UNIVERSITAT JAUME I

ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS

GRADO EN INGENIERÍA ELÉCTRICA

Desarrollo de herramientas libres para el análisis e identificación de sistemas y el ajuste de controladores PID

TRABAJO FIN DE GRADO

AUTOR: Alejandro Martínez Soler

DIRECTOR: Ignacio Peñarrocha Alós

Castellón, Septiembre de 2020

Resumen

En este proyecto se desarrollarán un total de cuatro aplicaciones informáticas. Estas aplicaciones pertenecen al ámbito de la ingeniería de control, permitiendo tanto la identificación de sistemas y el estudio de su comportamiento, o la optimización de un PID.

Estas aplicaciones están desarrolladas en el lenguaje de programación Python y son de uso libre. Nuevas aplicaciones, o siguientes versiones de las ya desarrolladas, pueden ser encontradas en la siguiente página web: https://github.com/AlexMartinez95/

Una de las aplicaciones posee el nombre de "Obtencion-de-una-f.d.t". Esta aplicación permite ajustar de manera manual una función de transferencia mediante la inserción de datos experimentales. La aplicación hace una gráfica con los datos insertados, el usuario posee unas deslizaderas que dibujan una segunda gráfica, estas deslizaderas poseen las variables típicas de una función de transferencia. Se muestra un valor del error entre ambas gráficas para determinar el grado de similitud entre ambas.

La siguiente aplicación recibe el nombre de "Fdt-ante-diferentes-entradas". Escribiendo una función de transferencia en la aplicación, se pueden elegir diferentes entradas (U(s)) para observar su comportamiento.

La tercera en nombrar, será la denominada "Representa_la_inversa_de_laplace", de nuevo, escribiendo la función de transferencia se puede obtener la representación gráfica de la inversa de Laplace de la salida de la misma, es decir, y(t), además también se puede obtener la derivada de y(t) y su integral.

La última aplicación, "Respuesta-de-PID" simulará un sistema en bucle cerrado con un controlador P, PD, PI o PID en función de lo que el usuario seleccione, se muestra gráficamente el comportamiento ante cambio de referencia escalón y la respuesta ante perturbaciones, los parámetros de los controladores son modificables para obtener la respuesta deseada; para ello y además de la representación gráfica, se calculan y dibujan por pantalla diversas variables, tales como el IAE, el tiempo de establecimiento, o la sobreoscilación.

Abstract

In this project, a total of four computer applications will be developed. These applications belong to the field of control engineering, allowing both the identification of systems and the study of their behavior, or the optimization of a PID.

These applications are developed in the Python programming language and are free to use. New applications, or following versions of those already developed, can be found on the following web page: https://github.com/AlexMartinez95

One of the applications has the name "Obtencion-de-una-f.d.t". This application allows you to manually adjust a transfer function by inserting experimental data. The application makes a graph with the inserted data, the user has some sliders that draw a second graph, these sliders have the typical variables of a transfer function. An error value between both graphs is shown to determine the degree of similarity between both.

The following application is called "Fdt-ante-diferentes-entradas". By writing a transfer function in the application, different inputs (U (s)) can be chosen to observe its behavior.

The third to be named, will be called "Representa_la_inversa_de_laplace", again, writing the transfer function you can obtain the graphical representation of the Laplace inverse of its output, that is, y(t), in addition you can also obtain the derivative of y(t) and its integral.

The last application, “Respuesta-de-PID” will simulate a closed-loop system with a P, PD, PI or PID controller depending on what the user selects, it graphically shows the behavior before step reference change and the response before disturbances, the parameters of the controllers are modifiable to obtain the desired response; for this and in addition to the graphic representation, various variables are calculated and drawn on the screen, such as the IAE, the settling time, or the overshoot.

Índice general

Índice de figuras

Índice de tablas

Memoria

Índice memoria

1. Objeto

El objetivo de este proyecto es desarrollar una serie de aplicaciones informáticas en Python con utilidades en el ámbito de la ingeniería de control, especializadas en la identificación y optimización del control de sistemas.

Este código será compartido en una página web donde, dadas las características del software libre Python, otros desarrolladores pueden intervenir para mejorar las aplicaciones desarrolladas. La página web donde se compartirá el código se llama github.com. E

La justificación de este proyecto reside en que este tipo de páginas permite el acceso al código a cualquier persona del mundo, con el único requisito de tener conexión a internet, este hecho facilita las facultades didácticas e industriales de las aplicaciones dada su simplicidad de distribución; además de abaratar los costes ya que todas estas herramientas son de uso totalmente gratuito, en comparación a aplicaciones de pago de similares características.

2. Alcance

El ámbito de este proyecto es la ingeniería eléctrica en la rama de control de sistemas, estas aplicaciones pueden obtener una función de transferencia a través de unos datos experimentales y calcular el error entre los datos y la función de transferencia, también pueden optimizar las características de un controlador P, PD, PI y PID según distintas medidas calculadas mediante la aplicación. Pueden calcular la inversa de Laplace de una función de transferencia, dibujándola por pantalla. Por último, dada una función de transferencia, se puede obtener en función de cualquier entrada deseada, su salida.

Estas aplicaciones poseen algunas limitaciones, solo pueden leer archivos de texto con el nombre “datos”, su funcionamiento es manual y no pueden calcular controladores del tipo PID en función del valor final de unos datos experimentales, otra de sus limitaciones es que no poseen la opción de control representado mediante frecuencias.

3. Antecedentes

Los PID son muy relevantes en cualquier industria ya que de ellos depende el control de múltiples sistemas, y no siempre es sencillo optimizar el comportamiento del PID sin una herramienta externa.

Hoy en día existen varios software potentes que ya tienen implementadas funcionalidades como las que aquí se pretende desarrollar, sin embargo estos servicios no son gratuitos y no todas las empresas pueden pagar el precio de los mismos, o simplemente puede no ser rentable comprarlos; un ejemplo de esta situación es el conocido Matlab, cuya licencia cuesta 2000€ actualmente.

Actualmente, existen varios países en vías de desarrollo que utilizan controladores, por ello, el software libre puede ayudar a dichos países a acelerar su crecimiento, además, también abarata la enseñanza haciéndola más homogénea, fomentando la igualdad.

Desde el departamento de ingeniería eléctrica en la rama de automatización y control de sistemas, se propuso la idea de desarrollar en un software libre, aplicaciones que no fueran de uso exclusivo para una empresa. El software libre consta de múltiples ventajas actualmente, tales como el ya mencionado desarrollo continuo de la aplicación, también permite la independencia tecnológica, ya que no requiere de ningún sistema operativo particular y tampoco actualizaciones del sistema, ya no solo ofrece un servicio de adquisición mucho más económico, si no también de mantenimiento y uso prolongado, ya que no requiere cuotas mensuales y además el código se adapta a los avances actuales por sí mismo si posee los suficientes usuarios interesados en él.

Debido al crecimiento de popularidad del lenguaje de programación Python, se ha propuesto este software libre para desarrollar el código ya que su aprendizaje puede ser de gran utilidad en el mundo laboral.

4. Introducción

4.1. Sistema

Un sistema es un conjunto de elementos que interaccionan entre sí. En la ingeniería de control, se pretende regular esas interacciones según las necesidades del usuario. Una entrada es una variable externa que altera los elementos del sistema, alterando su estado; la variable que se requiere controlar, se llama salida. Un sistema continuo es aquel cuyos valores de entrada y salida pueden tomar cualquier valor dentro de un rango, es decir, no solo ceros y unos. En este proyecto solo se estudiarán sistemas continuos.

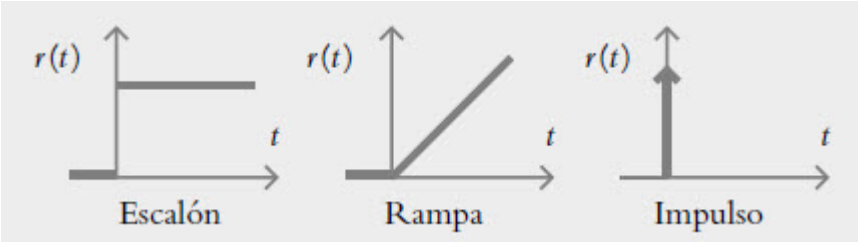
4.2 Tipos de entradas

En la realidad física, las entradas son señales de control y las salidas variables medibles. A pesar de que las entradas pueden ser totalmente arbitrarias hay tres que son las más habituales a la hora de estudiar su respuesta en la ingeniería de control.

-**Entrada impulso:** Esta entrada hace máxima la señal de control únicamente un instante de tiempo.
-**Entrada escalón:** Esta entrada hace máxima la señal de control a partir de un instante en el tiempo.
-**Entrada rampa:** Esta entrada crece de forma lineal de manera indefinida.

En la imagen 1, se muestran las entradas habituales en un sistema de control, correspondiendo la variable r(t) con la entrada.

Imagen 1: Entradas habituales en un sistema de control



Fuente: Documento “Empezando a entender el control automático” de la universidad Santo Tomás

Para manipular la salida a un valor deseado, es necesario colocar la entrada correcta, para ello es imprescindible conocer el comportamiento del sistema; la herramienta "F.d.t-ante-diferentes-entradas" permite observar las diferentes salidas de un sistema en función de los valores de las entradas. Para que la herramienta funcione correctamente primero hay que definir dicho sistema, así pues:

4.3 Posibilidades de obtención de una f.d.t.

Un sistema se puede representar con una ecuación matemática; esta ecuación puede obtenerse de dos formas:

- **Desarrollando las ecuaciones que componen el sistema:**

Si se tienen todos los datos necesarios y el sistema no es demasiado complejo, es posible desarrollar, mediante la física, las diversas ecuaciones que lo componen hasta obtener una en la que la señal de control u(t) se relacione directamente con la salida y(t); esta ecuación ha de ser lineal o aproximadamente lineal; si no lo es, se puede recurrir a su linealización utilizando el desarrollo de Taylor.

Una vez linealizada, se utiliza el operador matemático de la transformada de Laplace (s) para reducir el número de incógnitas temporales, tanto para la entrada como para la salida. Ahora, la salida y la entrada únicamente dependen del operador matemático s, por lo que pasan a llamarse
U(s) e Y(s) respectivamente; la entrada U(s) queda a un lado del igual multiplicando un número con operadores matemáticos s; todo este número se denomina G(s), o también, función de transferencia (f.d.t). Quedando finalmente la ecuación que se ve a continuación:

$$Y(s) = G(s) \cdot \Delta U(s)$$

La f.d.t es la relación directa entre la entrada y la salida del sistema; con lo que ya sería posible modelizar un controlador para variar la U(s) y obtener la Y(s) deseada.

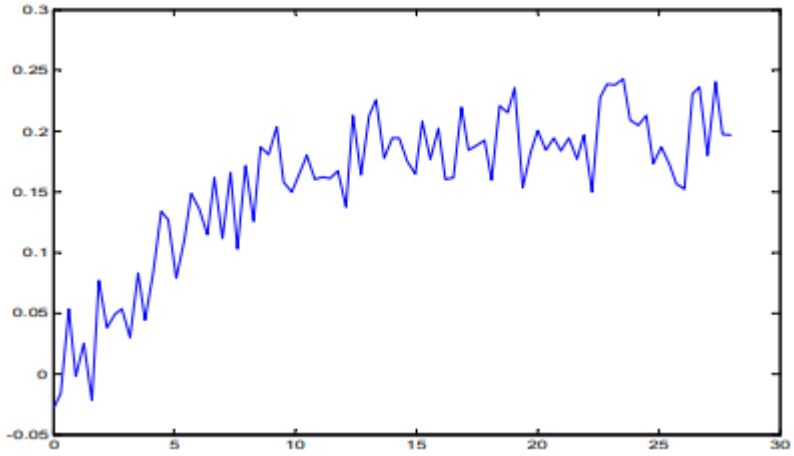
Una de las aplicaciones desarrolladas se encarga de calcular la inversa de Laplace de Y(s), es decir, calcular y(t), además de calcular la derivada e integral de esa y(t). Esta aplicación es la llamada "Representa_la_inversa_de_laplace". Esto puede ser útil para comprobar la certeza de

las ecuaciones desarrolladas, comparando el comportamiento del modelo real con el comportamiento del modelo de la inversa de Laplace.

- **De manera experimental:**

Para gran parte de los procesos industriales este es el método más utilizado ya que usualmente la complejidad de los sistemas dificulta mucho su análisis por ecuaciones basadas en principios físicos. Con este método se puede obtener de manera directa la función de transferencia. Para ello hay que realizar un ensayo y comprobar cómo se comporta el sistema ante él. Hay que tener precaución con el ruido ya que para valores bajos de la señal de control este afecta demasiado a los datos:

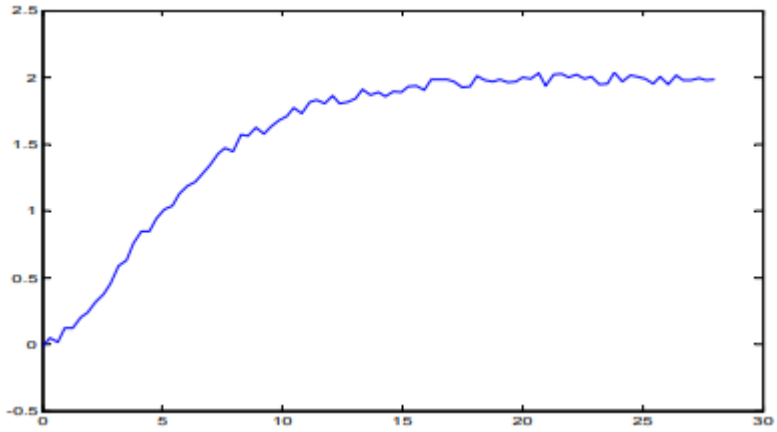
Imagen 2: Oscilación producida por el ruido (Valor de escala 0.3)



Fuente: Apuntes asignatura EE1023

Como se puede apreciar en la imagen 2 donde el valor de la escala de medida corresponde a 0.3, la oscilación producida por el ruido es demasiado grande y no permite una imagen definida del comportamiento del sistema. Esto se puede solucionar aumentando el valor de referencia donde el ruido tiene un peso relativo menor. En la imagen 3 se muestra como quedaría con un valor de escala de 2.5, en la que se aprecian variaciones por el ruido, pero quedando el comportamiento del sistema claramente definido.

Imagen 3: Oscilación producida por el ruido (Valor de escala 2.5)



Fuente: Apuntes asignatura EE1023

Para obtener dichos datos, normalmente se somete al sistema a una entrada ante referencia escalón y se espera a que se estabilice, después de esto los datos temporales, los de la salida y el valor de la entrada, se almacenan. Posteriormente, los datos se cargan en una aplicación gráfica que permite ajustar la curva de los datos obtenidos, con una curva basada en los valores de una f.d.t. típica ante el mismo tipo de entrada (en este caso, ante referencia escalón). Una de las aplicaciones desarrolladas se encarga de exactamente este principio, la llamada "Obtencion-de-una-f.d.t".

4.4 Ecuación típica de una f.d.t

Para agilizar el proceso de obtención de la f.d.t mediante esta técnica es un requisito conocer la estructura general de una f.d.t:

$$G(s) = \frac{k \cdot (1 + \beta \cdot s) \cdot e^{-Ts}}{s \cdot (1 + \tau_1 \cdot s) \cdot (1 + \tau_2 \cdot s) \cdot (1 + 2 \cdot \xi \cdot \omega_n^{-1} \cdot s + \omega_n^{-2} \cdot s^2)}$$

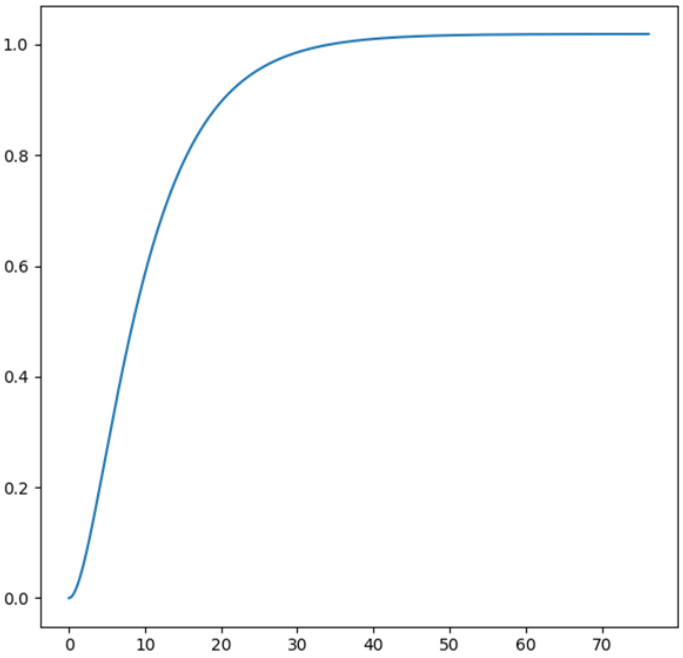
A continuación, se explicarán cómo afecta cada una de las variables de la ecuación:
En primer lugar, en el numerador se compone por la siguiente expresión:

$$k \cdot (1 + \beta \cdot s) \cdot e^{-Ts}$$

Donde:

***k*:** Se la conoce como la ganancia estática, y su valor siempre será el mismo que el valor en el que el sistema se estabiliza. En la imagen 4 se muestra un ejemplo, donde la ganancia tendría un valor de 1.

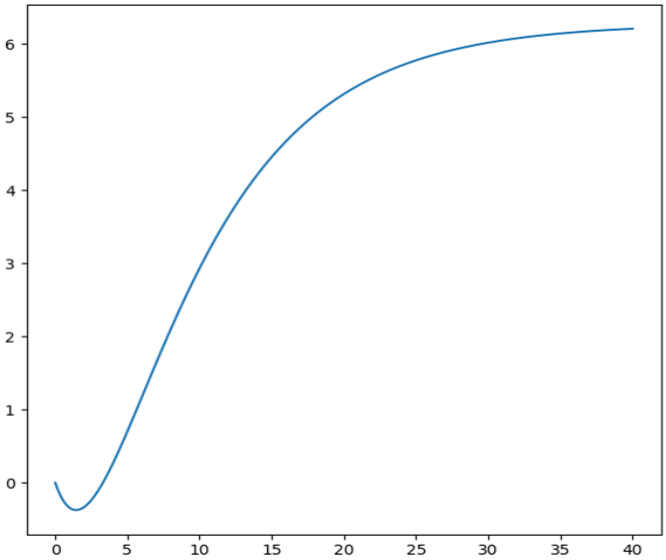
Imagen 4: Ejemplo representación ganancia estática (*k*)



Fuente: Elaboración propia

(*1+β·s*): Este término pertenece a los ceros del sistema; su comportamiento más característico es cuando β es negativa ya que produce un valle en el inicio temporal de la gráfica. En la imagen 5 se muestra un ejemplo donde β tendría valor negativo, formando de esta forma el valle característico.

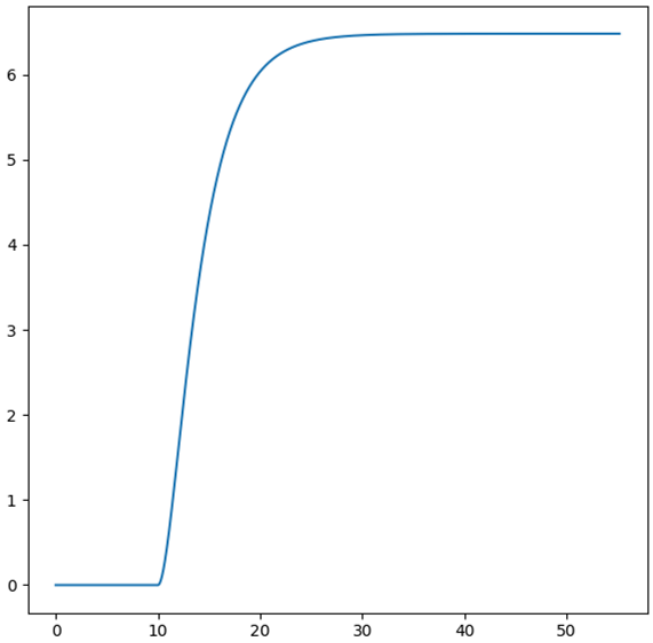
Imagen 5: Ejemplo representación ceros negativos en el sistema (*1+β·s*)



Fuente: Elaboración propia

($e^{-T \cdot s}$): Este término representa el retardo del sistema, es decir el tiempo que tarda el sistema en reaccionar ante la entrada debida a la acción de control. En la imagen 6 se muestra una representación de este suceso, con T=10s.

Imagen 6: Ejemplo representación retardo del sistema ($e^{-T \cdot s}$)



Fuente: Elaboración propia

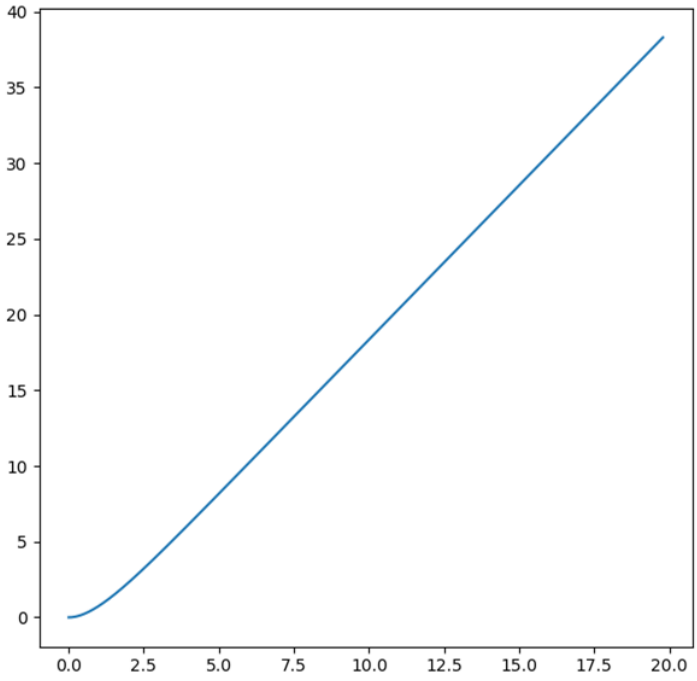
El denominador se compone de la siguiente expresión:

$$s * (1 + \tau 1 * s) * (1 + \tau 2 * s) * (1 + \tau 3 * s) * (1 + 2 * \xi * \omega n^{-1} + \omega n^{-2} * s^2)$$

Donde:

La ***s*** que multiplica todo el denominador se conoce como integrador, se caracteriza por hacer comportarse al sistema como una rampa ante referencia escalón. En la imagen 7 se muestra un ejemplo donde actúa dicho operador.

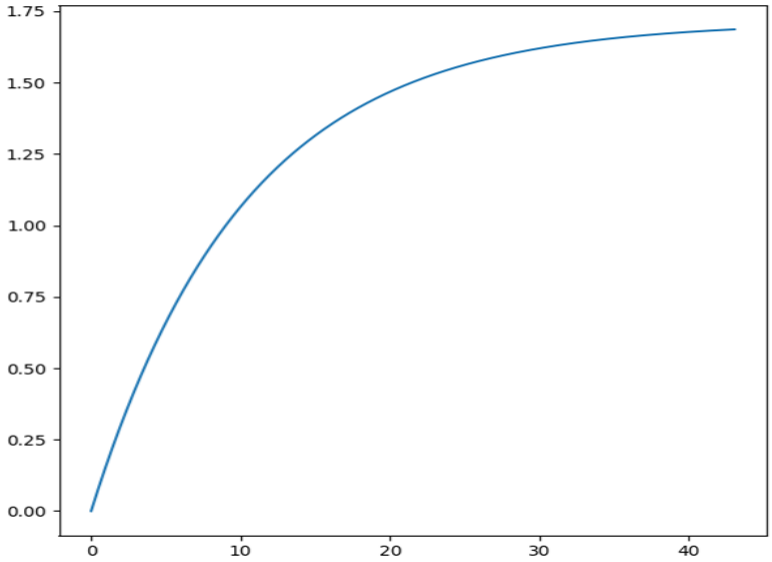
Imagen 7: Ejemplo representación integrador (*s*)



Fuente: Elaboración propia

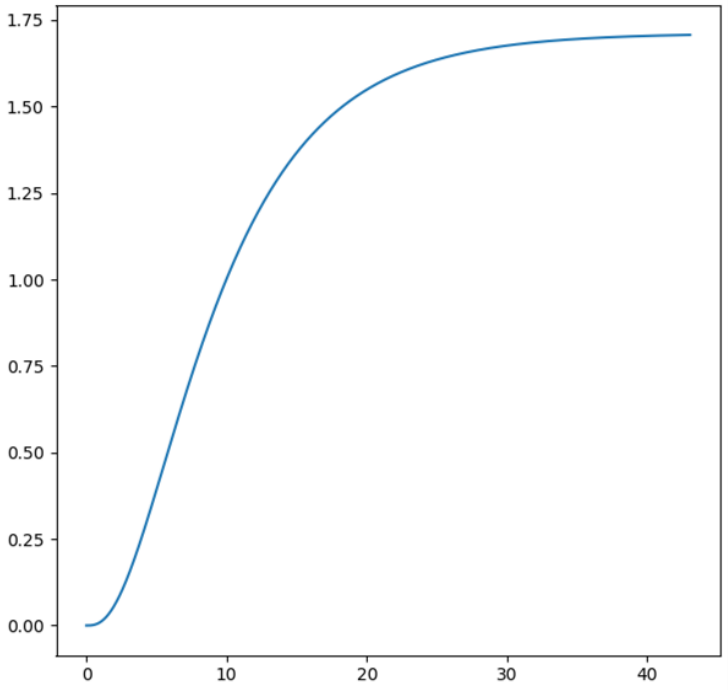
$(1+\tau_1 \cdot s)\cdot(1+\tau_2 \cdot s)\cdot(1+\tau_3 \cdot s)$: Estos son los polos del sistema, describen la rapidez con la que el sistema alcanza su valor máximo. Cuando menor sean sus valores más rápido alcanzará la referencia *k* el sistema; τ representa el tiempo de valor de establecimiento al 63%, si hay varios polos, el mayor es el que predomina ante los demás. La combinación de varios polos relevantes, produce arqueamientos en la curva. La imagen 8 representa una f.d.t con un único polo, habiendo dado valores en *k* de 1.7 y τ_1 de 10.2. Para ver el arqueamiento se ha visto conveniente incluir la imagen 9 con 3 polos, con valores de *k* de 1.7, τ_1 de 6.5, τ_2 de 2.5, τ_3 de 1.2.

Imagen 8: Ejemplo representación polos del sistema [$(1+\tau_1 \cdot s)\cdot(1+\tau_2 \cdot s)\cdot(1+\tau_3 \cdot s)$**]**



Fuente: Elaboración propia

Imagen 9: Ejemplo representación polos del sistema [$(1+\tau_1 \cdot s)$**]**



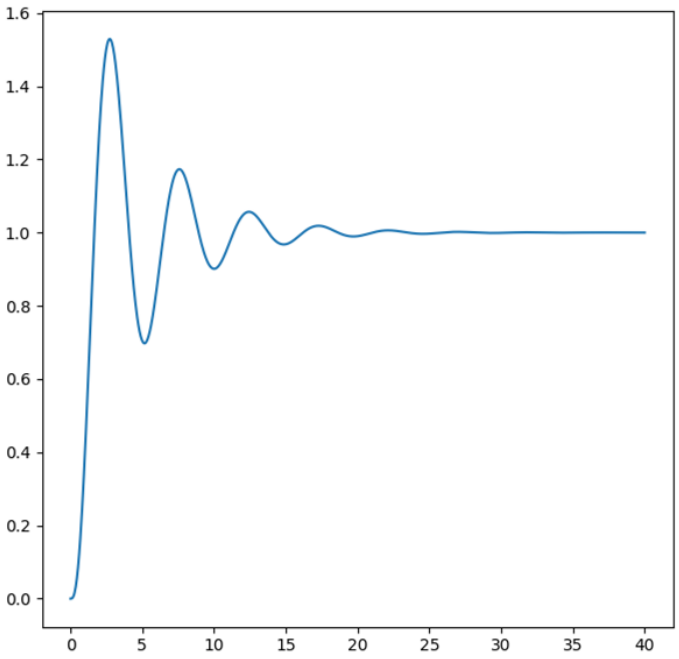
Fuente: Elaboración propia

$(1+2\cdot\xi\cdot s\cdot\omega n -1+\omega n -2 \cdot s2)$: Este fragmento representa los polos complejos del sistema, quienes son los responsables de las respuestas oscilatorias. Este efecto oscilatorio se mide con la sobreoscilación (δ) que se entiende como el valor máximo por encima del valor final de establecimiento. Cuanto mayores sean los polos reales en relación con los complejos, menos

relevantes serán los complejos, haciendo el sistema menos oscilatorio.

Incrementar el valor de la inversa de la frecuencia natural (ωn^{-1}) aumenta la duración de la sobreoscilación y por tanto el tiempo de establecimiento del sistema.
La relevancia del amortiguamiento (ξ) depende directamente del valor de ωn^{-1}, cuanto mayor sea ξmenor será la sobreoscilación, y cuanto menor sea ξmayor será la sobreoscilación.
En la imagen 10 se aprecia el comportamiento de los polos complejos y su facultad para provocar sobreoscilaciones. En el ejemplo se han usado los valores de k=1, τ_1=0.3, ωn^{-1}=0.76 y ξ=0.175.

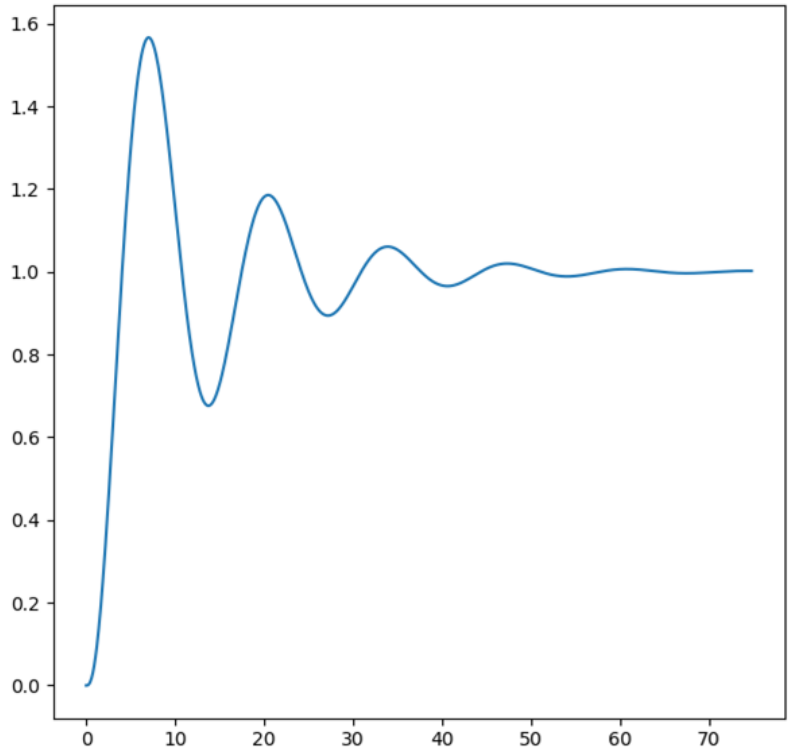
Imagen 10: Ejemplo representación polos complejos ($1+2\cdot\xi\cdot s\cdot\omega n^{-1}+\omega n^{-2}\cdot s^2$)



Fuente: Elaboración propia

En los siguientes ejemplos se variarán los valores de ωn^{-1}y ξpara entender su comportamiento. En el ejemplo de la imagen 11, habiendo usado los valores k=1, τ_1=0.3, ωn^{-1}=2.1 y ξ=0.175 se aprecia como el tiempo de establecimiento ha crecido de manera considerable debido al aumento de ωn^{-1}. En la imagen 12, se va a disminuir el valor de ξ para apreciar su comportamiento.

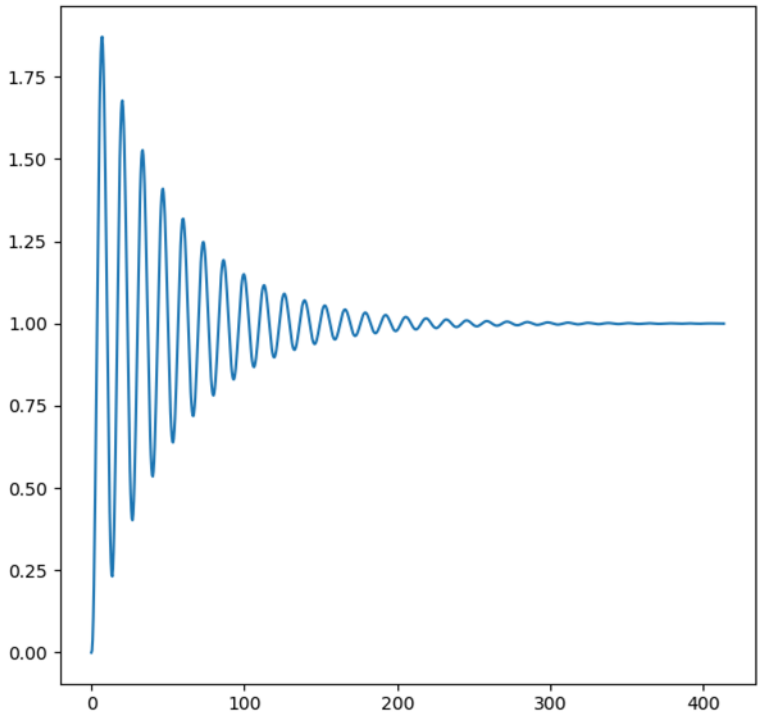
Imagen 11: Segundo ejemplo representación polos complejos ($1+2\cdot\xi\cdot s\cdot\omega n^{-1}+\omega n^{-2}\cdot s^2$)



Fuente: Elaboración propia

En ejemplo de la imagen 12, habiendo usado los valores k=1, τ_1=0.3, ωn^{-1}=2,105 y ξ=0,040 se aprecia que el tiempo de establecimiento y la cantidad de sobreoscilaciones aumentan considerablemente, reducir la sobre amortiguación a valores cercanos a 0 hace a un sistema con polos complejos muy oscilatorio.

Imagen 12: Tercer ejemplo representación polos complejos ($1+2\cdot\xi\cdot s\cdot\omega n^{-1}+\omega n^{-2}\cdot s^{2}$)



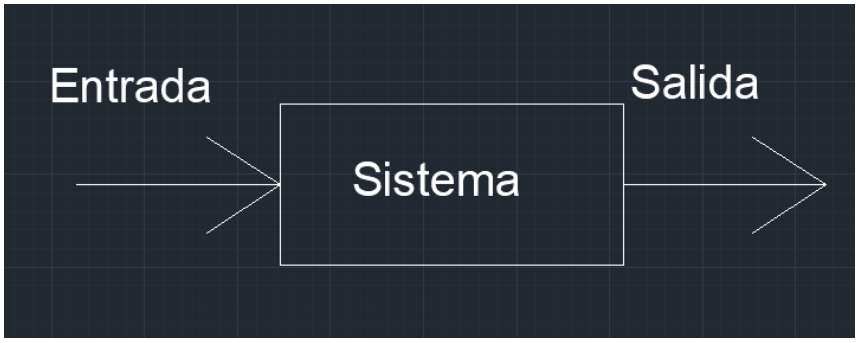
Fuente: Elaboración propia

4.5. Diagramas de bloques y sistemas de control

Antes de hablar de los controladores, es necesario conocer las estructuras de control. Los diagramas de bloques es la representación general utilizada en las estructuras de control, por lo que su conocimiento es indispensable.

