Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

Máster Universitario en
Ingeniería Industrial

Trabajo Fin de Máster

Uso de tecnologías Cape-Open para la integración de distintos simuladores de proceso, caso práctico.

Autor:
Sergio Santiago Rodríguez Illescas

Tutor:
José Juan Macías Hernández

Septiembre de 2022

Índice.

1. Resumen

El uso e implementación de software de simulación en la industria química se ha convertido en un factor fundamental para el diseño y la determinación de la viabilidad económica de cualquier tipo de proceso químico o equipos destinados a dichos procesos. Esto es debido a que los simuladores nos permiten determinar, con gran flexibilidad, los resultados del proceso simulado variando o fijando parámetros de diseño según nos convenga sin necesidad de llevar a cabo ningún tipo de inversión en material, equipos, instalaciones, etc.

Por otra parte, es necesario entender que, a pesar de los avances en calculo y computación, los resultados de dichos programas de simulación están condicionados por dos grandes factores, por una parte la habilidad del ingeniero que modela el proceso y por otra parte por los parámetros, aproximaciones, ecuaciones y cálculos que los desarrolladores de dichos software emplean y los cuales suelen estar enfocados a especializarse en obtener resultados muy precisos para procesos muy concretos, el cual suele ser la principal finalidad de uso o atractivo de venta del programa y obtener resultados aproximados a la realidad en el resto.

Es por ello que este trabajo de fin de Máster estudiara la integración de un caso práctico cuyas operaciones unitarias serán programadas en Python que, junto con el empleo de tecnologías CAPE-OPEN integradas en los software de simulación, permitan combinar las fortalezas de los distintos programas de simulación utilizados (propiedades de cálculo de equilibrio termodinámico y propiedades físicas, así como la programación y simulación de operaciones unitarias basadas en equipos reales) a fin de obtener una herramienta de simulación de procesos más potente, flexible y precisa.

2. Abstract

The use and implementation of process simulators software in the chemical industry has become a key factor for the design and the determination of economic viability of any process or equipment destinated to said process. This is because of the great flexibility simulation software gave us do to the fact they allow us to change or set process variables at our discretion and getting results out of them without the need of investing in any kind of materials, equipment, facilities, etc.

In the other hand, is necessary to understand that, despite the advances in calculus or computation power, the results obtain by this kind of simulation software are condition by two mayor factors. One is the ability of the engineer that has to model the process in the first place and the other are the parameters, approximations, equations and math that the developers of this kind of software use and they are usually focus on obtaining very precise results on very specific process which is usually the main purpose or selling point of the software and obtaining approximate results to the reality on the rest.

That is why, the main focus of this Master's thesis is to study the integration of a practical case with their unitary process units will be programmed in Python along with the use of CAPE-OPEN technologies already integrated in the simulation's software, will combine their main strengths (calculus properties of thermodynamic equilibriums, physical properties, simulation of unitary operations base on real equipment) in order to get a more powerful, precise and flexible simulation tool.

3. Objetivos.

Los objetivos que persigue este trabajo de fin de master son:

- Determinar las fortalezas de los programas de simulación estudiados, en este caso, UNISIM 390.1, COCO 3.5.0.2. y DWSIM 6.5.5.
- Plantear un caso práctico de un proceso químico real y realizar una simulación de dicho caso en UNISIM R390.1 y DWSIM 6.5.5. como marco base de comparación.
- Realizar una simulación de dicho caso práctico programando las distintas operaciones básicas en lenguaje Python en el simulador DWSIM y mediante Cape-Open aunar a dicha simulación el cálculo de propiedades termodinámicas y procesos del simulador COCO 3.5.0.2. para calcular y obtener resultados de simulación usando una combinación de los distintos simuladores.
- Analizar y comparar los resultados obtenidos entre las distintas simulaciones realizadas.

4. Objectives.

The objectives that this project seek to answer are:

- Determinate the strengths of the simulation's programs studied, in this case, UNISIM 390.1, COCO 3.5.0.2. and DWSIM 6.5.5.
- Propose a practical case of a real chemical process and simulate it in UNISIM R390.1 and DWSIM 6.5.5. as a basis of comparison.
- Make a simulation of the practical case using Python to code the different basic operations of the process and with Cape-Open join COCO 3.5.0.2. thermodynamic property packages to combine different simulations programs.
- Analise and compare the results obtain between the different simulations.

5. Introducción.

5.1. Evolución de la simulación de procesos.

La simulación de procesos nace en 1966 con el PROCESS, un software de simulación rudimentario que únicamente permitía la simulación de columnas de destilación. No fue hasta los años 1970 – 1980 cuando realmente se impulsó el desarrollo de herramientas para la simulación de procesos debido entre otros factores a:

- La estandarización del lenguaje FORTRAN entre científicos e ingenieros.
- Adopción del enfoque "secuencial modular" que sigue siendo el modelo de la arquitectura clásica en la mayoría de simuladores comerciales actuales.
- Las principales firmas de ingeniería, compañías petroquímicas y refinerías empiezan a desarrollar sus propios programas de simulación de manera independiente.
- Creación de proyecto ASPEN (Advanced System for Process Engineering) por el Departamento de Energía de Estados Unidos, esto daría como resultado la creación de Aspen Plus.

En 1982, se crea el simulador HYSIM el cual sería renombrado posteriormente como HYSYS y comercializados por Hyprotech Ltd., este simulador adquiere rápidamente fama por su potencia de simulación y la variedad de procesos disponibles para simular.

En 1997 Hyprotech Ltd. seria adquirida por AEA Technologies y acabaría finalmente en manos de AspenTech en 2002 la cual crearía una familia de simuladores basados en Aspen Plus y HYSYS.

Sin embargo, en 2004, y de acuerdo a las leyes antimonopolio, ApenTech fue obligada a vender la propiedad intelectual adquirida a Hyprotech a Honeywell, manteniendo la posibilidad de futuro desarrollo y comercialización de productos basados en dicha propiedad intelectual.

Por su parte Honeywell desarrollo UNISIM Design, basado en HYSYS y desarrollo una agresiva campaña de marketing ofreciendo licencias de programa a bajo precio, gran potencia y flexibilidad en cálculo de simulaciones estáticas y dinámicas e interconectividad con el resto de productos desarrollados por Honeywell (sensores, controladores, etc.) esta ha permitido extender considerablemente la presencia de

Honeywell en este mercado convirtiéndolo en referente en cuanto a software de simulación se refiere al haber sido adoptado en la mayor parte de la industria.[1]

5.2. Software libre de simulación de procesos.

Si bien el mercado comercial se encuentra dominado por software con licencias tales como UNISIM y HYSYS, existen otras alternativas gratuitas de características similares disponibles más extendidas en el entorno educativo y de empresas de menor envergadura.

Muchos de estos programas de simulación nacieron como Macros y programas de cálculo en Visual Basic de Excel y con el tiempo han evolucionado debido a las continuas contribuciones de usuarios y programadores. Ejemplos de esto son los simuladores gratuitos ampliamente extendidos y los cuales emplearemos para la realización de este estudio DWSIM y COCO.

5.3. CAPE-OPEN.

Esta iniciativa de origen europeo fue fundada en 1997 por entidades de reconocida reputación y gran peso en la industria química, farmacéutica y petroquímica como son Bayer ,BASF, BP, DuPont, Instituto Frances del Petróleo, Elf Aquitaine e Imperial Chemical Industries así, junto con otros 15 socios en los que se incluyen desarrolladores de software y universidades, plantearon el objetivo de estandarizar las especificaciones dirigidas a las tecnologías de simulación que permitirían la interconexión y operabilidad entre distintos entornos de simulación y componentes de modelado de terceros.

El desarrollo de esta estandarización en la comunicación entre simuladores tiene como ventaja, aparte de emplear las especialidades de modelado de cada simular:

-La reducción de costes de mantenimiento al estandarizar la interfaz y lenguaje del programa.

-Permitir la interconexión de las mejores características de modelado, no solo entre programas de simulación, sino también con otras herramientas de cálculo como MATLAB o Excel. Esto permitiría aprovechar las fortalezas de cada programa a fin de desarrollar una

simulación lo más fiel posible a la realidad.

Si bien este proyecto se dio por finalizado de forma exitosa en 1999 al establecer las bases para la interconectividad entre herramientas de simulación, también se estableció de forma subsecuente una entidad sin ánimo de lucro (CO-LaN), la cual sigue activa hoy en día, a fin de mantener los estándares de interconectividad y fomentar el desarrollo de las interfaces CAPE-OPEN.[2]

6. Introduction

6.1. Evolution of process simulators

Process simulation was born in 1966 with PROCESS, a rudimentary simulation software that it only allowed the simulation of distillation columns. It was not until the 1970s -1980s when the development of tools for process simulation was really promoted due to, among other factors:

- The standardization of FORTRAN among scientists and engineers.
- Adoption of the "modular sequential" approach which is still use as the basic architecture model in most of the current commercial simulators.
- The main engineering firms, petrochemical companies and refineries begin to develop their own simulation programs independently.
- Creation of the ASPEN Project (Advanced System for Process ENgineering) by the United State Department of Energy, this would result in the creation of ASPEN Plus.

In 1982, the HYSIM simulator was created, which would later be renamed HYSYS and marketed by Hyptotech Ltd., this simulator quickly became famous for its simulation power and the variety of processes available to simulate. In 1997 Hyprotech Ltd. would be acquired by AEA Technologies and would finally end up in the hands of AspenTech in 2002 which would create a family of simulators based on Aspen Plus and HYSYS.

However, in 2004, and in accordance with antitrust laws, ApenTech was force to sell the intellectual property acquired from Hyprotech to Honeywell but keeping rights of future development and commercialization of products based in said intellectual property.

For its part, Honeywell develop UNISIM Design, based on HYSYS, and created an aggressive marketing campaign, offering program licenses at low prices with access to great calculating power in static or dynamic simulations, as well as interconnectivity with the rest of the products develop by Honeywell (sensors, controllers, …) this has allowed Honeywell's presence to be considerably extended in this market, making it a benchmark in terms of simulation software by having been adopted in most of the industry.[1]

6.2. Open-access process simulation software.

While the commercial market is dominated by licensed software such as UNISIM and HYSYS, there are other free alternatives with similar features available more widely in the educational environment and for small companies.

Many of these simulation programs were born as macros and calculation programs in Excel's Visual Basic and have evolved over time due to the continuous contributions of users and programmers. Examples of this are the widely extended free simulators that will be use to carry out this study, DWSIM and COCO.

6.3. CAPE-OPEN

This initiative of European origin was founded in 1997 by entities of recognized reputation and with great weight in the chemical, pharmaceutical and petrochemical industry, such as, BAYER, BASF, BP DUPONT, French Petroleum Institute, Elf Aquitaine and Imperial Chemical Industries, as well as another 15 partners, including software developers and universities. They set the goal of standardizing specifications for simulation technologies that would enable interconnection and interoperability between different simulation environments and third-party modelling components.

Apart from using each simulation program's strong points, this would also have the advantages of:

- Reduction of maintenance costs by standardizing the interface and language of the program.

-Allow the interconnection of the best modelling features, not only between simulation programs but also other calculation tools such as MATLAB or

EXCEL. This, as said before, will make possible to use the strengths of each program in order to develop a simulation that is as faithful as possible to reality.

Although this project was successfully completed in 1999 by establishing the basis for interconnectivity between simulation tools, a non-profit entity (CO-LaN) was also subsequently established, which is still active today, in order to maintain interconnectivity standards and promote the development of CAPE-OPEN interfaces.[2]

7. Simuladores empleados.

7.1. UNISIM R390.1:

Desarrollado por Honeywell, es una de las versiones más extendidas de este software de simulación. Si bien existen versiones más recientes desarrolladas por Honeywell (la más reciente a fecha de este trabajo es la versión R480) se ha escogido trabajar en la versión R390.1 pues la universidad posee una licencia para su uso con fines educativos. Entre sus funciones destacan el cálculo de simulaciones en carácter estático y dinámico, acceso a extensas bases de datos y métodos de cálculo incluidas en la licencia de software.

7.2. COCO 3.5.0.2:

COCO es un entorno de simulación de estado estacionario gratuito desarrollado inicialmente como campo de pruebas para la arquitectura de Cape Open. Este evoluciono en un programa de simulación propiamente dicho que permite la interconexión con cualquier otro tipo de simulador de procesos. Este programa hace uso de bases de datos y librerías creadas y expandidas por sus usuarios y fuentes de licencia libre para el cálculo de operaciones unitarias o propiedades físicas de compuestos. Además, con el tiempo, se ha incorporado el uso de add-ins especializados en otro tipo de funciones más específicas tales como paquetes de cálculos hidráulicos y de vapor de agua, mejoras en el interfaz de simulación, ampliaciones de las bases de datos estándar, etc.

7.3. DWSIM:

DWSIM es un simulador de procesos gratuito open-source basado originalmente en macros programadas en Excel y VBA. Su principal fortaleza radica en que

los modelos de cálculo, bibliotecas de paquetes termodinámicos y operaciones pueden ser creados y compartidos por los propios usuarios (dado el hecho de ser open-source) por lo que posee una extensa librería de operaciones unitarias y cálculo de propiedades, contando incluso, con simulación dinámica limitada de procesos a diferencia de muchos otros softwares de simulación gratuitos. Sin embargo, también posee sus desventajas pues no puede hacer uso de herramientas de cálculo de pago o bajo licencia y, al incorporar código cedido de terceros, ve disminuida la estabilidad del programa esto último queda mitigada por el extenso soporte y parches que recibe este software de manera asidua. [3][4]

8. Caso práctico de estudio de estudio.

El caso práctico del que se realizaran las simulaciones es una adaptación al expuesto en el libro "Distillation," in Ludwing's Applied Process Design for Chemical an Petrochemical Plants, 4th ed., vol II por K.A. Coker.[5] [6]

En dicho problema se plantea la separación a presión atmosférica de una corriente equimolar de Benceno-Tolueno con un flujo de 100 kmol/h en condiciones de líquido saturado. La separación deberá llevarse a cabo para obtener una corriente de cabeza en el separador con una concentración molar del 95% Benceno y una corriente de fondo del separador del 95% en Tolueno.

