Modelado y simulación de procesos controlados para el curso de introducción al control de procesos

Jheison René Gutérrez Gómez

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

Modelado y simulación de procesos controlados para el curso de introducción al control de procesos

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 lazos de control realimentados ***On-Off*** y ***PID***. La metodología utilizada fue crear un algoritmo lineal, compuesto de módulos que tienen como función hacer la programación de este tipo de problemas intuitiva, lineal, fácil y comprensible. El algoritmo está compuesto por módulos, cada uno de los cuales tiene una finalidad específica: agrupar acciones y características similares de las variables que influyen en el proceso. Los módulos son: 1-Parámetros y constantes; 2-Condiciones iniciales; 3-Tiempo de simulación, paso temporal y número de iteraciones; 4-Creación de vectores para guardar datos; 5- Set Point y perturbaciones; 6-Parámetros para el controlador; 7-Actualización y llenado de los vectores para las variables del sistema; 8-Solución y programación de ecuaciones; 9- Programación de las acciones del controlador; y 10-Gráficas y resultados. El trabajo abordara tanto el modelamiento matemático de cada uno de los sistemas, así como la programación detallada de estos modelos y de esta manera se construiran cada uno de los simuladores.

La solución de los modelos matemáticos se realizó utilizando el método de Euler, un método sencillo, al igual que la programación de los simuladores. Los casos estudiados en este documento son:

1- Modelado y simulación de dos controladores, uno en ***On-Off*** y otro en ***PID*** para el control de nivel en un tanque.

2- Modelado y simulación de dos controladores, uno en ***On-Off*** y otro en ***PID*** para el control de temperatura en un intercambiador de calor.

3- Modelado y simulación de un controlador ***PID*** para el control de nivel y de presión del vapor en una caldera pirotubular.

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

Palabras clave: **Simulación, programación, control , proceso, modelado, método, variable de control, variable de proceso, perturbación, Set Point, PID, On-Off, modulo, Python, Jupyter Notebook**.

Abstract

The main objective of this work is to provide a practical guide of how to program and approach simulation problems, mainly models for feedback control loops ***On-Off*** and ***PID***. The used methodology was to create a linear algorithm, made up of modules that have as function to make the programming of this type of problems in an intuitive, linear, easy and understandable way. The algorithm is made up of modules, each of them has a specific purpose that is to group actions and similar characteristics of the variables that influence the process. The modules are: The algorithm is made up of modules, each of them has a specific purpose that is to group actions and similar characteristics of the variables that influence the process. The modules are: 1. Parameters and constants. 2. Initial conditions. 3. Simulation time, time partitions and number of iterations. 4. Creation of vectors to save data. 5. Set point and disturbances. 6. Parameters for the controller. 7. Update and filling the vectors for the system variables. 8. Equations programming and solution. 9. Programming of the actions of the controller. 10. Graphics and results. This work will approach both the mathematical modeling of each of the systems, as well as the detailed programming of these models and in this way will build each one of the simulators. The solution of the mathematical models was carried out using simple numeric methods, as well as simulator programming. The three cases studied in this document are:

1- Modeling and simulation of two controllers, one in ***On-Off*** and other in ***PID*** for level control in a tank.

2- Modeling and simulation of two controllers, one in ***On-Off*** and other in ***PID*** for the temperature control in a heat exchanger.

3- Modeling and simulation of a ***PID*** controller for the level and steam pressure control in a shell boiler.

All programming of this document was done using the programming language python 3, and the present work was built in it's interactive web environment of code execution ***Jupyter Notebook***.

Keywords: Simulation, programation, control, process, model, method, process variable, process control, process disturbance, Set Point, PID, ON-Off, module, Python, Jupyter Notebook

Índice

Índice de figuras

Lista de símbolos

Símbolos	Término	Unidad SI
A_0	Área del orificio	m^2
A_T	Área transversal	m^2
$C_{P,L}$	Calor especifico del líquido	$\frac{kj}{kg}$
$C_{P,V}$	Calor especifico del vapor	$\frac{kj}{kg}$
g	Aceleración de la gravedad	$\frac{m}{s^2}$
h	Longitud horizontal del la caldela	m
$h_{f,1-2}$	Perdidas por fricción	
h_{vap}	Calor latente	$\frac{kj}{kg}$
K_D	Parámetro de ganancia derivativa	s
K_I	Parámetro de ganancia integral	s
K_P	Parámetro ganancia proporcional	%
L	Longitud	m
M	Masa total	kg
$\bar{M}$	Masa molar	$\frac{kj}{kg}$
$\dot{m}_V$	Flujo másico de vapor	$\frac{kj}{s}$
$m_{in,L}$	Flujo másico de líquido a la entrada	$\frac{kj}{s}$
$m_{out,L}$	Flujo másico de líquido a la salida	$\frac{kj}{s}$
P_V	Presión de vapor	bar
Q	Flujo de calor	kw
Q_{req}	Flujo de calor requerido	kw
R	Constante universal de los gases	$\frac{kj}{kmolK}$
r	Radio de la caldera	m
T	Tempetarura	K
T_L	Temperatura líquido	K
T_{ebu}	Temperatura de ebullición	K
$T_{in,L}$	Temperatura de entrada del líquido	K
$T_{out,L}$	Temperatura de salida del líquido	K
T_{sat}	Temperatura de saturación	K
t	Tiempo	s
t_D	Tiempo derivativo	s
t_I	Tiempo integral	s
v	Velocidad	$\frac{m}{s}$
V	Volumen	m^3
V_L	Volumen del líquido	m^3
V_V	Volumen del vapor	m^3
V_{total}	Volumen total	m^3
$\dot{V}_{in,L}$	Flujo volumétrico de entrada del líquido	$\frac{m^3}{s}$
$\dot{V}_{out,L}$	Flujo volumétrico de salida del líquido	$\frac{m^3}{s}$
$\dot{V}_{in,V}$	Flujo volumétrico de entrada del vapor	$\frac{m^3}{s}$
ρ_V	Densidad del vapor	$\frac{kj}{m^3}$

Símbolos	Término	Unidad SI
Δu	Cambio a aplicar en la acción de control	Und. variable proceso

1. Introducción

El presente documento denominado "Simulación de procesos controlados para el curso de introducción al control de procesos", está diseñado para servir de guía en la programación y simulación de sistemas de control On-Off y PID. Este trabajo se realizó debido a la necesidad y dificultad que tienen algunas ramas de la ingeniería en modelar, programar y diseñar simulaciones que representen 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.

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 y comprensión, además de que este es 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 control como: el control P con el PID, y el On-Off con y sin zona muerta, también identificar los diferentes tipos de accionar de cada controlador y como representar esto por medio de un lenguaje de programación como Python.

En el desarrollo de este documento primeramente el Capítulo 2 se hablará un poco sobre el lenguaje de programación utilizado (Python 3) y también se hablará de los tipos de controladores estudiados en este trabajo. 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 controlador y sistema. Primero se desarrolla el sistema más sencillo, el de control de nivel de líquido en un tanque cilíndrico. Posteriormente, se continúa con el intercambio de calor entre dos sustancias y finalmente con el sistema de doble controlador, para el nivel de líquido y para la presión del vapor en una caldera pirotubular. En el Capítulo 4, se muestran los resultados gráficos para cada uno de los simuladores elaborados, para finalmente 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].

2.2. Controladores On-Off y PID

Un sistema de control es esa otra parte que hace posible la operación adecuada de un proceso. Aunque todo proceso puede funcionar sin sistema de control (lazo abierto), su seguridad operativa, calidad de producto y eficiencia no están garantizadas. Un sistema de control se encarga de recibir señales, realizar cálculos matemáticos para comparar la variable con su punto de ajuste y tomar la desición de en que cantidad actuar y transmitir esta señal al elemento final de control.

En los sistemas de control existen tres variables fundamentales: la variable de proceso, la variable de control y las variables que perturban el proceso. Pero existen otros parámetros importantes como el error, que es mas que la desviación de la salida frente a lo deseado o punto de ajuste (Set Point). Existen tres tipos de sistemas de control posibles: prealimentados, retroalimentados y combinado. En este trabajo solo se abordan controladores retroalimentados. El controlador en On-Off el más sencillo, ya que posee dos posibles acciones, de control o posiciones de la válvula. En cambio, la válvula para un controlador PID puede tomar infinitas posiciones de apertura.

Señalando lo mencionado anteriormente el controlador en On-Off solo posee dos posibles acciones, por lo tanto numéricamente se puede representar fácilmente entre un valor máximo y uno mínimo. En los controladores PID hay una acción correctiva al error o la diferencia entre la salida que se desea y la salida de la variable de proceso. Esto se hace añadiéndole unos parámetros correctivos al error llamados parámetro proporcional, integral y derivativo. A continuación se pueden observar las ecuaciones para un controlador P y PID respectivamente.

(t)= $\mathrm{K}_p(SP - VP) = K_P Error(t)$

(t)=$\mathrm{K}_P Error + K_I \int_{t_0}^{t} Error(t)dt + K_D \frac{dError(t)}{dt}$

Donde K_P,K_IyK_D son los parámetros proporcional, integral y derivativo respectivamente, VP es la variable de proceso, SP el Set Point y Δu es el cambio a aplicar en el instante actual. Para más información se recomienda consultar al lector el escrito libro trabajo sabático de [1]

3. Modelado y programación de controladores

En este capítulo se abordarán problemas que involucran principalmente sistemas de control realimentado. Los procesos abordados en los casos de estudio se modelaron matemáticamente y se programaron para realizar la respectiva simulación. En este capítulo se expondrá el algoritmo llevado a cabo para modelar controladores On-Off con zona y sin zona muerta y controladores PID en sistemas como: el control de nivel de líquido en un tanque, el control de temperatura de un líquido en un intercambio de calor y por último un sistema con doble controlador PID, para el nivel de líquido y la presión del vapor en una caldera pirotubular.

Las primeras secciones ilustran la deducción matemática para cada sistema estudiado y posteriormente se argumentará sobre la construcción, función y programación del algoritmo diseñado para simular cada uno de los sistemas estudiados. Se debe mencionar que los métodos numéricos utilizados para solucionar cada modelo matemático y la programación empleada en la construcción de los simuladores es sencilla, fácil e intuitiva.

3.1. Control de nivel de líquido en un tanque

3.1.1. Modelamiento matemático

Para la realización de este modelo, se tendrá en cuenta que el tanque es de geometría cilíndrica y está instalado verticalmente. Al tanque ingresa y sale agua liquida sin que ocurran procesos de transferencia de calor, transferencia de masa o reacción química. Lo primero para la deducción del modelo es realizar balances de materia y energía. La variable controlada del proceso proceso es el nivel del tanque, la variable manipulada es el flujo que sale del tanque, y la variable de perturbación es el flujo que ingresa a este mismo. Por el balance de masa total se llega a:

$$\frac{dV_L}{dt} = \dot{V}_{in,L} - \dot{V}_{out,L} \tag{1}$$

El volumen que ocupa el líquido dentro del tanque se puede expresar como el producto del área transversal de este por la altura del líquido en el tanque:

$$V_L = A_T L \tag{2}$$

Derivando en función del tiempo y sacando el área transversal como constante por ser un tanque cilíndrico vertical se tiene:

$$\frac{dV_L}{dt} = A_T \frac{dL}{dt} \tag{3}$$

Igualando las ecuaciones 3 y 1, y dejando al diferencial a la izquierda, se llega a:

$$\frac{dL}{dt} = \frac{\dot{V}_{in,L} - \dot{V}_{out,L}}{A_T} \tag{4}$$

Ahora, mediante un balance de energía mecánica en estado estacionario se obtiene:

$$\frac{P_1}{\rho_1} + gz_1 + \frac{v_1^2}{2} = \frac{P_2}{\rho_2} + gz_2 + \frac{v_2^2}{2} + h_{f,1-2} \tag{5}$$

En la que la presión en los dos puntos es la misma, presión atmosférica, y las pérdidas por fricción en orificio no se consideran porque se toma justo el punto antes de que el fluido cruce dicha abertura. Esto lleva a:

$$\frac{v^2}{2} = gL \tag{6}$$

Despejando la velocidad del fluido que sale por el orificio se tiene.

$$v = \sqrt{2gL} \tag{7}$$

Se sabe que el caudal de salida de líquido a través de una tubería u orificio circular se puede definir como:

$$\dot{V}_{out,L} = vA_o \tag{8}$$

Reemplazando la velocidad de la ecuación 8 con la expresión hallada en la ecuación 7 se tiene:

$$\dot{V}_{out,L} = A_o\sqrt{2gL} \tag{9}$$

Por último se reemplaza la ecuación 9 en la ecuación 4.

$$\frac{dL}{dt} = \frac{\dot{V}_{in,L} - A_o\sqrt{2gL}}{A_T} \tag{10}$$

La ecuación 10 representa el cambio que tiene el nivel de líquido como función del tiempo en un tanque, cuando este se vacía.

3.1.2. Programación del modelo para el control de nivel de líquido en un tanque 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 tienen los modulos en la solución de este tipo de sistemas, en este caso el control de nivel de líquido en un tanque. Se mostrará tanto la programación del controlador *On-Off* y posteriormente el *PID*. La finalidad de esto es mostrar lo intuitivo que puede ser la creación de este tipo de algoritmos, ya que la programación es casi la misma para los dos controladores, solo difieren de los módulos 6 y 9, recalcando que los módulos en ambos casos cumplen casi la misma función con parámetros distintos pero acciones parecidas.

Se debe mencionar que la construcción del código mezcla la programación de los dos tipos de controladores, es decir los pasos funcionan de forma similar para la construcción no solo de los diferentes sistemas estudiados sino tambien para ambos tipos de controladores *On-Off* y *PID*.

Descripción del proceso

- El tanque de nivel estudiado a continuación, es un recipiente en el cuál ingresa y sale líquido.
- La geometria utilizada para el tanque es cilíndrica vertical.
- La variable controlada en el proceso es el nivel del tanque. Esta variable se mantiene en su valor deseado regulando el flujo de salida del líquido.

Datos del proceso

- Fluido de proceso = Agua líquida a T=25 $[C]$.
- Variable de proceso (y)= Nivel de agua en el tanque ***Nivel_i*** $[m]$.
- Variable de control = Flujo volumétrico de salida ***FlujoV_out_i*** $[\frac{m^3}{s}]$.
- Variable de perturbación = Flujo volumétrico de entrada ***FlujoV_Liq_in_i*** $[\frac{m^3}{s}]$.
- Acción de control (u) = Apertura válvula de salida ***Val_Ctrl_i*** $[\,\%]$.
- Acción de perturbación (d) = Apertura de la válvula de entrada ***Val_Pert_i*** $[\,\%]$.

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, y matplotlib que permite gráficar los datos obtenidos.

[ ]:
```
import numpy as np  ##  Librería de python que permite realizar operaciones con␣
 ↪vectores y matrices.
import matplotlib.pyplot as pt  ##  Libreria de python que permite realizar␣
 ↪gráficas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo: área transversal del tanque, el área del orificio y el valor que se desea de Set Point.

[ ]:
```
Area= 1  ##  Área transversal del tanque [m^2].
Area_Orf= 1.87536e-4  ##  Área del orificio de descarga del tanque [m^2].
SP= 1.63165  ##  Asigna un valor al set point.
```

Parámetros de las sustancias y constantes

- En este paso del algoritmo se juntan los parámetros y contantes de las sustancias involucradas en el proceso, en este caso la única constante es la gravitacional.

[ ]:
```
g= 9.8  ##  Constante gravitacional [m/s^2].
```

MÓDULO 2- Condiciones iniciales en estado estacionario

- En este módulo se le dan valores iniciales o de entrada a las variables de proceso, control y perturbación. Se debe indicar que los valores en que inician estas variables son los valores de estado estacionario puesto que se asume que inicia de esta manera.

[ ]:
```
#ini=Inicial, in=Entrada.

Nivel_ini=SP  ##  Nivel inicial del tanque [m].
Val_Pert_ini= 50  ##  Apertura inicial de la válvula de entrada (perturbación)␣
 ↪[%].
Val_Ctrl_ini= 50  ##  Apertura inicial de la válvula de salida (acción de␣
 ↪control) [%].
FlujoV_Liq_in= 0.001060537  ##  Flujo volumétrico de entrada.
```

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 asume se realiza la medición, el número de particiones ***N_Part*** determina las veces que se desea fragmentar el intervalo de tiempo.
- Para la variable 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 dividiéndolo en N intervalos ***N_Part***. En este caso se asumen que son intervalos de tiempo.

- La variable ***t_Pert*** es el tiempo en el que se perturba el sistema. Para este caso se inicia cuando ha pasado el 30 % del número total de particiones temporales.

