Laboratorio Virtual

Versión 1.0

Kleiver J. Carrasco M.

15 de marzo de 2020

Contenido:

El Laboratorio Virtual de sistemas de control clásicos y difusos es una tesis realizada para la Universidad Nacional Experimental del Táchira como requisito parcial para optar al titulo de ingeniero en electrónica.

Autor: Kleiver J. Carrasco M.

email: kleiver615@gmail.com

Tutor académico: MSc. Ing. Juan R. Vizcaya R.

CAPÍTULO 1

Como usar

1.1 Requisitos

Como primer paso se debe descargar o clonar el repositorio de la aplicación alojado en github:

https://github.com/ezalorpro/LaboratorioVirtual

La aplicación fue programada en python y es necesario tener instalado una versión de python >= 3.7, adicionalmente, se necesitan los módulos listados a continuación:

```
matplotlib>=3.1.1
networkx==2.3
numpy>=1.17.3
parse==1.12.1
PySide2==5.13.0
scipy==1.3.1
```

Para instalar todos los módulos se puede utilizar el administrador de paquetes pip y el archivo requirements.txt ofrecido en el repositorio de la aplicación, el siguiente comando ejecuta la acción de instalación de los módulos:

```
pip install requirements.txt
```

1.2 Ejecución

La aplicación puede ser ejecutada al hacer un llamado de python sobre el archivo main.py, por lo cual se debe estar ubicado al mismo nivel que el archivo main.py:

```
python main.py
```

CAPÍTULO 2

La aplicación

2.1 Estructura del código

El código fue realizado utilizando una combinación de programación estructurada y programación orientada a objetos. En la siguiente imagen se observa un esquema que representa la jerarquía del programa:

En la primera columna se encuentra el archivo main o principal, el cual se encarga de la creación de la ventana gráfica y del llamado de los archivos «handlers», los cuales se observan en la columna dos (Análisis, Entonación, Lógica difusa y Simulación), los archivos «handler» tienen la tarea de manejar la interfaz gráfica y hacer de enlace entre el usuario y las rutinas.

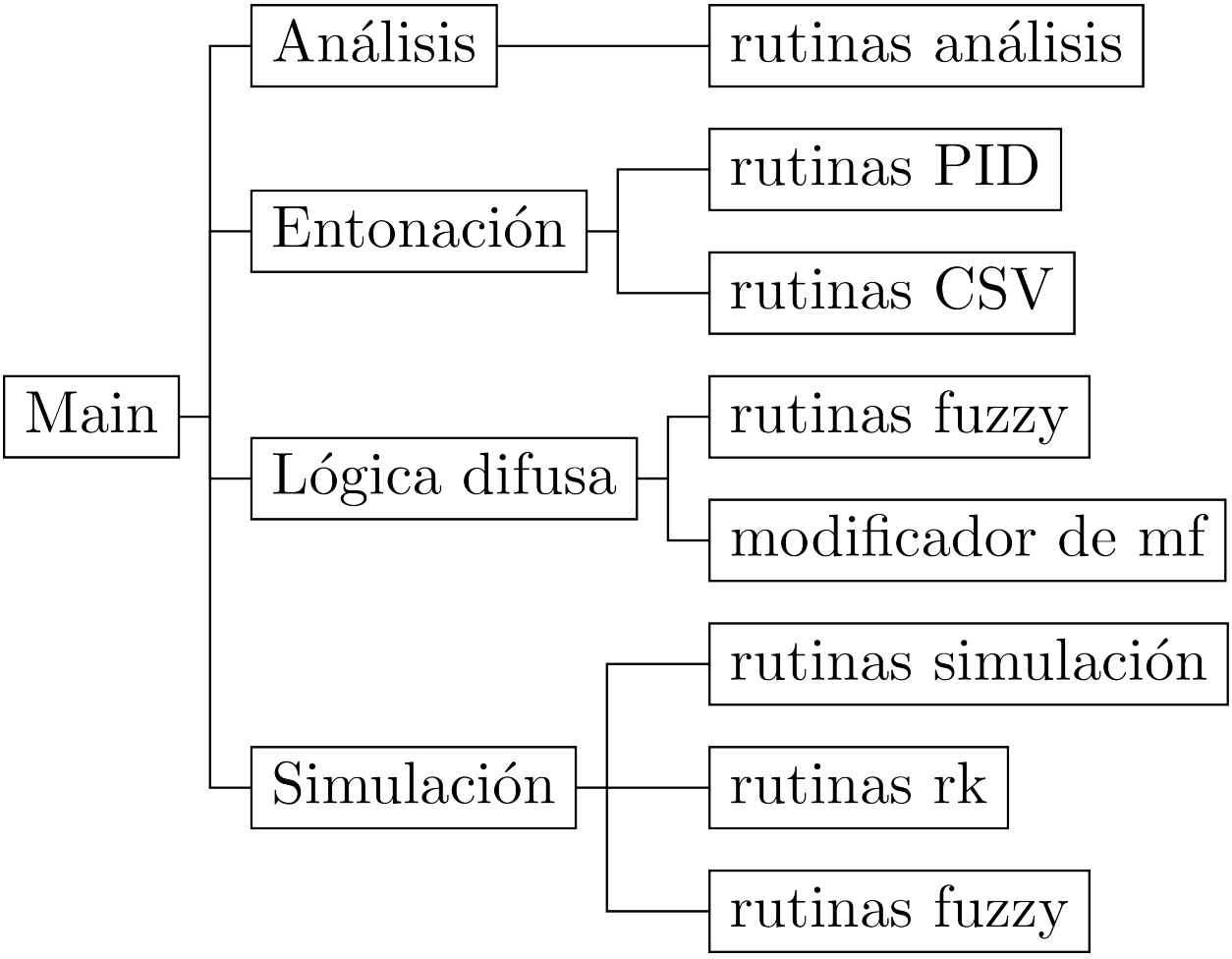
Los archivos de rutinas se observan en la tercera columna, su función es la de realizar los cálculos necesarios que solicite el usuario por medio de la interfaz gráfica.

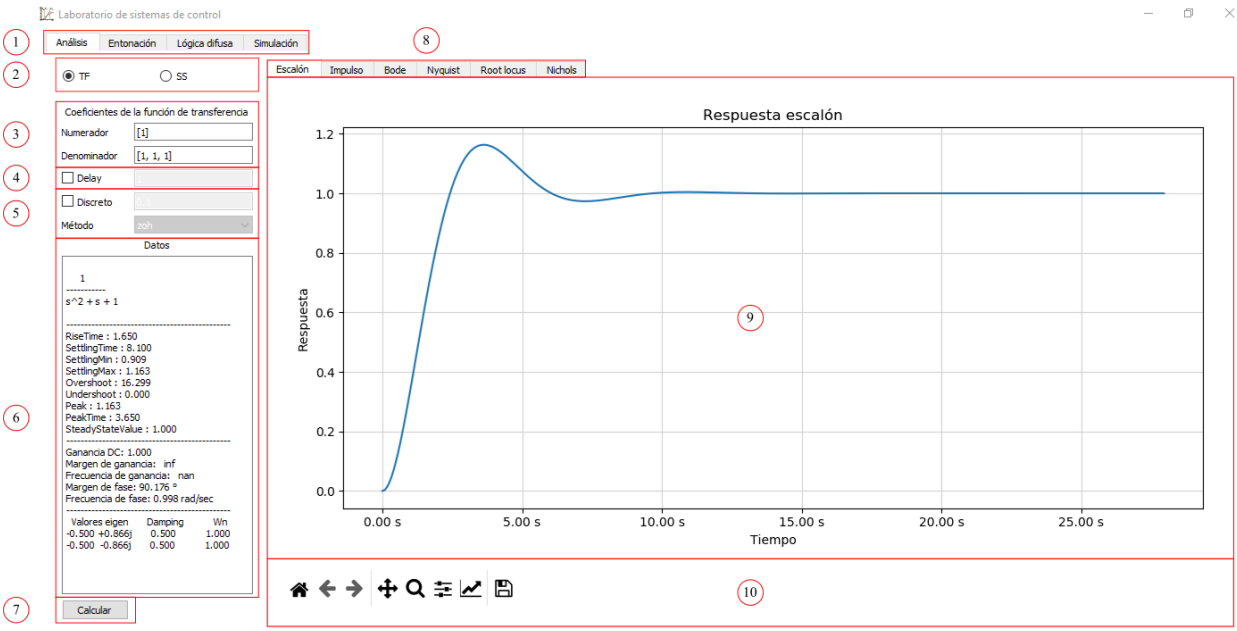
2.2 Interfaz gráfica

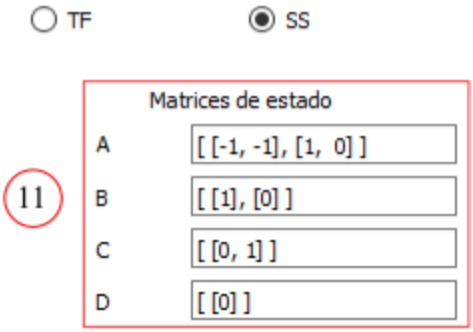
La interfaz gráfica fue realizada utilizando PySide2, las funciones de Laboratorio Virtual fueron separadas en pestañas. En las siguientes imágenes se señalan cada uno de los componentes que integran cada función.

2.2.1 Función de análisis

1. Pestañas de funciones
2. Selector de representación
3. Coeficientes de la función de transferencia
4. Agregado de Delay
5. Discretización del proceso
6. Datos del análisis
7. Botón para realizar el análisis

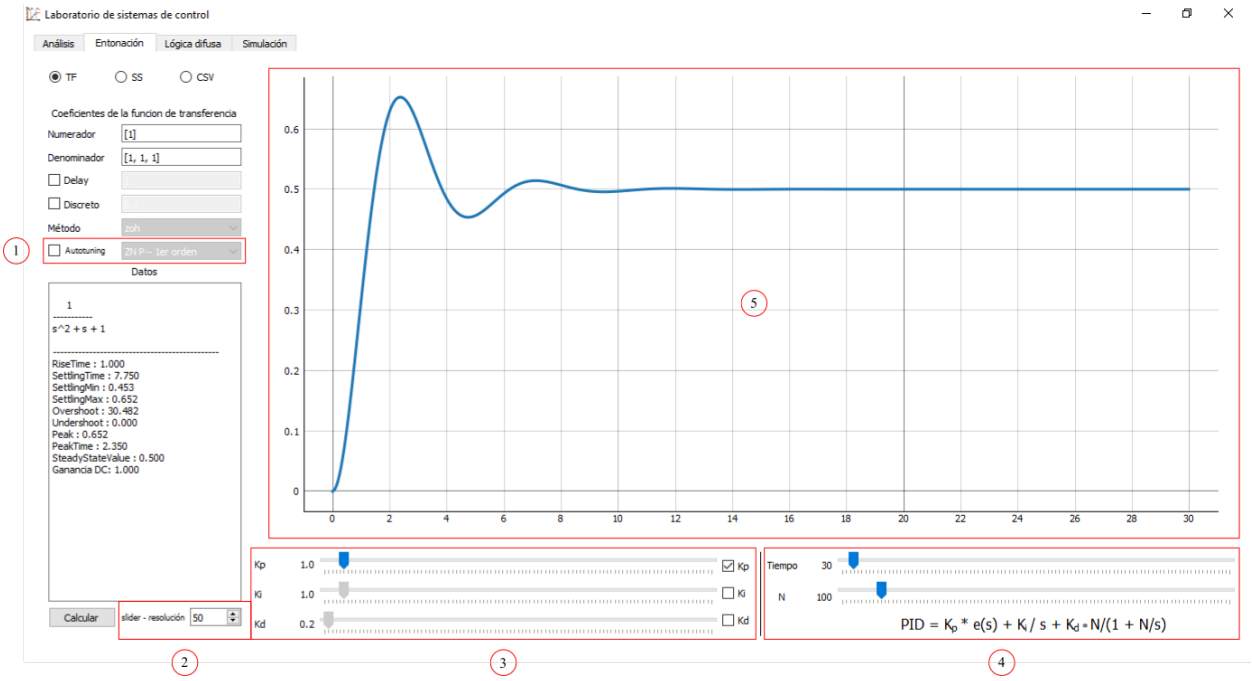

Main
Análisis
rutinas análisis
Entonación
rutinas PID
rutinas CSV
Lógica difusa
rutinas fuzzy
modificador de mf
Simulación
rutinas simulación
rutinas rk
rutinas fuzzy


Laboratorio de sistemas de control
1
Análisis
Entonación
Lógica difusa
Simulación
8
2
TF
SS
Escalón
Impulso
Bode
Nyquist
Root locus
Nichols
Coeficientes de la función de transferencia
3
Numerador
[1]
Denominador
[1, 1, 1]
4
Delay
5
Discreto
Método
zoh
Datos
1
---
s^2 + s + 1
RiseTime : 1.650
SettlingTime : 8.100
SettlingMin : 0.909
SettlingMax : 1.163
Overshoot : 16.299
Undershoot : 0.000
Peak : 1.163
6
PeakTime : 3.650
SteadyStateValue : 1.000
Ganancia DC: 1.000
Margen de ganancia: inf
Frecuencia de ganancia: nan
Margen de fase: 90.176 °
Frecuencia de fase: 0.998 rad/sec
Valores eigen Damping Wn
-0.500 +0.866j 0.500 1.000
-0.500 -0.866j 0.500 1.000
7
Calcular
Respuesta escalón
Respuesta
Tiempo
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.00 s
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
9
10

