Laboratorio Virtual de sistemas de control clasicos y difusos utilizando software libre

Versión 1.0

Kleiver J. Carrasco M.

09 de marzo de 2020

Contenido:

CAPÍTULO 1

Promociones de Widgets

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 basica 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

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 graficas en un QGraphicsView

class mlpwidget.**MlpWidget**(*parent=None*)
Clase basica para mostrar graficas 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 basica para mostrar graficas 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 graficas 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 graficas en subplots utilizando Matplotlib

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

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

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 graficas en un QGraphicsView

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

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

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

CAPÍTULO 2

Archivo principal (main)

2.1 Archivo Handler para la función de análisis

2.1.1 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

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

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

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

Parámetros

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

rutinas_analisis.**rutina_nichols_plot**(*self*, *system*)
Funcion para obtener el diagram de nichols del sistema y su respectiva graficacion, la graficacion se realizo de forma interna en la libreria de control, para esto se moodifico la funcion nichols_plot para poder enviar el axis y la figura, adicionalmente se realizaron algunas modificaciones para una mejor presentacion de la grafica

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

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

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

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

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

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

Parámetros

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

rutinas_analisis.**rutina_system_info**(*self*, *system*, *T*, *mag*, *phase*, *omega*)
Funcion 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*)
Funcion para la creacion del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmision directa la ecuacion de espacio de estados

Parámetros

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

rutinas_analisis.**system_creator_tf**(*self*, *numerador*, *denominador*)
Funcion para la creacion del sistema a partir de los coeficientes del numerador y del denominador de la funcion de transferencia

Parámetros

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

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

analisisHandler.**AnalisisHandler**(*self*)
Funcion principal para el manejo de la funcionalida de analisis de sistemas de control, se crean las señales a ejecutar cuando se interactua con los widgets incluyendo las validaciones de entradas

analisisHandler.**analisis_bool_discreto**(*self*)
Funcion para habilitar y deshabilitar el periodo de muestreo

analisisHandler.**analisis_stacked_to_ss**(*self*)
Funcion para cambiar de funcion de transferencia a ecuacion de espacio de estados

analisisHandler.**analisis_stacked_to_tf**(*self*)
Funcion para cambiar de ecuacion de espacio de estados a funcion de transferencia

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

analisisHandler.**ssA_validator**(*self*)
Validacion de la matriz de estados de la ecuacion de espacio de estados

analisisHandler.**ssB_validator**(*self*)
Validacion de la matriz de entrada de la ecuacion de espacio de estados

analisisHandler.**ssC_validator**(*self*)
Validacion de la matriz de salida de la ecuacion de espacio de estados

analisisHandler.**ssD_validator**(*self*)
Validacion de la matriz de transmision directa de la ecuacion de espacio de estados

analisisHandler.**ssdelay_validator**(*self*)
Validacion del delay de la ecuacion de espacio de estados

analisisHandler.**ssperiodo_validator**(*self*)
Validacion del periodo de muestreo de la ecuacion de espacio de estados

analisisHandler.**tfdelay_validator**(*self*)
Validacion del delay de la funcion de transferencia

analisisHandler.**tfdem_validator**(*self*)
Validacion del denominador de la funcion de transferencia

analisisHandler.**tfnum_validator**(*self*)
Validacion del numerador de la funcion de transferencia

analisisHandler.**tfperiodo_validator**(*self*)
Validacion del periodo de muestreo de la funcion de transferencia

2.2 Archivo Handler para la función de entonación de controladores PID

2.2.1 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*)
Funcion para obtener las ganancias del controlador PID a partir de los parametros del modelo de primer orden obtenidos de una respuesta escalon, las formulas son las dadas por Ziegler-Nichols y Cohen-Coon para una respuesta escalon 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*) – Metodo a utilizar

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

Funcion para obtener los parametros del modelo de primer orden de un sistema a partir de su respuesta escalon

Parámetros

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

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

Funcion para obtener la respuesta escalon del sistema en lazo cerrado en combinacion 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

