Modelado y simulación de procesos para el curso de operaciones de transferencia de masa

Jheison René Gutérrez Gómez

Universidad Nacional de Colombia
Facultad de Minas, Departamento de procesos y energía
Medellín, Colombia
2020

UNIVERSIDAD NACIONAL DE COLOMBIA

Modelado y simulación de procesos para el curso de operaciones de transferencia de masa

Jheison René Gutérrez Gómez

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de: Ingeniero Químico

Director: Hernán Darío Álvarez Zapata
IQ, M.Sc., Ph.D.

Universidad Nacional de Colombia
Facultad de Minas, Departamento de procesos y energía
Medellín, Colombia
2020

La vida es un proceso mágico, diseñado únicamente por la incomprensible ingeniería del universo.

A mi madre Luz Henny Gómez Murillo, la mujer más bondadosa y humilde que conozco, la que me dio la vida, el ser que más amo en el universo. Su fe, su amor, su animo y su apoyo incondicional siempre han estado en cada momento de mi vida.

A mi padre René Gutiérrez García, la persona que más me apoyo en este camino. Sus consejos, enseñanzas, su cariño y su orgullo me hacen ser un mejor hombre cada día. Fue la persona que me mostró por primera vez lo maravilloso del mundo de los procesos.

A mi hermano Juan David Gutiérrez Gómez, un amigo incondicional, mi confidente, y compañero de vida. No hay persona que me conozca mejor. Gracias hermano por hacerme sentir todo tu orgullo y respeto.

Agradecimientos

Agradecer principalmente al profesor Hernán Darío Álvarez Zapata que me guio en la realización de este trabajo, sus consejos y amplia experiencia se ven reflejados en este documento, además de ser un punto de referencia, ya que es una persona admirable tanto personal como profesionalmente.

Resumen

Este trabajo tiene como objetivo principal proporcionar una guía práctica de como programar y abordar problemas de simulación, principalmente modelos para procesos de transferencia de masa en operación lote puro, semi lote, cocorriente y contra corriente. En este documento se podrá ver las diferencias entre los modelos y como plasmar esto en un lenguaje de programación como Python. Se verá que la programación es muy parecida para algunos modelos, caso especial ocurre con el modo contracorriente, este tendrá un desarrollo diferente a los demás. El trabajo abordará tanto el modelamiento matemático de cada uno de los sistemas, así como la programación detallada de estos modelos y de esta manera construirán cada uno de los simuladores.

La solución de los modelos matemáticos se realizó utilizando métodos numéricos sencillos, al igual que la programación de los simuladores.

Los casos estudiados en este documento son:

1- Modelado y simulación de un equipo de transferencia de masa en operación por lotes.

2- Modelado y simulación de un equipo de transferencia de masa en operación semi-lote. .

3- Modelado y simulación de un equipo de transferencia de masa en operación cocorriente.

4- Modelado y simulación de un equipo de transferencia de masa en operación contra corriente.

Toda la programación de este documento se realizó utilizando el lenguaje de programación de ***Python 3***, y el presente trabajo se construyo en su entorno interactivo web de ejecución de código ***Jupyter Notebook***.

Palabras clave: **Simulación, programación, masa , proceso, modelado, método, lote, semi-lote, cocorriente, contra corriente, modulo, Python, Jupyter.**

Abstract

Lo realizo cuando me envíe las correcciones con el otro documento. ***Jupyter Notebook***.

Palabras clave: **Simulación, programación, masa , proceso, modelado, método, lote, semi-lote, cocorriente, contra corriente, modulo, Python, Jupyter.**

Abstract

Lo realizo cuando me envie las correcciones con el otro documento. ***keywords:.***

Índice

1. Introducción **14**

2. Marco teórico **14**
2.1. Python 14

3. Programación de simuladores **15**
3.1. Simulador dinámico y estático de un proceso por lotes de transferencia de masa. . . 15
3.1.1. Modelamiento matemático 15
3.1.2. Programación del modelo de transferencia de masa en operación lote puro en Python 16
3.2. Simulador dinámico y estático de un proceso por semi lotes de transferencia de masa 23
3.2.1. Modelamiento matemático 23
3.2.2. Programación del modelo de transferencia de masa en operación semi lote en Python 24
3.3. Simulador dinámico y estático de un proceso en cocorriente de transferencia de masa 35
3.3.1. Modelamiento matemático 35
3.3.2. Programación del modelo de transferencia de masa en operación cocorriente en Python 35
3.4. Simulador dinámico y estático de un proceso en contracorriente de transferencia de masa 42
3.4.1. Modelamiento matamático 42
3.4.2. Programación del modelo de transferencia de masa en operación contracorriente en Python 46

4. Resultados **55**
4.1. Resultados de la simulación para el proceso en lote puro 55
4.2. Resultados de la simulación para el proceso en Semi Lote 55
4.3. Resultados de la simulación para el proceso en Cocorriente 57
4.4. Resultados de la simulación para el proceso en Contracorriente 57

5. Conclusiones **59**

Índice de figuras

1. Proceso en operación Lote puro 55
2. Proceso en operación semi lote 55
3. Proceso en operación semi lote (CDE) 56
4. Proceso en operación Cocorriente 57
5. Proceso en operación Contracorriente 57

Lista de símbolos

Símbolos	Término	Unidad SI
$A_{F,T}$	Área de flujo	m^2
aM	Área de TdeMasa volumétrica	$\frac{m^2}{m^3}$
A_M	Área de TdeMasa	$\frac{m^2}{m^3}$
$A_{SupGota}$	Área superficial de una gota	m^2
$CargaE$	Carga molar de solvente puro fase extracto	$kmol$
$CargaR$	Carga molar de solvente puro fase refinado	$kmol$
$CargaMolR_{FasePura}$	Carga molar de refinado puro libre de soluto	$kmol$
$CargaMolE_{FasePura}$	Carga molar de extracto puro libre de soluto	$kmol$
D_{Emp}	Diámetro del empaque contenido en la torre	m
D_{Gota}	Diámetro de una gota	m
D_{Part}	Diámetro de partícula	m
D_{Torre}	Diámetro de la torre	m
F_{j}	Corrección de forma esférica	Adim
$FluxMolE_{FasePura}$	Flux molar de la fase extracto puro libre de soluto	$\frac{kmol}{m^2 s}$
$FluxMolR_{FasePura}$	Flux molar de la fase refinado puro libre de soluto	$\frac{kmol}{m^2 s}$
$Frac_{Vol,E}$	Fracción volumétrica de la fase extracto	Adim
k_x	Coeficiente local de TdeMasa para la fase refinado	$\frac{kmol}{m^2 s}$
k_y	Coeficiente local de TdeMasa para la fase extracto	$\frac{kmol}{m^2 s}$
K_Y	Coeficiente englobante de TdeMasa	$\frac{kmol}{m^2 s}$
K_{Ske}	Parámetro para la ecuación de A.H.P. Skelland	Adim
L_{Torre}	Longitud de la torre	m
M_E	Masa molar de la fase extracto	$\frac{kg}{kmol}$
M_R	Masa molar de la fase refinado	$\frac{kg}{kmol}$
n_A	Moles del soluto A	$Kmol$
$N_{Re,Gota}$	Número de Reynolds para la fase dispersa o extracto (contracorriente)	Adim

Símbolos	Término	Unidad SI
$N_{Sc,R}$	Número de Schmidt para la fase refinado	Adim
Num_{Gotas}	Número de gotas en la fase dispersa (Tolueno)	Adim
$Num_{PartEsp}$	Número de fragmentos de la torre	Adim
$\dot{n}_E$	Flujo molar de la fase estracto+soluto (contracorriente)	$\frac{kmol_A}{kmol_{Solucin}}$
$\dot{n}_{ES}$	Flujo molar de la fase extracto puro libre de soluto (contracorriente)	$\frac{Kmol}{s}$
$\dot{n}_{RS}$	Flujo molar de la fase refinado puro libre de soluto (contracorriente)	$\frac{Kmol}{s}$
N_A	Flux de transferencia de masa	$\frac{kmol}{m^2 s}$
$Paso_{Esp}$	Paso espacial	m
$Paso_{temp}$	Paso temporal	s
$Porosidad$	Porosidad operativa de la torre	Adim
Rel_{fases}	Relación de fases	Adim
t	Tiempo	s
v_E	Velocidad de la fase extracto	$\frac{m}{s}$
v_R	Velocidad de la fase refinado	$\frac{m}{s}$
v_s	Velocidad de deslizamiento	$\frac{m}{s}$
$\dot{V}_E$	Flujo volumétrico de la fase extracto	$\frac{m^3}{s}$
$\dot{V}_R$	Flujo volumétrico de la fase refinado	$\frac{m^3}{s}$
Vol_{Gota}	Volumen de una gota	m^3
Vol_{Total}	Volumen total	m^3
$x_{A,in}$	Fracción molar de entrada respecto a la fase extracto como un todo	$\frac{KmolA}{Kmol_{Solvente}}$
$x_{A,out}$	Fracción molar de salida respecto a la fase extracto como un todo	$\frac{KmolA}{Kmol_{Solvente}}$
$y_{A,in}$	Fracción molar de entrada respecto a la fase refinado como un todo	$\frac{KmolA}{Kmol_{Solvente}}$
$y_{A,out}$	Fracción molar de salida respecto a la fase refinado como un todo	$\frac{KmolA}{Kmol_{Solvente}}$

Símbolos	Término	Unidad SI
$X_{A,in}$	Fracción molar del soluto respecto del solvente libre de soluto	$\frac{KmolA}{Kmol_{SolventePuro}}$
$X_{A,out}$	Fracción molar del soluto respecto del solvente libre de soluto	$\frac{KmolA}{Kmol_{SolventePuro}}$
$X_{A,op}$	Fracción molar del soluto respecto del solvente libre de soluto en la línea de operación (Contracorriente)	$\frac{kg_A}{kg_{Agua}}$
$X_{A,eq}$	Concentración de A en equilibrio o en la interfaz (refinado)	$\frac{kg_A}{kg_{Agua}}$
$Y_{A,in}$	Fracción molar del soluto respecto del solvente libre de soluto	$\frac{KmolA}{Kmol_{SolventePuro}}$
$Y_{A,out}$	Fracción molar del soluto respecto del solvente libre de soluto	$\frac{KmolA}{Kmol_{SolventePuro}}$
Y_{A*}	Concentración ficticia	$\frac{KmolA}{Kmol_{SolventePuro}}$
$Y_{A,op}$	Fracción molar del soluto respecto del solvente libre de soluto en la línea de operación (Contracorriente)	$\frac{kg_A}{kg_{Agua}}$
$Y_{A,eq}$	Concentración de A en equilibrio o en la interfaz (Extracto)	$\frac{kg_A}{kg_{Tolueno}}$
μ_R	Viscosidad de la fase refinado	$\frac{kg}{ms}$
ρ_E	Densidad de la fase extracto	$\frac{kg}{m^3}$
ρ_R	Densidad de la fase refinado	$\frac{kg}{m^3}$
ΔL	Valor tomado para cada tramo de la torre	m
$\mathfrak{D}_{A-T}$	Difusividad de la nicotina en Tolueno	$\frac{m^2}{s}$
$\mathfrak{D}_{A-Agua}$	Difusividad de la nicotina en Agua	$\frac{m^2}{s}$
σ_R	Tensión superficial de la fase refinado (Agua)	$\frac{kg}{s^2}$
(i)	Indice que hace referencia al instante de tiempo	s
(j)	Indice que hace referencia a un tramo de la longitud de la torre	m

1. Introducción

1. El presente documento denominado "Simulación de procesos para el curso de operaciones de transferencia de masa", está diseñado para servir como guía en la programación y simulación de procesos de transferencia de masa. Se abordarán los procesos en operación lote puro, semi lote, cocorriente y contracorriente. Este trabajo se realizó debido a la dificultad que tienen algunas ramas dela ingeniería en modelar, programar y diseñar simulaciones que representan este tipo de sistemas.Lo que se busca en este documento es hacer el modelamiento y programación de esta clase de problemas, fácil, lineal e intuitiva, llevada a cabo en un lenguaje de programación básico y entendible para cualquier persona que no esté muy relacionada con el tema de la programación. El algoritmo desarrollado consistió en juntar acciones y características parecidas entre las variables del proceso, la finalidad de esto era proponer una serie de pasos a seguir para solucionar este tipo de sistemas, el lenguaje de programación utilizado fue Python por su facilidad de manejo comprensión, además de que este es un lenguaje muy conocido. La finalidad u objetivo principal de este trabajo era mostrar que el modelamiento y programación de este tipo de sistemas puede ser fácil y sencilla. Otro objetivo importante de este documento es comparar los distintos tipos de operación que poseen los equipos de transferencia de masa: operación en lote puro, semilote,cocorriente y contracorriente, también en como representar este tipo de procesos por medio den lenguaje de programación como Python. En el desarrollo de este documento primeramente el Capítulo 2 se tratará de relacionar al lector con el lenguaje de programación utilizado, (Python 3) y también se hablará acerca de los tipos operación que posee este tipo de procesos. En el Capítulo 3 inicia con la deducción de los modelos matemáticos que representan cada uno de los casos estudiados. Posteriormente, este capítulo continúa con la explicación detallada referente a la programación de los simuladores para cada modo de operación. Primero se desarrolla el sistema más sencillo, el modo lote puro. Posteriormente, se continúa con el modo semilote, cocorriente y se finaliza con el caso más complejo, el modo contracorriente. El Capítulo 4, muestra los resultados gráficos para cada uno de los simuladores elaborados para finalmente terminar con las conclusiones en el Capítulo 5.

2. Marco teórico

2.1. Python

Python es un intérprete orientado a objetos. Este es un lenguaje de alto nivel, simple y de fácil aprendizaje. Python se compone de módulos y paquetes, el cual le otorga ser un lenguaje de programación modular con reutilización de código. La instalación de este lenguaje de programación es totalmente gratuita y fácil acceso en la web. Python posee un entorno interactivo web de ejecución de código llamado Jupyter Notebook, con características especiales que incluyen transformación de datos, simulación numérica, modelamiento estadístico, visualización de datos, machine learning, Adicionalmente, como se verá en este documento, Jupyter permite mezclar documentos de texto con código [2].

Un análisis de transferencia de masa se lleva a cabo aplicando leyes de conservación de la materia y energía. En este documento se hablará principalmente de las leyes de conservación de la materia. En los sistemas de proceso se aplican balances de materia y energía. Como sé verá en este documento existen dos maneras en las que se puede partir un proceso: tomando cada fase como un todo (un solo sistema de proceso por fase) dentro del equipo (modelo de parámetros concentrados), o haciendo particiones espaciales en el sentido de uno de los ejes del equipo (modelado de

parámetros concentrados). La partición de parámetros concentrados (un solo sistema de proceso por fase), es la que se discute a lo largo de este documento y la citada en [1].

3. Programación de simuladores

En este capítulo se abordarán problemas que involucran principalmente procesos de transferencia de masa en operación: lote puro, semi-lote, cocorriente y contracorriente. Primeramente se verá un poco del modelamiento matemático y posteriormente se explicará la programación detallada de cada uno de los casos. Vale la pena resaltar que para realizar este tipo de modelos computacionales, se debe tener en cuenta lo mencionado en [1], ya que en este documento se describen detenidamente los pasos para la realización de este tipo de simuladores. Los pasos mencionados en [1] son: 1- Descripción del proceso y objetivo del modelo, 2- Hipótesis de modelado y nivel de detalle, 3- Definición de los sistemas de proceso, 4- Aplicación de la ley de conservación, 5- Determinación de la estructura básica del modelo, 6- Reconocimiento y acotamiento de variables, parámetros estructurales y constantes, 7- Propuesta de ecuaciones constitutivas y establecimiento de parámetros funcionales, 8- Conteo de grados de libertad, 9- Construcción del modelo computacional, 10- Validación del modelo. En este documento solo se abordarán los pasos 4,5,6,7 y 9. Para mayor información consultar las notas de [1].

3.1. Simulador dinámico y estático de un proceso por lotes de transferencia de masa.

3.1.1. Modelamiento matemático

En este sistema se tomará como ejemplo un proceso de transferencia de masa. Este proceso es realizado por un equipo. En este se cargan dos fases, una líquida que se asume como el refinado y otra sólida que representa la fase extracto. Lo que se pretende con este simulador es poder predecir el tiempo de simulación suficiente para que la fase refinado alcance la concentración deseada.

Para este se se debe tener un valor de las fracciones molares del soluto en el refinado tanto a al ingresar al equipo $y_{A,in}$ como al salir o llegar a la concentración deseada $y_{A,out}$. Para la fase extracto se sabe únicamente la fracción molar inicial $x_{A,in}$ porque la fracción molar a la salida se calcula mediante el método.