Las modificaciones que aplicaremos a este problema serán las siguientes:

- La corriente de alimentación al separador provendrá de la unión de dos corrientes con las siguientes características:

Corriente Alim. 1	Corriente Alim. 2
Flujo: 50 Kmol/h	Flujo: 50 Kmol/h
Concentración molar: 60% Tolueno y 40% Benceno	Concentración molar: 40% Tolueno y 60% Benceno
Presión: 3 atm	Presión: 3 atm
Condición de líquido saturado	Condición de líquido saturado

- La separación se llevará a cabo mediante un separador flash, por lo tanto, las condiciones de separación corresponderán al equilibrio liquido vapor de una sola etapa de separación por lo que no se podrá obtener la pureza del 95% de Benceno

en cabeza y 95% de Tolueno en fondo del caso original.

- La corriente de cabeza del separador se enfriará hasta su punto de burbuja.

Por otro lado, realizaremos algunas suposiciones a fin de simplificar el problema.

- Consideraremos que la mezcla Benceno-Tolueno formara una fase liquida/gaseosa ideal. [7]
- De acuerdo a lo anterior, la volatilidad relativa del Benceno respecto al Tolueno se puede calcular como la proporción de la presión parcial de ambos compuestos a la temperatura de interés. Debido a esto, las propiedades termodinámicas serán calculadas en las simulaciones mediante la ecuación de estado SRK (Soave-Redlich-Kwong).

9. Simulación mediante UNISIM R390.1

Para desarrollar la simulación en UNISIM, primero hay que configurar el entorno de simulación, esto es, la elección de componentes y el paquete termodinámico. Para ello, una vez abierto un nuevo proyecto en UNISIM se nos mostrara una ventana denominada “Simulation Basis Manager”, en la pestaña de componentes clicamos añadir (Figura 1.) y añadimos el Tolueno y el Benceno. (Figura 2.).

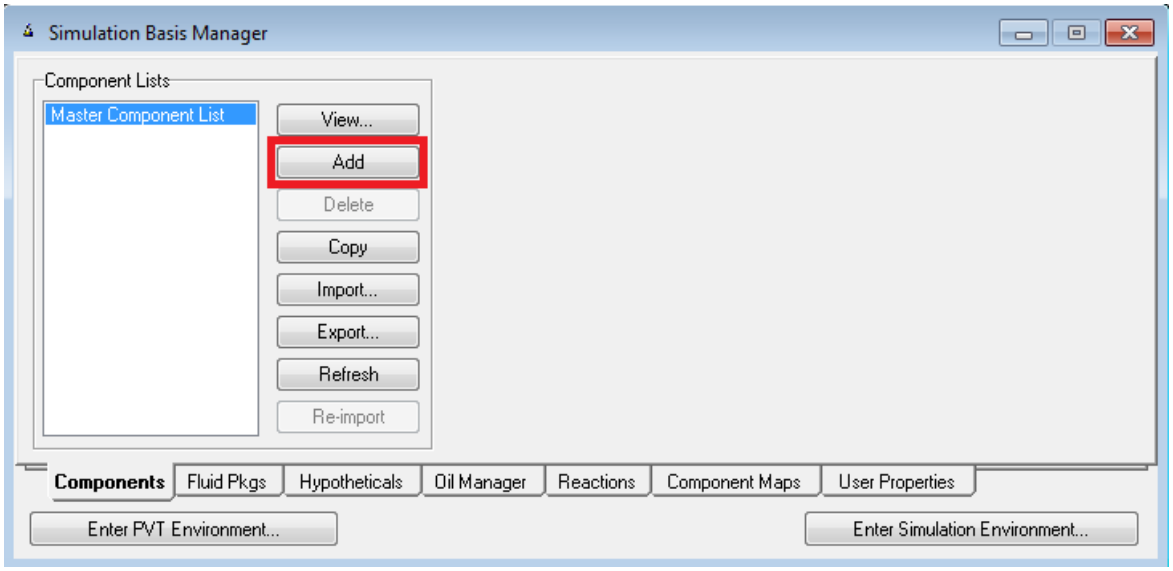

Figura 1. Pestaña de configuración de entorno de simulación UNISIM R390.1

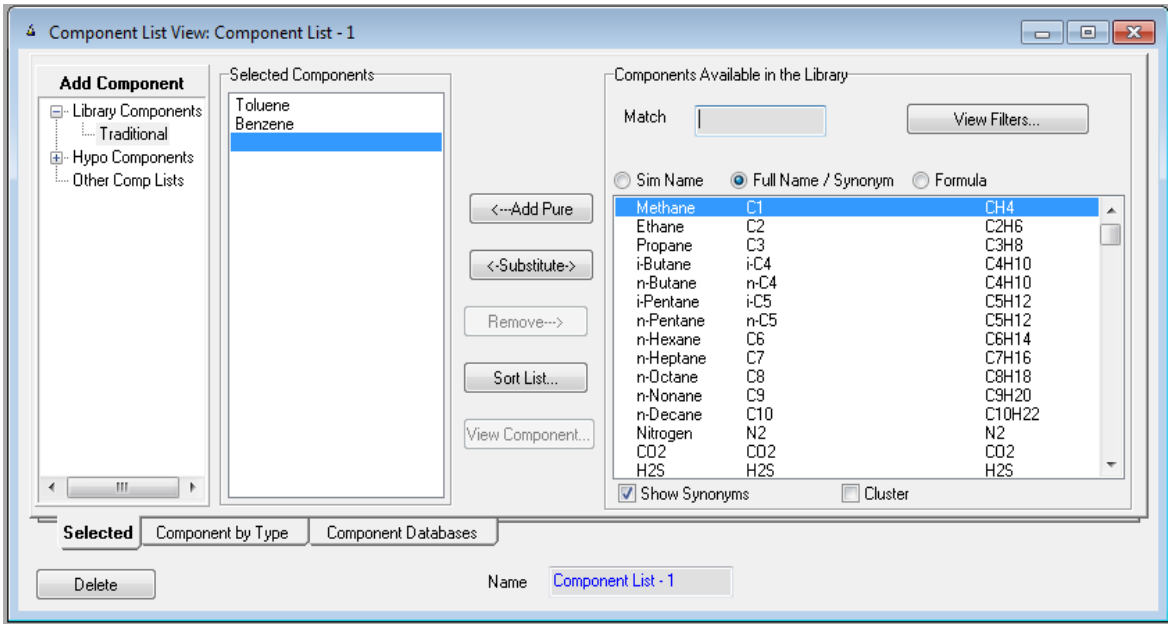

Figura 2. Adición de compuestos en UNISIM R390.1

Finalmente, seleccionamos la pestaña Fluid Pkgs y añadimos un nuevo paquete, este será la ecuación de estado que la simulación empleará, en este caso SRK (Figura 3.).

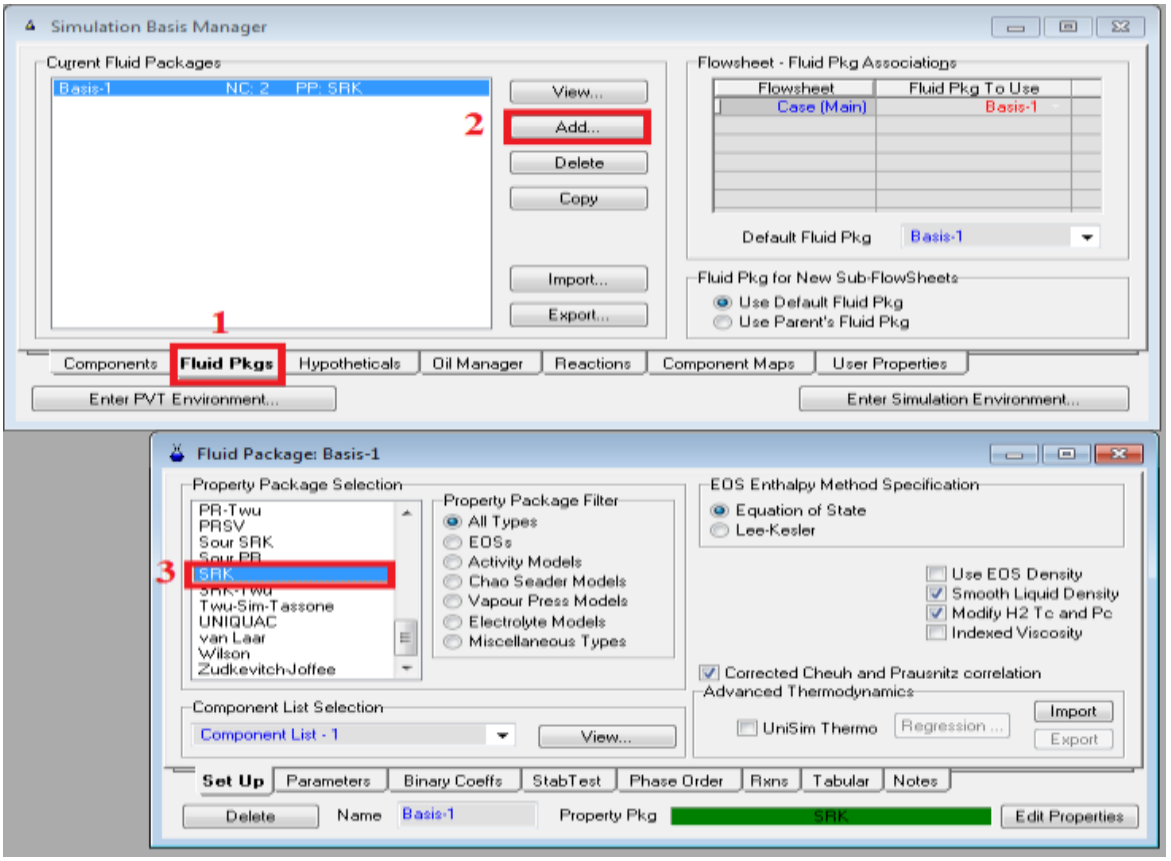

Figura 3. Selección de paquete termodinámico en UNISIM R390.1

Una vez realizado esto podemos proceder al entorno de simulación, en el, definiremos las corrientes y operaciones unitarias para definir el proceso, por lo que nos quedara (Figura 4.).

Corrientes:	Operaciones Unitarias:
Corriente Alim. 1.	Separador Flash.
Corriente Alim 2.	Mezclador.
Corriente de mezcla.	Válvula
Corriente al Separador.	Refrigerador (Cooler)
Corriente Cabeza del Separador.	-
Corriente Fondo del Separador.	-

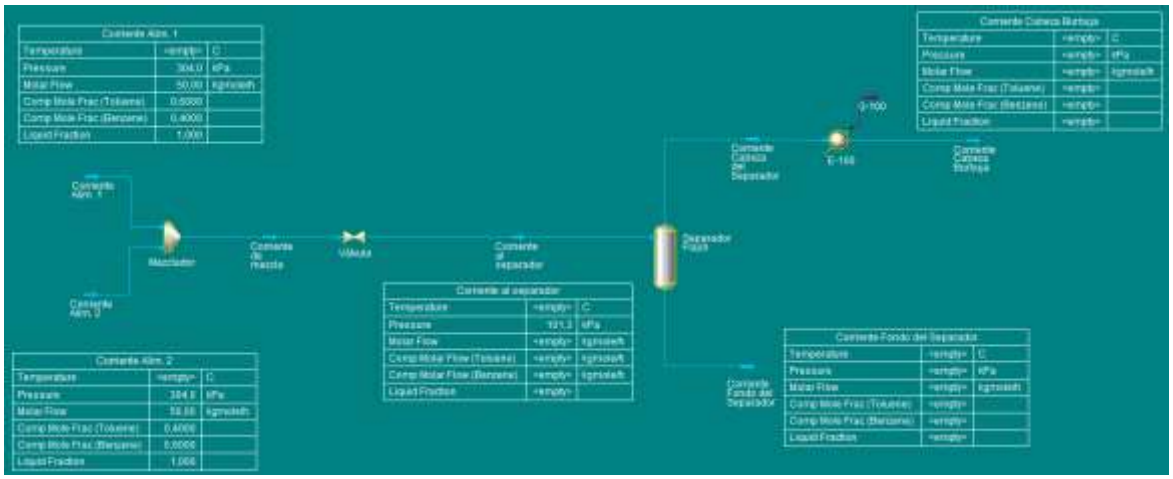

Figura 4. Diagrama de Flujo del proceso en UNISIM R390.1

A continuación, introducimos los datos del problema en las corrientes del proceso correspondientes para que el simulador calcule los resultados de las corrientes de cabeza y fondo del separador (Figura 5) los cuales tendrán los siguientes valores:

Corriente Cabeza Burbuja
Temperatura: 87,99 ºC
Presión: 101,3 kPa (1Atm)
Flujo molar: 21,93 Kmol/h
Fracción molar Tolueno: 33,57%
Fracción molar Benceno: 66,43%

Corriente Fondo del Separador
Temperatura: 93,73 ºC
Presión: 101,3 kPa (1Atm)
Flujo molar: 78,07 Kmol/h
Fracción molar Tolueno: 54,62%
Fracción molar Benceno: 45,38%

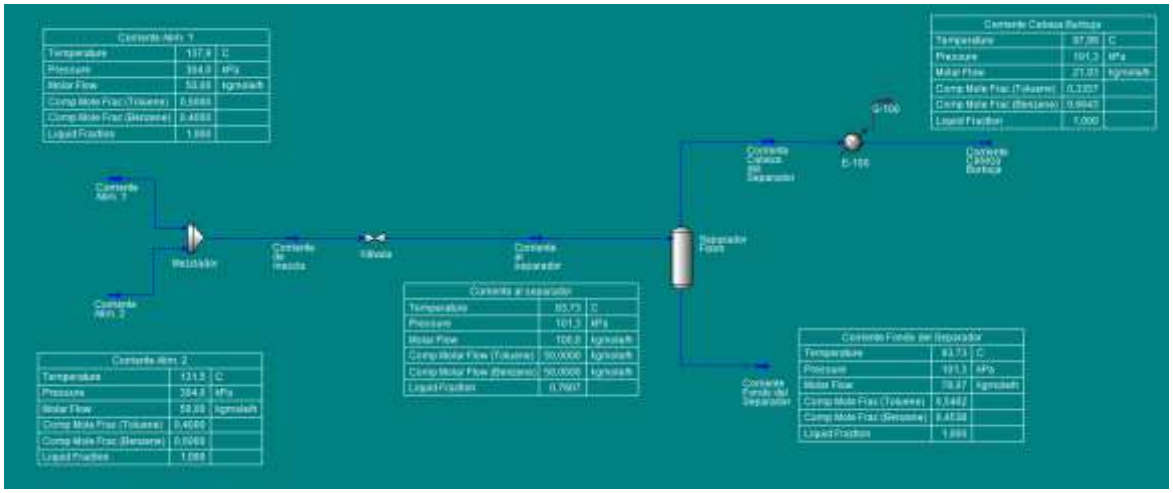

Figura 5. Diagrama resuelto del proceso en UNISIM R390.1

10. Simulación mediante DWSIM 6.5.5.[4]

Al iniciar el programa DWSIM, se nos presentara un menú con diferentes opciones, en nuestro caso elegiremos crear un nuevo proyecto de simulación (Figura 6.). Esto abrirá el entorno de simulación junto con el asistente “Wizard” el cual nos permitirá configurar una serie de opciones para la simulación.