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

Funcion 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, optional*) – Ganancia proporcional, defaults to 0
- **ki** (*float, optional*) – Ganancia integral, defaults to 0
- **kd** (*float, optional*) – Ganancia derivativa, defaults to 0
- **autotuning** (*bool, optional*) – Bandera para señar si es o no una operacion con auto tunning, defaults to False

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

Funcion para la creacion del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmision directa la ecuacion de espacio de estados

Parámetros

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

- **D** (*list*) – Matriz de transmision directa

rutinas_PID.**system_creator_ss_tuning**(*self*, *A*, *B*, *C*, *D*)
Funcion para la creacion del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmision directa la ecuacion de espacio de estados, adicionalmente se realiza el auto tuning utilizando el metodo escojido por le usuario

Parámetros

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

rutinas_PID.**system_creator_tf**(*self*, *numerador*, *denominador*)
Funcion para la creacion del sistema a partir de los coeficientes del numerador y del denominador de la funcion de transferencia

Parámetros

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

rutinas_PID.**system_creator_tf_tuning**(*self*, *numerador*, *denominador*)
Funcion para la creacion del sistema a partir de los coeficientes del numerador y del denominador de la funcion de transferencia, adicionalmente se realiza el auto tuning utilizando el metodo escojido por le usuario

Parámetros

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

2.2.2 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 identificacion de modelo y tunning con csv

rutinas_CSV.**actualizar_Datos**(*self*, *Kc*, *t0*, *t1*, *t2*, *kp*, *ki*, *kd*)
Funcion para mostrar los resultados obtenidos del modelo en un TextEdit

Parámetros

- **Kc** (*float*) – Ganancia del proceso
- **t0** (*float*) – Tiempo del inicio del escalon
- **t1** (*float*) – Tiempo del inicio de la respuesta del proceso ante el escalon
- **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*)

Funcion para obtener las ganancias del controlador PID a partir de los parametros del modelo de primer orden obtenidos de una respuesta escalon, las formulas son las dadas por Ziegler-Nichols y Cohen-Coon para una respuesta escalon 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*) – Metodo a utilizar

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

Fucion 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

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

Funcion para recalcular el controlador PID a partir de los datos del modelo de primer orden con el nueto tiempo t1, ademas, se grafica la data del csv junto con algunos parametros de la identificacion 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 escalon
- **t1** (*float*) – Tiempo del inicio de la respuesta del proceso ante el escalon
- **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 identificacion
- **y1** (*float*) – Punto y1 de la recta de identifiacion, 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*)

Funcion para calcular el controlador PID a partir de los datos del modelo de primer orden, ademas, se grafica la data del csv junto con algunos parametros de la identificacion 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 identifiacion, en este punto se encuentra el mayor cambio respecto al tiempo
- **y2** (*float*) – Punto y2 de la recta de identificacion
- **t0** (*float*) – Tiempo del inicio del escalon
- **t1** (*float*) – Tiempo del inicio de la respuesta del proceso ante el escalon
- **t2** (*float*) – Tiempo en el que el proceso alcanza el 63 % de su valor final respecto al cambio

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

Funcion 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 transformacion 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

Archivo para el manejo de la funcion de Tunning, sirve de intermediario entre la interfaz grafica y las rutinas de entonacion de controladores PID y la identificacion de modelos a partir de un archivo CSV y entonacion de PID para el mismo

TuningHandler.**LEFC_validator**(*self*)

Validacion del limite inferior del span de EFC

TuningHandler.**LVP_validator**(*self*)

Validacion del limite inferior del span de VP

TuningHandler.**PID_bool_discreto**(*self*)

Funcion para habilitar y deshabilitar el periodo de muestreo

TuningHandler.**PID_stacked_to_csv**(*self*)

Funcion para cambiar a csv

TuningHandler.**PID_stacked_to_ss**(*self*)

Funcion para cambiar a ecuacion de espacio de estados

TuningHandler.**PID_stacked_to_tf**(*self*)