```
[ ]: t_Inicial = 0  ##  Tiempo en que inicia la simulación [s].
     t_Final = 300  ## Tiempo en que finaliza la simulación [s].
     Paso = 0.5  ##  Tamaño de cada partición de intervalo de tiempo.
     N_Part= int((t_Final-t_Inicial)/(Paso))  ##  Determina el número de particiones␣
      ↪necesarias para resolver el método.
     tiempo = np.linspace(t_Inicial, t_Final, N_Part)  ##  Crea vector de tiempo para␣
      ↪poder graficarlo [s].
     t_Pert = int(0.2 * N_Part)  ##  Tiempo en que se aplica la perturbación del␣
      ↪tiempo final de simulación.
```

MÓDULO 4- Creación de vectores para guardar datos

- Este módulo construye vectores de cierta dimensión para las variables de interés, como se puede ver son las variables de proceso, perturbaciones, variables de control, Set Point y el error. Estos vectores tienen la finalidad de guardar los datos de cada una de las soluciones del método.
- 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 zeros.
- Las dimensiones para los vectores las otorga ***len(tiempo)***, indicando que se va a crear un vector de ceros con las dimensiones idénticas a las del vector ***tiempo***. Esto último se realiza con el fin de que las dimensiones en los vectores se actualicen automáticamente si se desea cambiar el valor del tiempo de simulación o paso.

```
[ ]: FlujoV_Liq_in_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el␣
      ↪flujo de entrada, (Variable de perturbación).
     Nivel_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el nivel del␣
      ↪tanque (y),(Variable proceso).
     Val_Pert_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para la apertura␣
      ↪de la valvula entrada (d), (Acción de perturbación).
     FlujoV_Liq_out_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el␣
      ↪flujo de salida de la válvula (Variable control).
     Error_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el error= Set␣
      ↪Point- nivel.
     SP_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el Set Point.
     Val_Ctrl_i = np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para la␣
      ↪apertura de la válvula salida (u), (Acción de control).
```

MÓDULO 5- Set Point y perturbaciones

Set Point

- En este caso para el vector que guarda los datos del Set Point ***SP_i***, que es un vector de ceros creado en el modulo 4, se le otorga un valor inicial (su valor de estado estacionario) en la primer posición del vector. Para Python la primer posición en un vector corresponde al

indice i=0, por ende es recomendable que la variable ***t_Inicial*** inicie en el tiempo cero.

- La segunda linea es únicamente válida en este caso para los controladores PID. Esta le dice al vector de Set point ***SP_i*** que cambie su valor en el tiempo ***t_SP_2***. De esta manera se pueden cambiar los valores del Set Point a cualquier instante de tiempo al valor que se desee, en este caso se aumenta el nivel del líquido en un centímetro.

```
[ ]: SP_i[t_Inicial:]= SP  ##  Llena el vector del Set Point con el valor asignado␣
      ↪como referencia.
     SP_i[t_SP_2:]= SP+0.01  ##  Cambia a un nuevo valor del Set Point, las␣
      ↪posiciones del vector desde t_SP_2 en adelante.
```

Perturbaciones

- Ahora se realiza un cambio en escalon similar al del set point pero ahora con la variable de perturbación.
- La segunda linea significa que después de iniciar la variable de perturbación en su estado estacionario, al tiempo que se le dio a la perturbación para que iniciara en el modulo 3, aumente su valor en un 40 %. De esta manera se puede simular la perturbación del sistema con un vector en Python.

```
[ ]: Val_Pert_i[t_Inicial:]=Val_Pert_ini  ##  Inicia el valor de apertura para la␣
      ↪válvula del líquido en estado estacionario [%].
     Val_Pert_i[t_Pert:]=Val_Pert_ini+40  ##  Cambia el valor de apertura de la␣
      ↪válvula.Tiempo en que inicia la perturbación t_Pert.
```

MÓDULO 6- Parámetros del controlador (ON-OFF)

- ***Módulo válido unicamente si se desea programar un control en On-Off.***
- En este módulo se escoge si el controlador **On-Off** se desea trabajar con o sin zona muerta.
- Si el controlador se decide trabajar con zona muerta, que no es nada más que un intervalo de libertad en el valor de la variable de proceso, se puede dar un valor a la banda de zona muerta.

```
[ ]: Band_ZM=0.01  ##  (+-) Límite o banda para la zona muerta [m]
     Zona_Muerta = #int(input("SIN zona muerta marque (0)--CON zona muerta marque(1)␣
      ↪= "))
```

MODULO 6- Parámetros del controlador (PID)

- ***Modulo valido únicamente si se desea programar un control PID.***
- En este módulo se le asignan valores a los parámetros del controlador PID.
- ***ITC*** es el intervalo de tiempo de la acción de control. Este valor puede cambiarse y es el intervalo de tiempo en que el controlador actualiza el valor de la acción de control. Este valor se puede cambiar en el módulo 3.
- ***e_1k*** y ***e_2k*** son los errores de la variable controlada en el tiempo pasado y en el tiempo antepasado. En el tiempo cero, estos errores son cero por iniciar en estado estacionario, esto quiere decir que el Set Point y la variable de proceso arrancan el mismo valor.

```
[ ]: Kp= -1000  ##  Parámetro de ganancia proporcional [%].
     ti= 10  ##  Parámetro para el tiempo integral [s].
     td= 4  ##  Parámetro para el tiempo derivativo [s].
     ITC= 1  ##  Intervalo de tiempo de la acción de control [s].
     e_1k= 0  ##  Error en el instante de tiempo anterior (i-1).
     e_2k= 0  ##  Error en el instante de tiempo anterior (i-2).
```

MODULO 7- Actualización y llenado de los vectores para las variables del sistema

- Es este paso del algoritmo primero se inicializa variable de control para el nivel del líquido ***Valv_Ctrl_i*** con su valor inicial. En el lenguaje de programación es indicar la posición del vector y darle un valor. En este caso se le da un valor a la primer posición del vector de datos creado en el módulo 4.
- Para empezar con la solución del método numérico, se inicia el ciclo iterativo, este comienza en cero que indica la primer posición de un vector en python, hasta la última posición del vector o número de iteraciones que las determina el valor dado en el modulo 3 para el número de particiones del vector tiempo.
- Las siguientes lineas para el nivel ***Nivel_i***, la válvula de control ***Val_Ctrl_i*** y el error ***Error_i***, se realizan para que al finalizar cada ciclo con un nuevo valor, este ingrese de nuevo al ciclo ***for***. De esta manera se resuelve el método con cada uno de los valores obtenidos al terminar cada ciclo, completando de esta manera el número de iteraciones, llenando así cada uno de los vectores que representan a las variables de interés con los resultados de cada iteración.
- En el siguiente bloque de código, hay dos variables de control ***Val_Ctrl_i***, esto porque una es exclusiva para el controlador en On-Off y la otra es únicamente para el controlador PID

```
[ ]: # AQUÍ EMPIEZA LA SOLUCIÓN ITERATIVA DEL MODELO CON SU CONTROLADOR#

     Nivel= Nivel_ini  ##  Asigna el valor inicial al nivel del tanque para iniciar␣
      ↪el método.

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

     # ACTUALIZACIÓN DE LAS VARIABLES DE PROCESO, CONTROL Y ERROR.

        Nivel_i[i] = Nivel  ##  Guarda los datos para cada solución de la Ecuación␣
      ↪diferencial del nivel.
        Error_i[i] = (SP_i[i] - Nivel)  ##  Guarda el error en cada iteración.
        Val_Ctrl_i[i] = Val_Ctrl_ini  ##  Guarda los datos para cada solución de la␣
      ↪variable de control.(Apertura de la válvula). Valido unicamente para el␣
      ↪controlador en On-Off.
        Val_Ctrl_i[t_Inicial]= Val_Ctrl_ini  ##  Guarda los datos para cada solución␣
      ↪de la variable de control.(Apertura de la válvula). Valido unicamente para el␣
      ↪controlador PID.
```

MÓDULO 8- Solución y programación de ecuaciones

Ecuaciones constitutivas

- En esta parte del módulo, se solucionan y actualizan las ecuaciones constitutivas del modelo

para el flujo volumétrico de la válvula de líquido a la entrada ***FlujoV_Liq_in_i***, que simplemente es el porcentaje de apertura por el flujo. Para el flujo volumétrico de salida del tanque ***FlujoV_out_i*** lo mismo, solo que está constituida por la ecuación de energía mecánica, ya que se trata de un tanque con un orificio en su parte inferior.

- La deducción del modelo matématico se puede ver en la Sección 3.1.1 de este documento.

[ ]:
```
    FlujoV_Liq_in_i[i]= (Val_Pert_i[i]/100)*FlujoV_Liq_in  ##  Soluciona y␣
↪actualiza los valores para la ecuación de flujo volumétrico de entrada [m3/s].
    FlujoV_Liq_out_i[i]= (Val_Ctrl_i[i]/100)*Area_Orf*np.sqrt(2*g*Nivel_i[i]) ␣
↪##  Soluciona y actualiza los valores para el flujo volumétrico de salida [m^3/
↪s].
```

Ecuaciones diferenciales

- En esta sección, se programa un método numérico para solucionar ecuaciones diferenciales. Por su simplicidad se programó el método Euler.
- Lo primero para resolver el método es despejar la derivada ***dLdt*** que denota el cambio de la variable de interés en el tiempo.
- Realizado el paso anterior, se cálcula el nuevo valor para el nivel en la siguiente partición de tiempo ***Nivel***. Esto se realiza sumándole al valor obtenido en la partición anterior ***Nivel_ini*** el cambio ***dLdt*** que tiene esta variable en el instante de tiempo anterior, multiplicado por el intervalo ***Paso*** de tiempo asignado en el módulo 3.
- Finalmente, la última línea de código otorga el valor actual ***Nivel*** a la variable inicial ***Nivel_ini*** para que se realice el proceso iterativo nuevamente hasta completar las particiones temporales.

[ ]:
```
    dLdt= (FlujoV_Liq_in_i[i]-FlujoV_Liq_out_i[i])/Area  ##  Soluciona la␣
↪derivada de la ED para el nivel.
    Nivel= Nivel_ini+dLdt*Paso  ##  Soluciona la ecuación diferencial,␣
↪agregandole al nivel inicial el cambio que tiene esta en una pequeña partición␣
↪o instante de tiempo (Paso).
    Nivel_ini= Nivel  ##  Actualiza el nivel para iniciar con el siguiente paso␣
↪[m].
```

MÓDULO 9- Programación de las acciones del controlador (ON-OFF)

- ***Módulo válido únicamente si se desea programar un control en On-Off.***
- Las acciones que realiza un controlador en On-Off son dos, ya que las variables de control en este caso válvulas de control On-Off, solo poseen dos estados, abiertas 100 % o cerradas completamente 0 %.
- Lo primero es indicar en caso de haber escogido o no la zona muerta en el modulo 6 las acciones respectivas para los dos casos.
- Para las acciones sin zona muerta ***Zona_Muerta==0***. La primer acción del condicional ***if*** limita a la variable de control en caso de que el error sea menor a cero, esto significa que la variable de proceso esta por encima del Set Point, entonces abre al máximo la válvula de control, ya que esta es la que controla el flujo que sale del tanque.
- En caso de que se escoja el controlador con zona muerta, el primer condicional ***if*** actualiza la acción de control normalemnte, ya que esta se encuentra en el rango de zona muerta. El error o la diferencia entre el Set Point y la variable de control debe ser menor al rango de

la banda de zona muerta, ya que hasta este valor es tolerable que suba o baje el nivel de líquido. Sin cambiar la posición que trae la válvula.

- Las siguientes lineas para el controlador con zona muerta, condicionales **elif**, solo accionan la variable de control como si se tratase de un controlador On-Off sin zona muerta, manteniendo los valores en su máximo o mínimo valor cuando sea necesario.

[ ]:
```
# APLICA PARA EL CONTROLADOR ON_OFF SIN ZONA MUERTA #

if Zona_Muerta==0:

        if Error_i[i]<0:  ##  Acción de control de regulación (corrección de la␣
 ↪perturbacion).

            Val_Ctrl_ini=100  ##  Corrige la acción de control, cuando la␣
 ↪variable de proceso es mayor al Set Point.

        else:

            Val_Ctrl_ini=0  ##  Corrige la acción de control, cuando la variable␣
 ↪de proceso es menor al Set Point.

# APLICA PARA EL CONTROLADOR ON_OFF CON ZONA MUERTA #

    else:  ##  Segunda acción de control CON Zona Muerta, si el error es mayor a␣
 ↪cero, se genera una respuesta de control de regulación contraria.

        if abs(Error_i[i])<Band_ZM:

            Val_Ctrl_ini=Val_Ctrl_i[i]  ##  Mantiene la acción de control en el␣
 ↪mismo punto.

        elif Error_i[i]<0:  ##  Control On-Off convencional.

            Val_Ctrl_ini=100  ##  Corrige la acción de control, cuando la␣
 ↪variable de proceso es mayor al Set Point.

        elif Error_i[i]>0:

            Val_Ctrl_ini=0  ##  Corrige la acción de control, cuando la variable␣
 ↪de proceso es menor al Set Point.
```

MÓDULO 9- Programación de las acciones del controlador (PID)

- ***Módulo válido únicamente si se desea programar un control PID***.
- En este módulo se escogera el tipo de controlador (PID) a usar, y se programan las acciones del controlador.
- El primer condicional ***if*** indica la acción a realizar en caso de requerir un controlador (P). Como se puede ver esto depende de los valores que se le dan a los parámetros integral ***ti*** y

derivativo ***td*** en el modulo 6..

- El condicional ***else*** funciona en caso de requerir un controlador (PID), como se puede ver lo único que cambia es la ecuación del controlador que se puede encontrar en las notas de *Trabajo de año sabático 2017, efectos dinámicos en operaciones unitarias, modelado de procesos para su análisis y control de Hernán D.Alvarez Z.*
- Las siguientes dos lineas actualizan los errores penúltimo y antepenúltimo, que son requeridos en las ecuaciones de los controladores.
- Para las acciones de este controlador, el primer condicional ***if*** mide el valor que posee la variable de control más el valor calculado en la ecuación del controlador, si este valor es mayor al valor máximo permitido, la siguiente linea de código lo condiciona para que se mantenga en este valor máximo, es decir completamente abierta.
- El condicional ***elif*** quiere decir en Python "de lo contrario si" es como un condicional ***if*** y ***else*** combinados. Este da la acción contraria a la variable de control, es decir, mide si la variable de control se encuentra en su valor mínimo permitido y la mantiene en este valor, lo que quiere decir que la válvula se encuentra completamente cerrada.
- Por último, el condicional ***else*** actualiza el valor de la válvula de control, dandole valores intermedios de apertura a la válvula de control, ya que este tipo de válvula si puede tomar valores diferentes de 0 % y 100 %, ya que este puede regular su apertura a diferentes posiciones, lo que no ocurre con las válvulas On-Off, que solo poseen dos posiciones.

[ ]:
```
#  CONTROLADOR (P).

    if (td==0 and ti>=9999):  ##  Control (P), depende de los parametros td y ti.
        Delta_u = Kp*ITC*Error_i[i]  ##  Ecuación para el control (P).


#  CONTROLADOR (PID).

    else:
         Delta_u= Kp*ITC*(Error_i[i]-e_1k) + Kp*(ITC/ti)*e_1k + Kp*(td/
 ↪ITC)*(Error_i[i]-2*e_1k+e_2k)  ##  Ecuación para el control (PID)


#  ACTUALIZACION DEL ERROR.

    e_1k=Error_i[i]  ##  Actualiza el error anterior (i-1).
    e_2k=e_1k  ##  Actualiza el error anterior (i-2).