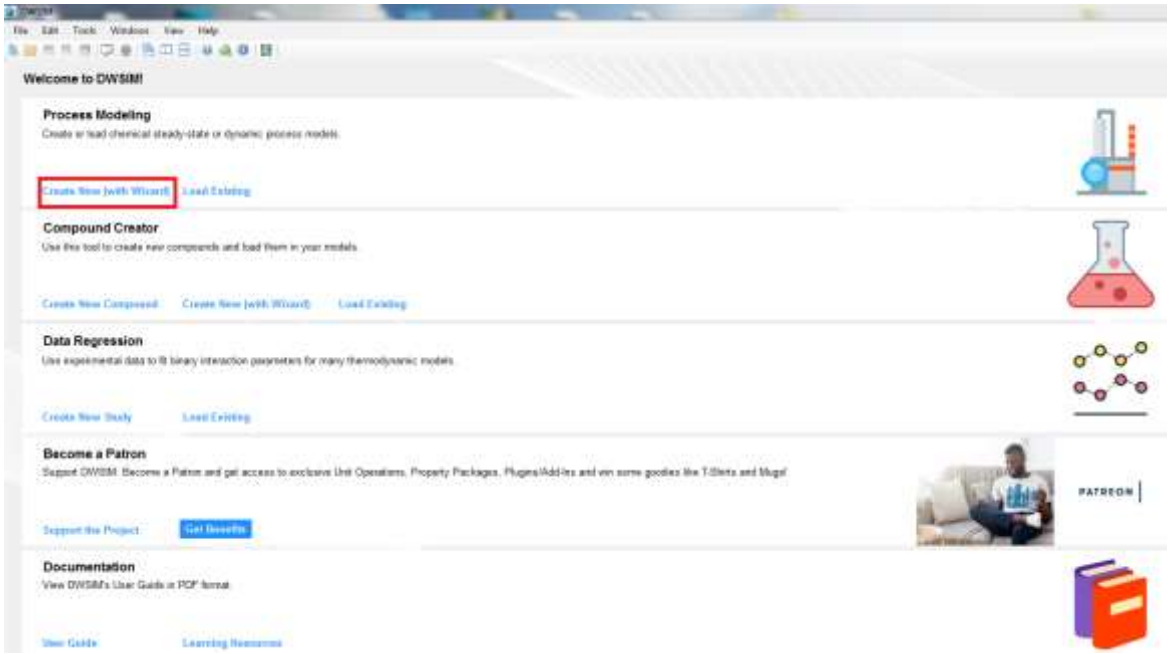

Figura 6. Diagrama resuelto del proceso en UNISIM R390.1

En primer lugar, seleccionaremos los componentes de los que hará uso la simulación, en este caso Tolueno y Benceno (Figura 7.). La siguiente opción será elegir el paquete termodinámico que usará la simulación, en este caso y al igual que en la simulación previamente discutida en UNISIM R390.1, usaremos la ecuación de estado SRK tal y como se encuentra en la base de datos del DWSIM (Figura 8.).

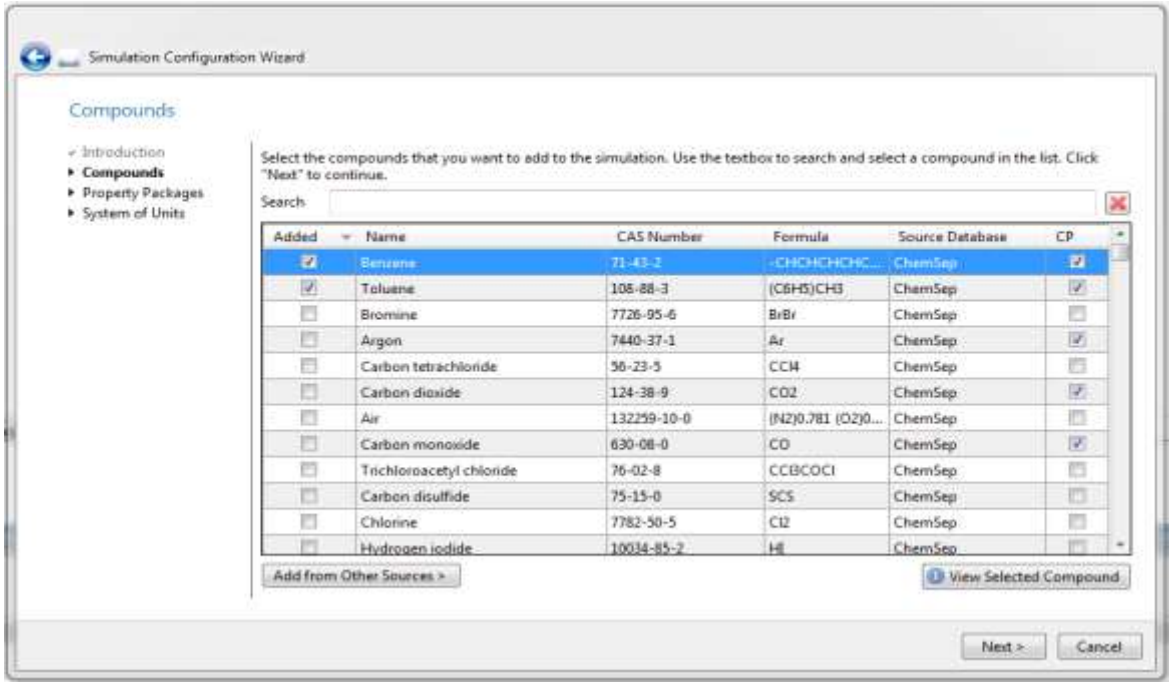

Figura 7. Selección de compuestos en DWSIM 6.5.0.0.

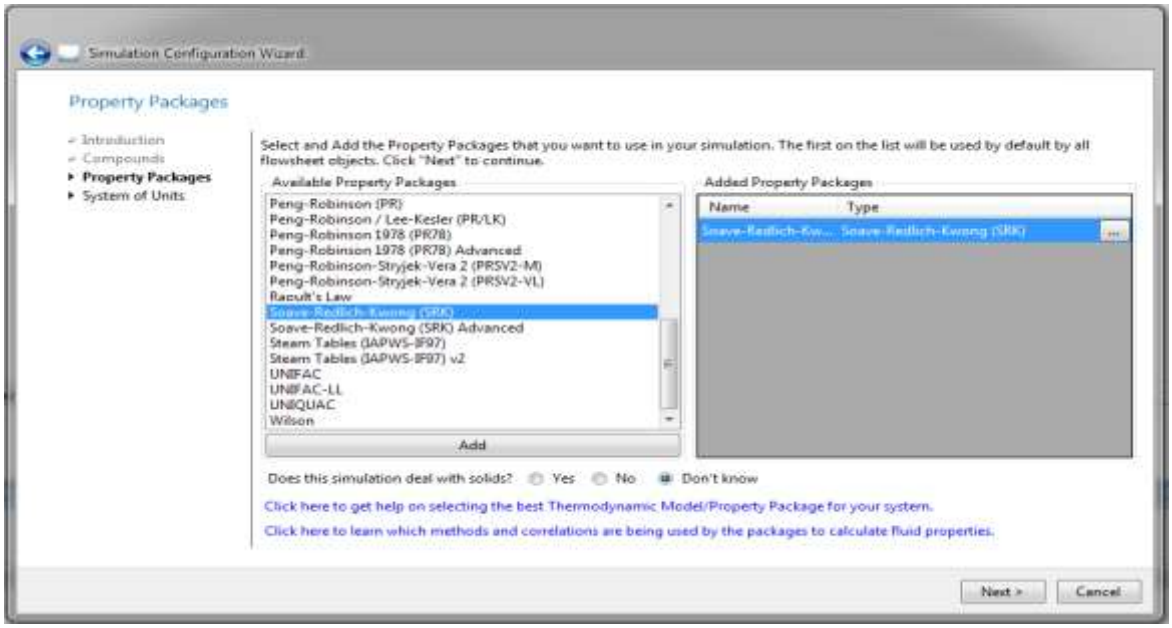

Figura 8. Selección de paquete termodinámico en DWSIM 6.5.0.0.

Finalmente, y a diferencia de UNISIM, DWSIM da la opción de antemano de personalizar las unidades de medida en la que la simulación trabajara para cada nuevo proyecto. En este caso se han cambiado las unidades de trabajo a las misma que tiene UNISIM a fin de facilitar la lectura de resultados.

Tras finalizar el asistente, entraremos al entorno de simulación propiamente dicho y podremos establecer las diferentes corrientes de materia y operaciones unitarias desde la barra de herramientas (Figura 9.) de forma parecida a UNISIM. Una vez conectadas las corrientes de materia y operaciones unitarias se introducirán los datos pertinentes de cada corriente lo cual permitirá al programa de simulación calcular el resultado de las corrientes de salida del separador. Estas corrientes tendrán los siguientes resultados:

Corriente Cabeza Burbuja
Temperatura: 87,6932 ºC
Presión: 101,3 kPa (1Atm)
Flujo molar: 22,083 Kmol/h
Fracción molar Tolueno: 33,18%
Fracción molar Benceno: 66,81%

Corriente Fondo del Separador
Temperatura: 93,7741 ºC
Presión: 101,3 kPa (1Atm)
Flujo molar: 77,917 Kmol/h
Fracción molar Tolueno: 54,76%
Fracción molar Benceno: 45,23%

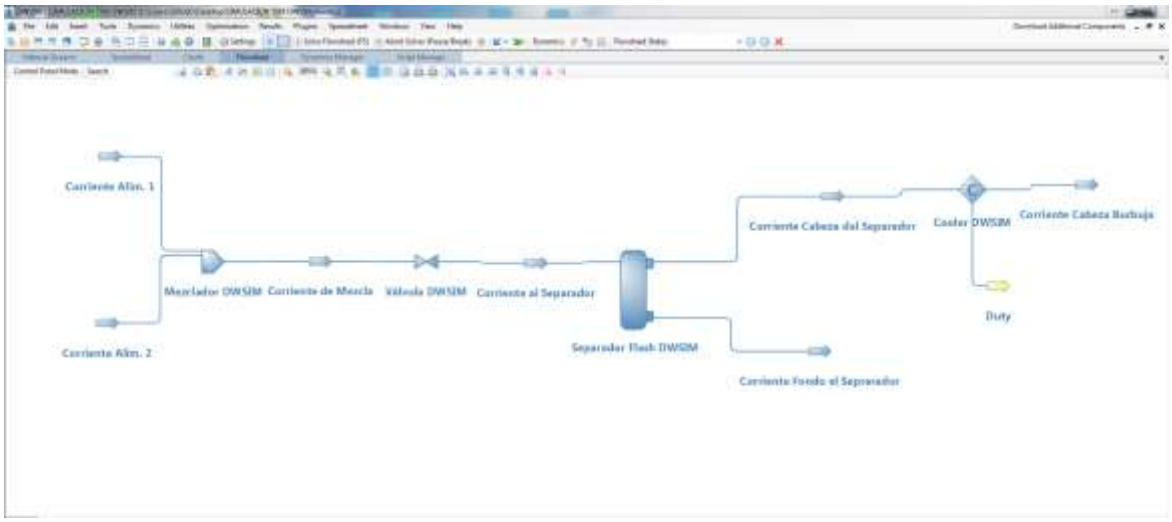

Figura 9. Diagrama resuelto del proceso en DWSIM 6.5.0.0.

11. Simulación mediante DWSIM 6.5.5. y COCO 3.5.0.2. con programación en Python.

El procedimiento para iniciar una simulación en DWSIM haciendo uso de paquetes termodinámicos de COCO 3.5.0.2. y operaciones unitarias en Python es similar al anteriormente descrito para simulaciones únicamente en DWSIM. Al iniciar el programa y crear un nuevo proyecto de simulación, se abrirá el asistente Wizard, igual que el caso anterior, seleccionamos los componentes químicos de la simulación (Tolueno y Benceno). El primer gran cambio llega con la elección del paquete termodinámico, en lugar de seleccionar SRK como en el caso anterior, seleccionaremos "CAPE-OPEN" (Figura 10.)

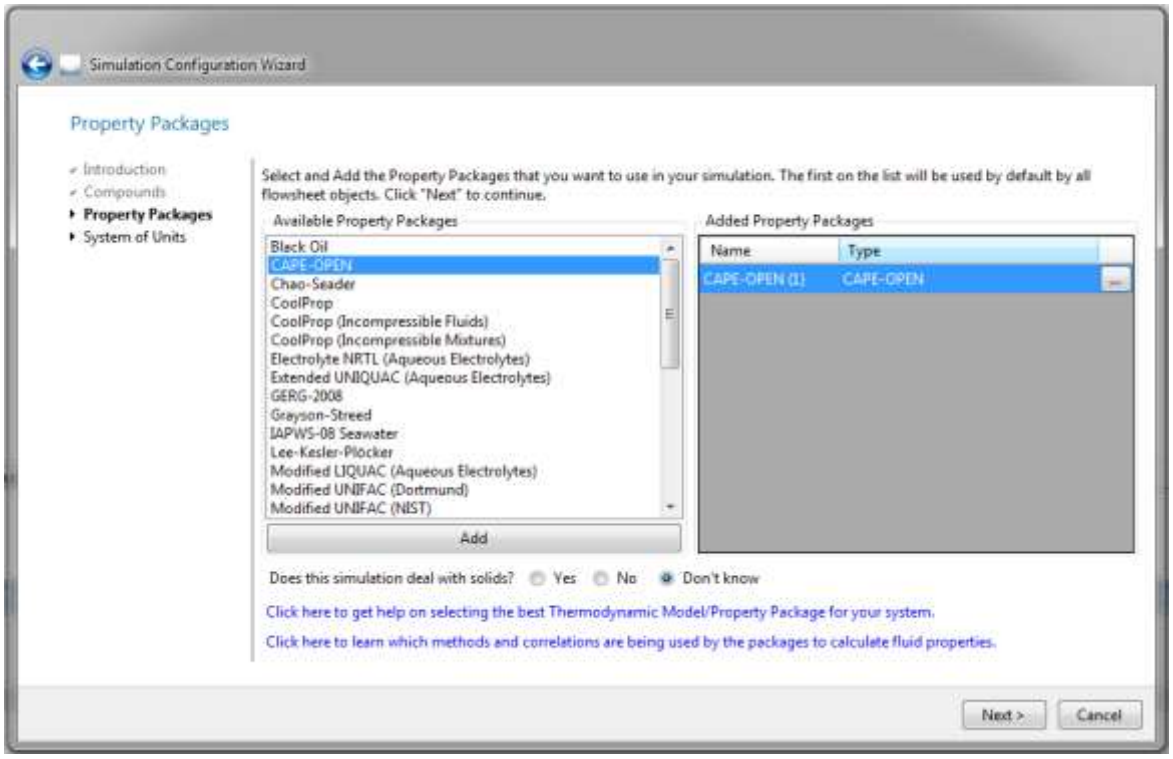

Figura 10. Paquete termodinámico en CAPE-OPEN de DWSIM 6.5.0.0.

