Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

CONTROL DMC DE UNA PLANTA DE LABORATORIO

Autor:

Antolín Alonso, Saúl

Tutor:

Zamarreño Cosme, Jesús María

Ingeniería de Sistemas y Automática

Valladolid, marzo 2024.

Agradecimientos

A mi familia, por su apoyo incondicional. A mis amigos y compañeros, por confiar en mí y hacer de la universidad un proceso inolvidable.

A la Universidad de Valladolid, por ofrecerme la oportunidad de cursar esta carrera.

A mi tutor, Jesús María Zamarreño, por su inmensa paciencia, disposición y ayuda a lo largo del desarrollo de este proyecto.

Resumen

Se ha desarrollado una interfaz en el desarrollador de aplicaciones *APP DESIGNER* de *MATLAB* que permite al usuario realizar control predictivo DMC de una planta (bien sea simulada o real) o control manual, estableciendo la comunicación con la intrumentación de la planta mediante el protocolo OPC (gracias a la *toolbox* de comunicaciones industriales). La versión actual solo permite el control de sistemas SISO (*Single Input Single Output*).

Como caso de estudio donde validar la aplicación, se ha seleccionado un sistema de dos tanques comunicados por su parte inferior, disponible en el laboratorio de Control de Procesos del departamento de Ingeniería de Sistemas y Automática. Uno de los depósitos recibe un caudal variable gracias a una bomba (su potencia sería la variable manipulada del controlador) y el otro evacúa cierto caudal gracias a una válvula manual de salida (su apertura sería una perturbación no medible). Además, el segundo tanque posee un sensor de nivel, que sería la variable controlada.

La interfaz permitirá al usuario establecer los parámetros del DMC y las variables manipulada y controlada, mostrará en tiempo real las gráficas de las variables deseadas además de exportarlas en formato *MATLAB* (para análisis posteriores). También tendrá la opción de guardar la configuración establecida por el usuario para recuperarla en posteriores sesiones de trabajo.

Palabras clave

Control predictivo DMC, control de procesos, aplicación, comunicaciones OPC, App Designer.

Abstract

An interface has been developed through *MATLAB*'s *APP DESIGNER* application developer that allows the user to perform DMC predictive control of a plant (real or simulated) or manual control. Communication with the plant's instrumentation will be via the OPC protocol (provided by the Industrial Communication Toolbox). The current version only allows control of *SISO* systems (*Single Input Single Output*).

For the validation of the aplication, a system of two tanks connected at the bottom, avaliable in the Process Control laboratory of the Department of Systems Engineering and Automation, was selected. One of the tanks receives a water flow thanks to a pump (its power would be the manipulated variable of the controller) and the other one evacuates a certain flow thanks to a manual outlet valve (its opening would be a non-measurable disturbance). In addiction, the second tank has a level sensor, wich would be the controlled variable.

The interface will allow the user to set the DMC parameters and the manipulated and controlled variables, display in real time the graphs of these variables and export them in *MATLAB* format (for further analysis). It also have the option to save the configuration set by the user for recall in subsequent work sessions.

Keywords

DMC predictive control, process control, application, OPC communications, App Designer.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN Y OBJETIVOS

1.1 INTRODUCCIÓN

La confección de la aplicación en software *MATLAB* permitirá a los usuarios acercarse al control predictivo DMC de una manera muy intuitiva. Gracias a que se realizará a través del protocolo OPC, se ofrece la flexibilidad de que el control se pueda realizar desde casa en un sistema simulado, facilitando al usuario la posibilidad de trabajar con el control en caso de que haya mucha demanda de las plantas reales.

La interfaz procura facilitar la experimentación y comprensión del control predictivo DMC. Además de permitir realizar control manual también, para que el usuario pueda realizar el experimento de identificación correspondiente (ya que al realizarse en *MATLAB* se pueden usar otras aplicaciones ya desarrolladas como *HIDEN*).

A la hora de poner en prueba su funcionamiento se ha optado por realizar el control de una planta del laboratorio de control de nivel de dos tanques comunicantes, de la cual se dispone de un sistema simulado (similar al real) que funciona también a través de un servidor OPC. Dicha planta consta de dos depósitos unidos por una tubería en su parte inferior (Figura 1). Al depósito de la derecha se le suministra un caudal, en este caso de agua, gracias a una bomba que toma esa agua de la bandeja en la que descansan los depósitos. El depósito de la izquierda posee una válvula manual de salida.

Además de estos elementos, que conformarían el funcionamiento básico, la planta consta de un sensor de nivel que nos permite conocer el estado de la planta y realizar así su control. La bomba se regulará a través del ordenador desde donde se ejecutará la interfaz. La válvula manual de salida, también regulable, se comporta como perturbación no medible. El control es capaz de corregir estas perturbaciones.

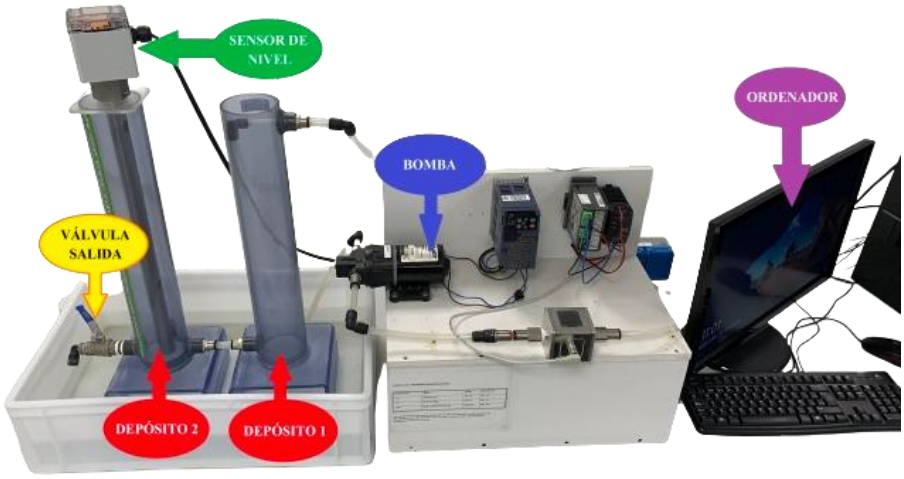


Figura 1: Estructura de la planta del laboratorio.

1.2 OBJETIVOS

Se procura la elaboración de una interfaz creada a través del desarrollador de aplicaciones de *MATLAB*: *App Designer*, que sea capaz de llevar a cabo el control predictivo DMC de un sistema (real o simulado) con comunicación OPC.

En dicha aplicación el usuario podrá:

- **Cargar la configuración y el modelo deseado** del sistema a controlar, bien sea nuevo o previamente manipulado por esta interfaz. La interfaz tiene la opción de guardar los parámetros y variables con los que se ha trabajado.
- **Elegir el servidor y sus señales a controlar**. La interfaz es configurable, pudiéndose aplicar a plantas diversas (simuladas o reales), las cuales a su vez tendrán diferentes señales disponibles para monitorizar.
- **Establecer los parámetros del DMC y monitorizar el comportamiento**. Este es el objetivo principal de la interfaz, permitiendo así que el alumno conozca y experimente un sistema de control DMC.

Además, podemos incluir como objetivo complementario que los alumnos pueden practicar previamente con este control sobre un sistema simulado (ya que es una de las facilidades de trabajar con un servidor OPC). Asimismo, los resultados del experimento se podrán almacenar en un archivo formato *MATLAB* para su posterior manipulación. La aplicación tendrá un manual de usuario disponible para facilitar su operación y una breve descripción en cada elemento interactivo con este mismo fin.

Finalmente, me gustaría destacar que para este proyecto no se van a tener en cuenta las perturbaciones (al menos las medibles), no forman parte del objetivo de la aplicación. El objetivo principal es el desarrollo de la interfaz y comunicaciones para que sea lo más amigable posible, pero para el algoritmo DMC se utilizará un desarrollo previo, válido para sistemas SISO (*Single Input Single Output*).

CAPÍTULO 2: FUNDAMENTOS DEL DMC

Antes de centrarnos en el control DMC, se hace necesario explicar el nacimiento y características de los controles predictivos basados en modelo (MPC, *Model Predictive Control*).

2.1 DESARROLLO HISTÓRICO

Las exigencias de optimizar costes y recursos, mejorar la calidad de producción, seguridad del proceso, respeto al medio ambiente, aumentar la flexibilidad de la planta para que pueda trabajar en un amplio rango de condiciones de operación, etc. requiere una búsqueda de la mejora de los sistemas de control para poder cumplir con estas demandas.

Los controladores PID son reguladores basados en señal, no tienen en cuenta el conocimiento explícito del proceso, están basados en un modelo muy sencillo, pero no usan este conocimiento para nada. Estos sistemas de control, en la industria, solucionan generalmente bien problemas de control monovariables. Sin embargo, no se garantiza un funcionamiento correcto en sistemas con lazos de dinámica compleja (retardos, fase no mínima, …). Además, en sistemas multivariables en los que sus variables interaccionen, estos lazos pueden empeorar seriamente el control y esta calidad de control influye en el rendimiento económico. Como el sistema debe tener en cuenta estas interacciones, es normal utilizar en los procesos de control aquellos controladores que permitan realizar un control multivariable, es decir, capaces de asimilar todas las variables del proceso, obtener un punto de operación aceptable y ser capaz de hacer que dichas variables alcancen el punto de operación deseado.

Siguiendo esta línea, aparecieron los primeros algoritmos que usaban un modelo dinámico del proceso para predecir la evolución futura de las variables de proceso del sistema una vez que eran aplicadas las acciones de control, logrando optimizar estas minimizando el error (sujeto a las restricciones de operación). El modelo de planta se obtenía mediante la respuesta del sistema ante una entrada escalón, que es más sencillo que la función de transferencia. Sobre este tipo de control se desarrolló el DMC (Dynamic Matrix Control), del que hablaremos posteriormente.

2.2 CONTROLES PREDICTIVOS

Un control predictivo MPC es un tipo de estrategia de control basada en el uso explícito de un modelo del proceso a controlar que busca la predicción del futuro comportamiento de la variable a controlar, es decir, actúa en el instante actual basándose en esta predicción del comportamiento [1]. Dentro de este tipo de controles, existen diversos algoritmos como el DMC IDCOM (*Identification-Command*), GPC, PFC.

2.2.1 Conceptos básicos

Entre las principales características comunes a estos tipos de control que forman el control predictivo destacan:

- Maneja problemas de control multivariable.
- Uso de manera explícita del modelo para predecir la salida en el horizonte temporal.
- Cálculo de acciones de control minimizando una función objetivo.
- Horizonte deslizante.

Las diferencias entre los diferentes tipos de control se encuentran en el modelo usado, perturbaciones y función objetivo a optimizar. Estas pequeñas diferencias son las causantes de posibles cambios de funcionamiento en lazo cerrado.

Dentro de las ventajas de los controles predictivos basados en modelo caben destacar:

- Permite trabajar con procesos de dinámica compleja y con restricciones en las variables (manipuladas o controladas): sistemas más complejos.
- Fácil de entender, por lo que no se requeriría de personal altamente cualificado.
- Compensa perturbaciones medibles.
- Abre las puertas a la optimización económica.

A pesar de esto, el MPC también tiene inconvenientes como el aumento de carga computacional (no supone un gran problema con la tecnología actual) y la necesidad de acertar con el modelo apropiado del proceso.

Sus principales elementos son:

- Modelo del proceso para efectuar las predicciones.
- Modelo de perturbación. No compete para este proyecto, no forma parte de los objetivos.
- Función objetivo.
- Método de la ley de control y cálculo de sus señales.

2.2.2 Funcionamiento

Para entender la estrategia que estos controladores siguen, nos ayudaremos de las siguientes imágenes (Figura 2 y Figura 3):

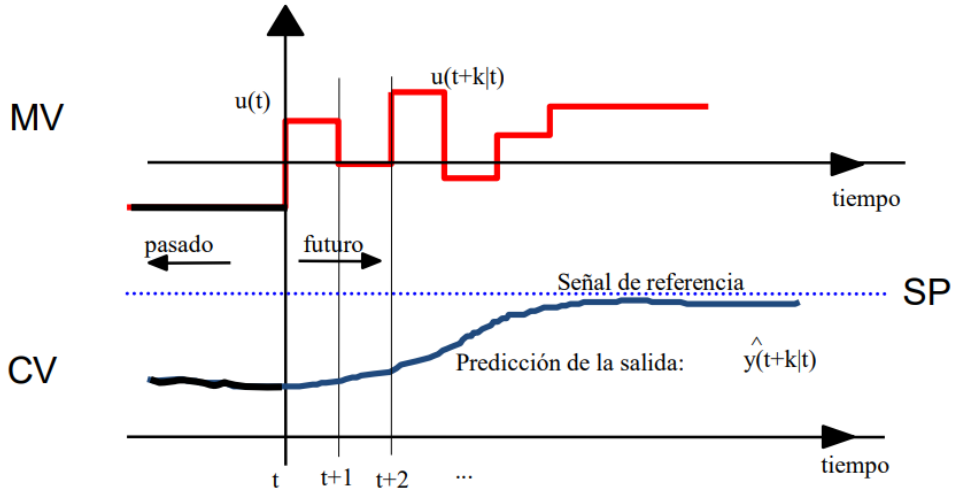


Figura 2: Funcionamiento del control predictivo.

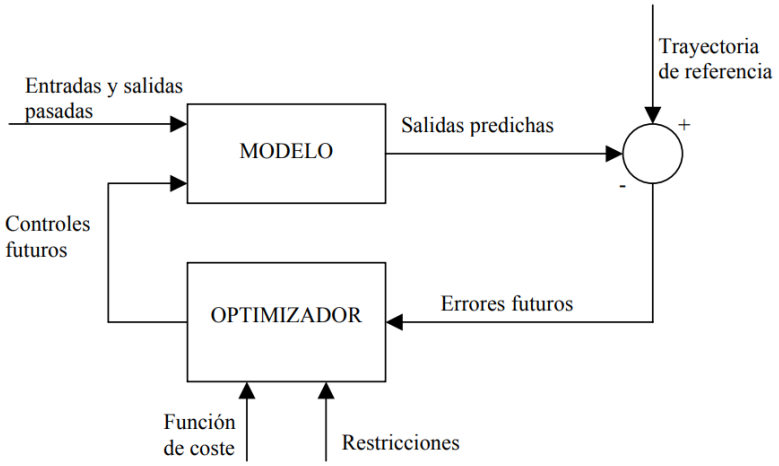


Figura 3: Estructura básica de un MPC.

Donde *MV* es la variable manipulada y *CV* la variable controlada. Los pasos para la aplicación del MPC son los siguientes:

1. En cada instante de tiempo t utilizamos el modelo para predecir la salida para un horizonte de predicción determinado. Estas predicciones son las $\hat{y}(t+k|t)$ que dependen de los valores en este instante y de las acciones futuras $u(t+k|t)$ que tienen que calcularse.
2. Estas señales futuras se calculan de modo que se optimice un criterio determinado manteniendo así el proceso lo más cerca posible de la señal de referencia.
3. Ignoramos todas las señales calculadas menos la que se envía al control $u(t|t)$.
4. Volvemos al paso 1 en el siguiente instante de tiempo y repetimos el procedimiento.

2.2.3 Modelos

Unos elementos importantes en los algoritmos de control predictivo basado en modelos son precisamente los modelos que se usan para las predicciones. Solamente se comentará el modelo de respuesta salto o escalón, que es el que se usa en el DMC.

Se trata de modelos de la forma $y(t) = \sum_{j=1}^{\infty} g_j \Delta u(t-j)$, donde los coeficientes g (coeficientes de la respuesta salto) representan la evolución de la salida al aplicar un salto/escalón en la entrada del proceso, de ahí el nombre de modelo. Estos modelos suelen truncarse de forma que el resultado final queda:

$$y(t) = \sum_{j=1}^{N} g_j \Delta u(t-j) + g_{N+1} u\big(t-(N+1)\big)$$

Este modelo es simple y no requiere conocer la estructura del proceso. Además, puede describir dinámicas no usuales y es poco sensible a errores. Por otro lado, solo se puede aplicar a procesos estables y, en general, contiene un gran número de parámetros, lo que puede dificultar el cálculo.

2.3 DYNAMIC MATRIX CONTROL - DMC

2.3.1 Modelo de predicción

Como se explicó anteriormente, los tipos de control predictivo difieren en el modelo que se va a usar, entre otras cosas. Para el control DMC usamos un modelo de respuesta salto/escalón:

$$y(t) = \sum_{i=1}^{\infty} g_i \cdot \Delta u(t-i) + n(t)$$

donde:

- **t**: instante de tiempo donde nos encontramos.
- **u**: señal de entrada al sistema.
- **y**: señal de salida del sistema.
- **g**: coeficientes del modelo, respuesta al escalón.
- **n(t)**: perturbaciones/ruido que actúa sobre el sistema.

Para realizar los cálculos de las predicciones, hay que desplazarse j pasos hacia el futuro, de tal forma que ahora tendríamos:

$$y(t+j) = \sum_{i=1}^{\infty} g_i \cdot \Delta u(t+j-i) + n(t+j)$$

de forma que $\begin{cases} \text{Si } j-i \geq 0 \rightarrow \text{Futuro.} \\ \text{Si } j-i < 0 \rightarrow \text{Pasado.} \end{cases}$

Si separamos los términos del pasado de los del futuro:

$$y(t+j) = \underbrace{\sum_{i=1}^{j} g_i \cdot \Delta u(t+j-i)}_{\text{Futuro}} + \underbrace{\sum_{i=j+1}^{\infty} g_i \cdot \Delta u(t+j-i) + n(t+j)}_{\text{Pasado}}$$

Supondremos que las perturbaciones futuras son las mismas que las actuales:

$$n(t+j) = n(t) = y_p(t) - \sum_{i=1}^{\infty} g_i \cdot \Delta u(t-i)$$

Una vez tenemos esto, podemos obtener el modelo completo de predicción:

$$\hat{y}(t+j) = \underbrace{\sum_{i=1}^{j} g_i \cdot \Delta u(t+j-i)}_{\text{Futuro}} + \underbrace{\sum_{i=j+1}^{\infty} g_i \cdot \Delta u(t+j-i) + y_p(t) - \sum_{i=1}^{\infty} g_i \cdot \Delta u(t-i)}_{\text{Pasado}}$$

Centrándonos en las acciones de control pasadas, llegamos a la siguiente expresión:

$$p_j = y_p(t) + \sum_{i=1}^{\infty} (g_{j+i} - g_i) \Delta u(t-i)$$

Para sistemas estables, se llega a un punto donde alcanzaremos el estacionario y el valor de la salida no variará más, a este punto le llamaremos N. Con lo que obtendríamos:

$$p_j = y_p(t) + \sum_{i=1}^{N} (g_{j+i} - g_i) \Delta u(t-i)$$

Volviendo a la expresión de la predicción:

$$\hat{y}(t+j) = \sum_{i=1}^{j} g_i \cdot \Delta u(t+j-i) + y_p(t) + \sum_{i=1}^{N} \left(g_{j+i} - g_i\right) \Delta u(t-i)$$

Para entender este concepto de controles pasados y futuros supondremos que no vamos a modificar el control, es decir, nuestro $\Delta u = 0$; entonces solo nos quedaría la respuesta anterior del sistema (lo que ya se movió en el pasado).

A este resultado se le conoce como **Respuesta Libre** (marcada con un recuadro rojo en la ecuación inferior), obteniendo en la salida una señal que no estaría influida por el control, de ahí su nombre. Al otro término restante, se le conoce como **Respuesta Forzada** (recuadro verde) ya que es la respuesta que nosotros podemos alterar, forzando la salida para obtener la respuesta deseada.

$$\hat{y}(t+j) = \sum_{i=1}^{j} g_i \cdot \Delta u(t+j-i) + y_p(t) + \sum_{i=1}^{N} \left(g_{j+i} - g_i\right) \Delta u(t-i)$$

Si desarrollamos la ecuación para obtener las predicciones ($j > 0$), nos daremos cuenta de que se pueden agrupar por matrices, obteniendo el siguiente resultado:

$$\begin{bmatrix} \hat{y}(t+1) \\ \hat{y}(t+2) \\ \vdots \\ \hat{y}(t+N) \end{bmatrix} = \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_N & g_{N-1} & \dots & g_1 \end{bmatrix} \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+N-1) \end{bmatrix} + \begin{bmatrix} p(t+1) \\ p(t+2) \\ \vdots \\ p(t+N) \end{bmatrix}$$

donde:

- $\hat{\boldsymbol{y}}$: predicciones de la salida.
- **G**: matriz dinámica del ensayo, incluye los coeficientes de la respuesta salto.
- **u**: señales de control.
- **p**: respuesta libre del sistema.

y finalmente obtenemos la expresión matricial del modelo de predicción del DMC.

$$\hat{y} = G \cdot u + p$$

2.3.2 Función Objetivo

Esta función tiene dos objetivos: busca minimizar el error provocado por la diferencia entre la consigna/referencia y la salida del proceso, y minimizar el esfuerzo del control sobre el sistema para gastar menos energía para alcanzar el punto deseado. Su expresión general viene dada de la siguiente forma:

$$J = \sum_{i=1}^{P} [y(t+i) - w(t+i)]^2 + \sum_{i=1}^{N_u} \beta [\Delta u(t+i-1)]^2$$

El primer sumatorio sería el encargado de lograr el primer objetivo y el otro sumatorio el segundo.