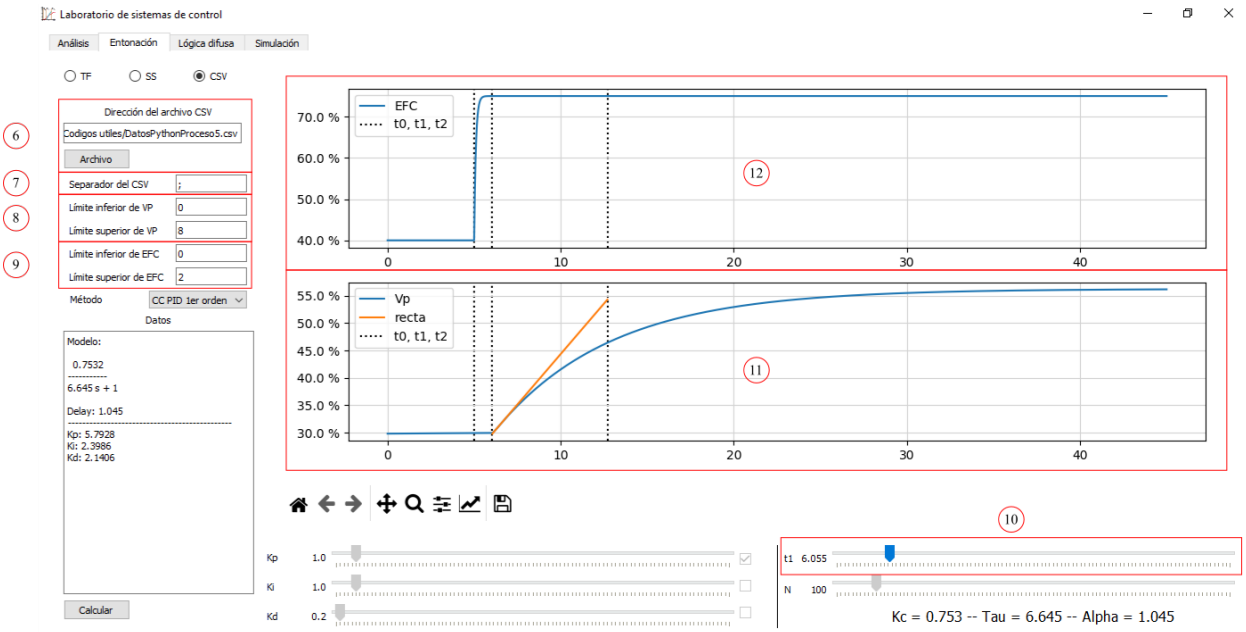


8. Pestañas de gráficas
9. Gráfica con Matplotlib
10. Barra de herramientas de la gráfica
11. Matrices de estados

2.2.2 Función de entonación de controladores PID

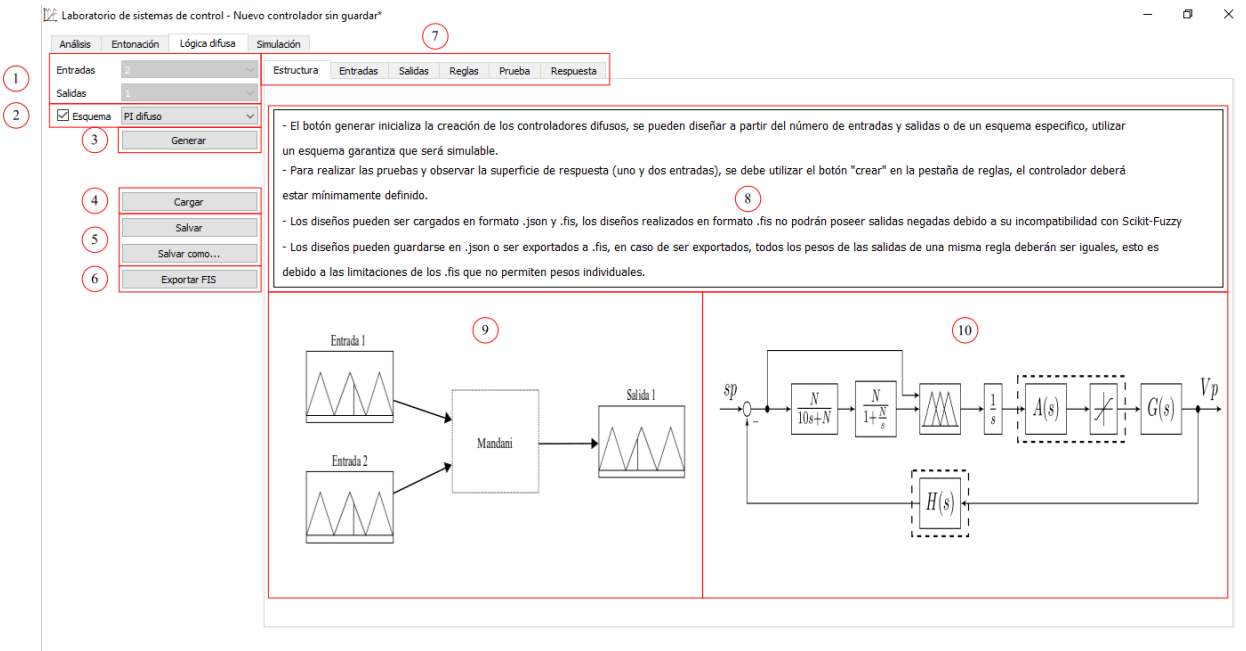


1. Función de entonación automática
2. Resolución de los sliders
3. Sliders de ganancias
4. Sliders de tiempo y coeficiente N
5. Gráfica con PyQtGraph
6. Carga del archivo CSV
7. Separador del archivo CSV

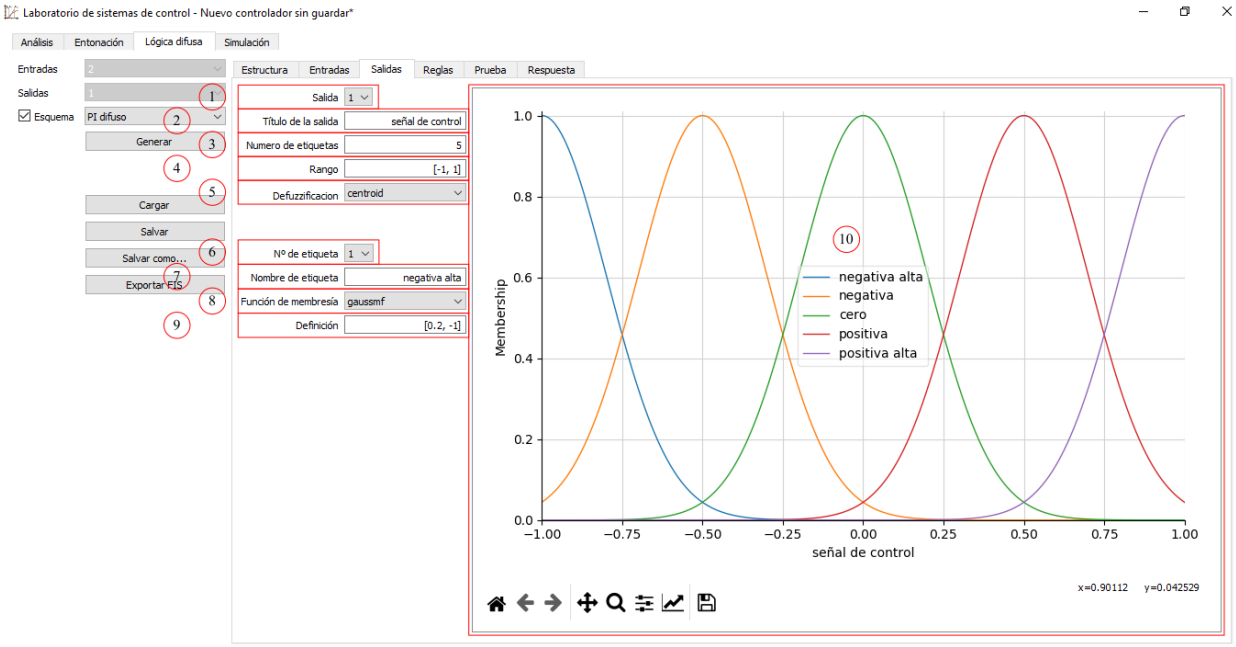


8. SPAN de la variable del proceso
9. SPAN de la entrada al proceso (EFC)
10. Slider para ajustar t_1
11. Gráfica de la variable del proceso
12. Gráfica de la entrada al proceso (EFC)

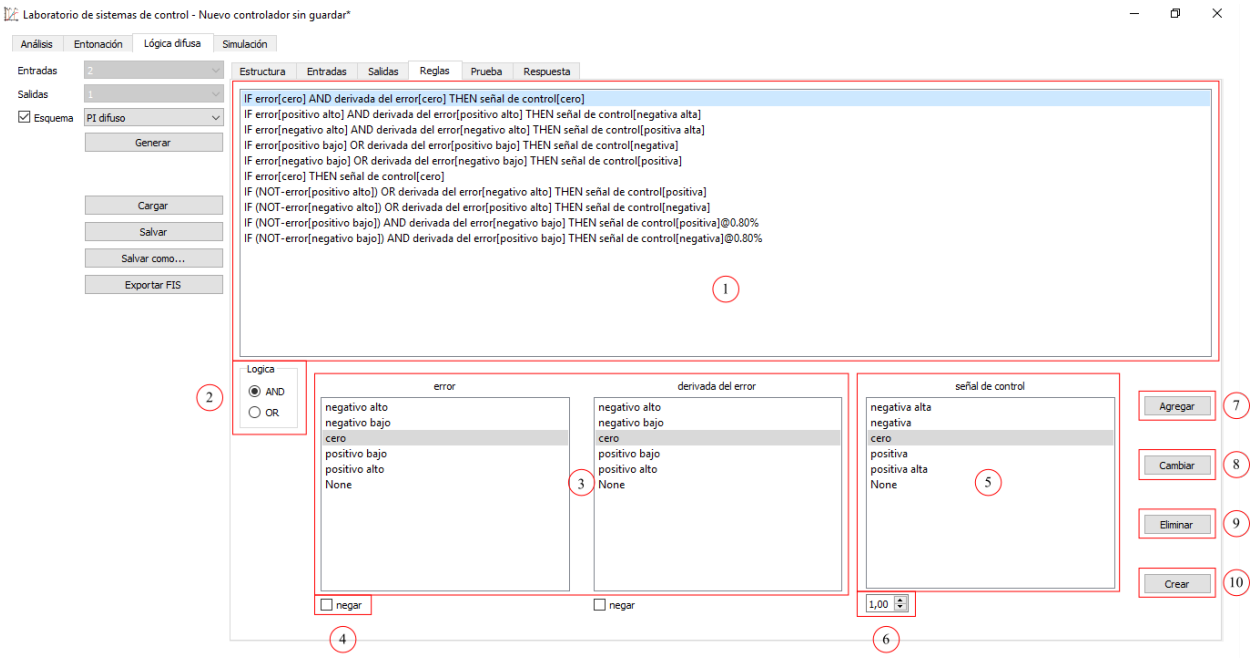
2.2.3 Función de diseño de controladores difusos



1. Número de entradas y salidas
2. Selección de esquema de control
3. Botón para iniciar el diseño
4. Botón para cargar un diseño
5. Botones para salvar los diseños
6. Botón para exportar el diseño a FIS
7. Pestañas para el diseño
8. Información general
9. Estructura de entradas y salidas
10. Esquema de control

