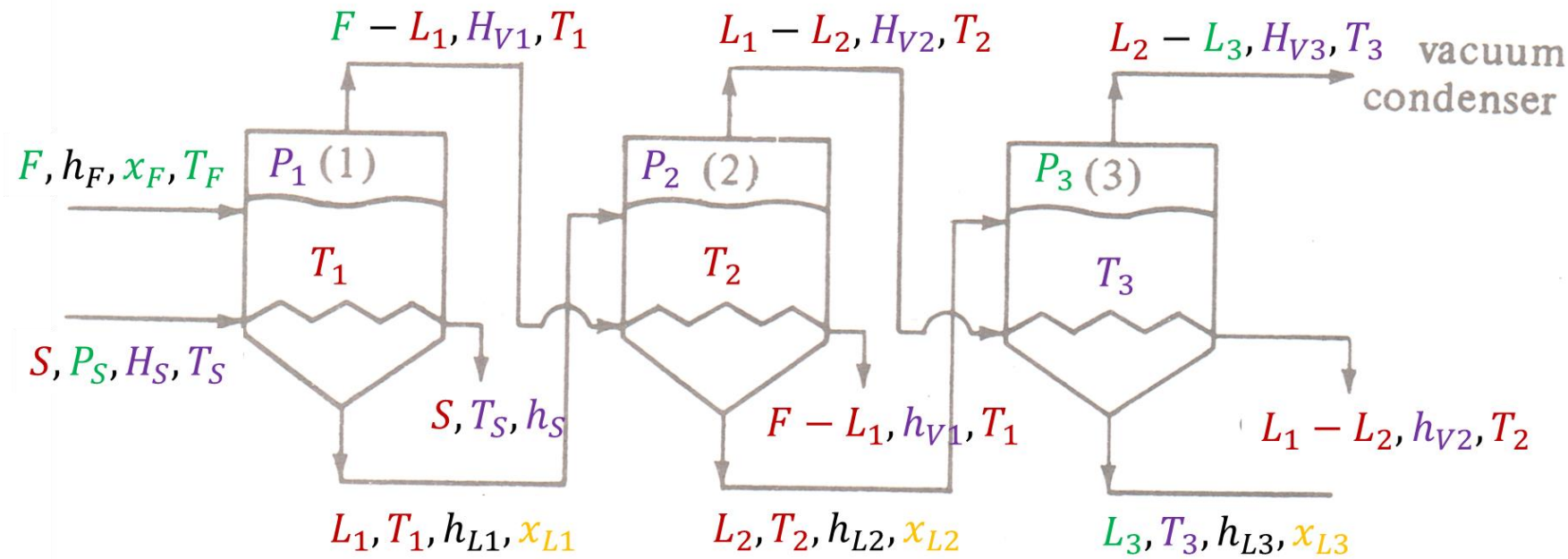
$$F x_F = L_1 x_{L1} = L_2 x_{L2} = L_3 x_{L3}$$