Como se muestra en la imagen 13, en los diagramas de bloques los sistemas se representan con rectángulos, y las entradas y salidas con flechas.

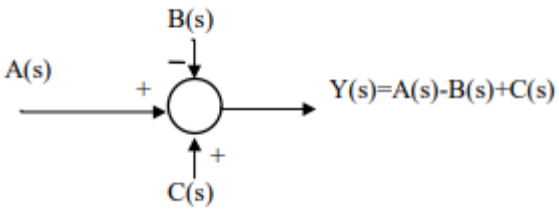
Imagen 13: Diagrama de bloques básico



Fuente: Elaboración propia

Las operaciones en los diagramas de bloques se pueden realizar con nodos sumadores, tales como los mostrados en la imagen 14:

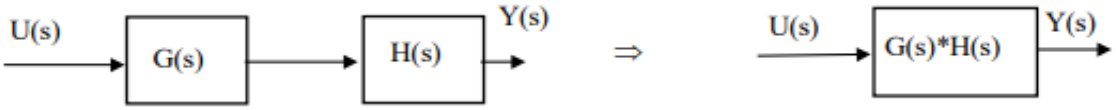
Imagen 14: Suma en los diagramas de bloques



Fuente: Apuntes asignatura EE1023

Otra forma de realizar operaciones es conectando dos bloques en cascada, realizando una multiplicación, tal como se muestra en la imagen 15:

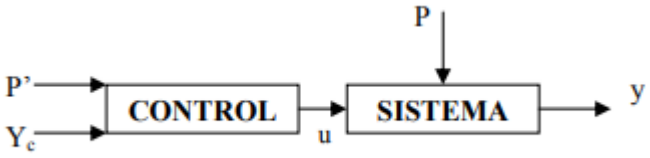
Imagen 15: Multiplicación en los diagramas de bloques



Fuente:Apuntes asignatura EE1023

Una estructura de control es un conjunto de operaciones matemáticas que se pueden representar en un diagrama de bloques. Existen varios tipos de estructuras de control:

Imagen 16: Lazo abierto

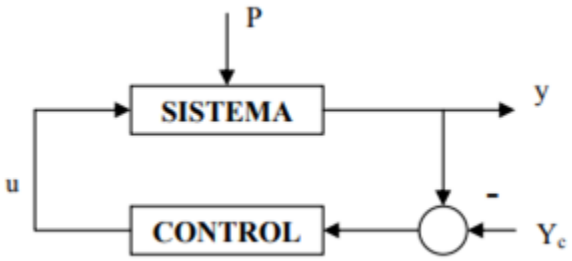


Fuente:Apuntes asignatura EE1023

En este ejemplo se observa que existen varias entradas en el sistema, una de ellas debida a la entrada del sistema *(u en el dibujo)* y otra a las perturbaciones *(P en el dibujo)* que se deben a procesos físicos, normalmente incontrolables. Su salida variará en función de la actuación de ambas entradas, el sistema de control tiene como propósito atenuar el efecto de las

perturbaciones indeseadas y obtener el valor de salida deseado por el usuario.
Esta estructura de control se considera de lazo abierto ya que la entrada del sistema de control solo está definida por ella misma *(Yc en el dibujo)* y posibles perturbaciones *(P' en el dibujo)* a la hora de colocar la entrada.

Imagen 17: Lazo cerrado

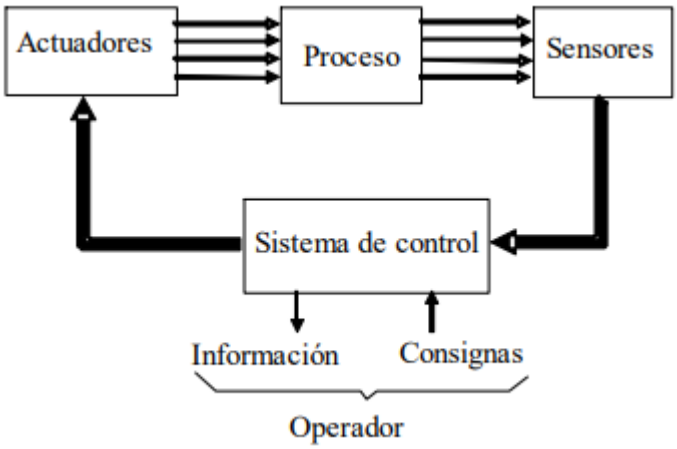


Fuente:Apuntes asignatura EE1023

En la imagen X la entrada del sistema de control depende de la salida del sistema, esto se conoce como realimentación; esto permite adaptar la salida del sistema de control según la necesidad de la salida del sistema; esto ocurre de manera automática debido a la construcción de la estructura.

La manera en la que se aplica este tipo de estructuras en la realidad física se puede comprender con la imagen X , correspondiente a un diagrama en bucle cerrado:

Imagen 18: Diagrama en bucle cerrado



Fuente:Apuntes asignatura EE1023

Para comunicar el sistema de control con el sistema *(denominado Proceso en el dibujo)* es necesario utilizar elementos físicos. Los sensores leen la salida actual del sistema, y los actuadores se encargan de transformar la salida del sistema de control en una entrada para el sistema. Las consignas introducidas por el usuario son las imposiciones de la salida del sistema, y la información son datos sobre las diferentes variables de la estructura de control; tanto las consignas como la información requieren de un usuario (Operador en el dibujo) para interpretar y exigir los diferentes valores.

En este proyecto se va a utilizar una estructura en lazo cerrado cuando se coloque el controlador, pero antes de representar la estructura concreta primero se explicará que tipo de controlador va a ser utilizado.

4.6 PID como mecanismo de control

Se va a utilizar el controlador más conocido a nivel mundial y más usado en las aplicaciones de control, se trata del controlador PID, el cual se explicará brevemente.
Se descompone en tres partes fundamentales, la parte proporcional (P), la parte integral (I) y la parte derivativa (D). Existen controladores formados solo por algunas de estas partes, como por ejemplo el controlador PI. Una forma sencilla de comprender su funcionamiento es observando directamente su ecuación. La ecuación se define de la siguiente manera:

$$u(t) = K_p\left(e_m(t) + \frac{1}{T_i}\int_0^t e_m(\tau)d\tau + T_d\frac{de_m(t)}{dt}\right)$$

Donde:

$u(t)$: Representa la salida del PID, es decir, la entrada de la f.d.t. a controlar.

$e_m(t)$= Es el equivalente entre la resta del valor deseado de Y(s) y el de la referencia, este valor varía con el tiempo.

Kp: Representa la parte proporcional del PID, cuanto mayor sea su valor más agresiva será la salida del PID.

Ti: Representa la parte integral del PID, gracias a esta parte se puede conseguir el valor final de la referencia sin errores permanentes ya que es capaz de acumular el error pasado y ajustarlo. Cuanto menor sea este término más relevante será este efecto.

Td: Representa la parte derivativa del PID y está enfocado a disminuir la velocidad de establecimiento, ya que la derivada del error habla sobre la tendencia del mismo. Aumentar este valor hace al PID más rápido pero también más oscilatorio, haciéndolo inviable para muchas aplicaciones. Es importante destacar que también puede amplificar mucho el ruido.

Aplicando la transformada de Laplace para $e_m(t)$ y una entrada u(t) se obtiene la ecuación que permite operar directamente con funciones de transferencia:

$$C(s) = \frac{U(s)}{E_m(s)} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

4.7 Medidas de interés de los controladores PID

Existen magnitudes medibles a tener en cuenta a la hora de optimizar un PID, algunas se contraponen a otras, por ejemplo un PID más rápido también será más oscilatorio. Estas magnitudes se pueden dividir entre referencia escalón y ante perturbación:

-Referencia escalón:

ts98: Es el tiempo que tarda la salida en alcanzar y permanecer entre el 98% y 1.02% del valor de la referencia

δ: La sobreoscilación máxima de la salida.

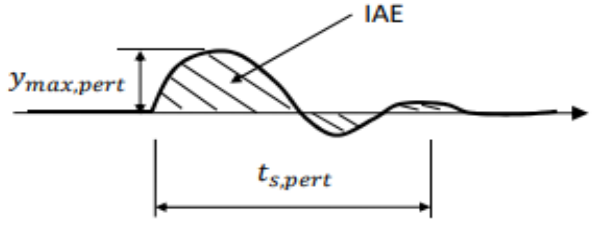
-Ante perturbaciones:

tspert: Es el tiempo de establecimiento en el que el sistema supera la perturbación y vuelve al valor original.

ymaxpert: Es el valor máximo que alcanza la perturbación.

IAE: Se define como la integral absoluta del error, es decir el área por debajo de la curva de la representación de la perturbación. En la imagen 19 se muestra una representación de esto.

Imagen 19: Representación IAE



Fuente:Apuntes asignatura EE1023

Su ecuación es: $IAE = \int_0^{\infty} |e(t)| dt$

4.8 Modificaciones respecto al PID original

Debido que el término derivativo suele ser problemático en relación al ruido de medida, existen maneras de suavizar su efecto, una solución típica es emplear un filtro que controle las variaciones bruscas del error medio, ahora, el término derivativo en C(s) queda como:

$$\frac{T_d s}{1 + \frac{T_d}{N} s}$$

De manera que amplificando el valor de N, se aumenta el efecto derivativo normal; un valor habitual de N es 10.

Otra modificación general en los PID se debe a que el cambio en la referencia puede ser demasiado drástico debido al término proporcional y a la derivada del mismo, ya que un cambio instantáneo provoca una derivada infinita , por tanto el cambio de referencia puede provocar sobreoscilaciones no deseadas, la solución habitual es ponderar esta referencia de la siguiente manera:

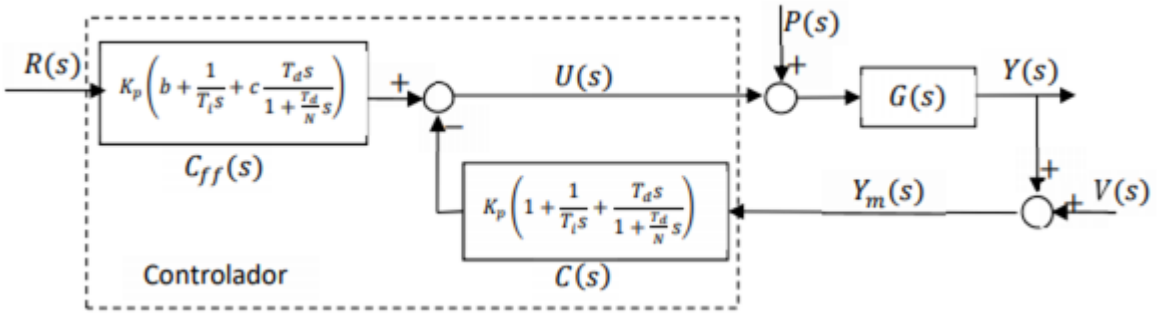
$$u(t) = K_p \left(b\, r(t) - y_m(t) + \frac{1}{T_i} \int_0^t e_m(\tau) d\tau + T_d \left(c \frac{dr(t)}{dt} - \frac{dy_m(t)}{dt} \right) \right)$$

Los términos b y c modifican el error en la referencia del PID, sus valores pueden ir desde 0 hasta 1. Así pues, juntando ambas modificaciones, y realizando la transformada de Laplace queda la siguiente ecuación:

$$U(s) = \underbrace{K_p \left(b + \frac{1}{T_i s} + c \frac{T_d s}{1 + \frac{T_d}{N} s} \right)}_{C_{ff}(s)} R(s) - \underbrace{K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{N} s} \right)}_{C(s)} Y_m(s)$$

La estructura de control típica es, por tanto, la que se muestra en la imagen 20:

Imagen 20:Estructura de control típica



Fuente:Apuntes asignatura EE1023

Donde:

R(s) es la referencia del usuario

U(s) la entrada a la G(s) que se quiere controlar

P(s) representa las perturbaciones

Y(s) la salida que se pretenda igualar a la referencia (R(s))

V(s) el ruido de medida del sensor que mide Y(s)

C(s) y **Cff(s)** representan las dos partes del controlador con ponderación

La herramienta informática desarrollada 'Respuesta-de-PID' funciona con la estructura de control explicada en este punto. Además, es capaz de mostrar los valores de las magnitudes medibles expresadas en el punto 4.7, es capaz de regular los valores Kp, Td, Ti, N, a y b según se requiera y de graficar tanto la referencia ante entrada escalón como ante perturbación.

4.9. Formato ISE y formato en paralelo

El formato de la ecuación del PID presentado es el más representativo del mismo, es el denominado ISA; sin embargo hay quien considera más cómodo otro formato de la ecuación del PID, llamado formato en paralelo. La única diferencia es la manera de representar ambas ecuaciones, la diferencia es si Kp multiplica la ecuación desde fuera del paréntesis o ya habiendo multiplicado las variables, reemplazando la variable 1/Ti por Ki, la variable Td por Kd y la variable Td/N por τd.

- Formato ISA:

$$U(s) = (Kp \cdot b + \frac{1}{Ti \cdot s} + \frac{c \cdot Td \cdot s}{1 + \frac{Td}{N} \cdot s}) \cdot R(s) - (Kp \cdot 1 + \frac{1}{Ti * s} + \frac{c \cdot Td \cdot s}{1 + \frac{Td}{N} \cdot s}) \cdot Ym(s)$$

- Formato paralelo:

$$U(s) = (Kp \cdot b + \frac{Ki}{s} + \frac{c \cdot Kd \cdot s}{1 + \tau d \cdot s}) \cdot R(s) - (Kp \cdot 1 + \frac{Ki}{s} + \frac{Kd \cdot s}{1 + \tau d \cdot s}) \cdot Ym(s)$$

Por este hecho, se decidió añadir una pestaña extra en la aplicación 'Respuesta-de-PID' para decidir en qué formato optimizar el PID, funcionando ambos formatos simultáneamente, permitiendo optimizar dos PID a la vez.

5. Normas y referencias aplicables

En este apartado se mencionan las referencias a obras de terceros que han sido empleadas para la realización del proyecto.

- Programas utilizados:

  - Spyder, versión 4.0.1,con compatibilidad para Python 3.7.6 64-bit | Qt 5.9.6 | PyQt5 5.9.2 | Windows 10

- Bibliografía:

- Apuntes de la asignatura EE1023- Sistemes Automàtics

- Apuntes de la asignatura EE1038 - Regulació Automàtica

- Información acerca de librerías y usos de funciones: tutorialspoint.com

- Información acerca de funciones de numpy: numpy.org

- Diversas consultas sobre asuntos de programación: stackoverflow.com

6. Definiciones y abreviaturas

Sistemas y diagrama de bloques:

C(s): Controlador en función del operador matemático s.

Cff(s): Parte del controlador con ponderación con referencia y en función del operador matemático s.

f.d.t: Función de transferencia.

G(s):Función de transferencia.

R(s): Referencia temporal en función del operador matemático s.

s:Operador matemático de la transformada de Laplace

P(s): Perturbación en función del operador matemático s.

u(t):Entrada temporal.

U(s):Entrada temporal en función del operador matemático s.

y(t):Salida temporal.

Y(s):Salida temporal en función del operador matemático s.

τ:Constante de tiempo al 63%

ξ: Sobreamortiguamiento.

β: Ceros del sistema.

e^{-Ts}: Retardo del sistema.

δ: Sobreoscilación.

ω_n^{-1}: Inversa de la frecuencia natural.

Control de sistemas:

b: Factor de amortiguamiento de la referencia .

c: Factor de amortiguamiento de la referencia en el término derivativo.

I.A.E: Integral absoluta del error.

Kp: Término proporcional del PID.

Kd: Resultado de la operación: Kp* Td

Ki: Resultado de la operación: $\frac{Kp}{Ti}$.

N: Término de filtrado de la parte derivativa de un PID.

Ti: Término integral del PID.

Td: Término derivativa del PID.

τn: Resultado de la operación: $\frac{Td}{N}$

Viabilidad económica:

i_n: Interés nominal

i_r: Interés real

IVA: Impuesto sobre el valor agregado.

PEM: Presupuesto de ejecución material.

PEC: Presupuesto de ejecución por contrata.

PR: Periodo de retorno.

TIR: Tasa interna de retorno.

VAN: Valor actual neto.

7. Requisitos de diseño

En este punto se citarán las condiciones a cumplir para desarrollar correctamente el proyecto.

-Interpretación de los datos escritos del usuario: El usuario debe escribir datos de manera natural y el programa debe ser capaz de leerlos.

-Obtención de datos experimentales a través de un fichero: Los datos de los ficheros también deben ser leídos e interpretados correctamente.

-Botones y deslizaderas: Deben ser intuitivos, estar correctamente nombrados y su valor debe ser leído correctamente.

-Representación gráfica fluida: Las gráficas deben adaptarse a la variación de los inputs del usuario.

-Tratado de datos de entrada para su posterior operación: Algunos datos de entrada no son homogéneos y requieren de código previo para poder ser utilizados.

-Cálculo de las variables obtenidas: Operar con los inputs leídos.

-Veracidad de los resultados: Los resultados deben ser correctos y comparables a otros realizados por aplicaciones similares.

-Homogeneidad entre el código de las distintas aplicaciones: Las distintas aplicaciones deben funcionar de maneras similares para que su comprensión se facilite, ya que uno de los objetivos de este proyecto es el desarrollo compartido con otras personas.

-Coherencia en el nombramiento de variables: Por el mismo motivo, las variables deben tener nombres fácilmente reconocibles y lo más explicativos posibles.

8. Propuesta de alternativas

En este punto se dispondrán las consideraciones previas a la realización del proyecto.
El proyecto se planteó en código abierto desde el principio debido a su finalidad distributiva y didáctica, sin embargo dentro de este tipo de código se plantearon dos lenguajes distintos a la hora de realizar las aplicaciones:

-**Java:** La ventaja principal frente a Python es que es más rápido y suele ser más eficiente solucionando problemas de rendimiento.

-**Python:** Es un lenguaje más sencillo, con una curva de aprendizaje más adecuada para la iniciación y permite el uso de variables antes de que se declaren, lo que facilita la escritura de código, y además permite la colaboración con otros desarrolladores online de manera muy sencilla.

Dadas las ventajas que ofrece cada uno, Python es la mejor opción ya que la utilidad de compartir sencillamente el código de manera online es uno de los objetivos del trabajo, además el código no será tan pesado como para tener problemas de rendimiento.

Una vez elegido Python, había dos propuestas para representar gráficamente las aplicaciones:

-**PyQT:** Posee su propia interfaz y eso hace su uso de primeras más sencillo, al ser una de las opciones gráficas más populares para Python, hay multitud de información sobre él en internet.

-**Tkinter**: Esta herramienta se estudió en una asignatura previa, no requiere de terceras aplicaciones y además su uso comercial no está limitado.

Después de deliberar, dado que el hecho de necesitar terceras aplicaciones hace más complejo su distribución, y el hecho de que su uso para empresas requiera un importe de pago, se eligió tkinter, herramienta en la que además se poseía cierta experiencia.

Por último, existía la elección entre la librería pandas y la librería numpy, dado que se estima que numpy es más veloz para tratar con menos de cincuenta mil filas, y dado que las matrices tratadas en estas aplicaciones tienen dimensión vectorial, se elige numpy como librería para el cálculo.

9. Resultados finales

En este punto se mostrarán el resultado de las cuatro aplicaciones desarrolladas y se explicará su propósito y utilización.

9.1 '**Representa_la_inversa_de_Laplace**'

El propósito de esta aplicación es verificar los cálculos de la f.d.t calculada a partir de un modelo físico, para ello se escribe la f.d.t en la aplicación y se selecciona la opción 'y(t)', si la gráfica resultante se comporta de la misma manera que el modelo físico, la f.d.t calculada es correcta, con lo que ya sería posible optimizar un PID utilizando la herramienta 'Respuesta-de-PID' para controlar la salida del sistema según se desee. A continuación se van a mostrar diferentes ejemplos de su uso y utilidad:

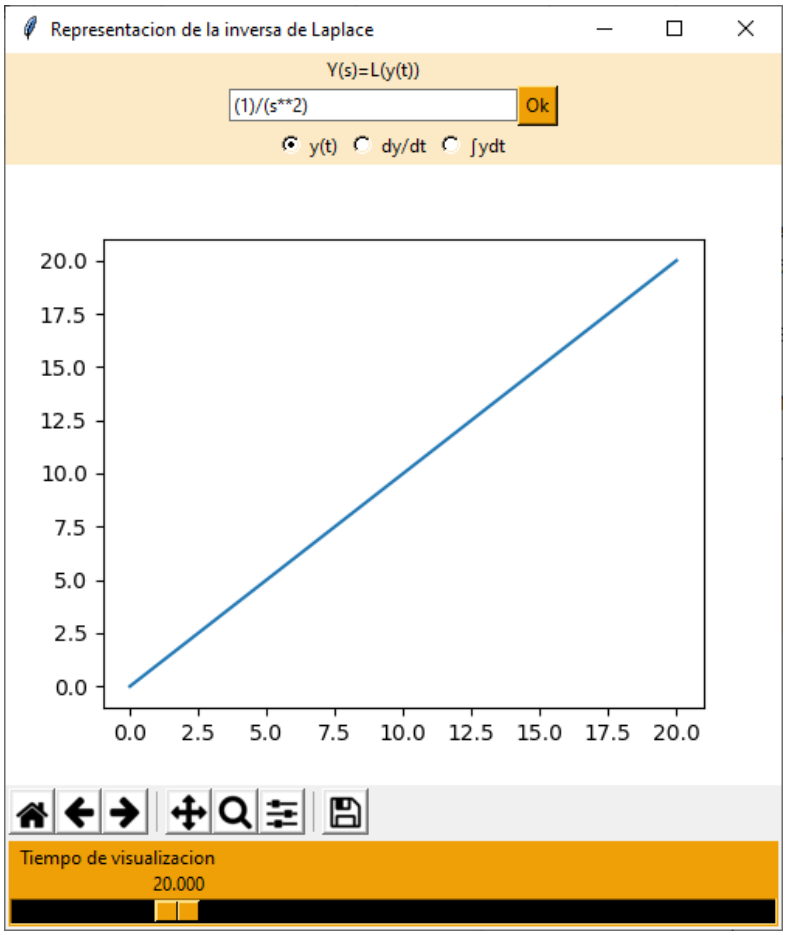
9.1.1 Ejemplos

- **Ejemplo 1:**

Para que la herramienta funcione, se debe elegir una de las tres opciones:

La inversa de Laplace de la ecuación $Y(s) = \frac{1}{s^2}$ pertenece a y(t)=t que corresponde a una recta de manera gráfica, como se puede observar en la imagen 21 la aplicación dibuja correctamente esta respuesta.

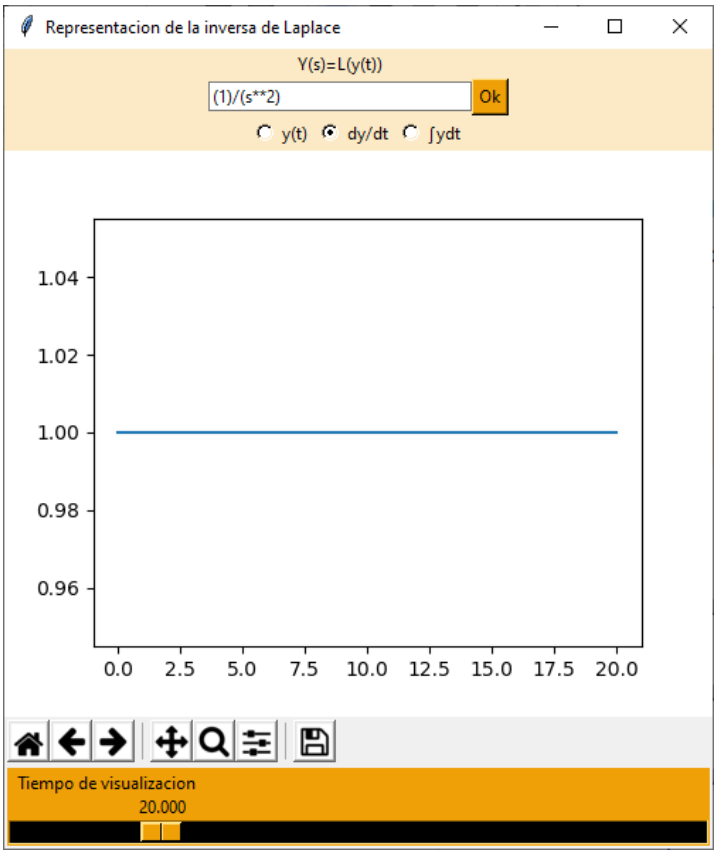
Imagen 21: Inversa de Laplace del ejemplo 1



Fuente:Elaboración propia

En la siguiente imagen se selecciona el otro botón, el resultado de la operación $y(t) = \frac{dy}{dt}$ sería equivalente a 1, como muestra la imagen 22.

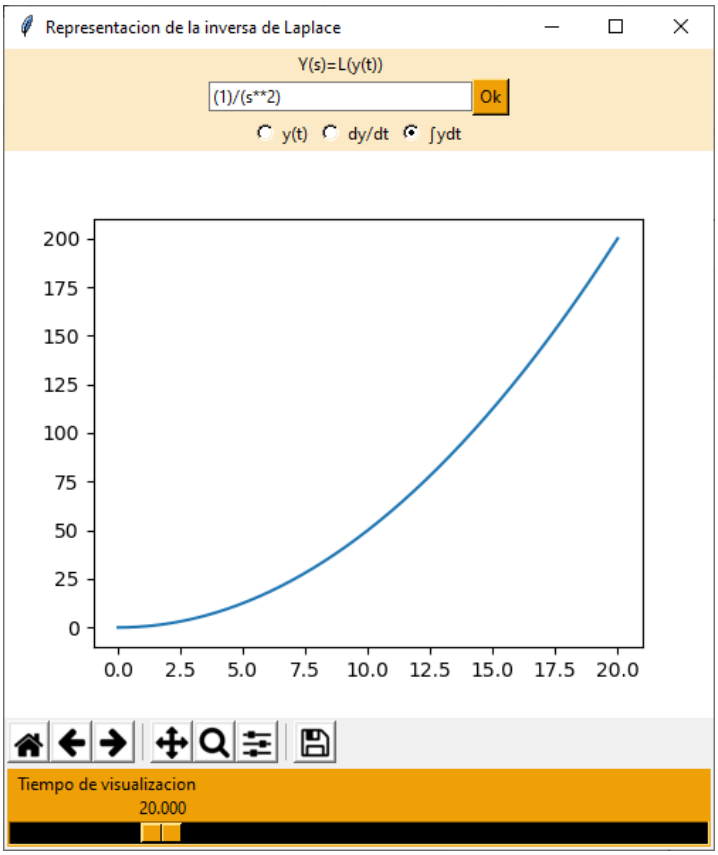
Imagen 22: Derivada de y(t) del ejemplo 1



Fuente:Elaboración propia

El resultado de la operación $y(t) = \int_0^\infty tdt$ sería equivalente a una función exponencial, como muestra la imagen 23.

Imagen 23: Integral de y(t) del ejemplo 1



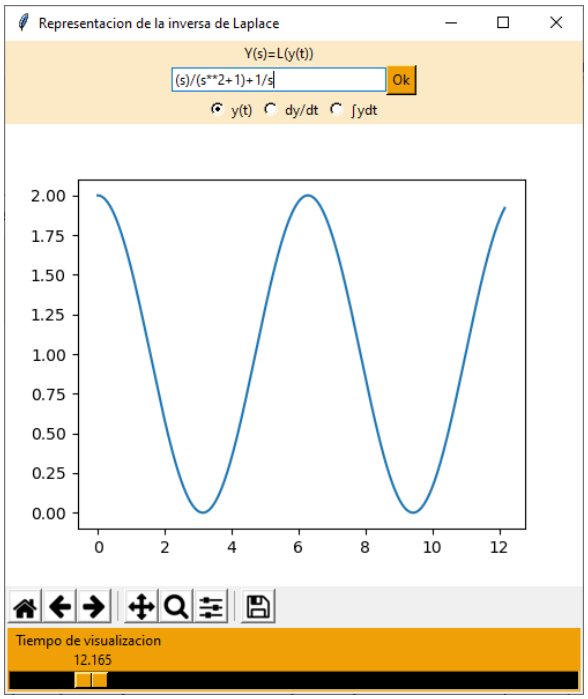
Fuente:Elaboración propia

- **Ejemplo 2**

La inversa de $Y(s) = \frac{s}{(s^2+1)} + \frac{1}{s}$ se conoce aplicando directamente la tabla de la inversa de Laplace, es decir: y(t)=cos(t)+1

La imagen 24 muestra correctamente que el punto de oscilación de un coseno en este caso es 1.

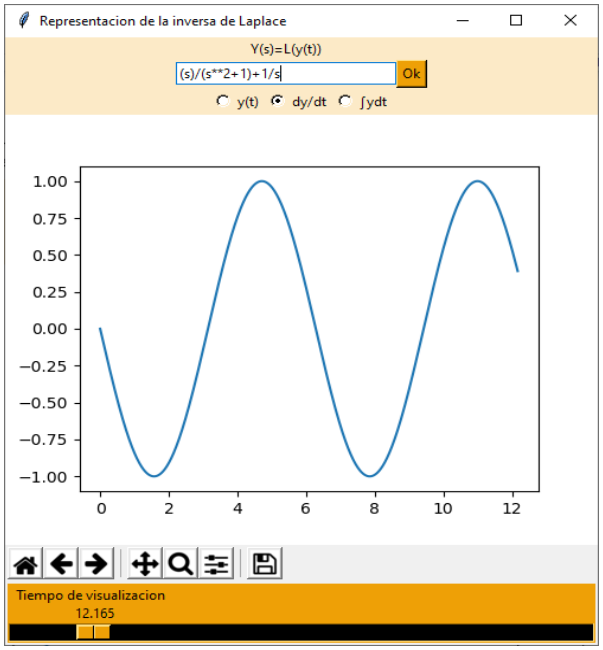
Imagen 24: Inversa de Laplace del ejemplo 2



Fuente:Elaboración propia

La derivada de la función anterior es el -sen(t), representado en la imagen 25.

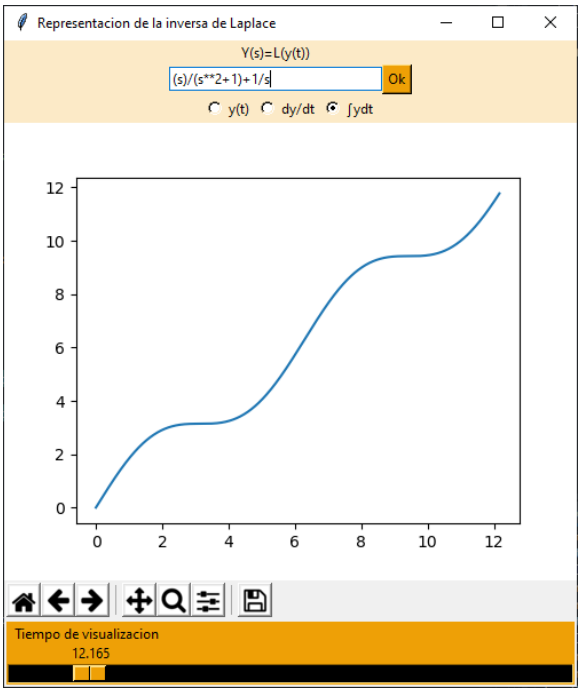
Imagen 25: Derivada de y(t) del ejemplo 2



Fuente:Elaboración propia

La integral de la función sería sen(t)+t ,representado en la imagen 26.

Imagen 26: Integral de y(t) del ejemplo 2

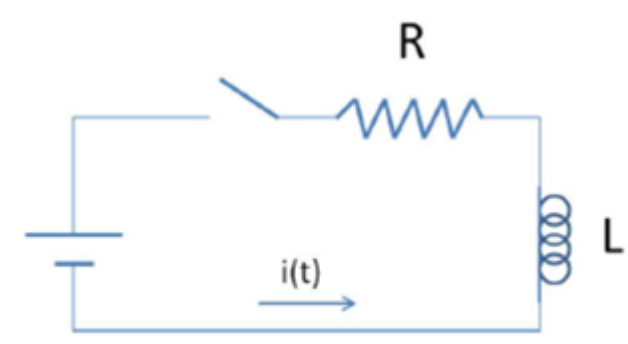


Fuente:Elaboración propia

- **Ejemplo 3**

Se pretende conocer la intensidad i(t) que circula por el circuito indicado en la imagen 27:

Imagen 27: Circuito ejemplo 3



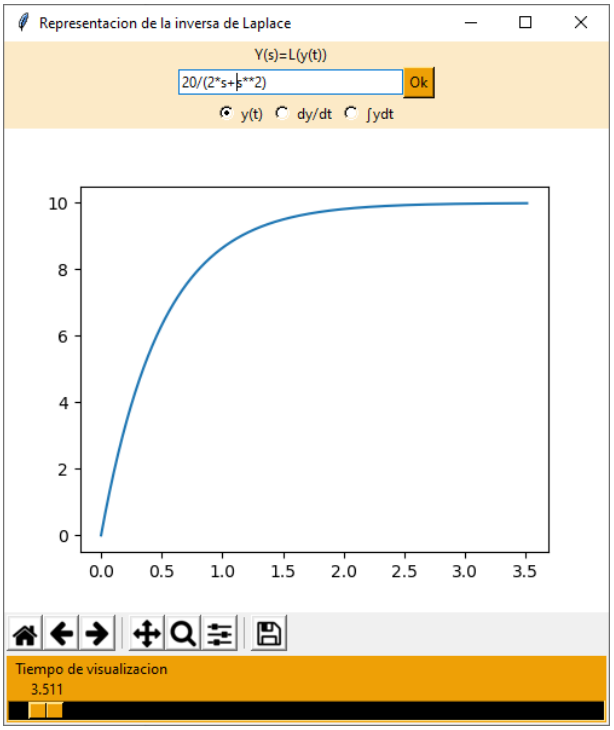
Fuente: Documento de Universidad politécnica de Madrid

Estando el desarrollo en el apartado 1.2 de los anexos, el resultado de la transformada de Laplace obtenido es:

$$I(s) = \frac{20}{2s + s^2}$$

Al principio la bobina actúa como una resistencia y consume voltaje, reduciendo la intensidad del sistema, sin embargo cerca de los 2s la bobina ya se comporta como un circuito cerrado, para la resistencia de 2Ω con un voltaje de 20V, obtiene una intensidad de 10A. Estos resultados eran los esperados, por lo que la transformada de Laplace se ha calculado correctamente como se muestra en la imagen 28 donde se representa i(t).

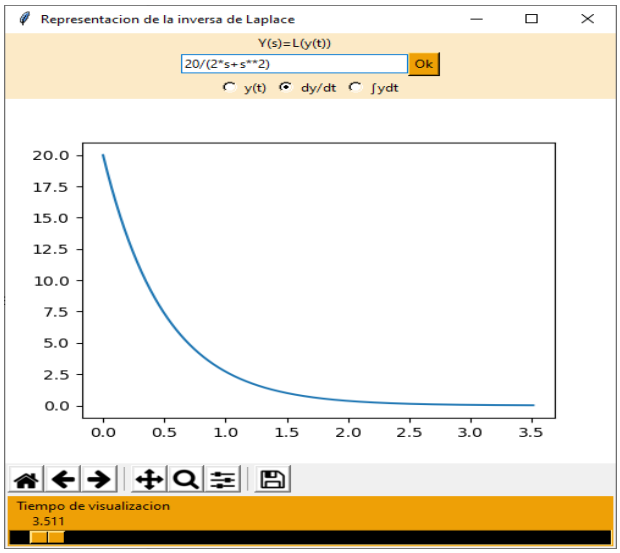
Imagen 28: Inversa de Laplace del ejemplo 3



Fuente:Elaboración propia

Además, también se tienen algunos datos adicionales; con la derivada de la i(t) vemos que la mayor variación de intensidad ha sido al cerrar el interruptor; después de eso su variación ha ido disminuyendo hasta estabilizarse en el valor final como se muestra en la imagen 29, que se conoce de la comprobación anterior, y son 10A.