En la fórmula cabe destacar los siguientes elementos:

- **P**: horizonte de predicción.
- **Nu**: horizonte de control.
- **y**: salida del proceso
- **w**: referencia.
- **u**: acción de control
- $\boldsymbol{\beta}$: parámetro de ponderación del esfuerzo de control.

Los parámetros del DMC serán explicados más adelante.

Teniendo en cuenta la expresión matricial de la función de predicción, obtenemos la forma matricial:

$$J = \Delta u^T(t)[G^T G + \beta I]\Delta u(t) - 2{e_0}^T G\Delta u(t) + {e_0}^T e_0$$

donde el término e_0 sería la diferencia entre la referencia y la respuesta libre.

Como se ha indicado al inicio del apartado, buscamos minimizar la función objetivo, entonces, si no hay restricciones:

$$\frac{\delta J}{\delta \Delta u} = 0 \;\rightarrow\; 2[G^T G + \beta I] - 2G^T e_0 = 0 \;\rightarrow\; \Delta u(t) = [G^T G + \beta I]^{-1} G^T e_0$$

$$\Delta u(t) = [G^T G + \beta I]^{-1} G^T \left(w - p_j\right)$$

Recordemos que las expresiones anteriores pueden aplicarse de forma fácil a sistemas multivariables. Simplemente tenemos que convertir las variables en vectores que recojan todas las señales, tanto de entrada como salida, obteniendo una matriz G de la forma:

$$G = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1n_u} \\ G_{21} & G_{22} & \cdots & G_{2n_u} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n_y 1} & G_{n_y 2} & \cdots & G_{n_y n_u} \end{bmatrix}$$

donde cada matriz G_{ij} contiene los coeficientes de la i-ésima respuesta escalón ante su entrada j-ésima.

En cuanto a las restricciones en los valores que pueden tomar las variables, señalar que aparecen de forma natural al formular los problemas de control: limitaciones físicas, límites de seguridad, velocidad de cambio, requisitos de calidad… y deben ser añadidas a la formulación del problema de optimización.

Para acabar, resaltar que las predicciones del estado estacionario son no sesgadas y (si la optimización lleva estas predicciones a la referencia) entonces la salida del proceso alcanzará la referencia proporcionando un error estacionario nulo.

2.3.4 Parámetros del control DMC

En este apartado hablaremos de los parámetros del DMC: aquellos factores que modifican las características de control DMC y que por tanto afectan a su comportamiento, obteniendo así una respuesta diferente.

Los parámetros son cuatro:

- Horizonte de predicción (P o N2): horizonte en el tiempo futuro de predicción de la salida para minimizar la función objetivo. Se busca minimizar/eliminar el error entre la referencia y las predicciones durante este horizonte temporal.
  Si el valor es pequeño, solo consideramos errores anteriores al punto donde acaba la referencia, logramos un control rápido, pero más oscilante. En cambio, si es muy grande, no mejora el resultado; a partir de cierto valor ya estaríamos en el valor de referencia y los errores serán prácticamente nulos, se logra un control más estable pero más lento.

- Horizonte de control (Nu): número de acciones de control permitidas. A más número de acciones (más grados de libertad), tendremos mejores soluciones porque el control tendrá más libertad para cambiar su acción, sacrificando la carga computacional.
  Como normalmente los sistemas no suelen tener integradores (orden cero), este valor será 1 como mínimo.

- Coeficiente de supresión de movimiento o Factor de peso (β): importancia del esfuerzo de control en la función objetivo. Penalizará los cambios en la acción de control, por lo que, cuanto mayor sea, el controlador hará menos cambios y pequeños.
  Cuanto mayor sea su valor, se dará más importancia a los cambios que sean menores y no tanto en llegar a la referencia. Se consiguen así sistemas más estables pero el control será más suave y lento.

- Factor de filtrado o suavidad (α): se considera como el coeficiente de un filtro de primer orden de tal forma que se considera una trayectoria de referencia a seguir desde el valor actual de la salida hasta el valor deseado de la referencia; valores próximos a uno implica un acercamiento suave y lento a la referencia.

También existen otras características del modelo de proceso que repercuten en el control DMC, como el horizonte de modelo o el tiempo de muestreo. Pero estas características no se considerarían parámetros de control.

CAPÍTULO 3: *APP DESIGNER*

En este capítulo se explican el aspecto y características más relevantes del desarrollador *App Designer* [2]. Se considera que el lector posee la información y conocimientos suficientes de las herramientas *MATLAB* y *SIMULINK*, por lo que no se abordarán estas dos herramientas. También cabe resaltar que la versión de *MATLAB* en la que se ha trabajado es la *R2022b*.

3.1 INTRODUCCIÓN

Actualmente, MATLAB cuenta con tres métodos para desarrollar aplicaciones. Cada herramienta ofrecería un tipo de operación diferente y unas funcionalidades ligeramente dispares entre ellas. Qué camino escoger se basaría en los objetivos de la interfaz y del modo de trabajo al que esté acostumbrado el usuario. Dichos métodos son:

- **Funciones MATLAB**.

Básicamente consistiría en programar por completo la interfaz a través de funciones, tanto el aspecto visual como el comportamiento (véase Figura 4). Se tendría que crear la figura y configurar los elementos deseados usando programación sin refinamiento (posición, tamaño, propiedades, '*callbacks*'…). Esto supondría una clara desventaja, ya que se invertirían unas cantidades de esfuerzo y trabajo excesivamente altas.

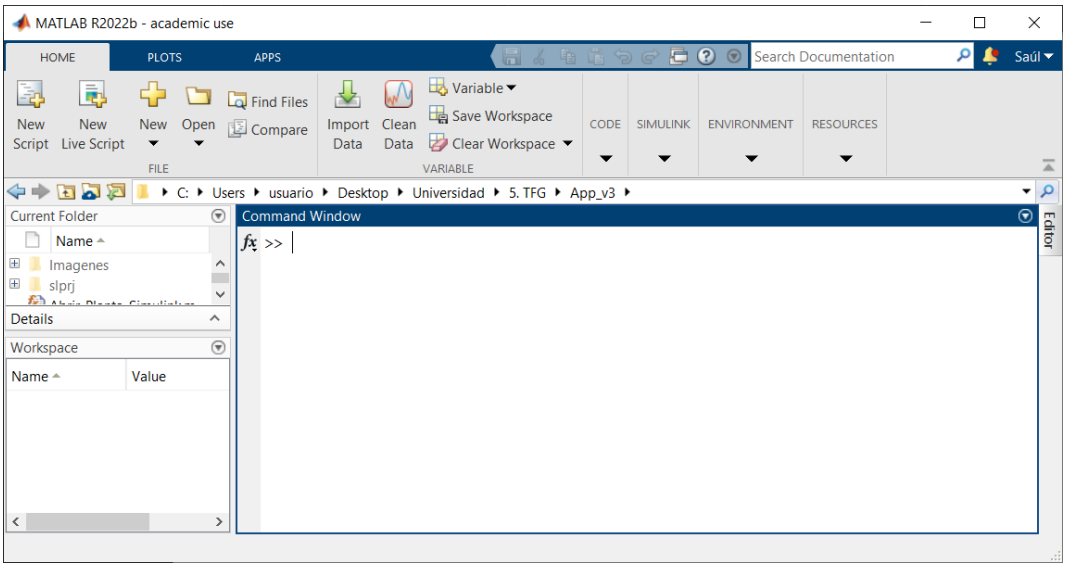


Figura 4: Entorno de trabajo de MATLAB.

- **GUIDE**.

Es una herramienta gráfica para diseñar interfaces (véase Figura 5). En este método tendríamos el diseño y el código de la aplicación en ventanas diferentes. Como ventajas podemos destacar:

- Tiempo de desarrollo reducido gracias a que no es necesario programar ciertos aspectos, por tanto, reducimos código.
- Parte de la interfaz se programa de manera automática como *callbacks* de diferentes elementos u otras funciones. Esto permite que el usuario se centre estrictamente en programar el comportamiento de los componentes.
- La distribución de los componentes se lleva a cabo arrastrando a estos en el entorno de trabajo lo cual nos otorga mayor sencillez para diseñar la interfaz. Además, otras propiedades también se pueden manipular desde el panel de propiedades, reduciendo también el tiempo de programación.

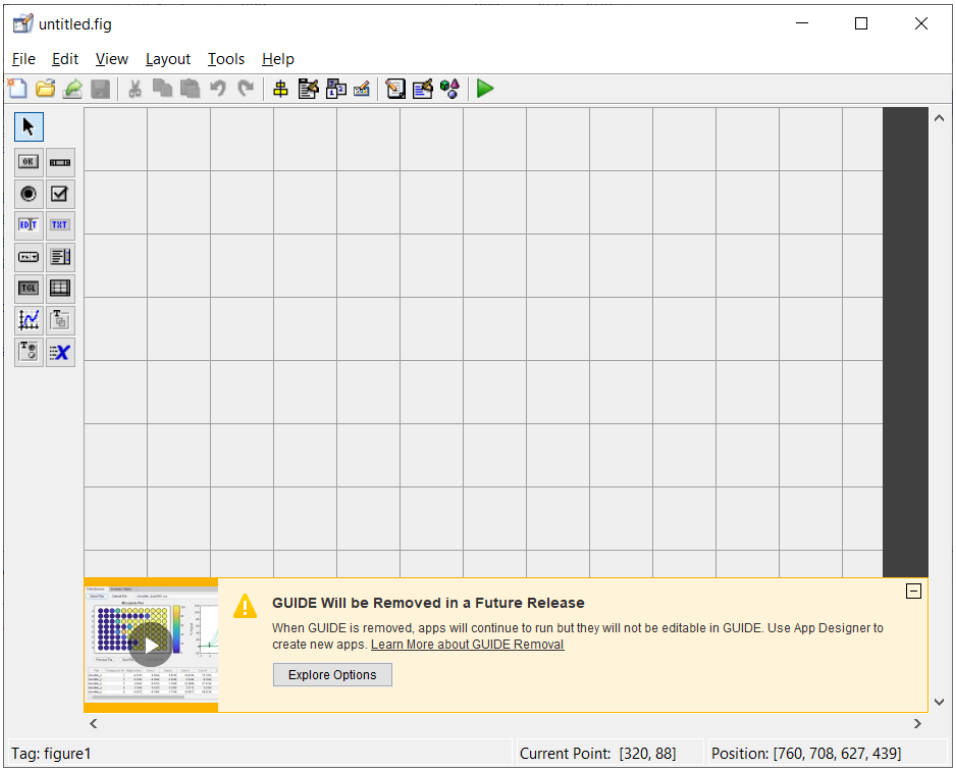


Figura 5: Entorno de la herramienta GUIDE.

Estos dos métodos, en oposición a *App Designer*, trabajan con los mismos tipos de gráficos.

- **APP DESIGNER**.

Se trata de una herramienta que tiene implementada *MATLAB* (desde su versión *R2016a*) que permite la creación de aplicaciones funcionales, pero no desarrolla software profesional. Incluye una versión del editor de *MATLAB* integrada y además permite distribuir las apps empaquetándolas en los archivos del instalador desde el propio *App*

Designer o creando aplicaciones web o de escritorio (para esta última opción se requeriría *MATLAB Compiler*).

Las pestañas de diseño de la interfaz (aspecto visual) y código (comportamiento) están vinculadas y los cambios que se realicen en una de las dos instantáneamente afecta a la otra. Esta herramienta tiene el objetivo de facilitar el diseño de interfaces y reducir el tiempo en su creación. Facilita el acceso y manipulación de las propiedades de los componentes, declaración de *callbacks* o cómo comparten estas la información. Es decir, el desarrollo de código de la interfaz se optimiza y es por eso, entre otras razones, que *GUIDE* va a desaparecer (véase Figura 5). En el apartado 3.6 DIFERENCIAS ENTRE GUIDE Y APP DESIGNER se puede ver las principales diferencias entre ambas herramientas.

En contrapartida, *App Designer* no dispone de todas las funciones gráficas, menús y barras de herramientas que sí tiene *MATLAB*. En cambio, el desarrollador tiene un mayor número de componentes interactivos como veremos más adelante.

3.2 ENTORNO DE TRABAJO

Para iniciar *App Designer* tenemos dos caminos sencillos:

- Escribir "*appdesigner*" en la ventana *Comand Window* de *MATLAB* (recuadro verde de la Figura 6).
- Desde la pestaña *HOME* de *MATLAB* realizamos la ruta: *New* → *App* (recuadros rojos de la Figura 6).

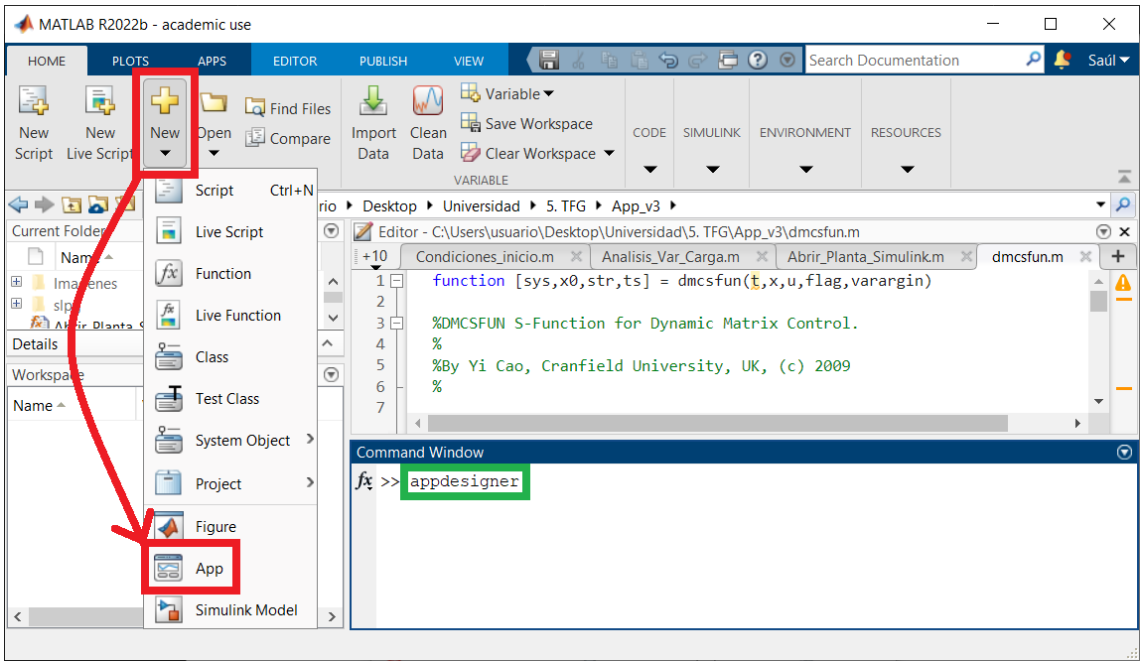


Figura 6: Acceso a *App Designer* desde *MATLAB*.

A continuación, se nos abre la ventana mostrada en la Figura 7, en la cual tenemos las opciones de abrir algún archivo creado anteriormente o una nueva interfaz desde cero.

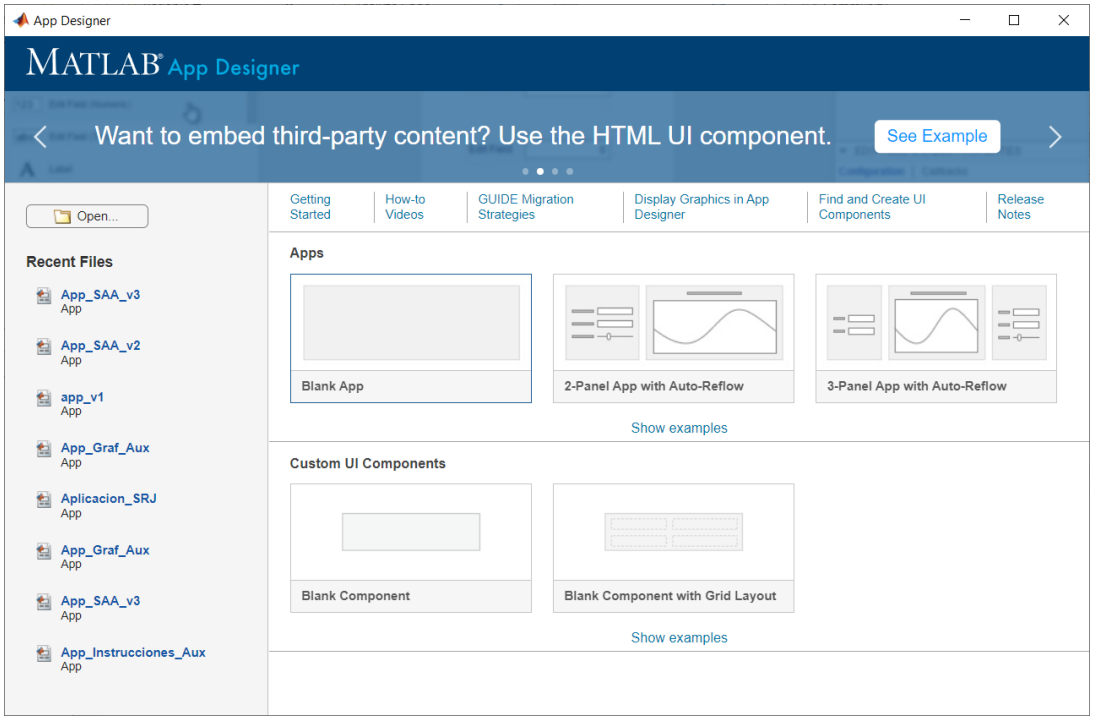


Figura 7: Ventana de inicio de *App Designer*.

Al escoger un nuevo proyecto, se nos abre la ventana de la nueva interfaz en la pestaña *Design View* (diseño) como se ve en la Figura 8. Al entorno de programación se accede en la pestaña *Code View* y tendríamos la vista de la Figura 9.

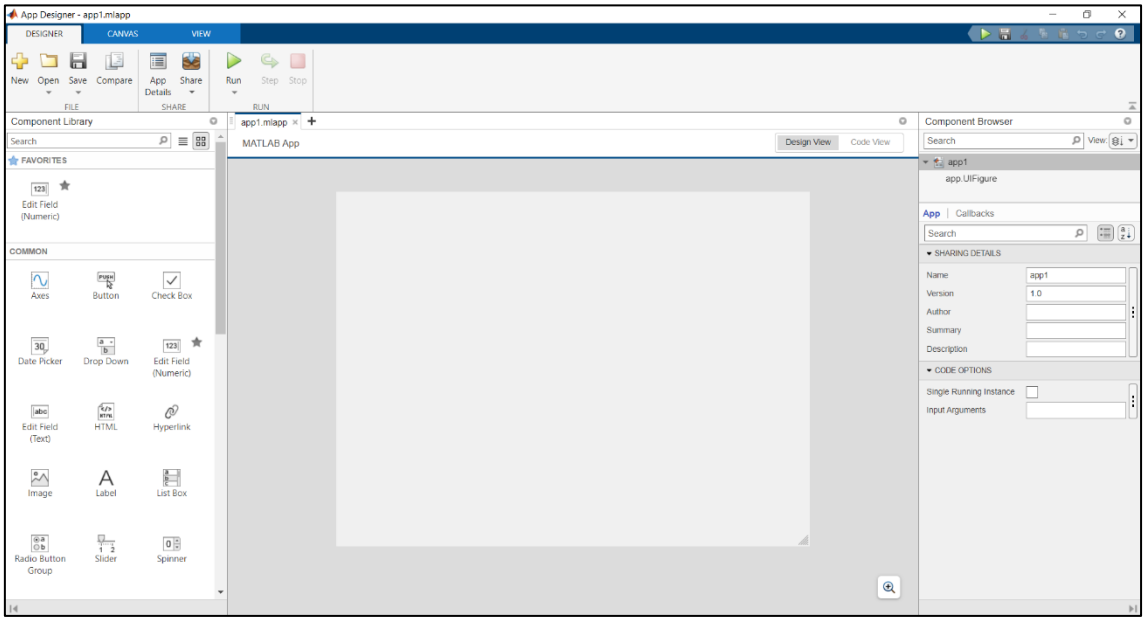


Figura 8: Vista del entorno de trabajo *Desing View*.