# ACCIÓNES DE CONTROL.

    if (Val_Ctrl_i[i]+Delta_u)>100:
        Val_Ctrl_i[i+1]=100  ##  Corrige la acción de control para que no pase␣
 ↪de su límite máximo.

    elif (Val_Ctrl_i[i]+Delta_u)<0:
```

```
        Val_Ctrl_i[i+1]=0  ##  Corrige la acción de control para que no pase de␣
 ↪su límite mínimo.

    else:
         Val_Ctrl_i[i+1]=Val_Ctrl_i[i]+Delta_u  ##  Acción de control dentro de␣
 ↪los límites establecidos, se actualiza normalmente sumandole el cambio.

Val_Ctrl_i=Val_Ctrl_i[:i+1]  ##  Ajusta el vector a las dimensiones requeridas.
```

MÓDULO 10- Gráficas y resultados

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

Gráfica: Nivel vs Tiempo

[ ]:
```
pt.figure("MODELO TANQUE (CONTROL PID)", [10,5])  ##  Crea figura y le otorga␣
 ↪dimensiones

pt.subplot(2, 3, 1)  ##  Hace una sub-figura (N°filas , N°columnas , Posicion de␣
 ↪la figura en el subplot).
pt.tight_layout(pad=3, w_pad=5, h_pad=2)  ##  Espaciado entre las figuras del␣
 ↪subplot (Espaciado entre la margen , Espaciado entre figuras (Horizontal) ,␣
 ↪Espaciado entre figuras (Vertical).
pt.grid(True)  ##  Agrega la cuadrilla a la grafica.
pt.plot(tiempo, Nivel_i, "k", linewidth=2)  ##  Grafica los datos requeridos (x␣
 ↪, y , Color de la linea , Grosor de la linea).
pt.plot(tiempo, SP_i, "y--", label="Set point", linewidth=1)  ##  Grafica el Set␣
 ↪Point.
pt.xlabel("Tiempo [s]")  ##  Agrega título al eje x.
pt.ylabel("Nivel [m]")  ##  Agrega título al eje y.
pt.ylim(min(Nivel_i)-0.01,max(Nivel_i)+0.01)  ##  Ajusta los limites en el eje␣
 ↪(y) para mejor visualización.
pt.title("Variable de proceso")
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).
pt.legend(loc="best")  ##  Agrega la leyenda en la grafica.("best" es la mejor␣
 ↪ubicación).
```

Gráfica: Perturbación vs Tiempo

[ ]:
```
pt.subplot(2, 3, 2)
pt.grid(True)
pt.plot(tiempo, Val_Pert_i, "b", linewidth=2)
pt.xlabel("Tiempo [s]")
```

```
pt.ylabel("Apertura válvula \nEntrada [%]")
pt.title("Perturbación")
pt.tick_params(labelsize=8)
pt.ylim(0, 110)
```

Gráfica: Acción de control vs Tiempo

[ ]:
```
pt.subplot(2, 3, 3)
pt.grid(True)
pt.plot(tiempo, Val_Ctrl_i, "g", linewidth=2)
pt.xlabel("Tiempo [s]")
pt.ylabel("Apertura válvula\nSalida [%]")
pt.title("Variable de control")
pt.tick_params(labelsize=8)
pt.ylim(0, 110)
```

Gráfica: Error vs Tiempo

[ ]:
```
pt.subplot(2, 3, 4)
pt.grid(True)
pt.plot(tiempo, Error_i, "r", linewidth=2)
pt.xlabel("Tiempo [s]")
pt.ylabel("Error [m]")
pt.tick_params(labelsize=8)
pt.ylim(min(Error_i)- 0.001, max(Error_i)+ 0.001)
```

Gráfica: Flujo volumétrico de líquido a la entrada vs Tiempo

[ ]:
```
pt.subplot(2,3,5)
pt.grid(True)
pt.plot(tiempo,FlujoV_Liq_in_i)
pt.ylabel("Flujo entrada [m\N{superscript three}/s]")
pt.xlabel("Tiempo [s]")
pt.tick_params(labelsize=8)
pt.ylim(min(FlujoV_Liq_in_i)-0.001,max(FlujoV_Liq_in_i)+0.001)
```

Gráfica: Flujo volumétrico de líquido a la salida vs Tiempo

[ ]:
```
pt.subplot(2, 3, 6)
pt.grid(True)
pt.plot(tiempo,FlujoV_Liq_out_i)
pt.ylabel("Flujo salida [m\N{superscript three}/s]")
pt.xlabel("Tiempo [s]")
pt.tick_params(labelsize=8)
pt.ylim(min(FlujoV_Liq_out_i)-0.001,max(FlujoV_Liq_out_i)+0.001)
```

3.2. Control de temperatura en un intercambio de calor

3.2.1. Modelamiento matemático.

El sistema de intercambio de calor estudiado podría verse como un típico intercambiador de tubos y coraza. En este la variable manipulada es el flujo de vapor que llena el tubo del intercambiador, la perturbación es el flujo de líquido que ingresa al intercambiador, y la variable de proceso es la temperatura a la cual sale el líquido del intercambiador. Por simplicidad en el modelo se despreciaran las resistencias térmicas la tubería, se modelara únicamente el intercambio de calor directo entre las dos sustancias. Para modelo matemático, primero se tendrá en cuenta que el vapor llena el tubo del intercambiador y se condensa, y el fluido de proceso es agua líquida que pasa a través de la coraza. Realizando un balance de energía en su estado dinámico para el fluido de proceso (agua circulando por la coraza), se tiene:

$$\frac{dT_L}{dt} = \frac{\dot{m}_{in,L}}{M}(T_{in,L} - T_{out,L}) + \frac{\dot{Q}_{req}}{MC_{p,L}} \tag{11}$$

Como ecuación constitutiva para la variable de control se tiene la expresión de balance térmico, tanto para el líquido como para el vapor en este caso aplicada para el líquido:

$$\dot{Q}_{req} = \dot{m}_{in,L}C_{p,L}(T_{in,L} - T_{out,L}) \tag{12}$$

De igual manera se supondrá que el vapor que llena el tubo del intercambiador solo transfiere su calor latente, es decir, sale como líquido condensado a la temperatura de saturación (no se sub-enfría), con lo cual el balance térmico para el vapor es:

$$\dot{Q}_{req} = \dot{m}_{in,V}h_{vap} \tag{13}$$

Igualando las ecuaciones 12 y 13 se puede calcular el flujo de vapor en estado estacionario necesario para calentar el líquido y mantenerlo en la temperatura deseada.

$$\dot{m}_{in,V} = \frac{\dot{m}_{in,L}C_{p,L}(T_{in,L} - T_{out,L})}{h_{vap}} \tag{14}$$

3.2.2. Programación del modelo de control de temperatura en Python.

A continuación se muestra la realización de los programas en ***Python*** utilizando su entorno interactivo web de ejecución de código ***Jupyter Notebook*** para así mostrar la aplicación que tienen los módulos en la solución de este tipo de sistemas en este caso la programación del modelo para un intercambio de calor.

Se mostrara tanto la programación del controlador ***On-Off*** y posteriormente el ***PID***. La finalidad de esto es mostrar lo intuitivo que puede ser la creación de este tipo de algoritmos, ya que la programación es casi la misma para los dos controladores siguiendo el orden de los módulos.

Se debe mencionar que la construcción del código mezcla la programación de los dos tipos de controladores, es decir los pasos funcionan de forma similar para la construcción no solo de los diferentes sistemas estudiados, sino también para ambos tipos de controladores ***On-Off*** y ***PID***.

Descripción del proceso

- El intercambio de calor estudiado a continuación, podría suponerse que ocurre en el típico intercambiador de calor de tubos y coraza. Este opera con un fluido de servicio, el cual es vapor de agua. Para este modelo se despreciará la resistencia de transferencia de calor de los tubos, es decir se programara el modelo únicamente con el intercambio de calor entre las sustancias.
- El fluido de proceso o "fluido a calentar", es agua. Esta ingresa aproximadamente a 25 °C. El flujo que ingresa al intercambiador es la variable que perturba el sistema, aunque también lo podría ser la temperatura de este mismo. Pero esta perturbación no posee un gran impacto en el sistema, ya que tendría que variar bastante para ver la perturbación actuar en el sistema, y esto por lo general no ocurre.
- El objetivo de control es asegurarse que el fluido de proceso, al salir del intercambiador, tenga una temperatura aproximada a 55°C.
- Tanto las acciones de control como de perturbación son realizadas por válvulas de control.

Datos del proceso

- Fluido de proceso = Agua líquida a T=(+-) 25 (Fluido por la coraza) $[C]$.
- Fluido de servicio = Vapor de agua proveniente de una caldera f(P_vapor), (Fluido por los tubos) $[\frac{kg}{s}]$.
- Variable de proceso (y)= Temperatura de salida del agua . ***T_out_i*** $[C]$.
- Variable de control = Flujo másico del vapor suministrado al intercambiador de calor. ***FlujoM_Vap_i*** $[\frac{kg}{s}]$.
- Variable de perturbación = Flujo volumétrico de entrada de agua ***FlujoM_Liq_in_i*** $[\frac{m^3}{s}]$.
- Acción de control (u) = Apertura válvula On-Off para de vapor ***Val_Ctrl_i*** $[\%]$.
- Acción de perturbación (d) = Apertura de la válvula para el líquido ***Valv_Pert_i*** $[\%]$.

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, y matplotlib que permite gráficar los datos obtenidos.

[ ]:
```
import numpy as np  ##  Librería de python que permite realizar operaciones con␣
 ↪vectores y matrices.
import matplotlib.pyplot as pt# Libreria de python que permite realizar gráficas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo, en este caso para el intercambiador de calor el volumen de este, el Set Point y la presión de vapor a la cual ingresa el vapor al intercambiador.

[ ]:
```
Volumen = 0.08  ##  Volumen del IdeCalor [m^3].
SP = 55+273.15  ##  Set Point [K].
```

```
P_Vapor= 7  ##  Presión de vapor de la caldera [bar].
```

Parámetros de las sustancias y constantes

- En este paso del algoritmo se determinan los parámetros y contantes de las sustancias como densidad, la masa que ocupa el agua en todo el volumen del intercambiador, la constante de los gases ideales, la masa molar del agua y los parámetros para calcular el calor especifico del agua líquida.

[ ]:
```
Densidad= 997  ##  Densidad agua a 25°C [kg/m^3].
Masa=Volumen*Densidad  ##  Masa de agua dentro de la coraza [kg].
R= 8.314  ##  Constante de los gases ideales [kj/kmol K].
M= 18.02  ##  Masa molar agua [Kg/Kmol].

A= 8.712  ##  Constante A para el Cp para el líquido. (Tabla C3. Smith Van Ness).
B= 1.25*1e-03  ##  Constante B para el Cp para el líquido. (Tabla C3. Smith Van␣
 ↪Ness).
C= -0.18*1e-06  ##  Constante C para el Cp para el líquido. (Tabla C3. Smith Van␣
 ↪Ness).
```

MÓDULO 2- Condiciones iniciales en estado estacionario

- En este módulo se le dan valores iniciales o de entrada a las variables de proceso, control y perturbación. Se debe indicar que los valores en que inician estas variables son los valores de estado estacionario porque el proceso inicia de esta manera.

[ ]:
```
#ini=Inicial, in=Entrada.

Val_Pert_ini = 50  ##  Apertura incial de la válcua de entrada (d) [%].
T_in = 20+273.15  ##  Temperatura de entrada (agua) al tanque [k].
Val_Ctrl_ini = 50  ## Apertura inicial de la válvula de vapor [%].
T_out_ini=SP  ##  Temperatura inicial de salida del proceso.
FlujoM_Liq_in=1 ##  Flujo másico de líquido que ingresa al intercambiador de␣
 ↪calor [kg/s].
FlujoM_Vap_in=0.06879425144  ##  Flujo másico de vapor que ingresa al IdeCalor␣
 ↪[kg/s].
FlujoM_Vap_Max=2*FlujoM_Vap_in  ##  Flujo másico máximo que puede alcanzar el␣
 ↪vapor [kg/s].
```

MÓDULO 3- Tiempo de simulación, paso temporal y numero 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 asume se realiza 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 dividiéndolo en N intervalos ***N_Part***. En este caso se asumen que son intervalos de tiempo.
- La variable ***t_Pert*** es el tiempo en el que se perturba el sistema. Para este caso se inicia cuando ha pasado el 20 % del número total de particiones temporales.

```
[ ]: t_Inicial = 0  ##  Tiempo en que inicia la simulación [s].
     t_Final = 100  ## Tiempo en que finaliza la simulación [s].
     Paso = 1  ## Tamaño de cada partición de intervalo de tiempo.
     N_Part= int((t_Final-t_Inicial)/(Paso))  ##  Determina el número de particiones␣
      ↪necesarias para resolver el método.
     tiempo = np.linspace(t_Inicial, t_Final, N_Part)  ##  Crea vector de tiempo para␣
      ↪poder graficarlo [s].
     Iter = N_Part  ##  Número de iteraciones, se puede decir que es igual al número␣
      ↪de particiones.
     t_Pert = int(0.2 * N_Part)  ##  Tiempo y valor en que se aplica la perturbación␣
      ↪[%].
```

MÓDULO 4- Creación de vectores para guardar datos

- Este módulo construye vectores de cierta dimensión para las variables de interés, como se puede ver son las variables de proceso, perturbaciones, variables de control, Set Point y el error. Estos vectores tienen la finalidad de guardar los datos de cada una de las soluciones del método.
- La librería ***numpy***, que se llama con las letras ***np***, permite crear vectores de ceros 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(tiempo)***, esto indica que se va a crear un vector de ceros con las dimensiones idénticas a las del vector
  tiempo. Esto ultimo se realiza con el fin de que las dimensiones en los vectores se actualicen automáticamente si se desea cambiar el valor del tiempo de simulación o paso.

```
[ ]: T_out_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para la temperatura␣
      ↪de salida, variable de proceso (y).
     Valv_Pert_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para la␣
      ↪(acción) de apertura de la válvula de entrada de líquido (d).
     Error_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el error= set␣
      ↪point- T_out.
     Val_Ctrl_i = np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para la␣
      ↪(acción) de control o apertura del válvula de vapor (u).
     SP_i=np.zeros(len(tiempo))  ##  Crea un vector de datos para el Set Point.
     FlujoM_Liq_in_i=np.zeros(len(tiempo))  ##  Crea un vector de datos para el flujo␣
      ↪de líquido (Fluido de proceso).
     Q_i=np.zeros(len(tiempo))  ##  Crea un vector de datos para el calor energía␣
      ↪suministrada.
     FlujoM_Vap_i=np.zeros(len(tiempo))  ##  Crea un vector de datos para el flujo␣
      ↪másico de vapor (Variable de control).
```

MÓDULO 5- SET POINT Y PERTURBACIONES

Set Point

- En este caso para el vector ***SP_i***, que es un vector de ceros creado en el módulo 4, se le otorga un valor inicial. Para Python la primer posición en un vector corresponde al índice i=0, por

ende es recomendable que la variable ***t_Inicial*** inicie en el tiempo cero.

```
[ ]: SP_i[t_Inicial:]=SP  ##  Llena el vector del Set Point con el valor asignado␣
      ↪como referencia.
```

Perturbaciones

- Ahora se realiza un cambio en escalon similar al set point pero ahora con la variable de perturbación.
- La segunda linea significa que después de iniciar la perturbación en su estado estacionario, al tiempo que se le dio a la perturbación para que iniciara en el módulo 3, aumente su valor en un 30 %. De esta manera se puede simular la perturbación del sistema con un vector en Python.

```
[ ]: Valv_Pert_i[t_Inicial:] = Val_Pert_ini  ##  Llena el vector para la perturbación␣
      ↪con el valor inicial asignado.
     Valv_Pert_i[t_Pert:] = Val_Pert_ini + 30  ##  Cambia el valor de la perturbación␣
      ↪al vector a un tiempo t_Pert(tiempo perturbación)
```

MÓDULO 6- Parámetros para el controlador (ON-OFF)

- ***Módulo valido unicamente si se desea programar un control en On-Off.***
- En este módulo se escoge si el controlador On-Off se desea trabajar con o sin zona muerta.
- Si el controlador se decide trabajar con zona muerta, que no es nada más que un intervalo de libertad en el valor de la variable de proceso, se puede dar un valor a la banda de zona muerta.

```
[ ]: Band_ZM= 1 ##  (+-) Límite o banda para la zona muerta [°C].
     Zona_Muerta = 0#int(input("SIN zona muerta marque (0)--CON zona muerta marque(1)␣
      ↪= "))
```

MÓDULO 6- Parámetros para el controlador (PID)