Esto abrirá una interfaz que permitirá cargar las bases de datos CAPE-OPEN instaladas en el ordenador, así como las distintas librerías que dicha base de datos engloba. A continuación, clicaremos la pestaña junto a la opción "Thermo Server / Prop. Package Manager" y elegiremos la base de datos "TEA CAPE-OPEN 1.1" que corresponde al paquete termodinámico del simulador COCO 3.5.0. 2. (Figura 11.).

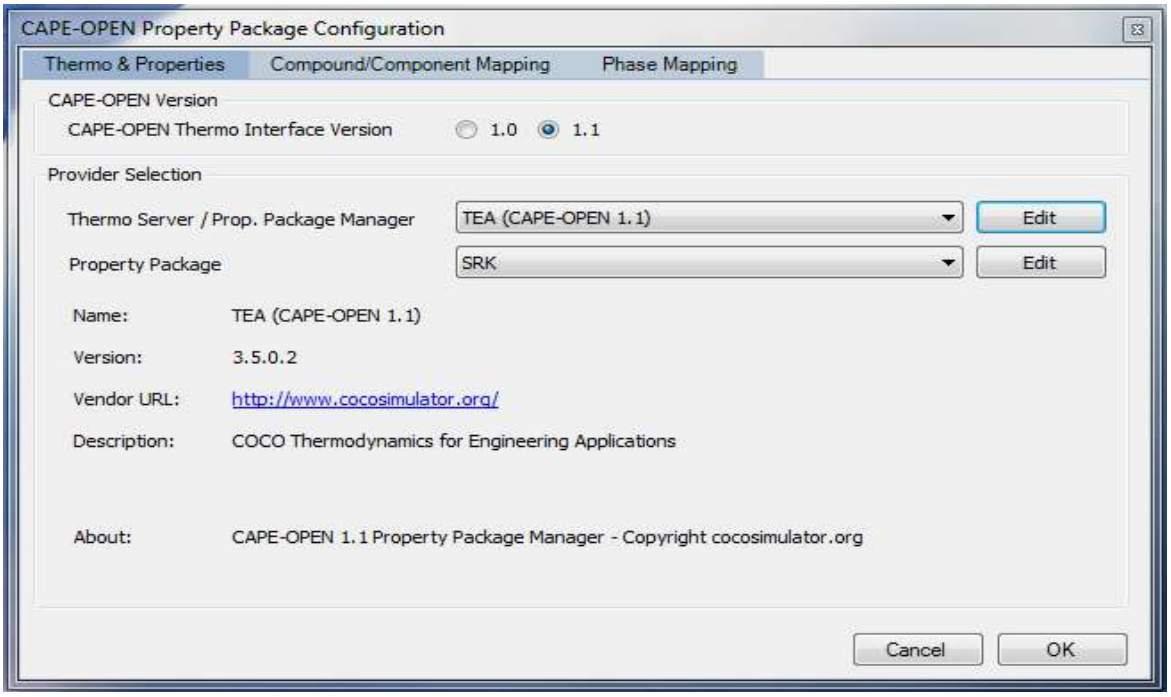

Figura 11. Selección del paquete termodinámico de COCO en DWSIM 6.5.0.0.

Finalmente elegimos la librería que deseamos cargar, en este caso, crearemos una nueva librería en la que agregaremos los componentes de tolueno y benceno, así como el modelo de SRK tal y como se encuentra en el simulador COCO. (Figura 12.).

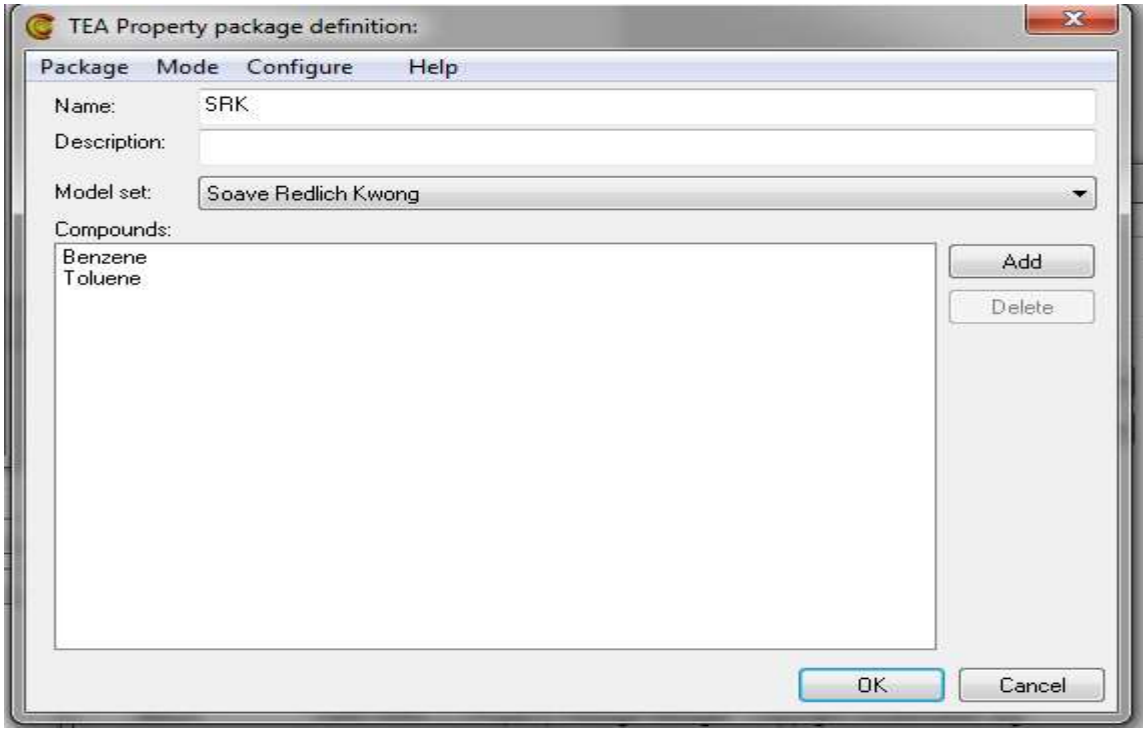

Figura 12. Configuración de paquete de propiedades de COCO en DWSIM 6.5.0.0.

Una vez en el entorno de simulación, introducimos las corrientes de materia como hicimos en el caso anterior. Sin embargo, en lugar de establecer las operaciones unitarias de la misma forma, elegimos en la ventana de la barra de operaciones la pestaña "USER MODELS" lo que nos dará la opción de establecer un script de Python el cual programaremos con el comportamiento y funcionamiento de la operación unitaria que deseemos (Figura 13.).

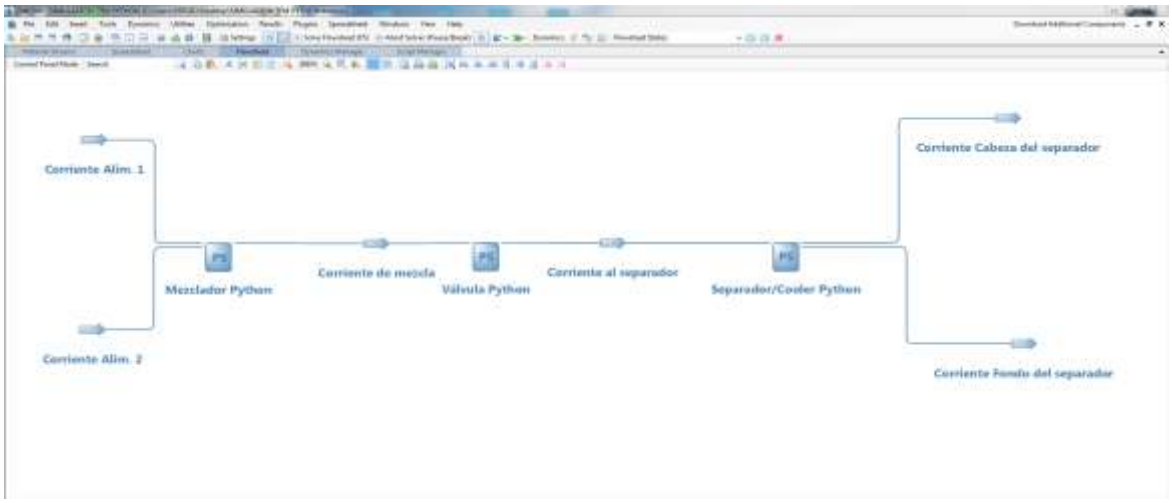

Figura 13. Diagrama del proceso de operaciones en Scripts de Python, DWSIM 6.5.0.0.

Tras realizar las conexiones entre las corrientes de materia y las operaciones unitarias procedemos a configurar dichas operaciones mediante Python, para ello, al clicaremos en la opción "Open Python Script Editor" lo cual abrirá el editor de código de Python con el que podremos configurar la operación unitaria.

11.1. Python [8][9]*:*

Si bien toda la simulación podría ser ejecutada en un único script de Python, se ha codificado cada operación por separado para que la apariencia sea parecida a las del resto de simuladores y los resultados de las distintas corrientes sea más fácil de observar ya que tendremos acceso directo a las corrientes intermedias. A continuación, se describirá la función y el código de cada operación unitaria.

11.2. Mezclador:

Se función consistirá en unir ambas corrientes de alimentación del sistema, calculando las variables termodinámicas resultantes de la mezcla. A diferencia de las

simulaciones en UNISIM y DWSIM en la que la fase se encuentra con una fracción muy pequeña vaporizada, la fase en la que se encuentra la mezcla a la salida de este mezclador será exclusivamente liquida.

11.2.1. Código Python mezclador:

```
from DWSIM.Thermodynamics import * #Carga en el script, el paquete termodinámico empleado en la simulación, en este caso el paquete CAPE-OPEN de COCO

alimentacion1=ims1 #Fija la corriente de alimentación 1 como la entrada 1 de la operación

alimentacion2=ims2 #Fija la corriente de alimentación 2 como la entrada 2 de la operación

Presion_1 = alimentacion1.GetProp("pressure", "Overall", None, "", "") #Almacena en la variable Presion_1 el valor de la presión de la corriente de alimentación 1

Presion_2 = alimentacion2.GetProp("pressure", "Overall", None, "", "") #Almacena en la variable Presion_2 el valor de la presión de la corriente de alimentación 2

flujomasico_1 = alimentacion1.GetProp("totalFlow" ,"Overall", None, "", "mass") #Almacena en la variable flujomasico_1 el valor del flujo masico de la corriente de alimentación 1

flujomasico_2 = alimentacion2.GetProp("totalFlow" ,"Overall", None, "", "mass") #Almacena en la variable flujomasico_2 el valor del flujo masico de la corriente de alimentación 2

fraccionmolar_1 = alimentacion1.GetProp("fraction", "Overall", None, "", "mole") #Almacena en la variable fraccionmolar_1 el valor de las fracciones molares de los componentes de la corriente de alimentación 1

fraccionmolar_2 = alimentacion2.GetProp("fraction", "Overall", None, "", "mole") #Almacena en la variable fraccionmolar_2 el valor de las fracciones molares de los componentes de la corriente de alimentación 2

flujomolar_1 = alimentacion1.GetProp("totalFlow" ,"Overall", None, "", "mole") #Almacena en la variable flujomasico_1 el valor del flujo molar de la corriente de alimentación 1

flujomolar_2 = alimentacion2.GetProp("totalFlow" ,"Overall", None, "", "mole") #Almacena en la variable flujomasico_2 el valor del flujo molar de la corriente de alimentación 2

entalpia_1 = alimentacion1.GetProp("enthalpy" ,"Overall", None, "Mixture", "mass") #Almacena en la variable entalpia_1 el valor de la entalpia de la mezcla liquida de la corriente de alimentación 1

entalpia_2 = alimentacion2.GetProp("enthalpy" ,"Overall", None, "Mixture", "mass") #Almacena en la variable entalpia_2 el valor de la entalpia de la mezcla liquida de la corriente de alimentación 2

Presion_mezcla = [0] #Crea un array para almacenar el cálculo de la presión de la variable de la corriente de salida
```

```
flujomasico_mezcla = [0] #Crea un array para almacenar el cálculo del
flujo masico de la variable de la corriente de salida

flujomolar_mezcla = [0] #Crea un array para almacenar el cálculo del
flujo molar de la variable de la corriente de salida

entalpia_mezcla = [0] #Crea un array para almacenar el cálculo de la
entalpia de variable la de la corriente de salida

#Programación del funcionamiento de la operación de mezcla

flujomasico_mezcla[0] = flujomasico_1[0] + flujomasico_2[0] #Calculara
el flujo masico de la corriente de salida como la suma de los flujos
masicos de entrada

flujomolar_mezcla[0] = flujomolar_1[0] + flujomolar_2[0] #Calculara el
flujo molar de la corriente de salida como la suma de los flujos molares
de entrada

entalpiatotal = (flujomasico_1[0] * entalpia_1[0]) + (flujomasico_2[0] *
entalpia_2[0]) #Calculara la entalpia total de cada corriente de
alimentación para luego realizar su suma

entalpia_mezcla[0] = entalpiatotal/flujomasico_mezcla[0] #Calculara la
entalpia especifica de la corriente de salida

flujomolartotal_comp1= (fraccionmolar_1[0] * flujomolar_1[0]) +
(fraccionmolar_2[0] * flujomolar_2[0]) #Calculara el valor de flujo
molar total para tolueno de ambas corrientes

flujomolartotal_comp2= (fraccionmolar_1[1] * flujomolar_1[0]) +
(fraccionmolar_2[1] * flujomolar_2[0]) #Calculara el valor de flujo
molar total para benceno de ambas corrientes

fraccionmolar_mezcla = [flujomolartotal_comp1 /
flujomolar_mezcla[0],flujomolartotal_comp2 / flujomolar_mezcla[0]]
#calculara la fracción molar de tolueno y benceno en la corriente de
salida

Presion_mezcla[0] = (Presion_1[0] + Presion_2[0]) * 0.5 #Calculara la
presión de la corriente de salida como la media entre las corrientes de
entrada

out = oms1 #Fija la corriente de salida como la variable out

out.Clear() #Elimina cualquier dato residual que pueda contener la
corriente de salida

out.SetProp("enthalpy", "Overall", None, "", "mass",entalpia_mezcla)
#Establece la entalpia especifica de la corriente de salida con el valor
calculado previamente

out.SetProp("pressure", "Overall", None, "", "", Presion_mezcla)
#Establece la presión de la corriente de salida con el valor calculado
previamente

out.SetProp("fraction", "Overall", None, "", "mole",
fraccionmolar_mezcla) #Establece la fracción molar de la corriente de
salida con el valor calculado previamente
```

```
out.SetProp("totalFlow", "Overall", None, "", "mass",
flujomasico_mezcla) #Establece el flujo masico de la corriente de salida
con el valor calculado previamente

out.PropertyPackage.DW_CalcEquilibrium(PropertyPackages.FlashSpec.P,Prop
ertyPackages.FlashSpec.H) # a partir del paquete termodinámico cargado
en la simulación el resto de propiedades basándose en presión y
temperatura
```

11.3. Válvula:

Su función será exclusivamente la de provocar la caída de presión que establezca el equilibrio liquido-vapor que se desea separar, así como calcular las variables termodinámicas restantes.