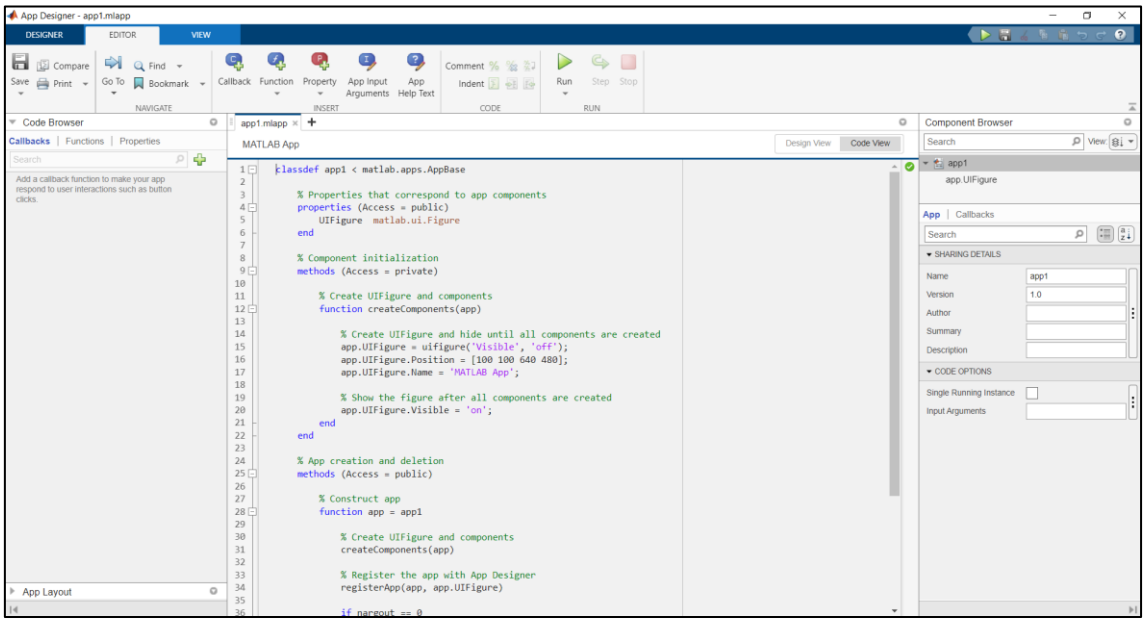


Figura 9: Vista del entorno de trabajo *Code View*.

Para cambiar entre ambas, tenemos sus accesos en la parte superior derecha como se muestra en la Figura 10.

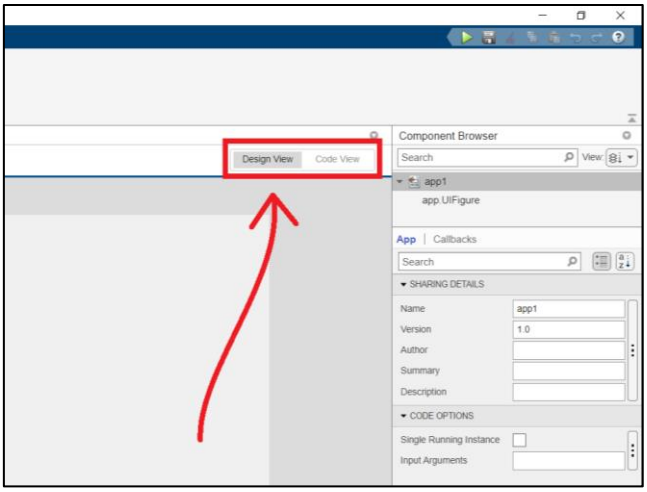


Figura 10: Cambio entre pestañas de diseño y programación.

Como se puede ver, la pestaña en la que nos encontramos tiene distintos paneles y menús asociados. Sin embargo, tenemos dos zonas comunes para ambas pestañas. La primera se trata del menú *DESIGNER* ubicado a la izquierda en la barra de herramientas (Figura 11).

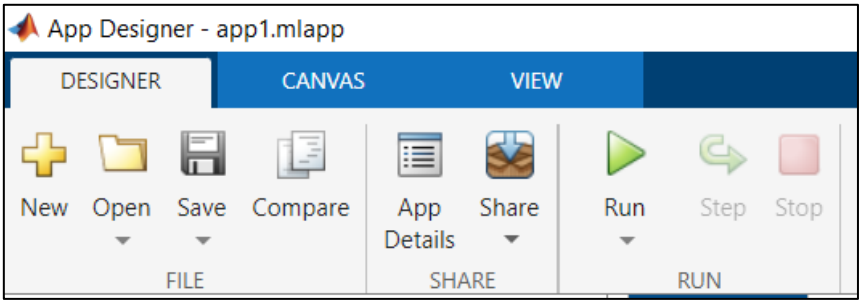


Figura 11: Vista del menú *DESIGNER* abierto.

Cuyas funciones son:

Tabla 1: Funciones del menú *DESIGNER*.

Menú *DESIGNER*	
 New	Crea una nueva aplicación.
 Open	Abre una aplicación existente.
 Save	Guarda la aplicación en su estado actual. También tiene la opción de crear una copia y de exportar la interfaz en formato '*.m*'.
 Compare	Compara la aplicación actual con una existente.
 App Details	Muestra los detalles de la interfaz: nombre, autor, detalles…
 Share	Crea el instalador de la app para *MATLAB*. Si tuviéramos *MATLAB Compiler* crearía el instalador para la aplicación web o escritorio.
 Run	Ejecuta la aplicación. Tiene alternativas como pausar en fallos, errores…
 Step Stop	Ejecutan un paso de la aplicación o la detiene sucesivamente. Estas opciones se habilitan al iniciar la interfaz.

La otra zona común se trata del panel ***Component Browser*** (Figura 12), que forma la lista de los elementos usados en nuestra interfaz y de sus propiedades visuales. Dichas propiedades se pueden manipular y varían según el componente seleccionado.

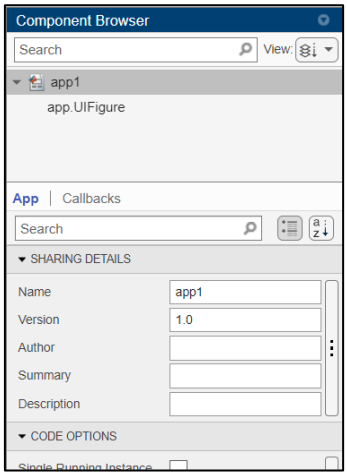


Figura 12: Panel del *Component Browser*.

Dentro de las secciones de cada pestaña tenemos:

A. Paneles de la pestaña *Design View*.

- **Editor**. Ofrece una visión previa del aspecto de la interfaz. Los componentes se disponen en el área de la aplicación para conferir su aspecto. Para mejorar la precisión disponemos de un ajuste de *zoom* en la zona inferior. Podemos también cambiar el área de la interfaz.
- ***Component Library***. Nos muestra los componentes interactivos que podemos usar en nuestra aplicación. Los elementos se clasifican en: *common*, *containers*, *figure tools* y *instrumentación*. Para usarlos basta con arrastrarlos al área de trabajo.

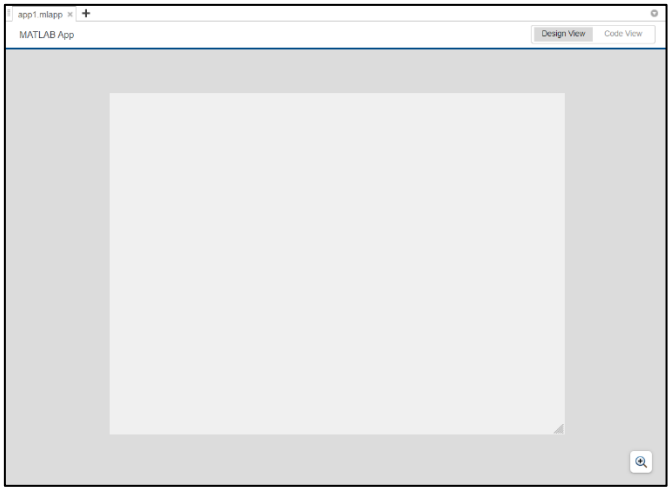


Figura 13: Editor.

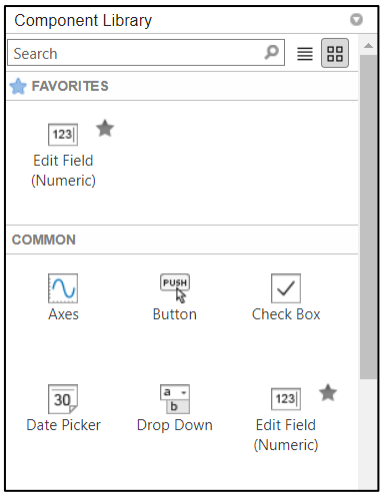


Figura 14: *Component Library*.

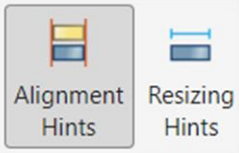
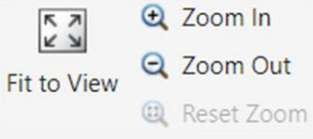
En la barra de herramientas tenemos dos pestañas: *CANVAS* y *VIEW*.

Tabla 2: Funciones del menú *CANVAS*.

Menú *CANVAS*		
Save	Guarda la aplicación en su estado actual. También tiene la opción de crear una copia y de exportar la interfaz en formato '*.m*'.	
Convert	Permite cambiar el formato de la aplicación de simple a varios paneles.	
	Alinea los componentes seleccionados.	
Same Size Grouping Reorder Tab Order	***Same Size***	Iguala el tamaño de los componentes seleccionados.
	Grouping	Agrupa los elementos seleccionados.
	Reorder	Reordena los elementos dentro del *Component Browser*.
	Tab Order	Nos permite ver el orden de colocación de elementos del *Component Browser*.

	Controles de ejecución de la aplicación, iguales que los explicados en la Tabla 1.

Tabla 3: Funciones del menú *VIEW*.

Menú *VIEW*		
	Activa o desactiva la rejilla visual de la zona de trabajo.	
	***Alignment Hints***	Guías para la alineación de objetos.
	Resizing Hints	Guías para la redimensión de elementos.
	Ajusta la vista de la interfaz al gusto del usuario.	

B. Paneles de la pestaña *Code View*.

- **Editor**. Nos muestra el código de la aplicación tanto el generado de forma automática por *App Designer* como el de los *callbacks* y funciones y propiedades implementadas. Véase Figura 15.
- **Code Browser**. Podemos ver la lista de *callbacks*, funciones y propiedades que se han ido creando en la interfaz. Se colocan a medida que se generan, pero el usuario puede reordenarlas a su disposición. Véase Figura 16.
- ***App Layout***. Vista reducida del aspecto de la interfaz. Véase Figura 17.

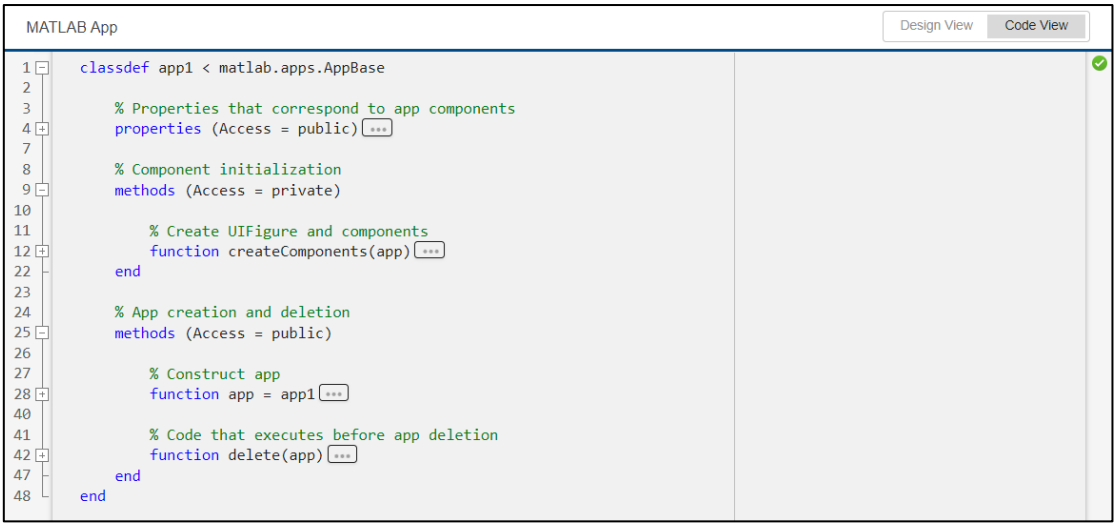


Figura 15: Editor de código.

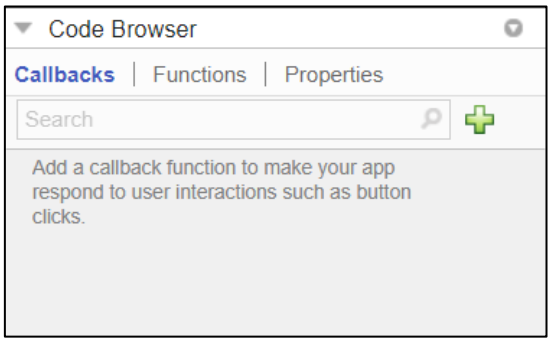


Figura 16: *Code Browser*.

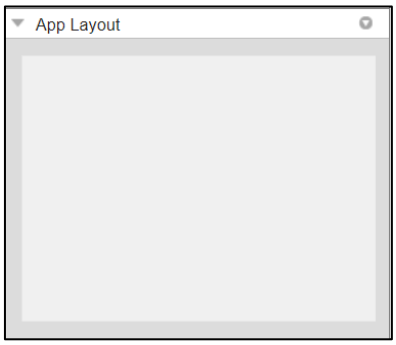


Figura 17: *App Layout*.

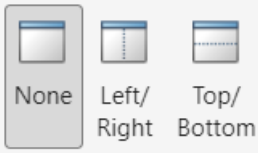
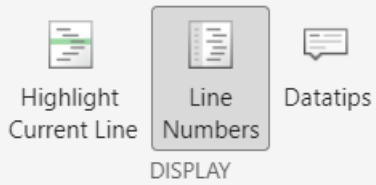
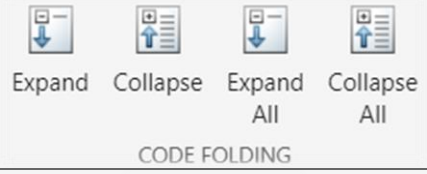
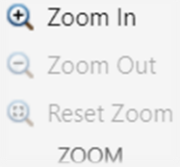
Al igual que para la pestaña de diseño, la barra de herramientas tiene dos pestañas: *EDITOR* y *VIEW*.

Tabla 4: Funciones del menú *EDITOR*.

Menú *EDITOR*		
Save	Guarda la aplicación en su estado actual. También tiene la opción de crear una copia y de exportar la interfaz en formato '*.m*'.	
Compare	Compara la aplicación actual con una existente.	
Print	Imprime el código.	
Go To	Mueve el cursor a la línea seleccionada dentro del documento.	
Find	Busca las palabras seleccionadas dentro del código.	
Bookmark	Añade/quita un marcador en el código.	
Callback	Crea un *callback*.	
Function	Crea una función de utilidad.	
App Input Arguments	Abre la ventana de gestión de argumentos de entrada de la app.	
App Help Text	Permite crear un comentario descriptivo de la interfaz.	
Comment Indent	***Comment***	Permite comentar o no comentar el texto seleccionado.
	Indent	Aplica la sangría deseada al texto seleccionado.

	Controles de ejecución de la aplicación, iguales que los explicados en la Tabla 1.

Tabla 5: Funciones del menú *VIEW*.

Menú *VIEW*		
	***None***	Vista única del código.
	Left/Right	Divide la vista en dos pestañas verticales.
	Top/Bottom	Divide la vista en dos pestañas horizontales.
	***Highlight***	Activar/desactivar el marcador de línea.
	Line Numbers	Activar/desactivar el número de línea.
	Datatips	Activar/desactivar las pistas de datos.
	Expandir/contraer una o todas las *callbacks* del código.	
	Ajuste de tamaño del código.	
	Activa/desactiva las ayudas de *App Designer*.	

3.3 COMPONENTES

El desarrollador dispone de una variedad de elementos de interacción para elaborar sus interfaces. En este apartado se mostrarán los distintos tipos y su función básica, ordenados por categorías como se vio en el apartado anterior (véase 3.2 ENTORNO DE TRABAJO).

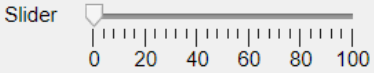
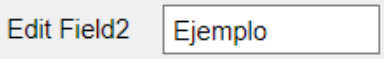
3.3.1 Componentes comunes '*Common*'

Dentro de los elementos comunes, podemos distinguir cada componente según la función que tengan.

A. Elementos para obtención de datos directa.

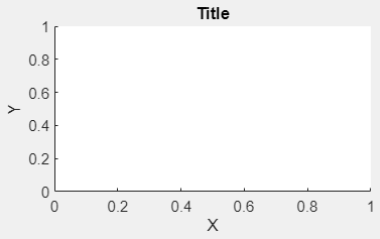
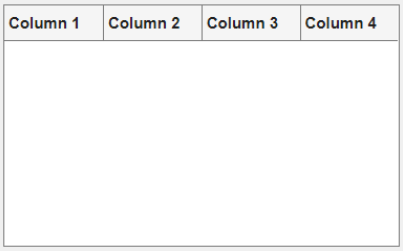
Se trata de los componentes que permiten al usuario introducir datos numéricos o en forma de texto.

Tabla 6: Componentes para introducción de datos.

	Componente	Diseño base	Características
Numéricos	 Edit Field (Numeric)		Permiten: ● Especificar un rango de valores (finito o infinito). Además, si se intenta introducir un valor fuera del margen o no válido, el propio desarrollador incluye un mensaje de error. ● Mostrar los valores en varios tipos de anotaciones (incluyendo redondeos y notación científica). ● Generar y programar *callbacks* que se activen cuando se cambie su valor. ● Ajustar con precisión el valor a introducir.
	 Spinner		
	 Slider		Permite: ● Especificar un rango de valores (finito o infinito). Además, no permite introducir un valor fuera del margen. ● Generar y programar *callbacks* que se activen cuando se cambie su valor. En cambio, no permite ajustar el valor con precisión.
Tipo texto	 Edit Field (Text)		Permiten: ● Especificar un rango de valores (finito o infinito). Además, si se intenta introducir un valor fuera del margen, el propio desarrollador incluye un mensaje de error. ● Especificar el tipo de texto que se está introduciendo: letras, números, caracteres… ● En el caso de Text Area, se pueden introducir más de una línea, en tal caso se mostraría una barra de desplazamiento.
	 Text Area		

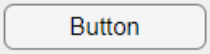
B. Elementos de representación de datos.

Tabla 7: Componentes para representación de datos.

Componente	Diseño base	Características
Axes		● Permite representar datos en una gráfica de 2D o en un diagrama de dispersión. ● No permite representar todas las funciones gráficas de *MATLAB*. Con ayuda de algunas *toolbox* se pueden ampliar las funciones gráficas o incluso representar algunos gráficos 3D.
Table		● Para completar la tabla, es obligatorio asignar valores mediante una matriz o vectores. ● Puede activar un *callback* si el usuario selecciona o manipula cualquiera de las celdas.

C. Elementos de ejecución de acciones.

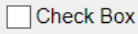
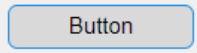
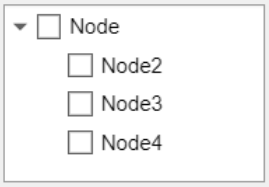
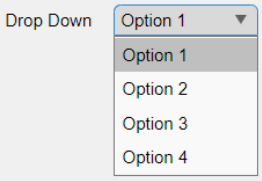
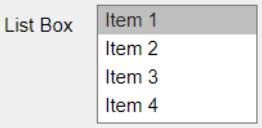
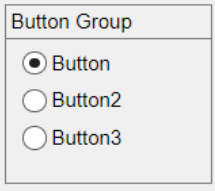
Tabla 8: Componentes de ejecución de acciones.