- ***Módulo valido únicamente si se desea programar un control PID.***
- En este módulo se le asignan valores a los parámetros del controlador PID.
- ***ITC*** es el intervalo de tiempo de la acción de control. Este valor se puede cambiar en el módulo 3.
- ***e_1k*** y ***e_2k*** son los errores de la variable controlada en el tiempo pasado y en el tiempo antepasado. En el tiempo cero, estos errores son cero por iniciar en estado estacionario. Esto quiere decir que el Set Point y la variable de proceso arrancan en el mismo valor.

```
[ ]: Kp= 20 ##  Parámetro de ganancia proporcional [%].
     ti= 0.08  ##  Parámetro para el tiempo integral [s].
     td= 100  ##  Parámetro para el tiempo derivativo [s].
     ITC= 1  ##  Intervalo de tiempo de la acción de control [s].
     e_1k= 0  ##  Error en el instante de tiempo anterior (i-1).
     e_2k= 0  ##  Error en el instante de tiempo anterior (i-2).
```

MÓDULO 7- Actualización y llenado de los vectores para las variables del sistema

- Es este paso del algoritmo primero se inicializa la variable de control para la temperatura de salida del líquido ***T_out*** con su valor inicial. En el lenguaje de programación es como indicar la posición del vector y darle un valor. En este caso se le da un valor a la primer posición del vector de datos creado en el módulo 4.
- Para empezar con la solución del método numérico, se inicia el ciclo iterativo, este comienza en cero que indica la primer posición de un vector en Python, hasta la última posición del vector o número de iteraciones que las determina el valor dado en el modulo 3 para el número de particiones del vector tiempo.
- Las siguientes líneas para la temperatura de salida del líquido ***T_out_i***, la válvula de control ***Val_Ctrl_i*** y el error ***Error_i***, se realizan para que al finalizar cada ciclo con un nuevo valor, este ingrese de nuevo al ciclo ***for***. De esta manera se resuelve el método con cada uno de los valores obtenidos al terminar cada ciclo, completando de esta manera el número de iteraciones, llenando así cada uno de los vectores que representan a las variables de interés con los resultados de cada iteración.
- En el siguiente bloque de código, hay dos variables de control ***Val_Ctrl_i***, esto porque una es exclusiva para el controlador en On-Off y la otra es únicamente para el controlador PID.

```
[ ]: # AQUÍ EMPIEZA LA SOLUCIÓN ITERATIVA DEL MODELO CON SU CONTROLADOR #

T_out= T_out_ini  ##  Asigna el valor inicial a la temperatura para iniciar el␣
 ↪método [K].

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

# ACTUALIZACIÓN DE LAS VARIABLES DE PROCESO, CONTROL Y ERROR.

    T_out_i[i] = T_out  ##  Guarda los datos para cada solución de la Ecuación␣
 ↪diferencial de Temperatura.
    Error_i[i] = (SP_i[i] - T_out_i[i])  ##  Guarda el error en cada iteración.
    Val_Ctrl_i[t_Inicial] = Val_Ctrl_ini  ##  Guarda los datos para cada␣
 ↪solución de la variable de control (Apertura de la válvula de vapor).
    Val_Ctrl_i[i] = Val_Ctrl_ini  ##  Guarda los datos para cada solución de la␣
 ↪variable de control (Apertura de la válvula de vapor).
```

MÓDULO 8- Solución y programación de ecuaciones

Ecuaciones constitutivas

- En esta parte del módulo, se solucionan y actualizan las ecuaciones constitutivas del modelo para el flujo másico de la válvula de líquido a la entrada ***FlujoV_Liq_in_i***, que simplemente es el porcentaje de apertura por el flujo. Para el flujo másico de vapor ***FlujoM_Vap_i***, lo mismo.
- La expresión utilizada para el calor especifico, se representa en el código como ***Cp_L***, es solo la ecuación para esta propiedad encontrada en la tabla C3 del libro de [6].
- La ecuación para cálcular el calor entregado por el vapor ***h_Vap***, es simplemente un polinomio que se ajustó a los datos para la entalpia de vapor saturado de la tabla A12 de [3].
- La última expresión es simplemente la deducida para el calor requerido en el modelo mate-

mático dado en este documento en la Sección 3.2.1.

[ ]:
```
    FlujoM_Liq_in_i[i] = (Valv_Pert_i[i] / 100) * FlujoM_Liq_in ##  Soluciona y␣
↪actualiza la ecuación para el flujo de proceso (perturbación).
    FlujoM_Vap_i[i]= (Val_Ctrl_i[i]/100)*FlujoM_Vap_Max  ##  Soluciona y␣
↪actualiza el flujo de vapor (acción de control).

    Cp_L= (A+(B*T_out)+(C*T_out**2))*((R)/(M))  ##  Soluciona y actualiza la␣
↪ecuación para el Cp del líquido [Kj/Kg].(Tabla C3. Smith Van Ness).
    h_Vap= -0.0006*P_Vapor**3+0.1689*P_Vapor**2-20.951*P_Vapor+2284.4  ## ␣
↪Cálcula el calor latente del vapor en función de la Presión de saturación.
    Q_i[i]= FlujoM_Vap_i[i]*h_Vap  ##  Soluciona la ecuación de balance termico,␣
↪esta es la energía que entrega el vapor de la caldera al IdeCalor (Calor␣
↪latente).
```

Ecuaciones diferenciales

- En esta sección, se programa un método numérico para solucionar ecuaciones diferenciales. Por su simplicidad se programó el método Euler.
- Lo primero para resolver el método es despejar la derivada ***dTdt*** que denota el cambio de la variable de interés en el tiempo.
- Realizado el paso anterior, se cálcula el nuevo valor para la temperatura en la siguiente partición de tiempo ***T_out***. Esto se realiza sumándole al valor obtenido en la partición anterior ***T_out_ini*** el cambio (derivada) que tiene esta variable en el instante de tiempo anterior, multiplicado por el intervalo **"Paso"** de tiempo asignado en el módulo 3.
- Finalmente, la última línea de código otorga el valor actual ***T_out*** a la variable inicial ***T_out_ini*** para que se realice el proceso iterativo nuevamente hasta completar las particiones temporales.

[ ]:
```
    dTdt =((FlujoM_Liq_in_i[i]/Masa)*(T_in-T_out_i[i]))+((1)/(Masa*Cp_L))*Q_i[i]␣
↪ ##  Soluciona la derivada de la ED para la temperatura.
    T_out= T_out_ini + dTdt * Paso  ##  Soluciona la ecuación diferencial,␣
↪agregandole a la temperatura inicial el cambio que tiene esta en una pequeña␣
↪partición o instante de tiempo (Paso).
    T_out_ini = T_out  ##  Actualiza la temperatura para iniciar con la␣
↪siguiente partición.
```

MÓDULO 9- Programación de las acciones del controlador (ON-OFF)

- ***Módulo válido unicamente si se desea programar un control en On-Off.***
- Las acciones que realiza un controlador en On-Off son dos, ya que las variables de control en este caso válvulas de control On-Off, solo poseen dos estados, abiertas 100 % o cerradas completamente 0 %.
- Lo primero es indicar en caso de haber escogido o no la zona muerta en el módulo 6 las acciones respectivas para los dos casos.
- Para las acciones sin zona muerta ***Zona_Muerta==0***. La primer acción del condicional ***if*** limita a la variable de control en caso de que el error sea mayor a cero esta se abra completamente dejando ingresar más flujo de vapor, esto quiere decir que la variable de proceso o temperatura de salida del líquido es menor al Set Point, por ende hay que aumentar el flujo

vapor para así aumentar su temperatura.

- En caso de que se escoja el controlador con zona muerta, el primer condicional ***if*** actualiza la acción de control normalmente, ya que esta se encuentra en el rango de zona muerta. El error o la diferencia entre el Set Point y la variable de control debe ser menor al rango de la banda de zona muerta, ya que hasta este valor es tolerable que suba o baje la temperatura.
- Las siguientes lineas para el controlador con zona muerta, condicionales **elif**, solo accionan la variable de control como si se tratase de un controlador On-Off sin zona muerta, manteniendo los valores en su máximo o mínimo valor cuando sea necesario.

[ ]:
```
# APLICA PARA EL CONTROLADOR ON_OFF SIN ZONA MUERTA #

if Zona_Muerta==0:

        if Error_i[i]>=0:  ##  Acción de control de regulación (corrección de la␣
 ↪perturbacion).

            Val_Ctrl_ini=100  ##  Corrige la acción de control, cuando la␣
 ↪variable de proceso es menor al Set Point.

        else:

            Val_Ctrl_ini=0  ##  Corrige la acción de control, cuando la variable␣
 ↪de proceso es mayor al Set Point.


# APLICA PARA EL CONTROLADOR ON_OFF CON ZONA MUERTA #

    else:  ##  Acción de control de regulación(corrección de la perturbacion).

        if abs(Error_i[i])<Band_ZM:

            Val_Ctrl_ini=Val_Ctrl_i[i]  ##  Mantiene la acción de control en el␣
 ↪mismo punto.

        else:  ##  Control On-Off convencional.

            if Error_i[i]<0:

                Val_Ctrl_ini=0  ##  Corrige la variable de proceso, cuando esta␣
 ↪es mayor al Set Point

            else:

                Val_Ctrl_ini=100  ##  Corrige la variable de proceso, cuando␣
 ↪esta es menor al Set Point
```

MÓDULO 9- Programación de las acciones del controlador (PID)

- ***Módulo valido unicamente si se desea programar un control PID.***
- En este módulo se escogera el tipo de controlador PID a usar, y se programan las acciones del controlador.
- El primer condicional ***if*** indica la acción a realizar en caso de requerir un controlador (P). Como se puede ver esto depende de los valores que se le dan a los parámetros integral ***ti*** y derivativo ***td***.
- El condicional ***else*** funciona en caso de requerir un controlador (PID). Como se puede ver lo unico que cambia es la ecuación del controlador.
- Las siguientes dos lineas actualizan los errores penúltimo y antepenúltimo, que son requeridos en las ecuaciones de los controladores, estas se determinaron anteriormente en este documento.
- Para las acciones de este controlador, el primer condicional ***if*** mide el valor que posee la variable de control más el valor calculado en la ecuación del controlador. Si este valor es mayor al valor máximo permitido, la siguiente linea de código lo condiciona para que se mantenga en este valor máximo, es decir completamente abierta en el paso de tiempo siguiente.
- El condicional ***elif*** quiere decir en Python "de lo contrario si" es como un condicional ***if*** y ***else*** combinados. Este da la acción contraria a la variable de control, es decir, mide si la variable de control se encuentra en su valor mínimo permitido y la mantiene en este valor en el paso de tiempo siguiente, lo que quiere decir que la válvula se encuentra completamente cerrada.
- Por último, el condicional ***else*** actualiza el valor de la válvula de control, dándole valores intermedios de apertura a la válvula de control, ya que este tipo de válvula si puede tomar valores diferentes de 0 % y 100 %, así que puede regula su apertura a diferentes posiciones, lo que no ocurre con las válvulas On-Off, que solo poseen dos posiciones.

[ ]:
```
#  CONTROLADOR (P).

    if (td==0 and ti>=9999):  ##  Control (P), depende de los parametros td y ti.
        Delta_u = Kp*ITC*Error_i[i]  ##  Ecuación para el control (P).


#  CONTROLADOR (PID).

    else:
         Delta_u= Kp*ITC*(Error_i[i]-e_1k) + Kp*(ITC/ti)*e_1k + Kp*(td/
 ↪ITC)*(Error_i[i]-2*e_1k+e_2k)  ##  Ecuación para el control (PID)


#  ACTUALIZACION DEL ERROR.

    e_1k=Error_i[i]  ##  Actualiza el error anterior (i-1).
    e_2k=e_1k  ##  Actualiza el error anterior (i-2).


# ACCIÓNES DE CONTROL.

    if (Val_Ctrl_i[i]+Delta_u)>100:
```

```
        Val_Ctrl_i[i+1]=100  ##  Corrige la acción de control para que no pase␣
↪de su límite máximo.

    elif (Val_Ctrl_i[i]+Delta_u)<0:
        Val_Ctrl_i[i+1]=0  ##  Corrige la acción de control para que no pase de␣
↪su límite mínimo.

    else:
         Val_Ctrl_i[i+1]=Val_Ctrl_i[i]+Delta_u  ##  Acción de control dentro de␣
↪los límites establecidos, se actualiza normalmente sumandole el cambio.

Val_Ctrl_i=Val_Ctrl_i[:i+1]  ##  Ajusta el vector a las dimensiones requeridas.
```

MÓDULO 10- Gráficas y resultados

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

Gráfica Temperatura vs Tiempo

[ ]:
```
pt.figure("MODELO INTERCAMBIADOR DE CALOR (CONTROL PID)", [10,5])  ##  Crea␣
↪figura y le otorga dimensiones.

pt.subplot(2, 3, 1)  ##  Hace una sub-figura (N°filas , N°columnas , Posicion de␣
↪la figura en el subplot).
pt.tight_layout(pad=3, w_pad=5, h_pad=2)  ##  Espaciado entre las figuras del␣
↪subplot (Espaciado entre la margen , Espaciado entre figuras (Horizontal) ,␣
↪Espaciado entre figuras (Vertical).
pt.plot(tiempo, T_out_i-273.15, "k", linewidth=1)  ##  Gráfica los datos␣
↪requeridos (x , y , Color de la línea , Grosor de la linea).
pt.plot(tiempo, SP_i-273.15, "y--", label="Set point", linewidth=2)  ##  Gráfica␣
↪el Set Point.
pt.grid(True)  ##  Agrega la cuadrilla a la gráfica.
pt.xlabel("Tiempo [s]")  ##  Agrega título al eje x.
pt.ylabel("Temperatura\nSalida [°C] ")  ##  Agrega título al eje y.
pt.ylim(min(T_out_i-273.15)-0.1,max(T_out_i-273.15)+0.1)  ##  Ajusta los límites␣
↪en el eje (y) para mejor visualización.
pt.title("Variable de proceso")
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).
pt.legend(loc="best")  ##  Agrega la leyenda en la grafica.("best" es la mejor␣
↪ubicación).
```

Gráfica: Perturbación vs Tiempo

[ ]:
```
pt.subplot(2, 3, 2)
pt.plot(tiempo, Valv_Pert_i, "r", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Apertura válvula\nLíquido [%]")
pt.title("Perturbación")
pt.ylim(min(Valv_Pert_i)-10,max(Valv_Pert_i)+10)
pt.tick_params(labelsize=8)
```

Gráfica: Acción de controls vs Tiempo

[ ]:
```
pt.subplot(2, 3, 3)
pt.plot(tiempo, Val_Ctrl_i, "b", linewidth=1)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Apertura válvula\n Vapor [%]")
pt.ylim(min(Val_Ctrl_i)-10,max(Val_Ctrl_i)+10)
pt.title("Variable de control")
pt.tick_params(labelsize=8)
```

Gráfica: Error vs Tiempo

[ ]:
```
pt.subplot(2, 3, 4)
pt.plot(tiempo, Error_i, "c", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Error [°C]")
pt.ylim(min(Error_i)-1,max(Error_i)+1)
pt.tick_params(labelsize=8)
```

Gráfica: Flujo másico de líquido a la entrada vs Tiempo

[ ]:
```
pt.subplot(2, 3, 5)
pt.plot(tiempo, FlujoM_Liq_in_i, "g", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Flujo agua [kg/s]")
pt.ylim(min(FlujoM_Liq_in_i)-0.1,max(FlujoM_Liq_in_i)+0.1)
pt.tick_params(labelsize=8)
```

Gráfica: Flujo másico de vapor a la entrada vs Tiempo

[ ]:
```
pt.subplot(2, 3, 6)
pt.plot(tiempo, FlujoM_Vap_i, "y", linewidth=1)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Flujo de vapor [kg/s]")
pt.ylim(min(FlujoM_Vap_i)-0.01,max(FlujoM_Vap_i)+0.01)
```

```
pt.tick_params(labelsize=8)
```

3.3. Control de nivel de líquido y presión del vapor en una caldera pirotubular

3.3.1. Modelamiento matemático

Para este sistema se modelarán y se programarán dos lazos de control, uno para el control del nivel de líquido y otro que controle la presión del vapor dentro de la caldera. La función de los controladores no es la misma, pero están relacionados entre sí, ya que al perturbar la presión al consumir mas más vapor de salida, la presión baja dentro de esta. Para que la presión se mantenga, la acción de control suministra calor a la caldera, lo que origina que mayor cantidad de líquido se evapore, bajando así el nivel. Por lo tanto, para mantener el nivel de líquido se acciona una válvula de control que permite que ingrese más o menos líquido para así mantener el nivel en su estado de referencia.