Or, in compact notation, $F x_F - L_i x_{Li} = 0$ for $i = 1,2,3$

Simplified case: $A_1 = A_2 = A_3 = A$ no boiling point elevation

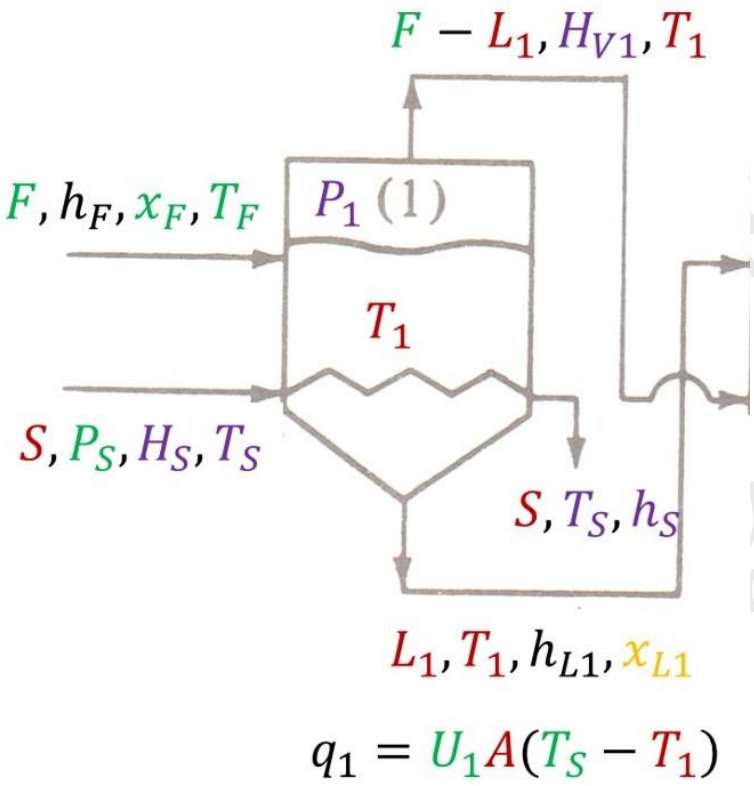
1. GIVEN: $F, x_F, T_F, P_S\ (or\ T_S), P_3\ (or\ T_3), L_3\ (or\ x_{L3}), U_1, U_2, U_3$
2. COMPUTE USING STEAM TABLE: $T_S\ (or\ P_S), \lambda_S, T_3\ (or\ P_3), \lambda_3$
3. CALCULATE: Independent variables: S, T_1, T_2, L_1, L_2, A

   Dependent variables: $h_F - h_{L1}, h_{L1} - h_{L2}, h_{L2} - h_{L3}$
4. COMPUTE USING STEAM TABLE: $P_1, P_2, \lambda_1, \lambda_2$
5. COMPUTE USING SOLUTE MASS BALANCE: $x_{L1}, x_{L2}, x_{L3}\ (or\ L_3)$



$q_1 = U_1 A(T_S - T_1)$ $q_2 = U_2 A(T_1 - T_2)$ $q_3 = U_3 A(T_2 - T_3)$

Energy Balance on 1st effect



$$q_1 = U_1 A(T_S - T_1)$$

- Energy balance equation is,

$$F(h_F - h_{L1}) + S\lambda_S = (F - L_1)\lambda_1 \quad (1a)$$

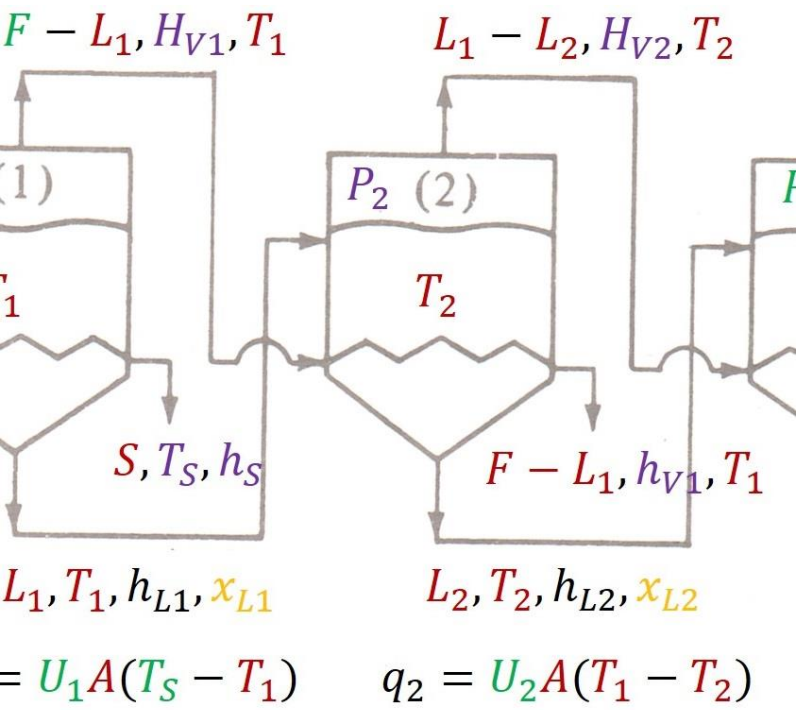
- Ignoring heat of solution,

$$FC_p(T_F - T_1) + S\lambda_S = (F - L_1)\lambda_1 \quad (1b)$$

- Rate of heat transfer:

$$q_1 = S\lambda_S = U_1 A(T_S - T_1) \quad (2)$$

Energy Balance on 2nd effect



- Energy balance equation is,

$$L_1(h_{L1} - h_{L2}) + (F - L_1)\lambda_1 = (L_1 - L_2)\lambda_2 \quad (3a)$$

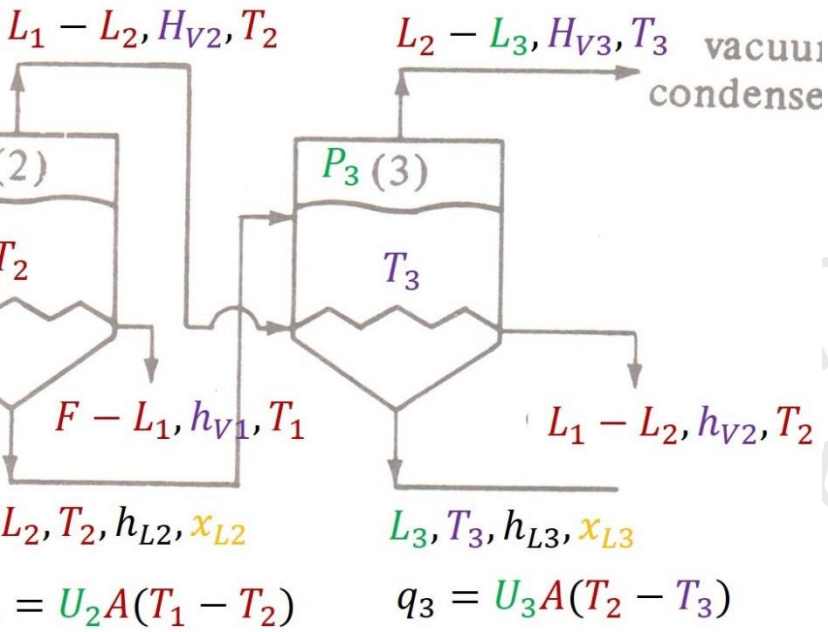
- Ignoring heat of solution,

$$L_1C_p(T_1 - T_2) + (F - L_1)\lambda_1 = (L_1 - L_2)\lambda_2 \quad (3b)$$

- Rate of heat transfer:

$$q_2 = (F - L_1)\lambda_1 = U_2A(T_1 - T_2) \quad (4)$$

Energy Balance on 3rd effect



- Energy balance equation is,

$$L_2(h_{L2} - h_{L3}) + (L_1 - L_2)\lambda_2 = (L_2 - L_3)\lambda_3 \quad (5a)$$