Con las fracciones molares de las corrientes en ambas fases, estas se convierten a fracciones molares respecto al solvente libre de soluto.

$$Y_{A,in} = \frac{y_{A,in}}{1 - y_{A.in}} \tag{1}$$

$$Y_{A,out} = \frac{y_{A,out}}{1 - y_{A.out}} \tag{2}$$

$$X_{A,in} = \frac{x_{A,in}}{1 - x_{A.in}} \tag{3}$$

donde A representa el soluto en las dos fases. Realizado el cálculo anterior y sabiendo también la cantidad total a cargar en el equipo de las dos fases, se cálculan las cargas de los solventes puros (libres de soluto) de la siguiente manera:

$$CargaMolR_{FasePura} = CargaMolR_{Cargada}(1 - Y_{A,in}) \tag{4}$$

$$CargaMolE_{FasePura} = CargaMolE_{Cargada}(1 - X_{A,in}) \tag{5}$$

Ahora se buscan expresiones para los coeficientes de transferencia de masa, para algunos parámetros que pueden afectar el valor de los coeficientes como la densidad o la viscosidad. Para este caso se utilizó la siguiente expresión para el coeficiente de transferencia de masa:

$$K_{Y,(i)} = CargaMolR_{(Cargada,i)} * 3{,}8 * 10^{-9} * (\frac{D_{part} CargaMolR_{(Cargada,i)} \bar{M}_R}{\mu_{R,i}(1 - Rel_{fases})})^{-0{,}33} \tag{6}$$

donde el diámetro de partícula D_{part}, la viscosidad μ_R, la masa molar del refinado $\bar{M}_R$ son constantes. El índice (i) hace referencia a la partición de tiempo en el instante actual. Ahora determinada una expresión para el coeficiente, se calcula el flux de transferencia de masa de la siguiente manera:

$$N_{A,(i)} = K_{Y,(i)}(Y_{A,(i)} - Y_{A*,(i)}) \tag{7}$$

es necesario para resolver el método saber que se transfiere son moles de un soluto, por lo tanto al multiplicar el flux de transferencia de masa por el área de transferencia que es una constante y un por el tiempo o una partición de el, se puede obtener el número de moles que se transfieren de una fase a otra en un instante de tiempo determinado.

$$N_{A(PasoTemp,i)} = N_{A,(i)} A_M Paso_{temp} \tag{8}$$

Una vez entendido esto, se actualizan las moles transferidas y las fracciones molares de la siguiente manera:

$$CargaMolR_{(Cargada,i+1)} = CargaMolR_{(Cargada,i)} - N_{A(PasoTemp,i)} \tag{9}$$

$$CargaMolE_{(Cargada,i+1)} = CargaMolE_{(Cargada,i)} + N_{A(PasoTemp,i)} \tag{10}$$

$$Y_{A,(i+1)} = Y_{A,(i)} - \frac{N_{A(PasoTemp,i)}}{CargaMolR_{FasePura}} \tag{11}$$

$$X_{A,(i+1)} = X_{A,(i)} + \frac{N_{A(PasoTemp,i)}}{CargaMolE_{FasePura}} \tag{12}$$

de esta manera, se obtienen las moles transferidas de una fase a la otra, y por ende se puede calcular y actualizar las concentraciones en cada iteración o instante de tiempo. Las condiciones en el tiempo (i+1), se calculan en función de lo que ocurre en el instante de tiempo anterior a (i). Se supone que las concentraciones a la salida en cada instante de tiempo son las condiciones de entrada para el siguiente y así sucesivamente hasta completar el tiempo final de determinado para la simulación.

3.1.2. Programación del modelo de transferencia de masa en operación lote puro en Python

A continuación se muestra la realización de los programas en ***Python 3.7*** utilizando su entorno interactivo web de ejecución de código ***Jupyter Notebook*** para así mostrar la aplicación que tiene este lenguaje de programación en la solución de este tipo de problemas.

Descripción del proceso

- El proceso consiste en transferir un soluto A de una sustancia líquida o refinado a una fase sólida o extracto.
- Se tendrá una torre en donde se introducen dos cargas, una que representa la fase refinado ***R*** el cual es un líquido y otra la fase extracto ***E*** el cual es un sólido.
- Se debe tener en cuenta que ambas fases se cargan en el equipo es decir, se ingresan por lotes.

Datos del proceso

- Fracción molar de soluto en el refinado respecto a la corriente como un todo ***y_A.***
- Fracción molar de soluto en el refinado respecto del solvente libre de soluto ***Y_A.***
- Fracción molar de soluto en el extracto respecto a la corriente como un todo ***x_A.***
- Fracción molar de soluto en el extracto respecto del solvente libre de soluto ***X_A.***
- Carga molar de la fase refinado como un todo ***CargaMol_R.***
- Carga molar de la fase extracto como un todo ***CargaMol_E.***
- Carga molar de la fase refinado respecto del solvente libre de soluto ***CargaR.***
- Carga molar de la fase extracto respecto del solvente libre de soluto ***CargaE.***

MÓDULO 0- Librerías de Python

- En este módulo simplemente se llaman las librerías típicas de Python. Numpy que sirve para realizar algunas operaciones con vectores y matrices, matplotlib que permite gráficar los datos obtenidos, y math que permite el uso funciones matemáticas como Logaritmos, exponenciales, seno, coseno, el número π entre otras.

[ ]:
```
import numpy as np ## Librería de python que permite realizar operaciones con␣
 ↪vectores.
import matplotlib.pyplot as pt ## Librería de python que permite realizar␣
 ↪gráficas.
import math as mt  ##  Librería de Python que permite realizar operaciones␣
 ↪matemáticas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo como: diámetro de una sola partícula o del sólido suponiendo que este posee geometría esférica, su volumen y el área, también las cargas molares de cada fase.

[ ]:
```
Diam_Part=0.002  ##  Diametro de partícula [m].
Vol_Part= (4/3)*mt.pi*(Diam_Part/2)**3  ##  Volumen de una partícula suponiendo␣
 ↪geometría esferica [m^3].
Area_Part=4*mt.pi*(Diam_Part/2)**2  ##  Área de la particula.

CargaMol_R=1.5  ##  Carga molar del refinado [kmol].
CargaMol_E=1.2  ##  Carga molar del extracto [kmol].
```

Parámetros de las sustancias y constantes

- En esta sección se reunen todos los parámetros y constantes que constituyen las ecuaciones que representan el modelo matemático tales como: Masa molar de refinado y extracto, viscosidades, densidades, número de partículas, el área de transferencia de masa y la relación entre las fases.

[ ]:
```
MMolar_R=22  ##  Masa molar refinado R [kg/kmol].
MMolar_E=35  ##  Masa molar extracto E [kg/kmol].
Viscosidad=0.6e-3  ##  Viscosidad del refinado[kg/m s]
```

```
Densidad_R=1000  ##  Densidad del refinado [kg/m^3].
Densidad_E=1850   ##  Densidad del extracto [kg/m^3].

DensidadMol_R=Densidad_R/MMolar_R  ## Densidad molar refinado [Kmol/m^3].
DensidadMol_E=Densidad_E/MMolar_E  ##  Densidad molar extracto [kmol/m^3].
VolR=CargaMol_R*DensidadMol_R  ##  Volumen del refinado en el equipo [m^3].
VolE=CargaMol_E*DensidadMol_E  ##  Volumen del extracto en el equipo [m^3].
Num_Part=VolE/Vol_Part  ##  Número de partículas.
Area_M=Num_Part*Area_Part  ##  Área de tranferencia de masa [m^2].
Rel_Fases=VolE/VolR  ##  Relación de volumen de fases.
```

MÓDULO 2- Condiciones iniciales

- En este módulo se introducen y calculan los valores iniciales para la concentración y la carga de cada una de las fases.

[ ]:
```
yA_in=0.3  ##  Concentración inicial del soluto en el refinado respecto a la␣
 ↪corriente como un todo .
xA_in=0.05  ##  Concentración inicial del soluto en el extracto respecto a la␣
 ↪corriente como un todo.
yA_out=0.19  ##  Concentración final del soluto en el refinado respecto a la␣
 ↪corriente como un todo.


YA_in=((yA_in)/(1-yA_in))  ##  Fracción molar de entrada del soluto en el␣
 ↪refinado respecto al solvente libre de soluto.
XA_in=((xA_in)/(1-xA_in))  ##  Fracción molar de entrada del soluto en el␣
 ↪extracto respecto al solvente libre de soluto.
YA_out=((yA_out)/(1-yA_out))  ##  Fracción molar de salida del soluto en el␣
 ↪extracto respecto al solvente libre de soluto.

CargaE=CargaMol_E*(1-XA_in)  ##  Flux del solvente puro en la fase extracto.␣
 ↪Recordar que estos flux son hipotéticos, ya que no ingresa ni sale masa del␣
 ↪equipo.
CargaR=CargaMol_R*(1-YA_in)  ##  Flux del solvente puro en la fase refinado.

mLdeO=-CargaE/CargaR  ##  Pendiente de la linea de operación.
```

MÓDULO 3- Tiempo de simulación, paso temporal y número de iteraciones

- Este módulo permite establecer el tiempo inicial y final para la simulación. El ***Paso*** es el intervalo de tiempo en el cual se desea realizar la medición, el número de particiones determina las veces que se desea fragmentar el intervalo de tiempo.
- Para la variable llamada ***tiempo***, se crea un vector con ayuda de la librería numpy, el parámetro linspace dentro del comando permite crear un vector desde un valor inicial a uno final dividiendolo en N intervalos ***N_Part***, en este caso se asumen que son intervalos de tiempo.
- La variable ***DiscretCDE*** y ***DiscretLdeO*** son discretizaciones de la curva de distribución de

equilibrio ***CDE*** y la línea de operación ***LdeO***, esto se realiza con el fin de que al cambiar el tiempo de simulación, estas dos variables se puedan graficar también respecto al número de particiones.

[ ]:
```
t_Inicial = 0  ##  Tiempo en que inicia la simulación [s].
t_Final = 6000#int(input("Ingrese el tiempo de simulación [s] = ")) ## Tiempo en␣
 ↪que finaliza la simulación [s].
Paso = 1 ##float(input("ingrese el tamaño del paso para el método Euler [s] =␣
 ↪"))  ## Tamaño de cada partición de intervalo de tiempo.
N_Part=int((t_Final-t_Inicial)/(Paso))  ##  Determina el número de iteraciones␣
 ↪necesarias para resolver el método.
tiempo = np.linspace(t_Inicial, t_Final, N_Part)  ##  Crea vector de tiempo para␣
 ↪poder graficarlo [s].
DiscretCDE=YA_in/N_Part  ##  Discretización de la fracción molar de soluto en el␣
 ↪refinado a la entrada para graficar la CDE.
DiscretLdeO=YA_out/N_Part  ##  Discretización de la fracciόm molar de soluto en␣
 ↪el refinado a la salida para graficar la linea de operación.
```

MÓDULO 4- Creación de vectores para guardar datos

- Este módulo construye vectores de cierta dimensión para guardar los datos de cada iteración para las variables de interes, como se puede ver principalmente se crean vectores para las concentraciones, cargas molares, la viscosidad, el coeficiente de transferencia de masa, la concentración ficticia, el flux de transferencia de masa, la concentración deseada, y las últimas cuatro líneas de código son para poder graficar la ***CDE.***
- La librería ***numpy*** que se llama con las letras ***np*** permite crear vectores de ceros o de unos, el parámetro ***len*** dentro del comando, indica el número de objetos que posee un vector, ***np.zeros*** quiere decir que se va a construir un vector de ceros.
- Las dimensiones para los vectores las otorga ***len(tiempo)***, esto índica que se va a crear un vector de ceros con las dimensiones identicas a las del vector ***tiempo***. Esto último se realiza con el fin de que las dimensiones en los vectores se actualicen automaticamente si se desea cambiar el valor del tiempo de simulación o paso.

[ ]:
```
ConcR=np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para la concentración␣
 ↪del refinado.
ConcE=np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para la concentración␣
 ↪del extracto.
MolesR=np.zeros(len(tiempo)+1) ##  Crea un vector de datos para las moles de␣
 ↪refinado.
MolesE=np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para las moles de␣
 ↪extracto.

Visc=np.zeros(len(tiempo))  ##  Crea un vector de datos para la viscosidad.
KY=np.zeros(len(tiempo))  ##  Crea un vector de datos para el coeficiente de␣
 ↪TdeMasa.
ConcFict=np.zeros(len(tiempo))  ##  Crea un vector de datos para la␣
 ↪concentración ficticia.
```

```
NA=np.zeros(len(tiempo))  ##  Crea un vector de datos para el flux de TdeMasa.
NA_deltat=np.zeros(len(tiempo))  ##  Crea un vector de datos para el flux de␣
 ↪TdeMasa en cada partición de tiempo.
ConcDeseada=np.zeros(len(tiempo))  ##  Crea un vector de datos para la␣
 ↪concentración deseada (fines gráficos).


ConcE_Fig=np.zeros(len(tiempo))  ##  Crea un vector de datos para la␣
 ↪concentración del extracto en la CDE.
CDE=np.zeros(len(tiempo))  ##  Crea un vector de datos para cada punto de la CDE.
ConcR_LdeO=np.zeros(len(tiempo))  ##  Crea un vector de datos para la␣
 ↪concentración del refinado en la linea de operación.
ConcE_LdeO=np.zeros(len(tiempo))  ##  Crea un vector de datos para la␣
 ↪concentración de extracto en la linea de operación.
```

MÓDULO 5- Programación de ecuaciones

Solución de ecuaciones constitutivas

- En esta sección se comienza la solución iterativa del método. Primeramente se le debe ingresar a la primer posición de los vectores creados los valores correspondientes a las condiciones iniciales, tanto para las concentraciones, como para las cargas molares en ambas fases, ya que estas son las variables de interes en la simulación.
- Como se puede observar, en esta sección se introducen todas las ecuaciones del modelo matemático que dependen de las concentraciones. La importancia de ingresar las ecuaciones en este módulo es que estas se actualizan en cada iteración o partición temporal, cosa que no pasaría si por ejemplo la viscosidad se considerara constante, además esto puede influir significativamente en los resultados de la simulación.
- También se ingresa el polinomio que representa las concentraciones ficticias, ya que con esta se calcula el flux de transferencia de masa.
- Se puede observar que el flux de transferencia de masa se actualiza dos veces ***NA*** y ***NA_deltat***, esto debido a que este debe actualizarse en cada partición temporal, como se evidencia al tratarse de una operación en lote puro no hay un flujo molar debido a que ambas fases se encuentran estáticas, por ende la dimensión espacial no se tiene en cuenta en este modelo y solo interesa el tiempo en que la transferencia de masa termina o hay saturación de la fase extracto.
- Por último, se tienen las líneas que actualizan las cargas molares y las concentraciones tanto para el refinado ***R*** como para el extracto ***E***, como se puede ver estas variables se actualizan en el tiempo siguiente [i+1] dependiendo del valor y el cambio que estas en el tiempo anterior [i].

[ ]:
```
# AQUÍ EMPIEZA LA SOLUCIÓN ITERATIVA.##

ConcR[0]=YA_in  ##  Otorga un valor inicial a la concentración de refinado para␣
 ↪resolver el método iterativo.
ConcE[0]=XA_in  ##  Otorga un valor inicial a la concentración de extracto para␣
 ↪resolver el método iterativo.
```

```
MolesR[0]=CargaMol_R  ##  Otorga un valor inicial a las moles de refinado para␣
 ↪resolver el método iterativo.
MolesE[0]=CargaMol_E  ##  Otorga un valor inicial a las moles de extracto para␣
 ↪resolver el método iterativo.

for i in range(0,N_Part,1):

    Visc[i]=Viscosidad+(ConcR[i]/3)**2  ##  Recalcula la viscosidad del refinado.
    KY[i]=MolesR[i]*3.8e-9*((Diam_Part*MolesR[i]*MMolar_R)/
 ↪(Visc[i]*(1-Rel_Fases)))**-0.33  ##  Actualiza el coeficiente englobante de␣
 ↪TdeMasa [kmol/m^2 s].


    ConcFict[i]=(9.524*ConcE[i]**3) -(7.302*ConcE[i]**2)+(2.064*ConcE[i])  ## ␣
 ↪Polinomio que representa la concentración ficticia.

    NA[i]=KY[i]*(ConcR[i]-ConcFict[i])  ##  Recalcula el flux de TdeMasa [kmol/
 ↪m^2 s].
    NA_deltat[i]=NA[i]*Area_M*Paso  ##  Recalcula las moles que se transfieren␣
 ↪en cada partición de tiempo [kmol].


    MolesR[i+1]=MolesR[i]-NA_deltat[i]  ##  Recalcula las moles en el  refinado␣
 ↪[Kmol].
    MolesE[i+1]=MolesE[i]+NA_deltat[i]  ##  Recalcula las moles en el extracto␣
 ↪[kmol].

    ConcR[i+1]=ConcR[i]-((NA_deltat[i])/(CargaR))  ##  Recalcula la␣
 ↪concentración del refinado.
    ConcE[i+1]=ConcE[i]+((NA_deltat[i])/(CargaE))  ##  Recalcula la␣
 ↪concentración del extracto.
```

Solución de ecuaciones para la CDE

- En esta sección se construye la curva de distribución de equilibrio ***CDE*** y la línea de operación.
- La primer línea para ***ConcE_Fig*** únicamente construye punto a punto la concentración del extracto para graficarlo en la CDE.
- La segunda línea para ***CDE*** no es mas que un polinomio que ajusta los datos experimentales para la CDE y poder de esta manera graficar la curva en función de la concentracion del extracto.
- Las siguientes líneas para ***ConcE_LdeO*** y ***ConcR_LdeO*** actualizan y construyen punto a puento las concentraciones en las dos fases para poder construir la línea de operación.
- Las expresiones para ***XA_final*** y ***YA_final*** son únicamente para graficar un punto en la CDE que represente las concentraciones finales para las dos fases.
- Para la ConcDeseada únicamente se llena cada iteración con el mismo valor, el valor deseado en la fase refinado. Esta acción se realiza para graficar una línea recta en la CDE para

reconocer la concentración deseada.

- Las últimas líneas de código se utilizan para ajustar los vectores a las dimensiones correspondientes.