- Primero se definirá el modelo matemático para el cambio del nivel, partiendo de que el domo de la caldera posee una geometría cilíndrica, y está ubicado horizontalmente. Para representar el volumen de un cilindro ubicado horizontalmente, se define la siguiente ecuación.

$$V = HA_T \tag{15}$$

Para este tipo de geometría, la longitud H es constante por tratarse de un cilindro ubicado horizontalmente, pero el área de este cambia. Dicha área se puede representar como:

$$A_T = \frac{\pi r^2}{2} + (L-r)(\sqrt{2rL-L^2}) + r^2 \text{arcsin}(\frac{L-r}{L}) \tag{16}$$

donde L es el nivel del líquido y r el radio de la tapa circular del cilindro. Reemplazando la ecuación 16 en la ecuación 15, se llega a una expresión que representa el volumen de líquido dentro de la caldera como función del nivel:

$$V_L = H[\frac{\pi r^2}{2} + (L-r)(\sqrt{2rL-L^2}) + r^2 \text{arcsin}(\frac{L-r}{L})] \tag{17}$$

Se puede notar que, si el nivel alcanza la misma longitud del radio se obtiene el volumen de medio cilindro. Esto porque se supondrá un nivel de llenado inicial para la caldera del 50 %. Derivando la ecuación 17 en función del nivel de líquido se tiene:

$$\frac{dV_L}{dL} = H(\sqrt{2rL-h^2} + \frac{(L-r)(2r-2L)}{2\sqrt{2rL-L^2}} + \frac{r}{\sqrt{1-(\frac{L-r}{r})^2}}) \tag{18}$$

Ahora se requiere una ecuación que represente el cambio de nivel en el tiempo. Según esto podemos definir:

$$\frac{dV_L}{dt} = \frac{dV_L}{dL}\frac{dL}{dt} \tag{19}$$

Derivando la ecuación 15 con respecto al tiempo y sacando la longitud horizontal H como constante se llega a:

$$\frac{dV_L}{dt} = H\frac{dA}{dt} \tag{20}$$

Ahora se realiza un balance de materia para el líquido dentro de la caldera en términos volumétricos.

$$\frac{dV_L}{dt} = \dot{V}_{in,L} - \dot{V}_{out,L} \tag{21}$$

Vale la pena resaltar de la ecuación 21, que el flujo de líquido que sale de la piscina es el mismo flujo que se evapora de esta. En otras palabras, se puede nombrar también como el flujo de vapor. Esto se debe a que a la piscina de líquido dentro de la caldera ingresa el agua líquida y sale en estado de vapor. Igualando las ecuaciones 20 y 16 se obtiene la siguiente expresión.

$$\frac{dA_T}{dt} = \frac{\dot{V}_{in,L} - \dot{V}_{out,L}}{H} \tag{22}$$

Ahora se puede expresar el cambio que sufre el área transversal de la caldera como:

$$\frac{dA_T}{dt} = \frac{dA_T}{dL}\frac{dL}{dt} \tag{23}$$

Derivando la ecuación 16 en función del nivel de líquido se obtiene:

$$\frac{dA_T}{dL} = \sqrt{2rL - L^2} + \frac{(L-r)(2r-2L)}{2\sqrt{2rL-L^2}} + \frac{r}{\sqrt{1-(\frac{L-r}{r})^2}} \tag{24}$$

Ahora reemplazando las ecuaciones 22 y 24 en la ecuación 23, despejando de esta última la derivada del nivel en el tiempo. Se llega a una expresión que representa el cambio de nivel del líquido en la caldera.

$$\frac{dL}{dt} = [\frac{\dot{V}_{in,L} - \dot{V}_{out,L}}{L}][\frac{1}{\sqrt{2rL-L^2} + \frac{(L-r)(2r-2L)}{2\sqrt{2rL-L^2}} + \frac{r}{\sqrt{1-(\frac{L-r}{r})^2}}}] \tag{25}$$

Obsérvese que se puede llegar al mismo resultado despejando de la ecuación 19 la derivada del nivel en el tiempo, y reemplazando en esta misma las ecuaciones 18 y 21.

Para determinar el flujo de líquido que se evapora, se realiza un balance térmico calculado aquí como el flujo inicial de vapor en la caldera, que se considera el de estado estacionario o arranque de la simulación:

$$\dot{V}_{out,L} = \frac{\dot{Q}_{req}}{\rho[C_{p,L}(T_{ebu} - T_{in}) + h_v]} = \dot{V}_{in,V} \tag{26}$$

donde el calor requerido Q_{req} es la variable manipulada para el sistema de presión, y el flujo de líquido que se evapora es $\dot{V}_{out,L}$ o $\dot{V}_{in,V}$ es perturbación para el sistema de nivel.

- A continuación, se deducirá el modelo matemático para el control de presión.

Lo primero es realizar un balance de materia para la masa de vapor contenida en la caldera:

$$\frac{dM_V}{dt} = \dot{m}_{in,V} - \dot{m}_{out,V} \tag{27}$$

Ahora de la ecuación de gases ideales para el vapor se tiene:

$$\rho_V = \frac{P_V \bar{M}}{RT} \tag{28}$$

Se sabe que el flujo másico se puede definir como el producto de la densidad por el flujo volumétrico:

$$\dot{M}_V = \rho_V \dot{V}_V \tag{29}$$

Reemplazando la ecuacion 28 en la ecuacion 29 se tiene que:

$$\dot{M}_V = \dot{V}_V \frac{P_V \bar{M}}{RT} \tag{30}$$

Derivando la ecuación (30) respecto al tiempo:

$$\frac{dM_V}{dt} = \frac{\bar{M}}{RT}[V_V \frac{dP_V}{dt} + P_V \frac{dV_V}{dt}] \tag{31}$$

Reemplazando la ecuación 31 en la ecuación 27, y despejando la derivada de la presión se llega a:

$$\frac{dP_V}{dt} = \frac{\frac{\bar{M}}{RT}(\dot{m}_{in,V} - \dot{m}_{out,V}) - P_V \frac{dV_V}{dt}}{V_V} \tag{32}$$

donde $V_V = V_{total} - V_L$. El flujo de vapor inicial se puede definir como el volumen de líquido que se evapora en la caldera. La siguiente ecuación quiere decir que el flujo de vapor que se genera es igual a la perdida de altura que posee el líquido dentro de la piscina de la caldera.

$$\frac{dV_V}{dt} = -H \frac{dA_T}{dt} \tag{33}$$

Derivando el área transversal en función respecto al tiempo, se obtiene:

$$\frac{dA}{dt} = \frac{dA}{dL}\frac{dL}{dt} \tag{34}$$

Reemplazando la ecuación 24 en la ecuación 34.

$$\frac{dA}{dt} = (\sqrt{2rL - L^2} + \frac{(L-r)(2r-2L)}{2\sqrt{2rL-L^2}} + \frac{r}{\sqrt{1-(\frac{L-r}{r})^2}})\frac{dL}{dt} \tag{35}$$

Sustituyendo la ecuación 35 en la ecuación 33:

$$\frac{dV_V}{dt} = -H(\sqrt{2rL - L^2} + \frac{(L-r)(2r-2L)}{2\sqrt{2rL-L^2}} + \frac{r}{\sqrt{1-(\frac{L-r}{r})^2}})\frac{dL}{dt} \tag{36}$$

Finalmente reemplazando la ecuación 3 en la ecuación 32, se obtiene una expresión matemática que representa el cambio de la presión de vapor en el tiempo como función del nivel:

$$\frac{dP_V}{dt} = \frac{\frac{\bar{M}}{RT}(\dot{m}_{in,V} - \dot{m}_{out,V}) - P_V[-H(\frac{(\sqrt{2rL-L^2}+(L-r)(2r-2L)}{2\sqrt{2rL-L^2}} + \frac{r}{\sqrt{1-(\frac{L-r}{r})^2}})\frac{dL}{dt}]}{V_{total} - V_L} \tag{37}$$

donde la derivada del nivel en el tiempo es la ecuación que se dedujo anteriormente para el sistema de nivel, ecuación 25. De esta manera se comprueba la relación matemáticamente entre los dos sistemas.

En la ecuación 26, como se sabe, el flujo de líquido evaporado es el mismo flujo inicial de vapor o flujo de vapor dentro de la caldera, y el flujo de salida de vapor es la perturbación. La variable de control para este sistema, es el calor requerido en la caldera, representado en la ecuación 26.

3.3.2. Programación del modelo de control de nivel de líquido y presión de vapor en una caldera en Python

A continuación se muestra la realización de los programas en ***Python*** utilizando su entorno interactivo web de ejecución de código ***Jupyter Notebook***, para así mostrar la aplicación que tienen los modulos en la solución de este tipo de sistemas, en este caso para el control de nivel de líquido y presión del vapor en una caldera pirotubular, es decir dos lazos de control para un sistema.

Para este sistema se programo la simulación para un controlador ***PID***. La finalidad de esto es mostrar lo intuitivo que puede ser la creación de este tipo de algoritmos, a diferencia del control de de nivel en un tanque y control de temperatura en un intercambiador, que se vieron anteriormente, este sistema posee dos lazos de control. En este ejemplo, se mostrara que asi el sistema posea varios lazos de control, el algoritmo creado funciona exactamente igual.

La programación para representar este sistema, se realiza exactamente igual a la programación de los sistemas anteriormente expuestos para controladores ***PID***, siguiendo el mismo orden del algoritmo creado modularmente.

Descripción del proceso

- Se supondrá una caldera de geometría cilíndrica ubicada horizontalmente.
- La variable de proceso 1, es la presión de vapor dentro de la caldera, que se regula con el flujo de energía ***Q_i.***
- La variable de proceso 2, es el nivel de la caldera, que se regula con el flujo de líquido que ingresa a la caldera ***FlujoM_Liq_in_i.***
- Se tiene en cuenta que la variación en el nivel depende del radio de la caldera porque al ser un cilindro horizontal su longitud ***H*** es constante.
- Los dos lazos de control comparten una variable, que es el flujo de líquido que se evapora. ***FlujoM_LiqEvap_i.***
- Al perturbar el sistema de presión sacando mas flujo de vapor a la salida, se debe aplicar una acción de control que suministre calor al sistema.
- Al suministrar calor al sistema, este evapora líquido, por ende baja el nivel, perturbando así el otro sistema, el de nivel.
- Al perturbar el sistema de nivel, se genera una acción de control que suministra flujo de líquido a la caldera.

Datos del proceso

Fluido de proceso = Agua líquida a T=25 $[C]$.

- Variable de proceso y_1 = Nivel de agua en la caldera ***Nivel_i*** $[m]$.
- Variable de proceso y_2 = Presión de vapor dentro de la caldera ***P_Vap_i*** $[bar]$.
- Variable manipulada para el "Nivel" (u) = Flujo de líquido que ingresa a la caldera ***FlujoM_Liq_in_i*** $[\frac{kg}{s}]$.
- Variable manipulada para la "Presión" (u) = Calor suministrado a la caldera desde una fuente externa ***Q_i*** $[kw]$.

- Variable de perturbación para el "Nivel" (d) = Flujo de líquido evaporado o flujo de vapor inicial ***FlujoM_LiqEvap_i*** $[\frac{kg}{s}]$.
- Variable de perturbación para la "Presión" (d) = Flujo de vapor que sale de la caldera ***FlujoM_Vap_out_i*** $[\frac{kg}{s}]$.

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 ### Libreria de python que permite realizar operaciones con␣
 ↪vectores.
import matplotlib.pyplot as pt# Libreria de python que permite realizar gráficas.
import math as mt  ##  Librería para usar funciones matemáticas.
```

MÓDULO 1- Parámetros y constantes

Parámetros del equipo

- En esta parte del módulo 1 se determinan los parámetros del equipo: la longitud horizontal de la caldera, el diámetro de esta, su radio, su volumen y los respectivos Set Points para ambos lazos de control.

[ ]:
```
H= 3  ##  Longitud de la caldera [m].
Diam= 1  ##  Diámetro de la caldera [m].
Rad= Diam/2  ##  Radio de la caldera [m]
Vol_Tot= H*mt.pi*(Diam/2)**2  ##  Volumen total de la caldera suponiendo␣
 ↪geometria cilindrica [m^3].
SP_N= Rad  ##  Set point para el nivel [m].
SP_P= 10  ##  Set point para la presión de vapor [Bar].
```

Parámetros de las sustancias y constantes

- En este paso del algoritmo se determinan los parámetros y constantes de las sustancias. En este caso se tienen la densidad, constante de los gases ideales, temperatura de ebullición del agua, masa molar, las constantes de la función del calor especifico tanto para vapor como líquido y las constantes de Antoine.

[ ]:
```
Dens= 1000  ##  Densidad del agua líquida [kg/m^3].
R= 8.314  ##  Constante de los gases [KJ/Kmol K].
Teb= 100+273.15  ##  Temperatura de ebullición del agua [K].
M= 18  ##  Masa molar del agua [kg/kmol].

Al= 8.712  ##  Constante A para el calculo del Cp (Líquido). (Tabla C3. Smith␣
 ↪Van Ness).
Bl= 1.25*1e-03  ##  Constante B para el Cp (Líquido). (Tabla C3. Smith Van Ness).
```

```
Cl= -0.18*1e-06  ##  Constante C para el Cp (Líquido). (Tabla C3. Smith Van␣
 ↪Ness).

Av= 3.470  ##  Constante A para el cálculo del Cp (Vapor). (Tabla C1. Smith Van␣
 ↪Ness).
Bv= 1.450*1e-03  ##  Constante B para el cálculo del Cp (Vapor). (Tabla C1.␣
 ↪Smith Van Ness).
Cv= 0  ##  Constante C para el cálculo del Cp (Vapor). (Tabla C1. Smith Van␣
 ↪Ness).
Dv= 0.121*1e5  ##  Constante D para el cálculo del Cp (Vapor). (Tabla C1. Smith␣
 ↪Van Ness).

A = 10.01  ##  Constante A para la ecuación de Antoine. (Propierties of gases␣
 ↪and liquids).
B = 1611  ##  Constante B para la ecuación de Antoine. (Propierties of gases and␣
 ↪liquids).
C = -51.4  ##  Constante C para ecuación de Antoine. (Propierties of gases and␣
 ↪liquids).
```

MÓDULO 2- Condiciones iniciales en estado estacionario

- En este módulo se le dan valores iniciales o de entrada a las variables de proceso, control y perturbación. Se debe indicar que los valores en que inician estas variables son los valores de estado estacionario puesto que se asume que el proceso inicia de esta manera.

[ ]:
```
#ini=Inicial, in=Entrada.**MODULO 2-  CONDICIONES INICIALES EN ESTADO␣
 ↪ESTACIONARIO**

T_in = 20+273.15  ##  Temperatura de entrada del líquido a la caldera [K].
Nivel_ini= Rad  ##  Nivel del tanque, este inicia suponiendo que la caldera es␣
 ↪horizontal y esta llena en un 50% [m].
P_Vapor_ini= SP_P ##  Presión de vapor inicial de la caldera [bar].
FlujoM_Vap_out= 1  ##  Flujo másico de vapor inicial que sale de la caldera [kg/
 ↪s].

FlujoM_Liq_in= 0.97849297  ##  Flujo másico de líquido inicial que ingresa a la␣
 ↪caldera (u) para el nivel [Kg/s].
FlujoM_L_Max= 2*FlujoM_Liq_in  ##  Flujo másico máximo que puede tener el␣
 ↪líquido al ingresar a la caldera [kg/s].

Q_ini= 2785.83  ##  Flujo de energía inicial (u) para la presión de vapor [kw].
Q_Max= 2*Q_ini  ##  Flujo de energía máxima que se puede ingresar al proceso␣
 ↪[kw].
```

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 asume se realiza 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 dividiéndolo en N intervalos ***N_Part***. En este caso se asumen que son intervalos de tiempo.
- La variable **"t_Pert"** es el tiempo en el que se perturba el sistema. Para este caso se inicia cuando ha pasado el 30 % del número total de particiones temporales.