- Ignoring heat of solution,

$$L_2C_p(T_2 - T_3) + (L_1 - L_2)\lambda_2 = (L_2 - L_3)\lambda_3 \quad (5b)$$

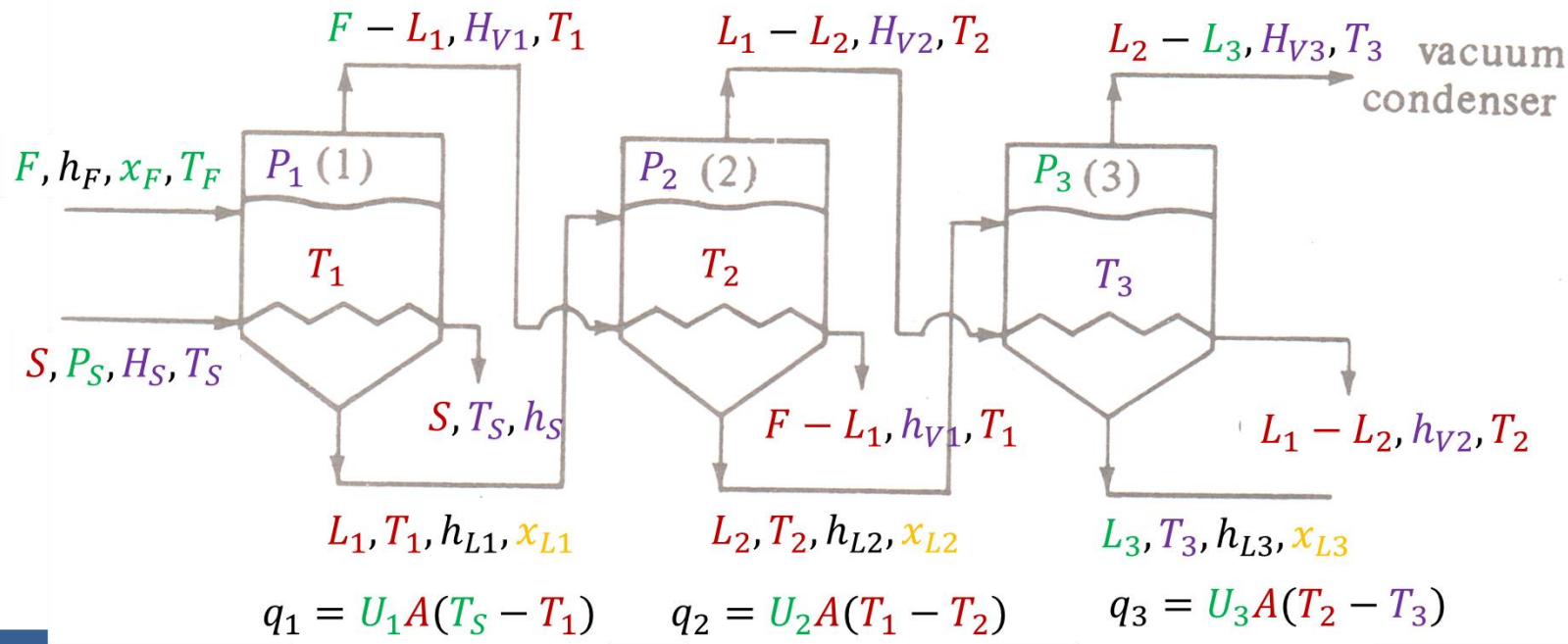
- Rate of heat transfer:

$$q_3 = (L_1 - L_2)\lambda_2 = U_3A(T_2 - T_3) \quad (6)$$

Final set of equations

1. $FC_p(T_F - T_1) + S\lambda_S = (F - L_1)\lambda_1$
2. $L_1C_p(T_1 - T_2) + (F - L_1)\lambda_1 = (L_1 - L_2)\lambda_2$
3. $(L_1 - L_2)\lambda_2 + L_2C_p(T_2 - T_3) = (L_2 - L_3)\lambda_3$
4. $S\lambda_S = U_1A(T_S - T_1)$
5. $(F - L_1)\lambda_1 = U_2A(T_1 - T_2)$
6. $(L_1 - L_2)\lambda_2 = U_3A(T_2 - T_3)$

- 6 equations in 6 variables S, T_1, T_2, L_1, L_2, A
- Then use solute mass balance to get x_{L1}, x_{L2}, x_{L3}



Solution using Badger-McCabe Method

I. Assume T_1, T_2

II. Compute L_1, L_2, S using energy balance equations (1),(2) and (3)

III. Use the variable-area equivalent of equations (4)-(6) to get heating surface areas A_1, A_2, A_3

$$q_1 = U_1 A_1 (T_S - T_1) \qquad q_2 = U_2 A_2 (T_1 - T_2) \qquad q_3 = U_3 A_3 (T_2 - T_3)$$

Note: We are given $A_1 = A_2 = A_3 = A$ but we may get $A_1 \neq A_2 \neq A_3$ since the guess T_1, T_2 may not be equal to the correct values. $q_1 = S\lambda_S$, $q_2 = (F - L_1)\lambda_1$, $q_3 = (L_1 - L_2)\lambda_2$ are known.

IV. If $A_1 \neq A_2 \neq A_3$, redistribute temperature drops to get new values of T_1, T_2 and go to step II

Redistribution of temperature drops

- Temperature drops in three effects using the guess (old) values of T_1, T_2 are given as,

$$\Delta T_1^{old} = T_S - T_1 = \frac{q_1}{U_1 A_1} \quad , \Delta T_2^{old} = T_1 - T_2 = \frac{q_2}{U_2 A_2} \; , \Delta T_3^{old} = T_2 - T_3 = \frac{q_3}{U_3 A_3} \tag{1}$$