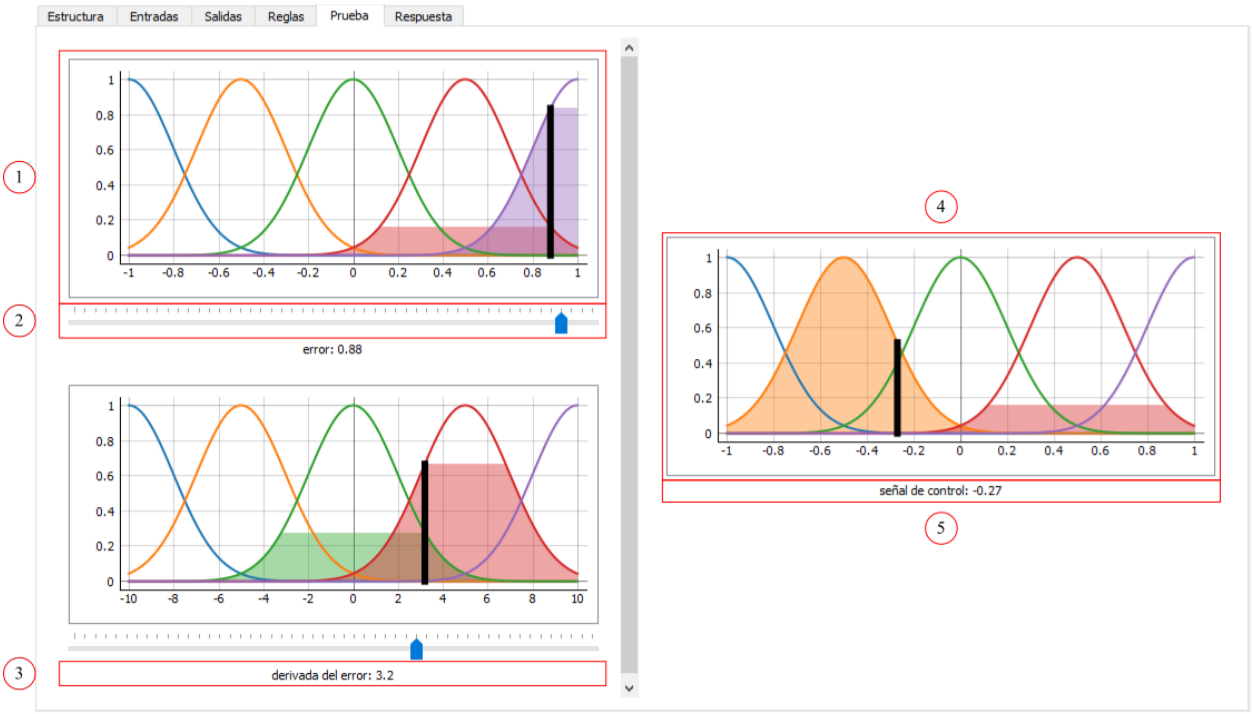


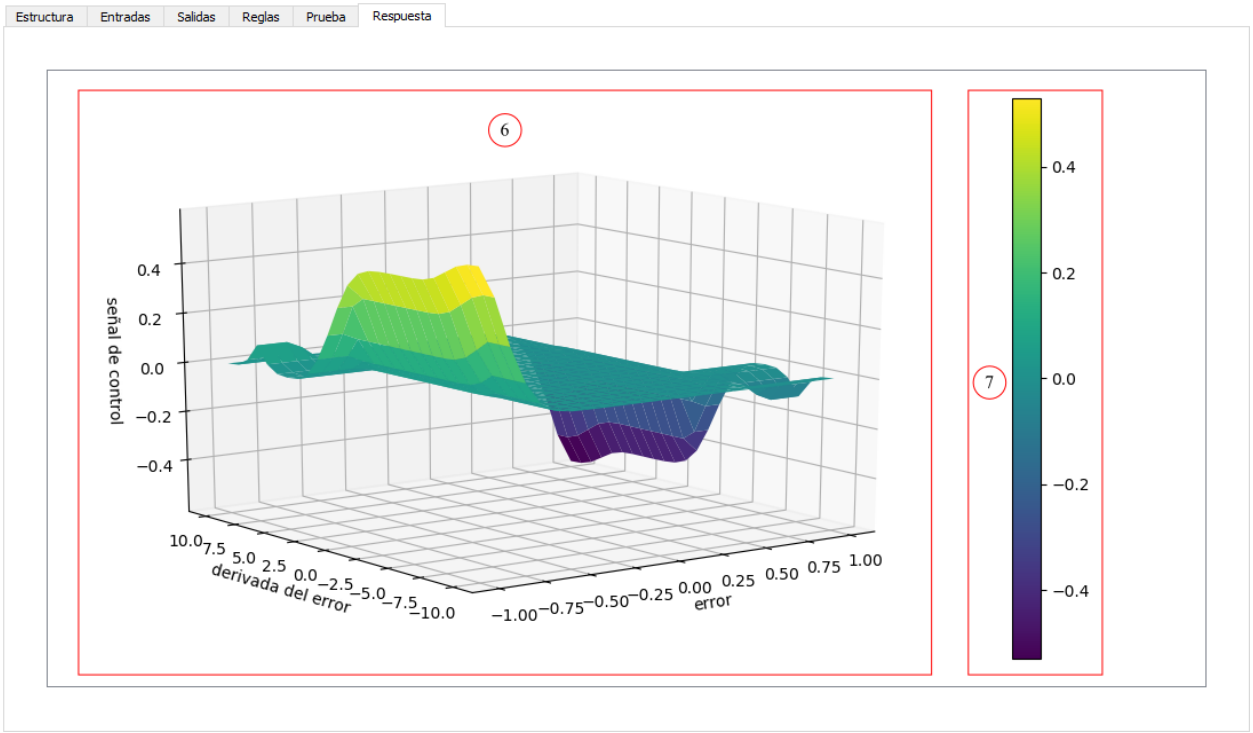
1. Número de entrada/salida
2. Nombre de la entrada/salida
3. Número de etiquetas
4. Rango de la entrada/salida
5. Método de defuzzificacion
6. Número de etiqueta
7. Nombre de la etiqueta
8. Tipo de función de membresía
9. Definición de la función de membresía
10. Gráfica de las funciones de membresía
1. Lista de reglas



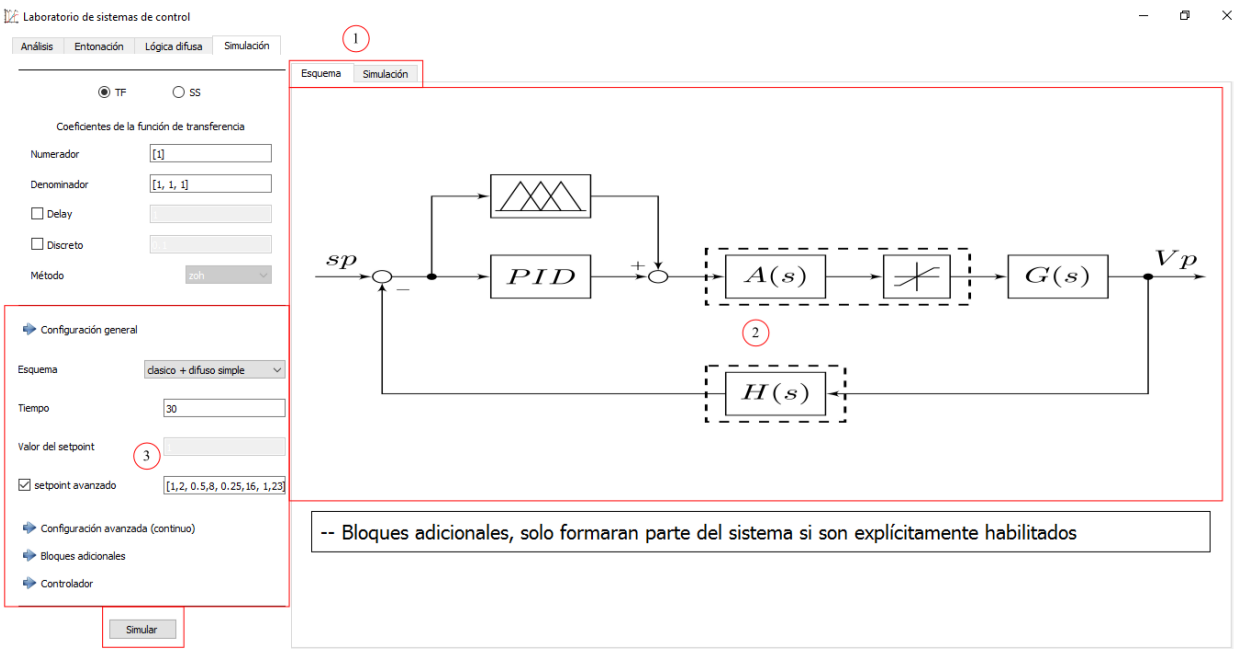
2. Lógica de las premisas
3. Etiquetas de las entradas
4. Opción para negar la entrada
5. Etiquetas de las salidas
6. Peso de la salida
7. Botón para agregar una regla
8. Botón para cambiar una regla
9. Botón para eliminar una regla
10. Botón para crear el controlador y realizar pruebas

1. Activación de reglas de forma gráfica para las entradas
2. Slider para asignar entrada
3. Valor de entrada
4. Activación de reglas de forma gráfica para las salidas
5. Valor de salida
6. Respuesta del controlador
7. Barra indicadora de altura (para dos entradas)


Estructura
Entradas
Salidas
Reglas
Prueba
Respuesta
1
2
error: 0.88
3
derivada del error: 3.2
4
señal de control: -0.27
5

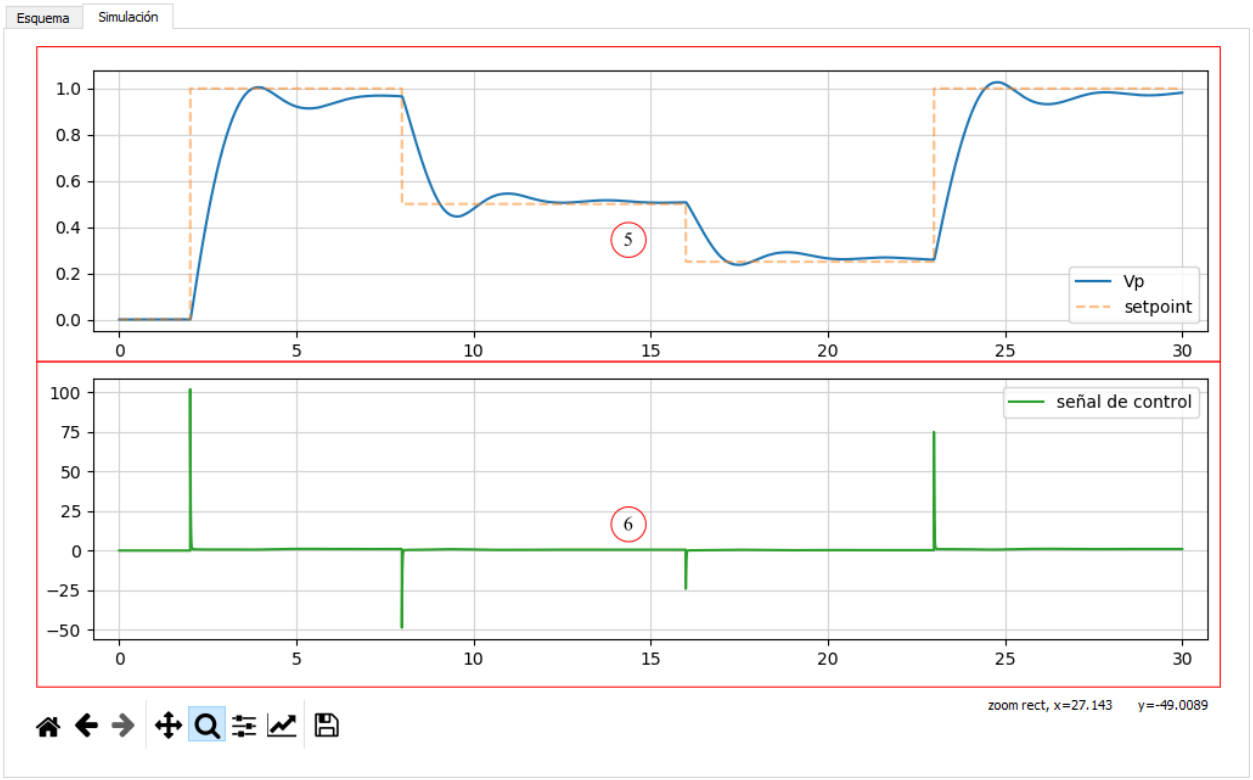

Estructura
Entradas
Salidas
Reglas
Prueba
Respuesta
6
señal de control
derivada del error
error
7

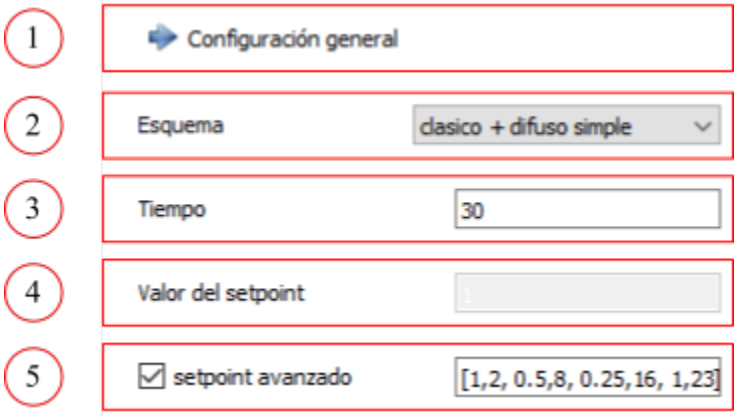
2.2.4 Función de simulación de sistemas de control