```
[ ]: t_Inicial = 0  ##  Tiempo en que inicia la simulación [s].
     t_Final = 5000  ## Tiempo en que finaliza la simulación [s].
     Paso = 1  ## Tamaño de cada partición de intervalo de tiempo.
     N_Part= int((t_Final-t_Inicial)/(Paso))  ##  Determina el número de particiones␣
      ↪necesarias para resolver el método.
     tiempo = np.linspace(t_Inicial, t_Final, N_Part)  ##  Crea vector de tiempo para␣
      ↪poder graficarlo [s].
     Iter = N_Part  ##  Número de iteraciones, se puede decir que es igual al número␣
      ↪de particiones.
     t_Pert = int(0.3 * N_Part)  ##  Tiempo y valor en que se aplica la perturbación␣
      ↪[%].
```

MÓDULO 4- Creación de vectores para guardar datos

- Este módulo construye vectores de cierta dimensión para las variables de interés, como se puede ver son las variables de proceso, perturbaciones, variables de control, Set Point y el error para ambos lazos de control, estos vectores tienen la finalidad de guardar los datos de cada una de las soluciones del método.
- La librería ***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 zeros.
- Las dimensiones para los vectores las otorga ***len(tiempo)***, indicando que se va a crear un vector de zeros 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 automáticamente si se desea cambiar el valor del tiempo de simulación o paso.

```
[ ]: Nivel_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para el nivel en la␣
      ↪caldera, variable de proceso (y).
     P_Vap_i = np.zeros(len(tiempo))  ##  Crea un vector de datos para la presión de␣
      ↪vapor dentro de la caldera, variable de proceso (y).
     SP_N_i= np.zeros(len(tiempo))  ##  Crea un vector de datos para el Set Point del␣
      ↪nivel.
     SP_P_i= np.zeros(len(tiempo))  ##  Crea un vector de datos para el Set Point de␣
      ↪la presión de vapor.
     Q_i= np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para el calor o␣
      ↪energía suministrad, variable de control para la presión de vapor.
     FlujoM_Liq_in_i= np.zeros(len(tiempo)+1)  ##  Crea un vector de datos para el␣
      ↪flujo de líquido (variable de control).
     FlujoM_LiqEvap_i= np.zeros(len(tiempo))  ##  Crea un vector de datos para el␣
      ↪flujo másico de vapor o líquido que se evapora.
```

```
FlujoM_Vap_out_i= np.zeros(len(tiempo))  ##  Crea un vector de datos para el␣
 ↪flujo másico de vapor que sale de la caldera, perturbación para el control de␣
 ↪presión(d).
Vol_Liq_i= np.zeros(len(tiempo)) ## Crea un vector para el volumen que ocupa el␣
 ↪líquido en la caldera.
Vol_Vap_i= np.zeros(len(tiempo)) ##  Crea un vector de datos para el volumen que␣
 ↪ocupa el vapor en la caldera.
Error_N_i= np.zeros(len(tiempo))  ##  Crea un vector de datos para guardar el␣
 ↪error para el nivel.
Error_P_i= np.zeros(len(tiempo))  ##  Crea un vector de datos para guardar el␣
 ↪error para la presión.
```

MÓDULO 5- Set Point y perturbaciones

Set Point

- En este caso para el vector ***SP_N_i*** y ***SP_P_i***, que son vectores de ceros creados en el modulo 4, se les otorga un valor inicial "su valor de estado estacionario" en la primer posición del vector. Para Python la primer posición en un vector corresponde al indice i=0, por ende es recomendable que la variable ***t_Inicial*** inicie en el tiempo cero.
- Para este caso se tienen dos vectores que se inician en su valor de estado estacionario, ya que las dos variables hacen parte del mismo sistema "Caldera".

[ ]:
```
SP_N_i[t_Inicial:]=SP_N ##  Llena el vector del Set Point con el valor asignado␣
 ↪como referencia para el nivel.
SP_P_i[t_Inicial:]=SP_P ##  Llena el vector del Set Point con el valor asignado␣
 ↪como referencia para la presión de vapor.
```

Perturbaciones

- Ahora se realiza un cambio en escalon similar al del set point pero ahora con la variable de perturbación.
- La segunda linea significa que despues de iniciar la perturbación en su estado estacionario, al tiempo que se le dio a la perturbación para que iniciara en el modulo 3, aumente su valor en un 50 %. De esta manera se puede simular la perturbación del sistema con un vector en Python.

[ ]:
```
FlujoM_Vap_out_i[t_Inicial:] = FlujoM_Vap_out  ##  Inicia el valor de apertura␣
 ↪para la válvula del vapor de salida en estado estacionario [kg/s].
FlujoM_Vap_out_i[t_Pert:] = FlujoM_Vap_out*1.5 ## Cambia el valor de apertura de␣
 ↪la válvula de vapor a la salida. Tiempo en que inicia la perturbación t_Pert).
```

MÓDULO 6- PARÁMETROS PARA DE CONTROLADOR (PID)

- En este módulo se le asignan valores a los parámetros del controlador PID.
- ***e_1k*** y ***e_2k*** son los errores de la variable controlada en el tiempo pasado y en el tiempo antepasado. En el tiempo cero, estos errores son cero por iniciar en estado estacionario. Esto quiere decir que el Set Point y la variable de proceso arrancan en el mismo valor.
- **ITC** es el intervalo en que el controlador actualiza el valor de la acción de control.

Parámetros para el controlador de nivel de líquido **PID***

```
[ ]: Kp_N= 500  ##  Parámetro de ganancia proporcional [Acción de control que se␣
      ↪produce [m^3/s  / m].
     ti_N= 10 ##  Parámetro para el tiempo integral [s].
     td_N= 0.01  ##  Parámetro para el tiempo derivativo [s].
     e_1kN= 0  ## Error en el instante de tiempo anterior (i-1).
     e_2k= 0  ## Error en el instante de tiempo anterior (i-2).
```

Parámetros para el controlador de presión de vapor **PID***

```
[ ]: Kp_P= 0.05 ##  Parámetro de ganancia proporcional [kj/Bar].
     ti_P= 0.01 ##  Parámetro para el tiempo integral [s].
     td_P= 1  ##  Parámetro para el tiempo derivativo [s].
     e_1kP= 0  ##  Error en el instante de tiempo anterior (i-1).
     e_2kP= 0  ##  Error en el instante de tiempo anterior (i-2).
     ITC= 1  ##  Intervalo de tiempo de la acción de control [s].
```

MÓDULO 7- Actualización y llenado de los vectores para las variables del sistema

- Es este paso del algoritmo primero se inicializa las variables de control para el nivel del líquido ***Nivel_i*** y para la presión de vapor ***P_Vap_i*** con su valor inicial. En el lenguaje de programación es indicar la posición del vector y darle un valor. En este caso se le da un valor a la primer posición del vector de datos creado en el módulo 4.
- Para empezar con la solución del método numérico, se inicia el ciclo iterativo, este comienza en cero que indica la primer posición de un vector en Python, hasta la última posición del vector o número de iteraciones que las determina el valor dado en el módulo 3 para el número de particiones del vector tiempo.
- Las siguientes lineas para el nivel del líquido ***Nivel_i***, presión del vapor ***P_Vap_i***, flujo másico de líquido que ingresa al intercambiador ***FlujoM_Liq_in_i***, el calor ingresado a la caldera ***Q_i*** , el error en la medición del nivel ***Error_N_i***, y el error en la medición en la presión del vapor ***Error_P_i*** se realizan para que al finalizar cada ciclo con un nuevo valor, este ingrese de nuevo al ciclo **for**. De esta manera se resuelve el método con cada uno de los valores obtenidos al terminar cada ciclo, completando de esta manera el número de iteraciones, llenando asi cada uno de los vectores que representan a las variables de interes con los resultados de cada iteracion.

```
[ ]: # AQUÍ EMPIEZA LA SOLUCIÓN ITERATIVA DEL MODELO CON SU CONTROLADOR #

     Nivel=Nivel_ini  ##  Asigna el valor inicial al nivel del tanque para iniciar el␣
      ↪método.
     P_Vapor=P_Vapor_ini  ##  Asigna el valor inicial al la presión de vapor para␣
      ↪iniciar el método.

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

     # ACTUALIZACIÓN DE LAS VARIABLES DE PROCESO, CONTROL Y ERROR.
```

```
   Nivel_i[i] = Nivel  ##  Guarda los datos para cada solución de la Ecuación␣
↪diferencial de Temperatura.
   Error_N_i[i]= SP_N_i[i]-Nivel  ##  Guarda el error en cada iteración para el␣
↪control de nivel.
   FlujoM_Liq_in_i[t_Inicial]=FlujoM_Liq_in ## Inicia el valor de flujo de␣
↪líquido en estado estacionario (NIVEL).[kg/s].

   P_Vap_i[i] = P_Vapor  ##  Guarda los datos para cada solución de la Ecuación␣
↪diferencial de Presión de vapor.
   Error_P_i[i]=SP_P_i[i]-P_Vapor  ##  Guarda el error en cada iteración para␣
↪el control de presión de vapor.
   Q_i[t_Inicial]=Q_ini ## Inicia el valor de flujo de energía en estado␣
↪estacionario [kw].(PRESIÓN).
```

MÓDULO 8- Programación de ecuaciones

Ecuaciones constitutivas

- En esta parte del módulo, se solucionan y actualizan las ecuaciones constitutivas del modelo.
- Las siguientes líneas del fragmento del código, representan las ecuaciones para la temperatura de saturación ***Tsat*** que es simplemente la representación de la temperatura de saturación de Antoine que se encuentra en [1].
- La ecuación para cálcular el calor entregado por el vapor ***H_Vap***, es simplemente un polinomio que se ajusto a los datos para la entalpía de vapor saturado de la tabla A12 de [3].
- La siguiente línea representa la ecuación para los calores específicos tanto para el líquido ***Cp_L*** como para el vapor ***Cp_V*** , este es un polinomio sacado de la tabla C3 de [6].
- Las siguientes representaciones matemáticas calculan el volumen que ocupan en la caldera tanto el líquido como vapor, son sacadas de la deducción matematíca dada en este docuemnto.
- La ultima línea de código representa simplemente el flujo másico de líquido que se evapora ***FlujoM_LiqEvap_i*** y este se cálcula por medio del balance de energía térmica.
- La deducción del modelo matématico se puede ver en la sección 2.3.1 de este documento.

[ ]:
```
   Tsat= (((-B)/(np.log10(P_Vap_i[i])-A))-C)+273.15  ##  Soluciona y actualiza␣
↪la ecuación de Antoine(1888) para la temperatura de saturación.The propierties␣
↪of gases and liquids [K].
   H_Vap=-0.0006*P_Vapor**3+0.1689*P_Vapor**2-20.951*P_Vapor+2284.4  ## ␣
↪Polinomio que cálcula el calor latente en función de la Presión de saturación␣
↪[kj/kg].(Con datos de la tabla A12 de [3].

   Cp_L=(Al+(Bl*Tsat)+(Cl*Tsat**2))*((R)/(M))  ##  Soluciona y actualiza la␣
↪ecuación para el Cp del líquido [Kj/kg K].(Tabla C3. Smith Van Ness).
   Cp_V=(Av+(Bv*Tsat)+(Cv*Tsat**2)+(Dv*Tsat**-2))*((R)/(M))  ##  Soluciona y␣
↪actualiza la ecuación para el Cp del vapor [kj/kg K].(Tabla C1. Smith Van␣
↪Ness).
```

```
    Vol_Liq_i [i]=(((mt.pi*Rad**2)/(2))+(Nivel_i[i]-Rad)*(np.
↪sqrt(2*Rad*Nivel_i[i]-Nivel_i[i]**2))+(mt.asin(((-Rad+Nivel_i[i])/
↪(Rad))))*(Rad**2))*H  ##  Soluciona y actualiza el volumen que ocupa el␣
↪líquido dentro de la caldera [m^3].
    Vol_Vap_i[i]=Vol_Tot-Vol_Liq_i [i]  ##  Soluciona y actualiza el volumen que␣
↪ocupa el vapor dentro de la caldera [m^3].

    FlujoM_LiqEvap_i[i]=((Q_i[i])/((Cp_L*(Teb-T_in))+H_Vap+(Cp_V*(Tsat-T_in)))) ␣
↪##  Soluciona y actualiza la ecuacion de flujo másico para el vapor que␣
↪ingresa [kg/s].
```

Ecuaciones diferenciales

- En esta sección, se programa un método númerico para solucionar ecuaciones diferenciales. Por su simplicidad se programo el método Euler.
- Lo primero para resolver el método es despejar las derivadas de cada una de las ecuaciones que representan el modelo ***dLdt*** para el cambio en el nivel de líquido dentro de la caldera y ***dPdt*** que representa el cambio en la presión del vapor, las dos derivadas en función del tiempo.
- Realizado el paso anterior, se cálcula el nuevo valor para el nivel en la siguiente partición de tiempo ***Nivel*** y para la presión del vapor ***P_Vapor***. Esto se realiza sumandole al valor obtenido en la partición anterior para el nivel ***Nivel_ini*** y para la presión del vapor ***P_Vapor_ini*** el cambio (derivada) que tiene esta variable en el instante de tiempo anterior, multiplicado por el intervalo ***Paso*** de tiempo asignado en el módulo 3.
- Se debe observar tanto en el modelo matemático como en el código, que la derivada de la presión del vapor depende de la derivada que representa el cambio en el nivel del líquido, es decir que los dos sistemas de control tienen una relación.
- Finalmente, la última línea de código actualiza las variables de proceso en cada partición temporal para iniciar con un nuevo valor el método iterativo.

[ ]:
```
# ECUACIÓN DIFERENCIAL PARA EL NIVEL.

    dLdt=((FlujoM_Liq_in_i[i]-FlujoM_LiqEvap_i[i])/(Dens*H))*((1)/(np.
↪sqrt(2*Nivel*Rad-Nivel**2)+((Nivel-Rad)*(2*Rad-2*Nivel)/(2*np.
↪sqrt(2*Rad*Nivel-Nivel**2)))+((Rad)/(np.sqrt(1-((-Rad+Nivel)/(Rad))**2))))) ␣
↪##  Soluciona la derivada de la ED para el nivel.
    Nivel=Nivel_ini+dLdt*Paso  ##  Soluciona la ecuación diferencial,␣
↪agregandole al nivel inicial el cambio que tiene esta en una pequeña partición␣
↪o instante de tiempo (Paso).
    Nivel_ini=Nivel  ##  Actualiza el nivel para iniciar con el siguiente paso␣
↪[m].

# ECUACIÓN DIFERENCIAL PARA LA PRESIÓN DE VAPOR.
```

```
    dPdt=(((R*Tsat)/
↪M)*(FlujoM_LiqEvap_i[i]-FlujoM_Vap_out_i[i])+P_Vapor*(H*dLdt*(((np.
↪sqrt(2*Rad*Nivel-Nivel**2))+(Nivel-Rad)*(2*Rad-2*Nivel))/(2*np.
↪sqrt(2*Rad*Nivel-Nivel**2)))+((Rad)/(np.sqrt(1-(((-Rad+Nivel)/(Rad))**2))))))/
↪(Vol_Vap_i[i]*100)  ##  Soluciona la derivada de la ED para la presión de␣
↪vapor.
    P_Vapor=P_Vapor_ini+dPdt*Paso  ##  Soluciona la ecuación diferencial,␣
↪agregandole a la presión de vapor inicial el cambio que tiene esta en una␣
↪pequeña partición o instante de tiempo (Paso).
    P_vapor_ini=P_Vapor  ##  Actualiza la presión de vapor para iniciar con el␣
↪siguiente paso [Bar].
```

MÓDULO 9- Programación de las acciones de controlador (PID)

- Las acciones descritas a continuación se aplican para ambos controladores.
- En este módulo se escogerá el tipo de controlador (PID) a usar, y se programan las acciones del controlador.
- El primer condicional ***if*** indica la acción a realizar en caso de requerir un controlador (P). Como se puede ver esto depende de los valores que se le dan a los parámetros integral ***ti*** y derivativo ***td***.
- El condicional **else** funciona en caso de requerir un controlador (PID). Como se puede ver lo único que cambia es la ecuación del controlador.
- Las siguientes dos lineas actualizan los errores penúltimo y antepenúltimo, que son requeridos en las ecuaciones de los controladores.
- Para las acciones de este controlador, el primer condicional **if** mide el valor que posee la variable de control más el valor calculado en la ecuación del controlador. Si este valor es mayor al valor máximo permitido, la siguiente línea de código lo condiciona para que se mantenga en este valor máximo ***FlujoM_L_Max*** y también ocurre lo mismo con el sistema de presión de vapor, si no que acá se mantiene el flujo máximo de calor permitido ***Q_Max***.
- El condicional **elif** quiere decir en Python "de lo contrario si" es como un condicional **if** y **else** combinados. Este da la acción contraria a la variable de control, es decir, mide si la variable de control se encuentra por debajo de su valor mínimo permitido y lo mantiene en este valor.
- Por último, el condicional **else** actualiza el valor de la válvula de control dandole valores intermedios de apertura a la válvula de control, ya que este tipo de válvula si puede tomar valores diferentes de 0 % y 100 %, así que puede regular su apertura a diferentes posiciones, lo que no ocurre con las válvulas On-Off, que solo poseen dos posiciones.

Control de nivel de líquido

[ ]:
```
# CONTROLADOR (P) PARA EL NIVEL.

    if (td_N==0 and ti_N>=9999):  ##  Control (P), depende de los parametros td␣
 ↪y ti.
        Delta_u = Kp_N*ITC*Error_N_i[i]  ##  Ecuación para el control (P).