- The temperature drop for $A_i = A$ for the same q_i obtained using guess (old) values of T_1, T_2 are given by,

$$\Delta T_i^{new} = \frac{q_i}{U_i A} = \Delta T_i^{old} \frac{A_i}{A} \quad i = 1,2,3 \tag{2}$$

- Since

$$\sum_{i=1}^{3} \Delta T_i^{new} = Temperature \;\; drop \; across \; system, \Delta T = T_S - T_3 \; (GIVEN)$$

We have from (2), $A = \frac{\sum_{i=1}^{3} A_i \Delta T_i^{old}}{\Delta T}$ (3)

Now use (2) to get ΔT_i^{new} and hence new values for T_1, T_2 for the new iteration. Continue until temperature drops converge.

Solution using Newton-Raphson Method

$$F_1: \quad FC_p(T_F - T_1) + S\lambda_S - (F - L_1)\lambda_1 = 0$$
$$F_2: \quad L_1C_p(T_1 - T_2) + (F - L_1)\lambda_1 - (L_1 - L_2)\lambda_2 = 0$$
$$F_3: \quad (L_1 - L_2)\lambda_2 + L_2C_p(T_2 - T_3) - (L_2 - L_3)\lambda_3 = 0$$
$$F_4: \quad S\lambda_S - U_1A(T_S - T_1) = 0$$
$$F_5: \quad (F - L_1)\lambda_1 - U_2A(T_1 - T_2) = 0$$
$$F_6: \quad (L_1 - L_2)\lambda_2 - U_3A(T_2 - T_3) = 0$$

Vector-Form

$$\mathbf{X} = \begin{pmatrix} S \\ T_1 \\ T_2 \\ L_1 \\ L_2 \\ A \end{pmatrix} \qquad \mathbf{F(X)} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} = \mathbf{0}$$

Recap: Equation Solving in many variables

All iterative methods starting with a guess vector

Iterate until $\left\|x^{(k+1)} - x^{(k)}\right\| < tolerance$

p-norm (usually we use $p = 2$): $\left\|\mathbf{X^{(k+1)}} - \mathbf{X^{(k)}}\right\| = \left(\sum_{i=1}^{n} \left|x_i^{(k+1)} - x_i^{(k)}\right|^p\right)^{\frac{1}{p}}$

- Jacobi iteration: To solve $\mathbf{X} = \mathbf{f}(\mathbf{X})$, perform $\mathbf{X^{(k+1)}} = \mathbf{f}(\mathbf{X^{(k)}})$
- Newton-Raphson Method: To solve $\mathbf{F}(\mathbf{X}) = \mathbf{0}$, perform $\mathbf{X^{(k+1)}} = \mathbf{X^{(k)}} + \mathbf{\Delta X^{(k)}}$

$\mathbf{\Delta X^{(k)}}$ is found by solving the linear equation

$$\mathbf{J^{(k)}} \cdot \mathbf{\Delta X^{(k)}} = -\mathbf{F}(\mathbf{X^{(k)}})$$