Componente	Diseño base	Características
Button		● Activa un *callback* al hacer clic sobre ellos. ● Se puede poner una imagen para resumir/describir su función.

D. Elementos de selección de estados excluyentes.

Los siguientes componentes permiten al usuario seleccionar dos estados excluyentes dentro del mismo elemento.

Tabla 9: Componentes de selección de estados excluyentes.

	Componente	Diseño base	Características
Selección de dos estados	Check Box		● Suele usarse si la elección puede describirse con pocas palabras. ● Al seleccionar/borrar la casilla de verificación, podemos activar un *callback*.
	State Button		● Suele usarse si la elección puede describirse con pocas palabras o si queremos usar un icono para su descripción. ● Al cambiar su estado, es decir presionar o liberar, podemos activar un *callback*.
Selección de dos o más estados	Tree (Check Box)		Cada alternativa/opción dentro de un mismo elemento, puede activar un *callback* si así se configura.
	Drop Down		
	List Box		
	Radio Button Group		

	Toggle Button Group	Button Group2 Button Button2 Button3	

E. Otros elementos.

Existen también otros componentes básicos cuyo funcionamiento no es común.

Tabla 10: Otros componentes.

Componente	Diseño base	Características
Image		Permite: ● Cargar una imagen de los archivos locales. ● Cargar un gráfico como el elemento *Axes*.
Label	Label	Se usan para mostrar información que se considere oportuna.
Hyperlink	Hyperlink	Se trata de un vínculo, lleva a una página web deseada.
HTML		Muestra marcas simples o archivos HTML vinculados.

Todos los elementos pueden ser configurados de base, o a través del programa en la pestaña *Code View*. Entre sus propiedades más comunes tenemos:

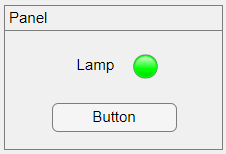
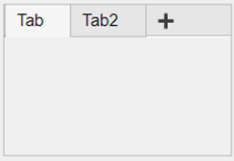
- ***Value***. Valor del elemento sea numérico o tipo texto.
- ***Visible***. Visibilidad del componente.
- ***Enable***. Accesibilidad del componente sin quitar su visión.
- ***Font/Color***. Tipo de fuente y colores del elemento.
- ***Position***. Posición del elemento en la interfaz.

Algunos, como el botón, tienen, como hemos visto, otras propiedades más como ***Icon*** (icono descriptivo), por ejemplo.

3.3.2 Componentes para organización '*Containers*'

Estos elementos se usan para establecer una agrupación de componentes dentro de la interfaz.

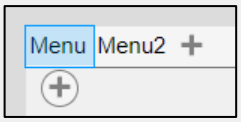
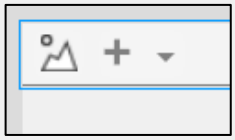
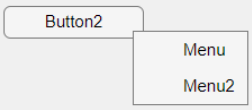
Tabla 11: Componentes de organización de elementos

Componente	Diseño base	Características
Grid Layout		Divide la interfaz en cuadrículas donde agrupar los componentes.
Panel		Crea un panel para agrupar los componentes en su interior.
Tab Group		Divide contenido en su interior en pestañas. Útil cuando no es necesario ver todo el contenido al mismo tiempo.

3.3.3 Componentes para creación de menús '*Figure Tools*'

Estos elementos crean menús, como por ejemplo una barra de herramientas, en la parte superior de la app generalmente.

Tabla 12: Componentes para creación de menús.

Componente	Diseño base	Características
Menu Bar		Añade un menú de textos en la parte superior izquierda de la aplicación. Cada menú/submenú se puede configurar para activar un *callback*.
Toolbar		Igual que el anterior, pero emplea iconos en lugar de textos.
Context Menu		● Crea un despliegue de opciones en donde cada acción puede activar un *callback*. ● Se puede asociar a la propia interfaz o a alguno de sus elementos.

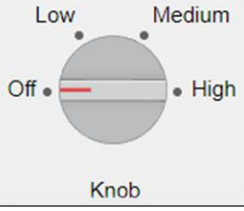
3.3.4 Elementos de señalización '*Instrumentación*'

Los componentes de este grupo nos permiten mostrar o cambiar el valor de ciertas variables o estados.

A. Elementos de selección de valores.

Estos componentes funcionan igual que los vistos en el apartado de elementos comunes (A. Elementos para obtención de datos directa.) pero con un aspecto visual diferente.

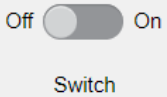
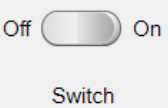
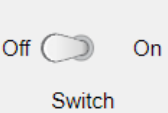
Tabla 13: Tabla de elementos de selección de valor.

Componente	Diseño base	Características
Knob		Permite: ● Especificar un rango de valores (finito o infinito). Además, no permite introducir un valor fuera del margen, pero no permite ajustarse con precisión. ● Generar y programar *callbacks* que se activen cuando se cambie su valor. ● En el caso de la ruleta discreta, los valores son fijos y entradas de texto.
Knob (Discrete)		

B. Elementos de selección de estados excluyentes.

Son componentes de igual funcionamiento que los vistos en elementos comunes (ver D. Elementos de selección de estados excluyentes.) pero con una visual más comprensible.

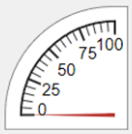
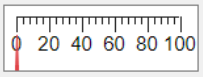
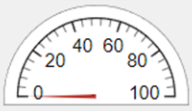
Tabla 14: Tabla de componentes de selección de estados excluyentes.

Componente	Diseño base	Características
Switch		● Permiten activar un *callback* al cambiar su estado. ● Pueden orientarse vertical u horizontalmente según las cuestiones de espacio deseadas. ● Se puede cambiar la nomenclatura de sus estados.
Switch (Rocker)		
Switch (Toggle)		

C. Elementos de señalización de valores.

Estos componentes permiten al usuario representar valores de variables como la velocidad, temperatura, intensidad de un circuito… Poseen un marcador móvil a través del cual se puede ver si los valores son adecuados con un simple vistazo.

Tabla 15: Componentes de señalización de valores.

Componente	Diseño base
Gauge	 Gauge
Gauge (90 Degree)	 Gauge
Gauge (Linear)	 Gauge
Gauge (Semicircular)	 Gauge

D. Elementos de señalización de estados.

Se trata de un componente que varía su color para indicar, según el criterio del programador, una serie de estados. Su color se puede cambiar a través de los *callbacks*.

Tabla 16: Componente de señalización de estados.

Componente	Diseño base
Lamp	Lamp 

Todos los componentes explicados en este apartado tienen las mismas propiedades comunes que los vistos en el apartado 3.3.1 Componentes comunes ‘Common’.

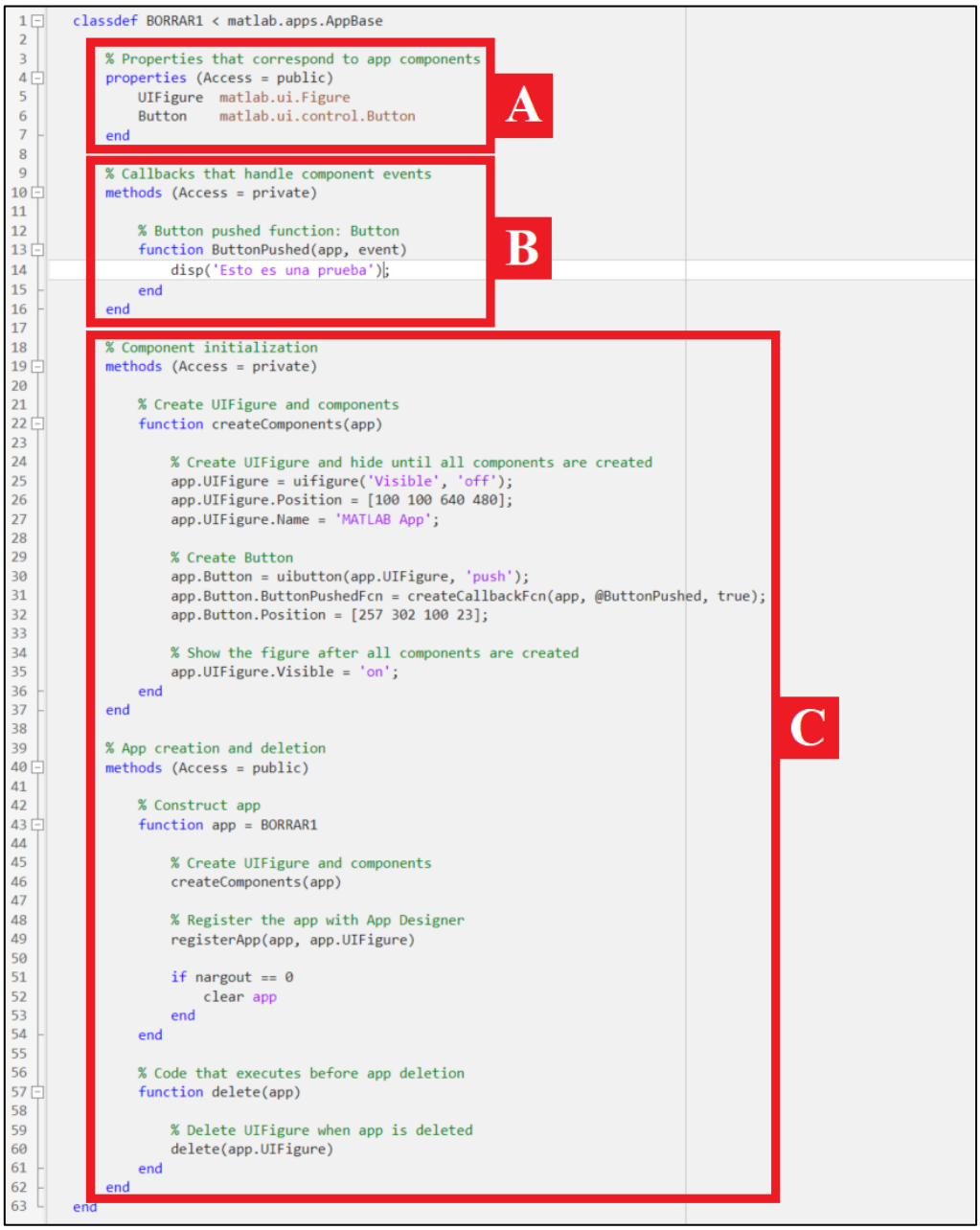
3.4 VENTANA '*CODE VIEW*'

Como ya hemos comentado en apartados anteriores (3.2 ENTORNO DE TRABAJO), la ventana *Code View* nos permite ver y manipular el código de la interfaz para programar el comportamiento. La pestaña está dividida en varios paneles: Editor, *Code Browser*, *App Layout* y *Component Browser*, pero solo nos centraremos en el primero.

El panel Editor contiene todo el código, tanto el generado automáticamente por el desarrollador (aparece sobre un fondo gris y no se puede manipular) como el generado por el programador, que estaría formado por *callbacks*, propiedades y funciones (aparece sobre un fondo blanco y sí es editable). Cualquier cambio en el diseño (como quitar componentes) o agregar *callbacks*, funciones o propiedades automáticamente se reflejan de forma inmediata en el código.

A la hora de estructurar el código, podemos distinguir tres zonas:

A. **Propiedades de la interfaz**. Se corresponde con la primera sección de la Figura 18. Esta división refleja las características de la aplicación y componentes y no es editable. Si el diseñador decide incorporar propiedades (para almacenar y comunicar información entre *callbacks*) o funciones de utilidad locales, *App Designer* las define en esta sección y sí es editable.
B. ***Callbacks***. Se corresponde con la segunda sección de la Figura 18. Todos los *callbacks* generados, tanto el de inicio (*startup*) como los de los componentes, aparecen reflejados en esta sección.
C. **Inicialización y creación de componentes**. Se corresponde con la tercera sección de la Figura 18. Como su nombre indica, esta parte está reservada para las características de los componentes: tamaño, posición, color… y no es editable por el programador. Al manipular los componentes, dichos cambios se reflejan en esta zona.




```
classdef BORRAR1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure  matlab.ui.Figure
        Button    matlab.ui.control.Button
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: Button
        function ButtonPushed(app, event)
            disp('Esto es una prueba');
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
            app.Button.Position = [257 302 100 23];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = BORRAR1

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Figura 18: Estructura del código del interfaz en *App Designer*.

3.4.1 Función '*Start-Up*'

Se trata de la función inicio de la app. Al abrir la aplicación se activa este *callback* y se ejecuta el código en ella. Esta función se usa para llamar a las carpetas, librerías, etc. necesarias, inicializar propiedades de algunos componentes o abrir imágenes en los elementos dispuestos para ello.

Para generarla podemos ir a la ventana *Design View*, clic derecho fuera de la interfaz y pinchar en "*Add StartupFcn callback*". También se puede acceder desde la pestaña *Code View* y en *Code Browser* dar al símbolo '+' (véase Figura 19).

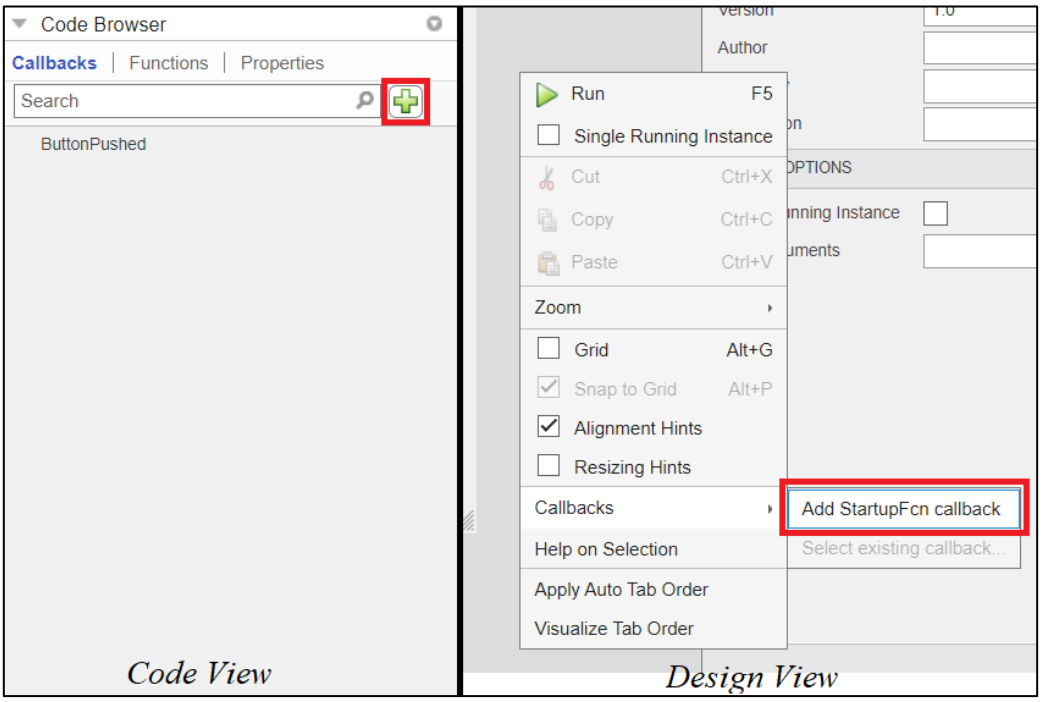


Figura 19: Creación de la función *Start-Up*.

3.4.2 '*Callbacks*'

Un *callback* es una función que se ejecuta al interaccionar con alguno de los elementos presentes en la interfaz, como pulsar un botón o mover una *slider*. Dichas interacciones deben estar definidas por el creador, es decir, el componente debe tener asociado un *callback* para que la función se ejecute.

Para crear uno, basta con hacer clic derecho en el elemento deseado y crear el *callback* determinado para ese objeto (véase Figura 20).

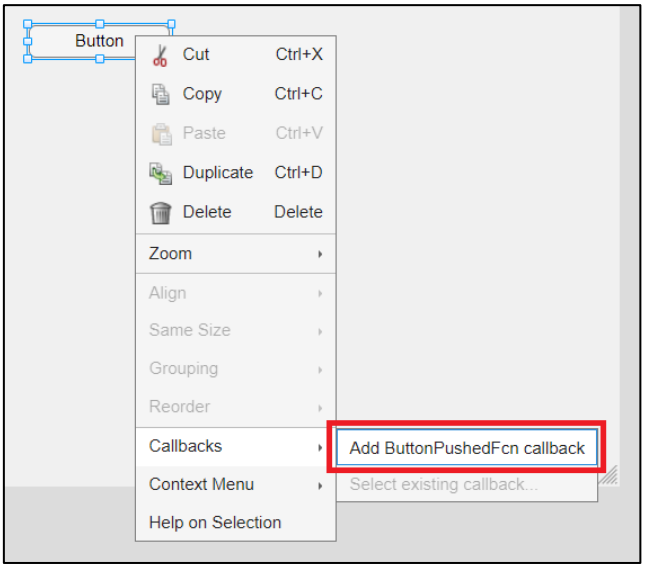


Figura 20: Creación de un *callback*.

Se puede asociar un elemento a un *callback* ya creado, para ello basta hacer clic derecho en el componente y seleccionar "*Select existing callback…*" (ver Figura 21).

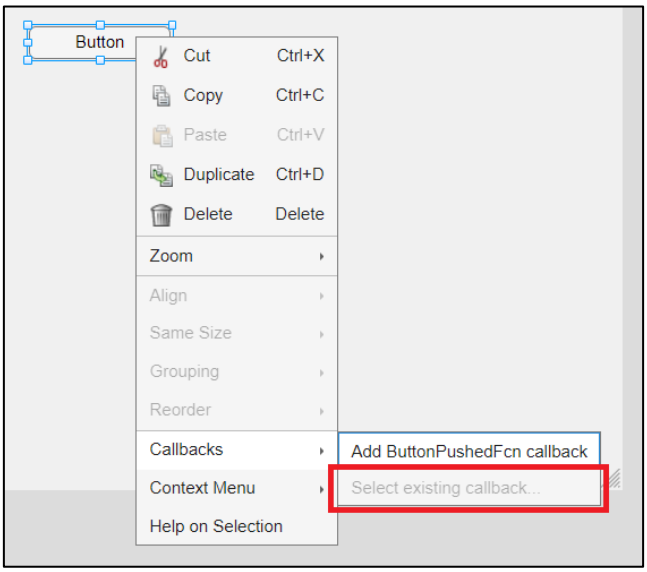


Figura 21: Selección de *callback* existente.

Una vez se cree el *callback*, *App Designer* abre instantáneamente el editor de código con la nueva función creada y coloca el cursor en ella.

3.4.3 Compartir datos entre *callbacks*

Los elementos encargados de almacenar la información para poder compartirla entre *callbacks* son las propiedades. Las propiedades son tipos de datos que forman parte de una estructura: *app*. A esta estructura le vamos añadiendo campos separados por puntos. Esto hace que la variable *app* contenga todas las propiedades y así se permite pasar todas las propiedades de golpe como argumentos de entrada a las propiedades. Por esto, para acceder a estas propiedades, debemos poner el prefijo "*app*" delante de la propiedad del modo: '*app.propiedad*'.

Para crearlas tenemos que dar al símbolo '+' en la pestaña *Propierties* del *Code Browser* como se muestra en la Figura 22.

Figura 22: Crear propiedades.

Podemos distinguir dos tipos de propiedades:

- **Propiedades privadas**. Son exclusivas de la interfaz, es decir, su uso solo es permitido por la aplicación (el *workspace* de *MATLAB* no tiene acceso a este tipo de propiedades). Recuadro rojo de la Figura 23.
- **Propiedades públicas**. Se pueden usar dentro y fuera de la interfaz. Su uso está pensado para que el espacio de trabajo de *MATLAB* pueda acceder a ellas mientras funciona la aplicación. Recuadro verde de la Figura 23.

Además de estas, *App Designer* genera propiedades que controlan el panel y el comportamiento de los objetos que no son accesibles por el usuario, pero sí puede acceder a ellas y modificar sus valores. Por ejemplo, cuando introducimos una lámpara en el editor de la pestaña *Design View*, *App Designer* crea la propiedad "*Color*" para que el programador pueda cambiar su color si así lo desea.

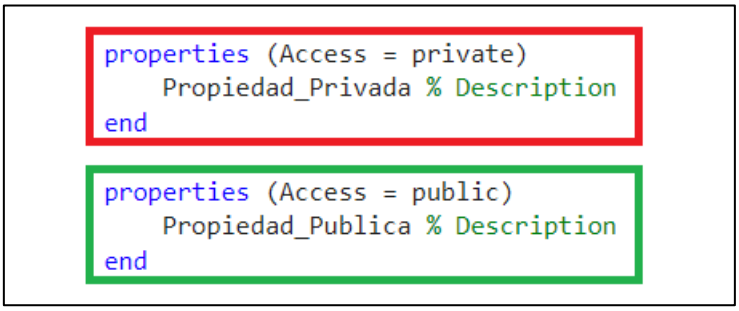


Figura 23: Tipos de propiedades en *App Designer*.

Para acceder a las propiedades desde el espacio de trabajo de *MATLAB*, debemos escribir el nombre de la aplicación y se devuelve como argumento el objeto *app*, que vimos al inicio de este apartado, con todas las propiedades públicas definidas en el instante en el que se hace la llamada a la estructura. Estas propiedades se siguen actualizando mientras la aplicación está ejecutándose.

Por ejemplo, con el comando

$$X = miaplicacion;$$

ejecutamos la interfaz "*miaplicacion*" y nos devuelve la estructura *app* en la variable 'X'. Si queremos acceder a una propiedad 'Y' debemos ejecutar el siguiente comando para ver su valor:

$$X.Y$$

3.4.4 Llamada a las funciones

En *App Designer* existen tres tipos de funciones:

- Funciones que se ejecutan al iniciar la aplicación, como la función *start-up*.
- *Callbacks*: funciones que se ejecutan al interactuar con los elementos de la interfaz.
- Funciones que realizan tareas y se pueden ejecutar dentro de la aplicación (privadas) o dentro y fuera de la aplicación (públicas). Las primeras se suelen usar cuando necesitamos usar los mismos comandos código en diferentes puntos de la aplicación, ahorrando así código. Las segundas se usan cuando necesitamos compartir esa función con otras interfaces, con *MATLAB* o con otros dispositivos conectados.

Creación y llamada de funciones

Para crear funciones en *App Designer* seguimos una ruta similar a la de las propiedades (véase 3.4.3 Compartir datos entre callbacks): vamos a la pestaña de *Code Browser* → *Functions* y clicamos en el símbolo '+' (véase Figura 24).

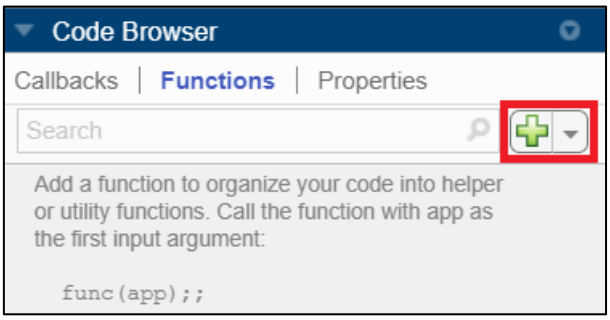


Figura 24: Crear funciones a través de *App Designer*.

A continuación, se crea el espacio en el editor de código donde se definirán todas las funciones que creemos (véase Figura 25). Si es la primera vez que creamos una función, App Designer genera el campo "*methods*". El identificador de la función es el campo "*func*" y es editable. Los argumentos de entrada se pueden definir en los paréntesis separados por comas y los de salida igual, pero en el campo "*results*" (si no se necesita un argumento de vuelta, simplemente habría que eliminar este campo).