11.3.1. Código Python Válvula:

```
from DWSIM.Thermodynamics import * #Carga en el script, el paquete
termodinámico empleado en la simulación, en este caso el paquete CAPE-
OPEN de COCO

alimentacionmezcla=ims1 #establece la corriente denominada "Corriente de
Mezcla" como la corriente de alimentación

Presion_mezcla = alimentacionmezcla.GetProp("pressure", "Overall", None,
"", "") #Almacena en la variable Presion_mezcla el valor de la presión
de la corriente de alimentación

flujomasico_mezcla = alimentacionmezcla.GetProp("totalFlow" ,"Overall",
None, "", "mass") #Almacena en la variable flujomasico_mezcla el valor
del flujo masico de la corriente de alimentación

fraccionmolar_mezcla = alimentacionmezcla.GetProp("fraction", "Overall",
None, "", "mole") #Almacena en la variable fraccionmolar_mezcla_mezcla
el valor de la fracción molar de la corriente de alimentación

flujomolar_mezcla = alimentacionmezcla.GetProp("totalFlow" ,"Overall",
None, "", "mole") #Almacena en la variable flujomolar_mezcla el valor
del flujo molar de la corriente de alimentación

entalpia_mezcla = alimentacionmezcla.GetProp("enthalpy" ,"Overall",
None, "Mixture", "mass") #Almacena en la variable entalpia_mezcla el
valor de la entalpia de la corriente de alimentación

Presion_separador = [0] #Crea un array para almacenar el cálculo de la
presión de la variable de la corriente de salida

flujomasico_separador = [0] #Crea un array para almacenar el cálculo del
flujo masico de la variable de la corriente de salida
flujomolar_separador = [0] #Crea un array para almacenar el cálculo del
flujo molar de la variable de la corriente de salida

entalpia_separador = [0] #Crea un array para almacenar el cálculo de la
entalpia de variable la de la corriente de salida

#Programación del funcionamiento de la válvula

 #El flujo de salida y el de entrada serán iguales
```

```
flujomasico_separador[0] = flujomasico_mezcla[0]
flujomolar_separador[0] = flujomolar_mezcla[0]

#Cálculo de entalpia de la corriente de salida
entalpiatotal_valv = (flujomasico_mezcla[0] * entalpia_mezcla[0])
entalpia_separador[0] = entalpiatotal_valv/flujomasico_separador[0]

 #La fracción molar de la entrada será igual a la salida
fraccionmolar_separador = fraccionmolar_mezcla

#Se calculará la caída de presión generada en la válvula
Presion_separador[0] = Presion_mezcla[0] * 0.3333333333

out = oms1 #Fija la corriente de salida como la variable out, las
variables son independientes entre los scripts

out.Clear() #Elimina cualquier dato residual que pueda contener la
corriente de salida

out.SetProp("enthalpy", "Overall", None, "", "mass",entalpia_separador)
#Establece la entalpia especifica de la corriente de salida con el valor
calculado previamente

out.SetProp("pressure", "Overall", None, "", "", Presion_separador)
#Establece la presión de la corriente de salida con el valor calculado
previamente

out.SetProp("fraction", "Overall", None, "", "mole",
fraccionmolar_separador) #Establece la fracción molar de la corriente de
salida con el valor calculado previamente

out.SetProp("totalFlow", "Overall", None, "", "mass",
flujomasico_separador) #Establece el flujo masico de la corriente de
salida con el valor calculado previamente

out.PropertyPackage.DW_CalcEquilibrium(PropertyPackages.FlashSpec.P,Prop
ertyPackages.FlashSpec.H) # a partir del paquete termodinámico cargado
en la simulación calculara el resto de propiedades basándose en presión
y temperatura
```

11.4. Separador Flash:

Su función será la de calcular y separar las corrientes de vapor y liquido resultantes de la caída de presión en la válvula. A diferencia de las simulaciones en UNISIM y DWSIM, el refrigerador para enfriar la corriente de vapor hasta su punto de burbuja se ha integrado en el separador flash debido a la forma en la que se ha codificado la operación en Python, esto evita programar una operación adicional y da muestra de la flexibilidad que aporta la programación en Python.

11.4.1. Código Python Separador Flash:

```
from DWSIM.Thermodynamics import * #Carga en el script, el paquete
termodinámico empleado en la simulación, en este caso el paquete CAPE-
OPEN de COCO

alimentacionseparador=ims1 #establece la corriente denominada "Corriente
al Separador" como la corriente de alimentación

Presion_separador = alimentacionseparador.GetProp("pressure", "Overall",
None, "", "") #Almacena en la variable Presion_separador el valor de la
presión de la corriente de alimentación

flujomasico_separador = alimentacionseparador.GetProp("totalFlow"
,"Overall", None, "", "mass") #Almacena en la variable
flujomasico_separador el valor del flujo masico de la corriente de
alimentación

flujomolar_separador = alimentacionseparador.GetProp("totalFlow"
,"Overall", None, "", "mole") #Almacena en la variable
flujomolar_separador el valor del flujo molar de la corriente de
alimentación

entalpia_separador = alimentacionseparador.GetProp("enthalpy"
,"Overall", None, "Mixture", "mass") #Almacena en la variable
entalpia_mezcla el valor de la entalpia de la corriente de alimentación

#A continuación, almacenaran los valores de composición de la fracción
vaporizada de la corriente de alimentación
molfrac_vapB=alimentacionseparador.GetPhase('Vapor').Compounds['Benzene'
].MoleFraction #fracción molar de benceno en la corriente de vapor

molfrac_vapT=alimentacionseparador.GetPhase('Vapor').Compounds['Toluene'
].MoleFraction #fracción molar de tolueno en la corriente de vapor

frac_vap = alimentacionseparador.GetProp("PhaseFraction" ,"Vapor",
None,"", "mole") #fracción molar vaporizada de la corriente de
alimentación

molfrac_cabeza = [molfrac_vapB,molfrac_vapT] #molfrac_cabeza es un array
con los valores de composición molar en la corriente vaporizada

molfrac_liqB=alimentacionseparador.GetPhase('Liquid1').Compounds['Benzen
e'].MoleFraction #fracción molar de benceno en la corriente de liquido

molfrac_liqT=alimentacionseparador.GetPhase('Liquid1').Compounds['Toluen
e'].MoleFraction #fracción molar de tolueno en la corriente de vapor

molfrac_fondo = [molfrac_liqB,molfrac_liqT] #molfrac_cabeza es un array
con los valores de composición molar en la corriente de liquido

#Creamos las variables en la que almacenaremos las propiedades
calculadas de las corrientes de cabeza y fondo respectivamente

Presion_cabeza = [0] #presión
flujomasico_cabeza = [0] #flujo masico
flujomolar_cabeza = [0] #flujo molar
entalpia_cabeza = [0] #entalpia

Presion_fondo = [0] #presión
flujomasico_fondo = [0] #flujo masico
flujomolar_fondo = [0] #flujo molar
```

```
entalpia_fondo = [0] #entalpia
#Programación del funcionamiento de la operación de separación

flujomolar_cabeza[0] = flujomolar_separador[0] * frac_vap[0] #Se calcula
el flujo molar de la corriente de cabeza como el flujo molar de
alimentación por la fracción vaporizada

flujomolar_fondo[0] = flujomolar_separador[0] - flujomolar_cabeza[0] #Se
calcula el flujo de fondo como la diferencia entre el flujo de
alimentación y el flujo de cabeza

flujomasico_cabeza[0] = flujomasico_separador[0] * frac_vap[0] #Se
calcula el flujo molar de la corriente de cabeza como el flujo molar de
alimentación por la fracción vaporizada

flujomasico_fondo[0] = flujomasico_separador[0] - flujomasico_cabeza[0]
#Se calcula el flujo de fondo como la diferencia entre el flujo de
alimentación y el flujo de cabeza

#Cálculo de entalpia de corrientes
entalpiatotal = (flujomasico_separador[0] * entalpia_separador[0])

entalpia_cabeza[0] = (entalpiatotal/flujomasico_cabeza[0]) #entalpia de
corriente de cabeza condensada a fase liquida

entalpia_fondo[0] = entalpiatotal/flujomasico_fondo[0]#entalpia de
corriente liquida de fondo

#Se establece la presión de cabeza y fondo como la presión de
alimentación
Presion_cabeza[0] = Presion_separador[0]
Presion_fondo[0] = Presion_separador[0]

out = oms1 #Fija la variable out como la corriente de salida en cabeza
del separador

out2 = oms2 #Fija la variable out2 como la corriente de salida en fondo
del separador

out.Clear() #Elimina cualquier dato residual que pueda contener la
corriente de cabeza


out.SetProp("pressure", "Overall", None, "", "", Presion_cabeza)
#Establece la presión de la corriente de cabeza con el valor calculado
previamente

out.SetProp("totalFlow", "Overall", None, "", "mole", flujomolar_cabeza)
#Establece el flujo molar de la corriente de cabeza con el valor
calculado previamente

out.SetProp("fraction", "Overall", None, "", "mole", molfrac_cabeza)
#Establece la fracción molar de la corriente de cabeza con el valor
calculado previamente

out.SetProp("enthalpy", "Overall", None, "", "mass",entalpia_cabeza)
#Establece el flujo masico de la corriente de cabeza con el valor
calculado previamente

out2.Clear() #Elimina cualquier dato residual que pueda contener la
corriente de fondo
```

```
out2.SetProp("pressure", "Overall", None, "", "", Presion_fondo)
#Establece la presión de la corriente de fondo con el valor calculado
previamente

out2.SetProp("totalFlow", "Overall", None, "", "mole", flujomolar_fondo)
#Establece el flujo molar de la corriente de fondo con el valor
calculado previamente

out2.SetProp("fraction", "Overall", None, "", "mole", molfrac_fondo)
#Establece la fracción molar de la corriente de fondo con el valor
calculado previamente
out2.SetProp("enthalpy", "Overall", None, "", "mass",entalpia_fondo)
#Establece el flujo masico de la corriente de fondo con el valor
calculado previamente

out.PropertyPackage.DW_CalcEquilibrium(PropertyPackages.FlashSpec.P,Prop
ertyPackages.FlashSpec.H) # a partir del paquete termodinámico cargado
en la simulación calculara el resto de propiedades basándose en presión
y temperatura

out2.PropertyPackage.DW_CalcEquilibrium(PropertyPackages.FlashSpec.P,Pro
pertyPackages.FlashSpec.H) # a partir del paquete termodinámico cargado
en la simulación calculara el resto de propiedades basándose en presión
y temperatura
```

12. Simulación en entorno de DWSIM en COCO 3.5.0.2 (mediante CAPE-OPEN)

Si bien anteriormente hemos rascado la superficie en cuanto a la versatilidad que ofrece la integración del estándar de CAPE-OPEN en la simulación de procesos, utilizando el paquete termodinámico SRK del simulador COCO 3.5.0.2, la implementación de dicho estándar en el núcleo de ambos simuladores (DWSIM y COCO) permiten una fusión completa de las herramientas de ambos simuladores, es por ello que, en este caso de estudio sustituiremos las operaciones básicas que previamente hemos usado (tanto en las implementadas propiamente dichas en DWSIM o programadas en Python) por las empleadas por el simulador COCO 3.5.0.2 lo cual en resumidas cuentas equivaldría a usar este simulador mediante la interfaz de usuario del simulador DWSIM.

Para ello, empezaremos creando una nueva simulación en DWSIM y procederemos a elegir el paquete termodinámico de la simulación de la misma forma que se encuentra descrita en el punto 9 así como en las figuras 10, 11, 12 y 13. De esta forma estableceremos el entorno de simulación con el paquete termodinámico del simulador COCO. A continuación, estableceremos las corrientes de materia como se han realizado en los ejemplos anteriores. Finalmente, solo quedan por establecer las operaciones unitarias,

para ello elegimos en la ventana de la barra de operaciones la pestaña “CAPE-OPEN” lo que nos dará la opción de establecer en la simulación bloques de operación en Cape Open, esto abrirá automáticamente una ventana de configuración en la que se muestran las operaciones en CAPE-OPEN disponibles (Figura 14.), en este caso estableceremos un mezclador, una válvula y un separador flash pertenecientes al simulador COCO y procederemos a realizar las conexiones de las corrientes y dichos bloques (Figura 15.), para ello, basta con seleccionar cada bloque lo que abrirá una pestaña de configuración (Figura 16.) en la que se podrán configurar dichas conexiones así como otros parámetros necesarios para configurar el comportamiento de la operación unitaria.