```
[ ]:    ConcE_Fig[i]=i*DiscretCDE  ##  Construye punto a punto la concentracion del␣
 ↪extracto para graficarlo en la CDE.
    CDE[i]=9.524*ConcE_Fig[i]**3 -7.302*ConcE_Fig[i]**2+2.064*ConcE_Fig[i]  ## ␣
 ↪Polinomio para graficar la CDE, este se gráfica punto a punto de acuerdo a la␣
 ↪discretización realizada para la CDE.



    ConcE_LdeO[i]=ConcE[0]+i*DiscretLdeO  ##  Aumenta a la concentración del␣
 ↪extracto el valor de cada partición para graficarlo.
    ConcR_LdeO[i]=ConcR[0]+mLdeO*(ConcE_LdeO[i]-ConcE[0])  ##  Cálcula y␣
 ↪actualiza la concentración del refinado en la CDE en función de la␣
 ↪concentración del extracto.

    XA_final=ConcE[i+1]  ##  Otorga el valor a concentración final de A en el␣
 ↪extracto para graficar dicho punto.
    YA_final=ConcR[0]+mLdeO*(XA_final-ConcE[0])  ##  Cálcula la concentración␣
 ↪final de A en el refinado para graficar dicho punto.

    ConcDeseada[i]=YA_out  ##  Llena el vector de concentración deseada con los␣
 ↪datos de concentración de salida que se desea para poder graficar la linea␣
 ↪recta.

ConcR=ConcR[:i+1]  ##  Ajusta el vector a las dimensiones requeridas.
ConcE=ConcE[:i+1]  ##  Ajusta el vector a las dimensiones requeridas.
MolesR=MolesR[:i+1]  ##  Ajusta el vector a las dimensiones requeridas.
MolesE=MolesE[:i+1]  ##  Ajusta el vector a las dimensiones requeridas.
```

MÓDULO 6- Gráficas y resultados

- En este módulo se utiliza la librería matplotlib que permite gráficar los resultados obtenidos.
- Las líneas de código son simplemente acciones y características para poder visualizar los datos.

Gráfica: Concentraciones vs tiempo

```
[ ]: pt.figure("TdeMasa Lote puro", [10,5])  ##  Crea figura y le otorga dimensiones.
pt.subplot(1,2,1)   ##  Hace una sub-figura (N°filas , N°columnas , Posicion de␣
 ↪la figura en el subplot).
pt.tight_layout(pad=5, w_pad=5, h_pad=4)  ##  Espaciado entre las figuras del␣
 ↪subplot (Espaciado entre la margen , Espaciado entre figuras (Horizontal) ,␣
 ↪Espaciado entre figuras (Vertical).
pt.plot(tiempo,ConcR,'k',linewidth=2,label='Concentración en el Refinado')  ## ␣
 ↪Grafica el tiempo vs Concentración del refinado.
```

```
pt.plot(tiempo,ConcE,'b',linewidth=2,label='Concentración en el Extracto')  ## ␣
 ↪Grafica el tiempo vs Concentración del extracto.
pt.plot(tiempo,ConcFict,'y',linewidth=2,label='Concentración ficticia')  ## ␣
 ↪Grafica el tiempo vs Concentración ficticia.
pt.plot(tiempo,ConcDeseada,'r',linewidth=2)  ##  Grafica el tiempo vs␣
 ↪Concentración deseada.
pt.xlabel("Tiempo [s]")  ##  Agrega título al eje x.
pt.ylabel('Concentración respecto\n del solvente libre de soluto ')  ##  Agrega␣
 ↪título al eje y.
pt.title('Concentración vs tiempo')  ##  Agrega título en la parte superior.
pt.grid(True)  ##  Agrega la cuadrilla a la imagen
pt.tick_params(labelsize=8)  ##  Ajusta el tamaño de los títulos para los ejes ␣
 ↪(x , y), y de igual manera ajusta el tamaño de los números en cuadrilla(Grid).
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 7})
leg.get_frame().set_alpha(0.5)  ##  Agrega la leyenda en la grafica.("best" es␣
 ↪la mejor ubicación).
pt.show()
```

Gráfica: Curva de distribución de equilibrio CDE y línea de operación

[ ]:
```
pt.subplot(1,2,2)
pt.grid(True)
pt.plot(ConcE_Fig,CDE,'k',label='CDE')
pt.plot(ConcE_LdeO,ConcR_LdeO,'y',label='Linea de operación guia (imaginaria)')
pt.plot(XA_in,YA_in,'ko',label="Punto inicial de operación")
pt.plot(XA_final,YA_final,'ro',label='Punto final de operación')
pt.plot(ConcE,ConcR,'b')
pt.xlabel('Concentración extracto')
pt.ylabel('Concentración refinado')
pt.title('Curva de distribución de equilibrio (CDE)')
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 7})
leg.get_frame().set_alpha(0.5)
pt.show()
```

3.2. Simulador dinámico y estático de un proceso por semi lotes de transferencia de masa

3.2.1. Modelamiento matemático

Para este sistema se tomara como ejemplo un proceso de transferencia de masa. Este proceso es realizado por un equipo o torre con lecho empacado. En este equipo se encuentra una fase cargada o estática, en este caso la fase extracto (empaque), y una fase refinado (Gas) que es continua y atraviesa el lecho poroso.

La diferencia de este sistema con el anterior es que las variables cambian tanto con el tiempo como con el espacio, por lo tanto se tendrá en cuenta que este sistema se ve afectado por una dimensión

espacial y una temporal. El procedimiento para determinar las moles transferidas es practicamente el mismo que por lotes, únicamente con la diferencia de que las ecuaciones que actualizan el flux molar de ambas fases se llenan teniendo en cuenta las dimensiones mencionadas.

Primero se calculan las fracciones molares como se realizó en las ecuaciones 1,2,3 para el proceso por lotes.

Realizado el cálculo anterior y sabiendo también la cantidad total a cargar en el equipo de la fase estática o extracto y el flux molar de la fase continua o refinado, se calculan las cargas de los solventes puros (libres de soluto) de la siguiente manera:

$$FluxMolR_{FasePura} = FluxMolR_{Entrada}(1 - Y_{A,in}) \tag{13}$$

$$CargaE_{FasePura} = CargaE_{Cargada}(1 - X_{A,in}) \tag{14}$$

Primero se determina una ecuación para el coeficiente de transferencia de masa.

$$K_{Y,(i,j)} = FluxMolR_{(FasePura,(i,j))} * 1{,}15 * (\frac{D_{emp} FluxMolR_{(FasePura,(i,j))} * \bar{M}_R}{\mu_R(1 - Porosidad)})^{-0{,}36} \tag{15}$$

Los índices (i) y (j), hacen referencia al número de instante de tiempo y partición de la torre respectivamente. Ahora se calcula el flux molar para cada tramo de la torre de la siguiente manera:

$$N_{A,(i,j)} = K_{Y,(i,j)}(Y_{A,(i,j)} - Y_{A*,(i,j)}) \tag{16}$$

$$N_{APasoEsp,(i,j)} = N_{A(i,j)} aMPaso_{esp} \tag{17}$$

por último se actualizan los flux y las concentraciones en ambas fases.

$$FluxMolR_{(i,j+1)} = FluxMolR_{(i,j)} - N_{APasoEsp,(i,j)} \tag{18}$$

$$CargaMolE_{(i+1,j+1)} = CargaMolE_{(i,j)} + N_{APasoEsp,(i,j)} \tag{19}$$

$$Y_{A,(i,j+1)} = Y_{A,(i,j)} - \frac{N_{APasoEsp,(i,j)}}{FluxMolR_{FasePura}} \tag{20}$$

$$X_{A,(i+1,j)} = X_{A,(i,j)} + \frac{N_{APasoEsp,(i,j)}}{FluxMolR_{FasePura}} \tag{21}$$

3.2.2. Programación del modelo de transferencia de masa en operación semi lote en Python

A continuación se muestra la realización de los programas en ***Python 3.7*** utilizando su entorno interactivo web de ejecución de código ***Jupyter Notebook*** para así mostrar la aplicación que tiene este lenguaje de programación en la solución de este tipo de problemas.

Descripción del proceso

- El proceso consiste en transferir masa, en este caso un soluto A de una fase gaseosa o refinado a una fase sólida o extracto.
- Como equipo se supondrá una torre empacada en donde la fase que se carga o que permanece estática es la fase extracto ***E***, y la fase refinado es la fase continua.
- Se debe tener en cuenta que en este proceso una fase se encuentra estática en la torre y la otra esta en movimiento por eso se nombre semi-lote.

Datos del proceso

- Fracción molar de soluto en el refinado respecto a la corriente como un todo ***y_A.***
- Fracción molar de soluto en el refinado respecto del solvente libre de soluto ***Y_A.***
- Fracción molar de soluto en el extracto respecto a la corriente como un todo ***x_A.***
- Fracción molar de soluto en el extracto respecto del solvente libre de soluto ***X_A.***
- Carga molar de la fase refinado como un todo ***FluxMol_R.***
- Carga molar de la fase extracto como un todo ***CargaMol_E.***
- Carga molar de la fase refinado respecto del solvente libre de soluto ***FluxMolR.***
- Carga molar de la fase extracto respecto del solvente libre de soluto ***CargaE.***

MÓDULO 0- Librerías de Python

- En este módulo simplemente se llaman las librerías típicas de Python. Numpy que sirve para realizar algunas operaciones con vectores y matrices, matplotlib que permite gráficar los datos obtenidos, y math que permite el uso funciones matemáticas como Logaritmos, exponenciales, seno, coseno, el número π entre otras.

[ ]:
```
import numpy as np ## Librería de python que permite realizar operaciones con␣
 ↪vectores.
import matplotlib.pyplot as pt ## Librería de python que permite realizar␣
 ↪gráficas.
import math as mt  ##  Librería de Python que permite realizar operaciones␣
 ↪matemáticas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo: diámetro del empaque utilizado, la longitud de la torre, el flux molar de la fase refinado, la carga molar del empaque utilizado, el área de flujo, el área de transferencia de masa y la porosidad del lecho.

[ ]:
```
Diam_Emp= 0.05  ##  Díametro del empaque usado [m].
Long= 2  ##  Longitud de la torre [m].
FluxMol_R= 0.065  ##  Flux  molar del refinado [kmol/m{2 s}].
CargaMol_E= 1  ##  Carga molar del extracto [kmol].
Area_F= 0.9  ##  Área de flujo de la torre [m^2].
aM=  37.4  ##  Área de tranferencia de masa [m^2/m^3].
Porosidad= 0.55  ##  Porosidad del lecho.
```

Parámetros de las sustancias y constantes

- En esta sección se reunen todos los parámetros y constantes que constituyen las ecuaciones que representan el modelo matemático tales como: Masa molar de refinado y extracto, viscosidades, densidades entre otras.

[ ]:
```
MMolar_R=11  ##  Masa molar refinado R [kg/kmol].
MMolar_E=35  ##  Masa molar extracto E [kg/kmol].
Viscosidad=1e-5  ##  Viscosidad del refinado[kg/m s]. 1e-5
DensidadM_Emp=5  ##  Densidad del empaque de la torre [kmol/m^3].
```

MÓDULO 2- Condiciones iniciales

- En este módulo se introducen y cálculan los valores iniciales para la concentración, la carga o el flux de cada una de las fases. También se introduce un parámetro llamado ***ConcSat_E*** el cual indica el grado de saturación de soluto en el empaque de la torre.

```
[ ]: yA_in= 0.3  ##  Concentración inicial del soluto en el refinado respecto a la␣
      ↪corriente como un todo.
     xA_in= 0.05  ##  Concentración inicial del soluto en el extracto respecto a la␣
      ↪corriente como un todo.
     yA_out= 0.18  ##  Concentración final del soluto en el refinado respecto a la␣
      ↪corriente como un todo.

     YA_in= ((yA_in)/(1-yA_in))  ##  Fracción molar de entrada del soluto en el␣
      ↪refinado respecto del solvente libre de soluto.
     XA_in= ((xA_in)/(1-xA_in))  ##  Fracción molar de entrada del soluto en el␣
      ↪extracto respecto del solvente libre de soluto.
     YA_out= ((yA_out)/(1-yA_out))  ##  Fracción molar de salida del soluto en el␣
      ↪extracto respecto del solvente libre de soluto.

     CargaE= CargaMol_E*(1-XA_in)  ##  Carga de solvente puro en la fase extracto␣
      ↪[kmol].
     FluxMolR= FluxMol_R*(1-YA_in)  ##  Flux del solvente puro en la fase refinado␣
      ↪[Kmol/m^2 s].
     ConcSat_E= 0.28  ##  Concentración de saturación del empaque.
```

MÓDULO 3- Tiempo de simulación, paso temporal, paso espacial y número de iteraciones

Paso temporal

- Este módulo permite establecer el tiempo inicial y final para la simulación. El ***PasoTemp*** es el intervalo de tiempo en el cual se desea realizar la medición.
- Para la variable llamada ***tiempo***, se crea un vector con ayuda de la librería numpy, el parámetro linspace dentro del comando que permite crear un vector desde un valor inicial a uno final dividiendolo en N intervalos ***NumPasosTemp***, en este caso se asumen que son intervalos de tiempo.
- La variable ***DiscretCDE*** y ***DiscretLdeO*** son discretizaciones de la curva de distribución de equilibrio ***CDE*** y de la línea de operación ***LdeO***, esto se realiza con el fin de que al cambiar el tiempo de simulación, estas dos variables se puedan graficar también respecto al número de particiones temporales.

```
[ ]: t_Inicial = 0  ##  Tiempo en que inicia la simulación [s].
     t_Final = 500#int(input("Ingrese el tiempo de simulación [s] = ")) ## Tiempo en␣
      ↪que finaliza la simulación [s].
     PasoTemp = 1 ##float(input("ingrese el tamaño del PasoTemp para el método Euler␣
      ↪[s] = "))  ## Tamaño de cada partición de intervalo de tiempo.
     NumPasosTemp= int((t_Final-t_Inicial)/(PasoTemp))  ##  Determina el número de␣
      ↪particiones necesarias para resolver el método.
```

```
tiempo = np.linspace(t_Inicial, t_Final, NumPasosTemp)  ##  Crea vector de␣
 ↪tiempo para poder graficarlo [s].
DiscretCDE= YA_in/NumPasosTemp  ##  Discretización de la fracción molar de␣
 ↪soluto en el refinado a la entrada para graficar en la CDE.
DiscretLdeO= YA_out/NumPasosTemp  ##  Discretización de la fracción molar de␣
 ↪soluto en el refinado a la salida para graficar la linea de operación.
```

Paso espacial

- En esta simulación, se incluirá también la dimensión espacial, ya que las concentración en una de las fases "extracto" cambia con el tiempo y con el espacio.
- Al igual que se realizó con el tiempo, se ingresa la altura final de la torre, se define un paso espacial, y de acuerdo con este se calcula un número de pasos espaciales ***NumPasosEsp*** o número de tramos en que se divide la altura de la torre.
- Se define un vector llamado ***Longitud*** que representa lo mismo que el vector ***tiempo*** pero con particiones espaciales, por último se calcula la carga molar en cada tramo para la fase sólida o extracto.