```
methods (Access = private)

    function results = func(app)

    end
end
```

Figura 25: Aspecto de una función nueva en *App Designer*.

También se pueden crear funciones desde *MATLAB*. La utilidad de esto reside en que el código se manipula desde el espacio de trabajo de *MATLAB*, ahorrando así código en *App Designer* y mejorando su legibilidad. Además, permite usar el archivo generado al crear la función de una manera mucho más sencilla. Para este método seguimos la ruta desde *MATLAB*: *Home* → *New* → *Function* (véase Figura 26).

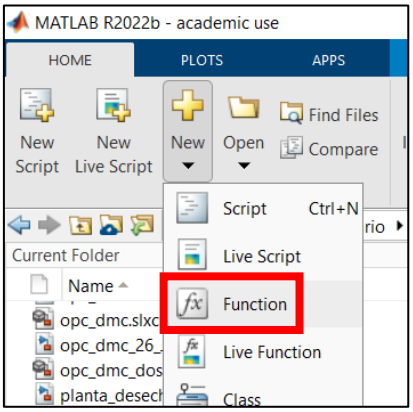


Figura 26: Crear funciones a través de *MATLAB*.

Para llamar a las funciones tendremos dos caminos distintos en función del tipo de función.

- Para funciones privadas basta con escribir el identificador y los argumentos de salida/entrada correspondientes en la zona donde queramos usarla.
- Para funciones públicas necesitamos que estas existan en el espacio de trabajo de la interfaz que llama a estas funciones. Por ejemplo, si tenemos una aplicación con una función pública y queremos llamar a la función desde una segunda interfaz, esta última tiene que realizar una llamada a la función y guardar los datos en una propiedad:

$$app2.propiedad = app1;$$

Ahora la segunda interfaz puede usar la función:

$$results = app2.propiedad.funcion\_app1;$$

3.4.5 Errores de código: detección y corrección

El desarrollador cuenta con varias formas de hallar y mostrar errores. Distinguiremos dos tipos de errores/avisos: advertencias de codificación y errores durante la ejecución del programa.

Advertencias de codificación.

App Designer señaliza errores mientras se está programando la interfaz mediante estas alertas de codificación. Se muestran como consejos de optimización y/o mejora de código, o como fallos de código.

En caso de que se cometa una falla del primer tipo, nos aparecerá un símbolo de advertencia en la esquina superior derecha de la pestaña *Code View* (clicando en este icono nos podremos desplazar entre los errores cometidos instantáneamente). Además, el fallo aparecerá en el código con un subrayado naranja como se muestra en la Figura 27.

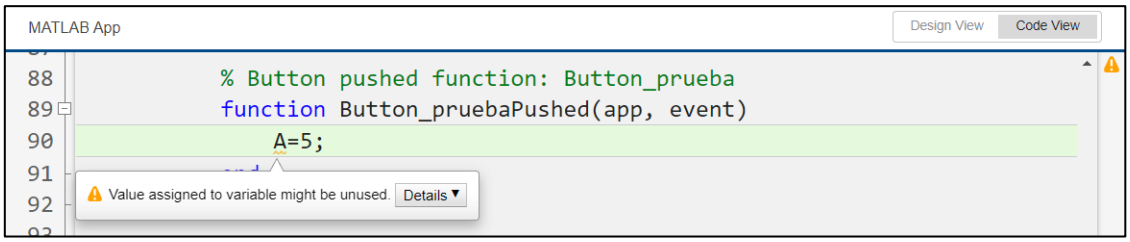


Figura 27: Advertencia de tipo optimización de código.

Si el fallo cometido tiene una solución simple, el propio *App Designer* ofrece la opción de corregirlo automáticamente (véase Figura 28).

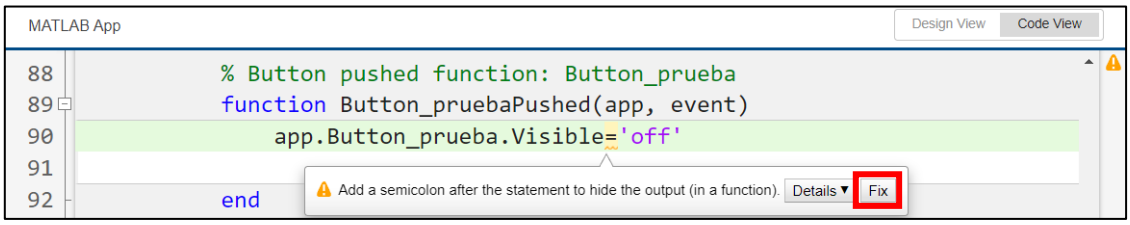


Figura 28: Advertencia de tipo optimización con opción de corrección automática.

En caso de cometer un error de fallo de código, nos aparecerá un símbolo de error en la esquina superior derecha de la pestaña *Code View* (clicando en este icono nos podremos desplazar entre los errores cometidos instantáneamente). Además, el fallo aparecerá en el código con un subrayado rojo como se muestra en la Figura 29.

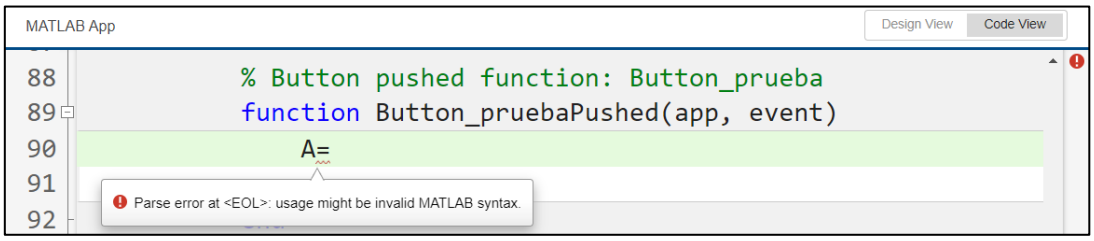


Figura 29: Advertencia de tipo fallo de código.

Estas advertencias las genera el *Code Analyzer* [3] y, a pesar de ser el mismo que el de *MATLAB*, no proporciona todas las funciones disponibles (algunos mensajes no ofrecen detalles y el ajuste de preferencias del *Code Analyzer* no tiene efectos sobre el editor de *App Designer*). Además, hasta la versión *R2022b*, las advertencias y los mensajes del *Code Analyzer* eran diferentes y aparecían de distinta forma.

Fallos durante la ejecución de la aplicación

Es posible que nos ocurra un fallo mientras estemos manejando la interfaz que no haya detectado el *Code Analyzer*. En este caso, *App Designer* marca con un icono de error la

línea donde se ha producido el error y nos dirige a él de forma automática. Haciendo clic en el icono nos mostrará un mensaje con una breve descripción del motivo del fallo, como es el caso de la Figura 30.

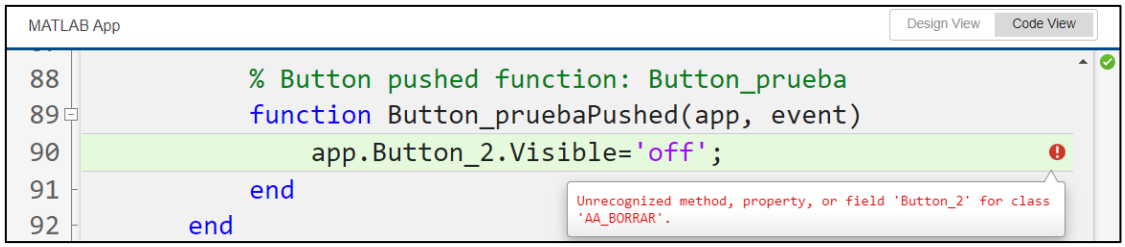


Figura 30: Fallo durante la ejecución de la app.

3.5 COMPILACIÓN DE APLICACIONES

Las interfaces desarrolladas con *App Designer* se pueden iniciar en computadoras que tengan versiones recientes a la *R2016a*. Basta con poner la dirección de la carpeta de la app como la ruta de trabajo de *MATLAB* y escribir el nombre en el *Command Window*.

Como se mencionó en el apartado 3.2 ENTORNO DE TRABAJO, *App Designer* da la opción de exportar la aplicación para ejecutarla dentro de *MATLAB* o como aplicación web/escritorio. En este caso, nosotros no necesitamos que la aplicación sea externa a *MATLAB*, por lo que exportaremos la aplicación para ejecutarla dentro del programa. Así se facilita la manipulación de variables (por si el usuario quiere graficar la entrada/salida) y mejora la compatibilidad con otras aplicaciones ya creadas (como *HIDEN* para la identificación de modelos).

Una vez finalicemos con el desarrollo, podemos compilar la app en un archivo "*.mlappinstall*", el cual permite, al usuario interesado, incluir la interfaz en la galería de aplicaciones de *MATLAB*, olvidándonos así de detalles de instalación o rutas de trabajo. A continuación, se muestra el procedimiento.

1. Seleccionamos el icono "Share" visto en el apartado 3.2 ENTORNO DE TRABAJO. Se nos abrirá una ventana como la de la Figura 31 donde podemos distinguir tres secciones: *Pick main file* a la izquierda (detalles relacionados con el archivo de la aplicación y demás documentos utilizados como imágenes, funciones…), *Describe your app* en el centro (detalles descriptivos de la interfaz: nombre, autor, resumen…) y *Package into installation* file a la derecha (ruta de salida de la compilación y el botón de inicio de compilación).
2. Corroborar y modificar que los campos que vienen por defecto son los correctos y rellenar aquellos que el creador considere oportuno. Los campos que *App Designer* rellena de forma automática son:
    - Nombre final de la app (recuadro rojo de la Figura 32).
    - Archivo principal a compilar (recuadro rosa de la Figura 32).
    - Ruta de destino (recuadro verde de la Figura 32).

Aquellos campos que el usuario puede rellenar se pueden encontrar en la Figura 33:

A. Icono de la aplicación.
B. Versión.
C. Información del desarrollador.
D. Captura de pantalla que representará a la aplicación.
E. Resumen y descripción de la interfaz.

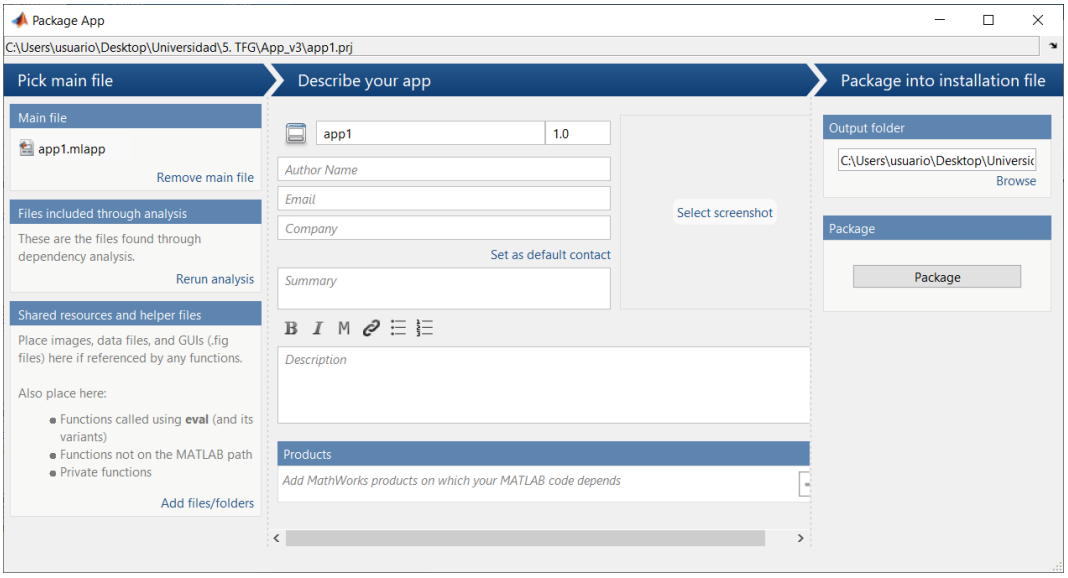


Figura 31: Ventana *Package App*.

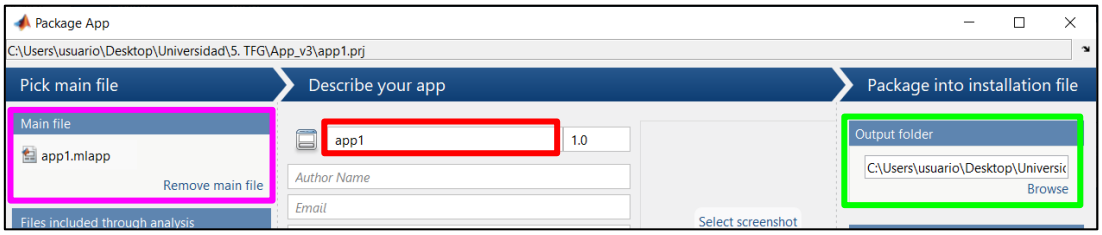


Figura 32: Campos rellenos por defecto.

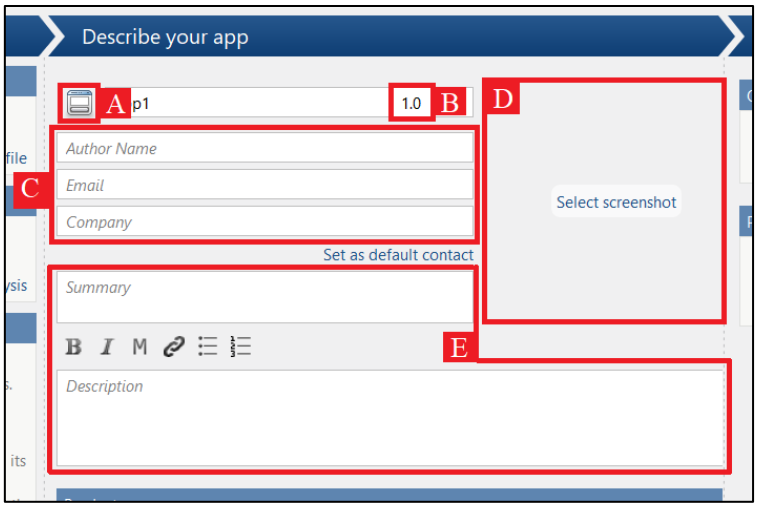


Figura 33: Campos descriptivos de la aplicación.

3. Comprobar que están incluidos los archivos externos utilizados en *App Designer* ya que *MATLAB* no encuentra archivos como imágenes externas, archivos de datos, funciones que no se hayan guardado en la ruta de trabajo o funciones privadas que no son propias de *MATLAB*, entre otros.
   Los archivos encontrados de forma automática se pueden encontrar en la sección "*Files included through análisis*" (recuadro rojo de la Figura 34). Si queremos añadir algún archivo, tenemos que hacer clic en "*Add files/folders*" en la sección "*Shared resources and helper files*" (recuadro verde de la Figura 34).

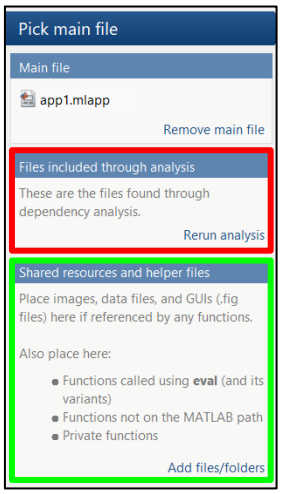


Figura 34: Secciones del *Pick main file*.

Es importante remarcar que pueden incluirse interfaces externas en el archivo compilado *mlappinstall*, sin embargo, puede ser que reduzcamos la compatibilidad de aplicación para ejecutarse en otros sistemas.

4. El compilador cuenta con la posibilidad de especificar qué herramientas son necesarias para que la aplicación funcione sin problemas. Esta sección se cuenta en la parte inferior de la ventana con el nombre de "*Products*", al hacer clic se nos abre la selección de herramientas que nosotros tenemos instaladas en el ordenador (véase Figura 35).

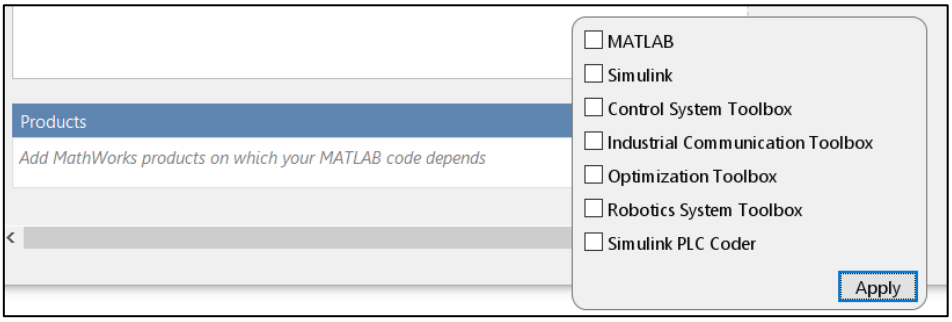


Figura 35: Sección *Products* y herramientas disponibles.

5. Hacer clic en "*Package*" (véase Figura 31) y esperar a que se compile la aplicación.

Una vez haya finalizado la compilación, se generará en la ruta establecida un fichero "*.prj*" que contiene todos los datos y archivos de la aplicación. Con este archivo podemos en un futuro actualizar la aplicación sin tener que volver a introducir todos los datos que no hayan sido modificados, como el nombre del creador, descripción…

Si se actualiza en un futuro la aplicación, es conveniente cambiar el número de versión, así el futuro usuario que quiera instalar la aplicación verá si va a trabajar con la última actualización.

El archivo *mlappintall* generado se puede compartir bien por correo electrónico, memoria USB y demás maneras de intercambiar información. También se puede subir a la plataforma de recursos compartidos de *MATLAB*: *MATLAB Central File Exchange* y todo usuario que quiera la aplicación podrá descargarla buscándola en el *Add-Ons* del menú de *MATLAB*.

3.6 DIFERENCIAS ENTRE *GUIDE* Y *APP DESIGNER*

Como vimos en la Figura 5 del apartado 3.1 INTRODUCCIÓN, el entorno de desarrollo *GUIDE* va a ser eliminado en futuras versiones. Esto se debe a que *App Designer*, además de las características que compartían ambas herramientas, ha ido adoptando, con ligeros cambios, muchas de las funcionalidades que le diferenciaban con el otro desarrollador. Aun así, el procedimiento de creación de aplicaciones es distinto ya que varían ciertos aspectos de desarrollo como: el código generado, creación y acceso de elementos, programación de *callbacks*, componente gráfico…

En este apartado, veremos de manera resumida las diferencias de los principales aspectos que hay entre ambos entornos de desarrollo.

3.6.1 Estructura del código

En caso de ***GUIDE*** el código lo formaban un conjunto de funciones y *callbacks* sin estructura definida. En cambio, *App Designer* tiene una configuración igual que una estructura/clase *MATLAB* mejorando así su comprensión. Su organización se vio en el apartado 3.4 VENTANA 'CODE VIEW': propiedades, funciones/callbacks establecidas por el creador y por último el código usado para la inicialización de la interfaz y la creación de componentes.

En ***App Designer***, para evitar que accidentalmente se modifique el código que se genera de forma automática, sólo es posible editar el código que genera el usuario (propiedades añadidas, funciones y *callbacks*). Además, solo se generan las *callbacks* cuando el creador lo solicita, evitando así crear funciones innecesarias por lo que reducimos código.

En cuanto a la compilación de la interfaz, *App Designer* genera un único archivo *.mlapp* que contiene lo necesario para iniciar la app.

3.6.2 Creación y configuración de elementos

Creación

Para crear un componente en ***GUIDE***, se usa la función '*uicontrol*' y con sus propiedades controlamos el aspecto y comportamiento de todos los componentes. No obstante, no todas las propiedades se pueden aplicar para todos los objetos, algunas solo se aplican a subconjuntos.

En *App Designer*, cada elemento tiene su función (por ejemplo, '*uislider*' crea un componente deslizable) y un conjunto de propiedades exclusivas del objeto. Logramos así que solo se pueda acceder a las propiedades de los componentes que se hayan creado.

Configuración

Con ***GUIDE*** necesitamos la función '*get*' para acceder a una propiedad de un componente y la función '*set*' para modificarla.

Con ***App Designer***, los elementos son propiedades de la aplicación, por lo que se usa la notación de puntos para acceder a ellas.

Por ejemplo, si queremos modificar el valor de un *Numeric Edit* Field a cero, tenemos los siguientes comandos en función de la herramienta:

- ***GUIDE***: $set(handles.EditField,'value') = 0;$
- ***App Designer***: $app.EditField.Value = 0;$

3.6.3 Argumentos de las *callbacks*

En ***GUIDE***, los argumentos son *hObject*, *eventdata* y *handles*. El primero se refiere al objeto que se está ejecutando, el segundo contiene información de los eventos específicos (como qué botón se ha pulsado) y el tercero es una estructura que alberga todos los elementos generados.

En ***App Designer*** solo tenemos dos argumentos de entrada: *app* y *event*. El primero hace referencia a la clase aplicación (la cual contiene todos los componentes de la interfaz) y el segundo contiene una referencia al componente UI y datos sobre el evento. Por tanto, mejoramos la comprensión del código y lo reducimos en comparación con GUIDE.

3.6.4 Compartir datos

En ***GUIDE*** podemos usar los comandos *setappdata* y *getappdata* para dar un valor a una variable y para leerlo, respectivamente. Por ejemplo, primero damos el valor a una variable de la siguiente manera:

$$setappdata(handles.Identificador,'variable',valor);$$

y después, recuperamos el valor en el *callback* deseado:

$$Dato = getappdata(handles.Identificador,'variable');$$

En ***App Designer***, como se vio en el apartado 3.4.3 Compartir datos entre callbacks, tendríamos primero que crear la propiedad:

$$app.propiedad;$$

Después, le damos el valor deseado:

$$app.propiedad = 0;$$

Y, por último, acceder a ella en el *callback* deseado:

$$valor = app.propiedad;$$

CAPÍTULO 4: IMPLEMENTACIÓN DEL CONTROL PREDICTIVO DMC

Para ejercer el control predictivo en la aplicación, haremos uso de un archivo MATLAB que contiene una función con el código necesario: ‘***dmc.m’***. Se trata del “corazón” de la aplicación, donde se realizan todos los cálculos relevantes del control. Este código ha sido desarrollado por Yi Cao, ingeniero en Sistemas de Control de la Universidad de Crandfield, en el año 2007 y se encuentra disponible en su página de *Mathworks* [4].

Como el archivo trabaja con código propio de *MATLAB*, tenemos la ventaja de no requerir de ‘*Toolbox*’, por lo que, gracias a su uso, nos prescinde de evitar de instalar software a mayores. Sí necesitaremos una *Toolbox* a la hora de tratar el tema de las comunicaciones, que veremos en el CAPÍTULO 5: INDUSTRIAL COMMUNICATION TOOLBOX.

4.1 ESTRUCTURA DEL ARCHIVO ‘*dmc.m*’

4.1.1 CÓDIGO DE LA FUNCIÓN

Este archivo tiene el siguiente el siguiente código:

```
function p=dmc(p)
   % DMC   Dynamic Matrix Control
   % P=DMC(P) determines the dynamic matrix control (input change) based on
   % the plant model (step response) and current measurement stored in the
   % structure P.
   % Input:
   %   P.sr - unit step response data
   %   P.u  - current input, initially 0
   %   P.v  - past input, initially empty
   %   P.G  - dynamic matrix, set by the initial call
   %   P.F  - matrix to calculate free response, set by the initial call
   %   P.k  - DMC gain, set by the initial call
   %   P.r  - reference (set point)
   %   P.a  - reference smooth factor
   %   P.p  - prediction horizon
   %   P.m  - moving horizon
   %   P.y  - current mrasurement
   %   P.la - performance criterion weight, i.e. J = ||r-y|| + p.la*||du||
   %          where du is the input change
   % Output:
   %   P.u  - new input for next step
   %   P.f  - updated free response
   %   P.G  - dynamic matrix, if it is the first step.
   %   P.k  - DMC gain, if it is the first step
   %
```

```
    % See Also: mpc

    % Version 1.0 created by Yi Cao at Cranfield University on 6th April 2008.

    % Input and output check
    error(nargchk(1,1,nargin));
    error(nargoutchk(0,1,nargout));

    % length of step response
    N=numel(p.sr);
    P=p.p;

    % initial setup
    if isempty(p.v)
        % number of past inputs to keep
        n=N-P;
        % storage for past input
        p.v=zeros(n,1);
        % matrix to calculate free response from past input
        x=p.sr(1:n);
        p.F=hankel(p.sr(2:P+1),p.sr(P+1:N))-repmat(x(:)',P,1);
        % dynamic matrix
        p.G=toeplitz(p.sr(1:P),p.sr(1)*eye(1,p.m));
        % calculate DMC gain
        R=chol(p.G'*p.G+p.la*eye(p.m));
        K=R\(R'\p.G');
        % only the first input will be used
        p.k=K(1,:);
        p.u=0;
    end
    if isempty(p.y)
        return
    end
    % free response
    f=p.F*p.v+p.y;
    % smooth reference
    nr=numel(p.r);
    if nr>=P
        ref=p.r(1:P);
    else
        ref=[p.r(:);p.r(end)+zeros(P-nr,1)];
    end
    w=filter([0 (1-p.a)],[1 -p.a],ref,p.y);
    % DMC input change
    u=p.k*(w-f);
    % past input change for next step
    p.v=[u;p.v(1:end-1)];
    % next input
    p.u=p.u+u(1);
end
```

No nos centraremos en el desarrollo de los cálculos, desarrollados en el apartado 2.3 DYNAMIC MATRIX CONTROL - DMC, así se relatará el código sin mucho detalle.

Como podemos ver, a la función solamente le tiene que llegar un argumento: una estructura que guarda todos los elementos relacionados con el cálculo del DMC. La

función calcula la matriz de control dinámica en función de la respuesta escalón y los datos guardados en la estructura *p*.

Antes de comenzar con los cálculos, podemos ver, como código comentado, una descripción de las variables de la estructura, tanto de entrada (respuesta escalón, matriz dinámica actual, consigna/referencia…) como de salida (entrada para el siguiente paso, respuesta libre actualizada, nueva matriz dinámica con posibles cambios…).

Lo primero que tenemos en la función es una comprobación de los valores de la salida del paso anterior '*p*.v' y la medida actual '*p.y*'. En el caso de la primera variable, la función realizará los cálculos necesarios (matriz dinámica y de respuesta libre, ganancia del sistema…) y establecería un valor de cero para la entrada del siguiente paso, si '*p.v*' está vacío. O se saltaría este paso en caso de que la salida del paso anterior fuera distinta de cero. Para la variable de la medida actual '*p.y*' nos sacaría de la función si esta fuera cero, o seguiría ejecutando el código siguiente.

Tras las comprobaciones de estas dos variables, la función continúa realizando los cálculos de respuesta libre y señal de control. Esta última es guardada en la estructura *p* para su uso en la siguiente iteración.

4.2 ALTERNATIVA: CONTROL DMC EN *SIMULINK*

Yi Cao también ha desarrollado un modelo en *SIMULINK* que permite el control de plantas [5], en su caso, de un intercambiador de calor. Dicho modelo se muestra en la Figura 36. Como se menciona en el siguiente apartado, nosotros optaremos por el uso del archivo *MATLAB*, así que se explicará este modelo sin entrar en mucho detalle. Simplemente quiero mencionar que en el caso de que en un futuro se quiera implementar un control a través de *SIMULINK*, tenemos este modelo como base.

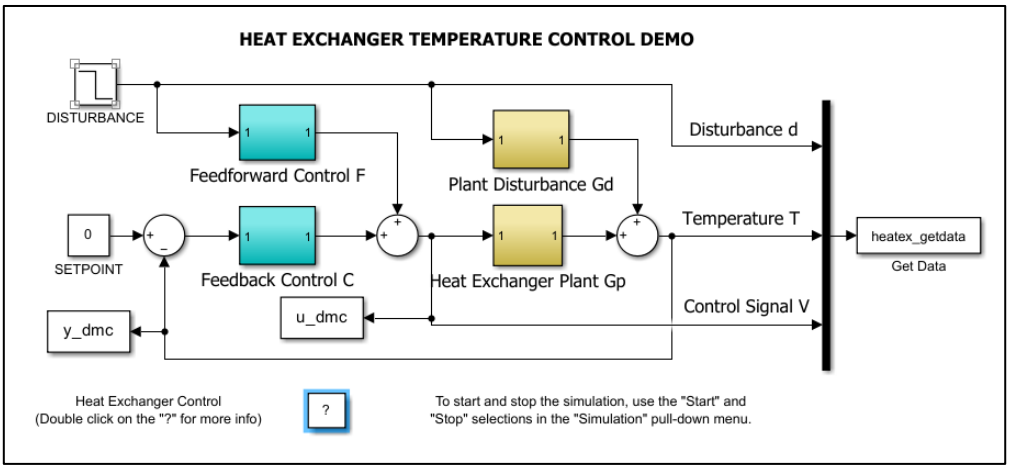


Figura 36: Modelo *SIMULINK* de una planta controlada por DMC.

A simple vista observamos que hay varios elementos y unos conjuntos de bloques. Veremos qué acciones realiza cada bloque de manera resumida.

Bloque de control predictivo: Feedback Control C.

Al bloque de control predictivo *Feedback Control C* le llega la resta de la consigna *SETPOINT* con la variable controlada (ver Figura 36) desde fuera a través del puerto 1 de la Figura 37. Y saca la respuesta del control DMC.

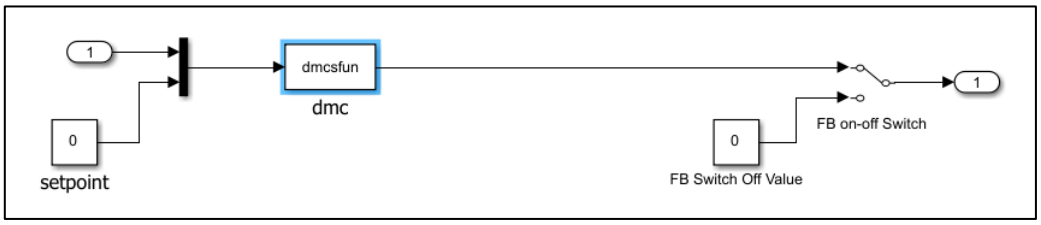


Figura 37: Contenido del bloque *Feedback Control C*.

En su interior podemos encontrar un bloque *S-Function* [6] que realiza una llamada a la función '*dmcsfun*' (véase Figura 37) la cual tiene el siguiente código:

```
function [sys,x0,str,ts] = dmcsfun(t,x,u,flag,varargin)

%DMCSFUN S-Function for Dynamic Matrix Control.
%By Yi Cao, Cranfield University, UK, (c) 2009

persistent K

switch flag

    %%%%%%%%%%%%%%%%%%
    % Initialization %
    %%%%%%%%%%%%%%%%%%
    case 0
        Ts=varargin{1};           % Sampling time
        K.sr=varargin{2};         % Step Response Model
        K.p=varargin{3};          % P, prediction horizon
        K.m=varargin{4};          % M, moving horizon
        K.v=[];              % History of control
        K.la=varargin{5};       % input weight, i.e. J = ||r-y|| + p.la*||du||
        if numel(varargin)>5
            K.a=varargin{6};           % reference smooth factor
        else
            K.a=0;
        end
        K.y=[];                 % Initialization DMC
        K=dmc(K);

        sizes = simsizes;
        sizes.NumContStates  = 0;
        sizes.NumDiscStates  = 0;
        sizes.NumOutputs     = 1;
        sizes.NumInputs      = 2;     % measured output and setpoint
        sizes.DirFeedthrough = 1;
        sizes.NumSampleTimes = 1;
        sys = simsizes(sizes);
        str = [];
```

```
        x0 = zeros(0,1);
        ts  = [Ts 0];

        %%%%%%%%%%
        % Update %
        %%%%%%%%%%
    case 2
        sys = x;

        %%%%%%%%%%
        % Output %
        %%%%%%%%%%
    case 3
        %       if t>50
        %           u;
        %       end
        K.y = u(1);       % measurement
        K.r = u(2);       % setpoint
        K = dmc(K);
        sys = K.u;
        %%%%%%%%%%%%
        % Terminate %
        %%%%%%%%%%%%
    case 9
        sys = []; % do nothing

        %%%%%%%%%%%%%%%%%%%
        % Unexpected flags %
        %%%%%%%%%%%%%%%%%%%
    otherwise
        error(['unhandled flag = ',num2str(flag)]);
end
```

Como es particular de este bloque, tenemos las cuatro entradas propias:

- Instante de tiempo ***t***.
- Vector de estados ***x***.
- Vector de entrada ***u***. que será la señal de entrada del bloque.
- Variable numérica entera ***flag***. Que nos servirá para decidir cómo actúa el control.

Además, tiene una variable más llamada ***vargargin*** que se trataría de una estructura que guarda los elementos para realizar el cálculo del control DMC, es decir, es la misma que la estructura *p* de la función vista en el apartado anterior (las variables de esta estructura se definen en los parámetros del bloque en *SIMULINK*).

Se trata de un bloque *S-Function* de nivel 1, es decir, también tiene cuatro salidas propias:

- Variable de retorno genérico ***sys***. Su valor dependerá del valor de la variable bandera.
- Valores de estado iniciales ***X0***. En caso de no existir estados, su valor es un vector vacío. En la función por defecto este valor se ignora, excepto cuando la bandera es cero.
- Variable ***str***. Se trata de una variable pensada para su uso en el futuro. Las funciones de este nivel lo devuelven como matriz vacía.

- Matriz de dos columnas ***ts*** que contiene los tiempos de muestreo y desplazamientos del bloque.

Dentro de la variable bandera, tenemos los siguientes posibles casos:

Tabla 17: Posibles valores de la variable bandera *flag*.

Valor	Descripción
0	Define las características básicas del bloque *S-Function*, incluidos los tiempos de muestreo, las condiciones iniciales de los estados continuos y discretos y la matriz de tamaños. En nuestro caso, además se inicializa el DMC.
1	Calcula las derivadas de las variables de estado continuas. Para nuestro caso no es necesaria.
2	Actualiza los estados discretos, los tiempos de muestreo y los requisitos de los pasos temporales principales.
3	Calcula las salidas de la función del bloque. En nuestro modelo, calculará el valor del DMC.
4	Calcula el tiempo del siguiente pulso en tiempo absoluto. Esta rutina sólo se utiliza cuando se especifica un tiempo de muestreo variable en el método de configuración. Nosotros usamos un valor fijo, por lo que esta opción no se usará.
9	Realiza todas las tareas necesarias al final de la simulación. En nuestro caso, no se hace nada.

Como podemos ver en el código, la función del bloque *S-Function* llama a otra función *dmc()* que calcula el valor del control para el caso 1 (inicializar) y el caso 3 (cálculo de la señal del control predictivo). El código de esta otra función se analiza en el apartado 4.1 CÓDIGO DEL ARCHIVO 'dmc.m'.

Bloque de la función de transferencia de la planta: *Heat Exchanger Plant Gp*.

Como indica la nomenclatura de este bloque, se trata de la función de transferencia del intercambiador de calor de la planta definida por Yi Cao. A este bloque le llega la suma de la señal de control y las perturbaciones, tras ser procesadas por un control *Feedforward*, y sale la señal de salida del intercambiador.

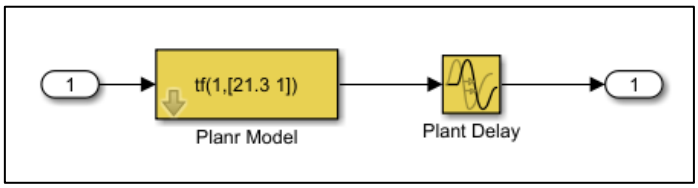


Figura 38: Contenido del bloque *Heat Exchanger Plant Gp*.

En su interior simplemente encontramos la función de transferencia del intercambiador y un bloque que induce un retardo (véase Figura 38).

Nuestra planta no necesita calcular la señal de salida, solo nos basta con la señal del control predictivo DMC, ya que esta señal se mandará al servidor y la planta de destino es la que calcula la salida. Entonces, este bloque no nos interesaría en nuestro diseño, se descartaría en un futuro (véase CAPÍTULO 6: DESARROLLO DE LA INTERFAZ).

Bloque de control de perturbaciones: *Feedforward Control F*.

A este bloque le llega una señal de una perturbación y saca la señal del control procesada por un control *Feedfordward*. Un control de este tipo busca compensar la perturbación antes de que afecte a la variable controlada. Produce una acción idéntica a la perturbación, pero en sentido contrario para que su efecto en la salida sea nulo. Este tipo de controles se usan para perturbaciones medibles y de efecto no controlable directamente.

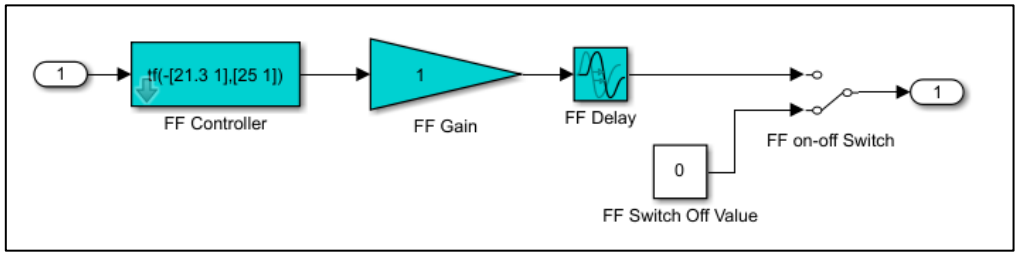


Figura 39: Contenido del bloque *Feedforward Control F*.

Fijándonos en la Figura 39 vemos que la señal de la perturbación pasa por la función de transferencia del control *Feedforward* y tras un pequeño retardo (al igual que en el apartado anterior) la sacaría en caso de que el bloque *switch* lo conectara con la salida.

Como indicamos en el apartado de objetivos (véase 1.2 OBJETIVOS), este proyecto no tendrá en cuenta las perturbaciones medibles, por lo que nosotros mantendríamos en cero el *switch*.

Bloque de la función de transferencia de las perturbaciones: *Plant Disturbance Gd*.

Al igual que para el intercambiador de calor, se trata de la función de transferencia de las perturbaciones. Le llega la señal de las perturbaciones y saca la señal de salida de las perturbaciones tras pasar por la función de transferencia. El contenido del bloque se muestra a continuación (Figura 40):

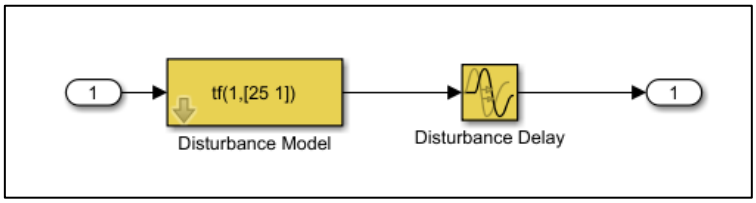


Figura 40: Contenido del bloque *Plant Disturbance Gd*.

Al igual que en el caso anterior, como no tendremos en cuenta este tipo de perturbaciones para este proyecto, se descartaría el bloque.

4.2 IMPLEMENTACIÓN DE LA FUNCIÓN

A la hora de incorporar el control DMC a la interfaz, se ha optado por operar con el modelo en código *MATLAB* por varios motivos:

- El principal es porque prescindiendo de *SIMULINK*, logramos reducir el coste computacional, mejorando el rendimiento del ordenador y de la aplicación.
- También permitimos que el usuario pueda acceder al código de la función de una manera más sencilla.
- Podemos exportar la aplicación para uso fuera de *MATLAB*, si fuera necesario.

Para implementar la función hay que tener en cuenta varias premisas que nos harán modificar ligeramente el archivo. Mirando el código en el apartado 4.1.1 CÓDIGO DE LA FUNCIÓN, vemos que los primeros cálculos (los de las matrices del control DMC) solo se realizan en la primera iteración: cuando no hay lecturas de la variable de proceso ni registro de la señal de control. Esto supone un problema si queremos cambiar el horizonte de predicción, el de control o el factor de peso una vez se ha ejecutado una vez esta función. En el primer caso, a la hora de calcular la respuesta libre, no se corresponden las dimensiones de la matriz de la respuesta libre *f* con la matriz *w* (usada para el cálculo de la señal de control). En el segundo caso, no se tendría en cuenta el valor del horizonte de control en el cálculo, ya que se usa para los cálculos la primera vez y después no vuelve a entrar en el condicional.

Teniendo en cuenta esto se han realizado las modificaciones oportunas, obteniendo el nuevo código:

```
function p=dmc(p)
% Input and output check
error(nargchk(1,1,nargin));
error(nargoutchk(0,1,nargout));

% Persistent var. for the change management of the prediction & control
horizon
persistent past_prediction_horizont;
persistent past_control_horizont;
persistent past_criterion_weight;
persistent resp_salto_interna;

% length of step response
N=numel(p.sr);
P=p.p;

% initial setup
if isempty(p.v)
    % storage for past input
    p.v=zeros(N,1);
    % assing the step resp to the persistent var for operations
    resp_salto_interna=p.sr;
    % matrix to calculate free response from past input
    x=p.sr(1:N);
    resp_salto_interna(N+1:N+P)=resp_salto_interna(N);
    p.F=hankel(resp_salto_interna(2:P+1),resp_salto_interna(P+1:N+P))-
repmat(x(:)',P,1);
    % dynamic matrix
    p.G=toeplitz(resp_salto_interna(1:P),resp_salto_interna(1)*eye(1,p.m));
    % calculate DMC gain
    R=chol(p.G'*p.G+p.la*eye(p.m));
    K=R\(R'\p.G');
    % only the first input will be used
    p.k=K(1,:);
    % Assign the value of MV only in the first iteration
    %p.u=0;
    p.u=p.iteracion_inicial;
    % Assign the value of pred. & control horizont to the persistent variable
    past_prediction_horizont=P;
    past_control_horizont=p.m;
    % Assign the value of performance criterion weight to the persistent
variable
    past_criterion_weight=p.la;
end
if isempty(p.y)
    return
end

% Prediction horizont change management
if (past_prediction_horizont~=P)
    % assing the step resp to the persistent var for operations
    resp_salto_interna=p.sr;
    % matrix to calculate free response from past input
    x=p.sr(1:N);
    resp_salto_interna(N+1:N+P)=resp_salto_interna(N);
    p.F=hankel(resp_salto_interna(2:P+1),resp_salto_interna(P+1:N+P))-
repmat(x(:)',P,1);
    % dynamic matrix
    p.G=toeplitz(resp_salto_interna(1:P),resp_salto_interna(1)*eye(1,p.m));
```

```
    % calculate DMC gain
    R=chol(p.G'*p.G+p.la*eye(p.m));
    K=R\(R'\p.G');
    % only the first input will be used
    p.k=K(1,:);
    % Assign the value of pred. horizont to the persistent variable
    past_prediction_horizont=P;
end

% Control horizont & criterion weight change management
if (past_control_horizont~=p.m || past_criterion_weight~=p.la)
    % dynamic matrix
    p.G=toeplitz(resp_salto_interna(1:P),resp_salto_interna(1)*eye(1,p.m));
    % calculate DMC gain
    R=chol(p.G'*p.G+p.la*eye(p.m));
    K=R\(R'\p.G');
    % only the first input will be used
    p.k=K(1,:);
    % Assign the value of control horizont to the persistent variable
    past_control_horizont=p.m;
    % Assign the value of performance criterion weight to the persistent
variable
    past_criterion_weight=p.la;
end

% free response
f=p.F*p.v+p.y;
% smooth reference
nr=numel(p.r);
if nr>=P
    ref=p.r(1:P);
else
    ref=[p.r(:);p.r(end)+zeros(P-nr,1)];
end
w=filter([0 (1-p.a)],[1 -p.a],ref,p.y);
% DMC input change
u=p.k*(w-f);
% past input change for next step
p.v=[u;p.v(1:end-1)];
% next input
p.u=p.u+u(1);

end
```

Añadiendo tres nuevas variables persistentes (que mantienen el valor cuando la función no se usa) podemos introducir varias nuevas condiciones, una para cada cambio de cada variable. En el momento que cambie el horizonte de predicción, el sistema entrará en el condicional que calcula de nuevo las matrices después de hacer una redimensión del vector que almacena la señal de control. Si se cambia el horizonte de control o el factor de peso, se entra en el condicional que solo calcula de nuevo las matrices. Además, para suavizar el cambio de control de manual a automático, en la primera iteración de la función se asigna el último valor de la variable manipulada a la señal de control, perturbando mínimamente la respuesta del sistema.

Para evitar tener que redimensionar los vectores de cálculo, se usa la variable persistente "*resp_salto_interna*" que toma el valor de la respuesta salto cada vez que se cambien alguno de los parámetros mencionados anteriormente (y se entren en sus respectivos bucles). Con esto se calculan las matrices de nuevo sin modificar los vectores de almacenamiento de valores.

De esta manera corregimos los problemas expuestos haciendo que el cambio de control sea lo más suave posible. Otra posible opción sería simplemente vaciar los vectores de las lecturas de la variable de proceso y de la señal de control calculada. El problema de esta solución radica en que perderíamos el control calculado hasta ese momento, pudiendo perder el estacionario y así tener que esperar a que se estabilice de nuevo.

Además, el código presentaba un '*bug*' debido a que se calculaban el vector '*p.v*' (almacenamiento del último valor del control) y la matriz de respuesta libre '*p.F*' limitando el valor del horizonte de predicción por debajo del número de coeficientes. Esto generaba dos problemas:

- Limitación en el valor del horizonte de predicción.
- Errores en el cálculo del valor de la señal de control. Esto hacía que el parámetro no se comportara como debe al cambiar su valor y provocaba en la señal de salida picos que no se correspondían con un comportamiento real, empeorando la calidad de control.

Entonces, sustituyendo la variable '*n*' por la suma del número de coeficientes y el horizonte de predicción, se corregían estos dos problemas, mejorando notablemente la calidad de la señal de control y, en consecuencia, la salida. A continuación, se muestran estos cambios en el código de la función:

CÓDIGO ANTERIOR

```
% number of past inputs to keep
n=N-P;
% storage for past input
p.v=zeros(n,1);
% matrix to calculate free response from past input
x=p.sr(1:n);
p.F=hankel(p.sr(2:P+1),p.sr(P+1:N))-repmat(x(:)',P,1);
```

CÓDIGO CORREGIDO

```
% storage for past input
p.v=zeros(N,1);
% assing the step resp to the persistent var for operations
resp_salto_interna=p.sr;
% matrix to calculate free response from past input
x=p.sr(1:N);
resp_salto_interna(N+1:N+P)=resp_salto_interna(N);
p.F=hankel(resp_salto_interna(2:P+1),resp_salto_interna(P+1:N+P))-
repmat(x(:)',P,1);
```

Figura 41: Cambios en el código para la corrección en el cálculo del valor del DMC.

CAPÍTULO 5: INDUSTRIAL COMMUNICATION TOOLBOX

En el contexto de la automatización industrial, industria 4.0, etc. la comunicación eficiente y fiable es un requisito obligatorio para garantizar el rendimiento óptimo y el control del sistema. Para abordar este desafío de comunicación en el entorno industrial, *MATLAB* nos ofrece la herramienta ***Industrial Communication Toolbox***, la cual es idónea para nuestro objetivo de realizar un control predictivo sobre una planta a través de un servidor OPC [7].

5.1 INTRODUCCIÓN

La *Industrial Communication Toolbox* es un conjunto de herramientas avanzadas dentro de *MATLAB*, diseñadas para facilitar la interacción y comunicación entre dispositivos y sistemas en entornos industriales. Esta *toolbox* ofrece una gran variedad de protocolos y recursos que permiten, en tiempo real, comunicación y control efectivo de dispositivos, sensores, actuadores… Además, incluye bloques de *SIMULINK* que permiten modelar control de supervisión online y realizar pruebas de *hardware-in-the-loop* en controladores (se trata de un tipo de simulación usada para desarrollo y comprobación de sistemas embebidos en tiempo real complejos).

5.1.1 CARACTERÍSTICAS IMPORTANTES

Amplia gama de protocolos de comunicación.

La herramienta admite una variedad de protocolos comunes como *OPC UA*, *OPC DA* [7] (es el que usaremos nosotros), *Modbus*, *CAN* (*Controller Area Network*), *PROFIBUS*, servidores *PI*, *MQTT*… Esto permite la integración sencilla con los diferentes sistemas y dispositivos existentes en la industria.

Además, gracias a la comunidad de la plataforma *Mathworks File Exchange*, se han ido desarrollando otros protocolos (como RTL-SDR para radios), simulaciones sobre comunicaciones inalámbricas o software para controlar dispositivos que usen este último tipo de comunicación…

Fácil adquisición de datos en tiempo real.

Permite la adquisición de datos en tiempo real desde dispositivos industriales. Fundamental para monitoreo y control en tiempo real de procesos y sistemas.

Capacidad de análisis y visualización integradas.

Integra capacidades de análisis y visualización en MATLAB, lo que permite a los usuarios analizar con detalle la información obtenida y generar visualizaciones significativas para toma de decisiones.

Interfaz intuitiva y fácil de manipular.

La *toolbox* proporciona una interfaz de usuario intuitiva y afable que simplifica la configuración y operación. Esto permite al creador centrar su interés en las aplicaciones industriales sin abundantes complicaciones técnicas.

5.1.2 APLICACIONES

La herramienta de comunicaciones industriales es usada en amplios sectores industriales para una gama de aplicaciones igual de amplia:

- Supervisión y control de procesos industriales.
- Integración y comunicación con sistemas de automatización.
- Adquisición de datos y monitoreo de sensores y actuadores.
- Diagnóstico y análisis de sistemas en tiempo real.

5.2 FUNCIONES PARA SERVIDORES OPC

A continuación, veremos las funciones básicas para poder iniciar la conexión y desconexión con un servidor OPC, cómo acceder a las variables de lectura/escritura, obtener información del servidor… Tanto para comandos de *MATLAB* como para bloques *SIMULINK*.

5.2.1 FUNCIONES PARA ENTORNO MATLAB

Función *opcserverinfo*.

Sintaxis:

$$nombre\_variable = opcserverinfo('nombre\ host')$$

Este comando nos da información de los servidores OPC disponibles en un determinado nodo, además permite guardar la información en una variable si así lo deseamos. La respuesta obtenida se muestra en la Figura 42.

Antes de continuar, aclarar que para este proyecto el servidor se incluye dentro del propio computador, es decir, es el ordenador local quien proporciona el servidor, por eso sólo apreciamos el servidor '*localhost*'. En caso de usar otro servidor, sustituiríamos *localhost* por su nombre o dirección IP.

```
Command Window
>> servers = opcserverinfo('localhost')

servers =

  struct with fields:

                Host: 'localhost'
            ServerID: {'OPC.reactor_van_der_vusse.1'  'OPC.cstr.1'  'deck.opc.server.1.0'}
   ServerDescription: {'reactor_van_der_vusse'  'cstr'  'Depositos_Labo'}
    OPCSpecification: {'DA2'  'DA2'  'DA2'}
   ObjectConstructor: {1×3 cell}
```

Figura 42: Respuesta del comando *opcserverinfo('localhost')*.

Es muy importante acordarse de introducir el identificador de nuestro servidor. En caso de no escribir su nombre nos aparecería información sobre la *toolbox* (véase Figura 43).

```
Command Window
>> opcserverinfo

ans =

  struct with fields:

     MATLABVersion: '9.13 (R2022b)'
       ToolboxName: 'Industrial Communication Toolbox'
    ToolboxVersion: '6.1 (R2022b)'
```

Figura 43: Respuesta del comando *opcserverinfo*.

Centrándonos en la Figura 42, el comando nos ofrece información sobre varios campos:

- ***Host***: nombre del ordenador.
- ***ServerID***: distintos servidores disponibles dentro de nuestro *host*.
- ***ServerDescription***: nombres de los servidores anteriores.
- ***OPCSpecification***: tipo de estándar que se usa para acceder a los datos, en nuestro caso sería *DA2*, que se trata de comunicación bidireccional de tipo cliente-servidor, perfecto para acceder a los datos a tiempo real.
- ***ObjectConstructor***: método para crear instancias de objetos relacionados con el servidor *OPC*.

Función *opcda*.

Sintaxis:

$$nombre\_variable = opcda('nombre\_host', 'nombre\ servidor')$$

Esta función crea un objeto de acceso a los datos del servidor, para el *Host* especificado y el identificador del servidor *ServerID*. Partiendo del comando *opcserverinfo* (véase Figura 42), definimos el objeto como se muestra en la Figura 44.

```
Command Window
>> da = opcda('localhost', servers.ServerID{3})

da =

Summary of OPC Data Access Client Object: localhost/deck.opc.server.1.0

   Server Parameters
      Host      : localhost
      ServerID  : deck.opc.server.1.0
      Status    : disconnected
      Timeout   : 10 seconds

   Object Parameters
      Group     : 0-by-1 dagroup object
      Event Log : 0 of 1000 events
```

Figura 44: Respuesta del comando *opcda*.

La función nos muestra parámetros del servidor al que nos conectamos y del objeto creado. En nuestro caso tenemos:

- ***Host***: nombre del nodo.
- ***ServerID***: servidor al que nos conectaremos. En nuestro caso es el número 3 de los disponibles vistos en la Figura 42.
- ***Status***: Estado de la conexión, inicialmente desconectado.
- ***Timeout***: tiempo máximo de espera para completar la instrucción al servidor en segundos.
- ***Group***: referencia al grupo de datos asociado con el servidor. Es la forma de organizar y gestionar los datos del servidor.
- ***Event Log***: es el registro de eventos asociados con el objeto.

Como vemos en el campo *Status*, no estamos conectados por defecto al servidor.

Función *connect*.

Sintaxis:

$$connect(variable\_objeto\_opcda)$$

Este comando conecta la variable objeto *opcda* con el servidor. Al ejecutarlo iniciamos la conexión y se nos abre la ventana del servidor (véase Figura 45).

Figura 45: Ventana creada al conectarse al servidor.

Llamando a la variable objeto *opcda*, podemos ver en su parámetro *Status* que ahora estamos conectados, su valor ha cambiado.

```
Command Window
>> da

da =

Summary of OPC Data Access Client Object: localhost/deck.opc.server.1.0

   Server Parameters
      Host      : localhost
      ServerID  : deck.opc.server.1.0
      Status    : connected
      Timeout   : 10 seconds

   Object Parameters
      Group     : 0-by-1 dagroup object
      Event Log : 0 of 1000 events
```

Figura 46: Parámetros del objeto *opcda* una vez nos conectamos al servidor.

Función *disconnect.*

Sintaxis:

$$disconnect(variable\_objeto\_opcda)$$

Al contrario que el comando anterior, éste nos desconecta del servidor. Presenta una particularidad y es que no nos cierra la ventana creada al conectarnos al servidor (véase Figura 45).

En caso de cerrar la ventana sin desconectarnos del servidor, *MATLAB* nos indica el evento con el siguiente aviso:

```
Command Window

OPC ShutDown event occurred at local time 17:40:52
    Reason: Server shutdown
```

Figura 47: Aviso mostrado al cerrar la conexión con el servidor de manera imprevista.

Función *addgroup.*

Sintaxis:

$$Variable\_grupo\_objetos = addgroup(variable\_objeto\_opcda)$$

Esta función nos añade un grupo al objeto *opcda* con el fin de que el cliente manipule y organice elementos de datos. Si el objeto ya está conectado al servidor cuando se ejecuta esta función, se solicita un nombre de grupo al servidor y si no se proporciona (o no está conectado) se asigna automáticamente un nombre único ("*grupoN*" siendo N un número entero).

Si queremos saber la información del nuevo grupo añadido, *MATLAB* nos muestra la siguiente información:

```
Command Window
>> GrpObj

GrpObj =

Summary of OPC Data Access Group Object: GRP0001089C

   Object Parameters
      Group Type   : private
      Item         : 0-by-1 daitem object
      Parent       : localhost/deck.opc.server.1.0
      Update Rate  : 1
      Deadband     : 0%

   Object Status
      Active       : on
      Subscription : on
      Logging      : off

   Logging Parameters
      Records      : 120
      Duration     : at least 120 seconds
      Logging to   : memory
      Status       : Waiting for START.
                     0 records available for GETDATA/PEEKDATA
```

Figura 48: Información de una variable grupo añadido a través de *addgroup*.

Función *browsenamespace*.

Sintaxis:

$$Lista\_elementos = browsenamespace(variable\_objeto\_opcda)$$

Esta función nos abre el cuadro de diálogo de los nodos disponibles del servidor del objeto *opcda* (véase Figura 49). En esta ventana podemos construir una lista, concretamente un arreglo, de nodos la cual se devuelve a la variable *Lista_elementos*.

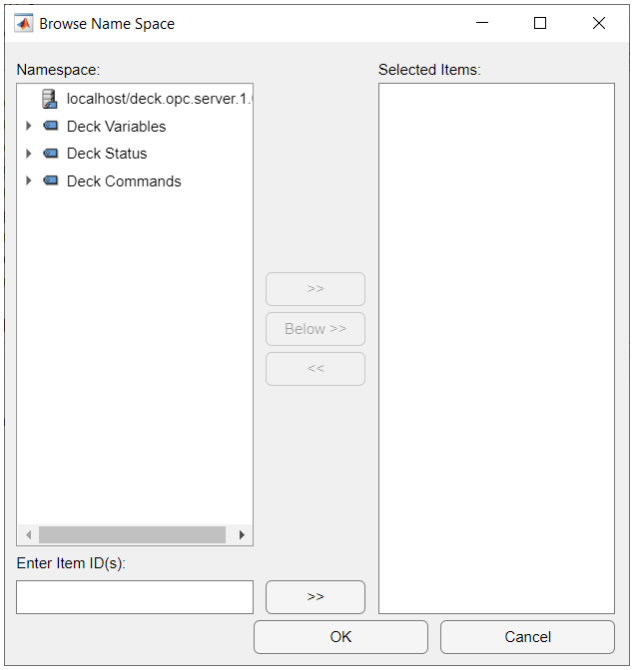


Figura 49: Aspecto de la ventana *Browse Name Space*.

Al finalizar nos muestra el contenido de nuestra variable '*lista_elementos*' (véase Figura 50).