Funcion para cambiar a funcion de transferencia

TuningHandler.**TuningHandler**(*self*)

Funcion principal para el manejo de la funcionalida de Tunning, se crean las señales a ejecutar cuando se interactua con los widgets incluyendo las validaciones de entradas

TuningHandler.**UEFC_validator**(*self*)

Validacion del limite superior del span de EFC

TuningHandler.**UVP_validator**(*self*)

Validacion del limite superior del span de VP

TuningHandler.**actualizar_sliders_ss**(*self*)

Funcion para ajustar la resolucion de los sliders con ecuacion de espacio de estados

TuningHandler.**actualizar_sliders_tf**(*self*)
Funcion para ajustar la resolucion de los sliders con funcion de transferencia

TuningHandler.**ajustar_atraso_manual**(*self*)
Funcion para ajutar el tiempo t1, despues de realizar el calculo para un archivo csv, se utiliza en caso de que la estimacion automatica no sea lo suficientemente buena

TuningHandler.**calcular_PID**(*self*)
Funcion para realizar el los calculos necesarios para la funcionalidad de entonacion de controaldores PID, el llamado a esta funcion se realizar por medio del boton calcular o cada vez que se modifique alguno de los sliders

TuningHandler.**calcular_autotuning**(*self*)
Funcion para realizar el los calculos necesarios para la funcionalidad de entonacion de controaldores PID con auto tunning, el llamado a esta funcion se realizar por medio del boton calcular si previamente se habilito la funcionalidad de auto tunning

TuningHandler.**calcular_csv**(*self*)
Funcion para realizar el los calculos necesarios para la funcionalidad de identificacion de modelos y entonacion de controlador PID, el llamado a esta funcion se realizar por medio del boton calcular

TuningHandler.**chequeo_de_accion**(*self*)
Para discriminar entre entonacion con funcion de transferencia, ecuacion de espacio de estados o identificacion de modelo con archivo csv

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

TuningHandler.**ssA_validator**(*self*)
Validacion de la matriz de estados de la ecuacion de espacio de estados

TuningHandler.**ssB_validator**(*self*)
Validacion de la matriz de entrada de la ecuacion de espacio de estados

TuningHandler.**ssC_validator**(*self*)
Validacion de la matriz de salida de la ecuacion de espacio de estados

TuningHandler.**ssD_validator**(*self*)
Validacion de la matriz de transmision directa de la ecuacion de espacio de estados

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

TuningHandler.**ssdelay_validator**(*self*)
Validacion del delay de la ecuacion de espacio de estados

TuningHandler.**ssperiodo_validator**(*self*)
Validacion del periodo de muestreo de la ecuacion de espacio de estados

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

TuningHandler.**tfdelay_validator**(*self*)
Validacion del delay de la funcion de transferencia

TuningHandler.**tfdem_validator**(*self*)
Validacion del denominador de la funcion de transferencia

TuningHandler.**tfnum_validator**(*self*)
Validacion del numerador de la funcion de transferencia

TuningHandler.**tfperiodo_validator**(*self*)
Validacion del periodo de muestreo de la funcion de transferencia

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

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

Parámetros

- **kp** (*float, optional*) – Ganancia proporcional, defaults to 0
- **ki** (*float, optional*) – Ganancia integral, defaults to 0
- **kd** (*float, optional*) – Ganancia derivativa, defaults to 0
- **autotuning** (*bool, optional*) – Bandera para señar si es o no una operacion con auto tunning, defaults to False
- **resolution** (*int, optional*) – Resolucion de los sliders, defaults to 50

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

2.3 Archivo Handler para la función de lógica difusa

2.3.1 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, optional*) – Lista de variables de entrada, defaults to None
- **OutputList** (*list, optional*) – Lista de variables de entrada, defaults to None
- **RuleEtiquetas** (*list, optional*) – 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