[ ]:
```
L_Inicial=0  ##  Valor de la longitud en el punto en donde inicia la simulación.
L_Final=Long  ##  Valor de la longitud final de la torre.
PasoEsp=0.05  ##  Valor para cada tramo en que se va a analizar la torre o paso␣
 ↪espacial.

PasoEspVol=PasoEsp*Area_F  ##  Paso espacial para cada tramo del volumen.
NumPasosEsp=int((L_Final-L_Inicial)/(PasoEsp))  ##  Número de pasos espaciales.
Longitud=np.linspace(L_Inicial,L_Final,NumPasosEsp)  ##  Crea un vector de␣
 ↪longitud para poder graficarlo.
CargaTramo=DensidadM_Emp*PasoEspVol  ##  Valor de la carga molar del sólido para␣
 ↪cada tramo de la torre.
```

MÓDULO 4- Creación de vectores para guardar datos

- En este módulo se construyen matrices de cierta dimensión para guardar los datos de cada iteración para las variables de interes, como se puede ver principalmente se crean matrices para las concentraciones, cargas molares, la viscosidad, el coeficiente de transferencia de masa, la concentración ficticia, el flux de transferencia de masa, la concentración deseada, y las últimas cuatro líneas de código son para poder graficar la ***CDE***, observese que estos vectores dependen tanto de la dimensión temporal como de la espacial.
- La librería ***numpy*** que se llama con las letras ***np*** permite crear matrices de ceros o de unos, el parámetro ***len*** desde el comando, indica el número de objetos que posee un vector, en este caso el temporal y espacial, ***np.zeros*** quiere decir que se va a construir una matriz de ceros.
- Las dimensiones de las matrices las otorga : ***len(tiempo)*** para las filas y ***len(Longitud)*** para las columnas , esto indica que se va a crear una matriz de ceros que depende de las dimensiones de los vectores ***Tiempo*** y ***Longitud***. Esto último se realiza con el fin de que las dimensiones en las matrices se actualicen automaticamente si se desea cambiar el valor del tiempo final de simulación o modificar la altura de la torre.

```
[ ]: ConcR=np.zeros((len(tiempo),len(Longitud)+1))  ##  Crea un vector de datos para
      ↪la concentración del refinado.
     ConcE=np.zeros((len(tiempo)+1,len(Longitud)))  ##  Crea un vector de datos para
      ↪la concentración del extracto.
     FluxR=np.zeros((len(tiempo)+1,len(Longitud)+1)) ##  Crea un vector de datos para
      ↪las moles de refinado.
     MolesE=np.zeros((len(tiempo)+1,len(Longitud)+1))  ##  Crea un vector de datos
      ↪para las moles de extracto.

     Visc=np.zeros((len(tiempo),len(Longitud)))  ##  Crea un vector de datos para la
      ↪viscosidad.
     K_Liq=np.zeros((len(tiempo),len(Longitud)))  ##  Crea un vector de datos para el
      ↪coeficiente de TdeMasa.
     KY=np.zeros((len(tiempo),len(Longitud)))  ##  Crea un vector de datos para el
      ↪coeficiente de TdeMasa englobante.
     ConcFict=np.zeros((len(tiempo),len(Longitud)))  ##  Crea un vector de datos para
      ↪la concentración ficticia.
     NA=np.zeros((len(tiempo),len(Longitud)))  ##  Crea un vector de datos para el
      ↪flux de TdeMasa.
     NA_PasoEsp=np.zeros((len(tiempo),len(Longitud)))  ##  Crea un vector de datos
      ↪para el flux de TdeMasa en cada partición de tiempo.
     ConcDeseada=np.zeros((len(tiempo),len(Longitud)))   ##  Crea un vector de datos
      ↪para la concentración deseada (fines graficos).

     ConcE_Fig=np.zeros(len(tiempo))  ##  Crea un vector de datos para la
      ↪concentración del extracto en la CDE.
     CDE=np.zeros(len(tiempo))  ##  Crea un vector de datos para cada punto de la CDE.
     ConcR_LdeO=np.zeros(len(tiempo))  ##  Crea un vector de datos para la
      ↪concentración del refinado en la linea de operación.
     ConcE_LdeO=np.zeros(len(tiempo))  ##  Crea un vector de datos para la
      ↪concentración de extracto en la linea de operación.
```

MÓDULO 5- Programación de ecuaciones

Solución y actualizacion de ecuaciones constitutivas

- Se debe ingresar a la primer posición de las matrices creadas los valores correspondientes a las condiciones iniciales para las concentraciones molares, carga molar de la fase refinado, el flux molar de la fase extracto, ya que estas son las variables de interes en la simulación.
- Posteriormente se comienza con llenado de las matrices en en función del tiempo (m) y el espacio (n) con su valor inicial. El primer ciclo for, llena toda la matriz para las concentraciones en ambas fases con el valor inicial porque se supone que al iniciar el proceso, las concentraciones en toda la torre son iguales.
- Como se puede observar, en esta sección se introducen todas las ecuaciones del modelo matemático que dependen de las concentraciones en el segundo ciclo for. La importancia de ingresar las ecuaciones en este módulo, es que estas se actualizan en cada iteración o partición temporal (i) y espacial (j), cosa que no pasaria si por ejemplo la viscosidad se considerara

constante, además esto puede influir significativamente en los resultados de la simulación.

- También se ingresa el polinomio que representa a las concentración ficticia, ya que con esta se calcula el flux de transferencia de masa.
- Los condicionales if y else, solo ponen límites al método. El primer condicional indica el cálculo del coeficiente de transferencia de masa en caso tal que se llegue a la concentración de saturación del empaque, si pasa esto significa que ya no hay transferencia de masa, ya que el empaque físicamente se encontraría completamente saturado de soluto.
- El segundo condicional if, elif y else, ponen límites a la concentración fictica para graficarla en la CDE.
- Posteriormente, se calcula el flux de transferencia de masa en función de la concentracíon ficticia, luego este mismo flux se recalcula en cada tramo de la torre ***NA_PasoEsp***.
- Despues de realizar el paso anterior, se actualizan las concentraciones y los flux o las cargas para cada una de las fases. El llenado para la matrices que representan la concentración ***ConcR(i)(j+1)*** y el flux del refinado ***FluxR(i)(j+1)*** se realizan de la misma manera. Se comienza llenando desde la columna dos porque la columna uno posee los valores del primer fragmento de la torre en función del tiempo, para esta variable, este valor es el mismo "el inicial", ya que físicamente el flux para la fase refinado es constante en el tiempo, pero para cada partición de la torre no lo es, es decir cada partición posee un flux diferente pero este es constante en el tiempo. Las últimas dos líneas de código se encargan de realizar esta acción, dandole siempre el mismo valor "el inicial" a la primer columna de la matriz tanto para el flux en la fase refinado como para la carga en la fase extracto.
- Para actualizar la matriz que representa la concentracion ***ConcE(i+1)(j)*** y las moles ***MolesE(i+1)(j+1)*** de la fase extracto se debe tener en cuenta el significado físico de cada variable. Como se puede ver las matrices se llenan diferente debido a que en esta fase no existe un flux molar constante en el tiempo, por el contrario esta es la fase que se encuentra estática en la torre. El llenado para la concentración del extracto ***ConcE(i+1)(j)*** se realiza respecto a la concentración en el instante de tiempo anterior, como se observa la concentración en la matriz para la fase extracto debe ser la misma en la primer fila, ya que esta representa la concentración inicial en toda la longitud de la torre en el tiempo inicial, por este motivo esta matriz se llena desde el instante de tiempo siguiente o (i+1). Para el llenado de la matriz que representa las moles para la fase extracto ***MolesE(i+1)(j+1)***, se debe tener en cuenta que esta depende de dos dimensiones debido a que las moles transferidas cambian en el instante de tiempo y tramo de la torre anterior. La explicación a esto es porque la fase extracto es un sólido que se carga y se encuentra estático en la torre y a este lo atraviesa una fase continua que le entrega soluto, esta fase ingresa por un tramo de la torre ocasionando que la carga molar en el empaque no sea la misma ni en el primer tramo de la torre ni en el siguiente, además esta variable tambien cambia al pasar el tiempo.
- Se puede ver que es de vital importancia comprender fisicamente lo que sucede dentro de la torre y como se comportan cada una de las variables en el proceso, como se vio anteriormente el llenado de las matrices se realiza de acuerdo al comportamiento de cada una de estas variables en la torre.

[ ]:
```
FluxR[0][0]=FluxMol_R  ##  Otorga un valor inicial a las moles de refinado para␣
 ↪resolver el método iterativo.
MolesE[0][0]=CargaMol_E/NumPasosEsp  ##  Otorga un valor inicial a las moles de␣
 ↪extracto para resolver el método iterativo.
```

```
for m in range(0,NumPasosTemp,1):
    for n in range(0,NumPasosEsp,1):

        ConcR[m][n]=YA_in  ##  Otorga un valor inicial a la concentración de␣
↪refinado para resolver el método iterativo.
        ConcE[m][n]=XA_in  ##  Otorga un valor inicial a la concentración de␣
↪extracto para resolver el método iterativo.

        ConcDeseada[m][n]=YA_out  ##  Otorga a la matriz de concentración␣
↪deseada valores de la concentración de salida.(Fines graficos).

for i in range(0,NumPasosTemp,1):
    for j in range(0,NumPasosEsp,1):

        Visc[i][j]=Viscosidad+(ConcR[i][j]/100)**2  ##  Cálcula y actualiza la␣
↪viscosidad del refinado.
        K_Liq[i][j]=FluxR[i][j]*1.15*((Diam_Emp*FluxR[i][j]*MMolar_R)/
↪(Visc[i][j]*(1-Porosidad)))**-0.36  ##  Cálcula y actualiza el coeficiente␣
↪englobante de TdeMasa [kmol/m^2 s].


        if (ConcSat_E-ConcE[i][j])>0:  ##  Condiciona y pone límites al␣
↪coeficiente de TdeMasa englobante.

            KY[i][j]=K_Liq[i][j]*(ConcSat_E-ConcE[i][j])**0.5
        else:
            KY[i][j]=K_Liq[i][j]*0

        if ConcFict[i][j]>0.3:  ##  Condiciona o pone límites para la␣
↪concentración en la concentración ficticia.
            ConcFict[i][j]=0.3

        elif ConcFict[i][j]<1e-5:
            ConcFict[i][j]=1e-5

        else:

            ConcFict[i][j]=(9.524*ConcE[i][j]**3) -(7.302*ConcE[i][j]**2)+(2.
↪064*ConcE[i][j])  ##  Polinomio que representa la concentración ficticia.     ␣
↪


        NA[i][j]=KY[i][j]*(ConcR[i][j]-ConcFict[i][j])  ##  Cálcula y actualiza␣
↪el flux de TdeMa2sa [kmol/m^2 s].
        NA_DeltaL[i][j]=NA[i][j]*aM*PasoEsp  ##  Cálcula y actualiza flujo molar␣
↪en cada partición de tiempo [kmol/ s].
```

```
        FluxR[i][j+1]=FluxR[i][j]-NA_DeltaL[i][j]  ##  Recalcula el flux molar␣
↪en el refinado refinado [Kmol].

        ConcR[i][j+1]=ConcR[i][j]-((NA_DeltaL[i][j])/(FluxMolR))  ##  Recalcula␣
↪la concentración en el refinado.

        if ConcR[i][j+1]<0:  ##  Condiciona y pone límites a la concentración en␣
↪el refinado.
            ConcR[i][j+1]=0  ##  Ya que la concentración no puede ser negativa␣
↪esta se queda en cero.

        MolesE[i+1][j+1]=MolesE[i][j]+NA_DeltaL[i][j]  ##  Recalcula las moles␣
↪en el extracto [kmol].
        ConcE[i+1][j]=ConcE[i][j]+((NA_DeltaL[i][j])/(CargaTramo))  ## ␣
↪Recalcula la concentración en el extracto.

        FluxR[i+1][0]=FluxR[0][0]  ##  Otorga nuevamente el valor inicial del␣
↪flux molar en el refinado a la primer partición en todo el tiempo.
        MolesE[i+1][0]=MolesE[0][0]  ##  Otorga nuevamente el valor inicial a␣
↪las moles en el extracto a la primer partición en todo el tiempo.
```

Solución de ecuaciones para la CDE

- En esta sección se construye la curva de distribución de equilibrio ***CDE*** y la línea de operación.
- La primer línea para ***ConcE_Fig*** únicamente construye punto a punto la concentración del extracto para graficarlo en la CDE.
- La segunda línea para ***CDE*** no es más que un polinomio que ajusta los datos experimentales para la CDE y poder de esta manera graficar la curva en función de la concentración del extracto.
- Las siguientes líneas para ***ConcE_LdeO*** y ***ConcR_LdeO*** actualizan y construyen punto a puento las concentraciones en las dos fases para poder construir la línea de operación.
- Las expresiones para ***XA_final*** y ***YA_final*** son únicamente para graficar un punto en la CDE que represente las concentraciones finales para las dos fases.
- Para la ConcDeseada únicamente se llena cada iteración con el mismo valor, el valor deseado en la fase refinado. Esta acción se realiza solo para graficar una línea recta en la CDE que represente la concentración deseada.

[ ]:
```
        mLdeO=((-CargaE/t_Final)/(FluxMolR)) ##  Pendiente de la linea de␣
↪operación.
        ConcE_Fig[i]=i*DiscretCDE  ##  Construye punto a punto la concentracion␣
↪del extracto para graficarlo en la CDE.
        CDE[i]=9.524*ConcE_Fig[i]**3 -7.302*ConcE_Fig[i]**2+2.064*ConcE_Fig[i] ␣
↪##  Polinomio para graficar la CDE, este se gráfica punto a punto de acuerdo a␣
↪la discretización realizada para la CDE.
```

```
        ConcE_LdeO[i]=ConcE[0][0]+i*DiscretLdeO  ##  Aumenta a la concentración␣
↪del extracto el valor de la partición asignada para graficarlo.
        ConcR_LdeO[i]=ConcR[0][0]+mLdeO*(ConcE_LdeO[i]-ConcE[0][0])  ##  Cálcula␣
↪y actualiza la concentración del refinado en la CDE en función de la␣
↪concentración del extracto.

        XA_final=ConcE[i+1]  ##  Otorga el valor a concentración final de A en␣
↪el extracto.
        YA_final=ConcR[0][0]+mLdeO*(XA_final-ConcE[0][0])  ##  Cálcula la␣
↪concentración final de A en el refinado.
```

MÓDULO 6- Gráficas y resultados

- En este módulo se utiliza la librería matplotlib que permite graficar los resultados obtenidos.
- Las líneas de código son simplemente acciones y características para poder visualizar los datos.
- El primer grupo para el tiempo, se realiza únicamente para poder graficar la concentración en cinco instantes de tiempo determinados a lo largo de la torre. El segundo grupo realiza la acción contraria, grafica la concentración en cinco tramos determinados de la torre en función del tiempo.

[ ]:
```
#  Otorga un valores para las particiones temporales donde se desea ver el␣
↪comportamiento gráfico del modelo.

tiempo1=NumPasosTemp*0
tiempo2=int(NumPasosTemp*0.1)
tiempo3=int(NumPasosTemp*0.2)
tiempo4=int(NumPasosTemp*0.3)
tiempo5=int(NumPasosTemp*0.4)

#  Otorga un valores para las particiones o tramos de la torre donde se desea␣
↪ver el comportamiento gráfico del modelo.

parti1=NumPasosEsp*0
parti2=int(NumPasosEsp*0.1)
parti3=int(NumPasosEsp*0.2)
parti4=int(NumPasosEsp*0.5)
parti5=int(NumPasosEsp*0.9)
```

Gráfica: Concentración en tiempos determinados para la fase refinado vs longitud

[ ]:
```
pt.figure("TdeMasa Semi Lote. COMPORTAMIENTO DE LA CONCENTRACIÓN EN LA TORRE EN␣
↪EL TIEMPO Y EL ESPACIO", [8,6])  ##  Crea figura y le otorga dimensiones.
pt.subplot(2,2,1)
```

```
pt.tight_layout(pad=5, w_pad=4, h_pad=4)  ##  Espaciado entre las figuras del␣
 ↪subplot (Espaciado entre la margen , Espaciado entre figuras (Horizontal) ,␣
 ↪Espaciado entre figuras (Vertical).
pt.plot(Longitud,ConcR[tiempo1,1:],'r',label='Tiempo 0 [s]',linewidth=2)  ## ␣
 ↪Grafica concentración en el refinado en función de la longitud a un instante␣
 ↪de tiempo determinado.
pt.plot(Longitud,ConcR[tiempo2,1:],'k',label='Tiempo 50 [s]',linewidth=2)
pt.plot(Longitud,ConcR[tiempo3,1:],'b',label='Tiempo 100 [s]',linewidth=2)
pt.plot(Longitud,ConcR[tiempo4,1:],'y',label='Tiempo 150 [s]',linewidth=2)
pt.plot(Longitud,ConcR[tiempo5,1:],'g',label='Tiempo 200 [s]',linewidth=2)
pt.plot(Longitud,ConcDeseada[0][0:],'c',label='Concentración␣
 ↪deseada',linewidth=2)
pt.grid(True)  ##  Agrega la cuadrilla a las gráficas.
pt.xlabel("Longitud [m]")  ##  Agrega el título al eje x.
pt.ylabel("Concentración en el Refinado")  ##  Agrega título al eje y.
pt.title("REFINADO (GAS)")  ##  Agrega título a la grafica.
pt.tick_params(labelsize=8)  ##  Ajusta el tamaño de los títulos para los ejes ␣
 ↪(x , y), y de igual manera ajusta el tamaño de los números en cuadrilla(Grid).
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 5})  ##  Otorga propiedades al cuadro de leyendas.
leg.get_frame().set_alpha(0.5)  ##  Otorga la propiedade de que la leyenda se␣
 ↪vea transparente.
pt.show()
```

Gráfica: Concentración en tiempos determinados para la fase extracto vs longitud

[ ]:
```
pt.subplot(2,2,2)
pt.plot(Longitud,ConcE[tiempo1][0:],'r', label='Tiempo = 0 [s]',linewidth=2)
pt.plot(Longitud,ConcE[tiempo2][0:],'k',label='Tiempo 50 [s]',linewidth=2)
pt.plot(Longitud,ConcE[tiempo3][0:],'b',label='Tiempo 100 [s]',linewidth=2)
pt.plot(Longitud,ConcE[tiempo4][0:],'y',label='Tiempo 150 [s]',linewidth=2)
pt.plot(Longitud,ConcE[tiempo5][0:],'g',label='Tiempo 200 [s]',linewidth=2)
pt.xlabel("Longitud [m]")
pt.ylabel("Concentración en el Extracto")
pt.title("EXTRACTO (SÓLIDO)")
pt.grid(True)
pt.tick_params(labelsize=8)
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 5})
leg.get_frame().set_alpha(0.5)
```

Gráfica: Concentración en tramos de la torre determinados para la fase refinado vs tiempo