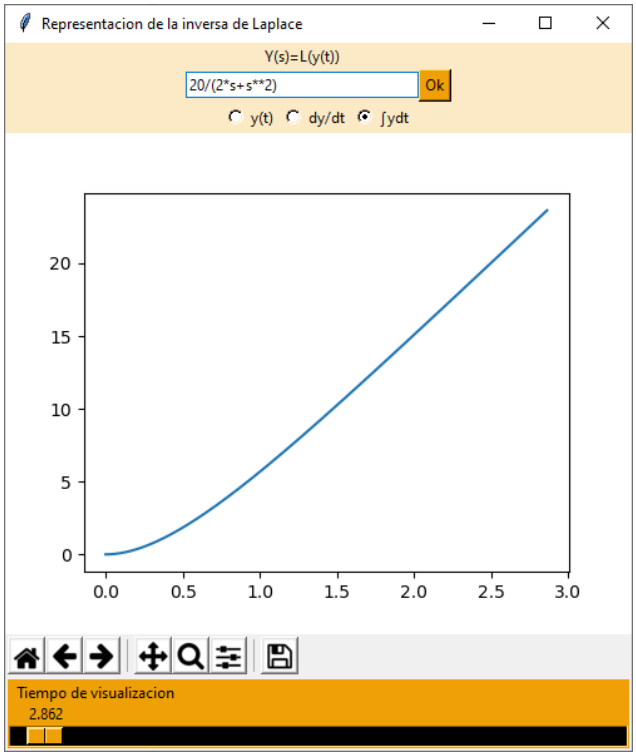
Imagen 29: Derivada de y(t) del ejemplo 3



Fuente:Elaboración propia

Y con la integral también se aprecia una ligera diferencia en la pendiente de la curva en los primeros momentos, haciendo notar la variación inicial en la intensidad. Mostrandose en la imagen 30.

Imagen 30: Integral de y(t) del ejemplo 3



Fuente:Elaboración propia

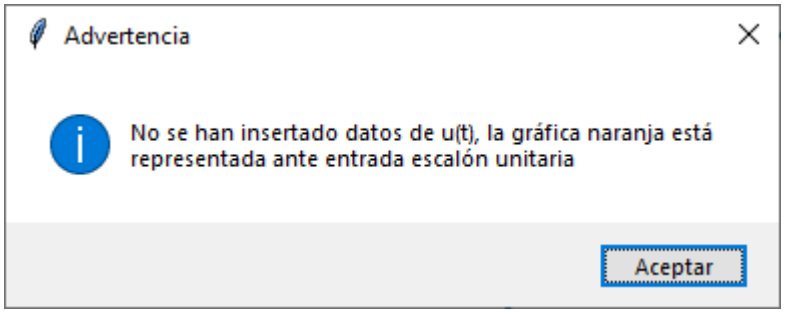
9.1.2 Conclusión

Es una aplicación funcional y de uso sencillo que no requiere conocimientos acerca de la ingeniería de control, con lo que puede ser utilizada en más ámbitos, su utilidad didáctica también es algo remarcable ya que permite observar varios aspectos de un sistema físico de una manera directa. Esta aplicación funciona rápido y no suele quedarse bloqueada.

9.2 Obtencion-de-una-f.d.t

Esta aplicación pretende calcular una f.d.t a partir de la inserción de unos datos experimentales. Una vez insertados presionando en el botón habilitado para ello, la f.d.t se calcula igualando la gráfica de los datos a otra que se modifica manualmente. Esta gráfica variable es una f.d.t que se calcula a partir de los valores de las deslizaderas y de los datos u(t) de entrada del archivo. Adicionalmente, y como alternativa, si se insertan datos experimentales sin una u(t), el programa interpreta los datos insertados como una entrada escalón unitaria y crea la gráfica a ajustar tambíen ante referencia escalón unitaria en consecuencia. Esto puede ser útil cuando por algún motivo no se pueda medir la u(t) pero si se tenga certeza de que se ha aplicado una entrada escalón unitaria. El programa muestra una ventana de advertencia en este caso, como la mostrada en la imagen 31:

Imagen 31: Advertencia de no inserción de u(t)



Fuente:Elaboración propia

De manera predeterminada, se muestran las dos gráficas, una azul que representaría los datos experimentales pulsando el botón “Cargar datos t,y(t) y u(t) de un fichero” y otra naranja que varía según elija el usuario. Si no se pulsa este botón, la gráfica azul será una f.d.t previamente programada, la cual puede modificarse según convenga. De la manera predeterminada, ambas gráficas son representadas mediante una f.d.t ante referencia escalón unitaria.

Es importante mencionar previamente que el error mostrado por pantalla en la aplicación hace referencia al error calculado como el valor medio de la diferencia entre cada uno de los puntos de las gráficas; donde *i* es el último punto de las gráficas:

$$Error = \frac{\sum_0^i (y(i)_{gráfica1} - y(i)_{gráfica2})}{i_{final}}$$

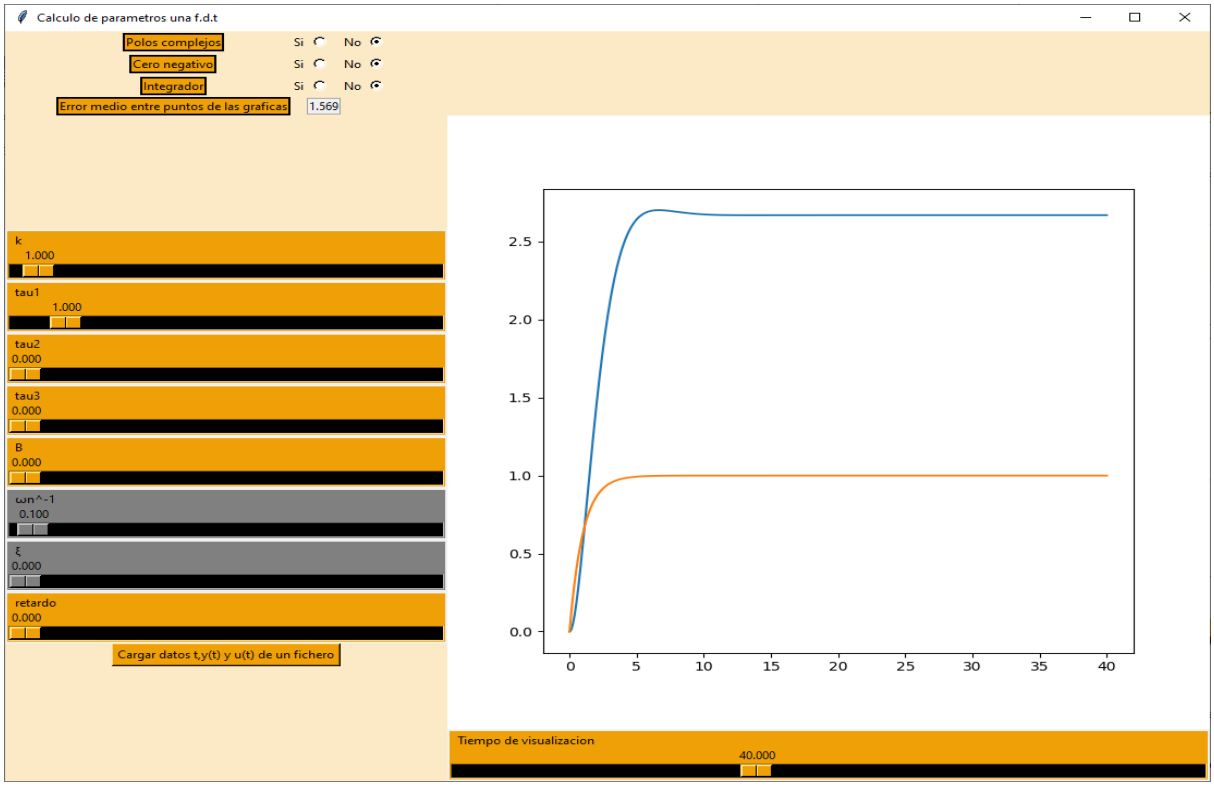
Este error debe entenderse como relativo a los valores de las gráficas, si se tratan valores del orden de 2000 en el eje y(t), un error de 10 es un error razonable, en cambio para el orden de 100 en el eje y(t) un error de 0.5 lo es.

9.2.1 Ejemplos

- **Ejemplo 1**

En la imagen 32 se observa cómo se han colocado unos datos experimentales de ejemplo (gráfica azul) presionando el botón "Cargar datos t,y(t) y u(t) de un fichero". La gráfica naranja representa la f.d.t actual según los valores de las deslizaderas.

Imagen 32: Representación parámetros de una f.d.t ejemplo 1

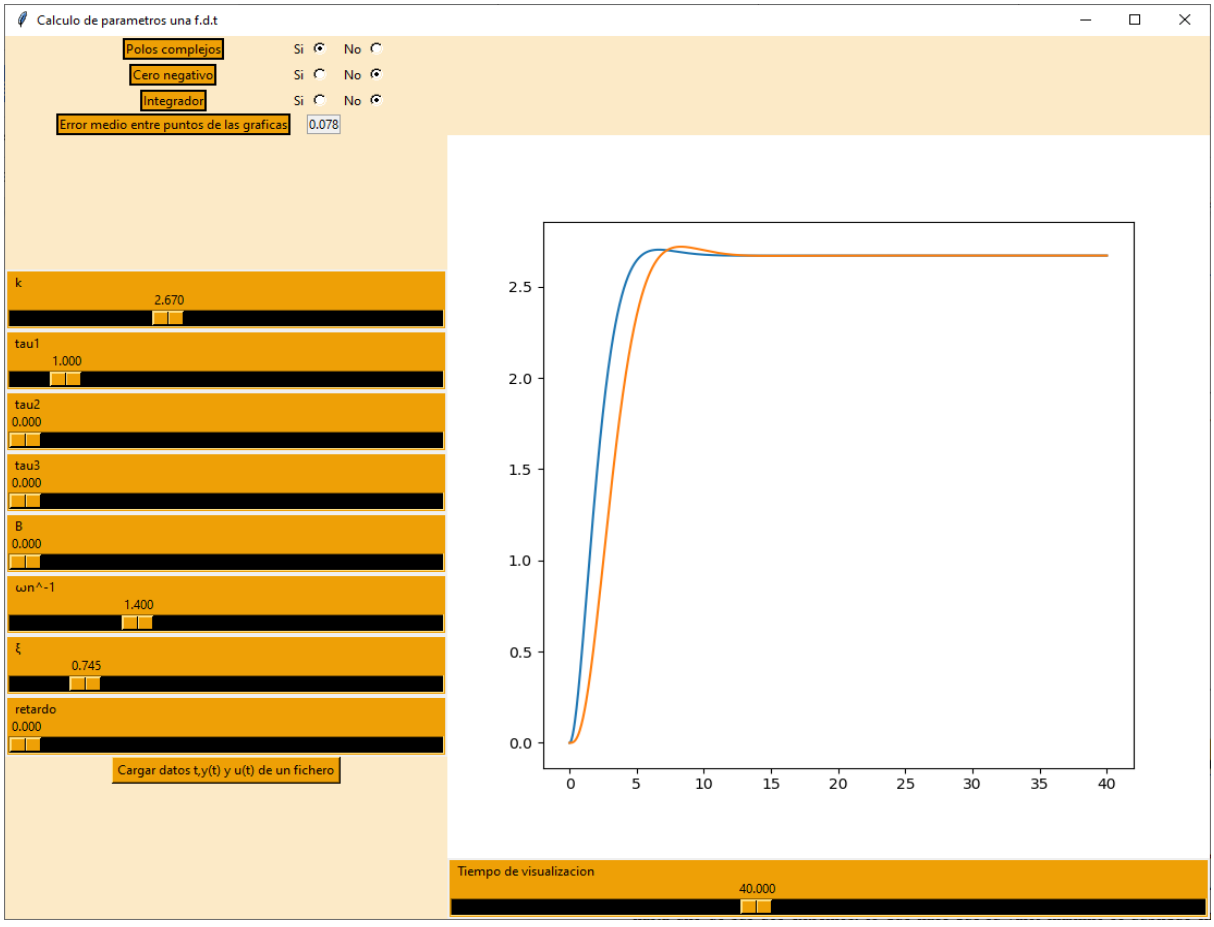


Fuente:Elaboración propia

Si se pretende igualar ambas gráficas, será necesario aplicar los conocimientos descritos en el punto 4.4 de la memoria. Se observa que la constante k del sistema a analizar está cerca de 2.6, además, se aprecia que la gráfica azul tiene una única sobreoscilación antes del punto de establecimiento; se cree que esto es debido a la existencia de números complejos en la f.d.t.

Para poder utilizar los polos complejos será necesario habilitarlos en la esquina superior izquierda, donde se puede apreciar que existen diversas opciones de configuración, además, aquí está la variable que muestra el error entre las gráficas a cada variación de las mismas.
Así pues, moviendo las deslizaderas y activando los polos, el resultado se muestra en la imagen 33.

Imagen 33: Representación parámetros de una f.d.t ejemplo 1 (ajuste 2)



Fuente:Elaboración propia

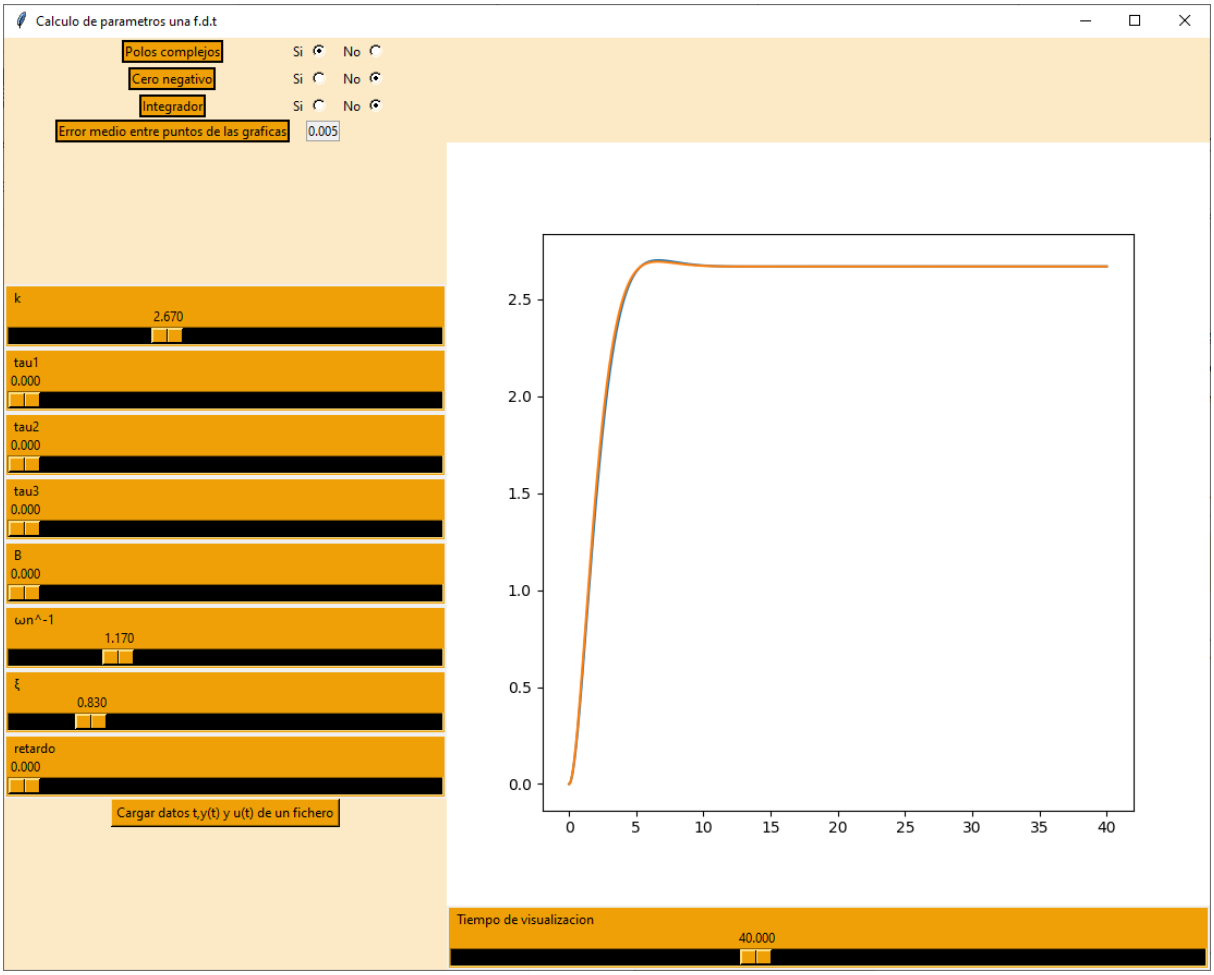
Antes de encontrar el valor de la deslizadera, se ha ajustado la precisión de la misma para poder ajustarla correctamente con la otra gráfica. Para ajustar la precisión de la deslizadera, se lleva la misma hasta uno de sus dos extremos, lo que hace que su valor máximo se duplique en el

caso de deslizarla al extremo derecho y se divida a la mitad en el caso izquierdo.

Se aprecia en la imagen 33 que la k es correcta y los polos complejos son demasiado oscilatorios, además también se pone en duda el polo real predeterminado □1, ya que la f.d.t de la gráfica azul parece demasiado rápida.

Eliminando el polo real y modificando el valor de los polos complejos, se consigue con una aproximación suficientemente fiable de la f.d.t experimental simulada. Se ve que el error que se muestra por pantalla en la imagen 34 se ha reducido considerablemente y es únicamente 0.005 entre los puntos de las gráficas.

Imagen 34: Representación parámetros de una f.d.t ejemplo 1 (ajuste 3)



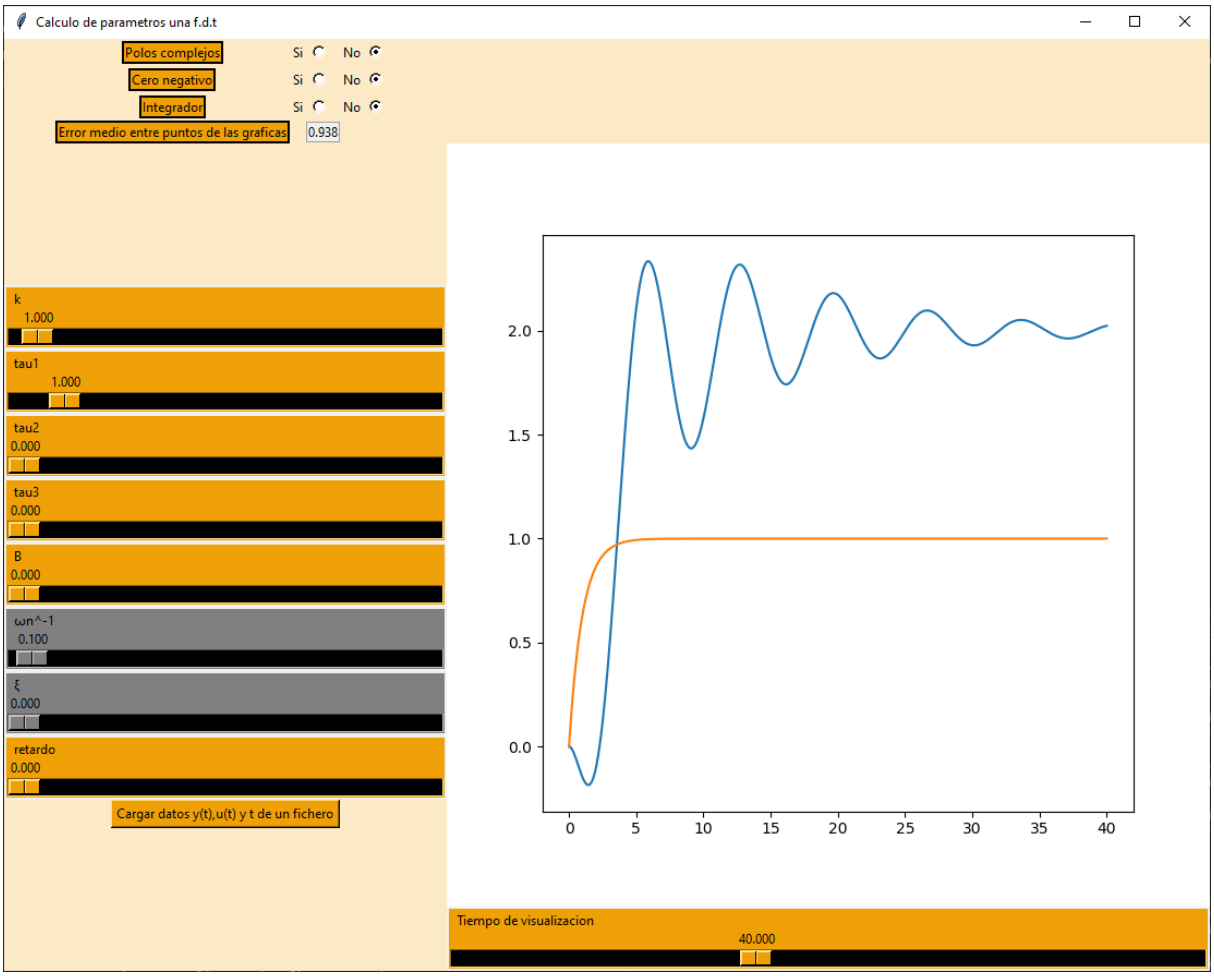
Fuente:Elaboración propia

- **Ejemplo 2**

Este ejemplo,como se muestra en la imagen 35, es mucho más oscilatorio que el anterior. Sin embargo no se puede ver el valor de establecimiento, así que será necesario mover el tiempo

de visualización para poder acertar con su medida.

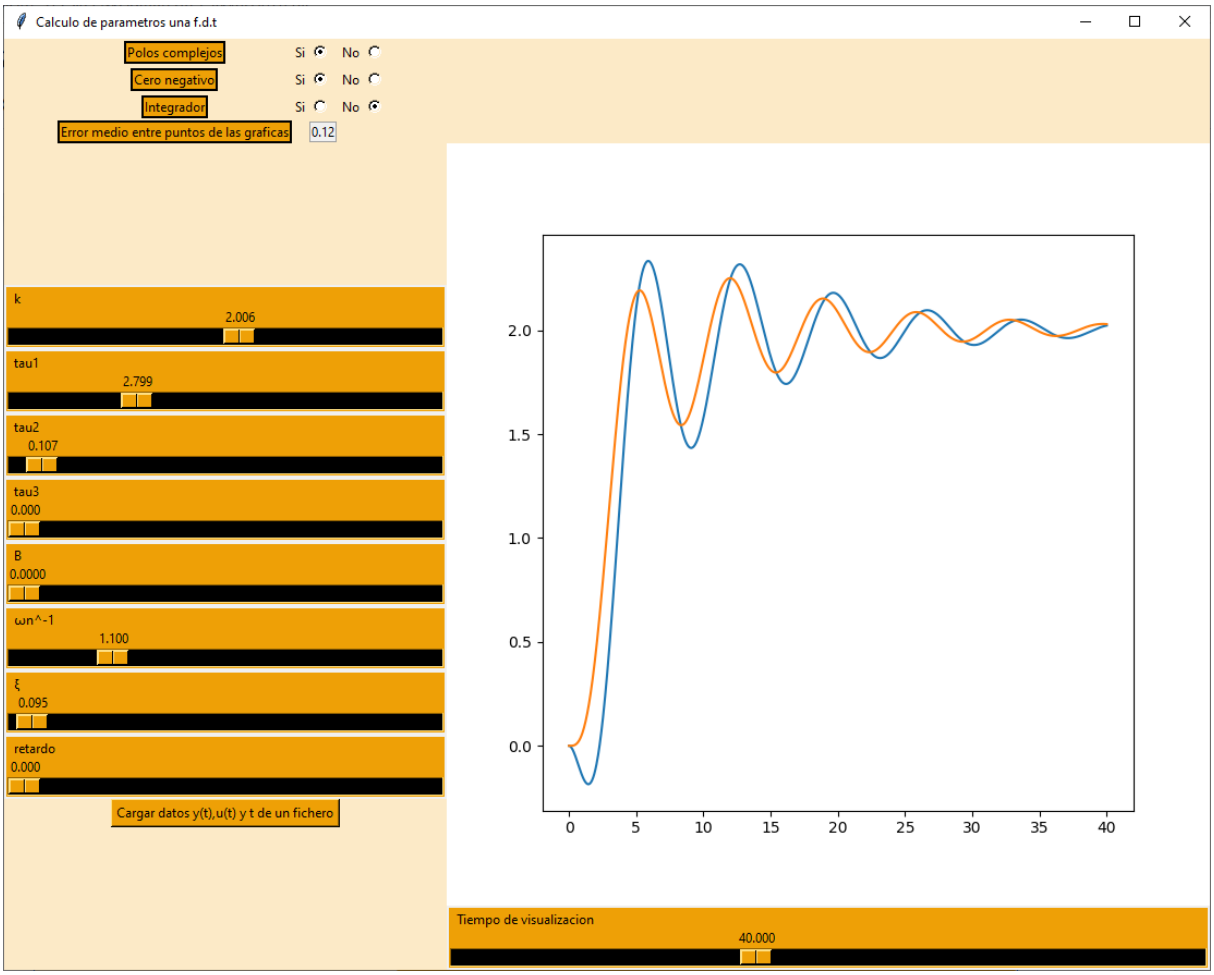
Imagen 35: Representación parámetros de una f.d.t ejemplo 2



Fuente:Elaboración propia

Debido a la característica oscilatoria del sistema, es más complicado hallar los valores □1 y □2, se ha modificado el tamaño de algunas deslizaderas para trabajar con mayor precisión. Llegando al ajuste mostrado en la imagen 36, todavía insuficiente.

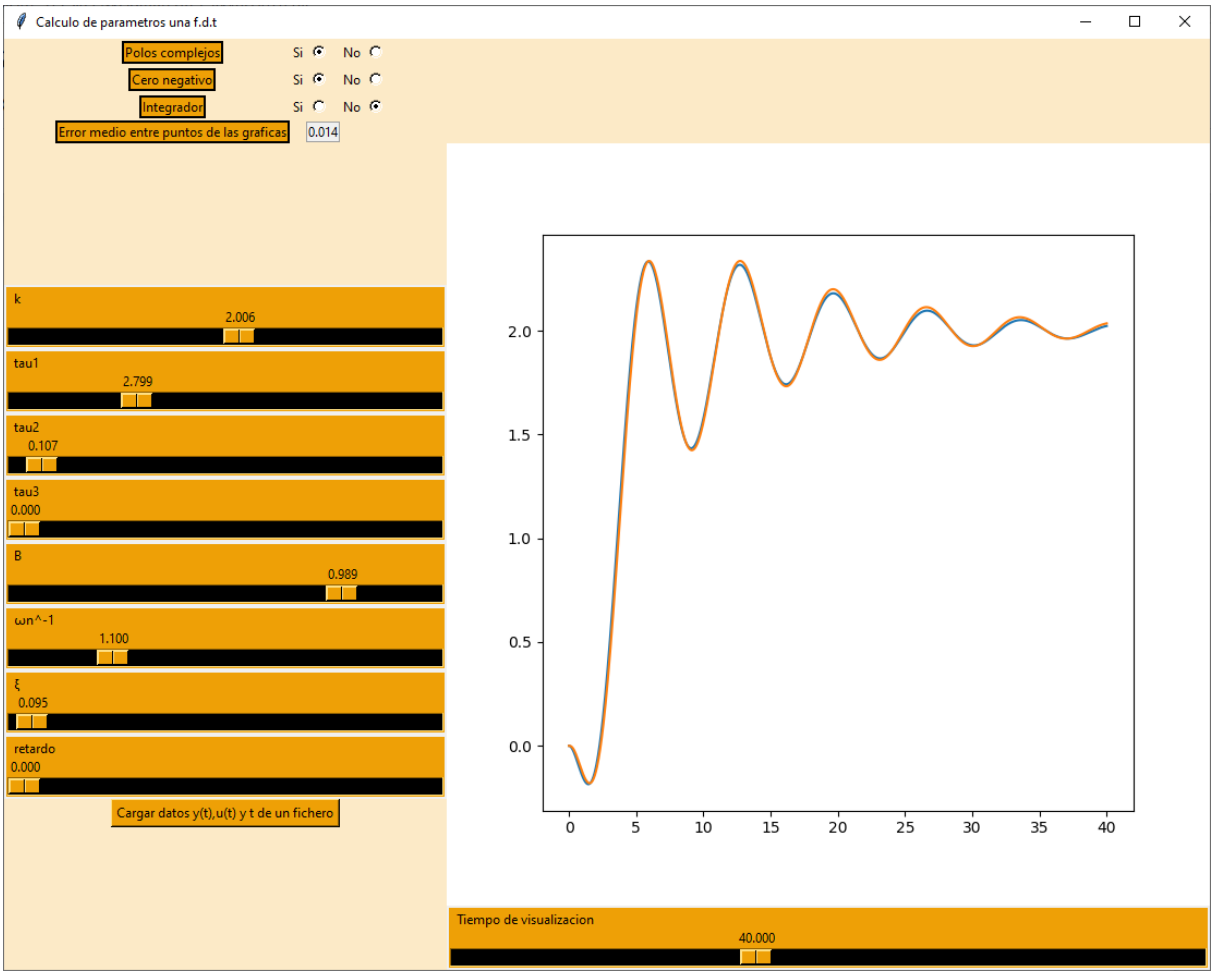
Imagen 36: Representación parámetros de una f.d.t ejemplo 2 (ajuste 2)



Fuente:Elaboración propia

Se aprecia también, que hay un cero negativo al inicio del sistema, así que habrá que habilitar la opción “cero negativo” por tanto, realizando las últimas modificaciones, el ajuste quedaría como el mostrado en la imagen 37.

Imagen 37: Representación parámetros de una f.d.t ejemplo 1 (ajuste 3)



Fuente:Elaboración propia

Se obtiene un error aceptable de 0.014, también se aprecia gráficamente que el comportamiento de ambas f.d.t es realmente similar.

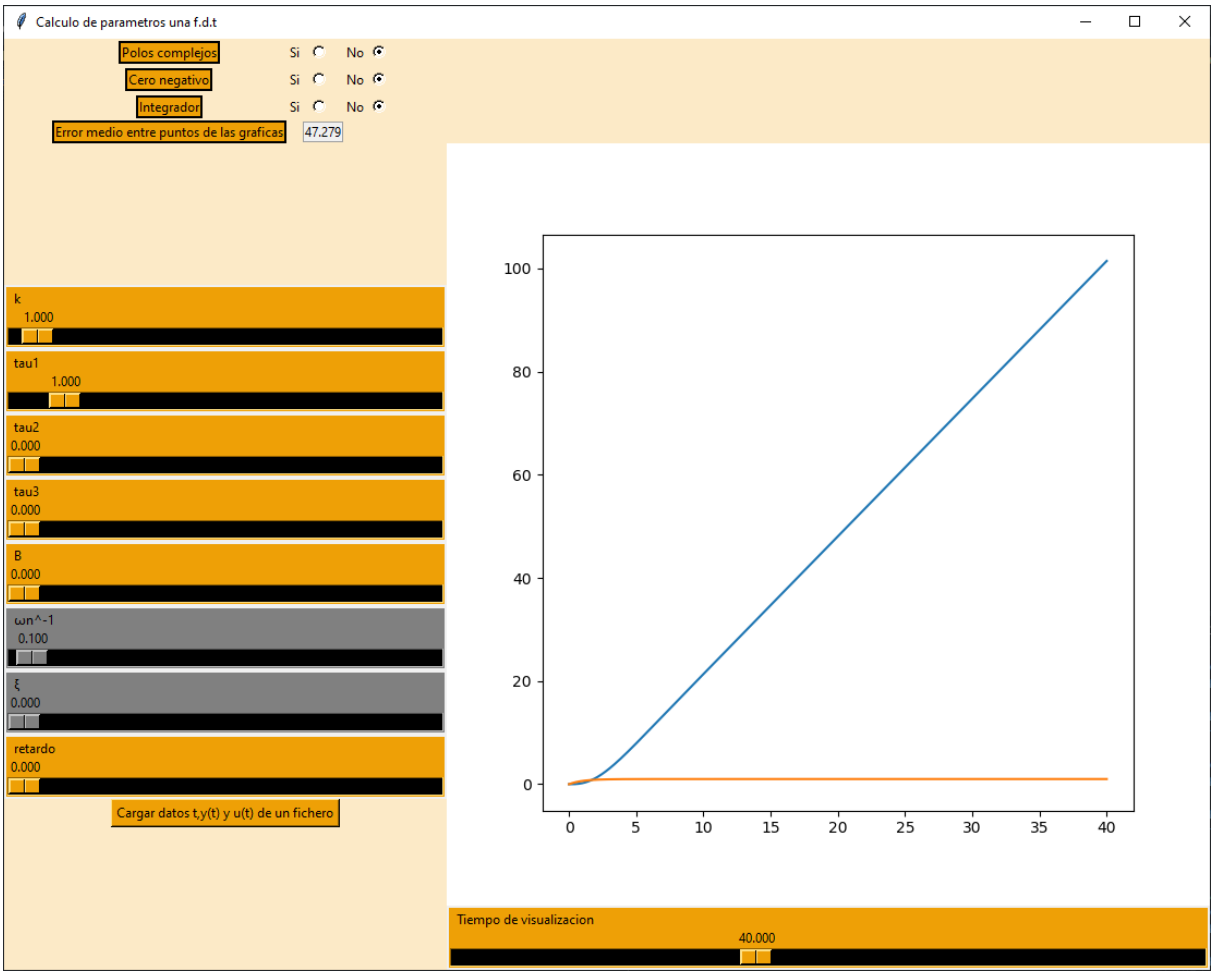
La f.d.t solución es:

$$\frac{2*(1-0.989s)}{(1+2.8s)*(1+0.1s)*}$$

- **Ejemplo 3**

En este ejemplo, como se observa en la imagen 38, se mostrará la eficacia de la f.d.t con integrador desarrollada:

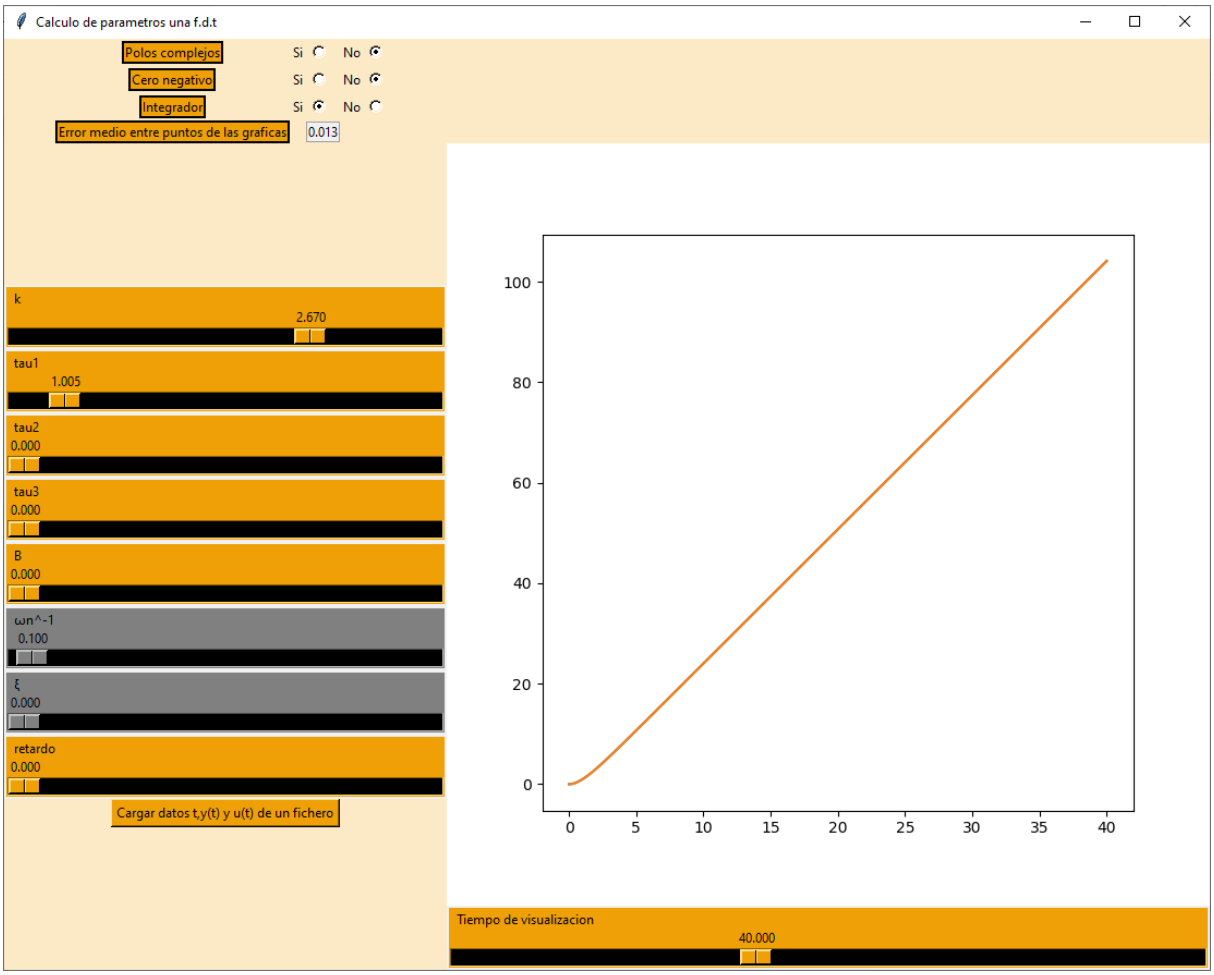
Imagen 38: Representación parámetros de una f.d.t ejemplo 3



Fuente:Elaboración propia

Habilitando el integrador y ajustando la ganancia k, se obtiene la representación del ajuste en la imagen 39.