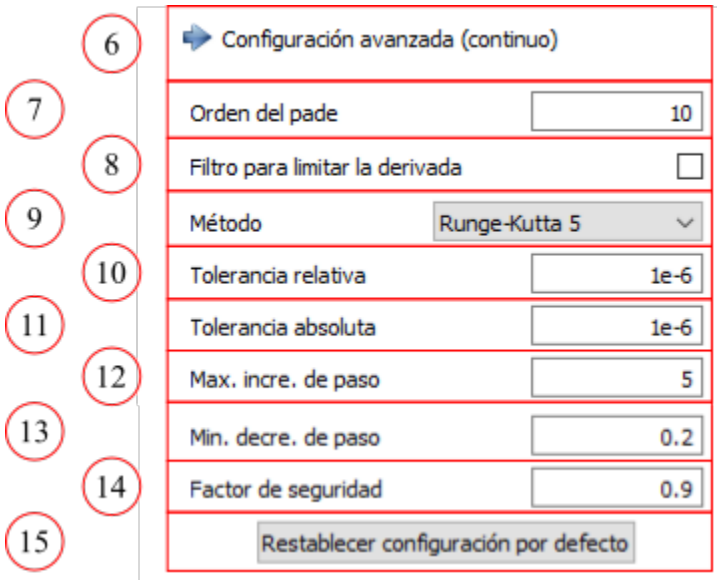


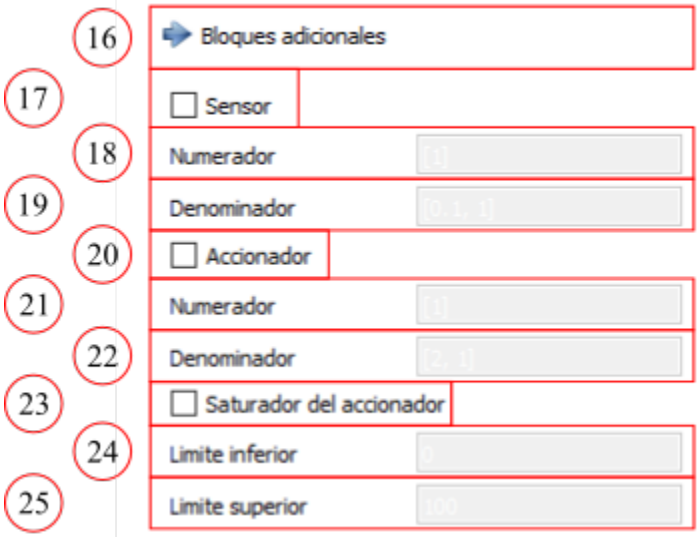
1. Pestañas de simulación
2. Esquema de control
3. Barras de configuración
4. Botón para simular
5. Gráfica de respuesta del sistema
6. Gráfica de la señal de control

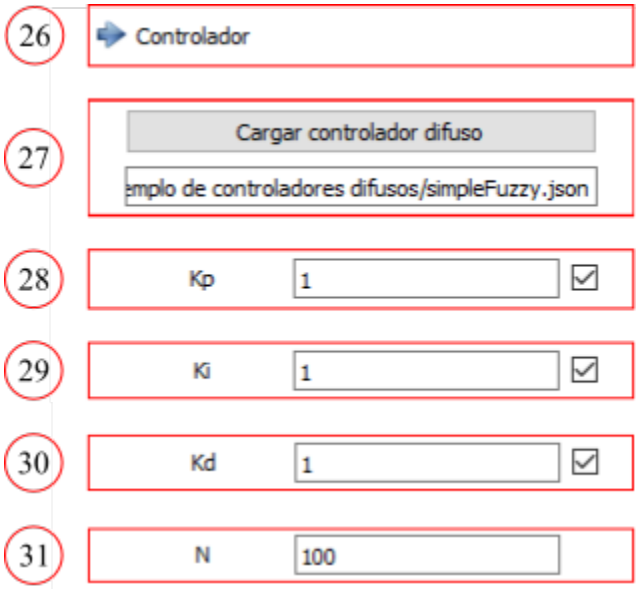
1. Barra de configuración general
2. Selección del esquema de control
3. Tiempo de simulación
4. Valor del setpoint
5. Setpoint avanzado (variable)
6. Barra de configuración avanzada
7. Orden del atraso por PADE
8. Activación del filtro para la derivada
9. Selección del método de Runge-Kutta
10. Tolerancia relativa para el paso variable
11. Tolerancia absoluta para el paso variable
12. Máximo incremento del tamaño de paso
13. Mínimo decremento del tamaño de paso
14. Factor de seguridad para el paso variable


Esquema
Simulación
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
5
Vp
setpoint
100
75
50
25
0
−25
−50
6
señal de control
zoom rect, x=27.143 y=-49.0089


1
Configuración general
2
Esquema
clasico + difuso simple
3
Tiempo
30
4
Valor del setpoint
5
setpoint avanzado
[1,2, 0.5,8, 0.25,16, 1,23]


6
Configuración avanzada (continuo)
7
Orden del pade
10
8
Filtro para limitar la derivada
9
Método
Runge-Kutta 5
10
Tolerancia relativa
1e-6
11
Tolerancia absoluta
1e-6
12
Max. incre. de paso
5
13
Min. decre. de paso
0.2
14
Factor de seguridad
0.9
15
Restablecer configuración por defecto


16
Bloques adicionales
17
Sensor
18
Numerador
19
Denominador
20
Accionador
21
Numerador
22
Denominador
23
Saturador del accionador
24
Limite inferior
25
Limite superior



15. Botón para reiniciar la configuración
16. Barra de bloques adicionales
17. Activación del sensor
18. Numerador del sensor
19. Denominador del sensor
20. Activación del accionador
21. Numerador del accionador
22. Denominador del accionador
23. Activación del saturador
24. Límite inferior del saturador
25. Límite superior del saturador
26. Barra de configuración del controlador
27. Controlador difuso
28. Ganancia proporcional del PID
29. Ganancia integral del PID
30. Ganancia derivativa del PID
31. Coeficiente N

CAPÍTULO 3

Documentación automática

Documentación automática generada a partir de los docstring del código fuente, esta documentación sirve como referencia de las clases y funciones empleadas en la aplicación.

3.1 Promociones de Widgets

3.1.1 focusLineEdit

Archivo para definir la clase FocusLineEdit, esta clases permite promocionar un lineEdit y agregar el evento focus con el fin de generar una señal al dar clic en un lineEdit

class focusLineEdit.**FocusLineEdit**
Clase básica para promocionar el lineEdit

Parámetros **QtWidgets** (*objectType*) – Clase base de los Widgets

focusInEvent(*event*)
Evento de focus para el lineEdit

Parámetros **event** (*tuple*) – Evento generado

3.1.2 mlpwidget

Archivo para definir las clases MlpWidget, MlpWidgetNoToolbar, MlpWidgetSubplot y MlpWidget3D, estas clases son utilizadas por qtdesigner para promocionar un QGraphicsView a las clases aca definidas en orden de mostrar las gráficas en un QGraphicsView

class mlpwidget.**MlpWidget**(*parent=None*)
Clase básica para mostrar gráficas utilizando Matplotlib

Parámetros **QGraphicsView** (*objectType*) – Clase base del QGraphicsView

__init__(*parent=None*)
Initialize self. See help(type(self)) for accurate signature.

class mlpwidget.**MlpWidget3D**(*parent=None*)
Clase básica para mostrar gráficas en 3D utilizando Matplotlib

Parámetros **QGraphicsView** (*objectType*) – Clase base del QGraphicsView

__init__(*parent=None*)
Initialize self. See help(type(self)) for accurate signature.

class mlpwidget.**MlpWidgetNoToolbar**(*parent=None*)
Clase para mostrar gráficas utilizando Matplotlib sin el toolbar

Parámetros **QGraphicsView** (*objectType*) – Clase base del QGraphicsView

__init__(*parent=None*)
Initialize self. See help(type(self)) for accurate signature.

class mlpwidget.**MlpWidgetSubplot**(*parent=None*)
Clase para mostrar gráficas en subplots utilizando Matplotlib

Parámetros **QGraphicsView** (*objectType*) – Clase base del QGraphicsView

__init__(*parent=None*)
Initialize self. See help(type(self)) for accurate signature.

3.1.3 pyqtgraphWidget

Archivo para definir las clases PgraphWidget y PgraphWidgetpid, estas clases son utilizadas por qtdesigner para promocionar un QGraphicsView a las clases aca definidas en orden de mostrar las gráficas en un QGraphicsView

class pyqtgraphWidget.**PgraphWidget**(*parent=None*)
Clase para las gráficas utilizadas en la prueba de los controladores difusos, PyQtGraph es acto para realizar gráficas en tiempo real

Parámetros **QGraphicsView** (*objectType*) – Clase base del QGraphicsView

__init__(*parent=None*)
Initialize self. See help(type(self)) for accurate signature.

class pyqtgraphWidget.**PgraphWidgetpid**(*parent=None*)
Clase para las gráficas utilizadas en el tunning de controladores PID, PyQtGraph es acto para realizar gráficas en tiempo real

Parámetros **QGraphicsView** (*objectType*) – Clase base del QGraphicsView

__init__(*parent=None*)
Initialize self. See help(type(self)) for accurate signature.

3.2 Archivo principal (main)

3.2.1 Archivo Handler para la función de análisis

Archivo de rutinas para la función de análisis

Archivo que contiene todas las rutinas necesarias para la funcionalidad de analisis de sistemas de control

rutinas_analisis.**margenes_ganancias**(*self*, *system*, *mag*, *phase*, *omega*)
Función para obtener el margen de ganancia y el margen de fase.

Parámetros

- **system** (*LTI*) – Representación del sistema
- **mag** (*numpyArray*) – Magnitud de la respuesta en frecuencia
- **phase** (*numpyArray*) – Fase de la respuesta en frecuencia
- **omega** (*numpyArray*) – Frecuencias utilizadas para la respuesta en frecuencia

Devuelve Margenes de ganancia y fase separados en margen de ganancia, margen de fase, frecuencia del margen de ganancia y frecuencia del margen de fase

Tipo del valor devuelto tuple(float, float, float, float)

rutinas_analisis.**rutina_bode_plot**(*self*, *system*)
Función para obtener la respuesta en frecuencia del sistema y su respectiva graficacion en diagrama de bode.

Parámetros **system** (*LTI*) – Representacion del sistema

Devuelve Respuesta en frecuencia separada en vector de magnitudes, vector de fases y vector de frecuencias

Tipo del valor devuelto tuple(numpyArray, numpyArray, numpyArray)

rutinas_analisis.**rutina_impulse_plot**(*self*, *system*, *T*)
Función para obtener la respuesta impulso del sistema y su respectiva graficacion.

Parámetros

- **system** (*LTI*) – Representacion del sistema
- **T** (*numpyArray*) – Vector de tiempo

Devuelve Respuesta impulso separada en vector de tiempo y vector de salida

Tipo del valor devuelto tuple(numpyArray, numpyArray)

rutinas_analisis.**rutina_nichols_plot**(*self*, *system*)
Función para obtener el diagrama de nichols del sistema y su respectiva graficacion, la graficacion se realizo de forma interna en la libreria de control, para esto se modificó la función nichols_plot para poder enviar el axis y la figura, adicionalmente se realizaron algunas modificaciones para una mejor presentación de la gráfica.

Parámetros **system** (*LTI*) – Representacion del sistema

rutinas_analisis.**rutina_nyquist_plot**(*self*, *system*)
Función para obtener la respuesta en frecuencia del sistema y su respectiva graficacion en diagrama de Nyquist.

Parámetros **system** (*LTI*) – Representacion del sistema

Devuelve Respuesta en frecuencia separada en vector de valores reales, vector de valores imaginarios y vector de frecuencias

Tipo del valor devuelto tuple(numpyArray, numpyArray, numpyArray)

rutinas_analisis.**rutina_root_locus_plot**(*self*, *system*)
Función para obtener el lugar de la raíces del sistema y su respectiva graficacion, la graficacion se realizo de forma interna en la libreria de control, para esto se modificó la función root_locus para poder enviar el axis y la figura.

Parámetros **system** (*LTI*) – Representacion del sistema

rutinas_analisis.**rutina_step_plot**(*self*, *system*, *T*)
Función para obtener la respuesta escalón del sistema y su respectiva graficacion.

Parámetros

- **system** (*LTI*) – Representacion del sistema

- **T** (*numpyArray*) – Vector de tiempo

Devuelve Respuesta escalón separada en vector de tiempo y vector de salida

Tipo del valor devuelto tuple(numpyArray, numpyArray)

rutinas_analisis.**rutina_system_info**(*self*, *system*, *T*, *mag*, *phase*, *omega*)

Función para mostrar los resultados obtenidos de los calculos en un TextEdit.

Parámetros

- **system** (*LTI*) – Representacion del sistema
- **T** (*numpyArray*) – Vector de tiempo
- **mag** (*numpyArray*) – Magnitud de la respuesta en frecuencia
- **phase** (*numpyArray*) – Fase de la respuesta en frecuencia
- **omega** (*numpyArray*) – Frecuencias utilizadas para la respuesta en frecuencia

rutinas_analisis.**system_creator_ss**(*self*, *A*, *B*, *C*, *D*)

Función para la creación del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmisión directa la ecuación de espacio de estados.

Parámetros

- **A** (*list*) – Matriz de estados
- **B** (*list*) – Matriz de entrada
- **C** (*list*) – Matriz de salida
- **D** (*list*) – Matriz de transmisión directa

Devuelve El sistema, el vector de tiempo, el sistema con delay y el sistema en el espacio de estados, si el sistema no tiene delay, ambos son iguales

Tipo del valor devuelto tuple(LTI, numpyArray, LTI, LTI)

rutinas_analisis.**system_creator_tf**(*self*, *numerador*, *denominador*)

Función para la creación del sistema a partir de los coeficientes del numerador y del denominador de la función de transferencia.