La configuración de dichas operaciones unitarias será especificada en la “Tabla A” y tendrán como fin el de mantener los mismos criterios que en los casos anteriores.

Mezclador:	
Caída de presión:	0 atm
Válvula:	
Caída de presión entrada:	2 atm
Presión a la salida	Diferencia entre entrada y caída de presión (Resultado: 1atm)
Separador Flash:	
Caída de presión:	0 atm
Aporte de calor (Heat duty)	0 W

Tabla A. Especificaciones de Operaciones unitarias CAPE-OPEN de COCO.

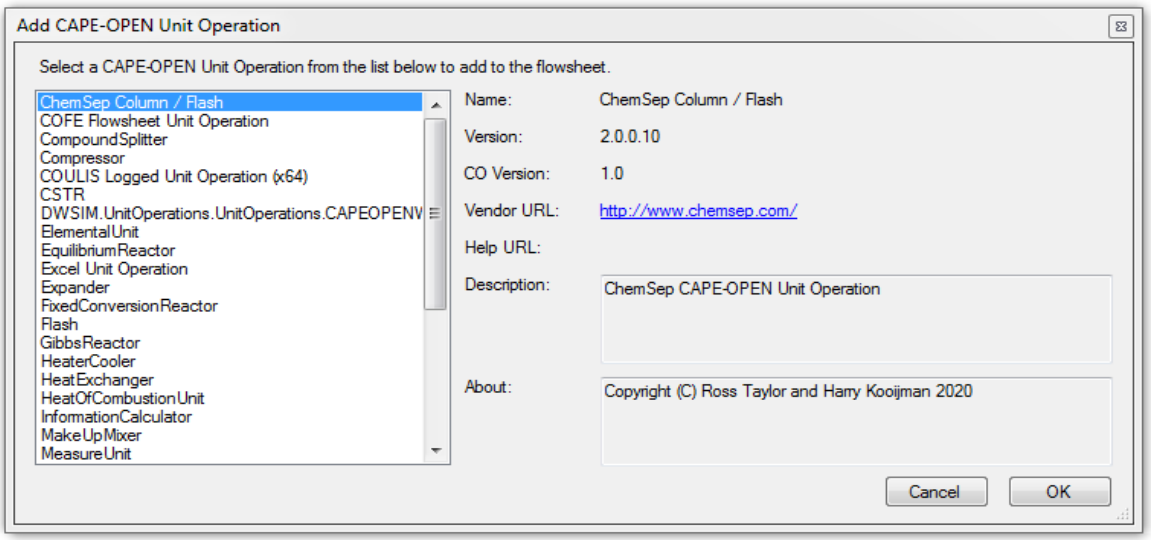

Figura 14. Ventana de configuración de operación unitaria en CAPE-OPEN.

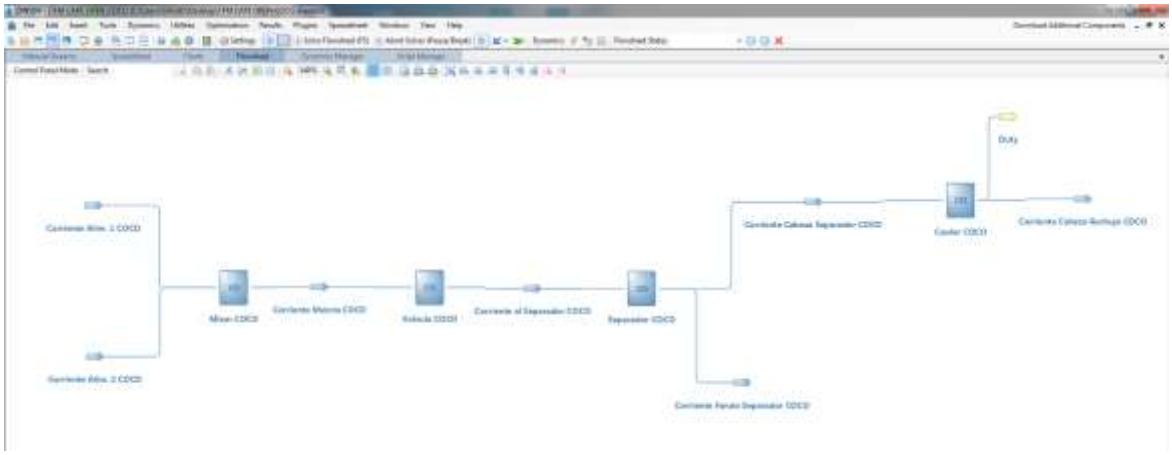

Figura 15. Diagrama del proceso con operaciones CAPE-OPEN de COCO.

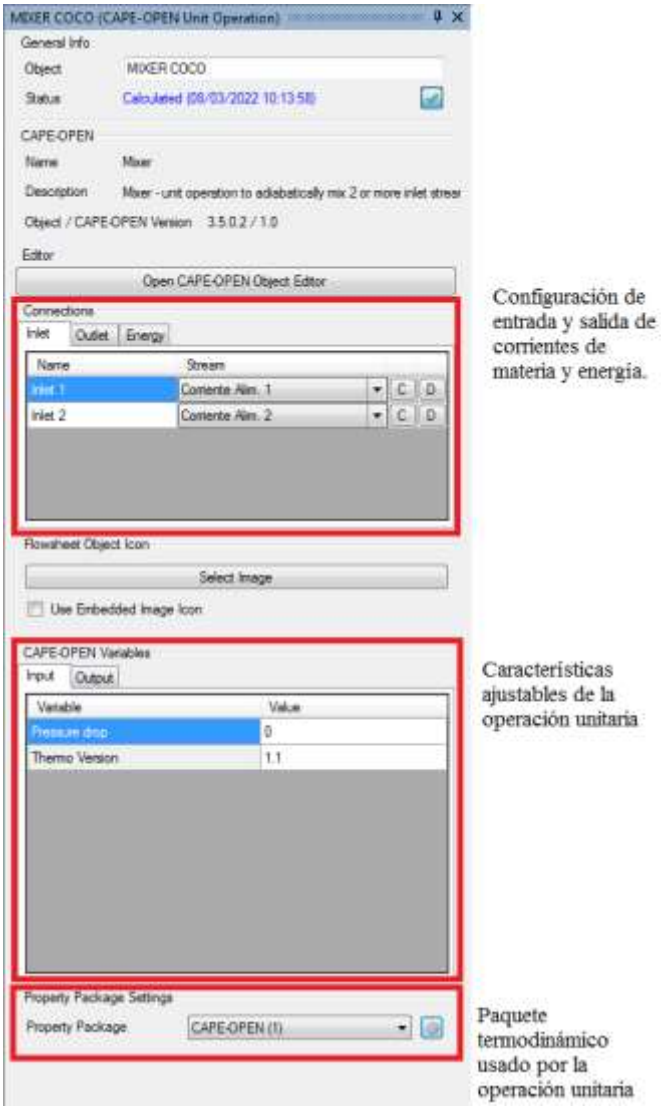

Figura 16. Cuadro de configuración del mezclador CAPE-OPEN de COCO.

De la "Tabla A" (pág. 28) se observa que tanto en el mezclador como en el separador flash nos permite introducir una caída de presión (ya sea para determinar la caída de presión propia del equipo o bien la del sistema en general), esto hace la inclusión de la válvula una operación redundante sin embargo se ha mantenido a fin de mantener un esquema similar a los apartados anteriores, así como de tener acceso a la información de las corrientes intermedias.

13. Simulación en entorno DWSIM mediante operaciones Unitarias en DWSIM, Python y COCO 3.5.0.2. (mediante CAPE-OPEN)

Si bien en los casos de estudio anteriores hemos realizado simulaciones individuales del proceso haciendo uso únicamente de operaciones unitarias procedentes de una sola fuente (ya sea DWSIM, programadas en Python o bien procedentes del simulador COCO), el origen y primera finalidad del estándar de comunicación de CAPE-OPEN ha sido la interconexión de programas de simulación a fin de poder emplear sus puntos fuertes, empleando de forma puntual e independiente las operaciones unitarias y métodos de calculo que poseen dichos simuladores. De esta forma, se puede conformar una simulación de proceso a partir de diversas fuentes cuyos resultados difieran de los obtenidos a aquellos obtenidos si solo se empleasen herramientas de un solo programa de simulación y más importante aún, que los resultados obtenidos sean los más próximos a la realidad del proceso estudiado.

Puesto que el ejemplo estudiado solo posee entre 3-4 operaciones unitarias (Mezclador, Válvula, Separador y Refrigerador) llevaremos a cabo las tres permutaciones posibles de simulador/operación unitaria a fin de observar las diferencias en cuanto a resultados obtenidos. (Nota: el paquete termodinámico empleado para las tres simulaciones es SRK proveniente del simulador DWSIM, además, la operación del refrigerador se realizará empleando la misma fuente que el empleado para simular el separador).

A continuación, se mostrará como se han configurado las distintas permutaciones realizadas, así como la organización y procedencia del método de cálculo para las operaciones unitarias que componen cada permutación:

		OPERACIONES UNITARIAS		
		MIXER	VALVULA	SEPARADOR/COOLER
SIMULACIÓN	Permutación A	COCO 3.5.0.2	PYTHON	DWSIM
	Permutación B	DWSIM	COCO 3.5.0.2	PYTHON
	Permutación C	PYTHON	DWSIM	COCO 3.5.0.2

Tabla B. Configuración de las simulaciones en entorno DWSIM mediante operaciones Unitarias en DWSIM, Python y COCO 3.5.0.2. (mediante CAPE-OPEN)

Configurando las distintas permutaciones (Figura 17. Figura 18. Figura 19.) según se muestra en la Tabla B. podremos estudiar el impacto que tienen las diferentes operaciones unitarias según su procedencia en el resultado de las corrientes en la simulación no solo entre las distintas permutaciones estudiadas, sino también compararlas con los casos previamente simulados en UNISIM, DWSIM, COCO y Python.

Figura 17. Permutación A de simulación COCO, DWSIM y Python.

Figura 18. Permutación B de simulación COCO, DWSIM y Python.

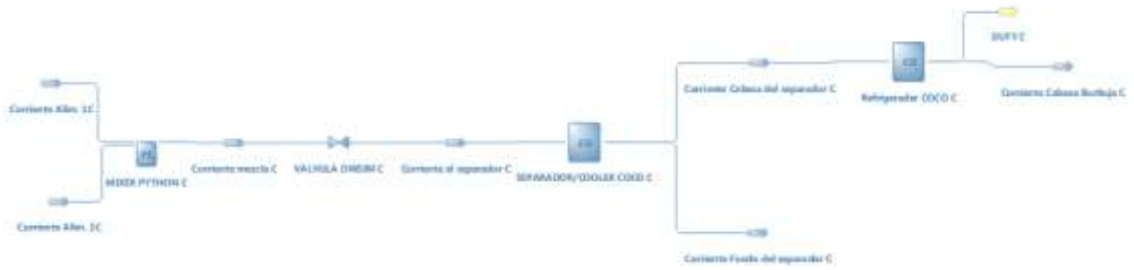

Figura 19. Permutación C de simulación COCO, DWSIM y Python.

14. Resultados de simulación, análisis y comparación.

A continuación, se presentarán los resultados obtenidos de los parámetros más relevantes de cada corriente obtenidos por los distintos simuladores:

RESULTADOS:

UNISIM							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	137,9	131,5	134,6	93,73	93,73	93,73	87,99
Presión (Atm)	3	3	3	1	1	1	1
Fracción Vap. (%)	0	0	0,001	0,2193	1	0	0
Fracción Tolueno Total (%)	60	40	50	50	33,57	54,61	33,57
Fracción Benceno Total (%)	40	60	50	50	66,42	45,38	66,42
Flujo molar (Kmol/h)	50	50	100	100	21,93	78,07	21,93
Duty Refrigeración (KW)	-	-	-	-	-	-	198,9

Tabla 1. Resultados de UNISIM R390.1

DWSIM							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	138,035	131,435	134,63	93,7741	93,7741	93,7741	87,6931
Presión (Atm)	3	3	3	1	1	1	1
Fracción Vap. (%)	0	0	0,00103176	0,22083	1	0	0
Fracción Tolueno Total (%)	60	40	50	50	33,1827	54,7663	33,1827
Fracción Benceno Total (%)	40	60	50	50	66,8173	45,2337	66,8173
Flujo molar (Kmol/h)	50	50	100	100	22,083	77,917	22,083
Duty Refrigeración (KW)	-	-	-	-	-	-	200,184

Tabla 2. Resultados DWSIM 6.5.0.0.

DWSIM con paquete termodinámico de COCO y operaciones programas en Python.							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	138,032	131,434	134,623	93,7643	-	93,7643	87,6853
Presión (Atm)	3	3	3	1	-	1	1
Fracción Vap. (%)	0	0	0	0,21945	-	0	0
Fracción Tolueno Total (%)	60	40	50	50	-	54,736	33,1548
Fracción Benceno Total (%)	40	60	50	50	-	45,264	66,8451
Flujo molar (Kmol/h)	50	50	100	100	-	78,055	21,945

Tabla 3. Resultados DWSIM con paquete termodinámico de COCO y operaciones programas en Python.

DWSIM con paquete termodinámico de COCO y operaciones de COCO.							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	138,03	131,435	133,065	93,71	93,71	93,73	87,64
Presión (Atm)	3	3	3	1	1	1	1
Fracción Vap. (%)	0	0	0	0,2114	1	0	0
Fracción Tolueno Total (%)	60	40	50	50	32,99	54,56	32,99
Fracción Benceno Total (%)	40	60	50	50	67,01	45,44	67,01
Flujo molar (Kmol/h)	50	50	100	100	21,14	78,86	21,14
Duty Refrigeración (KW)	-	-	-	-	-	-	191,616

Tabla 4. Resultados empleando CAPE-OPEN para simular COCO en entorno de DWSIM

DWSIM con paquete termodinámico DWSIM - PERMUTACION A							
Corriente	Corriente Alim. 1A	Corriente Alim. 2A	Corriente Mezcla A	Corriente al Separador A	Corriente de Cabeza Sep. A	Corriente de Fondo Sep. A	Corriente de Cabeza Burbuja A
Temperatura (ºC)	137,79	131,24	134,41	93,55	93,55	93,55	87,52
Presión (Atm)	3	3	3	1	1	1	1
Fracción Vap. (%)	0	0	0	0,219937	1	0	0
Fracción Tolueno Total (%)	60	40	50	50	33,09	54,77	33,09
Fracción Benceno Total (%)	40	60	50	50	66,91	45,23	66,91
Flujo molar (Kmol/h)	50	50	100	100	21,99	78,01	21,99
Duty Refrigeración (KW)	-	-	-	-	-	-	199,167

Tabla 5. Resultados Permutación A, Combinación DWSIM, COCO y Python.