Imagen 39: Representación parámetros de una f.d.t ejemplo 3 (ajuste 2)



Fuente:Elaboración propia

Adicionalmente se ha modificado un poco □1 para reducir aún más el error. Dado que el orden es de 100, el error de 0.013 es más que razonable.

- **Ejemplo 4**

Para este último ejemplo se ha creado un fichero de datos a modo de simulación con un fichero de datos experimentales, los datos del fichero son los siguientes:

Tabla 1: Datos del fichero (ejemplo 4)

t	y
0	0

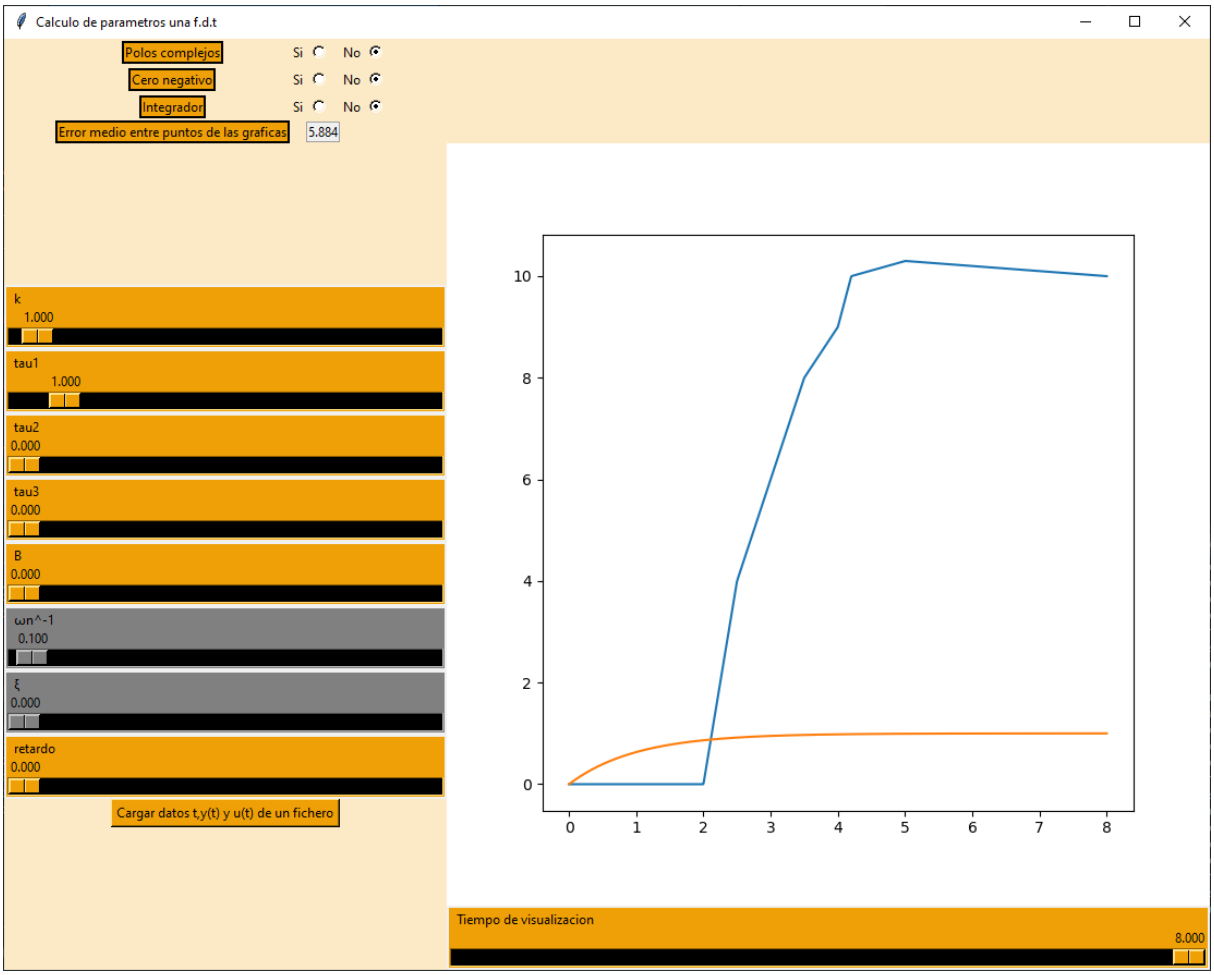
2	0
2.5	4
3	6
3.5	8
4	9
4.2	10
5	10.3
8	10

Fuente:Elaboración propia

Es relevante destacar que el programa interpreta los datos aunque su inserción tenga una sucesión de tiempos desigual, el programa modifica estos datos para que puedan ser utilizables.

La gráfica después de hacer click en el botón “Cargar datos t,y(t) y u(t) de un fichero” es la mostrada en la imagen 40:

Imagen 40: Representación parámetros de una f.d.t ejemplo 4

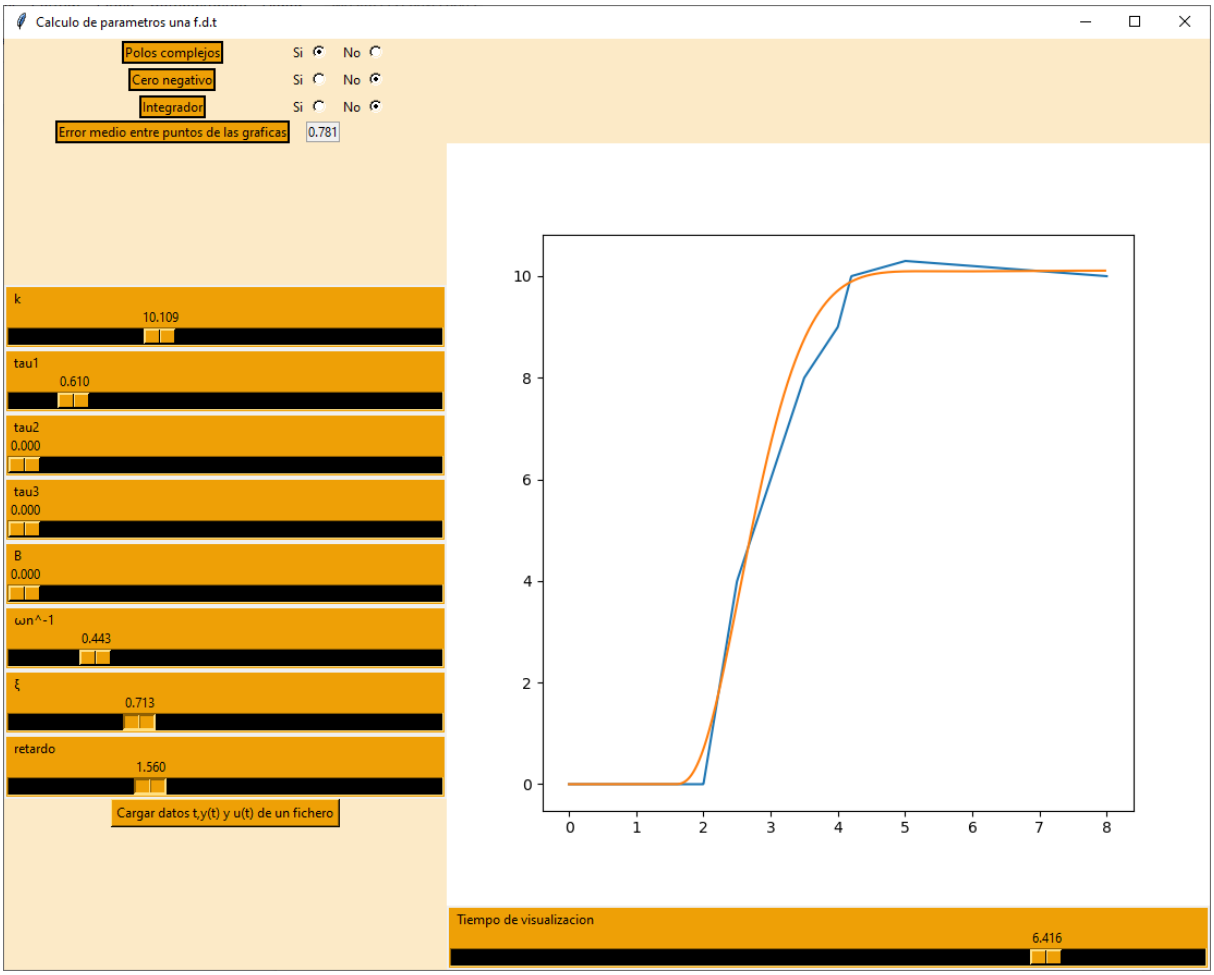


Fuente:Elaboración propia

Se aprecia como la gráfica naranja es la típica predeterminada que se ha visto en los anteriores ejemplos, esto es debido a que los datos insertados no poseían una u(t) y el programa establece una señal de entrada escalón automáticamente.

Se ha colado un retardo y se han habilitado los polos complejos para optimizar la curva, dado la irregularidad y que realmente los datos no seguían ningún tipo de f.d.t, no se puede esperar alcanzar un error demasiado bajo. El error es de 0.781 como se observa en la imagen 41.

Imagen 41: Representación parámetros de una f.d.t ejemplo 4 (ajuste 2)



Fuente:Elaboración propia

9.2.2 Conclusión

Esta aplicación requiere de conocimientos específicos a la hora de utilizarla, su utilidad didáctica merece ser mencionada ya que la posibilidad de modificar y observar el comportamiento de la gráfica da mucha información acerca de cómo funcionan las f.d.t. Adicionalmente, es sencillo modificar el programa para poner una f.d.t predeterminada concreta, permitiendo al posible instructor crear diversos ejercicios de práctica con facilidad. Esta aplicación también funciona de forma rápida y no suele colgarse.

9.3 ‘F.d.t-ante-diferentes-entradas’

Esta aplicación se correlaciona directamente con la anterior, ya que esta permite a una f.d.t someterse a diferentes tipos de entrada. Permite asignar una entrada frente a impulso, escalón,

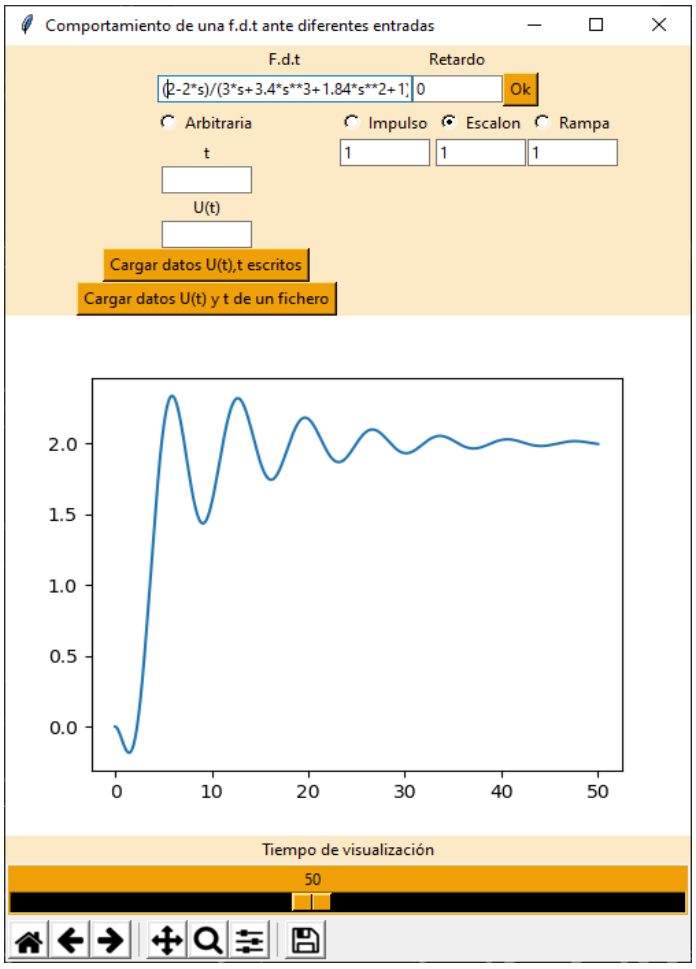
rampa y además, se le puede asignar de forma arbitraria cualquier otro valor de entradas en el tiempo, también se puede introducir esta información a través de un fichero de datos. Esto permitiría calcular una f.d.t con la aplicación anterior y posteriormente introducirla en esta otra, aplicando una nueva referencia arbitraria para corroborar el correcto ajuste anterior de la f.d.t. Sería posible introduciendo los valores de entrada deseados, predecir el comportamiento del sistema.

9.3.1 Ejemplos

- **Ejemplo 1:**

Se va a utilizar la ecuación oscilatorio del ejemplo 2 del punto 10.2.1, '**Obtencion-de-una-f.d.t**', para contrastar su veracidad. Seleccionando el botón de escalón para una entrada escalón, se genera la curva mostrada en la imagen 42. Como se puede apreciar, la curva es equivalente a la desarrollada anteriormente. Adicionalmente, es posible cambiar el tamaño del impulso, de la rampa, y del escalón.

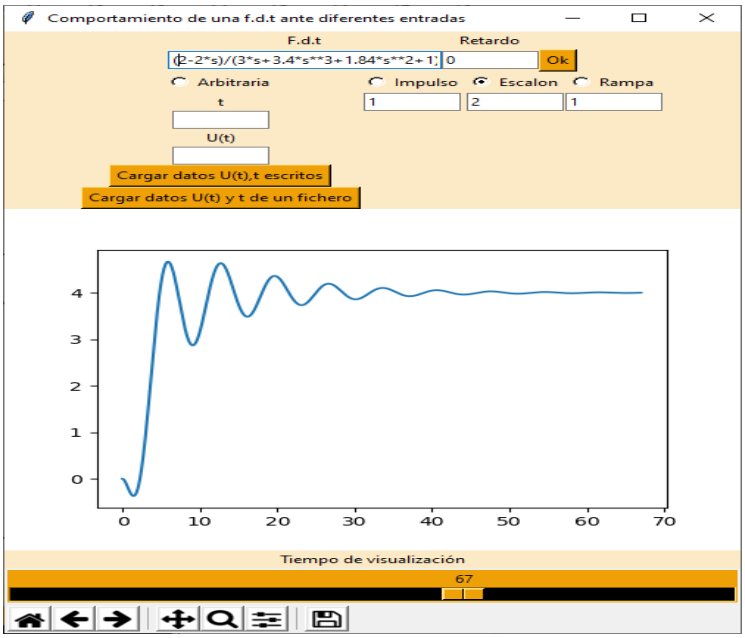
Imagen 42: Representación una f.d.t según sus entradas ejemplo 1



Fuente:Elaboración propia

Duplicando el tamaño del escalón, se aprecia en la imagen 43 como se duplica el valor final y las sobre oscilaciones del sistema.

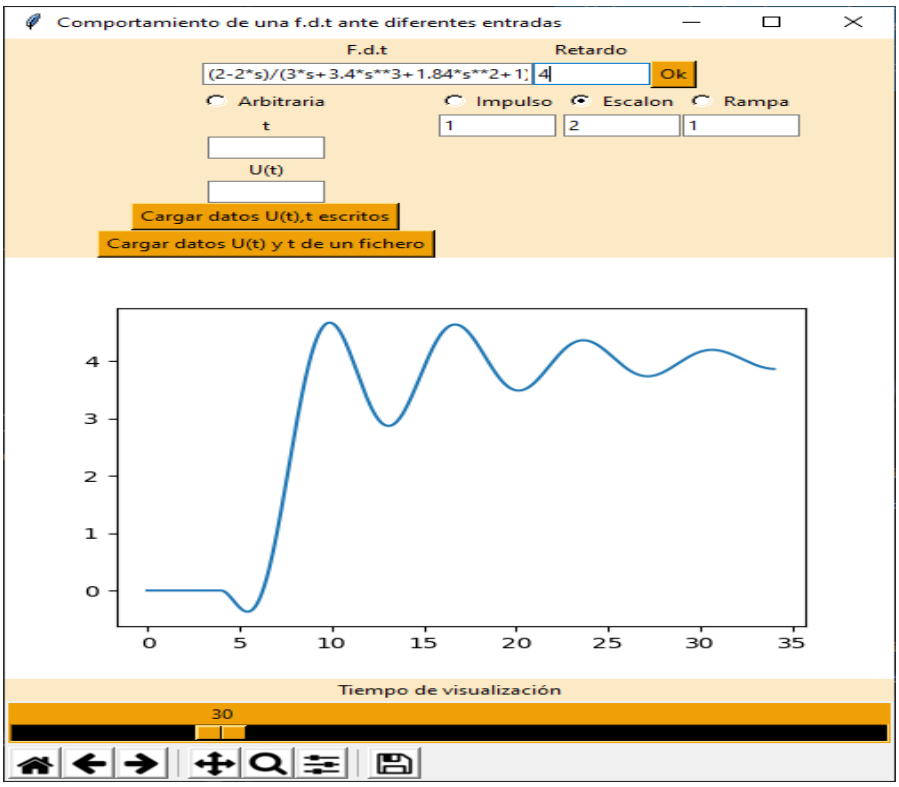
Imagen 43: Representación una f.d.t según sus entradas ejemplo 1 (entrada escalón=2)



Fuente:Elaboración propia

El sistema también puede simular el retardo escribiéndolo en el cuadrado correspondiente. tal como muestra la imagen 44.

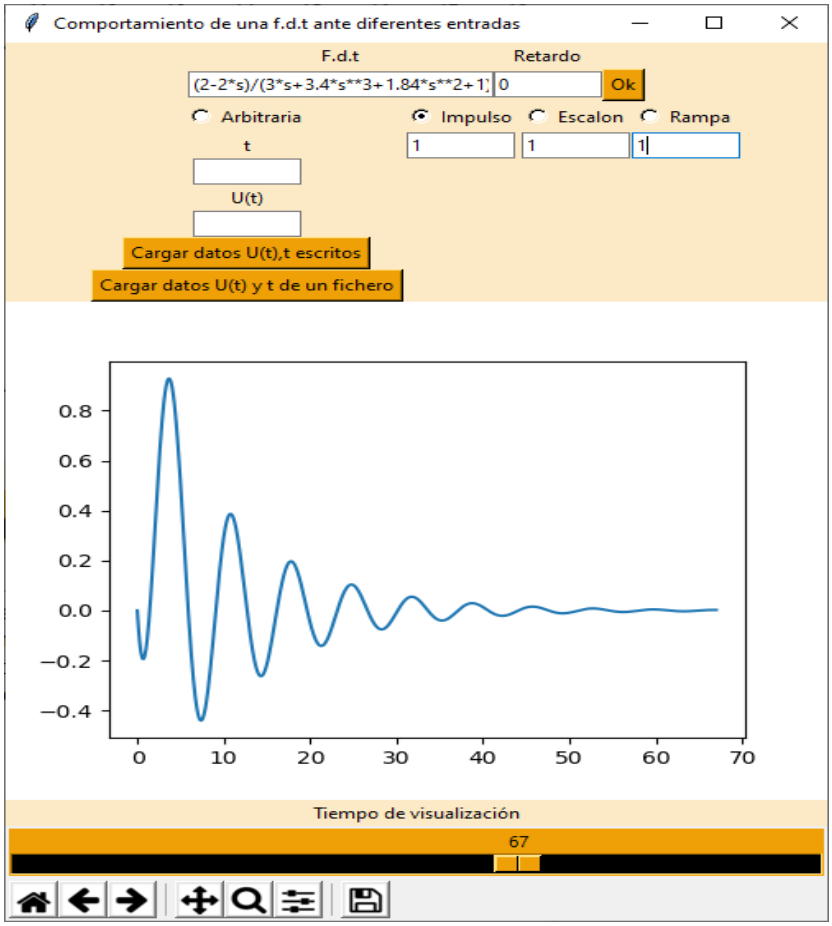
Imagen 44: Representación una f.d.t según sus entradas ejemplo 1 (retardo=4)



Fuente:Elaboración propia

Ante referencia impulso, se aprecia como la función acaba superándolo tras unos 60 segundos, es decir un sistema físico con esta f.d.t tardaría ese tiempo en estabilizarse, cosa que se podría testear en el caso de ser una f.d.t que se quiera poner a prueba.

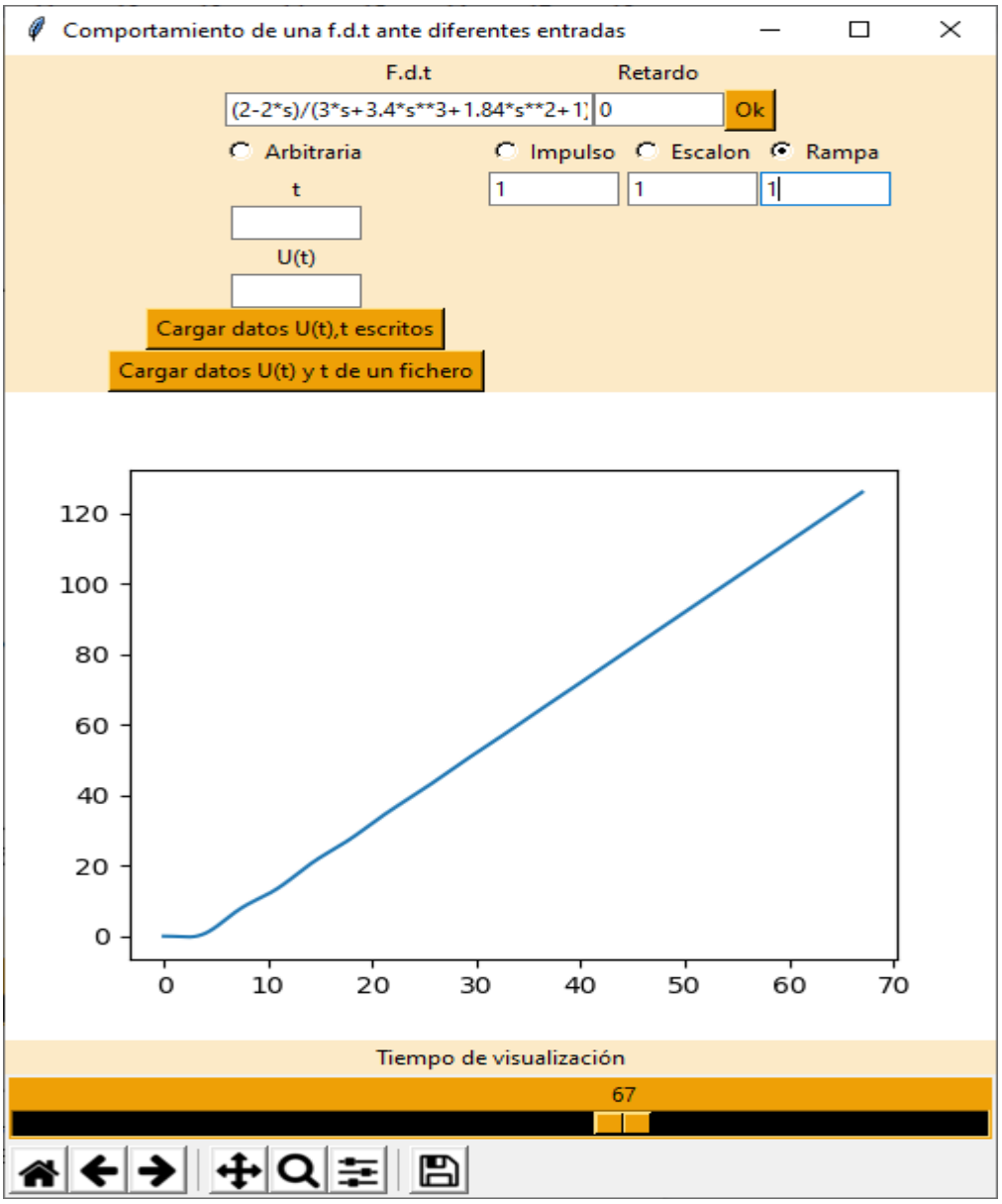
Imagen 45: Representación una f.d.t según sus entradas ejemplo 1 (impulso)



Fuente:Elaboración propia

La referencia rampa funciona como se esperaría. (ligeras perturbaciones en el inicio de la rampa)

Imagen 46: Representación una f.d.t según sus entradas ejemplo 1 (rampa)



Fuente:Elaboración propia

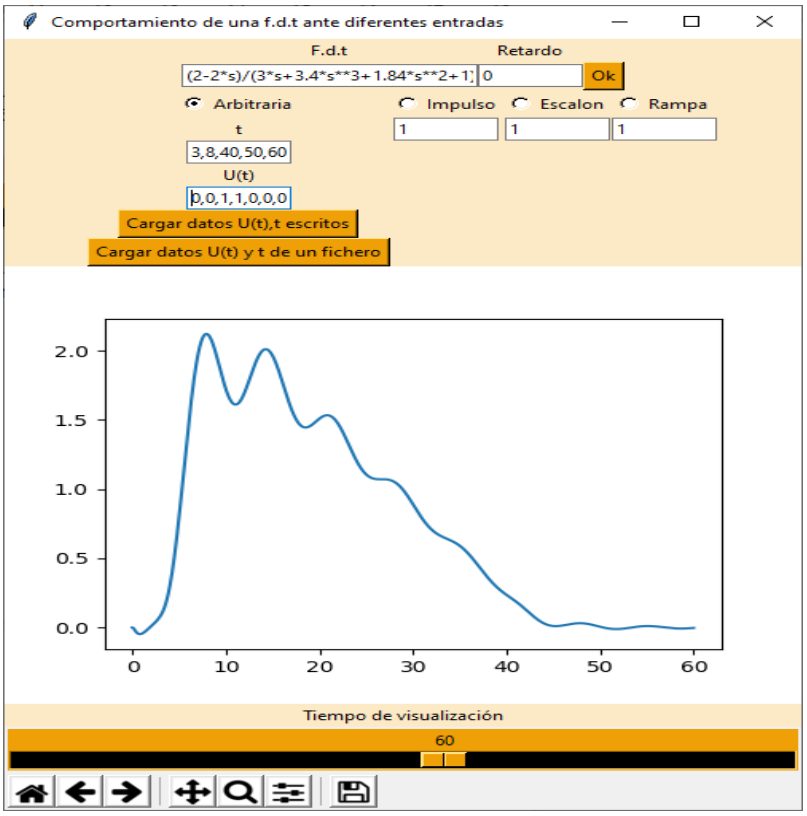
Por último, se van a escribir manualmente los siguientes datos:

t=[0,0.2,0.3,0.5,0.7,3,8,40,50,60]

u(t)=[1,1,0,0,0,1,1,0,0,0]

En la imagen 46, se aprecia que el sistema se comporta se manera esperada con los inputs , también se podría haber colocado cualquier valor de entrada distinto a 0 o a 1.

Imagen 47: Representación una f.d.t según sus entradas ejemplo 1(entrada arbitraria)



Fuente:Elaboración propia

9.3.2 Conclusion

Esta aplicación es un buen complemento de '[Obtencion-de-una-f.d.t]' por lo mencionado previamente. Si bien más limitada que las anteriores, su aplicación didáctica puede resultar atractiva porque permite describir el comportamiento de las f.d.t de una manera distinta a '[Representa_la_inversa_de_laplace]'.

9.4 "Respuesta-de-PID"

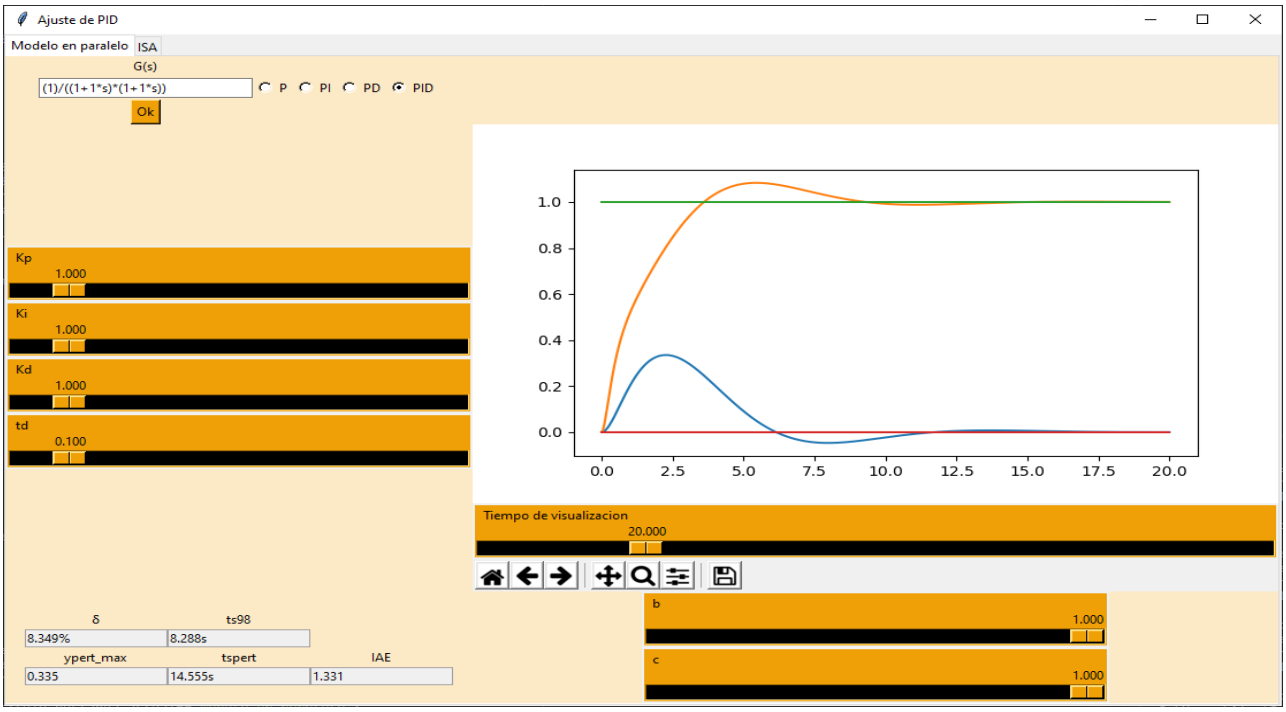
Esta aplicación permite, dada una G(s), calcular el comportamiento ante cambio de referencia escalón y ante perturbación, rectificadas por diversos tipos de controladores. Este software calcula el comportamiento de los controladores P, PI, PD y PID, como se mencionó en la introducción, operan con ponderación y filtrado del término derivativo. Esta aplicación es el paso último para conseguir controlar eficazmente un sistema; una vez obtenida y comprobada la f.d.t del sistema, se puede buscar aquí el comportamiento de control que más se ajuste a las necesidades requeridas, según los parámetros característicos de los PID.

9.4.1 Ejemplos

- **Ejemplo 1:**

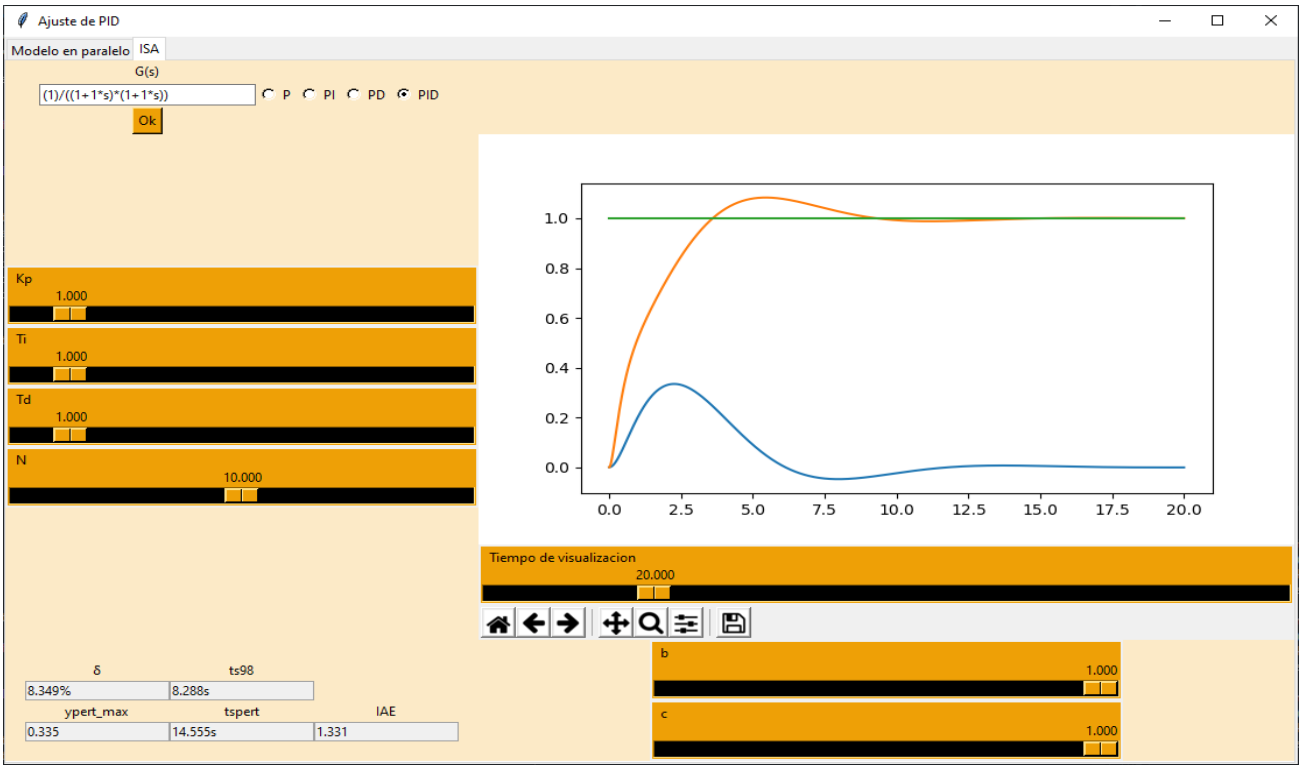
En este ejemplo se describirá la aplicación paso por paso, en la imagen 48, la gráfica naranja hace referencia ante la entrada por cambio de referencia, y la azul la respuesta ante perturbación, se puede observar abajo a la izquierda como se muestran los distintos valores a tener en cuenta a la hora de optimizar el PID. Además, se aprecia con claridad las distintas opciones para modificar el PID, los parámetros de ponderación a y b se encuentran debajo de la gráfica, los parámetros habituales del PID en paralelo, Kp, Ki y Kd, se encuentran a la derecha de la gráfica junto con td, la variable de filtrado de la derivada. Arriba de estas deslizaderas, se encuentra el cuadrado para rellenar con la G(s) deseada, y a su derecha el tipo de controlador que se desea utilizar. Arriba del todo se pueden intercambiar las pestañas, para pasar de formato en paralelo al formato ISA.