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

get_system()
Funcion tomada de yapflm (Yet Another Python Fuzzy Logic Module)

get_var(*vartype*, *varnum*, *start_line*, *end_line*)
Funcion 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 programatica 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 envie 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, optional*) – Lista de reglas, defaults to []

agregar_regla(*window*, *Etiquetasin*, *Etiquetasout*, *logica*)
Funcion 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
- **logica** (*bool*) – Logica a utilizar

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

Parámetros

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

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

Parámetros

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

cambiar_nombre_input(*window*, *i*, *nombre*)
Funcio 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*)

Funcio 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*, *logica*)

Funcion 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
- **logica** (*bool*) – Logica a utilizar

cambio_etinombre_input (*window*, *inputlist*, *i*, *n*, *old_name*)

Funcio 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*)

Funcio 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*)

Funcion 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*)
Funcion 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()
Funcion para crear el controlador difuso a partir de todas las reglas creadas

crear_etiquetas_input(*inputlist*)
Funcion 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*)
Funcion 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*)
Funcion para crear las variables de entrada a partir de la lista de variables de entrada

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

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

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

crear_plots_in(*window*, *ni*)
Funcion para crear los objetos de graficacion de PyQtGraph de la entrada, el codigo para la obtencion de los valores de salida y el graficado es una version altamente modificada de la funcion .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*)
Funcion para crear los objetos de graficacion de PyQtGraph de la salida, el codigo para la obtencion de los valores de salida y el graficado es una version altamente modificada de la funcion .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*)
Funcion para crear las reglas a partir de una lista que contiene toda la informacion necesaria, esta lista es creada en FuzzyHandler.py:

Cada posicion en la lista contiene un set de entradas, salidas y la logica a utilizar (AND o OR), a su vez, cada set es una lista que posee en cada posicion 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 informacion necesaria para crear las reglas

eliminar_regla(*index_rule*)
Funcion para eliminar una regla

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

graficar_mf_in(*window*, *i*)
Funcion para graficar las funciones de membresia de una entrada

Parámetros

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

graficar_mf_out(*window*, *o*)
Funcion para graficar las funciones de membresia de una salida

Parámetros

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

graficar_prueba_pyqtgraph(*window*, *ni*, *no*)
Funcion para actualizar la grafica en funcion de las nuevas entradas, codigo tomado y modificado de la funcion .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*)
Funcion 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*)
Funcion 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*)
Funcion 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*)
Funcion para actualizar la definicion de una funcion de membresia 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*)
Funcion para actualizar la definicion de una funcion de membresia 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*)
Funcion 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*)
Funcion 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

2.3.2 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

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

Parámetros

- **_** (*list*) – Definicion
- **mf** (*str*) – Nombre de la funcion de membresia a validar

Archivo para el manejo de la funcion 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*)
Funcion para crear etiquetas genericas

Parámetros

- **j** (*int*) – Numero de etiqueta
- **erange** (*list*) – Definicio de la funcion de membresia

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

FuzzyHandler.**actualizar_RulesEtiquetas_in**(*self*, *ni*, *new_name*, *old_name*)
Funcion 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*)
Funcion 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*)
Funcion manejar el cargado de controaldores previamente diseñados, se aceptan formatos .JSON y .FIS

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

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

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

FuzzyHandler.**crear_controlador**(*self*)
Funcion para crear el controlador a partir de toda la informacion recolectada, esta creacion 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*)
Funcion para iniciar el entorno de diseño para entradas y salidas genericas

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

FuzzyHandler.**definicion_in**(*self*)
Funcion para manejar el cambio de definicion de la funcion de membresia correspondiente a la etiqueta actual

FuzzyHandler.**definicion_out**(*self*)
Funcion para manejar el cambio de definicion de la funcion de membresia correspondiente a la etiqueta actual

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

FuzzyHandler.**deinificion_in_validator**(*self*)
Funcion para validar las definiciones de las funciones de membresia

FuzzyHandler.**deinificion_out_validator**(*self*)
Funcion para validar las definiciones de las funciones de membresia