DWSIM con paquete termodinámico DWSIM - PERMUTACION B							
Corriente	Corriente Alim. 1 B	Corriente Alim. 2 B	Corriente Mezcla B	Corriente al Separador B	Corriente de Cabeza Sep. B	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	137,79	131,24	134,41	93,56	-	93,91	87,53
Presión (Atm)	3	3	3	1	-	1	1
Fracción Vap. (%)	0	0	0,001	0,2208	-	0	0
Fracción Tolueno Total (%)	60	40	50	50	-	54,78	33,11
Fracción Benceno Total (%)	40	60	50	50	-	45,21	66,89
Flujo molar (Kmol/h)	50	50	100	100	-	77,91	22,09
Duty Refrigeración (KW)	-	-	-	-	-	-	-

Tabla 6. Resultados Permutación B, Combinación DWSIM, COCO y Python.

DWSIM con paquete termodinámico DWSIM - PERMUTACION C							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	137,79	131,24	134,41	93,56	93,56	93,56	87,53
Presión (Atm)	3	3	3	1	1	1	1
Fracción Vap. (%)	0	0	0,001	0,22088	1	0	0
Fracción Tolueno Total (%)	60	40	50	50	33,11	54,79	33,11
Fracción Benceno Total (%)	40	60	50	50	66,89	45,21	66,89
Flujo molar (Kmol/h)	50	50	100	100	22,088	77,911	22,088
Duty Refrigeración (KW)	-	-	-	-	-	-	200,031

Tabla 7. Resultados Permutación C, Combinación DWSIM, COCO y Python.

COMPARACION:

A partir de estos resultados obtenidos en las distintas simulaciones, procederemos a calcular la diferencia entre estos y los obtenidos por el simulador UNISIM, tomamos dicho marco de referencia puesto que dicho simulador es uno de los más extendidos en el mercado y, por lo tanto, la fiabilidad de sus resultados y sus resultados propiamente dichos serían los que se manejarían y aceptarían normalmente en la industria como verídicos. (La fórmula que se empleará para el cálculo de las diferencias entre los resultados de simulación será *RESULTADO UNISIM – RESULTADO SIMULACION*, esto significa que valores negativos corresponde a valores superiores de resultado de la simulación con respecto a los obtenidos con UNISIM).

Diferencia UNISIM-DWSIM							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	-0,135	0,065	-0,03	-0,0441	-0,0441	-0,0441	0,2969
Presión (Atm)	0	0	0	0	0	0	0
Fracción Vap. (%)	0	0	-0,00003	-0,00153	0	0	0
Fracción Tolueno Total (%)	0	0	0	0	0,3873	-0,1563	0,3873
Fracción Benceno Total (%)	0	0	0	0	-0,3973	0,1463	-0,3973
Flujo molar (Kmol/h)	0	0	0	0	-0,153	0,153	-0,153
Duty Refrigeración (KW)	-	-	-	-	-	-	-1,284

Tabla 8. Diferencia entre los resultados de UNISIM y DWSIM.

Diferencia UNISIM-DWSIM con paquete termodinámico de COCO y operaciones en PYTHON.							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	-0,13	0,065	-0,023	-0,0343	-	-0,0343	0,3047
Presión (Atm)	0	0	0	0	-	0	0
Fracción Vap. (%)	0	0	0,001	-0,00015	-	0	0
Fracción Tolueno Total (%)	0	0	0	0	-	-0,126	0,4152
Fracción Benceno Total (%)	0	0	0	0	-	0,116	-0,4251
Flujo molar (Kmol/h)	0	0	0	0	-	0,015	-0,015
Duty Refrigeración (KW)	-	-	-	-	-	-	-

Tabla 9. Diferencia entre los resultados de DWSIM y DWSIM empleando paquetes termodinámicos de COCO y operaciones programas en Python.

Diferencia UNISIM - DWSIM con paquete termodinámico de COCO y operaciones de COCO.							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	-0,13	0,065	1,535	0,02	0,02	0	0,35
Presión (Atm)	0	0	0	0	0	0	0
Fracción Vap. (%)	0	0	0,001	0,0079	0	0	0
Fracción Tolueno Total (%)	0	0	0	0	0,58	0,05	0,58
Fracción Benceno Total (%)	0	0	0	0	-0,59	-0,06	-0,59
Flujo molar (Kmol/h)	0	0	0	0	0,79	-0,79	0,79
Duty Refrigeración (KW)	-	-	-	-	-	-	7,284

Tabla 10. Diferencia entre los resultados de DWSIM y COCO.

Diferencia UNISIM - DWSIM con paquete termodinámico DWSIM - PERMUTACION A							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	0,11	0,26	0,19	0,18	0,18	0,18	0,47
Presión (Atm)	0	0	0	0	0	0	0
Fracción Vap. (%)	0	0	0,001	-0,000637	0	0	0
Fracción Tolueno Total (%)	0	0	0	0	0,48	-0,16	0,48
Fracción Benceno Total (%)	0	0	0	0	-0,49	0,15	-0,49
Flujo molar (Kmol/h)	0	0	0	0	-0,06	0,06	-0,06
Duty Refrigeración (KW)	-	-	-	-	-	-	-0,267

Tabla 11. Diferencia entre los resultados de DWSIM y Permutación A.

Diferencia UNISIM - DWSIM con paquete termodinámico DWSIM - PERMUTACION B							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	0,11	0,26	0,19	0,17	-	-0,18	0,46
Presión (Atm)	0	0	0	0	-	0	0
Fracción Vap. (%)	0	0	0	-0,0015	-	0	0
Fracción Tolueno Total (%)	0	0	0	0	-	-0,17	0,46
Fracción Benceno Total (%)	0	0	0	0	-	0,17	-0,47
Flujo molar (Kmol/h)	0	0	0	0	-	0,16	-0,16
Duty Refrigeración (KW)	-	-	-	-	-	-	-

Tabla 12. Diferencia entre los resultados de DWSIM y Permutación B.

Diferencia UNISIM - DWSIM con paquete termodinámico DWSIM - PERMUTACION C							
Corriente	Corriente Alim. 1	Corriente Alim. 2	Corriente Mezcla	Corriente al Separador	Corriente de Cabeza Sep.	Corriente de Fondo Sep.	Corriente de Cabeza Burbuja
Temperatura (ºC)	0,11	0,26	0,19	0,17	0,17	0,17	0,46
Presión (Atm)	0	0	0	0	0	0	0
Fracción Vap. (%)	0	0	0	-0,0015	0	0	0
Fracción Tolueno Total (%)	0	0	0	0	0,46	-0,18	0,46
Fracción Benceno Total (%)	0	0	0	0	-0,47	0,17	-0,47
Flujo molar (Kmol/h)	0	0	0	0	-0,158	0,158	-0,158
Duty Refrigeración (KW)	-	-	-	-	-	-	-1,131

Tabla 13. Diferencia entre los resultados de DWSIM y Permutación C.

Se ha omitido de las tablas las entalpias de las corrientes pues cada simulador (UNISIM, DWSIM y COCO) toman sus valores arbitrariamente de tablas de estado diferentes por lo que los resultados variaran según el paquete termodinámico y programa empleado. Esto sin embargo no tiene influencia en los resultados de los balances energéticos.

ANÁLISIS:[10]

De forma general, si se observan las distintas tablas, las diferencias entre las distintas variables apenas llega al 1%, siendo en muchos casos la diferencia tan insignificante que puede ser despreciable a efectos del resultado obtenido (siendo aceptable como márgenes de error o cuya cifra significativa queda fuera del error aceptado, esto es especialmente notable en el cálculo de fracciones de vapor contenidas en la corriente al separador). Sin embargo, esta diferencia puede también ser el resultado entre la interacción del método y paquete termodinámico empleado, así como de las operaciones unitarias y como estas se encuentran programadas para obtener un resultado. Es por ello que el siguiente análisis se llevara a cabo mediante ese enfoque, comparando los resultados con los paquetes termodinámicos y las características de las operaciones unitarias empleados.

Corriente Alim.1:

Se puede observar que la diferencia de temperatura existente entre las tablas 8, 9 y 10 (UNISIM frente a DWSIM, Python y COCO respectivamente) es igual en los casos de las tablas 9 y 10 (PYTHON y COCO), dichos casos emplean el paquete termodinámico de COCO lo cual explica la similitud en los resultados, por otra parte si comparamos estos resultados con los expuestos en la tabla 8 (UNISIM frente a DWSIM) se observa una diferencia de 0,005ºC, siendo la única diferencia entre ambos casos el uso del paquete termodinámico de DWSIM (en el caso de UNISIM frente a DWSIM) y el paquete termodinámico de COCO.

En el caso de las tablas 11, 12 y 13 (UNISIM frente a las Permutaciones del caso de estudio A, B, C) no existen ninguna diferencia ya que los tres casos emplean el mismo paquete termodinámico y por consiguiente su cálculo ha sido igual en los tres casos.

El resto de variables de esta corriente están definidas por los parámetros del caso de estudio por lo que no existen diferencias entre los casos estudiados.

Corriente Alim. 2:

En esta corriente se observan similitudes con la corriente anteriormente comentada, los casos expuestos en las tablas 8,9 y 10 presentan resultados iguales a pesar de que como se comentó anteriormente el caso de estudio de UNISIM frente a DWSIM emplea un paquete termodinámico distinto al de los empleados en los otros dos casos estudiados.

En el caso de las tablas 11, 12 y 13 no se observan diferencia por los mismos motivos expuestos en la corriente anteriormente analizada.

Corriente de Mezcla:

A partir de esta corriente se empiezan a introducir las diferentes operaciones básicas. En primer lugar, observamos que los resultados podrías agruparse en tres grupos, el primero se encuentra formado por las simulaciones realizadas en DWSIM y Python las cuales muestran una diferencia de temperatura con respecto a UNISIM muy similar (-0.03 y -0.023) así como una diferencia de fracción de vapor despreciable (-0.00003 y 0.001),

debemos recordar que para ambos casos no solo la operación de mezclado se ha llevado de forma distinta (en DWSIM empleando la operación de mezcla nativa del simular y en Python (Tabla 8. y Tabla 9.) la operación de mezcla programada independientemente) sino que los paquetes termodinámicos empleados también han sido distintos (en DWSIM se ha empleado el paquete implementado en dicho simulador mientras que en Python se ha empleado el integrado en COCO). El segundo grupo corresponde a la simulación llevada a cabo en COCO (Tabla 10), en dicha simulación se observa la mayor diferencia de temperatura de entre los demás casos estudiados (1.535 ºC) sin embargo la diferencia en fracción de vapor es igual a los obtenidos en la simulación en Python (Tabla 9.) y la permutación A (Tabla 11.), la cual empieza con la operación de mezcla del simulador COCO, pero el paquete termodinámico de DWSIM. Finalmente se encuentra el grupo de las tres permutaciones realizadas (Tabla 11., Tabla 12. y Tabla 13.) en las cuales se observa una temperatura igual en los tres casos (0.19 ºC) así como una diferencia en la fracción de vapor idéntica a la obtenida en UNISIM para las Permutaciones B y C (Tabla 12. y Tabla 13.).

Corriente al Separador:

Esta es la corriente entre la salida de la válvula responsable de la caída de presión y el separador, destacan los resultados obtenidos en las permutaciones B y C (Tabla 12. y Tabla 13. respectivamente) pues a pesar de emplear operaciones procedentes de diferentes simuladores se han obtenidos resultados iguales. Por otro lado, en la permutación A (Tabla 11.) la diferencia de fracción de vapor es la más similar a la obtenida en UNISIM de todos los casos estudiados lo cual contrasta con los resultados de la diferencia de temperatura ya que esta permutación posee la mayor diferencia con respecto al resto de casos estudiados (0.18 ºC). Por otra parte, los resultados obtenidos en DWSIM y Python (Tabla 8. Y Tabla 9. respectivamente) muestran un comportamiento similar a los obtenidos en la corriente de mezcla, ya que se mantienen una diferencia de temperatura entre ambas de en torno a 0.01 ºC y una fracción de vapor prácticamente igual entre ambas simulaciones.

Finalmente, la simulación llevada a cabo en COCO (Tabla 10.) tiene la mayor diferencia en cuanto a fracción de vapor se refiere mientras que la diferencia de temperatura es la menor entre los casos de estudio con respecto a la simulación de UNISIM. Sin embargo, comparados con los resultados obtenidos en la corriente anterior, se observa una drástica reducción en la diferencia de temperatura mientras que la diferencia de fracción de

vapor se ha incrementado muy ligeramente.

Corriente de cabeza del separador:

Esta corriente corresponde a la salida en fase gaseosa del separador por lo que esta corriente no aparece reflejada en la simulación de Python y la permutación B puesto que la forma en la que se encuentra codificada dicho separador da como resultado la salida condensada de la cabeza del separador.

Asimismo, las permutaciones A y C (Tabla 11. Y Tabla 13) muestran la misma tendencia que en las corrientes anteriores, valores de diferencia de temperatura similares (0.18ºC y 0.17ºC respectivamente), una fracción de tolueno/benceno similar entre ambas permutaciones, pero con composiciones de benceno/tolueno ligeramente inferiores en el caso de la permutación C lo cual puede deberse a la ligera diferencia en el flujo molar de dicha corriente, así como de sutiles diferencias en las propias operaciones unitarias que emplean dichas simulaciones.

Por otra parte, la simulación en DWSIM (Tabla 8.) mantiene el mismo valor de diferencia de temperatura con la corriente de entrada del separador y cuyo valor resultante (-0.0441 ºC) afecta a los valores de equilibrio y composición del gas resultante de la separación, esto da como resultado la menor diferencia en cuanto a fracción de benceno/tolueno en la mezcla con respecto a UNISIM en comparación con el resto de simulaciones. En cuanto al flujo molar, la diferencia entre DWSIM y UNISIM solo se encuentra superada por la simulación de la permutación C.