Imagen 48: Respuesta del PID, ejemplo 1 modelo en paralelo



Fuente:Elaboración propia

Imagen 49: Respuesta del PID, ejemplo 1 modelo ISA



Fuente:Elaboración propia

Se aprecia cómo las variables de la derecha de la gráfica en la imagen 49 han cambiado, sin embargo no lo ha hecho el comportamiento de la gráfica ya que estos valores son los equivalentes del modo en paralelo.

- **Ejemplo 2:**

Una empresa con cierto sistema físico descrito como:

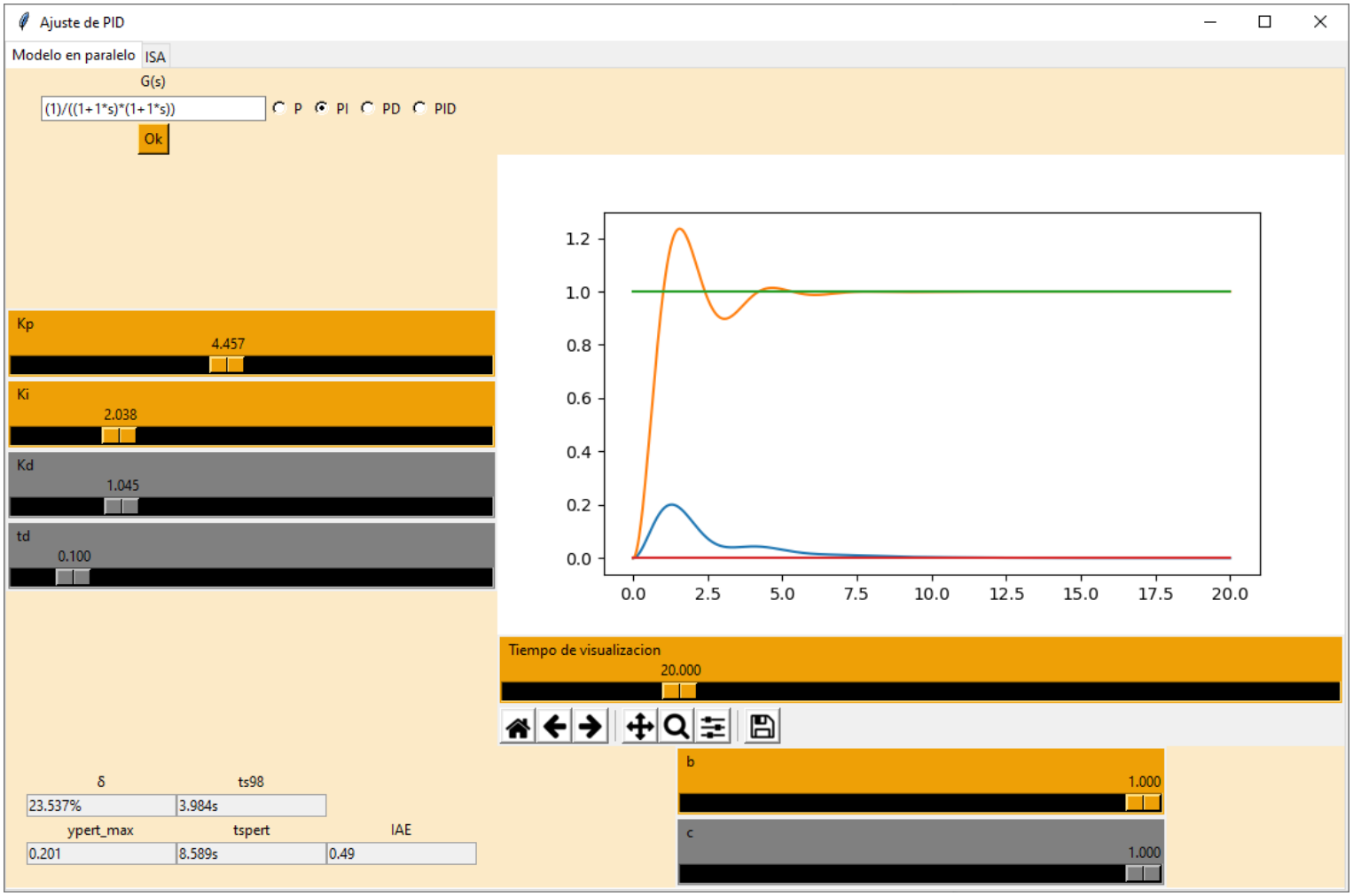
$$\frac{1}{(1+s)*(1+s)}$$

Propone comprar un controlador PI que consiga una sobreoscilación menor del 5% y un tiempo de establecimiento al 98% de 4s.

Para estimar su viabilidad se hace uso de la aplicación 'Respuesta-de-PID'. Después de probar diversas modificaciones, se obtienen los valores Kp=4.457 Ki=2.038, b=1 $\delta = 23.53\%$ ts98=3.984s, según muestra la imagen 50. Ha sido complicado obtener el tiempo de establecimiento, ya que no se dispone del control derivativo para aumentar la velocidad. Aún queda por ajustar el valor de la sobreoscilación, para ello se utilizará el factor de ponderación b; sin embargo no se espera poder conseguir ambos requisitos simultaneamente, ya que apenas

hay margen en los valores de Kp y Ki para permanecer en ts98, y reducir la sobreoscilación con el factor b disminuye la velocidad de control.

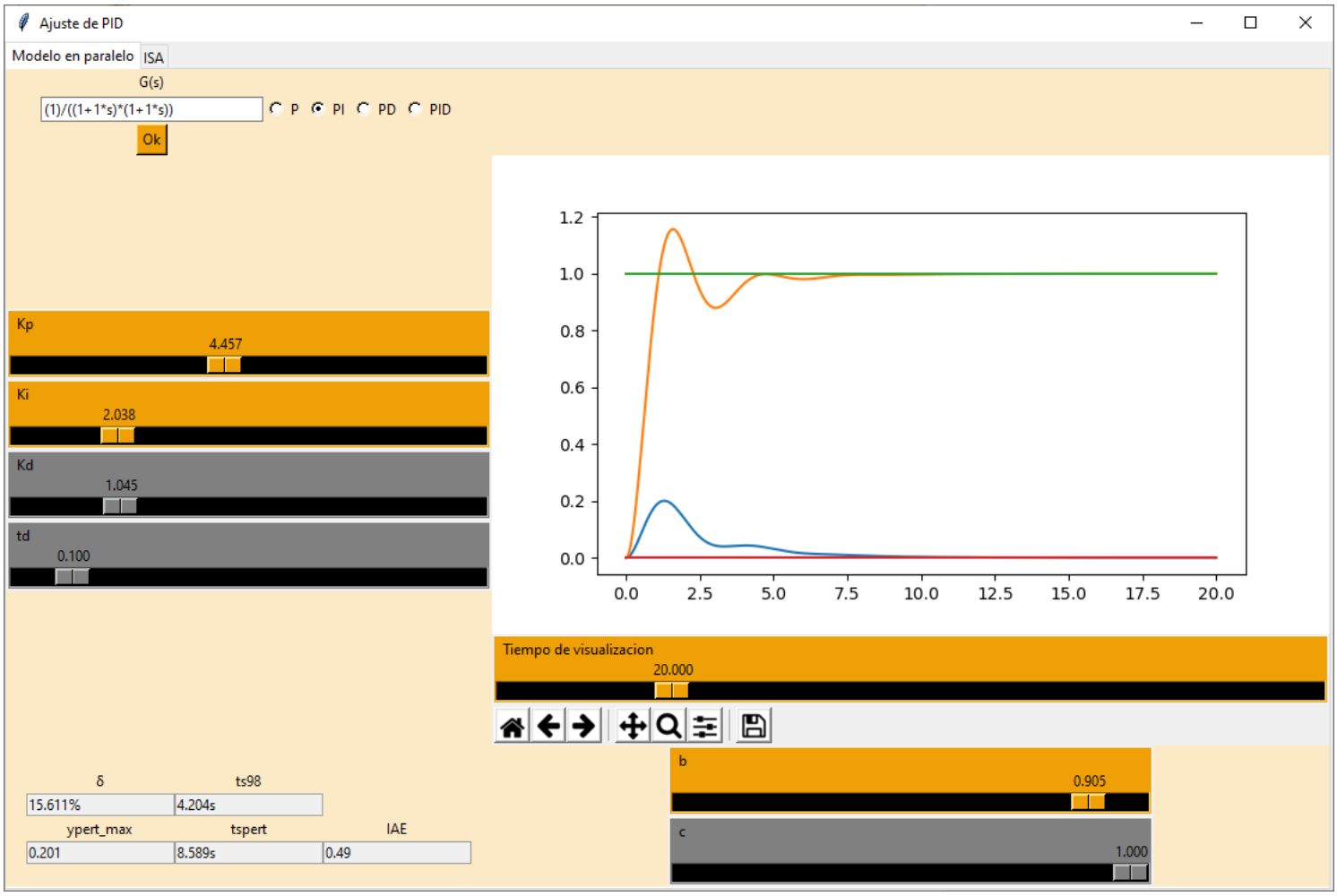
Imagen 50: Respuesta del PID, ejemplo 2 (primera modificación)



Fuente:Elaboración propia

Después de realizar algunos ajustes, se han conseguido los resultados: Kp=4.457 Ki=2.038, b=0.905 $\delta = 15.611\%$ ts98=4.204s, como se muestra en la imagen 51. Este es el mejor comportamiento del PI si ambos criterios son igual de relevantes, ya que reducir un poco más el valor de la sobreoscilación hace que el tiempo al 98% aumente en 2s.

Imagen 51: Respuesta del PID, ejemplo 2 (segunda modificación)

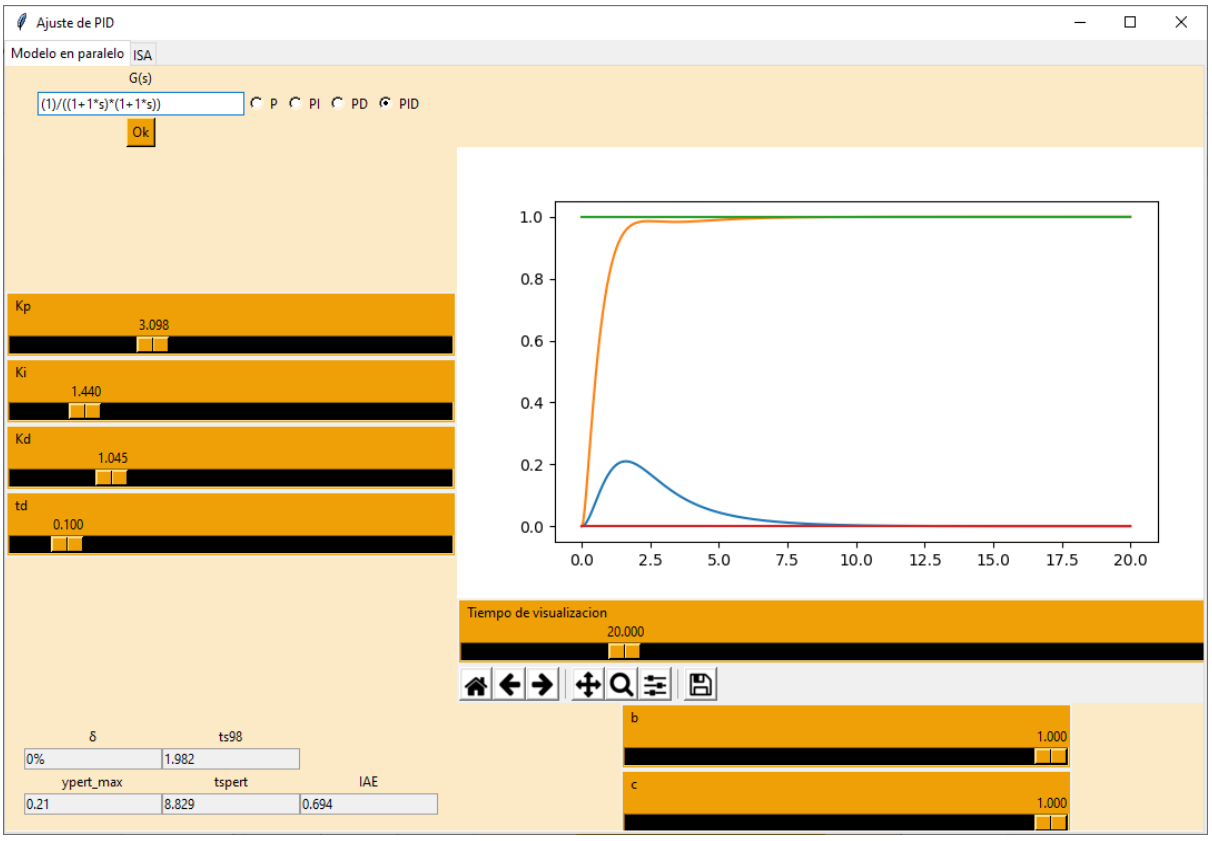


Fuente:Elaboración propia

No ha sido posible ajustar el PI a las especificaciones requeridas, por lo que se propone el control PID como opción alternativa de compra.

Con Kp=3.01 Ki=1.44 Kd=1.045 td= 0.1 y b=1. como se muestra en la imagen 52, se consigue una muy buena respuesta con menos de 2s de ts98 y 0% de sobreoscilación.

Imagen 52: Respuesta del PID, ejemplo 2 (tercera modificación)

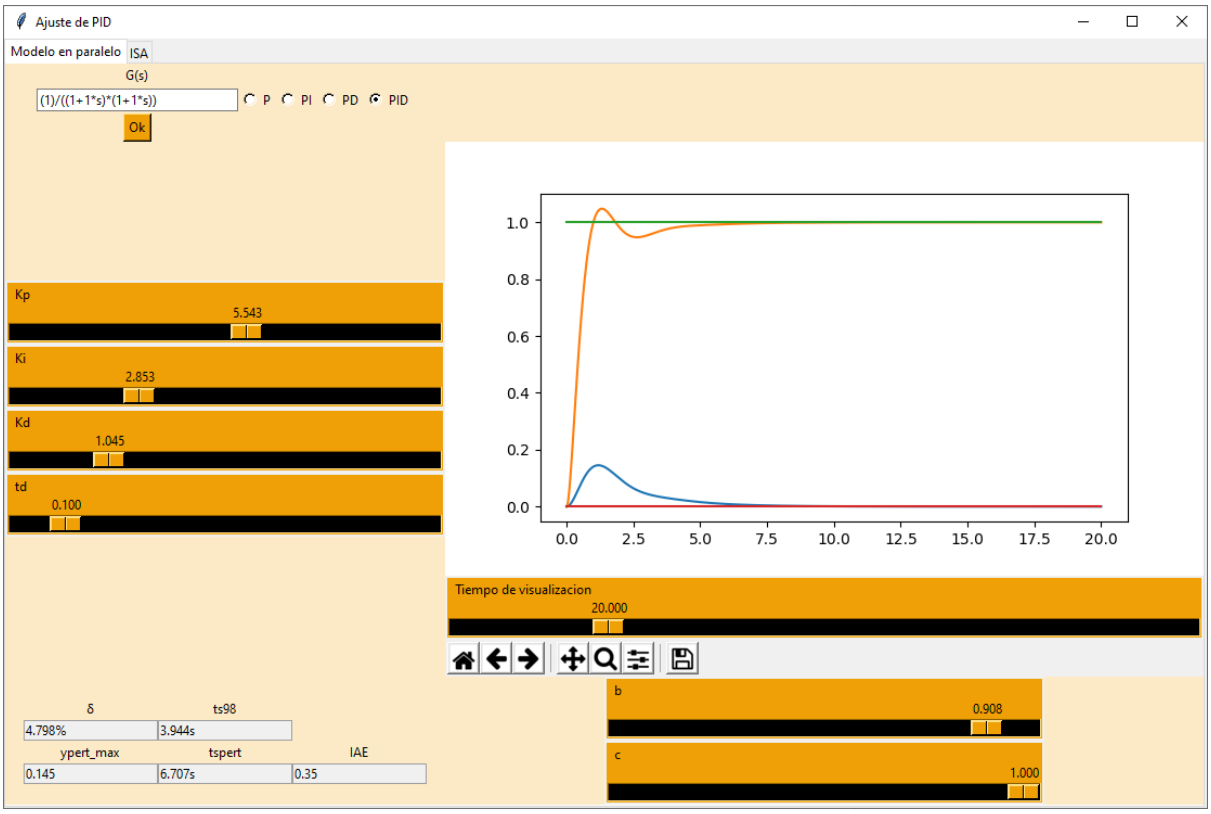


Fuente:Elaboración propia

Utilizando los valores: Kp=5.543, Ki=2.854 Kd=1.045 td=0.1 y b=0.908 según muestra la imagen 53, se consigue un comportamiento límite al requerido por la empresa, la sobreoscilación es del 4.8% y el ts98 son 3.9s.

El control PID es es una mejor opción para realizar la tarea deseada por la empresa.

Imagen 53: Respuesta del PID, ejemplo 2 (cuarta modificación)



Fuente:Elaboración propia

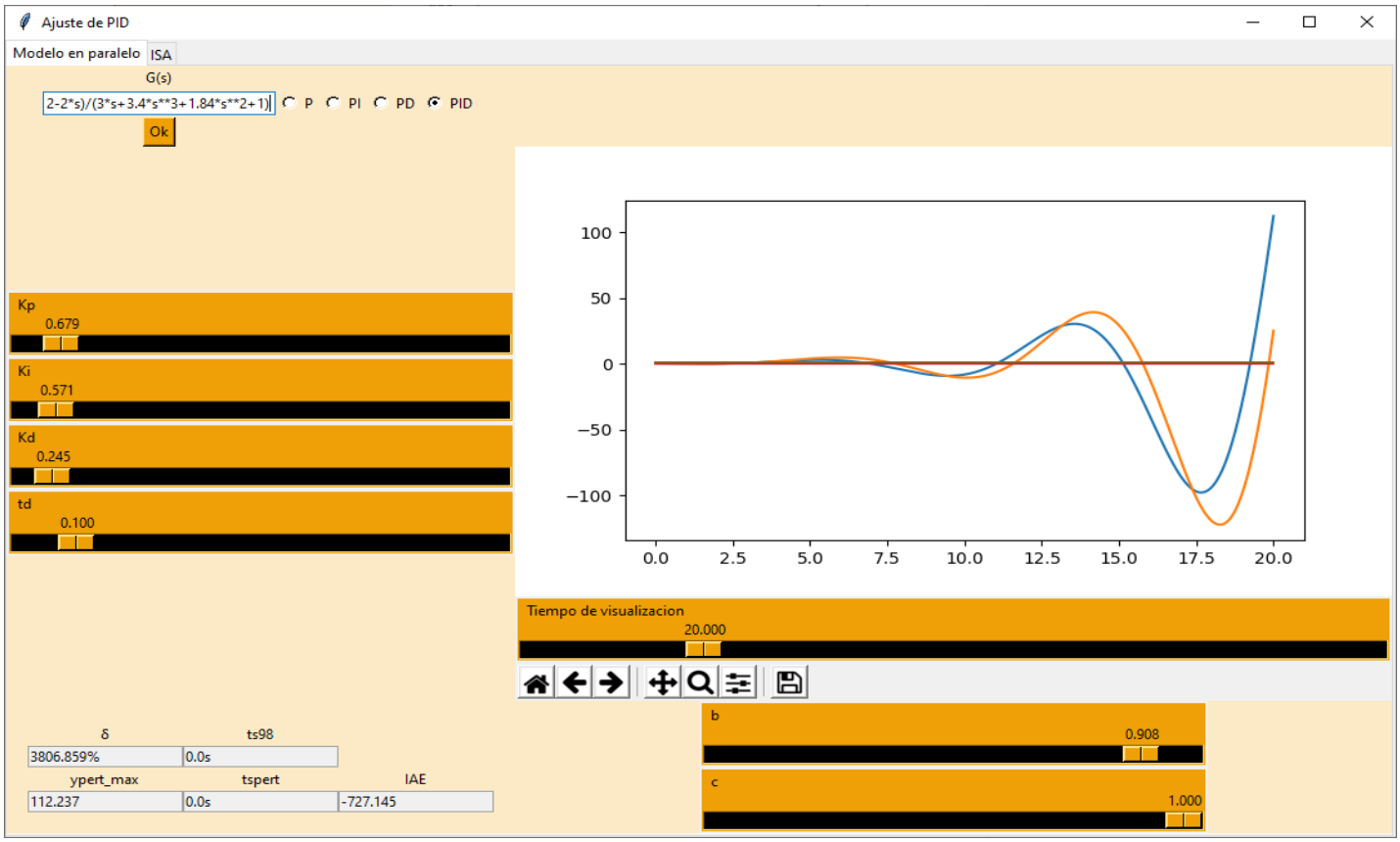
- **Ejemplo 3:**

Una vez comprobada la veracidad de la ecuación oscilatoria de las aplicaciones anteriores(Ejemplo 2 del punto 10.2.1, Ejemplo 1 del punto 10.3.1) descrita como:

$$\frac{2*(1-0.989s)}{(1+2.8s)*(1+0.1s)*}$$

Se requiere diseñar un controlador capaz de hacer que estabilice un cambio de referencia en menos de 25s y reducir su IAE por debajo de 15, el tiempo que esta ecuación tardaba en alcanzar un valor final por si misma era de 60s. La respuesta con el controlador PID es inestable y el sistema nunca alcanza el equilibrio, el término derivativo amplifica demasiado las oscilaciones del sistema. Esto está representado en la imagen 54.

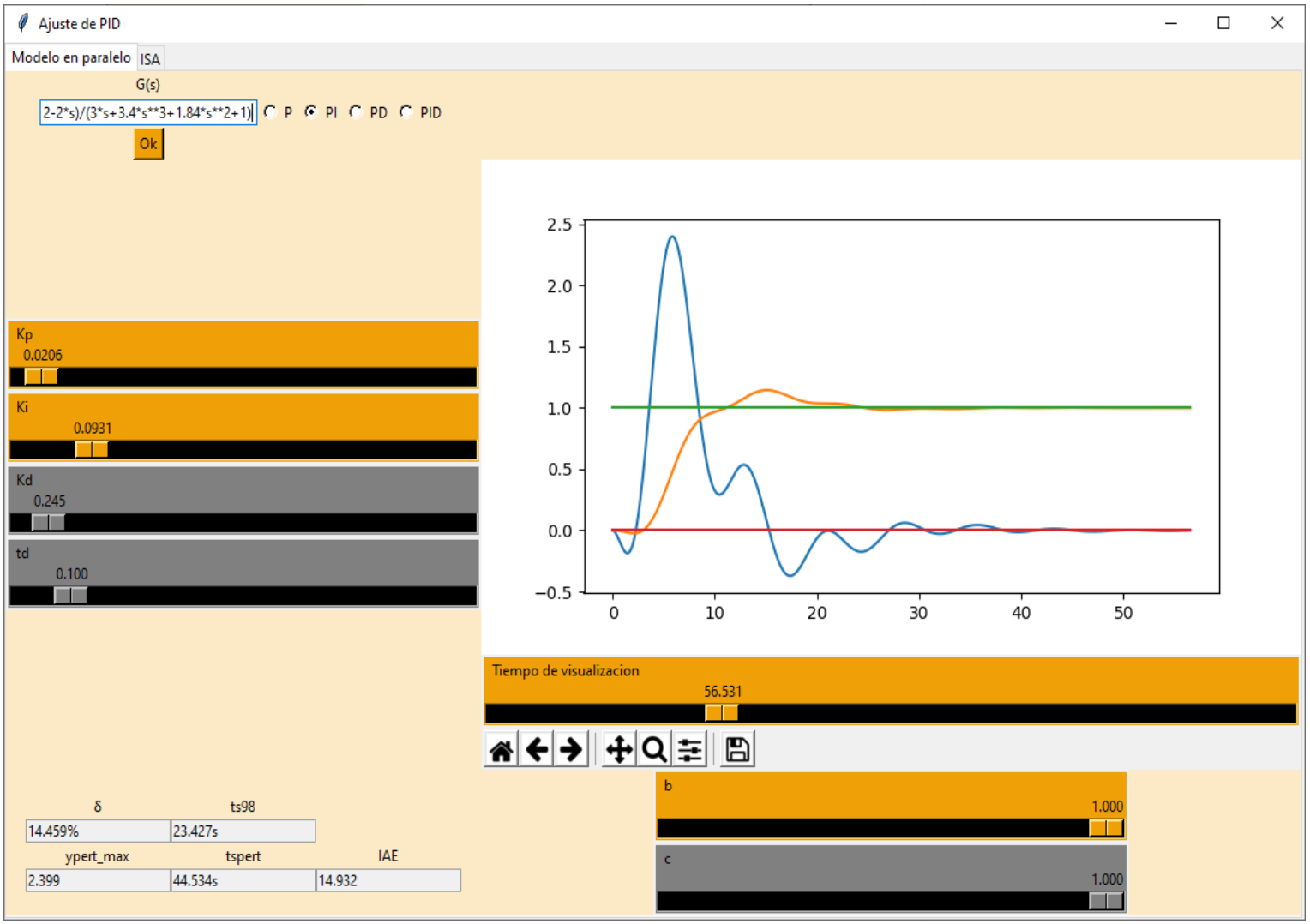
Imagen 54: Respuesta del PID, ejemplo 3



Fuente:Elaboración propia

Después de cambiar el tipo de controlador a PI y encontrar los valores aproximadamente óptimos, se obtienen los valores Kp=0.0206 IAE=14.932 $\delta = 14.459\%$ ts98=23.427s IAE=14.932, tal como muestra la imagen 55. Se ha conseguido cumplir con los criterios requeridos, sin embargo es muy difícil reducir el gran pico inicial en la perturbación o mejorar mucho más cualquier aspecto considerado favorable. Reducir el factor de ponderación b apenas tiene ningún efecto en la sobreoscilación o el ts98 ya que Kp es muy pequeño. Los sistemas muy oscilatorios son difíciles de controlar.

Imagen 55: Respuesta del PID, ejemplo 3 (segunda modificación)



Fuente:Elaboración propia

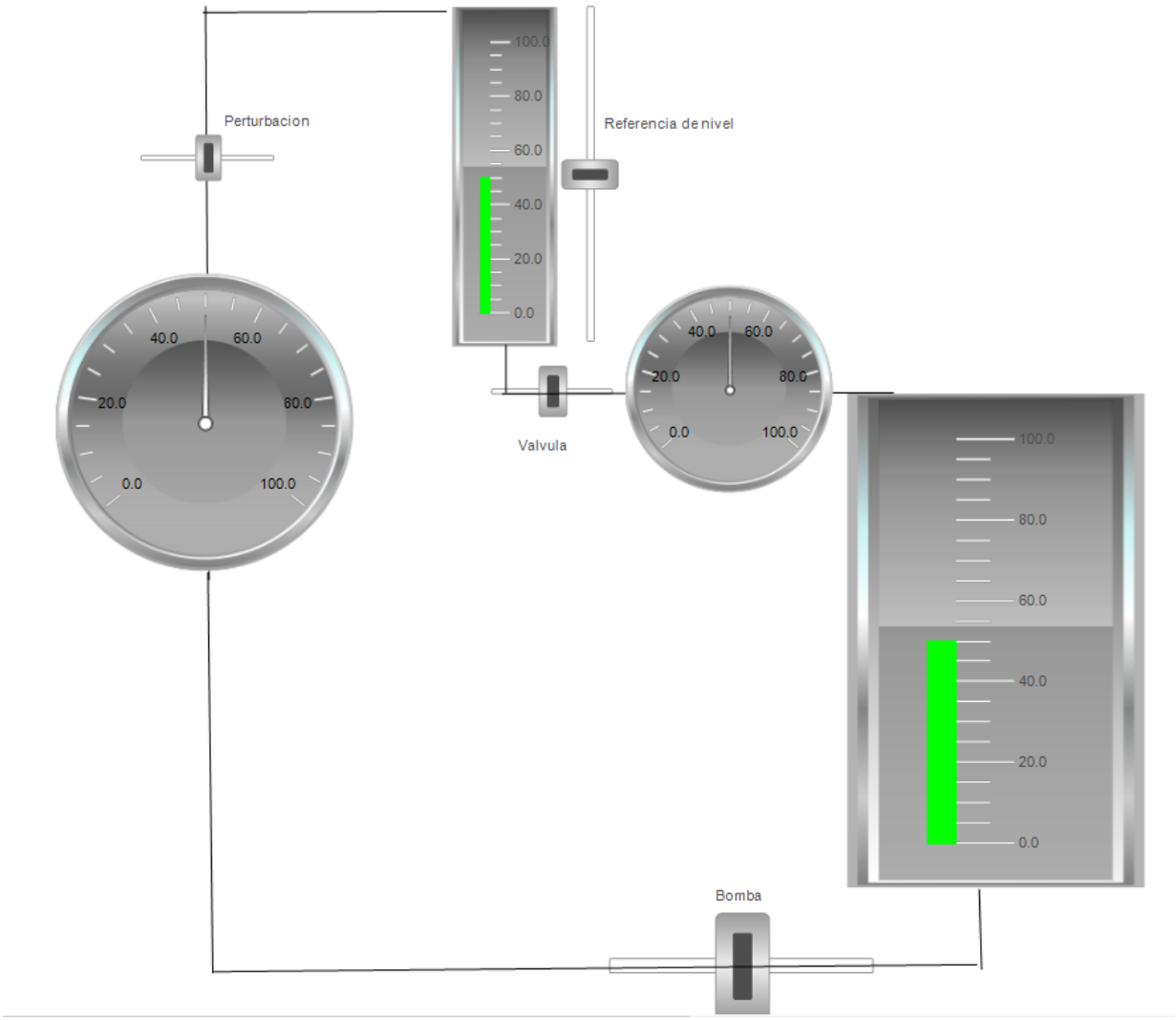
9.4.2 Conclusion

Esta aplicación es la más importante en el ámbito de control ya que le da una finalidad al resto, tiene una buena utilidad industrial gracias a sus múltiples opciones, y desde el apartado didáctico también funciona ya que se puede apreciar el comportamiento de los distintos valores de los PID con facilidad.

9.5 Ejemplo final

Utilizando el programa CODESYS se ha simulado un sistema de dos depósitos de agua, el sistema queda representado mediante la imagen 56.

Imagen 56: Representación del sistema de CODESYS



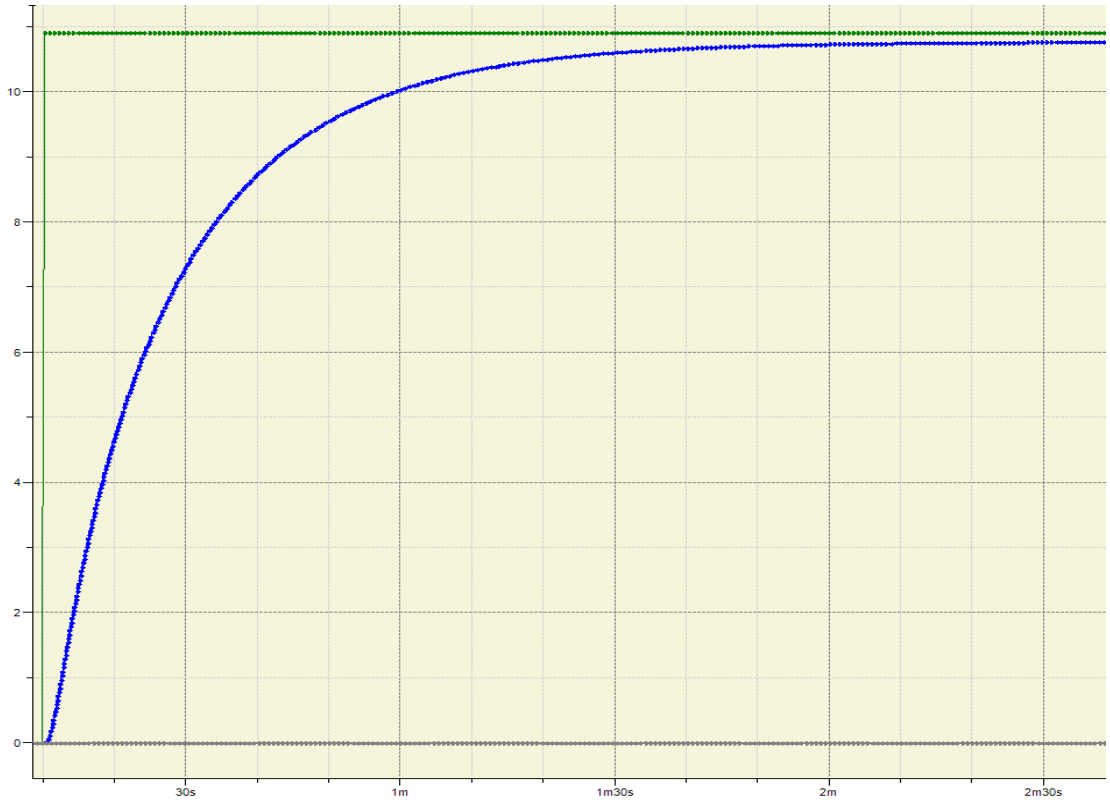
Fuente:Elaboración propia

Se tiene un depósito inferior y uno superior, la válvula se encuentra permanentemente abierta al 30%, de manera que siempre cae agua al depósito inferior. El depósito inferior posee una bomba con la que envía agua al depósito superior. Existe un perturbador para simular perturbaciones y dos caudalímetros que dan información sobre el caudal que circula por los mismos.

Se pretende encontrar un controlador PID capaz de establecer una referencia de nivel en el depósito superior.

Se opta por obtener la f.d.t mediante datos experimentales, para ello, primero se realiza un ensayo experimental en la herramienta CODESYS , dando un valor de entrada a la bomba y esperando a que se estabilice. Se almacenan los datos obtenidos en un fichero llamado “datos” en formato txt. Se observa el comportamiento del sistema en la imagen 57, siendo la línea verde la perteneciente a la bomba y la azul la perteneciente al nivel.