# CONTROLADOR (PID) PARA EL NIVEL.
    else:
```

```
        Delta_u= Kp_N*ITC*(Error_N_i[i]-e_1kN) + Kp_N*(ITC/ti_N)*e_1kN +␣
↪Kp_N*(td_N/ITC)*(Error_N_i[i]-2*e_1kN+e_2k)  ##  Ecuación para el control␣
↪(PID)

# ACTUALIZACION DEL ERROR.

    e_1kN=Error_N_i[i]  ##  Actualiza el error anterior para el nivel (i-1).
    e_2k=e_1kN  ##  Actualiza el error anterior para el nivel (i-2).

# ACCIÓN DE CONTROL PARA EL NIVEL.

    if (FlujoM_Liq_in_i[i]+Delta_u)>FlujoM_L_Max:
        FlujoM_Liq_in_i[i+1]=FlujoM_L_Max  ##  Corrige la acción de control para␣
↪que no pase de su límite máximo.

    elif (FlujoM_Liq_in_i[i]+Delta_u)<0:
        FlujoM_Liq_in_i[i+1]=0  ##  Corrige la acción de control para que no␣
↪pase de su límite mínimo.

    else:
          FlujoM_Liq_in_i[i+1]=FlujoM_Liq_in_i[i]+Delta_u  ##  Acción de control␣
↪dentro de los límites establecidos, esta se actualiza normalmente sumandole el␣
↪cambio.

    if Nivel_i[i]>=Diam*0.75:  ##  Corrige el nivel cuando se alcanza su máximo␣
↪permitido.
        Nivel_i[i]=Diam*0.75
        print("Nivel e líquido máximo de permitido")
        continue

    elif Nivel_i[i]<0:  ##  Corrige el nivel cuando se alcanza su mínimo␣
↪permitido.
        Nivel_i[i]=0
        print("Nivel de líquido mínimo  permitido")
        continue

    else:
        Nivel_i[i]=Nivel  ##  El nivel esta dentro de los límites pertimitos.
```

Control de presión del vapor

[ ]:
```
# CONTROLADOR (P) PARA LA PRESIÓN DE VAPOR.

    if (td_P==0 and ti_P>=9999):  ##  Control (P), depende de los parametros td␣
↪y ti.
        Delta_uP = Kp_P*ITC*Error_P_i[i]  ##  Ecuación para el control (P).
```

```
# CONTROLADOR (PI),(PD) Y (PID) PARA LA PRESIÓN DE VAPOR.

    else:
        Delta_uP= Kp_P*ITC*(Error_P_i[i]-e_1kP) + Kp_P*(ITC/ti_P)*e_1kP +␣
 ↪Kp_P*(td_P/ITC)*(Error_P_i[i]-2*e_1kP+e_2kP)  ##  Ecuación para el control␣
 ↪(PID)

# ACTUALIZACION DEL ERROR.

    e_1kP=Error_P_i[i]  ##  Actualiza el error anterior para la presión de vapor␣
 ↪(i-1).
    e_2kP=e_1kP  ##  Actualiza el error anterior para la presión de vapor (i-2).

# ACCIÓN DE CONTROL PARA LA PRESIÓN DE VAPOR.

    if (Q_i[i]+Delta_uP)>Q_Max:
        Q_i[i+1]=Q_Max  ## Corrige la acción de control para que no pase de su␣
 ↪límite máximo.

    elif (Q_i[i]+Delta_uP)<0:
        Q_i[i+1]=0  ##  Corrige la acción de control para que no pase de su␣
 ↪límite mínimo.

    else:
        Q_i[i+1]=Q_i[i]+Delta_uP  ##  Acción de control dentro de los límites␣
 ↪establecidos, se actualiza normalmente sumandole el cambio.

    if P_Vap_i[i]<=0.04:
        P_Vap_i[i]=0.04  ##  Corrige la presión de vapor cuando se alcanza su␣
 ↪mínimo permitido.
        print("Presión de vapor mínima permitida")
        continue

    elif P_Vap_i[i]>150:  ##  Corrige la presión de vapor cuando se alcanza su␣
 ↪máximo permitido.
        P_Vap_i[i]=150
        print("Presión de vapor máxima permitida")
        continue

    else:
        P_Vap_i[i]=P_Vapor  ##  La presión de vapor esta dentro de los límites␣
 ↪pertimitos.

FlujoM_Liq_in_i=FlujoM_Liq_in_i[:i+1]  ##  Ajusta el vector a las dimensiones␣
 ↪requeridas.
Q_i=Q_i[:i+1]   ##  Ajusta el vector a las dimensiones requeridas.
```

MÓDULO 10- Graficas y resultados

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

Gráfica: Presión de vapor vs Tiempo

[ ]:
```
pt.figure("MODELO CALDERA (CONTROL PID)", [12,7])  ##  Crea figura y le otorga␣
 ↪dimensiones.

pt.subplot(3,3,1)  ##  Hace una sub-figura (N°filas , N°columnas , Posicion de␣
 ↪la figura en el subplot) #|4 5 6|#.
pt.tight_layout(pad=5, w_pad=7, h_pad=4)  ##  Espaciado entre las figuras del␣
 ↪subplot (Espaciado entre la margen , Espaciado entre figuras (Horizontal) ,␣
 ↪Espaciado entre figuras (Vertical).
pt.plot(tiempo, P_Vap_i, "b", linewidth=2)  ##  Grafica los datos requeridos (x␣
 ↪, y , Color de la linea , Grosor de la linea).
pt.plot(tiempo, SP_P_i, "y--", label="Set point", linewidth=1.5)  ##  Grafica el␣
 ↪Set Point.
pt.grid(True)  ##  Agrega la cuadrilla a la grafica.
pt.xlabel("Tiempo [s]")  ##  Agrega título al eje x.
pt.ylabel("Presión \nde vapor [Bar]")  ##  Agrega título al eje y.
pt.title("Variables de proceso")
pt.ylim(min(P_Vap_i)-0.1,max(P_Vap_i)+0.1)  ##  Ajusta los límites en el eje (y)␣
 ↪para mejor visualización.
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).
pt.legend(loc="best")  ##  Agrega la leyenda en la grafica.("best" es la mejor␣
 ↪ubicación).
```

Gráfica: Flujo másico que sale de la caldera vs Tiempo

[ ]:
```
pt.subplot(3, 3, 2)
pt.plot(tiempo, FlujoM_Vap_out_i, "g", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Flujo de vapor\n salida [kg/s]")
pt.title("Perturbaciónes")
pt.ylim(min(FlujoM_Vap_out_i)-0.1,max(FlujoM_Vap_out_i)+0.1)
pt.tick_params(labelsize=8)
```

Gráfica: Potencia vs Tiempo

[ ]:
```
pt.subplot(3, 3, 3)
pt.plot(tiempo, Q_i, "b", linewidth=2)
```

```
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Flujo de\n calor [kw]")
pt.ylim(min(Q_i)-100,max(Q_i)+100)
pt.title("Variables de control")
pt.tick_params(labelsize=8)
```

Gráfica: Nivel vs Tiempo

[ ]:
```
pt.subplot(3, 3, 4)
pt.plot(tiempo, Nivel_i, "k", linewidth=2)
pt.plot(tiempo, SP_N_i, "y--", label="Set point", linewidth=1.5)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Nivel de \nlíquido [m] ")
pt.ylim(min(Nivel_i)-0.0001,max(Nivel_i)+0.0001)
pt.tick_params(labelsize=8)
pt.legend(loc="best")
```

Gráfica: Flujo másico de líquido que se evapora vs Tiempo

[ ]:
```
pt.subplot(3, 3, 5)
pt.plot(tiempo, FlujoM_LiqEvap_i, "y", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Flujo de líquido\nevaporado [kg/s]")
pt.ylim(min(FlujoM_LiqEvap_i)-0.1,max(FlujoM_LiqEvap_i)+0.1)
pt.tick_params(labelsize=8)
```

Gráfica: Flujo másico de líquido vs Tiempo

[ ]:
```
pt.subplot(3, 3, 6)
pt.plot(tiempo, FlujoM_Liq_in_i, "c", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Flujo de\n líquido [kg/s]")
pt.ylim(min(FlujoM_Liq_in_i)-0.1,max(FlujoM_Liq_in_i)+0.1)
pt.tick_params(labelsize=8)
```

Gráfica: Error de la presión del vapor vs Tiempo

[ ]:
```
pt.subplot(3, 3, 7)
pt.plot(tiempo, Error_P_i, "r", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Error Presión [Bar]")
pt.title("Error de Presión")
pt.ylim(min(Error_P_i)-0.1,max(Error_P_i)+0.1)
```