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

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

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

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

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

FuzzyHandler.**inputDic_creator**(*self*, *i*)
Funcion para crear entradas genericas

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

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

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

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

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

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

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

FuzzyHandler.**outputDic_creator**(*self*, *i*)
Funcion para crear salidas genericas

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

FuzzyHandler.**prueba_input**(*self*)
Funcion para la ejecucion del codigo correspondiente a la prueba del controlador

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

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

FuzzyHandler.**round_list**(*lista*)
Funcion para redondear los digitos de una lista

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

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

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

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

FuzzyHandler.**seleccion_entrada**(*self*)
Funcion para desplegar la informacion de la entrada seleccionada

FuzzyHandler.**seleccion_etiqueta_in**(*self*)
Funcion para desplegar la informacion de la etiqueta seleccionada de la entrada actual

FuzzyHandler.**seleccion_etiqueta_out**(*self*)
Funcion para desplegar la informacion de la etiqueta seleccionada de la salida actual

FuzzyHandler.**seleccion_mf_in**(*self*)
Funcion para manejar el cambio de funcion de membresia para la etiqueta seleccionada

FuzzyHandler.**seleccion_mf_out**(*self*)
Funcion para manejar el cambio de funcion de membresia para la etiqueta seleccionada

FuzzyHandler.**seleccion_salida**(*self*)
Funcion para desplegar la informacion de la salida seleccionada

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

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

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

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

Archivo que contiene la clase SimpleThread la cual ejecuta la simulacion de sistemas de control en hilo diferente al principal, esto se realiza de esta forma debido a que la simulacion 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 simulacion 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*) – Funcion a la que regresa una vez terminada la simulacion, plot_final_results de simulacionHandler.py
- **update_bar** (*function*) – Funcion para actualizar la barra de progreso, update_progresBar_function de simulacionHandler.py
- **error_gui** (*function*) – Funcion para mostrar los errores ocurridos durante la simulacion, error_gui de simulacionHandler.py
- **list_info** (*list*) – Lista con toda la informacion necesaria
- **parent** (*NoneType, optional*) – Sin efecto, defaults to None

run()

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

run_fuzzy()

Funcion para realizar la simulacion de sistemas de control de esquemas difusos

run_pid()

Funcion para realizar la simulacion de sistemas de control con controlador PID clasico

stop()

Funcion para detener el hilo

rutinas_simulacion.**controlador_validator**(*self*, *esquema*, *InputList*, *OutputList*, *RuleEtiquetas*)

Funcion 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*)

Funcion para la creacion del sistema a partir de la matriz de estado, matriz de entrada, matriz de salida y la matriz de transmision directa la ecuacion de espacio de estados

Parámetros

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

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

Funcion para la creacion del sistema a partir de los coeficientes del numerador y del denominador de la funcion de transferencia

Parámetros

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

2.4.2 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

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

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

Archivo para compilar los Runge-kutta explicitos y embebidos utilizando numba, los metodos quedan guardados en el archivo: metodos_RK.cp37-win32.pyd y pueden ser importados desde el archivo como una funcion de un modulo

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

Runge-Kutta con preservado de estabilidad fuerte de orden 3, en el metodo se asumio 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 transmision 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 integracion se continua con la salida de orden 3, en el metodo se asumio 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 transmision 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 integracion se continua con la salida de orden 4, en el metodo se asumio 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 transmision 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 integracion se continua con la salida de orden 5, en el metodo se asumio 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 transmision 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 integracion se continua con la salida de orden 4, en el metodo se asumio 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 transmision 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 metodo se asumio 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 transmision 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

rk_generator.**ralston3**(*A*, *B*, *C*, *D*, *x*, *h*, *inputValue*)

Runge-Kutta Ralston de orden 3, en el metodo se asumio 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 transmision 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 minimo error de truncamiento de orden 4, en el metodo se asumio 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 transmision 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 metodo se asumio 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 transmision 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 metodo se asumio 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 transmision 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 metodo se asumio 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 transmision 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 metodo se asumio 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 transmision 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 metodo se asumio 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 transmision directa
- **x** (*float64, 2d, C*) – Vector de estado
- **h** (*float64*) – Tamaño de paso
- **inputValue** (*float64*) – Valor de entrada al sistema