Parámetros

- **numerador** (*list*) – Coeficientes del numerador
- **denominador** (*list*) – Coeficientes del denominador

Devuelve El sistema, el vector de tiempo y el sistema con delay, si el sistema no tiene delay, ambos son iguales

Tipo del valor devuelto tuple(LTI, numpyArray, LTI)

Archivo para el manejo de la función de analisis de sistemas de control, sirve de intermediario entre la interfaz gráfica y las rutinas de analisis

analisisHandler.**AnalisisHandler**(*self*)

Función principal para el manejo de la funcionalida de analisis de sistemas de control, se crean las señales a ejecutar cuando se interactuá con los widgets incluyendo las validaciones de entradas

analisisHandler.**analisis_bool_discreto**(*self*)

Función para habilitar y deshabilitar el periodo de muestreo

analisisHandler.**analisis_stacked_to_ss**(*self*)

Función para cambiar de función de transferencia a ecuación de espacio de estados

analisisHandler.**analisis_stacked_to_tf**(*self*)
Función para cambiar de ecuación de espacio de estados a función de transferencia

analisisHandler.**calcular_analisis**(*self*)
Función para realizar el los calculos necesarios para la funcionalidad de analisis de sistemas de control, el llamado a esta función se realiza por medio del botón calcular

analisisHandler.**ssA_validator**(*self*)
Validación de la matriz de estados de la ecuación de espacio de estados

analisisHandler.**ssB_validator**(*self*)
Validación de la matriz de entrada de la ecuación de espacio de estados

analisisHandler.**ssC_validator**(*self*)
Validación de la matriz de salida de la ecuación de espacio de estados

analisisHandler.**ssD_validator**(*self*)
Validación de la matriz de transmisión directa de la ecuación de espacio de estados

analisisHandler.**ssdelay_validator**(*self*)
Validación del delay de la ecuación de espacio de estados

analisisHandler.**ssperiodo_validator**(*self*)
Validación del periodo de muestreo de la ecuación de espacio de estados

analisisHandler.**tfdelay_validator**(*self*)
Validación del delay de la función de transferencia

analisisHandler.**tfdem_validator**(*self*)
Validación del denominador de la función de transferencia

analisisHandler.**tfnum_validator**(*self*)
Validación del numerador de la función de transferencia

analisisHandler.**tfperiodo_validator**(*self*)
Validación del periodo de muestreo de la función de transferencia

3.2.2 Archivo Handler para la función de entonación de controladores PID

Archivo de rutinas para la función de entonación de PID

Archivo que contiene todas las rutinas necesarias para la funcionalidad de tunning de PID

rutinas_PID.**auto_tuning_method**(*self*, *k_proceso*, *tau*, *alpha*, *metodo*)
Función para obtener las ganancias del controlador PID a partir de los parametros del modelo de primer orden obtenidos de una respuesta escalón, las formulas son las dadas por Ziegler-Nichols y Cohen-Coon para una respuesta escalón en lazo abierto.

Parámetros

- **k_proceso** (*float*) – Ganancia del proceso
- **tau** (*float*) – Constante de tiempo del proceso
- **alpha** (*float*) – Tiempo muerto o delay del proceso
- **metodo** (*str*) – Método a utilizar

Devuelve Ganancias kp, ki y kd

Tipo del valor devuelto tuple(float, float, float)

rutinas_PID.**model_method**(*self*, *t*, *y*, *dc_gain*)

Función para obtener los parametros del modelo de primer orden de un sistema a partir de su respuesta escalón.

Parámetros

- **t** (*numpyArray*) – Vector de tiempo
- **y** (*numpyArray*) – Vector de respuesta
- **dc_gain** (*float*) – Ganancia DC del sistema

Devuelve Ganancia, constante de tiempo y tiempo muerto del proceso

Tipo del valor devuelto tuple(float, float, float)

rutinas_PID.**rutina_step_plot**(*self*, *system*, *T*, *kp*, *ki*, *kd*)

Función para obtener la respuesta escalón del sistema en lazo cerrado en combinación con un controlador PID y su respectiva graficacion.

Parámetros

- **system** (*LTI*) – Representacion del sistema
- **T** (*numpyArray*) – Vector de tiempo
- **kp** (*float*) – Ganancia proporcional
- **ki** (*float*) – Ganancia integral
- **kd** (*float*) – Ganancia derivativa

Devuelve Respuesta escalón separada en vector de tiempo y vector de salida

Tipo del valor devuelto tuple(numpyArray, numpyArray)

rutinas_PID.**rutina_system_info**(*self*, *system*, *T*, *y*, *kp=0*, *ki=0*, *kd=0*, *autotuning=False*)

Función para mostrar los resultados obtenidos de los calculos en un TextEdit

Parámetros

- **system** (*LTI*) – Representacion del sistema
- **T** (*numpyArray*) – Vector de tiempo
- **y** (*numpyArray*) – Vector de respuesta
- **kp** (*float, opcional*) – Ganancia proporcional, defaults to 0
- **ki** (*float, opcional*) – Ganancia integral, defaults to 0
- **kd** (*float, opcional*) – Ganancia derivativa, defaults to 0
- **autotuning** (*bool, opcional*) – Bandera para señalar si es o no una operación con auto tunning, defaults to False

rutinas_PID.**system_creator_ss**(*self*, *A*, *B*, *C*, *D*)

Función para la creación del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmisión directa.

Parámetros

- **A** (*list*) – Matriz de estados
- **B** (*list*) – Matriz de entrada
- **C** (*list*) – Matriz de salida
- **D** (*list*) – Matriz de transmisión directa

Devuelve El sistema, el vector de tiempo, el sistema con delay, el sistema en el espacio de estados y las ganancias kp, ki y kd. Si el sistema no tiene delay, ambos son iguales

Tipo del valor devuelto tuple(LTI, numpyArray, LTI, LTI, float, float, float)

rutinas_PID.**system_creator_ss_tuning**(*self*, *A*, *B*, *C*, *D*)

Función para la creación del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmisión directa, adicionalmente se realiza el auto tuning utilizando el método escogido por el usuario.

Parámetros

- **A** (*list*) – Matriz de estados
- **B** (*list*) – Matriz de entrada
- **C** (*list*) – Matriz de salida
- **D** (*list*) – Matriz de transmisión directa

Devuelve El sistema, el vector de tiempo, el sistema con delay, el sistema en el espacio de estados y las ganancias kp, ki y kd. Si el sistema no tiene delay, ambos son iguales

Tipo del valor devuelto tuple(LTI, numpyArray, LTI, LTI, float, float, float)

rutinas_PID.**system_creator_tf**(*self*, *numerador*, *denominador*)

Función para la creación del sistema a partir de los coeficientes del numerador y del denominador de la función de transferencia.

Parámetros

- **numerador** (*list*) – Coeficientes del numerador
- **denominador** (*list*) – Coeficientes del denominador

Devuelve El sistema, el vector de tiempo, el sistema con delay y las ganancias kp, ki y kd. Si el sistema no tiene delay, ambos son iguales

Tipo del valor devuelto tuple(LTI, numpyArray, LTI, float, float, float)

rutinas_PID.**system_creator_tf_tuning**(*self*, *numerador*, *denominador*)

Función para la creación del sistema a partir de los coeficientes del numerador y del denominador de la función de transferencia, adicionalmente se realiza el auto tuning utilizando el método escogido por el usuario.

Parámetros

- **A** (*list*) – Matriz de estados
- **B** (*list*) – Matriz de entrada
- **C** (*list*) – Matriz de salida
- **D** (*list*) – Matriz de transmisión directa

Devuelve El sistema, el vector de tiempo, el sistema con delay y las ganancias kp, ki y kd. Si el sistema no tiene delay, ambos son iguales

Tipo del valor devuelto tuple(LTI, numpyArray, LTI, float, float, float)

Archivo de rutinas para la función de entonación de PID utilizando data de un CSV

Archivo que contiene todas las rutinas necesarias para la funcionalidad de identificación de modelo y tunning con csv

rutinas_CSV.**actualizar_Datos**(*self*, *Kc*, *t0*, *t1*, *t2*, *kp*, *ki*, *kd*)

Función para mostrar los resultados obtenidos del modelo en un TextEdit

Parámetros

- **Kc** (*float*) – Ganancia del proceso
- **t0** (*float*) – Tiempo del inicio del escalón
- **t1** (*float*) – Tiempo del inicio de la respuesta del proceso ante el escalón
- **t2** (*float*) – Tiempo en el que el proceso alcanza el 63 % de su valor final respecto al cambio
- **kp** (*float*) – Ganancia proporcional
- **ki** (*float*) – Ganancia integral
- **kd** (*float*) – Ganancia derivativa

rutinas_CSV.**auto_tuning_method_csv**(*self*, *k_proceso*, *tau*, *alpha*, *metodo*)

Función para obtener las ganancias del controlador PID a partir de los parametros del modelo de primer orden obtenidos de una respuesta escalón, las formulas son las dadas por Ziegler-Nichols y Cohen-Coon para una respuesta escalón en lazo abierto

Parámetros

- **k_proceso** (*float*) – Ganancia del proceso
- **tau** (*float*) – Constante de tiempo del proceso
- **alpha** (*float*) – Tiempo muerto o delay del proceso
- **metodo** (*str*) – Método a utilizar

Devuelve Ganancias kp, ki y kd

Tipo del valor devuelto tuple(float, float, float)

rutinas_CSV.**calcular_modelo**(*self*, *dict_data*, *indexTime*, *indexVp*, *indexEFC*, *MinVP*, *MaxVP*, *MinEFC*, *MaxEFC*)

Función para calcular los parametros del modelo de primer orden

Parámetros

- **dict_data** (*dict*) – Diccionario con la data procesada del csv
- **indexTime** (*int*) – Indice que identifica al tiempo
- **indexVp** (*int*) – Indice que identifica a Vp
- **indexEFC** (*int*) – Indice que identifica al EFC
- **MinVP** (*float*) – Limite inferior de Vp
- **MaxVP** (*float*) – Limite superior de Vp
- **MinEFC** (*float*) – Limite inferior de EFC
- **MaxEFC** (*float*) – Limite superior de EFC

Devuelve Datos del modelo de primer orden, recta tangente y puntos asociados a la recta

Tipo del valor devuelto tuple(float, float, float, float, float, float, float, float, float)

rutinas_CSV.**calculos_manual**(*self*, *GraphObjets*, *Kc*, *t0*, *t1*, *t2*, *slop*, *y1*)

Función para recalcular el controlador PID a partir de los datos del modelo de primer orden con el nuevo tiempo t1, ademas, se grafica la data del csv junto con algunos parametros de la identificación del modelo y la nueva recta

Parámetros

- **GraphObjets** (*list*) – Lista de objetos de graficacion
- **Kc** (*float*) – Ganancia del proceso
- **t0** (*float*) – Tiempo del inicio del escalón
- **t1** (*float*) – Tiempo del inicio de la respuesta del proceso ante el escalón
- **t2** (*float*) – Tiempo en el que el proceso alcanza el 63 % de su valor final respecto al cambio
- **slop** (*float*) – Pendiente de la recta de identificación
- **y1** (*float*) – Punto y1 de la recta de identificación, en este punto se encuentra el mayor cambio respecto al tiempo

rutinas_CSV.**entonar_y_graficar**(*self*, *dict_data*, *Kc*, *tau*, *y1*, *y2*, *t0*, *t1*, *t2*)

Función para calcular el controlador PID a partir de los datos del modelo de primer orden, ademas, se graficá la data del csv junto con algunos parametros de la identificación del modelo

Parámetros

- **dict_data** (*dict*) – Diccionario con la data procesada del csv
- **Kc** (*float*) – Ganancia del proceso
- **tau** (*float*) – Constante de tiempo del proceso
- **y1** (*float*) – Punto y1 de la recta de identificación, en este punto se encuentra el mayor cambio respecto al tiempo
- **y2** (*float*) – Punto y2 de la recta de identificación
- **t0** (*float*) – Tiempo del inicio del escalón
- **t1** (*float*) – Tiempo del inicio de la respuesta del proceso ante el escalón
- **t2** (*float*) – Tiempo en el que el proceso alcanza el 63 % de su valor final respecto al cambio

Devuelve Lista de objetos de gráficas y lista de parametros de la recta para el modelado

Tipo del valor devuelto tuple(list[ObjectType, ObjectType, ObjectType, ObjectType, ObjectType, ObjectType, ObjectType, ObjectType], list[float, float, float, float, float, float])

rutinas_CSV.**procesar_csv**(*self*, *csv_data*)

Función para procesar la data del archivo csv, se crea una nueva data en un diccionario, se normalizan las escalas con el span y se transforma el tiempo a segundos. Para la transformación de tiempo a segundos los formatos aceptados son:

hh:mm:ss

mm:ss

ss

En cualquiera de los casos se llevara a segundos y se restara el tiempo inicial para que empiece en cero.