```
pt.tick_params(labelsize=8)
```

Gráfica: Error en el nivel del líquido vs Tiempo

[ ]:
```
pt.subplot(3, 3, 8)
pt.plot(tiempo, Error_N_i, "g", linewidth=2)
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Error Nivel [m]")
pt.title("Error de Nivel")
pt.ylim(min(Error_N_i)-0.0001,max(Error_N_i)+0.0001)
pt.tick_params(labelsize=8)
```

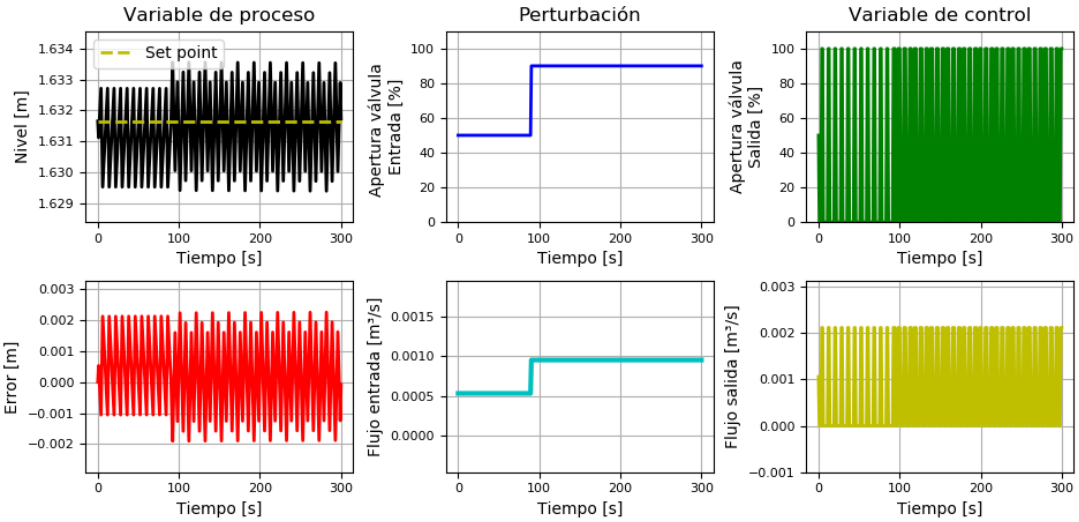
Gráfica: Volumen tanto del líquido como del vapor dentro de la caldera vs Tiempo

[ ]:
```
pt.subplot(3, 3, 9)
pt.plot(tiempo, Vol_Vap_i, "r", linewidth=2,label="Vapor")
pt.plot(tiempo, Vol_Liq_i, "b", linewidth=2,label="Líquido")
pt.grid(True)
pt.xlabel("Tiempo [s]")
pt.ylabel("Volumen \ncaldera [m\N{superscript three}]")
pt.title("Volumen Liq-Vap en la caldera")
pt.tick_params(labelsize=8)
pt.legend(loc="best")
```

4. Resultados

4.1. Resultados de la simulación para el control de nivel de líquido en un tanque

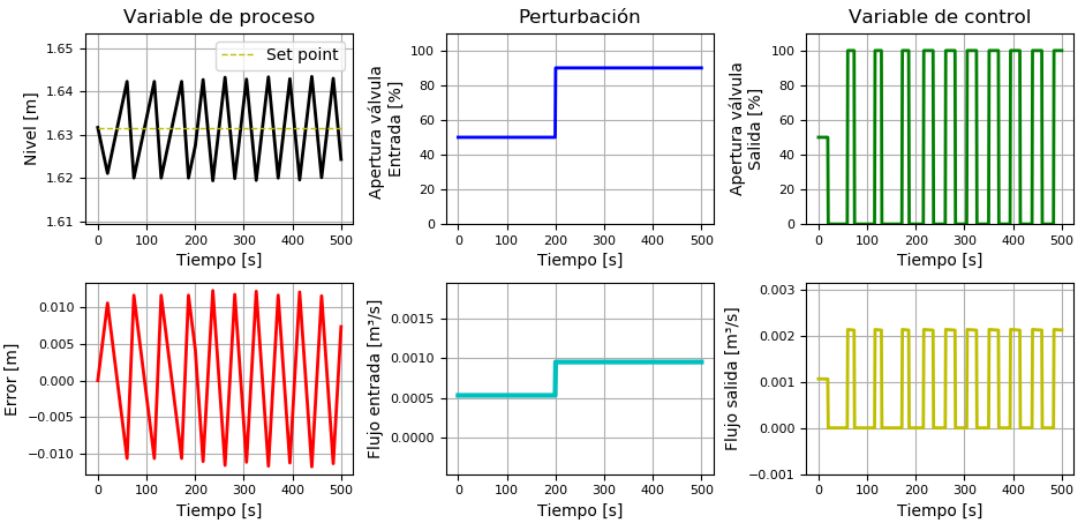
Figura 1: Control On-Off para el nivel de líquido en un tanque sin zona muerta



La figura 1 muestra seis gráficas, donde se puede observar el comportamiento de las variables que actúan en el proceso. En este caso un controlador (On-Off) para el nivel de líquido contenido en un tanque.

Al llegar a los 200 segundos de simulación se perturba el sistema y la válvula de control oscila con mayor frecuencia, esto se puede evidenciar en la figura 1 en la gráfica que representa la variable de control, donde esta oscilación puede verse gráficamente con mayor claridad.

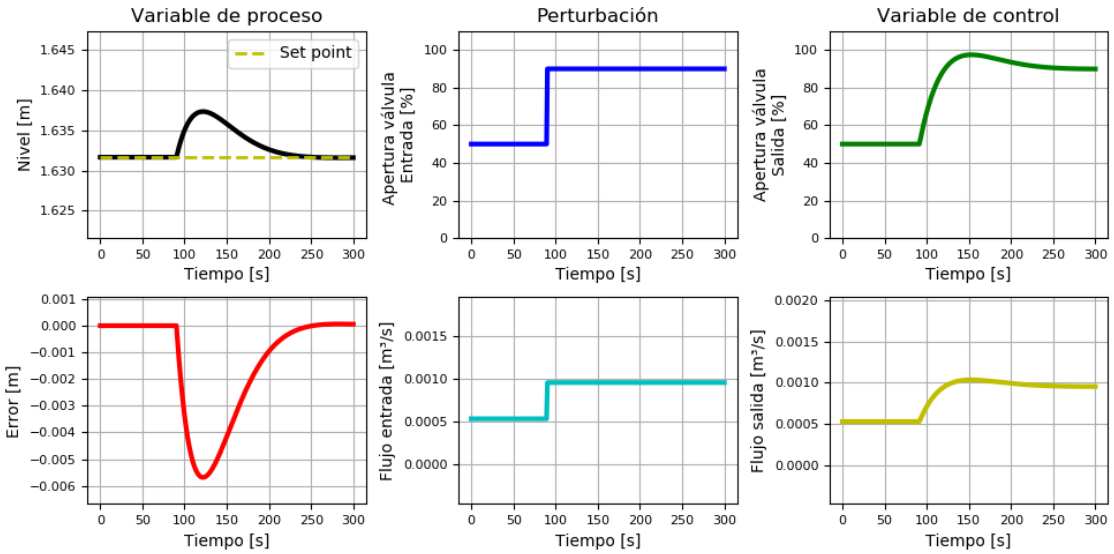
Figura 2: Control On-Off para el nivel de líquido en un tanque con zona muerta



La figura 2 muestra el mismo proceso que muestra la figura 1 pero este posee zona muerta, como se puede observar el controlador funciona adecuadamente, ya que la zona muerta es aproximadamente 0.01 m. La variable de proceso se mantienen en el rango de zona muerta aun perturbando el sistema a los mismos 200 segundos.

La variable de control muestra en su representación gráfica que al perturbar el sistema la frecuencia de apertura de la válvula de control es mayor a la que tenía antes de ser perturbado el sistema. Además de permanecer más tiempo cerrada que abierta, pero al ser perturbado el sistema parece que este tiempo se iguala.

Figura 3: Control PID para el nivel de líquido en un tanque

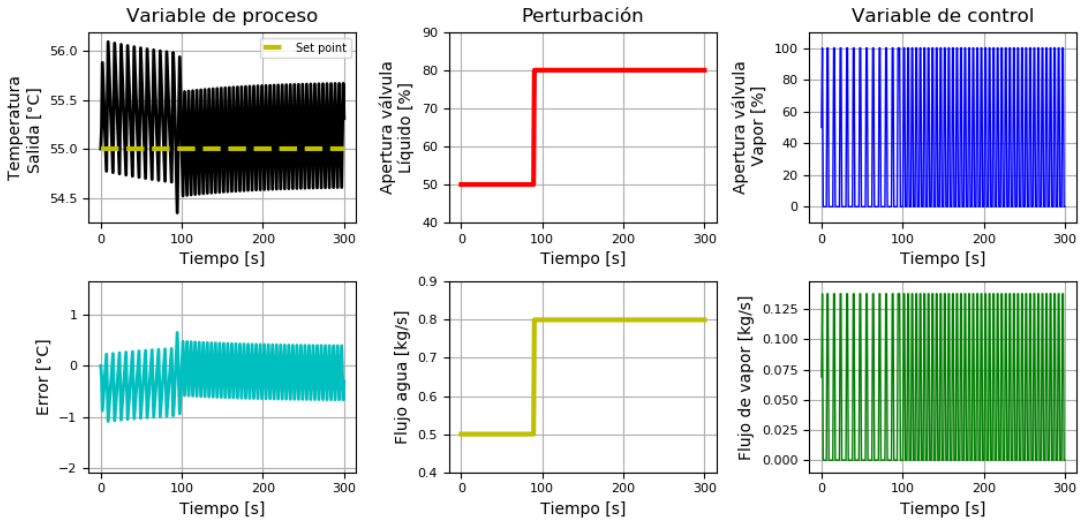


La figura 3 muestra el comportamiento para el control de nivel pero en un controlador PID. Se puede ver que al perturbar el sistema, este tarda aproximadamente 100 segundos en volver a su punto de ajuste (Set Point). Este tipo de controladores puede regular su acción de control en infinitas posiciones por eso la característica suave de la curva en comparación con los controladores en On-Off.

También se puede notar que al perturbar el sistema, aumentando el flujo de entrada de líquido, la variable de control responde adecuadamente y también aumenta el flujo de salida, esto permite que la variable de proceso pueda volver a su (Set Point).

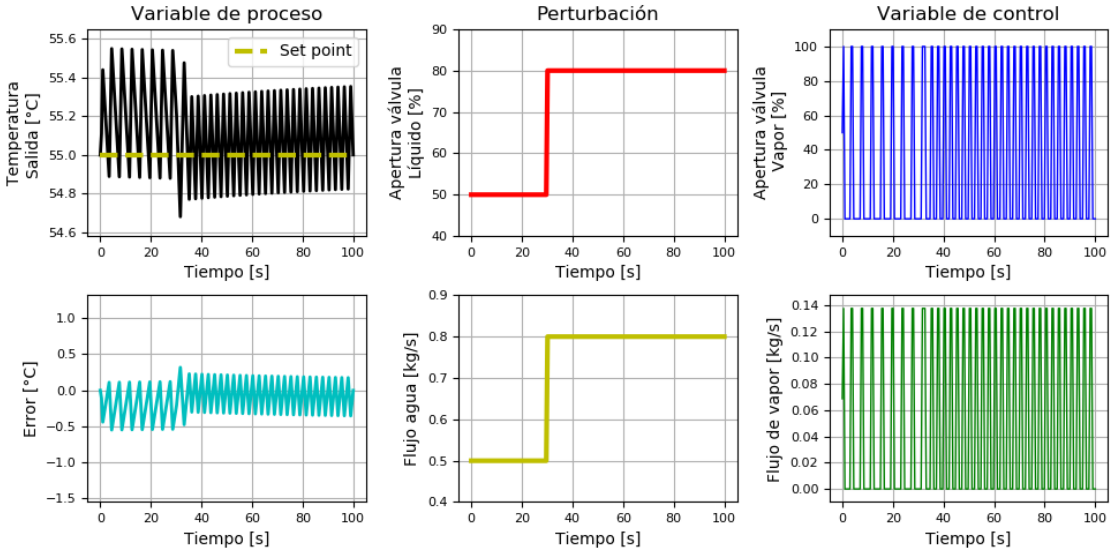
4.2. Resultados de la simulación para el control de temperatura en un intercambiador de calor

Figura 4: Control On-Off para la temperatura de líquido en un intercambio de calor sin zona muerta



La figura 4 se puede observar el comportamiento de un control On-Off para el intercambiador de calor. La gráfica para la variable de proceso muestra que antes de ser perturbada esta se mantenía por encima del (Set Point). Después de la perturbación, se mantiene el valor de la temperatura de la variable de control en un rango por encima y por debajo del (Set Point). También, se evidencia una mayor frecuencia en la apertura de la válvula de control, ya que se necesita mayor flujo de vapor para que el sistema responda.

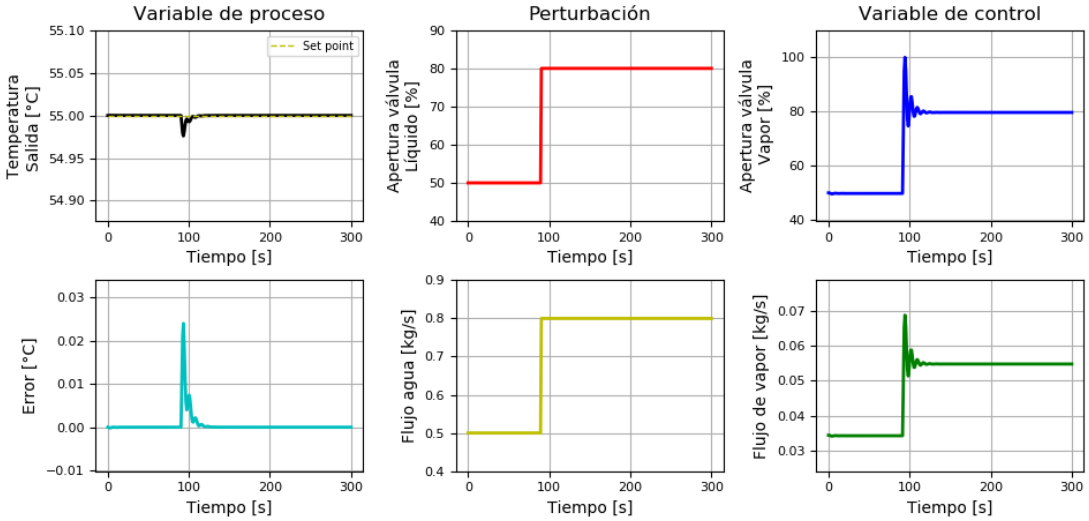
Figura 5: Control On-Off para la temperatura de líquido en un intercambio de calor con zona muerta



En la figura 5, se puede ver la representación del modelo para el controlador On-Off pero con zona muerta. El valor de esta es aproximadamente 1°C. Antes de iniciar la perturbación del sistema, la variable de control se mantiene por encima del (Set Point) en un rango aceptable para la zona muerta. Posteriormente, se perturba el sistema y la válvula de control oscila con mayor frecuencia. Si este se comparará con el control On-Off sin zona muerta, posee menos oscilaciones en la variable de control. Esto porque al haber zona muerta hay un mayor valor de tolerancia en la medición.

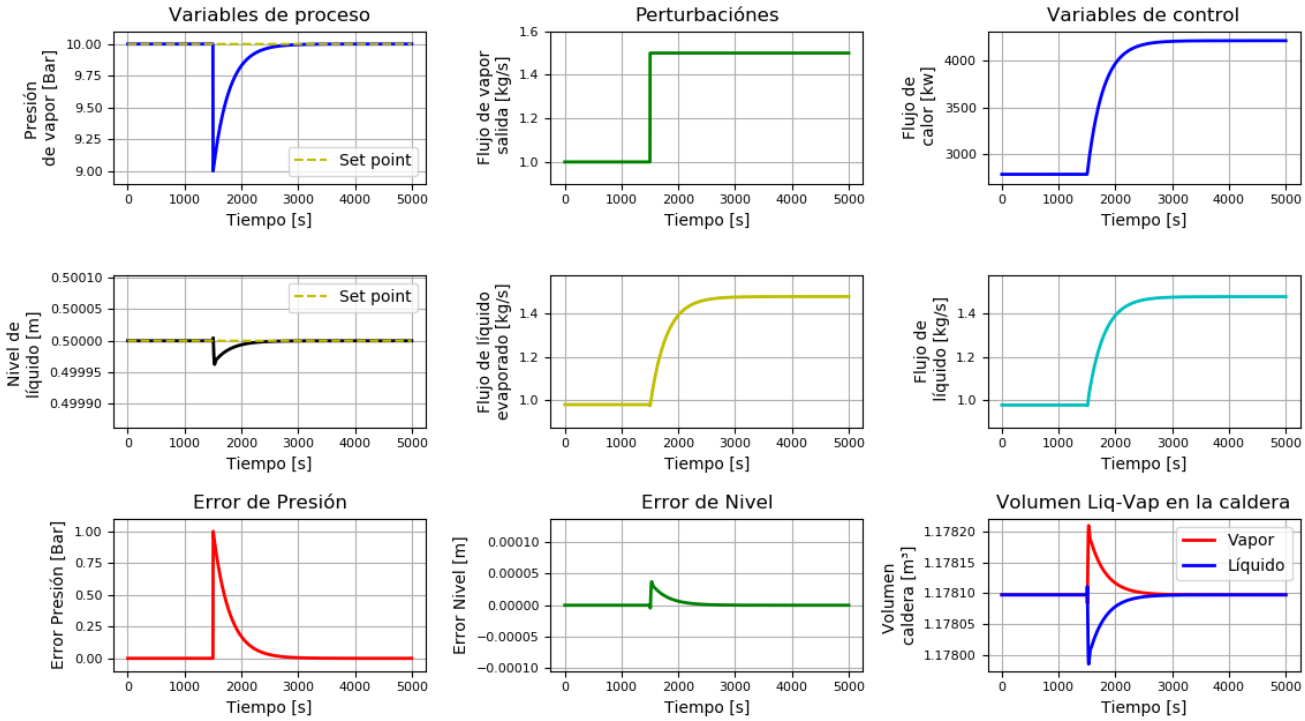
4.3. Resultados de la simulación para el control de nivel de líquido y presión del vapor en una caldera pirotubular

Figura 6: Control PID para la temperatura de líquido en un intercambio de calor



La figura 6, muestra el comportamiento de las variables de sistema en un controlador PID para un intercambiador de calor. Este se perturba aproximadamente a los 90 segundos de simulación. Se puede ver que el controlador responde rapidamente y mantiene la variable de proceso en su valor de (Set Point) en menos de diez segundos.

Figura 7: Control PID para el nivel de líquido y presión del vapor en una caldera pirotubular



La figura 7, muestra 9 gráficas. En esta se representan todas las variables del sistema para los controladores de la caldera, tanto el controlador para el nivel de líquido como para el controlador de presión del vapor dentro de la caldera. Se puede ver que hay tres columnas, cada columna representa las variables de proceso, perturbaciones y variable de control. La primer fila representa las variables para el controlador de presión del vapor. La segunda, representa las variables para el controlador de nivel de líquido.

Se puede observar que el sistema responde adecuadamente si este es perturbado, cuando la perturbación del sistema de control para la presión del vapor se perturba, se aumenta el flujo de calor y por este motivo el flujo de líquido que se evapora aumenta. Pero la variable de control del líquido rápidamente aumenta el flujo de líquido para mantener el nivel de este, y así al mismo tiempo la presión del vapor en la caldera. Según la simulación, después de ser perturbado el sistema, se tardará más de 2500 segundos.

También se muestra una gráfica que permite comparar en el tiempo el volumen que ocupan las dos fases del líquido dentro de la caldera.

5. Conclusiones

- En este trabajo se logró crear un algoritmo el cual fue capaz de simular sistemas de control realimentado. Partiendo de tres sistemas de proceso distintos se resolvieron los modelos de la misma manera, concluyendo así que se podrían simular más sistemas siguiendo los pasos del algoritmo creado.
- Como se pudo evidenciar en la programación de cada uno de los simuladores, la secuencia es la misma no importa si el controlador es On-Off o PID. La única diferencia que existe es que ambos poseen parámetros y acciones distintas, pero la solución es la misma para ambos controladores. Hay que tener en cuenta que la manera que se cálcula la acción de control es totalmente diferente.
- Se pudieron identificar fácilmente las diferencias que tienen los tipos de controlador, diferencias como en sus parámetros, acciones o ecuaciones en el caso del PID.
- Se podría decir que este documento sirve de guía para programar y simular sistemas de control, no solo para los casos de estudio propuestos, sino también para otros sistemas.
- En los resultados de las simulaciones para el controlador en On-Off el comportamiento de la válvula de control es diferente dependiendo de si se desea o no con zona muerta. Para el controlador con zona muerta, la válvula de control se acciona menos veces, ya que el controlador no mueve la válvula mientras la variable de proceso se encuentre en la zona muerta.
- El valor del paso en los controladores, principalmente en los On-Off, afecta el comportamiento de las simulaciones, esto debido a que a un intervalo de tiempo más corto, el método es más preciso pero el número de iteraciones aumenta.
- El valor para el tiempo de muestreo del controlador en los PID, también influye en las simulaciones. Numéricamente se puede definir que este tome acciones cada uno, medio, o menos segundos. Por lo general esto no ocurre porque sería obtener datos cada segundo lo que induce a gasto en la memoria de los equipos de cómputo, por tal razón, en la industria los intervalos de tiempo son mayores.

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.

[3] KENNETH,W.JR.RICHARDS, D.E «Termodinámica. »,McGRAW-HILL/ INTERAMETICANA DE ESPAÑA,S.A.U.,2001,Tabla. A-12,.

[4] SMITH,J.M,VAN NESS ,M.M «Introducción a la termodinámica en la Ingeniería Química tercera edición. »,McGRAW-HILL/ INTERAMETICANA DE ESPAÑA,S.A.U.,2007 ,Tabla. C1-C4 .

[5] GUZMAN,M.A,ORTIZ,L.V,PATIÑO,A.F,PEÑA,A,PINEDA,C.M «Análisis dinámico y sistemas de cotrol para una caldera pirotubular, Universidad Nacional de Colombia sede Medellín.,Introducción al control de procesos».

[6] IDARRAGA,J.M,LUNA,F.A,MIRANDA,S,RODRIGUEZ,M. «Representación, análisis dinámico de operación y sistemas de control de múltiples lazos para una caldera pirotubular. Universidad Nacional de Colombia sede Medellín.,Introducción al control de procesos. »