Imagen 57: Sistema de CODESYS ante entrada escalón sin controlador

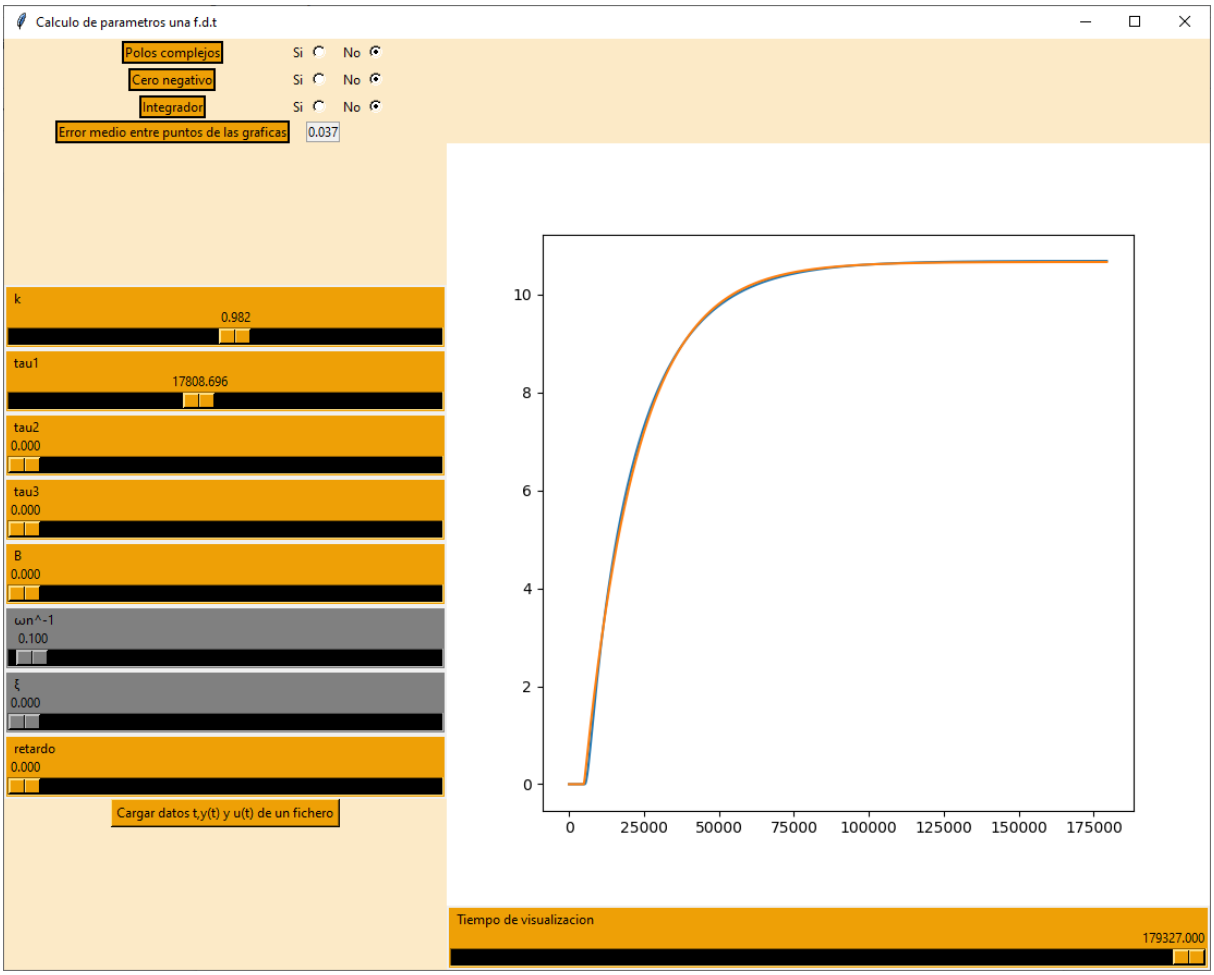


Fuente:Elaboración propia

Se utiliza la herramienta ‘**Obtencion-de-una-f.d.t**’para conseguir la f.d.t del sistema, tal como muestra la imagen 58, se consigue con un error entre puntos del 0.037, y la f.d.t es la siguiente:

$$\frac{0.982}{(1 + 17.8s)}$$

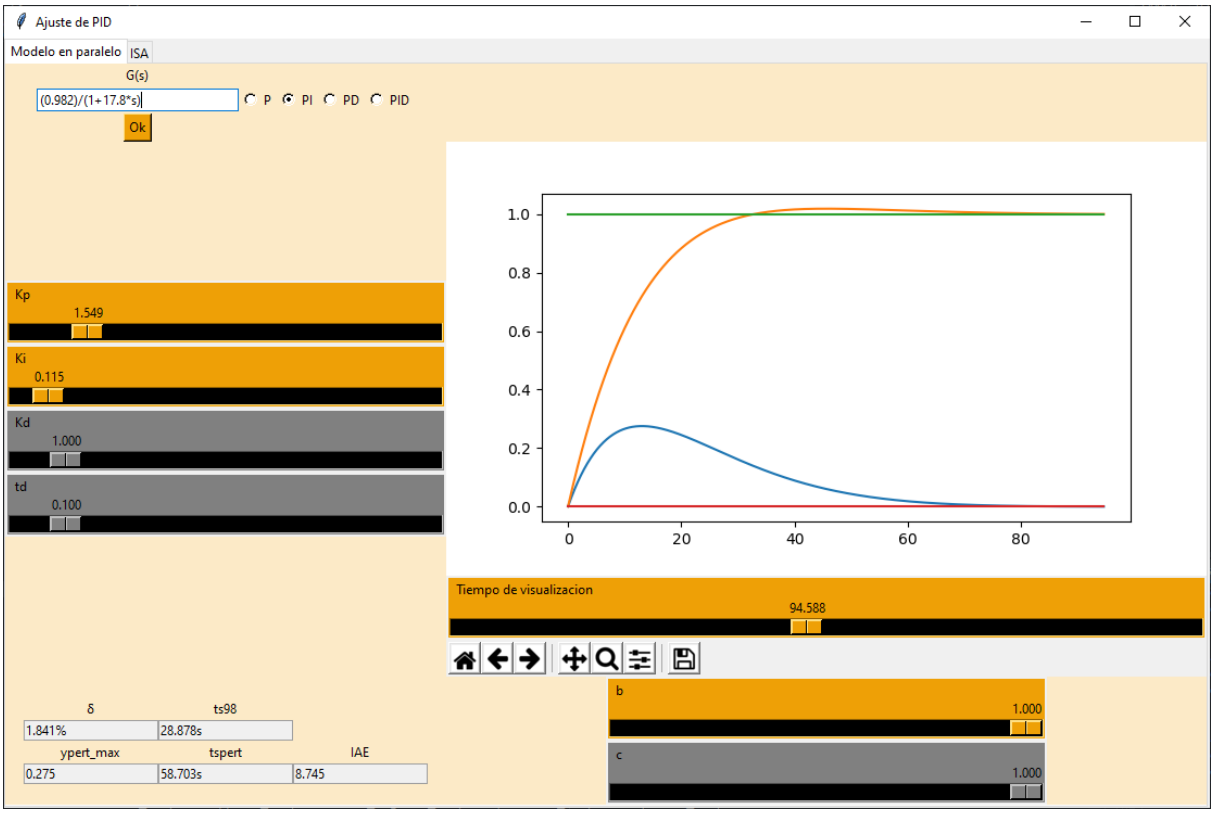
Imagen 58: Encontrando la f.d.t del sistema de CODESYS



Fuente:Elaboración propia

Con esta función de transferencia, se procede a calcular un PID que tenga buenas características, utilizando la herramienta desarrollada '**Respuesta-de-PID**', según muestra la imagen 59.

Imagen 59: Encontrando valores de PID para f.d.t de CODESYS

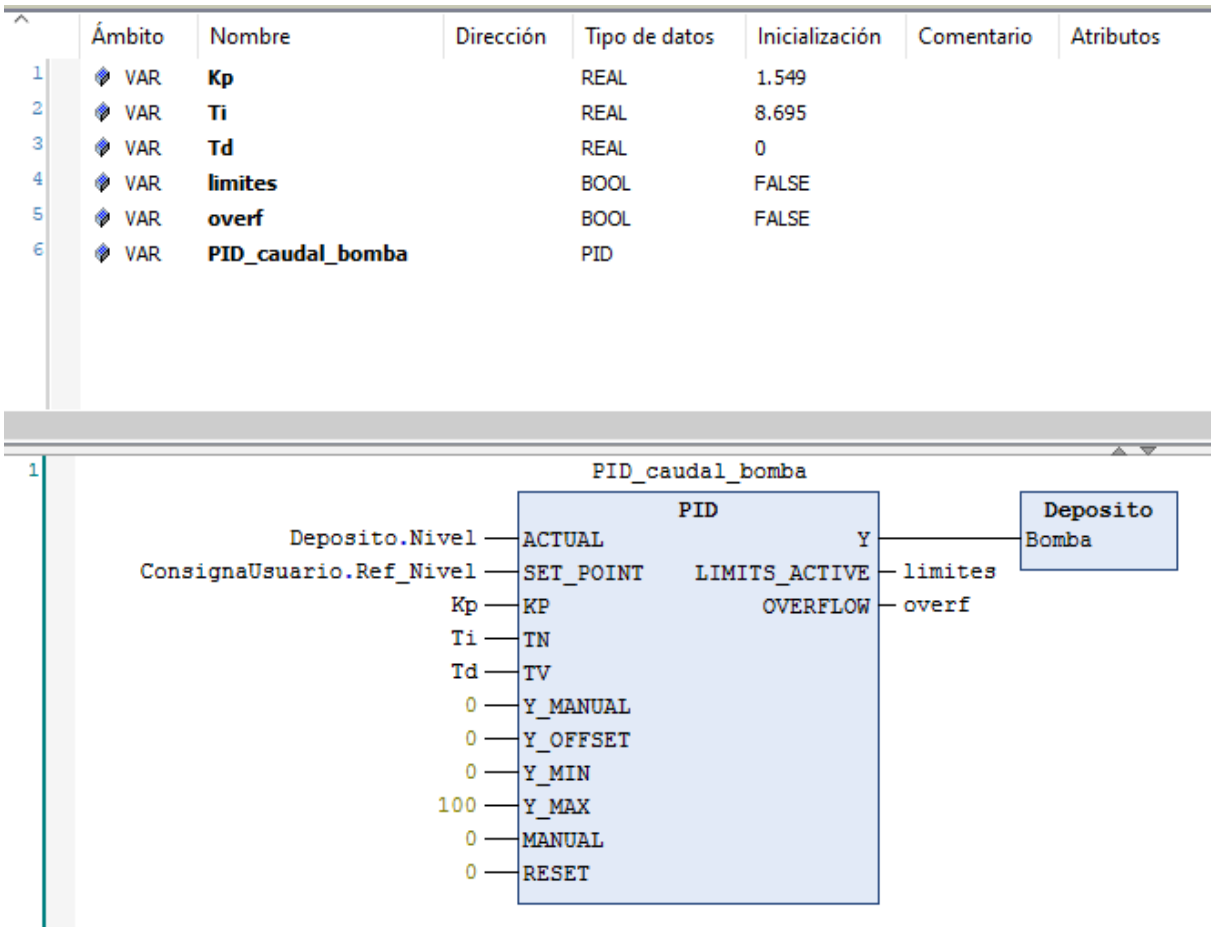


Fuente:Elaboración propia

Siendo el controlador PID, demasiado oscilatorio, se opta por el PI. Encontrado la respuesta más rápida y menos oscilatoria en: Kp=1.549, Ki=0.115 y b=1, con: sobreoscilación del 1.841%, un ts98 de 28.878s, ymax ante perturbación de 0.275, tspert=58.703s y IAE de 8.745. Reducir el factor de ponderación reducía ligeramente la sobreoscilación y aumentaba de manera considerable el tiempo de establecimiento al 98%, por lo que se decidió dejarlo en su valor predeterminado, 1.

Así pues, se procede a insertar los valores obtenidos en un PI programado en CODESYS según la imagen 60.

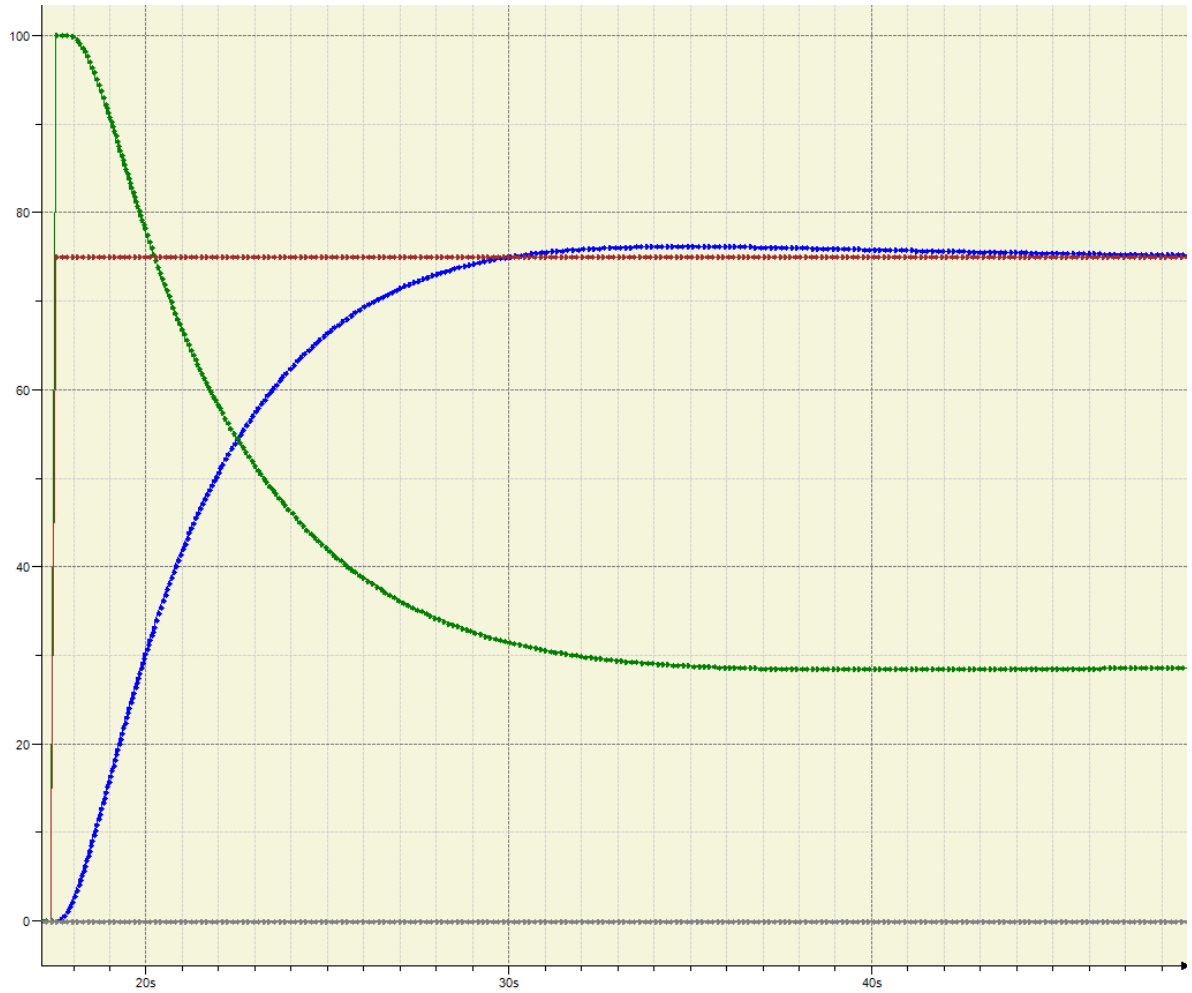
Imagen 60: Colocando valores en un PID de CODESYS



Fuente:Elaboración propia

Cambiando la referencia del nivel de 0% lleno a 75% lleno, se obtiene la siguiente respuesta en CODESYS, según muestra la imagen 61 representado en azul el nivel, en verde la acción de la bomba y en marrón el cambio de referencia.

Imagen 61: Respuesta ante cambio de referencia



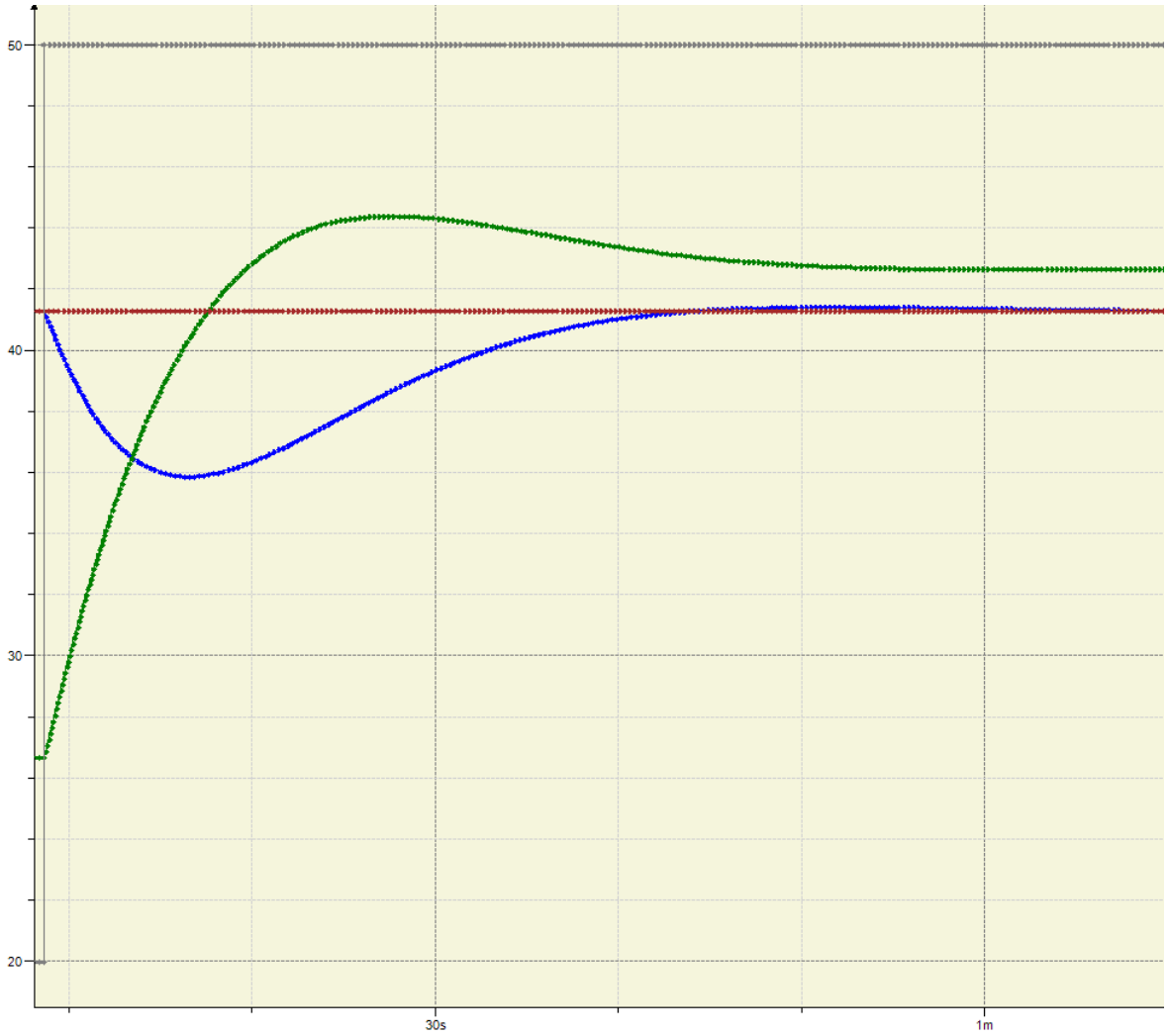
Fuente:Elaboración propia

Comenzando el escalón en el segundo 17.5, y alcanzando el valor de establecimiento al 98% en aproximadamente en el segundo 46, dando un total de ts98=28.5s. Se aprecia que el tiempo de establecimiento al 98% calculado por la herramienta y el simulado por CODESYS, son muy próximos. La sobreoscilación en CODESYS es, aproximadamente: ((76.5/75)-1)*100=2% y en la herramienta del 1.841%.

$$ts98_{herramienta} = 28.878s \ \delta_{herramienta} = 1.841\%$$
$$ts98_{CODESYS} \simeq 28.5s \qquad \delta_{CODESYS} \simeq 2\%$$

Ahora, se realiza un ensayo de perturbación, para ello se cambia una perturbación del 20% a una del 50%. El valor mínimo que alcanza el nivel se corresponde con 35.75, siendo el de referencia 45, tal y como se muestra en la imagen 62.

Imagen 62: Respuesta ante perturbación

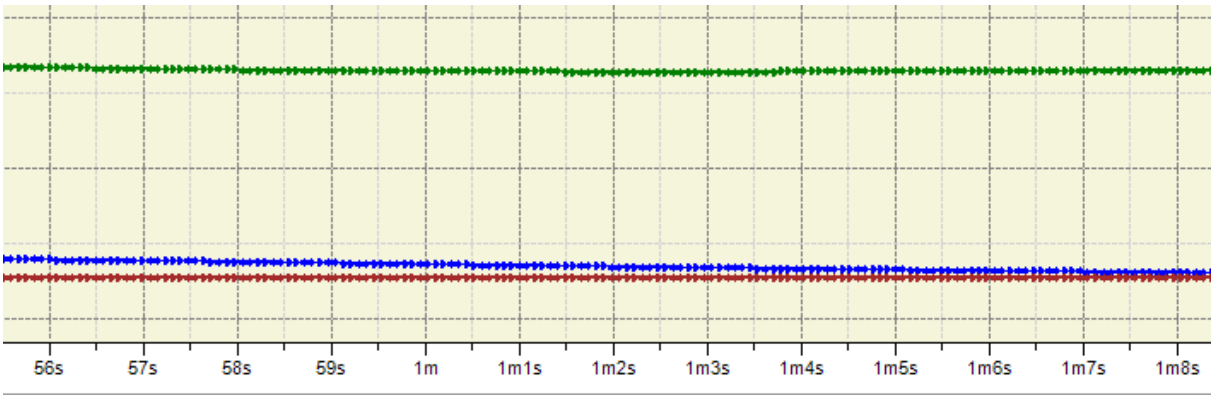


Fuente:Elaboración propia

Comenzando el escalón en 8.5s y terminando aproximadamente en el segundo 66.5 como se muestra en la imagen 63, se tiene que tspert=66.5-8.5=58s, para la herramienta 58.703s.

En azul el nivel, en verde la acción de la bomba, en marrón la referencia y en gris el cambio de perturbación.

Imagen 63: Zoom de respuesta ante perturbación



Fuente:Elaboración propia

$$tspert_{herramienta} = 58.703s \; ymaxpert_{herramienta} = 0.275$$

$$tspert_{CODESYS} \simeq 58s \qquad ymaxpert_{CODESYS} \simeq \frac{45}{35.75} - \frac{45}{45} = 0.258$$

9.5.1 Conclusión

Se puede decir que las herramientas funcionan de manera correcta dados los resultados obtenidos, ya que CODESYS se considera un software industrial, profesional y fiable reconocido en múltiples países.

10. Conclusión general

Se considera que se han cumplido los requisitos de diseño ya que se interpretan los datos del usuario y los datos experimentales de manera correcta, además, los botones y deslizaderas se considera que están bien diferenciados y representados. La fluidez de las gráficas es estable y correcta. Los datos se leen y se calculan de manera veraz y ordenada. La homogeneidad y coherencia en el código también se considera apta ya que todas las aplicaciones poseen una estructura de operaciones y actuación muy similar.

Las aplicaciones desarrolladas cumplen sus funciones de manera veraz y son capaces de cumplir los distintos objetivos propuestos en el punto 1 de la memoria, si bien poseen algunas limitaciones como las descritas en el alcance (punto 3 de la memoria), estas pueden ser eventualmente complementadas gracias a la compartición online de código mediante la página web github.com, las aportaciones son libres y por tanto, los motivos específicos de cada usuario que colabore pueden ser muy diversos. La popularidad de una aplicación en github.com

también aumenta el número de usuarios que deciden apoyarla de manera desinteresada.

La utilización de estas aplicaciones para procesos didácticos e industriales es loable y totalmente libre. Esto, como se comenta en el objetivo, puede ayudar de manera sustancial tanto a particulares como a empresas, siendo este proyecto capaz de ser aplicado en múltiples situaciones distintas, por múltiples personas.

11. Resumen del presupuesto

El resumen del presupuesto queda recogido en la tabla 2:

Tabla 2: Resumen presupuesto

Concepto	Importe (€)
Recursos humanos	3300
Recursos materiales	30.6
PEM	3330.6
Gastos generales y recargos fiscales	666.12
Subtotal	3996.72
Beneficio industrial	0
PEC	3996.72
Honorarios de proyectista	279.77
Presupuesto total sin impuestos	4276.5
IVA	898.06
Presupuesto total con impuestos	5174.55

12. Viabilidad económica

Aquí se estudiará la viabilidad económica de las aplicaciones en comparación con otras herramientas de características similares que no son gratuitas.

12.1 Parámetros económicos

Aqui se describen los parámetros monetarios simples.

-Inversión inicial

Aquí se expresan todos los costes necesarios para el inicio de la elaboración del proyecto, cuyo desglose de propiedades está descrito en el presupuesto, así pues, la inversión inicial asciende al importe de: 5174.55€

-Gastos totales, beneficio bruto y beneficio neto:

Se va a comparar el coste entre comprar anualmente una licencia de MATLAB, programa capaz de realizar utilidades como las aquí desarrolladas, con el coste anual que supondría este proyecto. Siendo el coste de renovar la licencia de matlab para un solo usuario 800€ y siendo el coste de las aplicaciones desarrolladas 0€.

$Beneficio_{bruto} = 800 - 0$=800€/año

Ya que el beneficio bruto es únicamente una reducción de costes, el beneficio bruto es equivalente al neto, por tanto:

$Beneficio_{bruto} = Beneficio_{neto} = 800$€/año

-Flujo de caja:

Respecto al flujo de caja, no se requiere ningún tipo de mantenimiento para el programa, lo que hace equivalente al $Beneficio_{neto}$ anual y al flujo de caja anual.

$Beneficio_{neto} = FC = 800$€/año

Asumiendo un valor de la tasa de inflación (IPC) del 1.4%, el valor anual descrito para múltiples años es el representado en la tabla 3.

Tabla 3: Resumen anual a 5 años de los beneficios y el flujo de caja

Año	Amortización	Beneficio bruto	Beneficio neto	Flujo de caja
1	0	800	800	800

2	0	811.2	811.2	811.2
3	0	822.4	822.4	822.4
4	0	833.6	833.6	833.6
5	0	844.8	844.8	844.8
6	0	856	856	856
7	0	867.2	867.2	867.2

12.2 Parámetros de rentabilidad

En este apartado se van a describir las distintas variables típicas relacionadas con la rentabilidad de proyecto, es decir, el VAN, el TIR y el PR. Se va a utilizar un interés nominal del 2.1%, calculando la tasa de interés real como:

$$i_r = \frac{i_n}{IPC} = \frac{0.021}{0.014} = 1.5$$

-Valor actual neto (VAN)

Aquí se va a reailzar el cálculo del VAN aplicando la ecuación:

$$VAN = -I_o + \sum_{n=1}^{N} \frac{FC_i}{(1+i_r)^n}$$

Donde FC_i es el flujo de caja descrito anteriormente, y I_o es la inversión inicial.

$$VAN = -5174.55 + \frac{800}{(1+0.015)^1} + \frac{811.2}{(1+0.015)^2} + \frac{822.4}{(1+0.015)^3} + \frac{833.6}{(1+0.015)^4} + \frac{844.8}{(1+0.015)^5} + \frac{856}{(1+0.015)^6} + \frac{867.2}{(1+0.015)^7}$$

Calculando el VAN a los 7 años previamente descritos, se obtiene: **321.31€**

-Tasa interna de retorno (TIR)

Este parámetro mide el tipo de interés para el cual el VAN se hace cero ante el horizonte de años planteados, ese horizonte es de 5 para este caso. La ecuación que lo describe es la siguiente:

$$TIR = i_r|_{VAN=0} = i_r| - I_o + \sum_{n=1}^{N} \frac{FC_i}{(1+i_r)^n}$$

El TIR del proyecto es de 3.05%.

-Período de retorno (PR)

También conocido como payback, es el tiempo en el que se recupera la inversión del proyecto, se calcula de la siguiente manera:

$$PR = \frac{I_o \cdot N}{\sum_{n=1}^{N} FC_i}$$

$$PR = \frac{5174.55 \cdot 5}{5495.87} = 6.59\ años$$

Es decir, el período de retorno son 6 años y 7 meses.

12.3 Análisis de los resultados:

El VAN ha dado un resultado positivo, lo que hace rentable la inversión en el horizonte de 7 años planteado.

El valor del TIR, de 3.05% también es un indicador de la rentabilidad del proyecto debido a que supera al interés real calculado del 1.5%.

En cuanto al PR, el período de retorno es de 6.59 años, menos que el horizonte de rentabilidad planteado, con lo que también indica rentabilidad en el proyecto.

En la tabla 4 se muestra un resumen de las medidas calculadas:

Tabla 4: Tabla resumen de la viabilidad económica

VAN	321,31€
TIR	3.05%
PR	6 años y 7 meses

Anexos

Índice de anexos

1.Cálculos

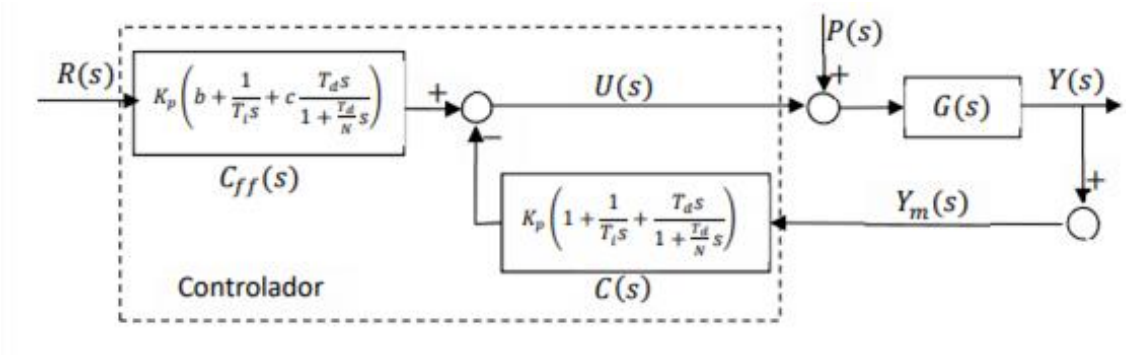
Aquí se describen los cálculos realizados.

1.1 Cálculos del diagrama de bloques

Se han realizado algunos cálculos relacionados con la estructura del diagrama de bloques, para calcular el comportamiento de Y(s) ante R(s) y P(s).

Dado el siguiente diagrama de bloques:

Imagen 64: Diagrama de bloques, cálculos



Fuente:Apuntes de la asignatura EE1023

Aplicando los conceptos desarrollados en el punto Y, se obtiene Y(s):

$$Y(s) = G(s) \cdot ((R(s) \cdot Cff(s) + P(s)) - (C(s) \cdot Y(s) + P(S)))$$

Ahora, desarrollando:

$$\frac{Y(s)}{Cff(s) \cdot R(s) + P(s)} = G(s) \cdot (1 - \frac{C(s) \cdot Y(s)}{R(s) \cdot Cff(s) + P(s)})$$

$$\frac{1}{Cff(s) \cdot R(s) + P(s)} = \frac{G(s)}{Y(s)} - \frac{C(s) \cdot G(s)}{Cff(s) \cdot R(s) + P(s)}$$

$$\frac{1 + C(s) \cdot G(s)}{Cff(s) \cdot R(s) + P(s)} = \frac{G(s)}{Y(s)}$$

Finalmente, se obtiene:

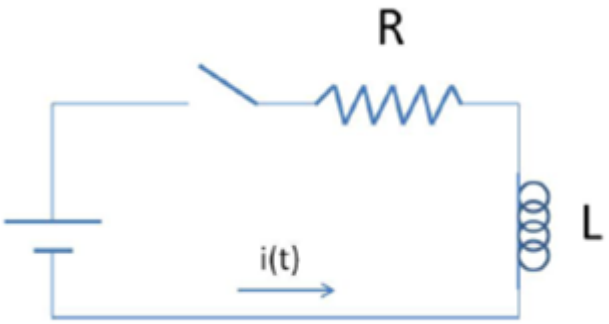
$$Y(s) = Cff$$

1.2 Cálculos del ejemplo 3 del punto 10.1.1 de la memoria

Se pretende conocer la intensidad i(t) que circula por el circuito indicado en la imagen 65, del

cual se conocen los siguientes valores: V=20V R=2Ω L=1H.

Imagen 65: Circuito eléctrico, cálculos



Fuente: Documento de Universidad politécnica de Madrid

Realizando la segunda ley de Kirchoff se tiene:

$$20 = 2 \cdot i(t) + 1 \cdot \frac{di}{dt}$$

Aplicando la transformada de Laplace:

$$\frac{20}{s} = 2 \cdot I(s) + s \cdot I(s)$$

Y despejando la I(s):

$$I(s) = \frac{20}{2s + s^2}$$

2. Código utilizado

En este apartado se explicará y mostrará el código desarrollado en las cuatro aplicaciones. La estructura de funcionamiento del código y funciones principales son similares en todas las aplicaciones, aumentando su homogeneidad, aumenta su comprensión para otros desarrolladores que participen en el proyecto.