[ ]:
```
pt.subplot(2,2,3)
pt.plot(tiempo,ConcR[0:,parti1],'r',label='Longitud 0 [m]',linewidth=2)
pt.plot(tiempo,ConcR[0:,parti2],'k',label='Longitud 0.2 [m]',linewidth=2)
pt.plot(tiempo,ConcR[0:,parti3],'b',label='Longitud 0.4 [m]',linewidth=2)
```

```
pt.plot(tiempo,ConcR[0:,parti4],'y',label='Longitud 1 [m]',linewidth=2)
pt.plot(tiempo,ConcR[0:,parti5],'g',label='Longitud 1.8 [m]',linewidth=2)
pt.plot(tiempo,ConcDeseada[0:,0],'c',label='Concentración deseada',linewidth=2)
pt.grid(True)
pt.xlabel("tiempo [s]")
pt.ylabel("Concentración en el Refinado")
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 5})
leg.get_frame().set_alpha(0.5)
pt.show()
```

Gráfica: Concentración en tramos de la torre determinados para la fase extracto vs tiempo

[ ]:
```
pt.subplot(2,2,4)
pt.plot(tiempo,ConcE[1:,parti1],'r',label='Longitud 0 [m]',linewidth=2)
pt.plot(tiempo,ConcE[1:,parti2],'k',label='Longitud 0,2 [m]',linewidth=2)
pt.plot(tiempo,ConcE[1:,parti3],'b',label='Longitud 0.4 [m]',linewidth=2)
pt.plot(tiempo,ConcE[1:,parti4],'y',label='Longitud 1 [m]',linewidth=2)
pt.plot(tiempo,ConcE[1:,parti5],'g',label='Longitud 1.8 [m]',linewidth=2)
pt.grid(True)
pt.xlabel("tiempo [s]")
pt.ylabel("Concentración en el extracto")
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 5})
leg.get_frame().set_alpha(0.5)  ##  Agrega la leyenda en la grafica.("best" es␣
 ↪la mejor ubicación).
pt.show()
```

Gráfica: Curva de distribución de equilibrio CDE y línea de operación

[ ]:
```
pt.figure("TdeMasa Semi Lote. CURVA DE DISTRIBUCIÓN DE EQUILIBRIO CDE", [8,6]) ␣
 ↪##  Crea figura y le otorga dimensiones.

pt.plot(ConcE_Fig,CDE,'k',label='CDE')
pt.plot(ConcE_LdeO,ConcR_LdeO,'y',label='Linea de operación')
pt.plot(XA_in,YA_in,'ko',label="Punto inicial de operación")
pt.plot(XA_final,YA_final,'ro',label='Punto final de operación')
pt.plot(ConcE[1:,6],ConcR[0:,6],'b',label="Evolución del proceso")
pt.xlabel('Concentración extracto')
pt.ylabel('Concentración refinado')
pt.title('Curva de distribución de equilibrio (CDE)')
pt.grid(True)
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True,␣
 ↪fancybox=True,prop={'size': 7})
leg.get_frame().set_alpha(0.5)  ##  Agrega la leyenda en la grafica.("best" es␣
 ↪la mejor ubicación).
pt.show()
```

3.3. Simulador dinámico y estático de un proceso en cocorriente de transferencia de masa

3.3.1. Modelamiento matemático

Para este sistema se tomará como ejemplo un proceso de transferencia de masa. El proceso es realizado por un equipo en donde ingresan dos corrientes o fases por el mismo lado de la torre, realizando el mismo recorrido. La fase gas es el refinado y la fase líquida el extracto. El modelo matemático es el mismo que el caso de semilote, unicamente con un cambio. Al estar las dos corrientes recorriendo la torre en la misma dirección, los flux de ambas fases permanecen constantes en el tiempo para cada tramo de la torre, por lo que se tomará el flux de transferencia de masa únicamente en función del tramo anterior (j).

$$N_{A\,(PasoEsp,j)} = N_{A,(j)} aMPaso_{esp} \tag{22}$$

El procedimiento para el cálculo de las moles transferidas y las fracciones molares es el mismo llevado a cabo en el sistema de lote puro, pero el flux se calcula como un semi lote, y con la diferencia de que en este caso no se calcula el flux de transferencia de masa en función del tiempo sino ahora se calcula en cada tramo de la torre $Paso_{Esp}$. Una vez entendido esto, se actualizan las moles transferidas y las fracciones molares de la siguiente manera:

$$FluxMolR_{(j+1)} = FluxMolR_{(j)} - N_{A\,PasoEsp,j} \tag{23}$$

$$FluxMolE_{(j+1)} = FluxMolE_{(j)} + N_{A\,PasoEsp,j} \tag{24}$$

$$Y_{A,(j+1)} = Y_{A,(j)} - \frac{N_{A\,PasoEsp,j}}{CargaR} \tag{25}$$

$$X_{A,(j+1)} = X_{A,(j)} + \frac{N_{A\,PasoEsp,j}}{CargaE} \tag{26}$$

de esta manera, se obtienen las moles y por ende las fracciones molares para el siguiente tramo de la torre.

3.3.2. Programación del modelo de transferencia de masa en operación cocorriente en Python

A continuación se muestra la realización de los programas en ***Python 3.7*** utilizando su entorno interactivo web de ejecución de código ***Jupyter Notebook*** para así mostrar la aplicación que tiene este lenguaje de programación en la solución de este tipo de problemas.

Descripción del proceso

- El proceso consiste en transferir un soluto A de una sustancia en fase gas "refinado" a una fase líquida "extracto".
- Se tendra una torre en donde las dos fases ingresan por la parte superior de la torre realizando el mismo recorrido.
- Se debe tener en cuenta que a diferencia de lote puro y semilote, este proceso no posee ninguna fase estática en la torre.

Datos del proceso

- Fracción molar de soluto en el refinado respecto a la corriente como un todo ***y_A.***
- Fracción molar de soluto en el refinado respecto del solvente libre de soluto ***Y_A.***

- Fracción molar de soluto en el extracto respecto a la corriente como un todo ***x_A.***
- Fracción molar de soluto en el extracto respecto del solvente libre de soluto ***X_A.***
- Flux molar de la fase refinado como un todo ***FluxMolR.***
- Flux molar de la fase extracto como un todo ***FluxMolE.***
- Flux molar de la fase refinado respecto del solvente libre de soluto ***FluxR.***
- Flux molar de la fase extracto respecto del solvente libre de soluto ***FluxE.***

MÓDULO 0- Librerías de Python

- En este módulo simplemente se llaman las librerias típicas de Python. Numpy que sirve para realizar algunas operaciones con vectores y matrices, matplotlib que permite gráficar los datos obtenidos, y math que permite el uso funciones matemáticas como Logaritmos, exponenciales, seno, coseno, el número π entre otras.

[ ]:
```
import numpy as np ## Librería de python que permite realizar operaciones con␣
 ↪vectores.
import matplotlib.pyplot as pt ## Librería de python que permite realizar␣
 ↪gráficas.
import math as mt  ##  Librería de Python que permite realizar operaciones␣
 ↪matemáticas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo como: la longitud de la torre, la porosidad del empaque, la corrección de porosidad por retenido del líquido y con estas dos últimas se calcula la porosidad operativa de la torre, se define el diámetro del empaque y el área de transferencia de masa.

[ ]:
```
Longitud= 3  ##  Longitud de la torre [m].
Poros= 0.75  ##  Porosidad del empaque.
CorreHoldup= 0.0333  ##  Corrección de porosidad por retenido del líquido.
PorosOper= Poros-CorreHoldup  ##  Porosidad operativa, se resta la porosidad por␣
 ↪retenido de líquido.
DiamEmp= 0.0472  ##  Diámetro del empaque utilizado.
aM= 37.4  ##  Área de transferencia de masa [m^2/m^3].
```

Parámetros de las sustancias y constantes

- En esta sección se reunen todos los parámetros y constantes que constituyen las ecuaciones que representan el modelo matemático tales como: Masa molar de refinado y extracto, viscosidades, densidades entre otras.

[ ]:
```
MR= 11  ##  Masa molar refinado R [kg/kmol].
ME= 260  ##  Masa molar extracto E [kg/kmol].
ViscosidadR= 1e-5  ##  Viscosidad de la fase refinado [kg/m s].
ViscosidadE= 2e-3  ##  Viscosidad de la fase Extracto [kg/m s].
```

MÓDULO 2- Condiciones iniciales

- En este módulo se introducen los valores iniciales para la concentración de soluto en ambas fases, para la carga o el flux de ambas fases y con esto se calculan las fracciones molares del soluto en ambas fases respecto del solvente libre de soluto y también los flux de ambas fases puras. También se calcula la pendiente de la línea de operación.

[ ]:
```
yA_in=0.4  ##  Concentración inicial del soluto en el refinado respecto a la␣
 ↪corriente como un todo .
xA_in=0.05  ##  Concentración inicial del soluto en el extracto respecto a la␣
 ↪corriente como un todo.
yA_out=0.19  ##  Concentración final del soluto en el refinado respecto a la␣
 ↪corriente como un todo.

FluxMolR= 0.05  ##  Flux molar de entrada de la fase refinado [kmolMix/m^2 s].
FluxMolE= 0.065  ##  Flux molar de entrada de la fase extracto [kmolMix/m^2 s].

YA_in=((yA_in)/(1-yA_in))  ##  Fracción molar de entrada del soluto en el␣
 ↪refinado respecto al solvente libre de soluto.
XA_in=((xA_in)/(1-xA_in))  ##  Fracción molar de entrada del soluto en el␣
 ↪extracto respecto al solvente libre de soluto.
YA_out=((yA_out)/(1-yA_out))  ##  Fracción molar de salida del soluto en el␣
 ↪extracto respecto al solvente libre de soluto.

FluxE=FluxMolE*(1-XA_in)  ##  Flux del solvente puro en la fase extracto.
FluxR=FluxMolR*(1-YA_in)  ##  Flux del solvente puro en la fase refinado.
mLdeO=-FluxE/FluxR  ##  Pendiente de la linea de operación.
```

MÓDULO 3- Paso espacial y número de iteraciones

- En esta simulación se tienen las dos fases con flujos constantes a través de la torre, por esta razón las variables se estudiarán en función de cada tramo de la torre.
- Primero se ingresa la altura total de la torre, se define un paso espacial, y de acuerdo con este se calcula un número de pasos espaciales ***NumPasosEsp*** o número de tramos en que se divide la altura de la torre.
- Se define un vector llamado ***Longitud*** que fragmenta la longitud de la torre.

[ ]:
```
L_Inicial=0  ##  Valor de la longitud en el punto en donde inicia la simulación.
L_Final=Longitud  ##  Valor de la longitud final de la torre.
PasoEsp=0.05  ##  Valor para cada tramo en que se va a analizar la torre o paso␣
 ↪espacial.
NumPasosEsp=int((L_Final-L_Inicial)/(PasoEsp))  ##  Número de pasos espaciales.
Longitud=np.linspace(L_Inicial,L_Final,NumPasosEsp)  ##  Crea un vector de␣
 ↪longitud para poder graficarlo.
```

MÓDULO 4- Creación de vectores para guardar datos

- Este módulo construye vectores de cierta dimensión para guardar los datos de cada iteración para las variables de interés, como se puede ver principalmente se crean vectores para las concentraciones, cargas molares, la viscosidad, el coeficiente de transferencia de masa,

la concentración ficticia, el flux de transferencia de masa, la concentración deseada, y las últimas cuatro líneas de código son para poder graficar la ***CDE.***

- La librería ***numpy*** que se llama con las letras ***np*** permite crear vectores de ceros o de unos, el parámetro ***len*** en el comando, índica el número de objetos que posee un vector, ***np.zeros*** quiere decir que se va a construir un vector de ceros.
- Las dimensiones para los vectores las otorga ***len(Longitud)***, esto índica que se va a crear un vector de ceros con las dimensiones identicas a las del vector ***Longitud***. Esto último se realiza con el fin de que las dimensiones en los vectores se actualicen automaticamente si se desea cambiar el valor para la altura de la torre.
- Observese que el proceso es casi el mismo que el realizado para lote puro, únicamente que en este sistema el flux de transferencia de masa se realiza en función de cada tramo de la torre, en lote puro se realiza en función de cada instante de tiempo.
- La variable ***DiscretCDE*** y ***DiscretLdeO*** son discretizaciones de la curva de distribución de equilibrio ***CDE*** y la línea de operación ***LdeO***, esto se realiza con el fin de que al cambiar la longitud de la torre, estas dos variables se puedan graficar también respecto al número de tramos totales.

[ ]:
```
ConcR=np.zeros(len(Longitud)+1)  ##  Crea un vector de datos para la␣
 ↪concentración del refinado.
ConcE=np.zeros(len(Longitud)+1)  ##  Crea un vector de datos para la␣
 ↪concentración del extracto.
MolesR=np.zeros(len(Longitud)+1) ##  Crea un vector de datos para las moles de␣
 ↪refinado.
MolesE=np.zeros(len(Longitud)+1)  ##  Crea un vector de datos para las moles de␣
 ↪extracto.

ViscR=np.zeros(len(Longitud))  ##  Crea un vector de datos para la viscosidad.
KY=np.zeros(len(Longitud))  ##  Crea un vector de datos para el coeficiente de␣
 ↪TdeMasa.
ConcFict=np.zeros(len(Longitud))  ##  Crea un vector de datos para la␣
 ↪concentración ficticia.
NA=np.zeros(len(Longitud))  ##  Crea un vector de datos para el flux de TdeMasa.
NA_tramo=np.zeros(len(Longitud))  ##  Crea un vector de datos para el flux de␣
 ↪TdeMasa en cada partición de longitud.
ConcDeseada=np.zeros(len(Longitud))  ##  Crea un vector de datos para la␣
 ↪concentración deseada (fines graficos).

ConcE_Fig=np.zeros(len(Longitud))  ##  Crea un vector de datos para la␣
 ↪concentración del extracto en la CDE.
CDE=np.zeros(len(Longitud))  ##  Crea un vector de datos para cada punto de la␣
 ↪CDE.
ConcR_LdeO=np.zeros(len(Longitud))  ##  Crea un vector de datos para la␣
 ↪concentración del refinado en la linea de operación.
ConcE_LdeO=np.zeros(len(Longitud))  ##  Crea un vector de datos para la␣
 ↪concentración de extracto en la linea de operación.
```

```
DiscretCDE=YA_in/NumPasosEsp   ##  Discretización de la fracción molar de soluto␣
 ↪en el refinado a la entrada para graficar la CDE.
DiscretLdeO=YA_out/NumPasosEsp  ##  Discretización de la fracción molar de␣
 ↪soluto en el refinado a la salida para graficar la linea de operación.
```

MÓDULO 5- Programación de ecuaciones

Solución de ecuaciones constitutivas

- En esta sección se comienza la solución iterativa del método. Primeramente se le debe ingresar a la primer posición de los vectores creados los valores correspondientes a las condiciones iniciales, tanto para las concentraciones como para los flux molares en ambas fases, ya que estas son las variables de interes en la simulación.
- Como se puede observar, en esta sección se introducen todas las ecuaciones del modelo matemático que dependen de las concentraciones. La importancia de ingresar las ecuaciones en este modulo es que estas se actualizan en cada iteración o partición temporal, cosa que no pasaría si por ejemplo la viscosidad se considerara constante, además esto puede influir significativamente en los resultados de la simulación.
- También se ingresa el polinomio que representa las concentraciones ficticias, ya que con esta se calcula el flux de transferencia de masa.
- Se puede observar que el flux de transferencia de masa se actualiza dos veces ***NA*** y ***NA_deltaL***, esto debido a que este debe actualizarse en cada tramo de la torre, como se evidencia al tratarse de una operación en cocorriente el flujo molar es constante en ambas fases, por ende la dimensión temporal no se tiene en cuenta en este modelo.
- Por último, se tienen las líneas que actualizan las cargas molares y las concentraciones tanto para el refinado ***R*** como para el extracto ***E***. Como se puede ver estas variables se actualizan en el tramo de la torre siguiente (j+1) dependiendo del valor y el cambio que estas en el tramo de la torre anterior (j).

[ ]:
```
ConcR[0]=YA_in  ##  Otorga un valor inicial a la concentración de refinado para␣
 ↪resolver el método iterativo.
ConcE[0]=XA_in  ##  Otorga un valor inicial a la concentración de extracto para␣
 ↪resolver el método iterativo.
MolesR[0]=FluxMolR  ##  Otorga un valor inicial a las moles de refinado para␣
 ↪resolver el método iterativo.
MolesE[0]=FluxMolE  ##  Otorga un valor inicial a las moles de extracto para␣
 ↪resolver el método iterativo.

for j in range(0,NumPasosEsp,1):


    ViscR[j]=ViscosidadR+(ConcR[j]/100)**2  ##  Recalcula la viscosidad del␣
 ↪refinado.
    KY[j]=MolesR[j]*0.15*((DiamEmp*MolesR[j]*MR)/(ViscR[j]*(1-PorosOper)))**-0.
 ↪36  ##  Actualiza el coeficiente englobante de TdeMasa [kmol/m^2 s].
```

```
   ConcFict[j]=(9.524*ConcE[j]**3) -(7.302*ConcE[j]**2)+(2.064*ConcE[j])  ## Polinomio que representa la concentración ficticia.

   NA[j]=KY[j]*(ConcR[j]-ConcFict[j])  ##  Recalcula el flux de TdeMasa [kmol/m^2 s].
   NA_tramo[j]=NA[j]*aM*PasoEsp  ##  Recalcula el flux de TdeMasa en cada partición de longitud [kmol/m^2 s].


   MolesR[j+1]=MolesR[j]-NA_tramo[j]  ##  Recalcula las moles en el  refinado [Kmol?m^2 s].
   MolesE[j+1]=MolesE[j]+NA_tramo[j]  ##  Recalcula las moles en el extracto [kmol?m^2 s].