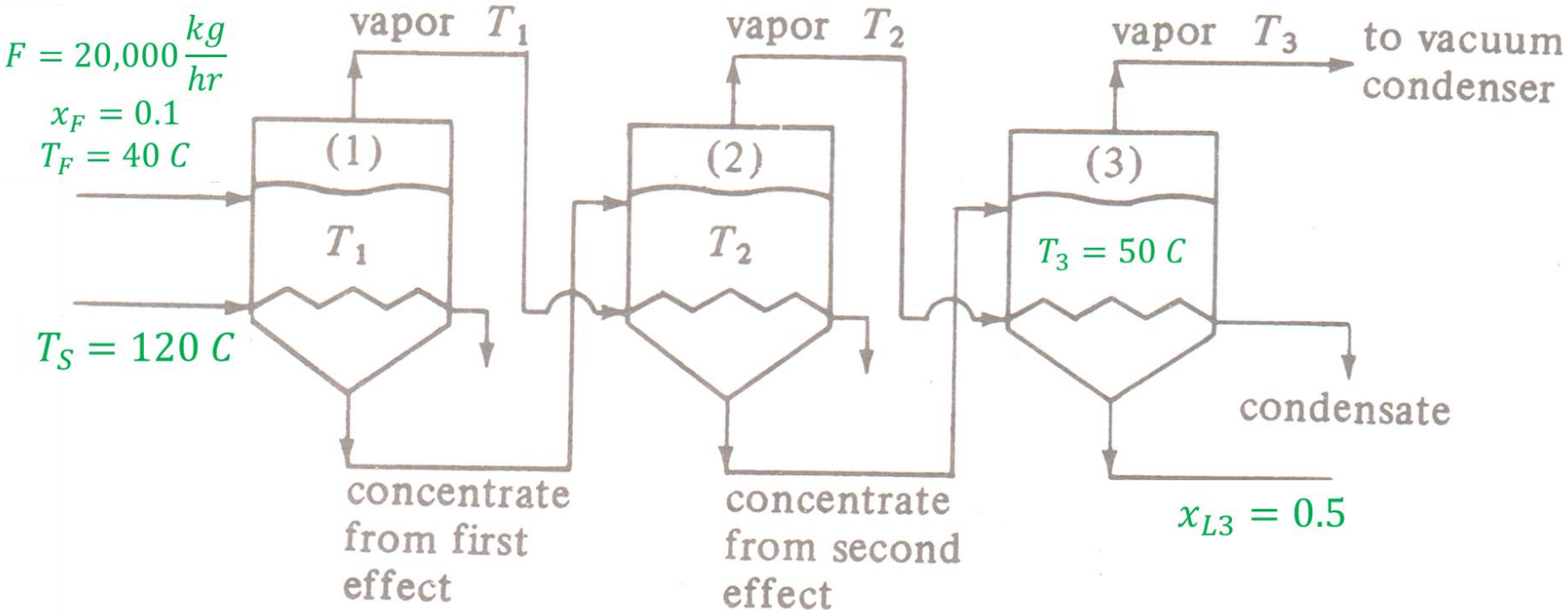
Jacobian, $\mathbf{J^{(k)}} = \begin{bmatrix} \frac{\partial F_1(\mathbf{X})}{\partial x_1} & \frac{\partial F_1(\mathbf{X})}{\partial x_2} & \cdots & \frac{\partial F_1(\mathbf{X})}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \frac{\partial F_n(\mathbf{X})}{\partial x_1} & \frac{\partial F_n(\mathbf{X})}{\partial x_2} & \cdots & \frac{\partial F_n(\mathbf{X})}{\partial x_n} \end{bmatrix}$ evaluated for $\mathbf{X} = \mathbf{X^{(k)}}$

Jacobian

$$\mathbf{J(X)} = \begin{pmatrix} \partial f_1/\partial X_1 & \partial f_1/\partial X_2 & \partial f_1/\partial X_3 & \partial f_1/\partial X_4 & \partial f_1/\partial X_5 & \partial f_1/\partial X_6 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \partial f_6/\partial X_1 & \partial f_6/\partial X_2 & \partial f_6/\partial X_3 & \partial f_6/\partial X_4 & \partial f_6/\partial X_5 & \partial f_6/\partial X_6 \end{pmatrix}$$

$$= \begin{pmatrix} \lambda_S & -FC_p & 0 & \lambda_1 & 0 & 0 \\ 0 & L_1C_p & -L_1C_p & C_p(T_1-T_2)-\lambda_1-\lambda_2 & \lambda_2 & 0 \\ 0 & 0 & L_2C_p & \lambda_2 & C_p(T_2-T_3)-\lambda_2-\lambda_3 & 0 \\ \lambda_S & U_1A & 0 & 0 & 0 & -U_1(T_S-T_1) \\ 0 & -U_2A & U_2A & -\lambda_1 & 0 & -U_2(T_1-T_2) \\ 0 & 0 & -U_3A & \lambda_2 & -\lambda_2 & -U_3(T_2-T_3) \end{pmatrix}$$