Las primeras líneas son equivalentes en las cuatro aplicaciones, son las siguientes:

Importación de las librerías necesarias:

```
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
from matplotlib.figure import Figure
import tkinter as tk
from tkinter import ttk
import control
import numpy as np
```

Inicialización de las variables que dan color a la aplicación y del operador matemático s:

```
naranja_fuerte_codigo_de_colores='#EEA006'
naranja_claro_codigo_de_colores='#FCEAC7'
s=control.tf('s')
```

2.1 Aplicación: 'Representa_la_inversa_de_laplace'

Esta función devuelve un mensaje de error por pantalla según el error ocurrido.

```
def ecuacion_mal_definida(tipo_de_error):
    if tipo_de_error==0:
        tk.messagebox.showinfo(message="Verifique la ecuación.", title="Error")
    if tipo_de_error==1:
        tk.messagebox.showinfo(message=
            "No se puede realizar esta transformacion, compruebe el orden del numerador y denominador.", title="Error")
    t_error=np.array([0,1,2,3,4,5,6,7])
    y_error=np.zeros(8,)
    fig.add_subplot(111).plot(t_error, y_error)
    canvas.draw()
```

Esta función lee los datos de la ventana y llama a la función graficado para dibujar.

```
def leer_datos_de_la_ventana():

    boton_seleccionado_leído=boton_seleccionado_para_leer.get()
    try:
        Y0=eval(input_de_la_ecuación_del_usuario.get())
    except SyntaxError:
        ecuacion_mal_definida(0)
    valor_ajustado_de_la_deslizadera_de_tiempo=np.linspace(0.0,deslizadera_tiempo_objeto.variable.get(), num=1000)
    graficado(Y0,s,boton_seleccionado_leído,valor_ajustado_de_la_deslizadera_de_tiempo)
```

Esta función dibuja los inputs dados por el usuario después de transformarlos de la manera esperada con la función control.impulse_respone

```
def graficado(Y0,s,boton_seleccionado_leído,valor_ajustado_de_la_deslizadera_de_tiempo):

    fig.clf()
    try:
        if boton_seleccionado_leído=='señal_y_t':
            t,y=control.impulse_response(Y0,valor_ajustado_de_la_deslizadera_de_tiempo)
        if boton_seleccionado_leído=='señal_dy_dt':
             t,y=control.impulse_response(Y0*s,valor_ajustado_de_la_deslizadera_de_tiempo)
        if boton_seleccionado_leído=='señal_int_y_dt':
            t,y=control.impulse_response(Y0/s,valor_ajustado_de_la_deslizadera_de_tiempo)
        fig.add_subplot(111).plot(t, y)
        canvas.draw()
    except ValueError:
        ecuacion_mal_definida(1)
```

La siguiente clase crea una clase deslizadera, que se repetirá de manera exacta a lo largo de las 4 aplicaciones. Esta clase coloca las deslizaderas, las hace dinámicas (actualiza el valor de las gráficas cuando se mueven), hace que su valor máximo y mínimo varíe para ajustar la precisión y el horizonte de la escala, además, de por supuesto, crear la deslizadera y guardar su valor actual en una variable. Para crear una deslizadera con estas características solo hay que llamar a la clase y darle los valores iniciales requeridos.

```
class Deslizadera:

    def __init__(self,master,texto,valor_inicial,valor_maximo,valor_minimo):

        self.dini=valor_inicial
        self.dmin=valor_minimo
        self.dmax=valor_maximo
        self.master=master
        self.variable=tk.DoubleVar()
        self.variable.set(valor_inicial)
        self.deslizadera=tk.Scale(master,variable=self.variable,
                                  from_=self.dmin,to=self.dmax,orient=tk.HORIZONTAL,label=texto,resolution=0.001,
                                  length=400,bg=naranja_fuerte_codigo_de_colores,troughcolor='black')
        self.deslizadera.pack(side=tk.BOTTOM,fill=tk.BOTH,expand=True)
        self.deslizadera.bind("<ButtonRelease-1>",self.update)

    def update(self,event):

       if self.dmax==self.dini:
            leer_datos_de_la_ventana()
       else:
            if self.variable.get()==0:

                self.dmax=self.dmax/2
                self.deslizadera.configure(to=self.dmax)

            if self.variable.get() >= (self.dmax-self.dmax/1000):

                self.dmax=self.dmax*2
                self.deslizadera.configure(to=self.dmax)

            if self.dmax < 10:

                self.deslizadera.configure(resolution=(self.dmax/1000))

            leer_datos_de_la_ventana()
```

Por último, aquí se muestra toda la parte gráfica de la aplicación realizada con tkinter.

```
ventana = tk.Tk()
ventana.wm_title("Representacion de la inversa de Laplace")
ventana.config(width=500, height=200,bg=naranja_claro_codigo_de_colores)

frame_canvas=tk.Frame(ventana)
frame_canvas.grid(row=3, column=0)
deslizadera_tiempo_objeto=Deslizadera(frame_canvas,"Tiempo de visualizacion",20,100,0.05)
fig = Figure(figsize=(5, 4), dpi=100)
canvas = FigureCanvasTkAgg(fig, master=frame_canvas)
toolbar = NavigationToolbar2Tk(canvas,frame_canvas)
canvas.get_tk_widget().pack(side=tk.TOP,fill=tk.BOTH,expand=True)

frame_parte_de_laplace=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)
frame_parte_de_laplace.grid(row=0,column=0)

input_de_la_ecuación_del_usuario = ttk.Entry(frame_parte_de_laplace, width=30)
input_de_la_ecuación_del_usuario.insert(0,"(1)/(s**2)")
input_de_la_ecuación_del_usuario.grid(row=1,column=0)

labelSistema=tk.Label(frame_parte_de_laplace,text="Y(s)=L(y(t))",bg=naranja_claro_codigo_de_colores)
labelSistema.grid(row = 0, column =0)

leer_datos_de_la_ventana_boton = tk.Button(frame_parte_de_laplace, text="Ok",command=leer_datos_de_la_ventana,bg=naranja_fuerte_codigo_de_colores)
leer_datos_de_la_ventana_boton.grid(row =1, column = 2, sticky=tk.E)

frameEntradas=tk.Frame(ventana)
frameEntradas.grid(row=1,column=0)

boton_seleccionado_para_leer = tk.StringVar()
boton_seleccionado_para_leer.set(0)

selectorseñal_y_t= tk.Radiobutton(frameEntradas,bg=naranja_claro_codigo_de_colores,
                                  text="y(t)", variable=boton_seleccionado_para_leer,value='señal_y_t',command=leer_datos_de_la_ventana)
selectorseñal_dy_dt= tk.Radiobutton(frameEntradas,bg=naranja_claro_codigo_de_colores,
                                    text="dy/dt", variable=boton_seleccionado_para_leer,value='señal_dy_dt',command=leer_datos_de_la_ventana)
selectorseñal_int_y_dt=  tk.Radiobutton(frameEntradas,bg=naranja_claro_codigo_de_colores,
                                      text="∫ydt", variable=boton_seleccionado_para_leer,value='señal_int_y_dt',command=leer_datos_de_la_ventana)

selectorseñal_y_t.grid(row = 0, column =1)
selectorseñal_dy_dt.grid(row = 0, column =2)
selectorseñal_int_y_dt.grid(row = 0, column = 3)
```

2.2 Aplicación: "Obtencion-de-una-f.d.t"

Después de importar las librerías y definir los colores:

Se inicializan los datos experimentales a 'no usar' y se define la f.d.t en caso de no insertar los datos, la fdt_predeterminada está al principio del código para que sea accesible su modificación.

```
usar_datos_experimentales='no usar'
fdt_predeterminada= (2.670)/((1+s)*s)
```

Función que regula el retardo del sistema:

```
def ajusta_el_retardo_segun_el_input_de_la_ventana(t_orig,y_orig,retardo):

    m=np.array([0])
    t=np.concatenate([m,t_orig+retardo])
    y=np.concatenate([m,y_orig])
    return t,y
```

Parte de la función que se encarga de leer los inputs del usuario y ejecutar el cálculo de la nueva f.d.t según los inputs. Esta parte del código recoge los inputs del usuario, el valor de los botones del integrador son 1 y 2, si el resultado es 1, la f.d.t no usa integrador, si el resultado es 2, la variable integrador pasa a valer s, haciendo que el sistema ahora tenga un integrador. El valor del cero_negativo_seleccionado actúa de manera parecida, valiendo 1 o -1 según el operario del programa crea conveniente.

```
def leer_datos_de_la_ventana():

    global fdt_predeterminada,fdt_estimada

    tau1=tau1_objeto.variable.get()
    tau2=tau2_objeto.variable.get()
    tau3=tau3_objeto.variable.get()
    k=k_objeto.variable.get()
    factor_de_amortiguamiento=factor_de_amortiguamiento_objeto.variable.get()
    frec_natural_elevado_a_menos_1=(frec_natural_objeto.variable.get())
    deslizadera_tiempo=deslizadera_tiempo_objeto.variable.get()
    retardo=retardo_objeto.variable.get()
    b=b_objeto.variable.get()
    polos_complejos_habilitados=polos_comp_sel.get()
    cero_negativo_seleccionado=cero_neg_sel.get()
    integrador=integrador_sel.get()

    if integrador==2: integrador=s
```

Aquí se aprecia que todo el denominador está multiplicado por la variable "integrador", también se puede observar el comportamiento de cero_negativo_seleccionado. Se puede apreciar como la ecuación utilizada cambia en función de si hay o no polos complejos, así como

la activación y desactivación de deslizaderas en tal caso.

Después de lo descrito, se crea una valor para la deslizadera del tiempo ajustado siempre a un vector 1000 valores ya que así es posible graficarlo con el resto de parámetros típicos con un vector de 1000 valores. También llama a la función cálculos.

```
if polos_complejos_habilitados==1:

    frec_natural_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
    factor_de_amortiguamiento_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
    fdt_estimada=(k*(1+b*cero_negativo_seleccionado*s))/(((1+tau1*s)*(1+tau2*s)*(1+tau3*s)*
            (1+((2*factor_de_amortiguamiento*s)*frec_natural_elevado_a_menos_1)+((s**2)*frec_natural_elevado_a_menos_1**2)))*integrador)

else:
    frec_natural_objeto.deslizadera.configure(state='disabled',bg='gray')
    factor_de_amortiguamiento_objeto.deslizadera.configure(state='disabled',bg='gray')
    fdt_estimada=(k*(1+b*cero_negativo_seleccionado*s))/(((1+tau1*s)*(1+tau2*s)*(1+tau3*s))*integrador)


t_deslizadera_ajustada=np.linspace(0.0,deslizadera_tiempo, num=1000)
calculos(t_deslizadera_ajustada,retardo)
```

Primera parte de esta función, aquí se identifica el tipo de datos que el usuario quiere utilizar y se calcula en consecuencia el error, el retardo, y la f.d.t a partir o bien de un escalón unitario o de la u(t) dada por el usuario con los datos experimentales.

```
def calculos(t_deslizadera_ajustada,retardo) :

    if usar_datos_experimentales=='t e y(t)':
        t_experimental_predeterminada=0
        y_experimental_predeterminada=0
        t_predeterminada_sin_retardo,y_predeterminada_sin_retardo=control.step_response(fdt_estimada,T=(t_deslizadera_ajustada))
        t_ajustada_al_retardo,y_ajustada_al_retardo=ajusta_el_retardo_segun_el_input_de_la_ventana(t_deslizadera_ajustada,
                                                                                                   t_predeterminada_sin_retardo
                                                                                                   ,y_predeterminada_sin_retardo,retardo)

        error=np.zeros(len(y_experimental))

        for i in range(len(y_experimental)):
            error[i]=np.absolute(y_ajustada_al_retardo[i]-y_experimental[i])
            if np.around(y_experimental[i],2)==np.around(k_objeto.variable.get(),2):
                np.delete(error,i)

    if usar_datos_experimentales=='t, y(t) y u(t)':

        t_experimental_predeterminada=0
        y_experimental_predeterminada=0
        t_predeterminada_sin_retardo,y_predeterminada_sin_retardo,xout=control.forced_response(fdt_estimada,U=u_experimental,
                                                                                               T=(t_deslizadera_ajustada))
        t_ajustada_al_retardo,y_ajustada_al_retardo=ajusta_el_retardo_segun_el_input_de_la_ventana(t_deslizadera_ajustada,
                                                                                                   t_predeterminada_sin_retardo
                                                                                                   ,y_predeterminada_sin_retardo,retardo)


        error=np.zeros(len(y_experimental))

        for i in range(len(y_experimental)):
            error[i]=np.absolute(y_ajustada_al_retardo[i]-y_experimental[i])

            if np.around(y_experimental[i],2)==np.around(k_objeto.variable.get(),2):
                np.delete(error,i)
```

Última parte del código donde se describe la última de las tres opciones con su cálculo del error, también se llama a la función graficado encargada de dibujar las gráficas.

```
if usar_datos_experimentales=='no usar':

    t_predeterminada_sin_retardo,y_predeterminada_sin_retardo=control.step_response(fdt_estimada,T=(t_deslizadera_ajustada))
    t_ajustada_al_retardo,y_ajustada_al_retardo=ajusta_el_retardo_segun_el_input_de_la_ventana(t_deslizadera_ajustada,
                                                                                               t_predeterminada_sin_retardo,
                                                                                               y_predeterminada_sin_retardo,retardo)
    t_experimental_predeterminada,y_experimental_predeterminada=control.step_response(fdt_predeterminada,T=(t_deslizadera_ajustada))
    error=np.zeros(len(y_predeterminada_sin_retardo))
    for i in range(len(y_predeterminada_sin_retardo)):
        error[i]=np.absolute(y_ajustada_al_retardo[i]-y_experimental_predeterminada[i])
        if np.around(y_experimental_predeterminada[i],2)==np.around(k_objeto.variable.get(),2):
            np.delete(error,i)

error_final=np.mean(error)
error_por_pantalla.configure(text=round(error_final,3))

graficado(t_experimental_predeterminada,y_experimental_predeterminada,t_ajustada_al_retardo,y_ajustada_al_retardo)
```

La función graficado se encarga de dibujar ambas gráficas según la elección del usuario:

```
def graficado(t_experimental_predeterminada,y_experimental_predeterminada,t_ajustada_al_retardo,y_ajustada_al_retardo):

    fig.clf()

    if usar_datos_experimentales=='no usar':
        fig.add_subplot(111).plot(t_experimental_predeterminada,y_experimental_predeterminada)
        fig.add_subplot(111).plot(t_ajustada_al_retardo, y_ajustada_al_retardo)

    if usar_datos_experimentales=='t, y(t) y u(t)' or usar_datos_experimentales=='t e y(t)':

        fig.add_subplot(111).plot(t_experimental, y_experimental)
        fig.add_subplot(111).plot(t_ajustada_al_retardo, y_ajustada_al_retardo)
    canvas.draw()
```

La siguiente función es la encargada de interpretar los datos experimentales del usuario, es la denominada recibir_datos_fichero y es llamada cuando se presiona el botón. Aqui se muestra la declaración de variables globales, la obtención de los datos, y el error cuando no se detecta una u_experimental en la tercera fila de la aplicación. Posteriormente, se inicializa la t_experimental restando su primer valor a todos los elementos del vector. Luego se transforma la y_experimental y la t_experimental a vectores de 1000 valores con los que poder trabajar de manera cómoda. También se inicializa un vector necesario en las siguientes líneas.

```
def recibir_datos_fichero():

    global usar_datos_experimentales,t_experimental,y_experimental,u_experimental

    usar_datos_experimentales='t, y(t) y u(t)'
    tyu_experimental_a_separar=np.loadtxt('datos.txt')
    t_experimental=tyu_experimental_a_separar[:,0]
    y_experimental=tyu_experimental_a_separar[:,1]
    try: u_experimental=np.array(tyu_experimental_a_separar[:,2])

    except IndexError:
        u_experimental=[0]
        usar_datos_experimentales='t e y(t)'
        tk.messagebox.showinfo(message="No se han insertado datos de u(t), la gráfica naranja está representada ante entrada escalón unitaria"
                               , title="Advertencia")
    t_experimental=np.subtract(t_experimental,np.min(t_experimental))
    tfin=t_experimental[-1]
    vector_tiempos=np.linspace(0,tfin,num=1000)
    y_experimental=np.interp(vector_tiempos,t_experimental,y_experimental)
    t_experimental=vector_tiempos.copy()
    u_experimental_corregida=[0]
```

Ahora se requiere ajustar el valor de u_experimental a un vector de 1000 datos, además, en el primer if se verifica que u_experimental no es un vector de ceros, en cuyo caso se aplicaría la opción de y(t) y t, prescindiendo de u(t). Si u_experimental posee menos de 1000 valores, se rellena el vector con el último valor de u_experimental, supuesto como el valor de establecimiento. En caso de que el vector tenga más de 1000 valores, se inicializan unas variables y se procede a realizar una serie de operaciones:

-Se calcula cuántas veces es el tamaño de u_experimental más grande que mil y se almacena en a.

-En un bucle, cada ‘a’ veces se calcula la media de ‘a’ valores de u_experimental y se almacenan en un nuevo vector llamado “u_experimental_corregida”.

Una vez realizado esto, se le da la vuelta al vector ya que debido al método de inserción de datos estos habían quedado del otro lado. Además, ya que probablemente sobrarán o faltarán valores debido al cálculo de a, hay tres alternativas:

-Que el tamaño del vector sea exactamente 1000, en cuyo caso no se realizan mas operaciones.

-Que el tamaño del vector sea menor de 1000, en cuyo caso se procede a rellenar con el último valor como se ha explicado previamente

-Que el tamaño del vector sea mayor de 1000, en cuyo caso se eliminan los últimos valores ya que serán los menos relevantes ya que se pretende querer alcanzar el valor de establecimiento, por lo que los últimos valores suelen ser repetidamente idénticos.

Por último, se iguala la u_experimental a la u _experimental_corregida para poderla usar globalmente.

```
if  (len(u_experimental)<1000) and (u_experimental!=[0]):
    valores_necesarios=1000-len(u_experimental)
    valor_establecimiento=u_experimental[-1]
    a=np.ones(valores_necesarios)*valor_establecimiento
    u_experimental=np.concatenate([u_experimental,a])
if len(u_experimental)>1000:
    b=0
    a=round(len(u_experimental)/1000)
    c=np.array([])
    u_experimental_corregida=np.array([])

    for i in range(len(u_experimental)):
        c=np.insert(c,0,u_experimental[i])
        b=b+1
        if a==b:
          u_experimental_corregida=np.insert(u_experimental_corregida,0,np.mean(c))
          b=0
          c=[]
    u_experimental_corregida=u_experimental_corregida[::-1]
    datos_sobrantes=len(u_experimental_corregida)-1000
    if datos_sobrantes<0:
        u_experimental_corregida=np.delete(u_experimental_corregida,[len(u_experimental_corregida)-1])
        valor_establecimiento=(u_experimental[-1])
        a=np.ones(-datos_sobrantes+1)*valor_establecimiento
        u_experimental_corregida=np.concatenate([u_experimental_corregida,a])

    if datos_sobrantes>0:
        for i in range(datos_sobrantes+1):
            u_experimental_corregida=np.delete(u_experimental_corregida,[len(u_experimental_corregida)-i])
    u_experimental=u_experimental_corregida
```

Antes de invocar a la función leer_datos_de_la_ventana, se ajusta la deslizadera del tiempo para que su valor puede alcanzar el de la t_experimental sin tener que arrastrala repetidas veces.

```
deslizadera_tiempo_objeto.dmax=np.max(t_experimental)
deslizadera_tiempo_objeto.deslizadera.configure(to=np.max(t_experimental))
deslizadera_tiempo_objeto.variable.set(np.max(t_experimental))
leer_datos_de_la_ventana()
```

Ahora, solo queda la parte gráfica, como se había mencionado antes, aquí está la clase deslizadera:

```
class Deslizadera:

    def __init__(self,master,texto,valor_inicial,valor_maximo,valor_minimo):

        self.dini=valor_inicial
        self.dmin=valor_minimo
        self.dmax=valor_maximo
        self.master=master
        self.variable=tk.DoubleVar()
        self.variable.set(valor_inicial)
        self.deslizadera=tk.Scale(master,variable=self.variable,
                                  from_=self.dmin,to=self.dmax,orient=tk.HORIZONTAL,label=texto,resolution=0.001,
                                  length=400,bg=naranja_fuerte_codigo_de_colores,troughcolor='black')
        self.deslizadera.pack(side=tk.BOTTOM,fill=tk.BOTH,expand=True)
        self.deslizadera.bind("<ButtonRelease-1>",self.update)

    def update(self,event):

       if self.dmax==self.dini:
            leer_datos_de_la_ventana()
       else:
            if self.variable.get()==0:

                self.dmax=self.dmax/2
                self.deslizadera.configure(to=self.dmax)

            if self.variable.get() >= (self.dmax-self.dmax/1000):

                self.dmax=self.dmax*2
                self.deslizadera.configure(to=self.dmax)

            if self.dmax < 10:

                self.deslizadera.configure(resolution=(self.dmax/1000))

            leer_datos_de_la_ventana()
```

A continuación se representan los elementos gráficos:

```
ventana= tk.Tk()
ventana.wm_title("Calculo de parametros una f.d.t")
ventana.configure(bg=naranja_claro_codigo_de_colores)
frame_canvas=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)
frame_botones=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)
frame_deslizaderas=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)

frame_canvas.grid(row=1, column=1)
frame_botones.grid(row=0, column=0)

frame_deslizaderas.grid(row=1,column=0)

fig = Figure(figsize=(7, 7), dpi=100)
canvas = FigureCanvasTkAgg(fig, master=frame_canvas)
canvas.get_tk_widget().pack(side=tk.TOP,fill=tk.BOTH,expand=True)
polos_comp_sel=tk.IntVar()
polos_comp_sel.set(0)
boton_polos_complejos_no=tk.Radiobutton(master=frame_botones,variable=polos_comp_sel,
                                        value=0,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_polos_complejos_si=tk.Radiobutton(master=frame_botones,variable=polos_comp_sel,
                                        value=1,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
label_complejos=tk.Label(master=frame_botones,text='Polos complejos',bg=naranja_fuerte_codigo_de_colores,relief='solid')
label_complejos.grid(row=0,column=0)
label_complejos_no=tk.Label(master=frame_botones,text='No',bg=naranja_claro_codigo_de_colores)
label_complejos_si=tk.Label(master=frame_botones,text='Si',bg=naranja_claro_codigo_de_colores)
label_complejos_no.grid(row=0,column=3)
label_complejos_si.grid(row=0,column=1)
boton_polos_complejos_no.grid(row=0,column=4)
boton_polos_complejos_si.grid(row=0,column=2)
```

```
cero_neg_sel=tk.IntVar()
cero_neg_sel.set(1)
boton_cero_negativo_no=tk.Radiobutton(master=frame_botones,variable=cero_neg_sel,value=1
                                      ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_cero_negativo_si=tk.Radiobutton(master=frame_botones,variable=cero_neg_sel,value=-1,
                                      command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
label_ceros=tk.Label(master=frame_botones,text='Cero negativo',bg=naranja_fuerte_codigo_de_colores,relief='solid')
label_ceros.grid(row=1,column=0)
label_cero_negativo_no=tk.Label(master=frame_botones,text='No',bg=naranja_claro_codigo_de_colores)
label_cero_negativo_si=tk.Label(master=frame_botones,text='Si',bg=naranja_claro_codigo_de_colores)
label_cero_negativo_no.grid(row=1,column=3)
label_cero_negativo_si.grid(row=1,column=1)
boton_cero_negativo_no.grid(row=1,column=4)
boton_cero_negativo_si.grid(row=1,column=2)

integrador_sel=tk.IntVar()
integrador_sel.set(1)
boton_integrador_sel_no=tk.Radiobutton(master=frame_botones,variable=integrador_sel,value=1
                                       ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_integrador_sel_si=tk.Radiobutton(master=frame_botones,variable=integrador_sel,value=2
                                       ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
label_integrador=tk.Label(master=frame_botones,text='Integrador',bg=naranja_fuerte_codigo_de_colores,relief='solid')
label_integrador.grid(row=2,column=0)
label_integrador_no=tk.Label(master=frame_botones,text='No',bg=naranja_claro_codigo_de_colores)
label_integrador_si=tk.Label(master=frame_botones,text='Si',bg=naranja_claro_codigo_de_colores)
label_integrador_no.grid(row=2,column=3)
label_integrador_si.grid(row=2,column=1)
boton_integrador_sel_no.grid(row=2,column=4)
boton_integrador_sel_si.grid(row=2,column=2)

error_label=tk.Label(master=frame_botones,bg=naranja_fuerte_codigo_de_colores,text='Error medio entre puntos de las graficas',relief='solid')
error_por_pantalla=ttk.Label(master=frame_botones,text='0',relief='solid')
error_label.grid(row=3,column=0)
error_por_pantalla.grid(row=3,column=2)
```

Aquí queda representado el resto:

```
boton_datos_fichero = tk.Button(frame_deslizaderas, text="Cargar datos t,y(t) y u(t) de un fichero"
                                ,command=recibir_datos_fichero ,bg=naranja_fuerte_codigo_de_colores)
boton_datos_fichero.pack(side=tk.BOTTOM)

deslizadera_tiempo_objeto=Deslizadera(frame_canvas,"Tiempo de visualizacion",40,100,0)
retardo_objeto=Deslizadera(frame_deslizaderas,"retardo",0,5,0)
factor_de_amortiguamiento_objeto=Deslizadera(frame_deslizaderas,"ξ",0,5,0)
frec_natural_objeto=Deslizadera(frame_deslizaderas,"ωn^-1",0.1,5,0.0001)
b_objeto=Deslizadera(frame_deslizaderas,"B",0,5,0)
tau3_objeto=Deslizadera(frame_deslizaderas,"tau3",0,10,0)
tau2_objeto=Deslizadera(frame_deslizaderas,"tau2",0,10,0)
tau1_objeto=Deslizadera(frame_deslizaderas,"tau1",1,10,0)
k_objeto=Deslizadera(frame_deslizaderas,"k",1,30,0)


leer_datos_de_la_ventana()

ventana.mainloop()
```

2.3 Aplicación: "F.d.t-ante-diferentes-entradas"

Después de importar las librerías y definir los colores:

De nuevo, la función para definir el retardo:

```
def ajusta_el_retardo_segun_el_input_de_la_ventana(t_orig,y_orig,retardo):

    m=np.array([0])
    t=np.concatenate([m,t_orig+retardo])
    y=np.concatenate([m,y_orig])
    return t,y
```

La función que avisa de un error de sintaxis:

```
def ecuacion_mal_definida(tipo_de_error):

    if tipo_de_error==0:
        tk.messagebox.showinfo(message="Verifique la ecuación.", title="Error")

    t_error=np.array([0,1,2,3,4,5,6,7])
    y_error=np.zeros(8,)
    fig.add_subplot(111).plot(t_error, y_error)

    canvas.draw()
```

La función que lee los datos de la ventana:

```
def leer_datos_de_la_ventana():

    boton_pulsado=variable_de_los_radio_button.get()
    try:
        G0=eval(input_del_fdt_en_la_ventana.get())
    except SyntaxError:
        ecuacion_mal_definida(0)
    tamaño_impulso=eval(input_impulso_en_la_ventana.get())
    tamaño_rampa=eval(input_rampa_en_la_ventana.get())
    tamaño_escalon=eval(input_escalon_en_la_ventana.get())
    valor_ajustado_de_la_deslizadera_del_tiempo=np.linspace(0.0,deslizadera_tiempo_objeto.variable.get(), num=1000)

    graficado(valor_ajustado_de_la_deslizadera_del_tiempo,tamaño_impulso,tamaño_escalon,tamaño_rampa,boton_pulsado,G0)
```

La función encargada de dibujar las gráficas:

```
def graficado(valor_ajustado_de_la_deslizadera_del_tiempo,tamaño_impulso,tamaño_escalon,tamaño_rampa,boton_pulsado,G0):

    fig.clf()

    retardo=eval(input_del_retardo_en_la_ventana.get())

    if boton_pulsado=='Impulso':
        t,y=control.impulse_response(G0*tamaño_impulso,T=(valor_ajustado_de_la_deslizadera_del_tiempo))
    if boton_pulsado=='Escalon':
        t,y=control.step_response(G0*tamaño_escalon,T=(valor_ajustado_de_la_deslizadera_del_tiempo))
    if boton_pulsado=='Rampa':
        t,y=control.step_response(G0*tamaño_rampa/s,T=(valor_ajustado_de_la_deslizadera_del_tiempo))
    if boton_pulsado=='Arbitraria':
        t,y,xout=control.forced_response(G0,T=(t_experimental),U=(u_experimental))

    t_a_dibujar,y_a_dibujar=ajusta_el_retardo_segun_el_input_de_la_ventana(t,y,retardo)

    fig.add_subplot(111).plot(t_a_dibujar, y_a_dibujar)
    canvas.draw()
```

Aquí se describen dos funciones, una para los datos insertados manualmente y otra para un fichero de datos; las operaciones realizadas ya han sido comentadas anteriormente.

```
def recibir_datos_fichero():

    global t_experimental,u_experimental

    t_junto_con_u_misma_matriz=np.loadtxt('datos.txt')
    t_experimental=t_junto_con_u_misma_matriz[:,0]
    u_experimental=t_junto_con_u_misma_matriz[:,1]
    leer_datos_de_la_ventana()

def recibir_ut_escrita_manualmente():

    global t_experimental,u_experimental

    u_experimental=eval(input_del_usuario_Ut.get())
    t_experimental=eval(input_del_usuario_tiempo.get())
    t_experimental=np.array(t_experimental)
    u_experimental=np.array(u_experimental)
    tfin=t_experimental[-1]
    vector_tiempos=np.linspace(0,tfin,num=1000)
    u_experimental=np.interp(vector_tiempos,t_experimental,u_experimental)
    t_experimental=vector_tiempos.copy()

    leer_datos_de_la_ventana()
```

De nuevo, la parte gráfica empieza con la deslizadera:

```
class Deslizadera:

    def __init__(self,master,texto,valor_inicial,valor_maximo,valor_minimo):

        self.dini=valor_inicial
        self.dmin=valor_minimo
        self.dmax=valor_maximo
        self.master=master
        self.variable=tk.DoubleVar()
        self.variable.set(valor_inicial)
        self.deslizadera=tk.Scale(master,variable=self.variable,
                                  from_=self.dmin,to=self.dmax,orient=tk.HORIZONTAL,label=texto,resolution=0.001,
                                  length=400,bg=naranja_fuerte_codigo_de_colores,troughcolor='black')
        self.deslizadera.pack(side=tk.BOTTOM,fill=tk.BOTH,expand=True)
        self.deslizadera.bind("<ButtonRelease-1>",self.update)

    def update(self,event):

       if self.dmax==self.dini:
            leer_datos_de_la_ventana()
       else:
            if self.variable.get()==0:

                self.dmax=self.dmax/2
                self.deslizadera.configure(to=self.dmax)

            if self.variable.get() >= (self.dmax-self.dmax/1000):

                self.dmax=self.dmax*2
                self.deslizadera.configure(to=self.dmax)

            if self.dmax < 10:

                self.deslizadera.configure(resolution=(self.dmax/1000))

            leer_datos_de_la_ventana()
```

Posteriormente, se describen los elementos gráficos:

```
ventana = tk.Tk()
ventana.wm_title("Comportamiento de una f.d.t ante diferentes entradas")
ventana.config(width=500, height=200,bg=naranja_claro_codigo_de_colores)
frame_canvas=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)
frame_canvas.grid(row=3, column=0)
fig = Figure(figsize=(5, 4), dpi=100)
canvas = FigureCanvasTkAgg(fig, master=frame_canvas)
toolbar = NavigationToolbar2Tk(canvas,frame_canvas)
canvas.get_tk_widget().pack(side=tk.TOP,fill=tk.BOTH,expand=True)
toolbar.update()
deslizadera_tiempo_objeto=Deslizadera(frame_canvas,"Tiempo de visualizacion",20,100,0.05)
frame_fdt_y_retardo=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)
frame_fdt_y_retardo.grid(row=0,column=0)
input_del_fdt_en_la_ventana = ttk.Entry(frame_fdt_y_retardo, width=30)
input_del_fdt_en_la_ventana.insert(0,"(1)/(1+2*s+3*s**2)")
input_del_fdt_en_la_ventana.grid(row=1,column=0)
label_fdt=tk.Label(frame_fdt_y_retardo,text="F.d.t",bg=naranja_claro_codigo_de_colores)
label_fdt.grid(row = 0, column =0)
input_del_retardo_en_la_ventana=ttk.Entry(frame_fdt_y_retardo,width=10)
input_del_retardo_en_la_ventana.grid(row=1,column=1)
input_del_retardo_en_la_ventana.insert(0,"0")
label_retardo=tk.Label(frame_fdt_y_retardo,text="Retardo",bg=naranja_claro_codigo_de_colores)
label_retardo.grid(row = 0, column =1)
boton_de_ok = tk.Button(frame_fdt_y_retardo, text="Ok",command=leer_datos_de_la_ventana,bg=naranja_fuerte_codigo_de_colores)
boton_de_ok.grid(row =1, column = 2, sticky=tk.E)
frame_de_Ut=tk.Frame(ventana,bg=naranja_claro_codigo_de_colores)
frame_de_Ut.grid(row=1,column=0)
boton_datos_fichero = tk.Button(frame_de_Ut, text="Cargar datos u(t) y t de un fichero",command=recibir_datos_fichero
                                ,bg=naranja_fuerte_codigo_de_colores)
boton_datos_fichero.grid(row=6, column=0)
input_impulso_en_la_ventana = ttk.Entry(frame_de_Ut,width=10)
input_rampa_en_la_ventana= ttk.Entry(frame_de_Ut,width=10)
input_escalon_en_la_ventana = ttk.Entry(frame_de_Ut,width=10)
input_impulso_en_la_ventana.insert(0,"1")
input_rampa_en_la_ventana.insert(0,"1")
input_escalon_en_la_ventana.insert(0,"1")
variable_de_los_radio_button = tk.StringVar()
variable_de_los_radio_button.set(0)
radio_button_impulso= tk.Radiobutton(frame_de_Ut, text="Impulso", variable=variable_de_los_radio_button,value='Impulso'
                                     ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
radio_button_escalon= tk.Radiobutton(frame_de_Ut, text="Escalon", variable=variable_de_los_radio_button,value='Escalon'
                                     ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
radio_button_rampa=  tk.Radiobutton(frame_de_Ut, text="Rampa", variable=variable_de_los_radio_button,value='Rampa'
                                    ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
radio_button_arbitraria=  tk.Radiobutton(frame_de_Ut, text="Arbitraria", variable=variable_de_los_radio_button,value='Arbitraria'
                                         ,command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
```

El resto quedan descritos como:

```
radio_button_arbitraria.grid(row = 0, column =0)
radio_button_impulso.grid(row = 0, column =1)
input_impulso_en_la_ventana.grid(row = 1, column=1)
radio_button_escalon.grid(row = 0, column =2)
input_escalon_en_la_ventana.grid(row = 1, column =2)
radio_button_rampa.grid(row = 0, column = 3)
input_rampa_en_la_ventana.grid(row = 1, column = 3)
input_del_usuario_tiempo = ttk.Entry(frame_de_Ut,width=10)
input_del_usuario_tiempo.grid(row=2,column=0)
input_del_usuario_tiempo_label=tk.Label(frame_de_Ut,text="t",bg=naranja_claro_codigo_de_colores)
input_del_usuario_tiempo_label.grid(row=1,column=0)
input_del_usuario_Ut = ttk.Entry(frame_de_Ut,width=10)
input_del_usuario_Ut_label=tk.Label(frame_de_Ut,text="U(t)",bg=naranja_claro_codigo_de_colores)
input_del_usuario_Ut.grid(row=4,column=0)
input_del_usuario_Ut_label.grid(row=3,column=0)
boton_datos_Ut_t_input_usuario = tk.Button(frame_de_Ut, text="Cargar datos u(t),t escritos",command=recibir_ut_escrita_manualmente
                                           ,bg=naranja_fuerte_codigo_de_colores)
boton_datos_Ut_t_input_usuario.grid(row=5, column=0)
ventana.mainloop()
```

2.4 Aplicación: "Respuesta-de-PID"

Después de importar, definir el operador matemático y los colores de la aplicación, se define la función para el error de syntaxis:

def ecuacion_mal_definida(tipo_de_error):

```
def ecuacion_mal_definida(tipo_de_error):

    if tipo_de_error==0:
        tk.messagebox.showinfo(message="Verifique la ecuación.", title="Error")

    t_error=np.array([0,1,2,3,4,5,6,7])
    y_error=np.zeros(8,)
    fig.add_subplot(111).plot(t_error, y_error)

    canvas.draw()
```

A continuación se muestra la funcion leer_datos_ventana que esta vez debe leer inputs de dos pestañas distintas, según el modelo seleccionado, también ajusta la deslizadera y llama a la función "Calculo_PID":

```
def leer_datos_de_la_ventana():


    if titulo_pestanya_actual=='Modelo en paralelo':
        Kp=Kp_objeto.variable.get()
        b=b_objeto.variable.get()
        c=c_objeto.variable.get()
        Ki=Ki_objeto.variable.get()
        Kd=Kd_objeto.variable.get()
        td=td_objeto.variable.get()
        variable_eleccion_de_controlador=variable_eleccion_de_controlador_boton.get()
        valor_de_la_deslizadera_del_tiempo=deslizadera_tiempo_objeto.variable.get()
        try:
            G0=eval(input_del_Gs_en_la_ventana_paralelo.get())
        except SyntaxError:
            ecuacion_mal_definida(0)

    if titulo_pestanya_actual=='ISA':
        Kp=Kp_objeto2.variable.get()
        b=b_objeto2.variable.get()
        c=c_objeto2.variable.get()
        Ti=Ti_objeto.variable.get()
        Td=Td_objeto.variable.get()
        N=N_objeto.variable.get()
        td=Td/N
        Ki=Kp/Ti
        Kd=Td*Kp
        variable_eleccion_de_controlador=variable_eleccion_de_controlador_boton_pestanya_ISA.get()
        valor_de_la_deslizadera_del_tiempo=deslizadera_tiempo_objeto2.variable.get()
        try:
            G0=eval(input_del_Gs_en_la_ventana_ISA.get())
        except SyntaxError:
            ecuacion_mal_definida(0)

    valor_ajustado_de_la_deslizadera_de_tiempo=np.linspace(0.0,valor_de_la_deslizadera_del_tiempo, num=1000)
    calculo_PID(Kp,Ki,Kd,b,c,td,valor_ajustado_de_la_deslizadera_de_tiempo,variable_eleccion_de_controlador,G0)
```