   ConcR[j+1]=ConcR[j]-((NA_tramo[j])/(FluxR))  ##  Recalcula la concentración del refinado.
   ConcE[j+1]=ConcE[j]+((NA_tramo[j])/(FluxE))  ##  Recalcula la concentración del extracto.
```

Solución de ecuaciones para la CDE

- En esta sección se construye la curva de distribución de equilibrio ***CDE*** y la línea de operación.
- La primer línea para ***ConcE_Fig*** únicamente construye punto a punto la concentración del extracto para graficarlo en la CDE.
- La segunda línea para ***CDE*** no es más que un polinomio que ajusta los datos experimentales para la CDE y poder de esta manera graficar la curva en función de la concentracion del extracto.
- Las siguientes líneas para ***ConcE_LdeO*** y ***ConcR_LdeO*** actualizan y construyen punto a punto las concentraciones en las dos fases para poder construir la línea de operación.
- Las expresiones para ***XA_final*** y ***YA_final*** son únicamente para graficar un punto en la CDE que represente las concentraciones finales para las dos fases.
- Para la ConcDeseada únicamente se llena cada iteración con el mismo valor, el valor deseado en la fase refinado. Esta acción se realiza para graficar una línea recta en la CDE para reconocer la concentración deseada.
- Las últimas líneas de código se utilizan para ajustar los vectores a las dimensiones correspondientes.

[ ]:
```
   ConcE_Fig[j]=j*PartCDE  ##  Construye punto a punto la concentracion del extracto para graficarlo en la CDE.
   CDE[j]=9.524*ConcE_Fig[j]**3 -7.302*ConcE_Fig[j]**2+2.064*ConcE_Fig[j]  ## Polinomio que representa la CDE.
```

```
    ConcE_LdeO[j]=ConcE[0]+j*PartLdeO  ##  Aumenta a la concentración del␣
 ↪extracto el valor de cada partición para graficarlo.
    ConcR_LdeO[j]=ConcR[0]+mLdeO*(ConcE_LdeO[j]-ConcE[0])  ##  Cálcula y␣
 ↪actualiza la concentración del refinado en la CDE en función de la␣
 ↪concentración del extracto.

    XA_final=ConcE[i+1]  ##  Otorga el valor a concentración final de A en el␣
 ↪extracto para graficar dicho punto.
    YA_final=ConcR[0]+mLdeO*(XA_final-ConcE[0])  ##  Cálcula la concentración␣
 ↪final de A en el refinado para graficar dicho punto.

    ConcDeseada[j]=YA_out  ##  Llena el vector de concentración deseada con los␣
 ↪datos de concentración de salida que se desea para poder graficar la linea␣
 ↪recta.

ConcR=ConcR[:j+1]  ##  Ajusta el vector a las dimensiones requeridas.
ConcE=ConcE[:j+1]  ##  Ajusta el vector a las dimensiones requeridas.
MolesR=MolesR[:j+1]  ##  Ajusta el vector a las dimensiones requeridas.
MolesE=MolesE[:j+1]  ##  Ajusta el vector a las dimensiones requeridas.
```

MÓDULO 6- Graficas y resultados

- En este módulo se utiliza la librería matplotlib que permite gráficar los resultados obtenidos.
- Las líneas de código son simplemente acciones y caracteristicas para poder visualizar los datos.

Gráfica: Concentraciones de refinado y extracto vs longitud de la torre

[ ]:
```
pt.figure("TdeMasa Cocorriente", [10,5])  ##  Crea figura y le otorga␣
 ↪dimensiones.
pt.subplot(1,2,1)   ##  Hace una sub-figura (N°filas , N°columnas , Posicion de␣
 ↪la figura en el subplot).
pt.tight_layout(pad=5, w_pad=5, h_pad=4)  ##  Espaciado entre las figuras del␣
 ↪subplot (Espaciado entre la margen , Espaciado entre figuras (Horizontal) ,␣
 ↪Espaciado entre figuras (Vertical).
pt.plot(Longitud,ConcR,'k',linewidth=2,label='Concentración en el Refinado')  ##␣
 ↪ Gráfica: Longitud vs Concentración del refinado.
pt.plot(Longitud,ConcE,'b',linewidth=2,label='Concentración en el Extracto')  ##␣
 ↪ Gráfica: Longitud vs Concentración del extracto.
pt.plot(Longitud,ConcFict,'y',linewidth=2,label='Concentración ficticia')  ## ␣
 ↪Grafica el longitud vs Concentración ficticia.
pt.plot(Longitud,ConcDeseada,'r',linewidth=2)  ##  Grafica el longitud vs␣
 ↪Concentración deseada.
pt.xlabel("Longitud [m]")  ##  Agrega título al eje x.
pt.ylabel('Concentración respecto\n del solvente libre de soluto ')  ##  Agrega␣
 ↪título al eje y.
pt.title('Concentración vs longitud')  ##  Agrega título en la parte superior.
```

```
pt.grid(True)  ##  Agrega la cuadrilla a la imagen
pt.tick_params(labelsize=8)  ##  Ajusta el tamaño de los títulos para los ejes
 ↪(x , y), y de igual manera ajusta el tamaño de los números en cuadrilla(Grid).
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,
 ↪prop={'size': 7})
leg.get_frame().set_alpha(0.5)  ##  Agrega la leyenda en la grafica.("best" es
 ↪la mejor ubicación).
pt.show()
```

Gráfica: Curva de distribución de equilibrio CDE y línea de operación

[ ]:
```
pt.subplot(1,2,2)
pt.grid(True)
pt.plot(ConcE_Fig,CDE,'k',label='CDE')
pt.plot(ConcE_LdeO,ConcR_LdeO,'y',label='Linea de operación guia (imaginaria)')
pt.plot(XA_in,YA_in,'ko',label="Punto inicial de operación")
pt.plot(XA_final,YA_final,'ro',label='Punto final de operación')
pt.plot(ConcE,ConcR,'b')
pt.xlabel('Concentración extracto')
pt.ylabel('Concentración refinado')
pt.title('Curva de distribución de equilibrio (CDE)')
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,
 ↪prop={'size': 7})
leg.get_frame().set_alpha(0.5)
pt.show()
```

3.4. Simulador dinámico y estático de un proceso en contracorriente de transferencia de masa

A continuación se desarrolla un ejemplo tomado de [1], este dice de la siguiente manera: Se toma como equipo una torre de extracción líquido-líquido operando a contracorriente. Se resalta en este caso, para resolver el modelo dinámico se resuelve primero el modelo estacionario, este último servira de ejemplo para el modelo computacional, ya que el modelo dinámico es un poco mas complejo. Dicha solución se hace para comprobar la factibilidad del punto de operación del equipo en términos de separación. El algoritmo de solución del modelo estacionario, que no es trivial, fue elaborado por Vanesa Inés Arenas A., María Paulina Salcedo C. y Santiago Agudelo R, como parte de su trabajo dirigido de grado en Ingeniería Química, Unal Medellín, semestre 02 de 2016.

3.4.1. Modelamiento matamático

Primero que todo se determina el balance de masa total para los sistemas de proceso, en este caso para el sistema de proceso (refinado) y se encuentra en unidades molares. La forma de nomenclar las fases es: X_A para la fase refinado y Y_A para la fase extracto.

$$\frac{dN_{R,j}}{dt} = \dot{n}_{R,(j-1)} - A_{M,j}N_{A,j} - \dot{n}_{R,j} \tag{27}$$

Puesto que los valores de las concentraciones de nicotina (Soluto A) de operación para las corrientes de entrada y esperadas para las corrientes de salida, son bajas, se puede asumir que las moles totales dentro de cada SdeP en una rodaja no cambian apreciablemente, por lo que $\frac{dN_{R,(j)}}{dt} = 0$. Además, como se conoce la entrada pero $A_{M,(j)}$ y $N_{A,(j)}$ se calculan por ecuaciones constitutivas (son parámetros), la única incógnita de la ecuación del balance total es el flujo molar de salida. Por lo tanto, desdejándolo, se llega a la expresion final para el balance total.

$$\dot{n}_{R,(j)} = \dot{n}_{R,(j-1)} - A_{A,(j)} N_{A,(j)} \tag{28}$$

Ahora se realiza un balance de masa por componente, en este caso para el soluto A (Nicotina).

$$\frac{dN_{A,(j)}}{dt} = X_{A,(j-1)} \dot{n}_{RS,(j-1)} - A_{M,(j)} N_{A,(j)} - X_{A,(j)} \dot{n}_{RS,(j)} \tag{29}$$

con $N_{A,(i)}$ las moles de soluto en el sistema de proceso (refinado) en la posición (j) y X_A la fracción molar del soluto A en la fase libre de soluto o fracción molar de A con base en el solvente puro, en la corriente de refinado que indica el segundo subíndice. $\dot{n}_{RS,(j-1)}$ es el flujo de solvente del refinado (agua) en la corriente que indica el segundo subíndice. Se debe usar el flujo molar del solvente del refinado y no el del refinado completo puesto que las concentraciones, como se acaba de decir, están en la fracción libre de soluto o con base en el solvente puro. En la ecuacion anterior es posible espresar las moles totales de nicotina dn el sistema de proceso (refinado) posición (j) como a continuación, usando el criterio de agitación perfecta del sistema de proceso:

$$\frac{dN_{A,(j)}}{dt} = X_{A,(j)} N_{RS,(j)} \tag{30}$$

desde la que aplicando la definición de derivada de un producto se obtiene:

$$\frac{dN_{A,(j)}}{dt} = X_{A,(j)} \frac{N_{RS,(j)}}{dt} + N_{RS,(j)} \frac{X_{A,(j)}}{dt} \tag{31}$$

Volviendo a aplicar el supuesto de moles constantes dentro del sistema de proceso debido a las bajas concentraciones de soluto, se puede tomar en este caso que $\frac{N_{RS,(j)}}{dt} = 0$ y por lo tanto:

$$\frac{dN_{A,(j)}}{dt} = N_{RS,(j)} \frac{X_{A,(j)}}{dt} \tag{32}$$

la que reemplazada en la ecuación del balance de masa por componente y despejando el diferencial, brinda la expresión final del balance de masa por componente para el sistema de proceso (refinado) en la posición (j):

$$\frac{X_{A,(j)}}{dt} = \frac{1}{N_{RS,(j)}} (X_{A,(j-1)} \dot{n}_{RS,(j-1)} - A_{M,(j)} N_{A,(j)} - X_{A,(j)} \dot{n}_{RS,(j)}) \tag{33}$$

Ahora para el sistema de proceso (extracto) que es la solución de nicotina en tolueno. La deducción de todos los balances es la misma que para el refinado, por eso posee el signo (+), ya que en el sistema de proceso refinado se tenia un signo (-). Realizando los mismo balances del sistema de proceso (refinado) para el sistema de proceso (extracto) se tienen las siguientes expresiones:

$$\dot{n}_{E,(j)} = \dot{n}_{E,(j+1)} - A_{A,(j)} N_{A,(j)} \tag{34}$$

$$\frac{Y_{A,(j)}}{dt} = \frac{1}{N_{RS,(j)}} (Y_{A,(j+1)} \dot{n}_{ES,(j+1)} + A_{M,(j)} N_{A,(j)} - Y_{A,(j)} \dot{n}_{ES,(j)}) \tag{35}$$

Las expresiones anteriores, determinan la estructura básica del modelo. Luego se realiza un reconocimiento de las variables y constantes de cada una de las ecuaciones que componene el modelo, esto se verá detenidamente en la programación del simulador.

Las ecuaciones constitutivas que componen este modelo son:

Para el área de transferencia de masa.

$$A_{M,(j)} = Num_{gotas} A_{sup,gotas} \tag{36}$$

Número de gotas:

$$Num_{gotas} = \frac{Frac_{Vol,E} Vol_{Total}}{Vol_{gota}} \tag{37}$$

Fracción volumétrica del extracto:

$$Frac_{Vol,E} = \frac{\dot{V}_E}{\dot{V}_E + \dot{V}_R} \tag{38}$$

Flujo volumétrico para la fase refinado y extracto respectivamente:

$$\dot{V}_R = \frac{\dot{n}_{R,(j-1)} MR}{\rho R} \tag{39}$$

$$\dot{V}_E = \frac{\dot{n}_{E,(j+1)} ME}{\rho E} \tag{40}$$

Volumen total:

$$V_{Total,(j)} = \Delta L \pi \frac{D^2_{Torre}}{4} \tag{41}$$

El fragmento de la torre a analizar:

$$\Delta L = \frac{L_{Torre}}{Nmero_{Particiones}} \tag{42}$$

Volumen de una gota:

$$Vol_{gota} = \frac{\pi D^3_{Gota}}{6} \tag{43}$$

Área superficial de una gota:

$$A_{sup,gota} = \pi D^2_{Gota} \tag{44}$$

Flux de transferencia de masa:

$$N_{A,(j)} = k_{X,(j)}(X_{A,(j)} - X_{A,equi,(j)}) \tag{45}$$

Coeficiente de masa en la fase continua, se usa la expresión ***A.H.P***. Skelland reportada en el manual del ingeniero químico (Green & Perry,2008):

$$k_{X,(j)} = \frac{\mathfrak{D}_{A-T}}{D_{Gota}}[2 + 0{,}463 N_{Re,Gota,(j)}^{0,484} N_{Sc,R,(j)}^{0,339} (\frac{D_{Gota} g^{1/3}}{\mathfrak{D}_{A-Agua}^{2/3}})] F_j \tag{46}$$

Donde F_j es un parámetro de corrección por forma esférica asumida:

$$F_j = 0{,}281 + 1{,}615 K_{Ske,(i)} + 3{,}73 K_{Ske,(i)}^2 - 1{,}874 K_{Ske,(i)}^3 \tag{47}$$

se define la correlación Ske, propuesto por el autor:

$$K_{Ske,(j)} = N_{Re,Gota,(j)}^{1/8} (\frac{\mu_R}{\mu_E})^{1/4} (\frac{\mu_R v_s}{\sigma_R})^{1/6} \tag{48}$$

donde las viscosidades μ son constantes y se define la velocidad de deslizamiento, definida para operación en contracorriente como la suma de los valores absolutos de la velocidad de la fase continua R y la fase dispersa E.

$$v_s = abs(v_R) + abs(v_E) \tag{49}$$

de la que aparecen las velocidades de las dos fases como dos nuevos parámetros funcionales. La velocidad de la fase continua R se calcula como la de un fluido que fluye por un tubo de diámetro conocido, con flujo volumétrico conocido, pero compensada por la presencia del otro fluido (las gotas), usando una corrección del área de flujo con la fracción volumétrica $Frac_{Vol,E}$ de la fase dispersa E en la torre:

$$v_R = (1 - Frac_{Vol,E}) \frac{\dot{V}_R}{A_{F,T}} \tag{50}$$

siendo el área de flujo $A_{F,T}$ un parámetro funcional difinido como:

$$A_{F,T} = \pi \frac{D_{Torre}^2}{4} \tag{51}$$

Por otra parte se define la velocidad de la fase dispersa:

$$v_E = Frac_{Vol,E} \frac{\dot{V}_E}{A_{F,T}} \tag{52}$$

también se define el número de Reynolds de una gota, ya que es un parámetro funcional.

$$N_{Re,Gota,(j)} = \frac{D_{Gota} v_s \rho_{R,(j)}}{\mu_R} \tag{53}$$

y el número de schmidt de la fse continua R:

$$N_{Sc,R,(j)} = \frac{\mu_R}{\rho_{R,(j)}\mathfrak{D}_{A-T}} \tag{54}$$

Como siguiente paso se debe hallar la concentración de equilibrio $X_{A,equ}$ o concentración de interfase. Para esto se debe usar la ecuación de la línea de fuerza impulsora (FI), que conecta al punto de operación actual (Subíndice op) con la curva de distribución de equilibrio o (CDE) o punto de equilibrio (subíndice eq):

$$Y_{A,op,(j)} = -\frac{k_{X,(j)}}{k_{Y,(j)}}(X_{A,op,(j)} - X_{A,eq,(j)}) + Y_{A,eq,(j)} \tag{55}$$

Por otra parte, se requiere la ecuación de la CDE, para lo que se obtienen datos de las concentraciones de equilibrio desde el diagrama ternario para Agua-Nicotina-Tolueno, no mostrado aqui. De esta manera se obtiene la siguiente expresión para la CDE:

$$Y_{A,eq,(j)} = 16{,}07781726 X_{A,eq,(j)} + \frac{0{,}08593565238 X_{A,eq,(j)}}{X_{A,eq,(j)} + 0{,}0003360234205} \tag{56}$$

ahora se sabe que justamente donde se cortan la recta de la línea de fuerza impulsora (FI) y la (CDE), se encuentra el punto de equilibrio actual, la cual contiene la coordenada $X_{A,eq,(j)}$ buscada para la partición (j). Por lo tanto, se reemplaza la ecuación de la (CDE) en la ecuación de la línea de fuerza impulsora se tiene:

$$-Y_{A,op,(j)} - \frac{k_{X,(j)}}{k_{Y,(j)}}(X_{A,op,(j)} - X_{A,eq,(j)}) + 16{,}07781726 X_{A,eq,(j)} + \frac{0{,}08593565238 X_{A,eq,(j)}}{X_{A,eq,(j)} + 0{,}0003360234205} \tag{57}$$

solucionando la ecuación anterior que es no líneal, se pueden encontrar los puntos de equilibrio y asi poder calcular el flux de transferencia de masa, de esta manera se hallan los puntos en equilibrio o estado estacionario para cada rodaja de la torre.

3.4.2. Programación del modelo de transferencia de masa en operación contracorriente en Python

A continuación se muestra la realización de los programas en ***Python 3.7*** utilizando su entorno interactivo web de ejecución de código ***Jupyter Notebook*** para así mostrar la aplicación que tiene este lenguaje de programación en la solución de este tipo de problemas.

Descripción del proceso

- El proceso consiste en un proceso de extracción líquido-líquido, en este caso un soluto A de una fase líquida o refinado (Agua) a otra fase líquida o extracto (Tolueno).
- Se debe tener en cuenta que en este proceso las fases ingresan por lados contrarios de la torre.

Datos del proceso

- Fracción molar de soluto en el refinado respecto a la corriente como un todo ***y_A.***
- Fracción molar de soluto en el refinado respecto del solvente libre de soluto ***Y_A.***
- Fracción molar de soluto en el extracto respecto a la corriente como un todo ***x_A.***
- Fracción molar de soluto en el extracto respecto del solvente libre de soluto ***X_A.***
- Flujo molar de la fase refinado (Agua) ***FlujoMolR.***
- Flujo molar de la fase extracto (Tolueno) ***FlujoMolE.***

MÓDULO 0- Librerías de Python

- En este módulo simplemente se llaman las librerías típicas de Python. Numpy que sirve para realizar algunas operaciones con vectores y matrices, matplotlib que permite gráficar los datos obtenidos, y math que permite el uso funciones matemáticas como Logaritmos, exponenciales, seno, coseno, el número π entre otras.

[ ]:
```
import numpy as np ## Librería de python que permite realizar operaciones con␣
 ↪vectores.
import matplotlib.pyplot as pt ## Librería de python que permite realizar␣
 ↪gráficas.
import math as mt  ##  Librería de Python que permite realizar operaciones␣
 ↪matemáticas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo como: Longitud de la torre, diámetro de la torre, el área de flujo y el valor de volumen para cada partición.