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

Archivo para compilar las funciones encargadas de la simulacion 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 funcion de un modulo

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

Funcion 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

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

Funcion 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

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

Funcion 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

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

simulacionHandler.**N_validator**(*self*)

Validacion del valor N

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

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

simulacionHandler.**accionadordem_validator**(*self*)
Validacion del denominador de la funcion de transferencia correspondiente al accionador

simulacionHandler.**accionadornum_validator**(*self*)
Validacion del numerador de la funcion de transferencia correspondiente al accionador

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

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

simulacionHandler.**configuration_data**(*self*)
Funcion para cambiar la configuracion del solver a utilizar

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

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

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

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

simulacionHandler.**get_pathcontroller1**(*self*)
Funcion para obtener la direccion al archivo del controlador difuso

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

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

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

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

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

simulacionHandler.**maxstep_validator**(*self*)
Validacion del incremento maximo de paso

simulacionHandler.**minstep_validator**(*self*)
Validacion del decremento minimo de paso

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

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

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

simulacionHandler.**restablecer_configuracion**(*self*)
Funcion para restablecer la configuracion avanzada por defecto

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

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

simulacionHandler.**sensordem_validator**(*self*)
Validacion del denominador de la funcion de transferencia correspondiente al sensor

simulacionHandler.**sensornum_validator**(*self*)
Validacion del numerador de la funcion de transferencia correspondiente al sensor

simulacionHandler.**simulacion_stacked_to_ss**(*self*)
Funcion para cambiar de funcion de transferencia a ecuacion de espacio de estados

simulacionHandler.**simulacion_stacked_to_tf**(*self*)
Funcion para cambiar de ecuacion de espacio de estados a funcion de transferencia

simulacionHandler.**ssA_validator**(*self*)
Validacion de la matriz de estados de la ecuacion de espacio de estados

simulacionHandler.**ssB_validator**(*self*)
Validacion de la matriz de entrada de la ecuacion de espacio de estados

simulacionHandler.**ssC_validator**(*self*)
Validacion de la matriz de salida de la ecuacion de espacio de estados

simulacionHandler.**ssD_validator**(*self*)
Validacion de la matriz de transmision directa de la ecuacion de espacio de estados

simulacionHandler.**ssdelay_validator**(*self*)
Validacion del delay de la ecuacion de espacio de estados

simulacionHandler.**ssperiodo_validator**(*self*)
Validacion del periodo de muestreo de la ecuacion de espacio de estados

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

simulacionHandler.**tfdelay_validator**(*self*)
Validacion del delay de la funcion de transferencia

simulacionHandler.**tfdem_validator**(*self*)
Validacion del denominador de la funcion de transferencia

simulacionHandler.**tfnum_validator**(*self*)
Validacion del numerador de la funcion de transferencia

simulacionHandler.**tfperiodo_validator**(*self*)
Validacion del periodo de muestreo de la funcion de transferencia

simulacionHandler.**tiempo_validator**(*self*)
Validacion del tiempo de simulacion

simulacionHandler.**update_progresBar_function**(*self*, *value*)
Funcion para actualizar la barra de progreso de la simulacion, esta funcion es utilizada por el QThread

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

Archivo principal, en orden de ejecutar la aplicacion, 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 definicion de toda la interfaz grafica. Desde aca se ejecutan los archivos Handler, quienes representan los enlaces entre las rutinas y la interfaz grafica 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 envia 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, optional*) – Sin efecto, defaults to None

closeEvent(*event*)

Evento pera el cerrado de la ventana

resource_path(*relative_path*)

Funcion para generar direcciones absolutas a partir de direcciones relativas

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

CAPÍTULO 3

Indices y tablas

- genindex
- modindex
- search

Índice de Módulos Python