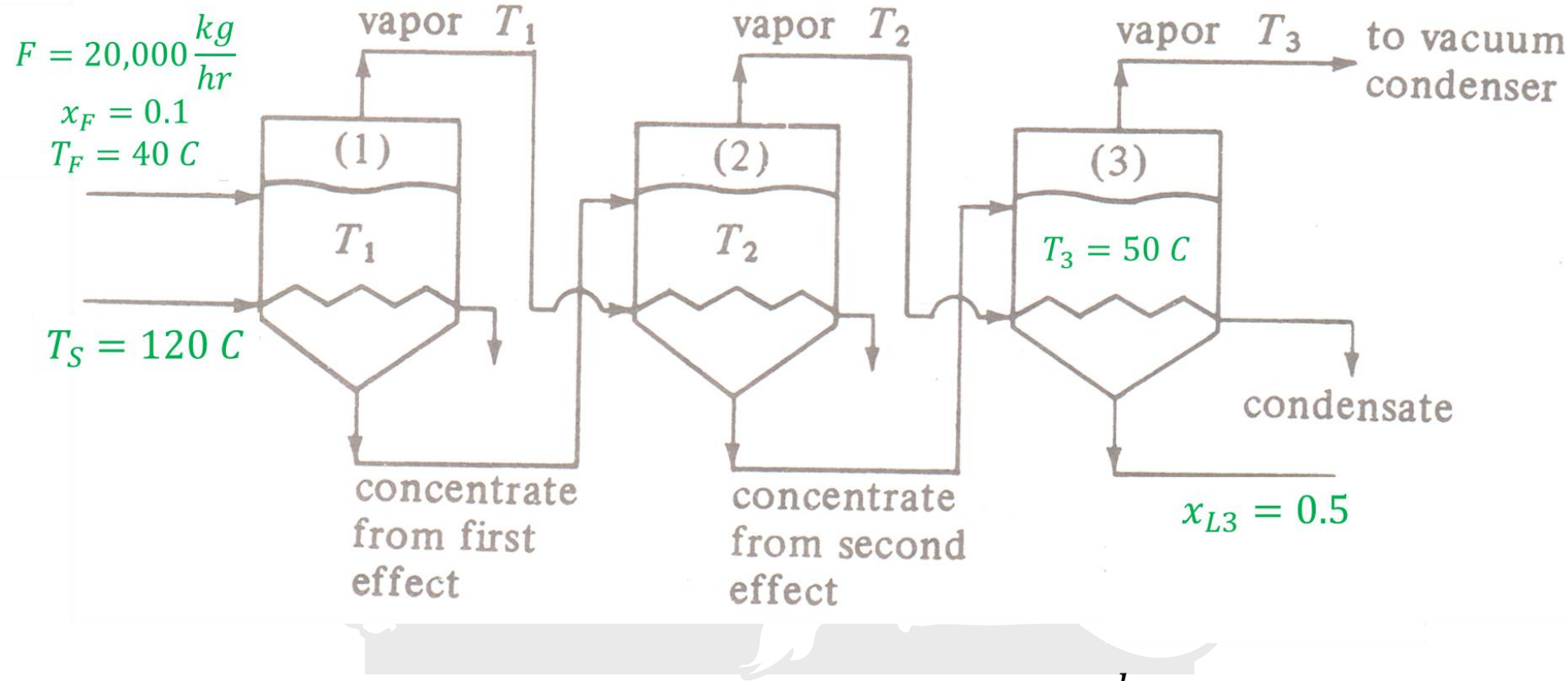
Example



$$q_1 = 3000A(T_S - T_1) \quad q_2 = 1800A(T_1 - T_2) \quad q_3 = 1200A(T_2 - T_3)$$

$$C_p = 4\,\frac{kJ}{kg \cdot K} \qquad \lambda_S = \lambda_1 = \lambda_2 = \lambda_3 = \lambda = 2{,}000\ kJ/kg \qquad U_i\ in\ units\ of\ \frac{kJ}{m^2 \cdot hr \cdot K}$$

Solute Mass Balance



$$L_3(0.5) = 20000(0.1) \Longrightarrow L_3 = 4000\,\frac{kg}{hr}$$

Energy Balance

$$FC_p(T_F - T_1) + S\lambda = (F - L_1)\lambda \Longrightarrow F(T_F - T_1) + \frac{S\lambda}{C_p} = \frac{(F-L_1)\lambda}{C_p}$$

$$\Longrightarrow 20{,}000(40 - T_1) + 500S = (20{,}000 - L_1)500$$

$$q_1 = 500S = 3000A(120 - T_1)$$

$$L_1C_p(T_1 - T_2) + (F - L_1)\lambda = (L_1 - L_2)\lambda \Longrightarrow L_1(T_1 - T_2) + \frac{(F - L_1)\lambda}{C_p}$$