[ ]:
```
Longitud= 3.1 ##  Longitud de la torre [m].
DiametroTorre= 1  ##  Diámetro de la torre [m].
AreaF=mt.pi*(DiametroTorre/2)**2  ##  Área de flujo total de la torre [m^2].
```

Parámetros de las sustancias y constantes

- En esta sección se reunen todos los parámetros y constantes que constituyen las ecuaciones que representan el modelo matemático.

[ ]:
```
g= 9.8  ## Constante gravitacional [m/s^2].
MMT= 92.138  ##  Masa molar del Toluneo [kmol/kg].
MMA= 18  ##  Masa molar del agua [kmol/kg].
MMN= 162.232  ##  Masa molar de la nicotina [kmol/kg].
Ro_Tolueno= 857.7  ## Densidad del tolueno puro [kg/m^3].
Ro_Agua= 995.7  ## Densidad del agua pura [kg/m^3].
Ro_Nicot= 1010  ## Densidad de la nicotina pura [kg/m^3].
TenSuperTol= 0.0291  ##  Tensión superficial de la fase dispersa TOLUENO [N/m].
ViscTol= 5.24825e-4  ##  Viscocidad del tolueno puro [Pa-s].
ViscAgua= 8e-4  ##  Viscocidad del agua pura [Pa-s].
ViscNicot= 3.442e-3  ##  Viscocidad de la nicotina pura [Pa-s].
Dif_NicAgua= 6.8753e-10  ##  Difusividad de la nicotina en agua [m^2/s].
Dif_NicTol= 1.62033e-9  ##  Difusividad de la nicotina en Tolueno [m^2/s].
```

```
DiamGota= 0.001  ##  Diametro de cada gota [m].
AreaGota= mt.pi*DiamGota**2  ##  Área de una sola gota de la fase dispersa [m^2].
VolGota=((mt.pi*(DiamGota**3))/(6))  ##  Volumen de cada gota [m^3].
```

MÓDULO 2- Condiciones iniciales

- En este módulo se introducen los valores iniciales para la concentración de soluto en ambas fases, para la carga o el flux de ambas fases y con esto se calculan las fracciones molares del soluto en ambas fases respecto del solvente libre de soluto y también los flux de ambas fases puras.

[ ]:
```
FlujoMolR= 0.2  ##  Flujo molar de la fase refinado [kmol/s].
FlujoMolE= 0.0158  ##  Flujo molar de la fase extracto [kmol/s].

YA_in=  0.005  ##  Fracción molar de entrada del soluto en el extracto respecto␣
 ↪al solvente libre de soluto [kmol Nicotina/ kmol Extracto].
YA_out=0.1  ##  Fracción molar de salida de soluto para el extracto.
XA_in= 0.0087  ##  Fracción molar de entrada del soluto en el refinado respecto␣
 ↪al solvente libre de soluto [kmol Nicotina/ kmol Refinado].
XA_out= 0.0012  ##  Fracción molar de salida del soluto en el refinado respecto␣
 ↪al solvente libre de soluto a la salida de la torre [kmol Nicotina/ kmol␣
 ↪Refinado].
```

MÓDULO 3- Paso espacial y número de iteraciones

- En esta simulación se tienen las dos fases con flujos constantes a través de la torre, por esta razón las variables se estudiarán en función de cada tramo de la torre.
- Primero se ingresa la altura total de la torre, se define un paso espacial, y de acuerdo con este se calcula un número de pasos espaciales ***NumPasosEsp*** o número de tramos en que se divide la altura de la torre.
- Se define un vector llamado ***Long*** que fragmenta la longitud de la torre.
- El término ***DiscCDE*** parte la concentración final en la CDE en el número de pasos espaciales, esto con el fin de que al cambiar la longitud de la torre o el valor del paso espacial, la CDE pueda graficarse sin ningun problema.
- El termino ***VolPart*** es el volumen que tendra cada una de las particiones en la torre.
- ***Iter*** y ***Tolerancia*** son parámetros que se utilizan para darle convergencia al método que calcula las concentraciones en equilibrio. (Mas adelante se aclara mejor).

[ ]:
```
Long_Inicial= 0  ##  Longitud inicial de la simulación [s].
Long_Final = Longitud  ##  Longitud final de la torre [m].
PasoEsp = 0.05 ## Longitud para cada tramo de la torre [m].
NumPasosEsp=int((Long_Final-Long_Inicial)/(PasoEsp))  ##  Determina el número de␣
 ↪particiones necesarias para resolver el método.
Long = np.linspace(Long_Inicial, Long_Final, NumPasosEsp)  ##  Crea vector de␣
 ↪longitud para poder graficarlo [s].
DiscCDE=0.012/NumPasosEsp  ##  Discretización de la fracción molar de soluto en␣
 ↪el refinado a la entrada para graficar la CDE.
VolPart= AreaF*PasoEsp  ##  Valor de cada partición de volumen [m^3].
```

```
Iter= 100  ##  Número de iteraciones para lograr convergencia.
Tolerancia=1e-8  ##  Tolerancia aceptada para las iteraciones.
```

MÓDULO 4- Creación de vectores para guardar datos

- Este módulo construye vectores de cierta dimensión para guardar los datos de cada iteración para las variables de interes, como se puede ver principalmente se crean vectores para las concentraciones, cargas molares, la viscosidad, el coeficiente de transferencia de masa, la concentración ficticia, el flux de transferencia de masa, la concentración deseada, y las últimas cuatro líneas de código son para poder graficar la ***CDE.***
- La libreria ***numpy*** que se llama con las letras ***np*** permite crear vectores de zeros o de unos, la palabra ***len*** indica el número de objetos que posee un vector, ***np.zeros*** quiere decir que se va a construir un vector de ceros.
- Las dimensiones para los vectores las otorga ***len(Longitud)***, esto indica que se va a crear un vector de ceros con las dimensiones identicas a las del vector ***Longitud***. Esto ultimo se realiza con el fin de que las dimensiones en los vectores se actualicen automaticamente si se desea cambiar el valor para la altura de la torre.
- Observese que el proceso es casi el mismo que el realizado para lote puro, unicamente que en este sistema el flux de transferencia de masa se realiza en función de cada tramo de la torre, en lote puro se realiza en función de cada instante de tiempo.

[ ]:
```
XA_LdeO= np.zeros(len(Long)+1)  ##  Crea un vector de datos para la␣
 ↪concentración del soluto A en el refinado para la LdeO.
YA_LdeO=np.zeros(len(Long)+1)  ##  Crea un vector de datos para la concentración␣
 ↪del soluto A en el extracto para la LdeO.

XA=np.zeros(len(Long)+1) ##  Crea un vector de datos para la concentración del␣
 ↪soluto A en el refinado para las concentraciones en equilibrio.
YA=np.zeros(len(Long)+1)  ##  Crea un vector de datos para la concentración del␣
 ↪soluto A en el extracto para las concentraciones en equilibrio.

XA_Sup=np.zeros(len(Long))  ##  Crea un vector de datos para la concentración␣
 ↪del soluto A para en el refinado, esta es la supuesta para poder hallar la␣
 ↪verdadera concentración en el equilibrio.
XA_Part=np.zeros(len(Long)+1)  ## ##  Crea un vector de datos para la partición␣
 ↪de la concentración del soluto A en el refinado.


AM=np.zeros(len(Long))  ## Crea un vector de datos para calcular el área de␣
 ↪transferencia de masa.
NA_EE=np.zeros(len(Long))  ## Crea un vector de datos para calcular el flux de␣
 ↪transferencia de masa.

XA_Equ=np.zeros(len(Long))  ##  Crea un vector de datos para el calculo de la␣
 ↪concentración del soluto A (refinado) en equilibrio o interfaz.
YA_Equ=np.zeros(len(Long))  ##  Crea un vector de datos para el calculo de la␣
 ↪concentración del soluto A (extracto) en equilibrio o interfaz.
```

```
XA_CDE=np.zeros(len(Long))  ##  Crea un vector de datos para la concentración␣
 ↪del soluto A en la fase refinado para graficar la CDE.
YA_CDE=np.zeros(len(Long))  ##  Crea un vector de datos para la concentración␣
 ↪del soluto A en la fase extracto para graficar la CDE.
```

MÓDULO 5- Programación de ecuaciones

Solución de ecuaciones constitutivas

- En esta sección se comienza la solución iterativa del método. Primeramente se le debe ingresar a la primer posición de los vectores creados los valores correspondientes a las concentraciones iniciales y de salida únicamente para la fase refinado, las concentraciones de entrada para la LdeO, la pendiente de la LdeO y se calcula el intervalo de concentraciones para la fase refinado.
- Posteriormente, el ciclo for actualiza las variables mencionadas anteriormente.
- Se incluye un error y un contador para poder realizar la iteración con el ciclo while.

[ ]:
```
## AQUÍ EMPIEZA LA SOLUCIÓN ITERATIVA.##

XA[0]=XA_in  ##  Otorga un valor inicial en la posición cero del vector para la␣
 ↪concentracion de A en la fase refinado.
YA[0]=YA_out  ##  Otorga un valor inicial en la posición cero del vector para la␣
 ↪concentracion de A en la fase extracto.
XA[NumPasosEsp]=XA_out  ##  Otorga el valor de salida o final en la última␣
 ↪partición de la torre para la fase refinado.

XA_LdeO[0]= XA_in  ##  Otorga el valor inicial a la concentración de refinado␣
 ↪para iniciar con la construcción de la linea de operación.
YA_LdeO[0]= YA[0] ##  Otorga el valor inicial a la concentración de extracto␣
 ↪para iniciar con la construcción de la linea de operación.

mLdeO=((YA_in-YA[0])/(XA[NumPasosEsp]-XA_in))  ##  Cálcula la pendiente de la␣
 ↪linea de operación en funcion de la variación de concentraciónes del extracto␣
 ↪y refinado.

XA_IntTotal= XA_in-XA[NumPasosEsp]  ##  Intervalo de concentración del refinado␣
 ↪desde la entrada hasta la salida con concentración deseada [kmol Nicotina/␣
 ↪kmol Refinado].

for j in range(0,NumPasosEsp,1):

    XA_LdeO[j+1]=XA_LdeO[j]-(XA_IntTotal/NumPasosEsp)  ##  Otorga el siguiente␣
 ↪valor a la concentración en el refinado para construir la linea de operación.
    YA_LdeO[j+1]=mLdeO*(XA_LdeO[j+1]-XA[NumPasosEsp])+YA_in  ##  Otorga el␣
 ↪siguiente valor a la concentración en el extracto para construir la linea de␣
 ↪operación.
```

```
    XA_Sup[j]=XA_in-j*(XA_IntTotal/NumPasosEsp)  ##  Se dan valores semilla para␣
↪la concentración de A en la fase refinado.

    if j==0:  ##  Condiciona la concentración de cada partición para que la␣
↪siguiente comience con la salida de la última.
        XA_Part[j]=XA_in
    else:

        XA_Part[j]=XA[j-1]

    if j==0:
        YA[j]=YA[0]


    ErrorEst= 2*Tolerancia  ## Se define un error estimado.

    Contador=0  ##  Contador para ingresar al ciclo while.
```

CICLO WHILE

- En este ciclo se solucionan y actualizan todas las ecuaciones del modelo, ecuaciones como: fracciones molares, densidades, masas molares,viscosidades, flujos, fracciones volumétricas, velocidades, número de Reynolds para la fase dispersa, número de Schmidt, número de Skelland, los coeficientes locales de TdeMasa y la pendiente de la fuerza impulsora.
- Posteriormente se define la ecuación que representa la CDE, y a esta misma se le realiza el método de Newton para encontrar las raices de la ecuación no líneal, recordando que estas raices representan cada valor en la interfaz o en equilibrio de la fase refinado.
- Con el valor de equlibrio para la fase refinado, se puede calcular la concentración para la fase extracto, el número de gotas, el flux de transferencia de masa y el área de TdeMasa.
- Con los parámetros calculados en el paso anterior, se procede a determinar las concentraciones en ambas fases en operación, esto para poder graficar la línea de operación (LdeO).
- En caso de que el valor semilla y el calculado no tengan la tolerancia aceptable, se reemplaza el valor semilla por el calculado e ingresa de nuevo al ciclo. Este proceso se repite en cada partición de la torre.