Parámetros **csv_data** (*numpyArray*) – Data del csv

Devuelve Data extraida del archivo CSV asi como indices, máximos y mínimos de la data

Tipo del valor devuelto tuple(dict, list[int, int, int, float, float, float, float])

Archivo para el manejo de la función de Tunning, sirve de intermediario entre la interfaz gráfica y las rutinas de entonación de controladores PID y la identificación de modelos a partir de un archivo CSV y entonación de PID para el mismo

TuningHandler.**LEFC_validator**(*self*)
Validación del limite inferior del span de EFC

TuningHandler.**LVP_validator**(*self*)
Validación del limite inferior del span de VP

TuningHandler.**PID_bool_discreto**(*self*)
Función para habilitar y deshabilitar el periodo de muestreo

TuningHandler.**PID_stacked_to_csv**(*self*)
Función para cambiar a csv

TuningHandler.**PID_stacked_to_ss**(*self*)
Función para cambiar a ecuación de espacio de estados

TuningHandler.**PID_stacked_to_tf**(*self*)
Función para cambiar a función de transferencia

TuningHandler.**TuningHandler**(*self*)
Función principal para el manejo de la funcionalida de Tunning, se crean las señales a ejecutar cuando se interactuá con los widgets incluyendo las validaciones de entradas

TuningHandler.**UEFC_validator**(*self*)
Validación del limite superior del span de EFC

TuningHandler.**UVP_validator**(*self*)
Validación del limite superior del span de VP

TuningHandler.**actualizar_sliders_ss**(*self*)
Función para ajustar la resolución de los sliders con ecuación de espacio de estados

TuningHandler.**actualizar_sliders_tf**(*self*)
Función para ajustar la resolución de los sliders con función de transferencia

TuningHandler.**ajustar_atraso_manual**(*self*)
Función para ajustar el tiempo t1, despues de realizar el calculo para un archivo csv, se utiliza en caso de que la estimación automática no sea lo suficientemente buena

TuningHandler.**calcular_PID**(*self*)
Función para realizar el los calculos necesarios para la funcionalidad de entonación de controladores PID, el llamado a esta función se realizar por medio del botón calcular o cada vez que se modifique alguno de los sliders

TuningHandler.**calcular_autotuning**(*self*)
Función para realizar el los calculos necesarios para la funcionalidad de entonación de controladores PID con auto tunning, el llamado a esta función se realizar por medio del botón calcular si previamente se habilito la funcionalidad de auto tunning

TuningHandler.**calcular_csv**(*self*)
Función para realizar el los calculos necesarios para la funcionalidad de identificación de modelos y entonación de controlador PID, el llamado a esta función se realizar por medio del botón calcular

TuningHandler.**chequeo_de_accion**(*self*)
Para discriminar entre entonación con función de transferencia, ecuación de espacio de estados o identificación de modelo con archivo csv

TuningHandler.**csv_path**(*self*)
Función para cargar el archivo csv

TuningHandler.**ssA_validator**(*self*)
Validación de la matriz de estados de la ecuación de espacio de estados

TuningHandler.**ssB_validator**(*self*)
Validación de la matriz de entrada de la ecuación de espacio de estados

TuningHandler.**ssC_validator**(*self*)
Validación de la matriz de salida de la ecuación de espacio de estados

TuningHandler.**ssD_validator**(*self*)
Validación de la matriz de transmisión directa de la ecuación de espacio de estados

TuningHandler.**ss_habilitar_sliders_checkbox**(*self*)
Función para habilitar ganancias antes y despues del auto tuning con ecuación de espacio de estados

TuningHandler.**ssdelay_validator**(*self*)
Validación del delay de la ecuación de espacio de estados

TuningHandler.**ssperiodo_validator**(*self*)
Validación del periodo de muestreo de la ecuación de espacio de estados

TuningHandler.**tf_habilitar_sliders_checkbox**(*self*)
Función para habilitar ganancias antes y despues del auto tuning con función de transferencia

TuningHandler.**tfdelay_validator**(*self*)
Validación del delay de la función de transferencia

TuningHandler.**tfdem_validator**(*self*)
Validación del denominador de la función de transferencia

TuningHandler.**tfnum_validator**(*self*)
Validación del numerador de la función de transferencia

TuningHandler.**tfperiodo_validator**(*self*)
Validación del periodo de muestreo de la función de transferencia

TuningHandler.**tiempo_slider_cambio**(*self*)
Para discriminar entre entonación e identificación de modelo con archivo csv

TuningHandler.**update_gain_labels**(*self*, *kp=0*, *ki=0*, *kd=0*, *autotuning=False*, *resolution=50*)
Función para actualizar los labels que representan las ganancias, se ejecuta cada vez que un slider de ganancias cambia.

Parámetros

- **kp** (*float, opcional*) – Ganancia proporcional, defaults to 0
- **ki** (*float, opcional*) – Ganancia integral, defaults to 0
- **kd** (*float, opcional*) – Ganancia derivativa, defaults to 0
- **autotuning** (*bool, opcional*) – Bandera para señalar si es o no una operación con autotunning, defaults to False
- **resolution** (*int, opcional*) – Resolución de los sliders, defaults to 50

TuningHandler.**update_time_and_N_labels**(*self*)
Función para actualizar los labels que representan al tiempo y al valor N

3.2.3 Archivo Handler para la función de lógica difusa

Archivo de rutinas que contiene las clases FuzzyController y FISParser

Archivo que contiene las clases FuzzyController y FISParser, para administrar el controlador difuso y cargar y exportar archivos .fis respectivamente

class rutinas_fuzzy.**FISParser**(*file*, *InputList=None*, *OutputList=None*, *RuleEtiquetas=None*)

Clase para cargar y exportar archivos .fis, para cargar los archivos FIS las funciones get_system, get_vars,get_var y get_rules fueron tomadas de yapflm:

Yet Another Python Fuzzy Logic Module: https://github.com/sputnick1124/yapflm

Para obtener los datos necesarias del .fis, de allí, se aplica la función fis_to_json para completar el parsin. En el caso de la exportación, se realiza utilizando la función json_to_fis

__init__(*file*, *InputList=None*, *OutputList=None*, *RuleEtiquetas=None*)

Constructor de la clase, inicializa las variables a utilizar y selecciona entre cargar el fis o exportarlo dependiendo de las variables con las que se cree el objeto

Parámetros

- **file** (*str*) – Dirección del archivo a cargar o exportar
- **inputlist** (*list, opcional*) – Lista de variables de entrada, defaults to None
- **OutputList** (*list, opcional*) – Lista de variables de entrada, defaults to None
- **RuleEtiquetas** (*list, opcional*) – Lista con la información necesaria para crear las reglas, defaults to None

fis_to_json()

Función para completar la creación del controlador a partir de un archivo .fis

Devuelve Conjunto de entradas, salidas y reglas

Tipo del valor devuelto tuple(list)

get_rules()

Función tomada de yapflm (Yet Another Python Fuzzy Logic Module)

get_system()

Función tomada de yapflm (Yet Another Python Fuzzy Logic Module)

get_var(*vartype*, *varnum*, *start_line*, *end_line*)

Función tomada de yapflm (Yet Another Python Fuzzy Logic Module)

get_vars()

Función tomada de yapflm (Yet Another Python Fuzzy Logic Module)

json_to_fis()

Función para exportar el controlador en formato .fis

class rutinas_fuzzy.**FuzzyController**(*inputlist*, *outputlist*, *rulelist=[]*)

Clase para administrar el controlador difuso, a partir de la misma se puede crear el controlador difuso e ir creandolo de forma programática por medio de la interfaz grafica definida en Ui_VentanaPrincipal.py y manejada en FuzzyHandler.py

__init__(*inputlist*, *outputlist*, *rulelist=[]*)

Se utiliza para inicializar el controlador con las entradas y salidas del mismo, en caso de que se envié el parametro opcional, rulelist, se crea el controlador a partir de las reglas suministradas y queda listo para usar

Parámetros

- **inputlist** (*list*) – Lista de variables de entrada
- **outputlist** (*list*) – Lista de variables de salida
- **rulelist** (*list, opcional*) – Lista de reglas, defaults to []

agregar_regla(*window*, *Etiquetasin*, *Etiquetasout*, *lógica*)
Función para crear una regla a partir de un set

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **Etiquetasin** (*list*) – set de entrada
- **Etiquetasout** (*list*) – set de salida
- **lógica** (*bool*) – Lógica a utilizar

Devuelve Ultima regla agregada

Tipo del valor devuelto ObjectType

calcular_valor(*inputs*, *outputs*)
Función para calcular las salidas del controlador dado sus entradas, esta función se utiliza en la funcionalidad de simulación de sistemas de control

Parámetros

- **inputs** (*list*) – Lista con los valores de entrada
- **outputs** (*list*) – Lista viciá del tamaño del numero de salidas

Devuelve Lista de valores de salida del controlador difuso

Tipo del valor devuelto list

cambiar_metodo(*window*, *o*, *metodo*)
Función para cambiar el método defuzzificacion de una salida

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **o** (*str*) – Numero de salida
- **metodo** – Nombre del nuevo método de defuzzificacion

cambiar_nombre_input(*window*, *i*, *nombre*)
Función para cambiar el nombre de una entrada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **i** (*int*) – Numero de entrada
- **nombre** (*str*) – Nuevo nombre de la entrada

cambiar_nombre_output(*window*, *o*, *nombre*)
Función para cambiar el nombre de una salida

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **o** (*int*) – Numero de salida
- **nombre** (*str*) – Nuevo nombre de la salida

cambiar_regla(*window*, *Etiquetasin*, *Etiquetasout*, *index_rule*, *lógica*)
Función para cambiar una regla a partir de un nuevo set

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **Etiquetasin** (*list*) – set de entrada
- **Etiquetasout** (*list*) – set de salida
- **index_rule** (*int*) – Indice indicando la regla a cambiar
- **lógica** (*bool*) – Lógica a utilizar

Devuelve Regla cambiada

Tipo del valor devuelto ObjectType

cambio_etinombre_input(*window*, *inputlist*, *i*, *n*, *old_name*)
Función para cambiar el nombre de una etiqueta en la entrada seleccionada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **inputlist** (*list*) – Lista de variables de entrada
- **i** (*int*) – Numero de entrada
- **n** (*int*) – Numero de etiqueta
- **old_name** (*str*) – Nombre anterior

cambio_etinombre_output(*window*, *outputlist*, *o*, *n*, *old_name*)
Función para cambiar el nombre de una etiqueta en la salida seleccionada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **outputlist** (*list*) – Lista de variables de salida
- **o** (*int*) – Numero de salida
- **n** (*int*) – Numero de etiqueta
- **old_name** (*str*) – Nombre anterior

cambio_etiquetas_input(*window*, *inputlist*, *i*)
Función para actualizar las etiquetas de entrada del controlador

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **inputlist** (*list*) – Lista de variables de entrada
- **i** (*int*) – Numero de entrada

cambio_etiquetas_output(*window*, *outputlist*, *o*)
Función para actualizar las etiquetas de salida del controlador

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **outputlist** (*list*) – Lista de variables de salida
- **o** (*int*) – Numero de salida

crear_controlador()
Función para crear el controlador difuso a partir de todas las reglas creadas

crear_etiquetas_input(*inputlist*)
Función para crear las etiquetas de una entrada a partir de la lista de variables de entrada

Parámetros **inputlist** (*list*) – Lista de variables de entrada

crear_etiquetas_output(*outputlist*)
Función para crear las etiquetas de una salida a partir de la lista de variables de salida

Parámetros **outputlist** (*list*) – Lista de variables de salida

crear_input(*inputlist*)
Función para crear las variables de entrada a partir de la lista de variables de entrada

Parámetros **inputlist** (*list*) – Lista de variables de entrada

Devuelve Variables de entrada

Tipo del valor devuelto list

crear_output(*outputlist*)
Función para crear las variables de salida a partir de la lista de variables de salida

Parámetros **outputlist** (*list*) – Lista de variables de salida

Devuelve Variables de de salida

Tipo del valor devuelto list

crear_plots_in(*window*, *ni*)
Función para crear los objetos de graficacion de PyQtGraph de la entrada, el código para la obtención de los valores de salida y el graficado es una version altamente modificada de la función .view() de Scikit-Fuzzy. Las modificaciones realizadas fueron necesarias para cambiar matplotlib por PyQtGraph

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **ni** (*int*) – Numero de entradas

crear_plots_out(*window*, *no*)
Función para crear los objetos de graficacion de PyQtGraph de la salida, el código para la obtención de los valores de salida y el graficado es una version altamente modificada de la función .view() de Scikit-Fuzzy. Las modificaciones realizadas fueron necesarias para cambiar matplotlib por PyQtGraph

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **no** (*int*) – Numero de salidas

crear_reglas(*rulelistC*)
Función para crear las reglas a partir de una lista que contiene toda la información necesaria, esta lista es creada en FuzzyHandler.py:

Cada posición en la lista contiene un set de entradas, salidas y la lógica a utilizar (AND o OR), a su vez, cada set es una lista que posee en cada posición otra lista con la etiqueta, el numero de entrada/salida y si esta o no negada para el caso de las entradas, en caso de ser salida contiene el peso asignado

Parámetros **rulelistC** (*list*) – Lista con la información necesaria para crear las reglas

Devuelve Lista de reglas

Tipo del valor devuelto list

eliminar_regla(*index_rule*)
Función para eliminar una regla

Parámetros **index_rule** (*int*) – Indice indicando la regla a eliminar

graficar_mf_in(*window*, *i*)
Función para graficar las funciones de membresía de una entrada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **i** (*int*) – Numero de entrada

graficar_mf_out(*window*, *o*)
Función para graficar las funciones de membresía de una salida

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **o** (*int*) – Numero de salida

graficar_prueba_pyqtgraph(*window*, *ni*, *no*)
Función para actualizar la grafica en función de las nuevas entradas, código tomado y modificado de la función .view() de Scikit-Fuzzy y adaptado para su uso con PyQtGraph

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **ni** (*int*) – Numero de entradas
- **no** (*int*) – Numero de salidas

graficar_respuesta_2d(*window*, *inrange*, *no*)
Función para graficar la respuesta del controlador en caso de poseer una entrada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **inrange** (*list*) – Rango de la variable de entrada
- **no** (*int*) – Numero de salidas

graficar_respuesta_3d(*window*, *inrange1*, *inrange2*, *no*)
Función para graficar la superficie de respuesta del controlador en caso de poseer 2 entradas

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **inrange1** (*list*) – Rango de la variable de entrada uno
- **inrange2** (*list*) – Rango de la variable de entrada dos
- **no** (*int*) – Numero de salidas

prueba_de_controlador(*window*, *values*, *ni*, *no*)
Función para realizar la prueba del controlador