$$= \frac{(L_1 - L_2)\lambda}{C_p}$$

$$\Longrightarrow L_1(T_1 - T_2) + (20{,}000 - L_1)500 = (L_1 - L_2)500$$

$$q_2 = (20{,}000 - L_1)500 = 1800A(T_1 - T_2)$$

$$L_2C_p(T_2 - T_3) + (L_1 - L_2)\lambda = (L_2 - L_3)\lambda \Longrightarrow L_2(T_2 - T_3) + (L_1 - L_2)\lambda/C_p$$

$$= (L_2 - L_3)\lambda/C_p$$

$$\Longrightarrow L_2(T_2 - 50) + (L_1 - L_2)500 = (L_2 - 4{,}000)500$$

$$q_3 = (L_1 - L_2)500 = 1200A(T_2 - 50)$$

6 equations in 6 unknowns T_1, T_2, S, L_1, L_2, A

Jupyter code

```
from numpy import matrix,array
from numpy.linalg import solve,norm
```

```
#input variables
F=20000 #mol/hr
TF=40 #celsius
TS=120 #celsius
T3=50 #celcius

L3=4000 #kg/hr, found from solute balance using xF=0.1 and xL3=0.5

#property values, assumed independent of temperature and composition
Cp=4 #kJ/kg.K
lambdaS=2000 #kJ/kg
lambda1=2000 #kJ/kg
lambda2=2000 #kJ/kg
lambda3=2000 #kJ/kg
U1=3000 #kJ/hr-m^2-K
U2=1800 #kJ/hr-m^2-K
U3=1200 #kJ/hr-m^2-K
```

Jupyter code

```
#fn(X) is the function vector where X is the vector of unknown variables
#[S T1 T2 L1 L2 A]
def fn(X):
    return array([F*Cp*(TF-X[1])      +X[0]*lambdaS        -(F-X[3])*lambda1,
            X[3]*Cp*(X[1]-X[2]) +(F-X[3])*lambda1    -(X[3]-X[4])*lambda2,
            X[4]*Cp*(X[2]-T3)   +(X[3]-X[4])*lambda2 -(X[4]-L3)*lambda3,
            X[0]*lambdaS        -U1*X[5]*(TS-X[1]),
            (F-X[3])*lambda1    -U2*X[5]*(X[1]-X[2]),
            (X[3]-X[4])*lambda2 -U3*X[5]*(X[2]-T3)])

#Jacobian matrix where X is the vector of unknown variables
def Jacobian(X):
    return matrix([[lambdaS,-F*Cp,   0.0,      lambda1,                          0.0,                    0.0],
            [0.0,    X[3]*Cp,-X[3]*Cp,  Cp*(X[1]-X[2])-lambda1-lambda2, lambda2,                    0.0],
            [0.0,    0.0,     X[4]*Cp,  lambda2,                        Cp*(X[2]-T3)-lambda2-lambda3, 0.0],
            [lambdaS,    U1*X[5], 0.0,      0.0,                            0.0,                 -U1*(TS-X[1])],
            [0.0,   -U2*X[5], U2*X[5], -lambda1,                         0.0,               -U2*(X[1]-X[2])],
            [0.0,    0.0,    -U3*X[5],  lambda2,                        -lambda2,            -U3*(X[2]-T3)]])
```

Jupyter code

```
#initial guess of X]
X=array([100.0,30.0,40.0,5000.0,5000.0,1000.0])
error=1.0
tolerance=1.0e-12
count=1
while error>tolerance:
    #compute step size
    ndX=solve(Jacobian(X),fn(X))
    error=norm(ndX)
    X=X-ndX
    count=count+1
print("solution achieved in ", count, " iterations")

print("X=",X)
```

```
solution achieved in  8  iterations
X= [  7208.13105138    102.20699396     82.78902881  15280.14870709
   9966.87862385    270.0735722 ]
```

```
fn(X)
```

```
array([  1.86264515e-09,  -1.86264515e-09,   0.00000000e+00,
         3.72529030e-09,  -1.86264515e-09,   0.00000000e+00])
```