Finalmente, la simulación realizada en COCO (Tabla 10.) mantiene la tendencia presente en el resto de corrientes, la menor diferencia de temperatura de entre todas las simulaciones (0.02ºC) y la mayor diferencia de flujo molar y composición benceno/tolueno de entre todas las simulaciones realizadas.

Corriente de Fondo del separador:

La corriente de Fondo del separador está compuesta por la salida liquida del fondo del separador de fase. De los parámetros analizados, las diferencias en los resultados del flujo molar se deben al principio de conservación de materia y a la existencia de diferencias en los resultados obtenidos en la corriente de salida de cabeza del separador

(Entrada = Salida puesto que no existe acumulación) por lo tanto la cantidad de más que posea la corriente de cabeza será lo que le falte a la corriente de fondo.

Destaca que, a pesar de emplear operaciones básicas de distinta procedencia, en las permutaciones A, B y C, los resultados obtenidos son prácticamente iguales entre ellas habiendo una ligera variación en cuanto a diferencia de flujo molar y composición de benceno /tolueno. Por otro lado, en los casos de la diferencia entre UNISIM y DWSIM, así como de la diferencia entre UNISIM y Python observamos que se mantiene la tendencia mostrada en las corrientes anteriores pues los resultados son prácticamente iguales entre ambos casos. Finalmente, la simulación realizada en COCO (Tabla 10.) cambia la tendencia presente en el resto de corrientes, ya que no existe diferencia de temperatura con respecto a UNISIM y posee la menor diferencia de flujo molar y composición benceno/tolueno de entre todas las simulaciones realizadas.

Corriente de Cabeza Burbuja:

En esta corriente los parámetros de flujo y composición de benceno/tolueno son iguales a los expuestos en la corriente de cabeza del separador, tan salo varia la temperatura y el trabajo de refrigeración para llevar la corriente al punto de burbuja.

En el caso de la simulación en Python y de la permutación B, la manera en la que se encuentra codificado el separador fuerza la corriente de salida en cabeza directamente al punto de burbuja por lo que no hay datos del trabajo de refrigeración de la corriente.

Podemos observar que en esta corriente se siguen manteniendo las tendencias observadas en otras corrientes anteriormente analizadas, en primer lugar, las simulaciones de la permutación A, permutación B y la permutación C mantienen una diferencia de temperatura simular (0.47-0.46 ºC) sin embargo, si bien dichas simulaciones emplean el mismo paquete termodinámico, entre las permutaciones A y C existe una diferencia de entorno a 1kW en el trabajo de refrigeración lo cual refleja el efecto del flujo molar en dicha corriente ya que en la simulación en DWSIM (la cual también emplea el paquete termodinámico de DWSIM), si bien la diferencia de temperatura es algo menor que en las obtenidas en las distintas permutaciones, los resultados obtenidos en el trabajo de refrigeración se asemejan bastante al obtenido en la permutación C teniendo ambas un flujo molar muy similar. Como contrapartida, podemos observar que, en el caso de la simulación

en COCO, la cual tiene la mayor diferencia en cuanto a flujo molar se refiere, también tiene la mayor diferencia en el trabajo de refrigeración.

Finalmente, si bien existe consistencia y semejanzas entre algunos de los casos de estudio realizados, observando cada una individualmente y su diferencia con los resultados obtenidos en UNISIM destaca que las mayores diferencias se encuentren en la fracción de tolueno/benceno de la salida de vapor del separador lo cual indica una mayor sensibilidad a la hora de calcular los resultados de la fase gaseosa de las corrientes de materia entre UNISIM y su contra parte en CAPE-OPEN.

15. Conclusiones.

Como bien hemos observado en el punto anterior, los resultados obtenidos se pueden clasificar en tres grupos los cuales muestran semejanzas o claras diferencias entre ellos. Estos grupos son:

1º Grupo: Se encuentra formado por las simulaciones realizadas en DWSIM y Python (con paquete termodinámico de COCO), ambas simulaciones dan resultados consistentes y muy próximos a los obtenidos en un software comercial como es UNISIM.

2º Grupo: Este grupo lo agrupa la simulación de COCO, destaca que los resultados obtenidos en esta simulación abarcan ambos extremos de la escala, en ciertos parámetros de algunas corrientes, esta simulación obtiene resultados más próximos a UNISIM que el grupo 1, sin embargo, en otros parámetros de esas mismas corrientes, la diferencia entre esta simulación y UNISIM son las de mayor valor. Esto, junto con las diferencias obtenidas en la simulación en Python (que también emplea el paquete termodinámico de COCO) hacen ver que existen diferencias sutiles a la hora de calcular las propiedades de las corrientes en las operaciones unitarias entre ambas simulaciones.

3º Grupo: Este grupo lo conforman las distintas permutaciones de operaciones unitarias procedentes de distintos softwares (DWSIM, COCO y Python) los cuales tienen un comportamiento intermedio entre el 1º grupo y el 2º ya que, si bien las diferencias obtenidas en dichas permutaciones se alejan más de los resultados obtenidos en UNISIM por las simulaciones en DWSIM y Python, tampoco adquieren valores tan altos como lo hace la simulación en COCO en ciertos parámetros.

Finalmente se puede deducir en los resultados obtenidos, que una de las mayores influencias en el resultado obtenido en las distintas corrientes de la simulación es el paquete termodinámico empleado así como el simulador de procedencia del mismo, además, el estado de agregación de la corriente también parece tener influencia en los resultados obtenidos ya que si bien las distintas simulaciones realizadas tienen cierta semejanza entre sí, existe una clara y pronunciada diferencia entre los resultados obtenidos en fracciones de composición de tolueno/benceno en corrientes liquidas y más claramente en corrientes gaseosas con respecto a los resultados obtenidos en UNISIM. Por otra parte, si bien no se debe olvidar la influencia de cómo funcionan las operaciones básicas en los distintos simuladores (como bien se muestra entre las diferencias en las simulaciones de Python y COCO) queda de manifiesto la utilidad de la integración CAPE-OPEN en programas de simulación ya que esto permite flexibilizar y aprovechar al máximo los puntos fuertes de cada uno de ellos según la operación llevada a cabo y las características y elementos que conforman las corrientes de dicho proceso. Al mismo tiempo, esta interconectividad potencia la utilidad del software de simulación que si bien en la actualidad el mercado se encuentra dominado por simuladores bajo licencia como UNISIM y HYSYS, otras opciones disponibles de forma libre pueden suplir las necesidades del usuario medio, así como dar las herramientas y medios a los usuarios más avanzados para competir de forma directa con dichos softwares de pago, obteniendo resultados de simulación que nada tienen que envidiar a los software bajo licencia. Esto se base en dos aspectos cruciales, el primero la apertura y transparencia del funcionamiento del software al usuario, generalmente debido a la naturaleza open-source de estos así como una agresiva adopción del uso de CAPE-OPEN lo que ha potenciado significativamente la utilidad de estos programas pues ya no se requiere del aprendizaje o uso de distintos programas para distintas finalidades, tan solo el conocimiento de los puntos fuertes y débiles de cada simulador y en el caso de ser necesario, llamar a aquellos paquetes externos que sean necesarios con el fin de aprovechar sus ventajas y cubrir las necesidades del usuario.

Paralelamente, el empleo de Python como medio para la programación de simulaciones u operaciones unitarias abre un abanico enorme al usuario pues este puede programar su comportamiento como mejor crea conveniente, eliminando cualquier barrera existente y manipulando todos los aspectos de la simulación directamente. Sin embargo, para un empleo correcto, el usuario requerirá de un conocimiento más detallado tanto en el proceso llevado a cabo como en el lenguaje de programación y como se organiza dicho

lenguaje dentro del programa de simulación (llamada de variables, comandos, etc.) lo que sin duda requerirá de una inversión mayor de tiempo a fin de dominar esta metodología de trabajo. Además, no debemos olvidar que estas operaciones programadas en Python, a diferencia de las implementadas en los programas de simulación, obviaran características de la operación según el usuario crea o decida programar dichas operaciones, un ejemplo de ello, es lo ocurrido en el mezclador realizado para este ejemplo, tanto en DWSIM como en UNISIM, la mezcla resultante tiene una pequeña fracción vaporizada debido al cambio de temperatura ocurrido en la mezcla y a que ambas corrientes entran al mezclador en su punto de burbuja hecho que el script programado no tiene en cuenta. Por otra parte, la ventaja de programar las operaciones mediante Python es la de manejar directamente los aspectos de la operación que se está programando, así como de las variables obtenidas ya que si se realiza correctamente se pueden omitir completamente otras operaciones y equipos que, por ejemplo, no son caso de estudio ya que obtenemos el resultado directamente del cálculo realizado por el programa. Este es el caso del refrigerador en la corriente de cabeza de nuestro ejemplo, el cual se encuentra integrado directamente en el separador. Esto, si se tienen los conocimientos necesarios, simplifican la simulación y el trabajo de programación que se debe llevar a cabo.

16. Conclusions.

As we observe in point 13., the results we obtain from the simulations can be organize in three groups, in which they share clear differences and similarities. These groups are:

1º Group: Form by the simulations in DWSIM and Python (with thermodynamic property packages from COCO), both kind of simulations have consistence and very close results to those obtain in a commercial software like UNISIM.

2º Grupo: This group is form by the simulation COCO, from the results obtain in this simulation we observe it covers both ends of the scales, in some parameters of some material streams, the results of this simulation are closer than any other to UNISIM while in other parameters of the same material streams it has the highest difference to UNISIM. This alongside the Python simulation (that is also use the COCO thermodynamic package) show the existence of differences in how the unitary operations are code.

3º Group: This group is form by the different variations of unitary operations from different software (DWSIM, COCO and Python) which all three of them have an intermediate behaviour between groups 1º and 2º, this is because, even if the difference obtain from UNISIM are bigger than the ones obtain with DWSIM and Python, they do not reach values as high as COCO

Finally, we can deduct from the results we obtain that one of the biggest influences in the result of the material streams are the thermodynamic package and the source of it, furthermore the state of aggregation of the stream seems to also have an impact specially in the benzene / toluene composition, with a clear difference in liquid streams and a more pronounce difference in gas streams relative to the results obtain in UNISIM. On the other hand, we cannot forget the existence of the influence on how the unitary operation of the simulation is coded between the different simulation (as seen between the differences between COCO and Python) this show the value of integration of CAPE-OPEN in simulation programs because it allows for more flexibility and boost the usage of individual strong points in the simulators according to the operations of material steam that have to be calculated. At the same time, interconnectivity enhance the utility of simulation software that even with the dominance of UNISIM and HYSYS in the licensed market, other available to the open public options can compete and cover the needs of the average user and becoming a tool that can directly compete with this licensed software in the hands of the more skilled user. This is caused by two crucial aspects, the first one the transparency and open nature of the software to the user, generally do to the open-source nature of said software and second due to the aggressive adoption of CAPE-OPEN which increase the versatility to the user since they do not need to learn each individual program but only in what they are good at in order to implement them.

Lastly, the use of Python as means to code entire simulations or specific unitary operations allows the user to eliminate any barrier in the program and manipulate every aspect of the simulation. Nevertheless, for a correct use, the user must have a detail knowledge of the process as well as the programming language and how it Works inside the simulator (variable calls, commandos, …) this, without a doubt, requires a greater investment of time and effort in order to master the work methodology. Also, we cannot forget the operations program in Python, unlike the already implemented in the simulators, will omit aspects not coded or omitted by the programmers, an example of this occurs in the

mixer coded in the Project, both in UNISIM and DWSIM, the resulting mix have a little vapor fraction while in the one coded in Python this fraction does not exist. On the other hand, the greatest advantage of programming operation using Python is managing the different aspects of the simulation, allowing to complete omitting operations if they are not subject to study as show in the example created where the cooler was integrated in the separator unit. This can simplify the simulation work and the simulation itself.

17. ANEXO: Enlace de descarga de simulaciones.

Enlace de GOOGLE DRIVE para la descarga de las simulaciones realizadas. Disponible en la URL:

https://drive.google.com/drive/folders/1fc-tlYWVknCp2E4Dk8HMKKJV3EEf4bgR?usp=sharing

18. Bibliografía

[1] Pagina web AC-TRAIN Science and Ingineering Training, “Historia de la simulación de procesos”. Disponible en la URL:

https://www.ac-train.com/historia-de-la-simulacion-de-procesos/

[2] Página web CO-LaN, “About CAPE-OPEN” Historia y evolución del CAPE-OPEN. Disponible en la URL:

https://www.colan.org/general-information-on-co-lan/

[3] Página web oficial DWSIM, “About DWSIM” Historia y evolución. Disponible en la URL:

https://DWSIM.org/index.php/about/

[4] Canal oficial DWSIM en Youtube, DWSIM Simulator, Información, videos y tutoriales de uso. Disponible en la URL:

https://www.youtube.com/channel/UCzzBQrycKoN5XbCeLV12y3Q

[5] K.A. Coker “Distillation,” in Ludwing’s Applied Process Design for Chemical and

Petrochemical Plants, 4th ed., vol II, Burlington, MA, Gulf Professional Publishing, 2010, pp 1-268.

[6] Goyal, Maneet, Kadam, Pratik & Pandya, Anand. (2015). "Design of Distillation Column for Separating a Mixture of Benzene and Toluene". 10.13140/RG.2.1.1286.0406. Disponible en la URL:

https://www.researchgate.net/publication/305280465_Design_of_Distillation_Column_for_Separating_a_Mixture_of_Benzene_and_Toluene

[7] K. V. Narayanan, "Properties of Solutions," in A Textbook of Chemical Engineering Thermodynamics, New Delhi, Prentice Hall of India Private Limited, 2001, p. 273.

[8] DWSIM CLASS LIBRARY DOCUMENTATION, Documentación y ayudas para la programación de código Python en el simulador open-source DWSIM. Disponible en la URL:

https://DWSIM.org/api_help/html/R_Project_DWSIM_Class_Library_Documentation.htm

[9] Pág. web SOURCEFORGE, subforo DWSIM – OPEN-SOURCE PROCESS SIMULATOR, Foro de ayuda Python Scripting. Disponible en la URL:

https://sourceforge.net/p/DWSIM/discussion/scripting/

[10] Cañizares Gutiérrez, Carlos Daniel, "Estudio comparativo de cálculo de propiedades entre programas de simulación. El caso de UNISIM y DWSIM" Trabajo de Fin de grado. Curso 2020-2021, Universidad de la Laguna . Disponible en la URL:

https://riull.ull.es/xmlui/handle/915/24753