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **values** (*list*) – Valores de entradas dados por el usuario con los sliders
- **ni** (*int*) – Numero de entradas

- **no** (*int*) – Numero de salidas

update_definicion_input(*window*, *inputlist*, *i*, *n*)
Función para actualizar la definicion de una función de membresía en la entrada seleccionada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **inputlist** (*list*) – Lista de variables de entrada
- **i** (*int*) – Numero de entrada
- **n** (*int*) – Numero de etiqueta

update_definicion_output(*window*, *outputlist*, *o*, *n*)
Función para actualizar la definicion de una función de membresía en la salida seleccionada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **outputlist** (*list*) – Lista de variables de salida
- **o** (*int*) – Numero de salida
- **n** (*int*) – Numero de etiqueta

update_rango_input(*window*, *inputlist*, *i*)
Función para actualizar el universo de discurso de una entrada

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **inputlist** (*list*) – Lista de variables de entrada
- **i** (*int*) – Numero de entrada

update_rango_output(*window*, *outputlist*, *o*)
Función para actualizar el universo de discurso de una salida

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **outputlist** (*list*) – Lista de variables de salida
- **o** (*int*) – Numero de salida

Archivo para el cambio de definición entre funciones de membresía

Archivo para el cambio de definición entre funciones de membresía, los cambios se realizan en dos pasos:

old_mf -> trimf trimf -> new_mf

De este modo se reduce el número de casos a codificar. Por otro lado, también contiene la función para realizar la validación de las definiciones ingresadas por el usuario

modificadorMf.**update_definicionmf**(*self*, *old_mf*, *definicion*, *new_mf*)
Función para la transformación equivalente entre funciones de membresía

Parámetros

- **old_mf** (*str*) – Nombre de la antigua función de membresía

- **definicion** (*list*) – Lista con los valores correspondiente a la definición de la antigua función de membresía
- **new_mf** (*str*) – Nombre de la nueva función de membresía

Devuelve Definición de la función de membresía y tooltip

Tipo del valor devuelto tuple(list[:], str)

modificadorMf.**validacion_mf**(*self*, *_*, *mf*)
Función para validar las definiciones ingresadas por el usuario

Parámetros

- **_** (*list*) – Definición
- **mf** (*str*) – Nombre de la función de membresía a validar

Archivo para el manejo de la función de diseño de controladores difusos, sirve de intermediario entre la interfaz grafica y la clase creada para manejar el controlador difuso definida en rutinas_fuzzy.py

FuzzyHandler.**EtiquetasDic_creator**(*self*, *j*, *erange*)
Función para crear etiquetas genéricas

Parámetros

- **j** (*int*) – Numero de etiqueta
- **erange** (*list*) – Definición de la función de membresía

Devuelve Diccionario con la información de la etiqueta

Tipo del valor devuelto dict

FuzzyHandler.**FuzzyHandler**(*self*)
Función principal para el manejo de diseño de controladores difusos, se crean las señales a ejecutar cuando se interactuá con los widgets

FuzzyHandler.**actualizar_RulesEtiquetas_in**(*self*, *ni*, *new_name*, *old_name*)
Función para actualizar el nombre en las reglas previamente creadas con el nuevo nombre de una etiqueta

Parámetros

- **ni** (*int*) – Numero de entrada
- **new_name** (*str*) – Nuevo nombre para la etiqueta a cambiar
- **old_name** (*str*) – Antiguo nombre de la etiqueta a cambiar

FuzzyHandler.**actualizar_RulesEtiquetas_out**(*self*, *no*, *new_name*, *old_name*)
Función para actualizar el nombre en las reglas previamente creadas con el nuevo nombre de una etiqueta

Parámetros

- **no** (*int*) – Numero de salida
- **new_name** (*str*) – Nuevo nombre para la etiqueta a cambiar
- **old_name** (*str*) – Antiguo nombre de la etiqueta a cambiar

FuzzyHandler.**cargar_controlador**(*self*)
Función manejar el cargado de controladores previamente diseñados, se aceptan formatos .JSON y .FIS

FuzzyHandler.**cargar_esquema**(*self*)
Función para iniciar el entorno de diseño a partir de un esquema de control seleccionado

FuzzyHandler.**cerrar_prueba**(*self*)
Función para cerrar las pestañas de pruebas ante cambios en el controlador difuso

FuzzyHandler.**check_esquema_show**(*self*)
Función para mediar entre entradas y salidas genéricas y esquemas de control

FuzzyHandler.**crear_controlador**(*self*)
Función para crear el controlador a partir de toda la información recolectada, esta creación se realiza con el fin de realizar la prueba del controlador y observar la superficie de respuesta del controlador en caso de poseer una o dos entradas

FuzzyHandler.**crear_tabs**(*self*)
Función para iniciar el entorno de diseño para entradas y salidas genéricas

FuzzyHandler.**crear_vectores_de_widgets**(*self*)
Función para el almacenado de widgets en listas para acceder a ellos por indices

FuzzyHandler.**definicion_in**(*self*)
Función para manejar el cambio de definicion de la función de membresía correspondiente a la etiqueta actual

FuzzyHandler.**definicion_out**(*self*)
Función para manejar el cambio de definicion de la función de membresía correspondiente a la etiqueta actual

FuzzyHandler.**defuzz_metodo**(*self*)
Función para manejar el metodo de defuzzificacion para la salida seleccionada

FuzzyHandler.**deinificion_in_validator**(*self*)
Función para validar las definiciones de las funciones de membresía

FuzzyHandler.**deinificion_out_validator**(*self*)
Función para validar las definiciones de las funciones de membresía

FuzzyHandler.**exportar_fis**(*self*)
Función manejar el exportado del controlador diseñado a formato .FIS

FuzzyHandler.**guardar_controlador**(*self*)
Función manejar el guardado del controlador diseñado

FuzzyHandler.**guardarcomo_controlador**(*self*)
Función manejar el guardado en un nuevo archivo del controlador diseñado

FuzzyHandler.**imagen_entradas**(*self*)
Función para establecer la imagen del numero de entradas

FuzzyHandler.**imagen_salidas**(*self*)
Función para establecer la imagen del numero de salidas

FuzzyHandler.**inputDic_creator**(*self*, *i*)
Función para crear entradas genéricas

Parámetros **i** (*int*) – Numero de entrada

Devuelve Diccionario con la información de la entrada

Tipo del valor devuelto dict

FuzzyHandler.**nombre_entrada**(*self*)
Función para manejar el cambio de nombre de la entrada seleccionada

FuzzyHandler.**nombre_etiqueta_in**(*self*)
Función para manejar el cambio de nombre de la etiqueta seleccionada de la entrada actual

FuzzyHandler.**nombre_etiqueta_out**(*self*)
Función para manejar el cambio de nombre de la etiqueta seleccionada de la salida actual

FuzzyHandler.**nombre_salida**(*self*)
Función para manejar el cambio de nombre de la salida seleccionada

FuzzyHandler.**numero_de_etiquetas_in**(*self*)
Función para manejar el numero de etiquetas para la entrada seleccionada

FuzzyHandler.**numero_de_etiquetas_out**(*self*)
Función para manejar el numero de etiquetas para la salida seleccionada

FuzzyHandler.**outputDic_creator**(*self*, *i*)
Función para crear salidas genéricas

Parámetros **i** (*int*) – Numero de salida

Devuelve Diccionario con la información de la salida

Tipo del valor devuelto dict

FuzzyHandler.**prueba_input**(*self*)
Función para la ejecución del código correspondiente a la prueba del controlador

FuzzyHandler.**rango_in**(*self*)
Función para manejar el rango para la entrada seleccionada

FuzzyHandler.**rango_out**(*self*)
Función para manejar el rango para la salida seleccionada

FuzzyHandler.**round_list**(*lista*)
Función para redondear los dígitos de una lista

Parámetros **lista** (*list*) – Lista con valores a redondear

Devuelve Lista con valores redondeados

Tipo del valor devuelto list

FuzzyHandler.**rule_list_agregar**(*self*)
Función para crear una nueva regla a partir de las etiquetas seleccionadas para cada entrada y salida

FuzzyHandler.**rule_list_cambiar**(*self*)
Función para modificar una regla

FuzzyHandler.**rule_list_eliminar**(*self*)
Función para eliminar una regla

FuzzyHandler.**rule_list_visualizacion**(*self*)
Función para mostrar las reglas creadas para el controlador actual en un listWidget

FuzzyHandler.**seleccion_entrada**(*self*)
Función para desplegar la información de la entrada seleccionada

FuzzyHandler.**seleccion_etiqueta_in**(*self*)
Función para desplegar la información de la etiqueta seleccionada de la entrada actual

FuzzyHandler.**seleccion_etiqueta_out**(*self*)
Función para desplegar la información de la etiqueta seleccionada de la salida actual

FuzzyHandler.**seleccion_mf_in**(*self*)
Función para manejar el cambio de función de membresía para la etiqueta seleccionada

FuzzyHandler.**seleccion_mf_out**(*self*)
Función para manejar el cambio de función de membresía para la etiqueta seleccionada

FuzzyHandler.**seleccion_salida**(*self*)
Función para desplegar la información de la salida seleccionada

FuzzyHandler.**seleccionar_etiquetas**(*self*)
Función para seleccionar las etiquetas correspondientes a cada entrada/salida de la regla seleccionada

FuzzyHandler.**show_esquema**(*self*)
Función para establecer la imagen del esquema de control seleccionado

3.2.4 Archivo Handler para la función de simulación de sistemas de control

Archivo que contiene las rutinas de simulación y la clase SimpleThread (QtThread)

Archivo que contiene la clase SimpleThread la cual ejecuta la simulación de sistemas de control en hilo diferente al principal, esto se realiza de esta forma debido a que la simulación puede tardar en algunos casos varios segundos, de ejecutarse en el hilo principal presentaria un comportamiento de bloqueo en la ventana principal

class rutinas_simulacion.**SimpleThread**(*window*, *regresar*, *update_bar*, *error_gui*, *list_info*, *parent=None*)
Clase para realizar la simulación de sistemas de control en un hilo diferente al principal

Parámetros **QThread** (*ObjectType*) – Clase para crear un hilo paralelo al principal

__init__(*window*, *regresar*, *update_bar*, *error_gui*, *list_info*, *parent=None*)
Constructor para recibir las variables y funciones del hilo principal

Parámetros

- **window** (*object*) – Objeto que contiene a la ventana principal
- **regresar** (*function*) – Función a la que regresa una vez terminada la simulación, plot_final_results de simulacionHandler.py
- **update_bar** (*function*) – Función para actualizar la barra de progreso, update_progresBar_function de simulacionHandler.py
- **error_gui** (*function*) – Función para mostrar los errores ocurridos durante la simulación, error_gui de simulacionHandler.py
- **list_info** (*list*) – Lista con toda la información necesaria
- **parent** (*NoneType, opcional*) – Sin efecto, defaults to None

run()
Función a ejecutar cuando se hace el llamado a self.start()

run_fuzzy()
Función para realizar la simulación de sistemas de control de esquemas difusos

Devuelve Respuesta obtenida en la simulación dividida en vector de tiempo, vector de salida, vector de la señal de control y vector del setpoint

Tipo del valor devuelto tuple(list[float], deque[float], deque[float], list[float])

run_pid()
Función para realizar la simulación de sistemas de control con controlador PID clásico

Devuelve Respuesta obtenida en la simulación dividida en vector de tiempo, vector de salida, vector de la señal de control y vector del setpoint

Tipo del valor devuelto tuple(list[float], deque[float], deque[float], list[float])

stop()
Función para detener el hilo

rutinas_simulacion.**controlador_validator**(*self*, *esquema*, *InputList*, *OutputList*, *RuleEtiquetas*)
Función para validar los controladores difusos con respecto al esquema de control seleccionado

Parámetros

- **esquema** (*int*) – Esquema de control seleccionado representado por un valor
- **InputList** (*list*) – Lista de entradas
- **OutputList** (*list*) – Lista de salidas
- **RuleEtiquetas** (*list*) – Lista con set de reglas

rutinas_simulacion.**system_creator_ss**(*self*, *A*, *B*, *C*, *D*)

Función para la creación del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmisión directa.

Parámetros

- **A** (*list*) – Matriz de estados
- **B** (*list*) – Matriz de entrada
- **C** (*list*) – Matriz de salida
- **D** (*list*) – Matriz de transmisión directa

Devuelve El sistema creado

Tipo del valor devuelto LTI

rutinas_simulacion.**system_creator_tf**(*self*, *numerador*, *denominador*)

Función para la creación del sistema a partir de los coeficientes del numerador y del denominador de la función de transferencia.