[ ]:
```
while ErrorEst>Tolerancia or  Contador<Iter:



        xA_Sup=((XA_Sup[j])/(1+XA_Sup[j]))  ##  Fracción molar respecto de la␣
↪fase como un todo.
        yA=((YA[j])/(1+YA[j]))

        XwA_Sup=XA_Sup[j]*(MMN/MMA)  ##  Conversión a fracción molar respecto␣
↪del solvente puro.
```

```
        YwA_Sup=YA[j]*(MMN/MMT)

        xwA_Sup=((XwA_Sup)/(1+XwA_Sup))  ##  Conversión a fracción másica␣
↪respecto de la fase como un todo.
        ywA_Sup=((YwA_Sup)/(1+YwA_Sup))


        Rho_R=1/((xwA_Sup/Ro_Nicot)+((1-xwA_Sup)/(Ro_Agua)))  ##  Cálculo y␣
↪actualizacion de las densidades.
        Rho_E=1/((ywA_Sup/Ro_Nicot)+((1-ywA_Sup)/(Ro_Tolueno)))

        MMR_Mezcla=MMA*(1-xA_Sup)+MMN*xA_Sup ##  Cálculo y actualizacion de las␣
↪masas molares.
        MME_Mezcla=MMT*(1-yA)+MMN*yA

        ViscR= np.exp(xwA_Sup*np.log(ViscNicot)+(1-xwA_Sup)*np.log(ViscAgua)) ␣
↪##  Cálcuo y actualización del las viscosidades.
        ViscE= np.exp(ywA_Sup*np.log(ViscNicot)+(1-ywA_Sup)*np.log(ViscTol))

        FMolTotR=FlujoMolR-FlujoMolR*XA_Sup[j]  ##  Cálcula y actualiza el flujo␣
↪molar de la fase refinado.
        FMasTotR=FMolTotR*MMR_Mezcla  ##  Cálcula y actualiza el flujo masico de␣
↪la fase refinado.
        FVolTotR=FMasTotR/Rho_R  ##  Cálcula y actualiza el flujo volumétrico de␣
↪la fase refinado.

        FMolTotE=FlujoMolE+FlujoMolE*YA[j]  ##  Cálcula y actualiza el flujo␣
↪molar de la fase extracto.
        FMasTotE=FMolTotE*MME_Mezcla  ##  Cálcula y actualiza el flujo masico de␣
↪la fase extracto.
        FVolTotE=FMasTotE/Rho_E  ##  Cálcula y actualiza el flujo volumétrico de␣
↪la fase extracto.

        FracVolE=((FVolTotE)/(FVolTotR+FVolTotE))  ##  Cálcula y actualiza la␣
↪fracción volumétrica en la fase extracto.
        VelR=((FVolTotR)/(AreaF*(1-FracVolE)))  ##  Cálcula y actualiza la␣
↪velocidad de la fase refinado.
        VelE=((FVolTotE)/(AreaF*FracVolE))  ##  Cálcula y actualiza la velocidad␣
↪de la fase extracto.
        VelFases=VelR+VelE  ##  Cálcula y actualiza la velocidad de␣
↪deslizamiento de las fases.

        ReGota= ((DiamGota*VelFases*Rho_R)/(ViscR))  ##  Cálcula y actualiza el␣
↪número de Reynolds de la gota.
        ScR=((ViscR)/(Rho_R*Dif_NicAgua))  ##  Cálcula y actualiza el número de␣
↪Schmidt en el refinado.
```

```
        K_Skll=(ReGota**(1/8))*((ViscR/ViscE)**(1/4))*(((ViscR*VelFases)/
↪(TenSuperTol))**(1/6))  ##  Cálcula y actualiza la correlación de A.H.P.␣
↪Skelland.
        F_Skll=0.281+(1.615*K_Skll)+(3.73*K_Skll**2)-(1.874*K_Skll**3)  ## ␣
↪Cálcula y actualiza la corrección de forma esférica de la correlacion de A.H.P.
↪ Skelland.
        kX= (Dif_NicTol/DiamGota)*(2+0.463*(ReGota**0.484)*(ScR**0.
↪339)*(((DiamGota*(g**(1/3)))/(Dif_NicAgua**(2/3)))**(0.072)))*F_Skll  ## ␣
↪Cálcula y actualiza el coeficiente local de TdeMasa en la fase refinado.
        kY=(0.00375*VelFases/(1+(ViscE/ViscR)))  ##  Cálcula y actualiza el␣
↪coeficiente local de TdeMasa en la fase extracto.

        mFI=-kY/kX  ##  Cálcula y actualiza la pendiente de la fuerza impulsora␣
↪(FI).

        ##### MÉTODO DE NEWTON ####

        f = lambda x: -YA[j]-mFI*(XA_Sup[j]-x)+16.07781726*x+((0.08593565238*x)/
↪(x+0.000336023425))  ## Se define el polinomio que representa la CDE.

        XA_Equ[0]=XA_Sup[0]  ##  Se inicia el método de Newton para encontrar␣
↪las raices del polinomio y asi hallar el valor de las concentraciones en el␣
↪equilibrio.

        XA_Equ[j]= op.newton(f,XA_Equ[j],tol=1e-8,maxiter=100)  ##  Ejecuta el␣
↪método de Newton.

        YA_Equ[j]=16.07781726*XA_Equ[j]+((0.08593565238*XA_Equ[j])/(XA_Equ[j]+0.
↪000336023425))    ##  Cálcula la concentracion de la fase extracto en el␣
↪equilibrio.

        NA_EE[j]=kX*(XA_Sup[j]-XA_Equ[j])  ##  Cálcula el flux de transferencia␣
↪de masa en el equilibrio.

        Num_Got=FracVolE*(VolPart/VolGota)  ##  Determina el número de gotas de␣
↪la fase dispersa.
        AM[j]=Num_Got*AreaGota  ##  Cálcula el área de transferencia de masa.
        XA[j]=(XA_Part[j]*FlujoMolR-AM[j]*NA_EE[j])/FlujoMolR  ##  Cálcula la␣
↪concentración de A en el refinado en la línea de operación.
        YA[j+1]=(YA[j]*FlujoMolE-AM[j]*NA_EE[j])/FlujoMolE   ##  Cálcula la␣
↪concentración de A en el extracto en la línea de operación.


        ErrorEst=abs(XA[j]-XA_Sup[j])  ##  Analiza el error entre la␣
↪concentración supuesta y la calculada por el método.
```

```
        Contador=Contador+1  ##  Aumenta de uno en uno el contador del ciclo␣
↪while.

        XA_Sup[j]=XA[j]  ##  Condición para que en dado caso la concentración␣
↪supuesta no sea la misma cálculada, ingrese de nuevo al ciclo con la␣
↪concentración calculada.
```

Solución de ecuaciones para la CDE

- En esta sección se calculan valores para las concentraciones del soluto en ambas fases para graficar la CDE.

[ ]:
```
    XA_CDE[j]=j*DiscCDE  ##  Gráfica la concentración del soluto A en el␣
↪refinado en la CDE.
    YA_CDE[j]=16.07781726*XA_CDE[j] + (0.08593565238*XA_CDE[j])/(XA_CDE[j]+0.
↪000336023425) ##  Gráfica la concentración del soluto A en el extracto en la␣
↪CDE.
```

MÓDULO 6- Gráficas y resultados

- En este módulo se utiliza la librería matplotlib que permite gráficar los resultados obtenidos.
- Las líneas de código son simplemente acciones y caracteristicas para poder visualizar los datos.

Gráfica: Línea de operacion LdeO

[ ]:
```
pt.figure("TdeMasa contracorriente", [10,5])  ##  Crea figura y le otorga␣
 ↪dimensiones.
pt.plot(XA,YA,'k',linewidth=2)  ##  Grafica el tiempo vs Concentración deseada.
pt.xlabel("Refinado")  ##  Agrega título al eje x.
pt.ylabel('Extracto')  ##  Agrega título al eje y.
pt.title('Concentración y CDE')  ##  Agrega título en la parte superior.
pt.grid(True)  ##  Agrega la cuadrilla a la imagen
pt.tick_params(labelsize=8)  ##  Ajusta el tamaño de los títulos para los ejes ␣
 ↪(x , y), y de igual manera ajusta el tamaño de los números en cuadrilla(Grid).
leg = pt.legend(loc='best', ncol=1, mode="center", shadow=True, fancybox=True,␣
 ↪prop={'size': 7})
leg.get_frame().set_alpha(0.5)  ##  Agrega la leyenda en la grafica.("best" es␣
 ↪la mejor ubicación).
```

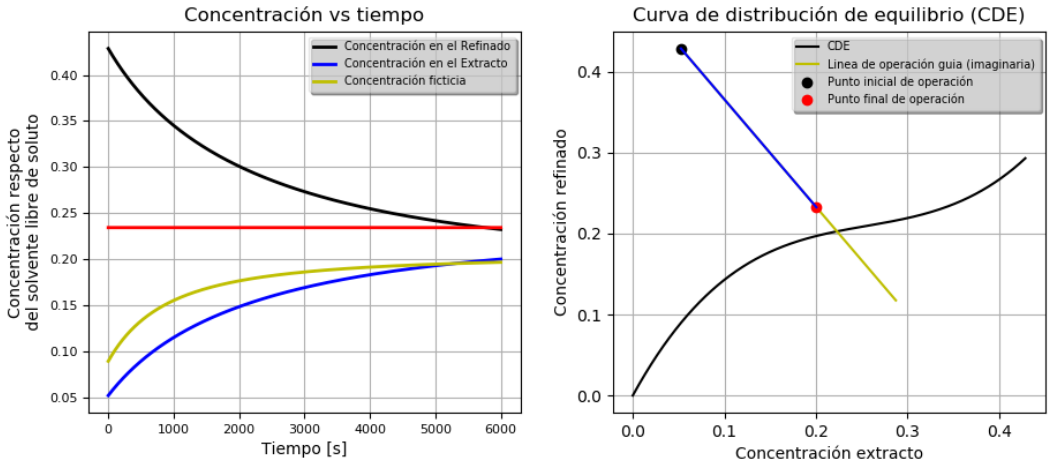
Gráfica: Curva de distribución de equilibrio (CDE)

[ ]:
```
pt.plot(XA_CDE,YA_CDE,'b',label='CDE')
pt.legend(loc='best')
```

4. Resultados

4.1. Resultados de la simulación para el proceso en lote puro

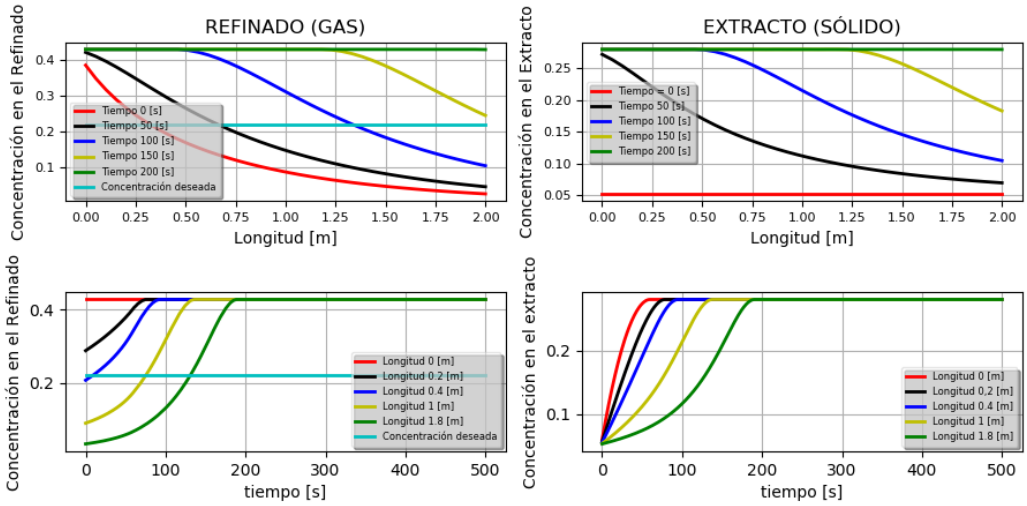
Figura 1: Proceso en operación Lote puro



En la figura 1, se pueden ver dos gráficas. La de la parte izquierda es la evolución en el tiempo del proceso, y la derecha posee la gráfica de la curva de distribución de equilibrio del sistema junto con la línea de operación. Vale la pena decir que el fragmento que sobrepasa la CDE es únicamente porque esta línea es un apoyo para el trazo de la línea de operación del proceso.

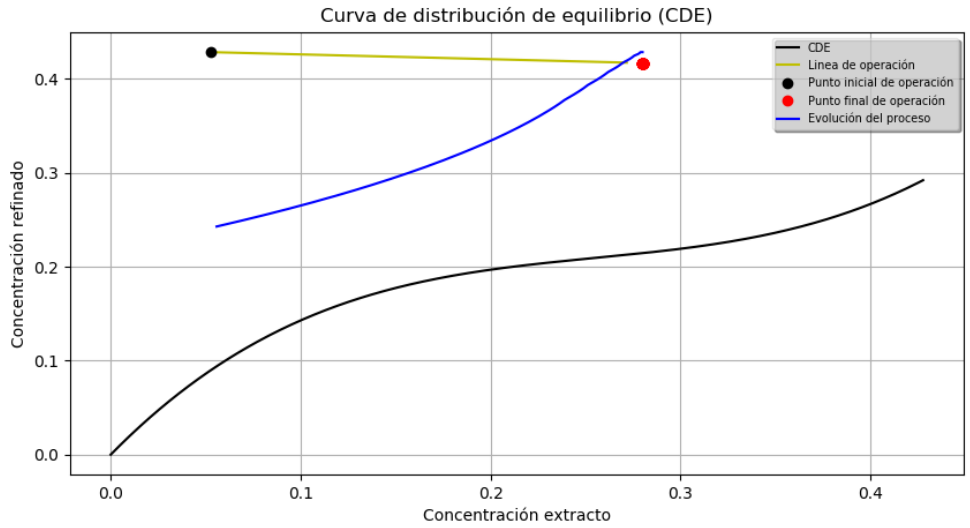
4.2. Resultados de la simulación para el proceso en Semi Lote

Figura 2: Proceso en operación semi lote



En la figura 2, se puede observar la evolución de la concentración de soluto A, tanto en el refinado como en el extracto. Las dos gráficas superiores, representan el comportamiento de la concentración en refinado y extracto en función de la longitud o cada tramo de la torre. Esto se realiza a intervalos de tiempo determinados, los cuales están enmarcados en las gráficas. Las otras dos gráficas de la parte inferior, representan el comportamiento de las concentraciones en función del tiempo a una longitud de la torre determinada.

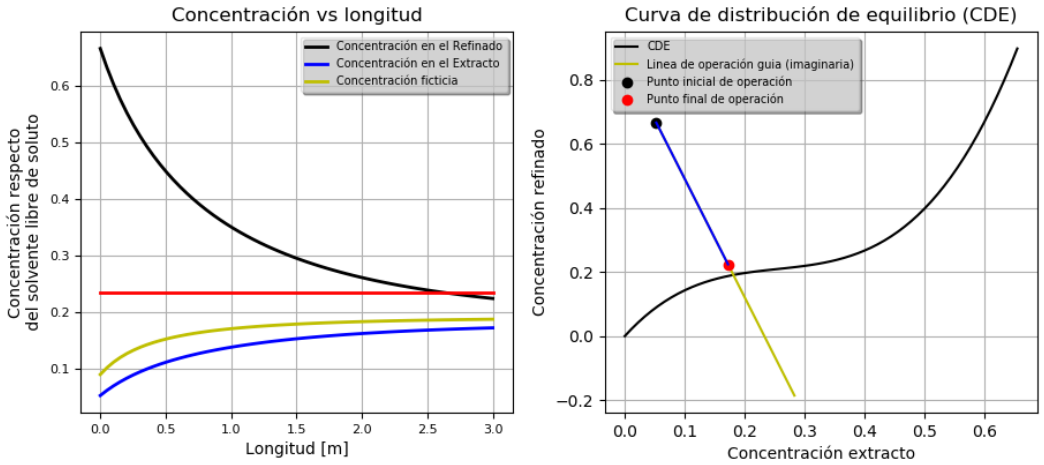
Figura 3: Proceso en operación semi lote (CDE)



En la figura 3 se puede ver la curva de distribución de equilibrio o (CDE) para el sistema en operación semilote. En esta se puede identificar la CDE, la línea de operación, los puntos iniciales y finales de operación y como es la evolución del proceso.

4.3. Resultados de la simulación para el proceso en Cocorriente

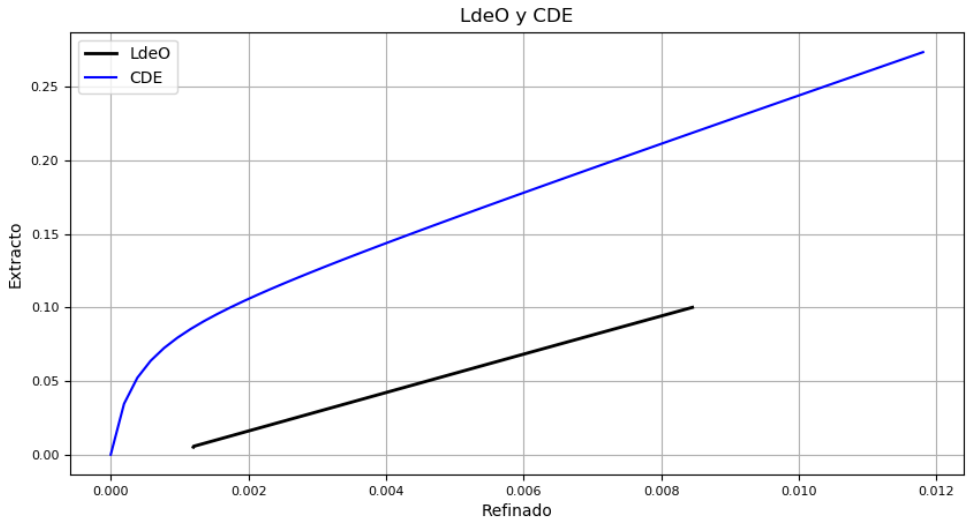
Figura 4: Proceso en operación Cocorriente



La figura 4 representa en la gráfica izquierda la evolución del proceso en función de la longitud de la torre o en cada tramo de esta. Se puede ver que la concentración de refinado alcanza su cometido a los 2.5 metros en la torre. En la parte derecha, se puede observar la curva de distribución de proceso del sistema, obsérvese que la línea de operación real no puede cruzar la (CDE), la línea que la cruza es únicamente un trazo que se utiliza para ayudar a trazar la verdadera linea de operación.

4.4. Resultados de la simulación para el proceso en Contracorriente

Figura 5: Proceso en operación Contracorriente



En la figura 5 se identifica para el proceso en contracorriente, la curva de distribución de equilibrio y la línea de operación hallada mediante el método iterativo expuesto anteriormente.

5. Conclusiones

- En este trabajo se logró crear un algoritmo capaz de simular procesos de transferencia de masa. Partiendo de cuatro modos de operación diferentes se resolvieron básicamente de la misma manera, pero no iguales siguiendo los pasos del algoritmo creado.
- Como se pudo evidenciar en la programación de cada uno de los simuladores, la secuencia es casi la misma no importa si es lote puro, semilote, cocorriente y contracorriente. La única diferencia que existe es que todos poseen parámetros y acciones distintas, pero el modo de operación es casi parecido entre todos los modos de operación vistos en este docuemento.
- Se pudieron identificar fácilmente las diferencias que tienen los diferentes modos de operación que puede tener un equipo de transferencia de masa.
- Se podría decir que este documento sirve de guía para programar y simular operaciones de transferencia de masa en diferentes tipos de operación.
- A un paso temporal y espacial mas cortos, el método se vuelve un poco mas exacto pero el gasto computacional es mas grande.
- Se pudo ver que los modos de operación mas complejos que se pueden programar o simular son: semi lote y contracorriente, el resto tiende a tener un modelamiento y programación un poco menos complicado.

Referencias

[1] ALVAREZ, H.D. «Trabajo de año sabático 2017: Efectos dinámicos en operaciones unitarias. Modelado de procesos para su análisis y control. »,págs. 19–29, 2017.

[2] ZULUAGA, C.C. «Simulación de procesos químicos (2019) Sofware Alternativos», *Operations Research*, **8**, págs. 32–41, 2019.