Esta es la función encargada de diferenciar ambas pestañas:

```
def pestanya_seleccionada(event):

    global titulo_pestanya_actual

    pestanya_actual=event.widget.select()
    titulo_pestanya_actual=event.widget.tab(pestanya_actual,"text")
```

A continuación se representa la función calculo_PID donde dependiendo de la elección de la pestaña y del tipo de PID, se configuran la distintas opciones relacionadas con las variables a utilizar en el controlador y la activación y desactivación de deslizaderas

```
def calculo_PID(Kp,Ki,Kd,b,c,td,valor_ajustado_de_la_deslizadera_de_tiempo,variable_eleccion_de_controlador,G0):

        if variable_eleccion_de_controlador=='P':
            if titulo_pestanya_actual=='Modelo en paralelo':
                Ki_objeto.deslizadera.configure(state='disabled',bg='gray')
                Kd_objeto.deslizadera.configure(state='disabled',bg='gray')
                b_objeto.deslizadera.configure(state='disabled',bg='gray')
                c_objeto.deslizadera.configure(state='disabled',bg='gray')
                td_objeto.deslizadera.configure(state='disabled',bg='gray')

            if titulo_pestanya_actual=='ISA':
                Ti_objeto.deslizadera.configure(state='disabled',bg='gray')
                Td_objeto.deslizadera.configure(state='disabled',bg='gray')
                N_objeto.deslizadera.configure(state='disabled',bg='gray')
                b_objeto2.deslizadera.configure(state='disabled',bg='gray')
                c_objeto2.deslizadera.configure(state='disabled',bg='gray')


            controlador=control.tf([Kp],[1])
            controlador_ff=controlador

        if variable_eleccion_de_controlador=='PI':

            if titulo_pestanya_actual=='Modelo en paralelo':
                Ki_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
                Kd_objeto.deslizadera.configure(state='disabled',bg='gray')
                b_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
                c_objeto.deslizadera.configure(state='disabled',bg='gray')
                td_objeto.deslizadera.configure(state='disabled',bg='gray')

            if titulo_pestanya_actual=='ISA':
                Td_objeto.deslizadera.configure(state='disabled',bg='gray')
                Ti_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
                b_objeto2.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
                c_objeto2.deslizadera.configure(state='disabled',bg='gray')
                N_objeto.deslizadera.configure(state='disabled',bg='gray')

            controlador=control.tf([Kp,Ki],[1,0])
            controlador_ff=control.tf([Kp*b,Ki],[1,0])
```

```
    if variable_eleccion_de_controlador=='PD':

        if titulo_pestanya_actual=='Modelo en paralelo':

            Ki_objeto.deslizadera.configure(state='disabled',bg='gray')
            Kd_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            b_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            c_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            td_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)

        if titulo_pestanya_actual=='ISA':

            Ti_objeto.deslizadera.configure(state='disabled',bg='gray')
            Td_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            b_objeto2.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            c_objeto2.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            N_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)

        controlador_ff=control.tf([Kp*b*(td)+c*Kd,Kp*b],[td,1])
        controlador=control.tf([Kd+Kp*(td),Kp],[td,1])


    if variable_eleccion_de_controlador=='PID':

        if titulo_pestanya_actual=='Modelo en paralelo':
            Ki_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            Kd_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            b_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            c_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            td_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)

        if titulo_pestanya_actual=='ISA':
            Ti_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            Td_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            b_objeto2.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            c_objeto2.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)
            N_objeto.deslizadera.configure(state='normal',bg=naranja_fuerte_codigo_de_colores)

        controlador_ff=control.tf([(Kp*(td)*b+Kd*c),(b*Kp+Ki*(td)),Ki],[td,1,0 ])
        controlador=control.tf([(Kp*(td)+Kd),(Kp+Ki*(td)),Ki],[td,1,0 ])
```

Una vez definidos todos los PID, se procede a definir las ecuaciones que siguen y se calcula su respuesta según perturbación y cambio de referencia, cuyo resultado se almacena en las variables indicadas a continuación.

```
bucle_cerrado_respuesta_ante_cambio_de_referencia=controlador_ff*G0/(1+controlador*G0)
bucle_cerrado_respuesta_ante_perturbacion=G0/(1+controlador*G0)
t,y_respuesta_ante_cambio_de_referencia=control.step_response(bucle_cerrado_respuesta_ante_cambio_de_referencia
                                                              ,valor_ajustado_de_la_deslizadera_de_tiempo)
t,y_respuesta_ante_perturbacion=control.step_response(bucle_cerrado_respuesta_ante_perturbacion
                                                      ,valor_ajustado_de_la_deslizadera_de_tiempo)
```

En la siguiente imagen se describe el cálculo de la sobreoscilación y del ts98, primero se inicializa el ts98 a 0, posteriormente para conseguir el ts98 se buscan valores entre el 98 y el 1.02 por ciento del valor de establecimiento de y_respuesta_ante_cambio_de_referencia, cuando se encuentra uno se considera que ese es el escogido, si la función sale del rango, ese valor deja de serlo. Una vez conseguido el valor en y_respuesta_ante_cambio_de_referencia del ts98, simplemente se iguala la posición en su vector a la posición del vector ts98, ya que poseen la misma dimensión, por lo que ese valor de y corresponde a ese valor de t. La sobreoscilación es simplemente el valor máximo de y_respuesta_ante_cambio_de_referencia dividido por el valor final y multiplicado por 100, y menos 100 al resultado de esa operación para dejar el porcentaje en el que excede al valor final. Si la sobreoscilación es negativa (controladores PD, P) se considera 0.

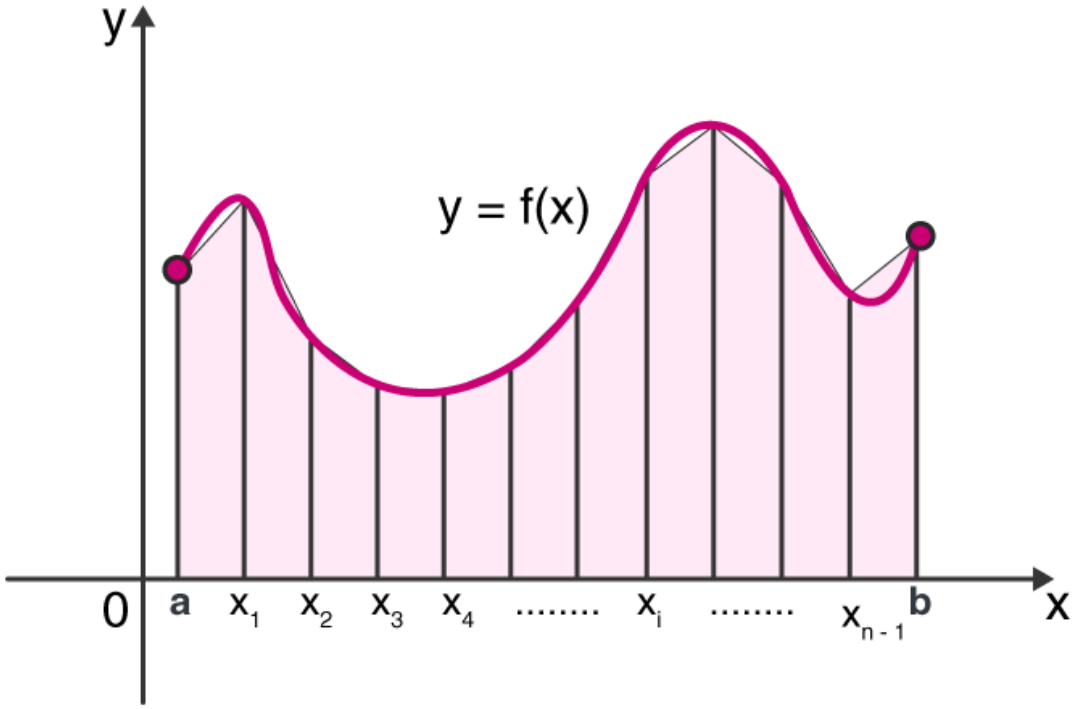
```
ts98_encontrado=0
for i in range(len(y_respuesta_ante_cambio_de_referencia)):

    if (y_respuesta_ante_cambio_de_referencia[i]>=(0.98*y_respuesta_ante_cambio_de_referencia[-1])
        and y_respuesta_ante_cambio_de_referencia[i]<=1.02*(y_respuesta_ante_cambio_de_referencia[-1]) and ts98_encontrado==0):
        y_en_ts98=i
        ts98_encontrado=1

    elif (y_respuesta_ante_cambio_de_referencia[i]<0.98*y_respuesta_ante_cambio_de_referencia[-1]
          or y_respuesta_ante_cambio_de_referencia[i]>1.02*y_respuesta_ante_cambio_de_referencia[-1]):
        ts98_encontrado=0
        y_en_ts98=0
ts98=t[y_en_ts98]
ts98_string=str(round(ts98,3))+"s"
sobreoscilacion=(((np.amax(y_respuesta_ante_cambio_de_referencia))/y_respuesta_ante_cambio_de_referencia[-1])*100)-100
```

Ahora es turno del cálculo de IAE, tspert y ymaxpert, tspert se calcula con un proceso idéntico al de ts98, esta vez sus máximos y mínimos quedan definidos como los valores -0.98 y 0.02, ymaxpert también se calcula de una manera idéntica a la sobreoscilación. Para el IAE, se aproxima su comportamiento mediante la suma trapezoidal de las áreas bajo la curva como muestra la imagen 66.

Imagen 66: Representación de trapecios bajo una curva



Fuente: byjus.com, página web de aprendizaje de matemáticas.

Existe diferenciación entre respuesta positiva y negativa a la hora de calcular el IAE para poder sumar siempre las áreas sin que se resten, ya que el IAE es la integral absoluta del error. Así pues, el código queda como:

```
area_trapecio=0
for i in range(len(y_respuesta_ante_perturbacion)):

      if  y_respuesta_ante_perturbacion[i]>=0:
          base_trapecio=y_respuesta_ante_perturbacion[i-1]
          base_segunda_trapecio=y_respuesta_ante_perturbacion[i]
          altura_trapecio=t[i]-t[i-1]
          area_trapecio=((base_trapecio+base_segunda_trapecio)*altura_trapecio/2)+area_trapecio

      if  y_respuesta_ante_perturbacion[i]<0:
          base_trapecio=-y_respuesta_ante_perturbacion[i-1]
          base_segunda_trapecio=-y_respuesta_ante_perturbacion[i]
          altura_trapecio=t[i]-t[i-1]
          area_trapecio=((base_trapecio+base_segunda_trapecio)*altura_trapecio/2)+area_trapecio

      IAE=area_trapecio


tspert_encontrado=0
for i in range(len(y_respuesta_ante_perturbacion)):
    if (y_respuesta_ante_perturbacion[i]>=-0.98 and y_respuesta_ante_perturbacion[i]<=0.02 and tspert_encontrado==0):
        y_en_tspert=i
        tspert_encontrado=1
    elif (y_respuesta_ante_perturbacion[i]<-0.98 or y_respuesta_ante_perturbacion[i]>0.02):
        tspert_encontrado=0
        y_en_tspert=0
tspert=t[y_en_tspert]
tspert_string=str(round(tspert,3))+"s"
ypertmax=np.amax(y_respuesta_ante_perturbacion)
```

Terminando con esta función, se escriben las variables calculadas en los diversos labels de ambas pestañas, y se llama a la función graficado:

```
if titulo_pestanya_actual=='Modelo en paralelo':
    valor_sobreoscilacion.configure(text=sobreoscilacion_string)
    valor_ts98.configure(text=ts98_string)
    valor_ypert.configure(text=round(ypertmax,3))
    valor_tspert.configure(text=tspert_string)
    valor_IAE.configure(text=round(IAE,3))
if titulo_pestanya_actual=='ISA':
    valor_sobreoscilacion2.configure(text=sobreoscilacion_string)
    valor_ts98_2.configure(text=ts98_string)
    valor_ypert2.configure(text=round(ypertmax,3))
    valor_tspert2.configure(text=tspert_string)
    valor_IAE2.configure(text=round(IAE,3))


graficado(y_respuesta_ante_perturbacion,y_respuesta_ante_cambio_de_referencia,t)
```

Función graficado:

```
def graficado(y_respuesta_ante_perturbacion,y_respuesta_ante_cambio_de_referencia,t):


    if titulo_pestanya_actual=='Modelo en paralelo':
        fig.clf()
        fig.add_subplot(111).plot(t, y_respuesta_ante_perturbacion)
        fig.add_subplot(111).plot(t, y_respuesta_ante_cambio_de_referencia)
        fig.add_subplot(111).plot(t, np.ones(len(y_respuesta_ante_cambio_de_referencia)))
        fig.add_subplot(111).plot(t, np.zeros(len(y_respuesta_ante_cambio_de_referencia)))
        canvas.draw()

    if titulo_pestanya_actual=='ISA':
        fig.clf()
        fig.add_subplot(111).plot(t, y_respuesta_ante_perturbacion)
        fig.add_subplot(111).plot(t, y_respuesta_ante_cambio_de_referencia)
        fig.add_subplot(111).plot(t, np.ones(len(y_respuesta_ante_cambio_de_referencia)))
        canvas2.draw()
```

De nuevo, la clase de la deslizadera:

```
class Deslizadera:

    def __init__(self,master,texto,valor_inicial,valor_maximo,valor_minimo):

        self.dini=valor_inicial
        self.dmin=valor_minimo
        self.dmax=valor_maximo
        self.master=master
        self.variable=tk.DoubleVar()
        self.variable.set(valor_inicial)
        self.deslizadera=tk.Scale(master,variable=self.variable,
                                  from_=self.dmin,to=self.dmax,orient=tk.HORIZONTAL,label=texto,resolution=0.001,
                                  length=400,bg=naranja_fuerte_codigo_de_colores,troughcolor='black')
        self.deslizadera.pack(side=tk.BOTTOM,fill=tk.BOTH,expand=True)
        self.deslizadera.bind("<ButtonRelease-1>",self.update)

    def update(self,event):

       if self.dmax==self.dini:
            leer_datos_de_la_ventana()
       else:
            if self.variable.get()==0:

                self.dmax=self.dmax/2
                self.deslizadera.configure(to=self.dmax)

            if self.variable.get() >= (self.dmax-self.dmax/1000):

                self.dmax=self.dmax*2
                self.deslizadera.configure(to=self.dmax)

            if self.dmax < 10:

                self.deslizadera.configure(resolution=(self.dmax/1000))

            leer_datos_de_la_ventana()
```

Primera parte gráfica de la pestaña “modelo en paralelo”

```
titulo_pestanya_actual='Modelo en paralelo'
ventana = tk.Tk()
ventana.wm_title("Ajuste de PID")
ventana.config(width=500, height=200)

pestanya_maestra=ttk.Notebook(ventana)
pestanya_paralelo=tk.Frame(pestanya_maestra,bg=naranja_claro_codigo_de_colores)
pestanya_ISA=tk.Frame(pestanya_maestra,bg=naranja_claro_codigo_de_colores)
pestanya_maestra.add(pestanya_paralelo, text="Modelo en paralelo")
pestanya_maestra.add(pestanya_ISA, text="ISA")
pestanya_maestra.pack()
pestanya_maestra.bind("<<NotebookTabChanged>>",pestanya_seleccionada)

frame_canvas_paralelo=tk.Frame(pestanya_paralelo,bg=naranja_claro_codigo_de_colores)
frame_canvas_paralelo.grid(row=1, column=1)
fig = Figure(figsize=(7, 4), dpi=100)
canvas= FigureCanvasTkAgg(fig, master=frame_canvas_paralelo)
toolbar = NavigationToolbar2Tk(canvas,frame_canvas_paralelo)
canvas.get_tk_widget().pack(side=tk.TOP,fill=tk.BOTH,expand=True)

toolbar.update()

frame_del_PID=tk.Frame(pestanya_paralelo,bg=naranja_claro_codigo_de_colores)
frame_del_PID.grid(row=1,column=0)

frame_del_PID_bc=tk.Frame(pestanya_paralelo,bg=naranja_claro_codigo_de_colores)
frame_del_PID_bc.grid(row=2,column=1)

c_objeto=Deslizadera(frame_del_PID_bc,"c",1,1,0)
b_objeto=Deslizadera(frame_del_PID_bc,"b",1,1,0)
td_objeto=Deslizadera(frame_del_PID,"td",0.1,1,0)
Kd_objeto=Deslizadera(frame_del_PID,"Kd",1,10,0)
Ki_objeto=Deslizadera(frame_del_PID,"Ki",1,10,0)
Kp_objeto=Deslizadera(frame_del_PID,"Kp",1,10,0)

deslizadera_tiempo_objeto=Deslizadera(frame_canvas_paralelo,"Tiempo de visualizacion",20,100,0.05)

frame_Gs=tk.Frame(pestanya_paralelo,bg=naranja_claro_codigo_de_colores)
frame_Gs.grid(row=0,column=0)

input_del_Gs_en_la_ventana_paralelo = ttk.Entry(frame_Gs, width=30)
input_del_Gs_en_la_ventana_paralelo.insert(0,"(1)/((1+1*s)*(1+1*s))")
input_del_Gs_en_la_ventana_paralelo.grid(row=1,column=0)

label_G0=tk.Label(frame_Gs,text="G(s)",bg=naranja_claro_codigo_de_colores)
label_G0.grid(row=0,column=0)

frame_resultados=tk.Frame(pestanya_paralelo,bg=naranja_claro_codigo_de_colores)
frame_resultados.grid(row=2,column=0)
```

Segunda parte gráfica de la pestaña “modelo en paralelo”

```
valor_sobreoscilacion=ttk.Label(frame_resultados, width=20,relief='solid')
valor_sobreoscilacion_titulo=tk.Label(frame_resultados,bg=naranja_claro_codigo_de_colores, width=1,text="δ")
valor_sobreoscilacion.grid(row=2,column=0)
valor_sobreoscilacion_titulo.grid(row=1,column=0)
valor_ts98=ttk.Label(frame_resultados, width=20,relief='solid')
valor_ts98_titulo=tk.Label(frame_resultados,bg=naranja_claro_codigo_de_colores,width=10,text="ts98")
valor_ts98.grid(row=2,column=1)
valor_ts98_titulo.grid(row=1,column=1)
valor_ypert=ttk.Label(frame_resultados, width=20,relief='solid')
valor_ypert_titulo=tk.Label(frame_resultados,bg=naranja_claro_codigo_de_colores, width=10,text="ypert_max")
valor_ypert.grid(row=4,column=0)
valor_ypert_titulo.grid(row=3,column=0)
valor_tspert=ttk.Label(frame_resultados, width=20,relief='solid')
valor_tspert_titulo=tk.Label(frame_resultados, bg=naranja_claro_codigo_de_colores, width=10,text="tspert")
valor_tspert.grid(row=4,column=1)
valor_tspert_titulo.grid(row=3,column=1)
valor_IAE_titulo=tk.Label(frame_resultados,width=10,text="IAE",bg=naranja_claro_codigo_de_colores)
valor_IAE_titulo.grid(row=3,column=2)
valor_IAE=ttk.Label(frame_resultados, width=20,relief='solid')
valor_IAE.grid(row=4,column=2)

boton_de_ok = tk.Button(frame_Gs, text="Ok",command=leer_datos_de_la_ventana,bg=naranja_fuerte_codigo_de_colores)
boton_de_ok.grid(row =2, column =0)

variable_eleccion_de_controlador_boton=tk.StringVar()
variable_eleccion_de_controlador_boton.set(0)

boton_P=tk.Radiobutton(frame_Gs, text="P", variable=variable_eleccion_de_controlador_boton,  value='P'
                       , command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_P.grid(row=1,column=1)
boton_PI=tk.Radiobutton(frame_Gs, text="PI", variable=variable_eleccion_de_controlador_boton,  value='PI'
                        , command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_PI.grid(row=1,column=2)
boton_PD=tk.Radiobutton(frame_Gs, text="PD", variable=variable_eleccion_de_controlador_boton,  value='PD'
                       , command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_PD.grid(row=1,column=3)
boton_PID=tk.Radiobutton(frame_Gs, text="PID", variable=variable_eleccion_de_controlador_boton,  value='PID'
                         , command=leer_datos_de_la_ventana,bg=naranja_claro_codigo_de_colores)
boton_PID.grid(row=1,column=4)
```

Primera parte gráfica de la pestaña “ISA”, nótese que se ha separado esta parte de la anterior mediante un ‘if 1’

```
if 1:

    frame_canvas_ISA=tk.Frame(pestanya_ISA,bg=naranja_claro_codigo_de_colores)
    frame_canvas_ISA.grid(row=1, column=1)
    fig2 = Figure(figsize=(7, 4), dpi=100)
    canvas2= FigureCanvasTkAgg(fig, master=frame_canvas_ISA)
    toolbar2 = NavigationToolbar2Tk(canvas2,frame_canvas_ISA)
    canvas2.get_tk_widget().pack(side=tk.TOP,fill=tk.BOTH,expand=True)
    toolbar2.update()
    frame_del_PID=tk.Frame(pestanya_ISA,bg=naranja_claro_codigo_de_colores)
    frame_del_PID.grid(row=1,column=0)
    frame_del_PID_bc=tk.Frame(pestanya_ISA,bg=naranja_claro_codigo_de_colores)
    frame_del_PID_bc.grid(row=2,column=1)
    c_objeto2=Deslizadera(frame_del_PID_bc,"c",1,1,0)
    b_objeto2=Deslizadera(frame_del_PID_bc,"b",1,1,0)
    N_objeto=Deslizadera(frame_del_PID,"N",10,20,0)
    Td_objeto=Deslizadera(frame_del_PID,"Td",1,10,0)
    Ti_objeto=Deslizadera(frame_del_PID,"Ti",1,10,0)
    Kp_objeto2=Deslizadera(frame_del_PID,"Kp",1,10,0)
    deslizadera_tiempo_objeto2=Deslizadera(frame_canvas_ISA,"Tiempo de visualizacion",20,100,0.05)
    frame_Gs=tk.Frame(pestanya_ISA,bg=naranja_claro_codigo_de_colores)
    frame_Gs.grid(row=0,column=0)
    input_del_Gs_en_la_ventana_ISA = ttk.Entry(frame_Gs, width=30)
    input_del_Gs_en_la_ventana_ISA.insert(0,"(1)/((1+1*s)*(1+1*s))")
    input_del_Gs_en_la_ventana_ISA.grid(row=1,column=0)
    label_G0=tk.Label(frame_Gs,text="G(s)",bg=naranja_claro_codigo_de_colores)
    label_G0.grid(row=0,column=0)
    frame_resultados=tk.Frame(pestanya_ISA,bg=naranja_claro_codigo_de_colores)
    frame_resultados.grid(row=2,column=0)
    valor_sobreoscilacion2=ttk.Label(frame_resultados, width=20,relief='solid')
    valor_sobreoscilacion2.grid(row=2,column=0)
    valor_sobreoscilacion_titulo=tk.Label(frame_resultados,bg=naranja_claro_codigo_de_colores, width=1,text="δ")
    valor_sobreoscilacion_titulo.grid(row=1,column=0)
    valor_ts98_2=ttk.Label(frame_resultados, width=20,relief='solid')
    valor_ts98_2.grid(row=2,column=1)
    valor_ts98_titulo=tk.Label(frame_resultados,bg=naranja_claro_codigo_de_colores, width=10,text="ts98")
    valor_ts98_titulo.grid(row=1,column=1)
    valor_ypert2=ttk.Label(frame_resultados, width=20,relief='solid')
    valor_ypert_titulo=tk.Label(frame_resultados,bg=naranja_claro_codigo_de_colores, width=10,text="ypert_max")
    valor_ypert2.grid(row=4,column=0)
    valor_ypert_titulo.grid(row=3,column=0)
    valor_tspert2=ttk.Label(frame_resultados, width=20,relief='solid')
    valor_tspert_titulo=tk.Label(frame_resultados, bg=naranja_claro_codigo_de_colores,width=10,text="tspert")
    valor_tspert2.grid(row=4,column=1)
    valor_tspert_titulo.grid(row=3,column=1)

    valor_IAE_titulo=tk.Label(frame_resultados,width=10,text="IAE",bg=naranja_claro_codigo_de_colores)
    valor_IAE_titulo.grid(row=3,column=2)
    valor_IAE2=ttk.Label(frame_resultados,width=20,relief='solid')
    valor_IAE2.grid(row=4,column=2)
```

Segunda y última parte gráfica de modelo ‘ISA’, además del ventana.main.loop().

```
    boton_de_ok = tk.Button(frame_Gs, text="Ok",command=leer_datos_de_la_ventana,bg=naranja_fuerte_codigo_de_colores)
    boton_de_ok.grid(row =2, column = 0)

    variable_eleccion_de_controlador_boton_pestanya_ISA=tk.StringVar()
    variable_eleccion_de_controlador_boton_pestanya_ISA.set(0)

    boton_P=tk.Radiobutton(frame_Gs, text="P", variable=variable_eleccion_de_controlador_boton_pestanya_ISA
                           ,bg=naranja_claro_codigo_de_colores,  value='P', command=leer_datos_de_la_ventana)
    boton_P.grid(row=1,column=1)
    boton_PI=tk.Radiobutton(frame_Gs, text="PI", variable=variable_eleccion_de_controlador_boton_pestanya_ISA
                            , bg=naranja_claro_codigo_de_colores, value='PI', command=leer_datos_de_la_ventana)
    boton_PI.grid(row=1,column=2)
    boton_PD=tk.Radiobutton(frame_Gs, text="PD", variable=variable_eleccion_de_controlador_boton_pestanya_ISA
                           ,bg=naranja_claro_codigo_de_colores,  value='PD', command=leer_datos_de_la_ventana)
    boton_PD.grid(row=1,column=3)
    boton_PID=tk.Radiobutton(frame_Gs, text="PID", variable=variable_eleccion_de_controlador_boton_pestanya_ISA
                            , bg=naranja_claro_codigo_de_colores, value='PID', command=leer_datos_de_la_ventana)
    boton_PID.grid(row=1,column=4)


ventana.mainloop()
```

Pliego de condiciones

Índice de pliego de condiciones

1. Introducción

Para la réplica de este proyecto, en el presente capítulo se recogen las condiciones técnicas, económicas, administrativas y legales pertinentes, estructuradas en:

● Disposiciones de carácter general: donde se describe los temas relacionados con el objetivo y el resto de datos de partida del proyecto, así como el personal adecuado para ejecutarlo y los requisitos técnicos y legales.

● Documentación: contiene el conjunto de cláusulas y restricciones administrativas que detallan el formato de los entregables que integran el documento final del proyecto.

● Especificaciones técnicas: detalla las condiciones en las cuales debe llevarse a cabo el proyecto.

1.2. Disposiciones de carácter general

En este apartado se recogen las normas básicas de ejecución y gestión de los aspectos generales del proyecto.

1.2.1. Objetivo

El objeto del proyecto es el diseño y desarrollo de diversas aplicaciones informáticas capaces de simular el control de un sistema, obtener la ecuación del sistema, y de calcular un PID que permita controlar dicho sistema. Las acciones y requisitos para llevarlo a cabo se rigen por las condiciones indicadas a continuación, así como las órdenes establecidas por el director del proyecto.

1.2.2. Emplazamiento

La redacción y el desarrollo del proyecto se ha llevado a cabo en el área de Ingeniería Eléctrica en el Departamento de Ingeniería de Sistemas Industriales y Diseño y en el itinerario de Control y automatización de sistemas, y está sujeto a las condiciones particulares de dicho departamento.

1.2.3. Personal

El proyecto está integrado por un Director, cuyas funciones principales consisten en coordinar la evolución del mismo, marcar las líneas de desarrollo y evaluar la validez de los resultados

obtenidos. Por otra parte, el proyecto también está integrado por el adjudicatario del proyecto, que es el encargado de la realización del mismo y que se encuentra supeditado a las instrucciones del Director del proyecto.

1.2.4. Responsabilidad

El autor del proyecto deberá disponer de sus propios directorios de trabajo, siendo responsable de su organización, así como del mantenimiento de sus datos y de mantener varias versiones del mismo trabajo en distintos soportes informáticos a fin de evitar posibles pérdidas de datos. La versión definitiva será la verificada por el Director y estará aislada con su respectiva copia de seguridad.
Al tratarse de una versión académica del documento, su autor y su director, así como el departamento correspondiente y la universidad, quedan exentos de cualquier problema o disconformidad que pudiera derivar de su aplicación industrial, independientemente de que se haya realizado atendiendo efectivamente al resto de disposiciones recogidas en el presente pliego de condiciones.

1.3. Documentación

El autor del proyecto entregará una copia del mismo, en versión física o digital, al Departamento de Ingeniería de Sistemas Industriales y Diseño de la Universitat Jaume I, siendo el citado proyecto a partir de ese momento propiedad del departamento y pudiendo ser usado para otros fines. Las partes que deben integrar el proyecto vienen recogidas en la norma UNE 1570012014, y son:

- Memoria: este documento detalla los pasos que se deben seguir para la ejecución del proyecto. Empieza con una breve introducción sobre la necesidad inicial que impulsó la elaboración del proyecto, exponiendo después los objetivos esperados.
- Anejos: se adjunta información que desarrolla, justifica o explica algún apartado de la memoria o cualquier otro documento del proyecto.
- Pliego de condiciones: regula las condiciones entre el promotor del proyecto y las personas que lo van a ejecutar.
- Presupuesto: recoge la cantidad económica a la que asciende el proyecto.
- Planos: contiene la información gráfica para la comprensión y correcta ejecución del proyecto.

1.4. Especificaciones técnicas

En este apartado se recogen las normas de ejecución y gestión relativas a los aspectos de carácter técnico del proyecto.

1.4.1. Requisitos técnicos para el funcionamiento de las herramientas

Para realizar la programación y creación de las herramientas, es necesario seguir las siguientes prescripciones técnicas expuestas en este apartado. Las características del equipo informático necesario para realizar los cálculos y los requisitos para un buen funcionamiento son: procesador quad-core con mínimo 2 GHz por núcleo, mínimo 4 GB de memoria RAM y sistema operativo de 64 bits operando sobre Windows 10. Dependiendo de los datos importados, es posible que las características varíen tanto a la baja como a la alta. También se requiere un editor de Python, como el programa SPYDER utilizado en este proyecto.

1.4.2. Ejecución

Es requisito que el operario de las herramientas posea conocimientos sobre los detalles básicos de control de sistemas, ya que se requiere de los mismos para poder utilizarlas. Las aplicaciones funcionan de manera intuitiva, todos los inputs poseen su propio texto indicando su significado. El usuario debe colocar los inputs pertinentes y seleccionar el tipo de salida que desea, así pues, evaluando estos datos, el operario puede decidir si repetir la operación para obtener unos resultados más favorables, o utilizar los datos obtenidos en otra de las aplicaciones desarrolladas, utilizarla en el algún medio físico, o bien, comprobar que los resultados son los esperados.

Se ha de tener en cuenta que la notación con la que se han de escribir las ecuaciones es la utiliza por python, separando el numerador y numerador con paréntesis, multiplicando con el símbolo “*”, utilizando “**” para las potencias, y además, se requiere escribir la f.d.t como un único numerador y denominador.

A la hora de cargar los datos experimentales, se debe tener en cuenta que el fichero debe estar en la misma carpeta que la aplicación, y este debe llamarse “datos” en formato .txt.

Presupuesto

Índice de presupuesto

En este documento se explican los diversos gastos monetarios necesarios para el desarrollo de este proyecto. Para desglosar el coste de dichos gastos, se han elaborado dos secciones de precios, el presupuesto de ejecución material (PEM), el presupuesto de ejecución por contrata (PEC) y el presupuesto total incluyendo el IVA.

1. Costes humanos

Aquí se describen los costes humanos del proyecto.

Tabla 5: Costes de personal

	Coste(€/h)	Horas(h)	Total(€)
Ingeniero eléctrico	22	150	3.300

2. Costes materiales

El único coste es el ordenador con el que se han desarrollado las aplicaciones, del que se ha asumido un periodo de amortización de 4 años, el período de uso para el proyecto ha sido dos meses.

Tabla 6: Costes materiales

	Coste(€/u)	Amortización(años)	Período de uso(meses)	Amortizado(€)	Unidades	Total
Ordenador personal	734,4€	4	2	30.6	1	30.6€

El coste amortizado se calcula como:

$$\frac{1 \cdot coste_{total}}{12 \cdot 4} = 30.6€$$

El porcentaje del coste amortizado respecto al total para el desarrollo del programa se calcula como:

$$\frac{coste_{amortizado}}{coste_{total}} \cdot 100 = 4.1\%$$

3. Presupuesto de ejecución material (PEM)

Sumando los gastos anteriormente descritos, se puede obtener el PEM, como se muestra en la siguiente tabla:

Tabla 7: Resumen del PEM

Concepto	Importe(€)
Recursos humanos	3300
Recursos materiales	30.6
PEM	3330.6

Asciende el presupuesto de ejecución material del presente proyecto a la expresada cantidad de DOS MIL CIENTO TREINTA EUROS CON SEIS CÉNTIMOS

Castellón de la Plana, 3 de septiembre de 2020
Alejandro Martínez Soler

4. Presupuesto de ejecución por contrata (PEC)

Dado el presupuesto por ejecución material, es posible calcular el presupuesto de ejecución por contrata, este presupuesto hace referencia al gasto total incluyendo gastos generales y recargos fiscales.

Este gasto adicional se puede asumir el 20% del PEM, así pues:
$Gastosgeneralesyrecargosfiscales = PEM \cdot 20\% = 3330.6 \cdot 0.2 = 666.12€$

Tabla 8: Resumen del PEC

Concepto	Importe(€)
Gastos generales y recargos fiscales	666.12
Subtotal	3330.6
Beneficio industrial	0
PEC	3996.72

Asciende el presupuesto de ejecución por contrata del presente proyecto a la expresada cantidad de TRES MIL NOVECIENTOS NOVENTA Y SEIS CON SETENTA Y DOS CÉNTIMOS.

Castellón de la Plana, 3 de septiembre de 2020

Alejandro Martínez Soler

5. Presupuesto total después de impuestos

Para calcular el presupuesto total del proyecto, es necesario el uso del presupuesto de ejecución por contrata. Para tener en cuenta los honorarios del proyecta, se considera que estos serán del 7% del valor del PEC.

$$Honorarios\ del\ proyectista = 0.07 * PEC = 3996.72 \cdot 0.07 = 279.77€$$

Entonces, sin aplicar todavía el IVA, el resultado total sería:

$$Presupuesto\ total\ sin\ iva = 279.77€ + 3996.72€ = 4276.5€$$

Aplicando el IVA, actualmente al 21%:

$$Presupuesto\ total\ con\ IVA = 4276.5€ \cdot 1.21 = 5174.55€$$

$$Coste\ adicional\ debido\ al\ IVA = 5174.55€ - 4276.5€ = 898.6€$$

Tabla 9: Resumen del presupuesto con impuestos

Concepto	Importe(€)
Honorarios del proyectista	297.77
Presupuesto total sin impuestos	4276.5
Coste del IVA	898.6
Presupuesto total con impuestos	5174.55

Asciende el presupuesto total después de impuestos del proyecto a la expresada cantidad de TRES MIL TRESCIENTOS DIEZ CON DIECIOCHO CÉNTIMOS.

Castellón de la Plana, 3 de septiembre de 2020

Alejandro Martínez Soler