Parámetros

- **numerador** (*list*) – Coeficientes del numerador
- **denominador** (*list*) – Coeficientes del denominador

Devuelve El sistema creado

Tipo del valor devuelto LTI

Archivo para definir los algoritmos de ajuste del tamaño de paso para los Runge-kutta explícitos y embebidos

Archivo para definir los algoritmos de ajuste del tamaño de paso para los Runge-kutta explícitos y embebidos, en el caso de los métodos explícitos se utiliza el método de doble paso

rutinas_rk.**rk_doble_paso_adaptativo**(*systema*, *h_ant*, *tiempo*, *tbound*, *xVectB*, *entrada*, *metodo*, *ordenq*, *rtol*, *atol*, *max_step_increase*, *min_step_decrease*, *safety_factor*)

Función para definir y manejar el ajuste del tamaño de paso por el método de doble paso para Runge-kutta's explícitos, la función está realizada de forma específica para trabajar con sistemas de control representados con ecuaciones de espacio de estados

Parámetros

- **systema** (*LTI*) – Representación del sistema de control
- **h_ant** (*float*) – Tamaño de paso actual
- **tiempo** (*float*) – Tiempo actual
- **tbound** (*float*) – Tiempo máximo de simulación
- **xVectB** (*numpyArray*) – Vector de estado

- **entrada** (*float*) – Valor de entrada al sistema
- **metodo** (*function*) – Runge-Kutta a utilizar: RK2, Rk3, etc.
- **ordenq** (*int*) – Orden del método
- **rtol** (*float*) – Tolerancia relativa
- **atol** (*float*) – Tolerancia absoluta
- **max_step_increase** (*float*) – Máximo incremento del tamaño de paso
- **min_step_decrease** (*float*) – Mínimo decremento del tamaño de paso
- **safety_factor** (*float*) – Factor de seguridad

Devuelve El tamaño de paso anterior, el nuevo tamaño de paso, la salida y el vector de estado

Tipo del valor devuelto tuple(float, float, float, numpyArray)

rutinas_rk.**rk_embebido_adaptativo**(*systema*, *h_ant*, *tiempo*, *tbound*, *xVectr*, *entrada*, *metodo*, *ordenq*, *rtol*, *atol*, *max_step_increase*, *min_step_decrease*, *safety_factor*)

Función para definir y manejar el ajuste del tamaño de paso para Runge-kutta's embebidos, la función esta realizada de forma específica para trabajar con sistemas de control representados con ecuaciones de espacio de estados

Parámetros

- **systema** (*LTI*) – Representación del sistema de control
- **h_ant** (*float*) – Tamaño de paso actual
- **tiempo** (*float*) – Tiempo actual
- **tbound** (*float*) – Tiempo máximo de simulación
- **xVectB** (*numpyArray*) – Vector de estado
- **entrada** (*float*) – Valor de entrada al sistema
- **metodo** (*function*) – Runge-Kutta a utilizar: DOPRI54, RKF45, etc.
- **ordenq** (*int*) – Valor del método de menor orden
- **rtol** (*float*) – Tolerancia relativa
- **atol** (*float*) – Tolerancia absoluta
- **max_step_increase** (*float*) – Máximo incremento del tamaño de paso
- **min_step_decrease** (*float*) – Mínimo decremento del tamaño de paso
- **safety_factor** (*float*) – Factor de seguridad

Devuelve El tamaño de paso anterior, el nuevo tamaño de paso, la salida y el vector de estado

Tipo del valor devuelto tuple(float, float, float, numpyArray)

Archivo para compilar los Runge-kutta explicitos y embebidos utilizando numba

Archivo para compilar los Runge-kutta explícitos y embebidos utilizando numba, los metodos quedan guardados en el archivo: metodos_RK.cp37-win32.pyd o metodos_RK.cpython-37m-x86_64-linux-gnu.so dependiendo del sistema operativo utilizado y pueden ser importados como un modulo cualquiera y utilizar las funciones allí definidas.

rk_generator.**SSPRK3**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta con preservado de estabilidad fuerte de orden 3, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**bogacki_shampine23**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta embebido de Bogacki-Shampine 3(2), la integración se continua con la salida de orden 3, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**cash_karp45**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta embebido de Cash-Karp 4(5), la integración se continua con la salida de orden 4, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**dopri54**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta embebido de Dormand-Prince 5(4), la integración se continua con la salida de orden 5, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**fehlberg45**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta embebido de Fehlberg 4(5), la integración se continua con la salida de orden 4, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**heun3**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta Heun de orden 3, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**norm**(*x*)

Función para calcular la norma RMS de un vector. Función tomada de SciPy

Parámetros **x** (*numpyArray*) – Vector

Devuelve Norma rms del vector ingresado

Tipo del valor devuelto float

rk_generator.**ralston3**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta Ralston de orden 3, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**ralston4**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta Ralston con mínimo error de truncamiento de orden 4, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**runge_kutta2**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta de orden 2, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**runge_kutta3**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta de orden 3, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa

- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**runge_kutta4**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta de orden 4, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**runge_kutta5**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta de orden 5, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

rk_generator.**tres_octavos4**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)
Runge-Kutta 3/8 de orden 4, en el método se asumió entrada constante, por lo que se descarta t + h*cs

Parámetros

- **A** (*float64, 2d, F*) – Matriz de estados
- **B** (*float64, 2d, C*) – Matriz de entrada
- **C** (*float64, 2d, C*) – Matriz de salida
- **D** (*float64, 2d, C*) – Matriz de transmisión directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

Archivo para compilar las funciones encargadas de la simulación en tiempo discreto utilizando numba

Archivo para compilar las funciones encargadas de la simulación en tiempo discreto utilizando numba, las funciones quedan guardadas en el archivo: discreto_sim.cp37-win32.pyd y pueden ser importadas desde el archivo como una función de un modulo

discreto_generator.**PID_discreto**(*error*, *ts*, *s_integral*, *error_anterior*, *kp*, *ki*, *kd*)

Función para calcular el PID en forma discreta

Parámetros

- **error** (*float*) – Señal de error
- **ts** (*float*) – Periodo de muestreo
- **s_integral** (*float*) – Acumulador de la señal integral
- **error_anterior** (*deque Object*) – deque con el error anterior
- **kp** (*float*) – Ganancia proporcional
- **ki** (*float*) – Ganancia integral
- **kd** (*float*) – Ganancia derivativa

Devuelve La salida del controlador PID discreto y el error actual

Tipo del valor devuelto tuple(float, float, deque[float])

discreto_generator.**derivadas_discretas**(*error*, *ts*, *error_anterior*)

Función para calcular la derivada del error y la segunda derivada del error

Parámetros

- **error** (*float*) – Señal de error
- **ts** (*float*) – Periodo de muestreo
- **error_anterior** (*deque Object*) – deque con el error anterior

Devuelve La salida del controlador PID discreto

Devuelve La primera y segunda derivada de la señal de error y el error actual

Tipo del valor devuelto tuple(float, float, deque[float])

discreto_generator.**ss_discreta**(*A*, *B*, *C*, *D*, *x*, *_*, *inputValue*)

Función para calcular la respuesta del sistema por medio de la representacion discreta de las ecuaciones de espacio de estados

Parámetros

- **ss** (*LTI*) – Representacion del sistema
- **x** (*numpyArray*) – Vector de estado
- **_** (*float*) – No importa
- **inputValue** (*float*) – Valor de entrada al sistema

Devuelve La salida del sistema y el vector de estado

Tipo del valor devuelto tuple(float, numpyArray)

Archivo para el manejo de la función de simulación de sistemas de control, sirve de intermediario entre la interfaz grafica y la clase creada para manejar la simulación en una hilo distinto, esto es debido al tiempo que puede llegar a tomar cada simulación

simulacionHandler.**N_validator**(*self*)
Validación del valor N

simulacionHandler.**SimulacionHandler**(*self*)
Función principal para el manejo de la funcionalida de simulación de sistemas de control, se crean las señales a ejecutar cuando se interactúa con los widgets incluyendo las validaciones de entradas

simulacionHandler.**accion_esquema_selector**(*self*)
Función para mostrar los widgets indicados en función del esquema seleccionado

simulacionHandler.**accionadordem_validator**(*self*)
Validación del denominador de la función de transferencia correspondiente al accionador

simulacionHandler.**accionadornum_validator**(*self*)
Validación del numerador de la función de transferencia correspondiente al accionador

simulacionHandler.**atol_validator**(*self*)
Validación de la tolerancia absoluta

simulacionHandler.**calcular_simulacion**(*self*)
Función para inicializar el QThread y realizar los calculos de la simulación

simulacionHandler.**configuration_data**(*self*)
Función para cambiar la configuración del solver a utilizar

Devuelve Datos necesarios para el solver

Tipo del valor devuelto tuple(function, function, list[function, int, float, float, float, float, float])

simulacionHandler.**error_gui**(*self*, *error*)
Función para mostrar los errores que pudiesen ocurrir durante la simulación, esta función es utilizada por el QThread

Parámetros **error** (*int*) – Indicador del error

simulacionHandler.**escalonAvanzado_validator**(*self*)
Validación del escalon avanzado

simulacionHandler.**escalon_validator**(*self*)
Validación del escalon simple

simulacionHandler.**get_pathcontroller1**(*self*)
Función para obtener la dirección al archivo del controlador difuso

simulacionHandler.**get_pathcontroller2**(*self*)
Función para obtener la dirección al archivo del controlador difuso 2 (PD)

simulacionHandler.**inferiorSaturador_validator**(*self*)
Validación del limite inferior del saturador

simulacionHandler.**kd_validator**(*self*)
Validación de la ganancia derivativa

simulacionHandler.**ki_validator**(*self*)
Validación de la ganancia integral

simulacionHandler.**kp_validator**(*self*)
Validación de la ganancia proporcional

simulacionHandler.**maxstep_validator**(*self*)
Validación del incremento máximo de paso

simulacionHandler.**minstep_validator**(*self*)
Validación del decremento mínimo de paso

simulacionHandler.**pade_validator**(*self*)
Validación del orden del pade

simulacionHandler.**plot_final_results**(*self*, *result*)
Función para graficar los resultados finales de la simulación

Parámetros **result** (*list*) – Lista con los resultados obtenidos

simulacionHandler.**restablecer_configuracion**(*self*)
Función para restablecer la configuración avanzada por defecto

simulacionHandler.**rtol_validator**(*self*)
Validación de la tolerancia relativa

simulacionHandler.**safetyFactor_validator**(*self*)
Validación del factor de seguridad

simulacionHandler.**sensordem_validator**(*self*)
Validación del denominador de la función de transferencia correspondiente al sensor

simulacionHandler.**sensornum_validator**(*self*)
Validación del numerador de la función de transferencia correspondiente al sensor

simulacionHandler.**simulacion_stacked_to_ss**(*self*)
Función para cambiar de función de transferencia a ecuación de espacio de estados

simulacionHandler.**simulacion_stacked_to_tf**(*self*)
Función para cambiar de ecuación de espacio de estados a función de transferencia

simulacionHandler.**ssA_validator**(*self*)
Validación de la matriz de estados de la ecuación de espacio de estados

simulacionHandler.**ssB_validator**(*self*)
Validación de la matriz de entrada de la ecuación de espacio de estados

simulacionHandler.**ssC_validator**(*self*)
Validación de la matriz de salida de la ecuación de espacio de estados

simulacionHandler.**ssD_validator**(*self*)
Validación de la matriz de transmisión directa de la ecuación de espacio de estados

simulacionHandler.**ssdelay_validator**(*self*)
Validación del delay de la ecuación de espacio de estados

simulacionHandler.**ssperiodo_validator**(*self*)
Validación del periodo de muestreo de la ecuación de espacio de estados

simulacionHandler.**superiorSaturador_validator**(*self*)
Validación del limite superior del saturador

simulacionHandler.**tfdelay_validator**(*self*)
Validación del delay de la función de transferencia

simulacionHandler.**tfdem_validator**(*self*)
Validación del denominador de la función de transferencia

simulacionHandler.**tfnum_validator**(*self*)
Validación del numerador de la función de transferencia

simulacionHandler.**tfperiodo_validator**(*self*)
Validación del periodo de muestreo de la función de transferencia

simulacionHandler.**tiempo_validator**(*self*)
Validación del tiempo de simulación

simulacionHandler.**update_progresBar_function**(*self*, *value*)
Función para actualizar la barra de progreso de la simulación, esta función es utilizada por el QThread

Parámetros **value** (*float*) – Valor en porcentaje del progreso

Archivo principal, en orden de ejecutar la aplicación, este es el archivo a ejecutar

class main.**MainWindow**(*parent=None*)
Clase principal del programa, esta clase hereda de QMainWindow y Ui_MainWindow, la primera es la clase base de ventanas que ofrece Qt mientras que la segunda es la clase que se crea a partir de qtdesigner y quien posee toda la definición de toda la interfaz gráfica. Desde aca se ejecutan los archivos Handler, quienes representan los enlaces entre las rutinas y la interfaz gráfica de cada una de las funciones del laboratorio virtual, estos Handlers se tratan como si fueran una extension de esta clase, por tanto, se les enviá self y se recibe self y se sigue tratando como si fuera parte de la clase.

Parámetros

- **QtWidgets** (*ObjectType*) – Clase base de ventana ofrecida por Qt
- **Ui_MainWindow** (*ObjectType*) – Clase con la interfaz grafica autogenerada con qtdesigner

__init__(*parent=None*)
Constructor de la clase, aca se inicializan los objetos de las clases heredadas y se hacen los llamados a los Handlers

Parámetros **parent** (*NoneType, opcional*) – Sin efecto, defaults to None

closeEvent(*event*)
Evento pera el cerrado de la ventana

resource_path(*relative_path*)
Función para generar direcciones absolutas a partir de direcciones relativas

Parámetros **relative_path** (*str*) – dirección relativa

Devuelve dirección absoluta

Tipo del valor devuelto str

CAPÍTULO 4

Indices y tablas

- genindex
- modindex
- search

Índice de Módulos Python

Índice

Símbolos

A

B

C

D

E

F

G

S

T

U

V