```
Command Window
>> ItmList = browsenamespace(da)

ItmList =

  1×1 cell array

    {'Deck Variables.depo2_Altura'}
```

Figura 50: Variable resultado de la función *browsenamespace*.

Función *additem*.

Sintaxis:

$$Variable\_elemento = additem(variable\_grupo\_objetos, 'nombre\ variable')$$

El uso de esta función agrega el elemento dado por '*nombre_variable*' al grupo de objetos y crea el objeto '*variable_elemento*'. Al ejecutarla nos muestra información de la nueva variable objeto:

```
Command Window
>> IObj = additem(GrpObj,ItmList{1})

IObj =

Summary of OPC Data Access Item Object: Deck Variables.depo2_Altura

   Object Parameters
      Parent        : GRP00010894
      Access Rights : read

   Object Status
      Active        : on

   Data Parameters
      Data Type     : double
      Value         : 0
      Quality       : Good: Non-specific
      Timestamp     : 13-Oct-2023 17:19:56
```

Figura 51: Información de la variable grupo creada a través del comando *additem*.

Función *readValue* y *writeValue*.

Sintaxis:

$$Variable\_lectura = readValue(variable\_objeto\_opcda, 'nombre\ nodo')$$

$$writeValue(variable\_grupo\_opcda, nodo, 'nuevo\ valor')$$

Como es de suponer, las funciones se encargan de leer y escribir datos en los nodos, respectivamente. Aunque la *toolbox* nos proporcione estas funciones, podemos realizar la función de lectura con el comando *read* de *MATLAB* y escribir el valor con una asignación simple.

5.2.3 BLOQUES PARA ENTORNO SIMULINK

Industrial Communication Toolbox nos proporciona básicamente cuatro bloques que nos permitirán realizar todas las funciones vistas y necesarias para la conexión de una manera más intuitiva y sencilla de comprender.

Bloque OPC *Configuration*.

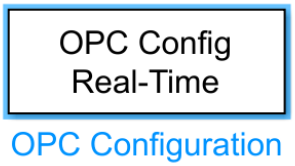


Figura 52: Aspecto del bloque *OPC Configuration*.

Se trata del bloque encargado de realizar la selección y conexión del servidor con la planta o modelo *SIMULINK*. Al abrirlo nos encontramos con la ventana de la Figura 53. Se nos permite controlar los parámetros relacionados con los clientes, errores, latencia…

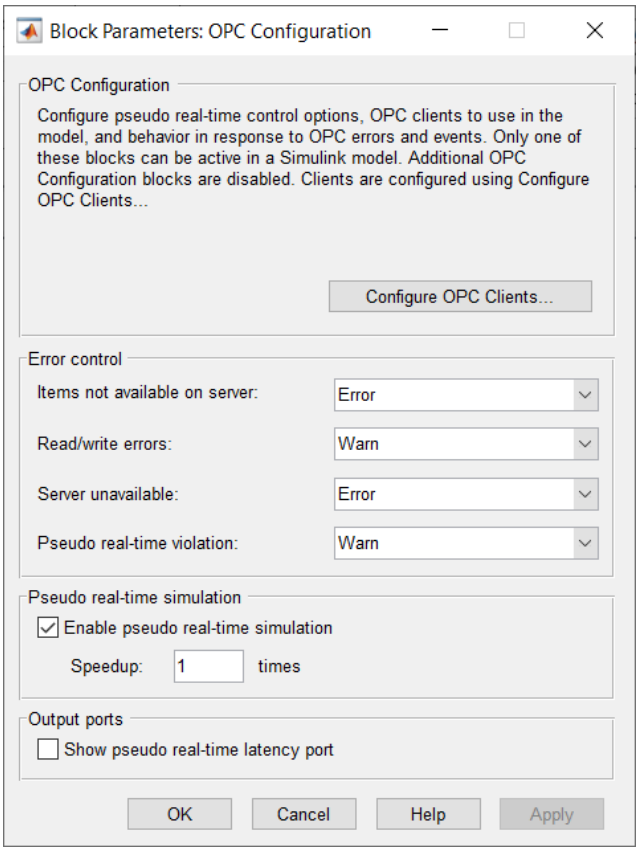


Figura 53: Ventana de parámetros del bloque *OPC Configuration*.

Accediendo al botón "*Configure OPC Clients*…" se nos abre el cuadro emergente de gestión de clientes (véase Figura 54). Dentro tenemos las opciones de agregar, editar y eliminar cliente; conectarnos y desconectarnos con el servidor.

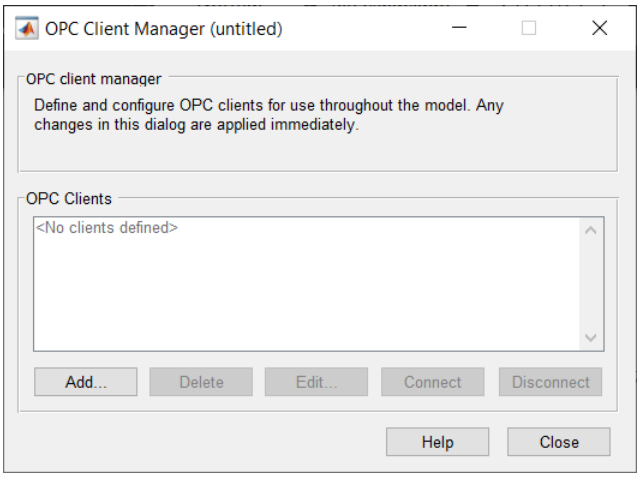


Figura 54: Cuadro emergente de gestión de clientes OPC.

Para agregar un servidor daremos clic en "*Add*…" y se nos abrirá otra ventana con los servidores disponibles (véase Figura 55). Al igual que en la función opcda, podemos definir el parámetro *Timeout* en segundos.

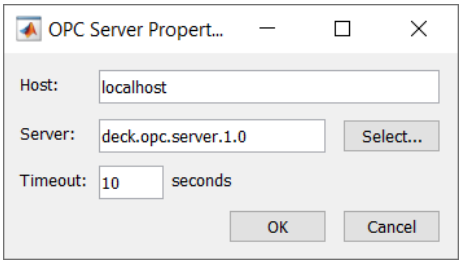


Figura 55: Ventana emergente de los servidores disponibles.

Una vez tenemos seleccionado un servidor, se iniciará automáticamente la conexión con éste y nos aparecerá como cliente disponible en la ventana de clientes OPC de la Figura 54.

Como veremos a continuación, la configuración del servidor también se puede realizar desde otros bloques, como el de lectura y escritura. A pesar ello, el bloque de *OPC Configuration* es imprescindible para que funcione el modelo, sin él no se establece la conexión con el servidor.

Comparando este bloque con las funciones vistas en el apartado 5.2.1 FUNCIONES PARA ENTORNO MATLAB, vemos que se agrupan las funciones *opcserverinfo*, *opcda*, *connect* y *disconnect* de una manera más fácil de entender y configurar.

Bloque OPC *Read*.

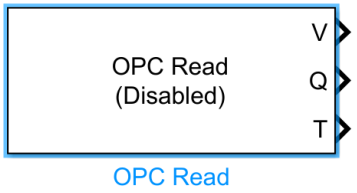


Figura 56: Aspecto del bloque *OPC Read*.

Este bloque tiene la función de leer los datos provenientes de un nodo determinado del servidor. Sus tres salidas son: *V* (valor del nodo), *Q* (puerto de calidad) y *T* (puerto de marca de tiempo). Al acceder a los parámetros del bloque nos encontramos con la siguiente ventana:

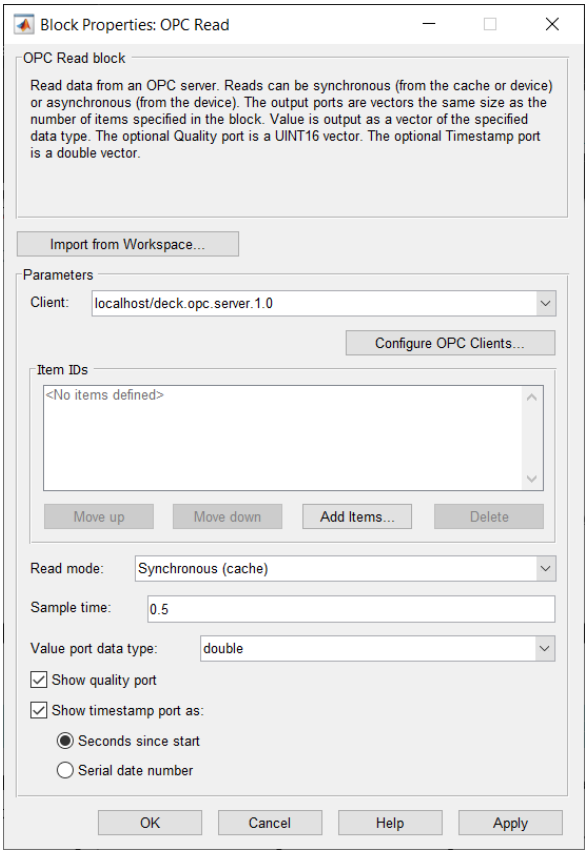


Figura 57: Ventana de parámetros del bloque *OPC Read*.

Entre sus parámetros, además de la elección de nodo, tenemos elección del tipo de lectura, tiempo de muestreo, tipo de dato de la lectura y mostrar/ocultar los puertos de calidad y tiempo. Este bloque permite importar la variable desde el *workspace* si la tenemos definida y, como dijimos en el apartado del bloque OPC *Configuration*, también nos ofrece la opción de elegir el servidor.

Para elegir un nodo de lectura, daremos clic al botón "*Add Items*…" y se nos abrirá la ventana emergente *Browse Name Space* como la vista en la Figura 49. El procedimiento es el mismo.

Comparando de nuevo este elemento con las funciones del apartado 5.2.1 FUNCIONES PARA ENTORNO MATLAB, vemos que con este bloque nos ahorraríamos las funciones de *addgroup*, *additem*, *browsenamespace* y *readValue*.

Bloque OPC *Write*.

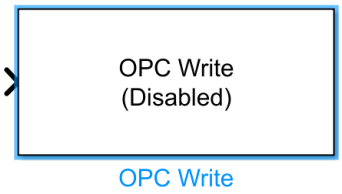


Figura 58: Aspecto del bloque *OPC Write*.

El uso de este bloque nos permitirá escribir un valor en un nodo determinado. Solamente tiene una entrada, que será la señal que lleve el valor que queramos dar a la variable del servidor. Abriendo el componente, nos encontramos con la siguiente ventana.

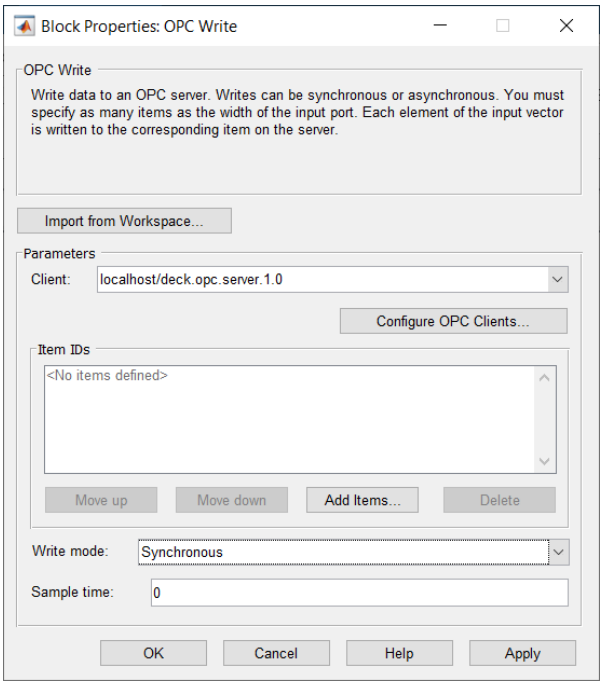


Figura 59: Ventana de parámetros del bloque *OPC Write*.

Entre sus parámetros tenemos, además de la elección del nodo de escritura, el modo de escritura y el tiempo de muestreo. Al igual que para el bloque *OPC Read*, este elemento permite escribir en una variable importada desde el workspace y, al igual que se indicó en el apartado del bloque OPC *Configuration*, también tenemos la opción de elegir el servidor.

Para elegir un nodo de escritura, daremos clic al botón "*Add Items*…" y se nos abrirá la ventana emergente *Browse Name Space* como la vista en la Figura 49. El procedimiento es el mismo.

Comparando de nuevo este elemento con las funciones del apartado 5.2.1 FUNCIONES PARA ENTORNO MATLAB, vemos que con este bloque nos ahorraríamos las funciones de *addgroup*, *additem*, *browsenamespace* y *writeValue*.

Bloque OPC *Quality Parts*.

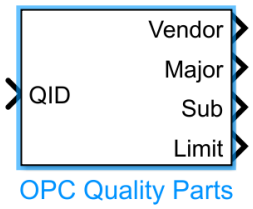


Figura 60: Aspecto del bloque *OPC Quality Parts*.

El bloque de partes de calidad OPC transforma un vector ID de calidad de acceso a datos OPC en cuatro partes:

- ***Vendor***: estado del proveedor. Muestra la información ID de calidad específica del proveedor. Es un entero entre 0 y 255.
- ***Major***: calidad mayor. Nos indica el valor de la calidad. Tiene tres posibles valores: 0 (calidad mala), 1 (incierta) o 3 (calidad buena).
- ***Sub***: subestado de calidad. Cada estado de calidad mayor tiene un subestado adicional que describe la calidad del valor con más detalle. La interpretación del valor del subestado depende de si su calidad principal es buena, incierta o mala. Es un entero entre 0 y 7.
- ***Limit***: estado límite del valor de calidad. Indica si el valor no está limitado (0), está fijado en un límite inferior (1), está fijado en un límite superior (2) o es constante (3).

El bloque OPC *Read* genera ID de calidad que muestra por el puerto *Q*. Pasando esta señal por este bloque obtendríamos la información detallada en este apartado.

CAPÍTULO 6: DESARROLLO DE LA INTERFAZ

El código fuente de la aplicación se puede encontrar en los anexos (véase ANEXO A2: CÓDIGO FUENTE). A su vez, cómo operar la interfaz, es decir, el manual de usuario también se explicará más adelante (véase 10.1 MANUAL DE USUARIO). Por estos motivos en este apartado solo se explicarán aquellos detalles de interés del desarrollo.

6.1 DISEÑO DE LA INTERFAZ

El objetivo del desarrollo de la aplicación es ofrecer una experiencia sencilla y entendible sobre el control predictivo DMC. Para ello se ha optado por el diseño a través de módulos, obligando a realizar un comportamiento casi secuencial al alumno/usuario. Con esto se ha buscado que el usuario no lleva a cabo acciones extrañas que puedan perjudicar el correcto funcionamiento del programa.

La separación por módulos se ha hecho a través del elemento *Tab Group* (véase 3.3.3 Componentes para creación de menús 'Figure Tools') separando la carga, o configuración, del modelo del control del sistema.

Dentro de cada módulo del elemento, se han separado los componentes con elementos *Panel* (véase 3.3.2 Componentes para organización 'Containers') según el grupo de acciones al que pertenecen.

Además, se ha incluido una barra de herramientas con el elemento *Menu Bar* (véase 3.3.3 Componentes para creación de menús 'Figure Tools') donde se incluyen las opciones más relevantes (como guardar el modelo o abrir el menú de ayuda) para favorecer el acceso a estas acciones al usuario.

Para facilitar la comprensión de las acciones necesarias y/o en ejecución, se dispone de un cuadro de texto informativo (*Edit field text*) y varias luces (*Lamp*) que tomarán diferentes colores para informar al usuario del estado de la aplicación.

Centrándonos en la manipulación, para evitar que el usuario se intente saltar algún paso o ejecute acciones no contempladas, la interfaz mantendrá inhabilitados aquellos componentes que, accionados antes de tiempo, perjudicarían el comportamiento o provocarían fallos. Dichos elementos se irán habilitando a medida que se completen ciertas acciones de configuración.

6.1.1 DISEÑO DEL MÓDULO DE CARGA DE MODELO

Tras iniciar la aplicación, la interfaz abre el módulo correspondiente a la configuración del modelo. Su aspecto se muestra a continuación (véase Figura 61).

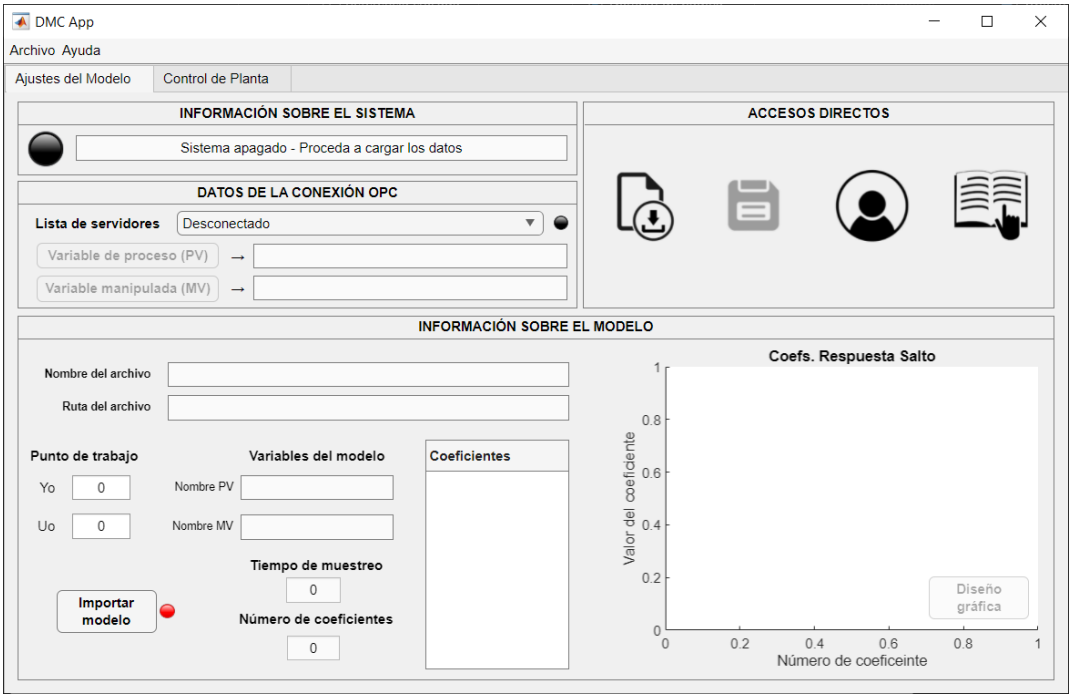


Figura 61: Diseño de la pestaña "*Ajustes del Modelo*".

Como se acaba de mencionar, esta pestaña es la encargada de mostrar al usuario las opciones de carga de modelo y selección del servidor OPC. Podemos distinguir cuatro zonas agrupadas en paneles:

- Información sobre el sistema.
- Datos de la conexión OPC.
- Información sobre el modelo.
- Accesos directos.

A continuación, explicaremos qué función o funciones tiene cada panel.

Panel: *Información sobre el sistema*.

Como su nombre indica el objetivo de este conjunto es informar al usuario de las acciones que se están llevando a cabo, los errores ocurridos y qué elementos le falta al sistema para iniciar el control.

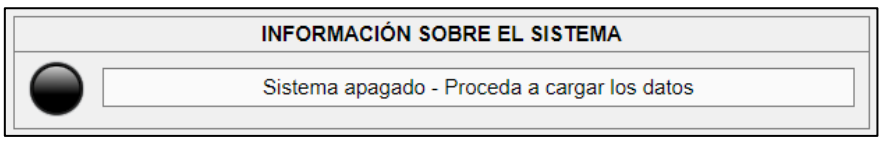


Figura 62: Aspecto del panel "*Información sobre el sistema*".

Básicamente se compone de dos elementos:

- Una lámpara que mostrará un color según el estado en el que se encuentre el sistema.
- Un campo de texto que mostrará mensajes según el estado del sistema, informando al usuario de qué le puede faltar para iniciar su configuración, que se está ejecutando…

Panel: *Datos de la conexión OPC.*

Este panel se encarga de realizar la conexión y configuración del servidor OPC al que nos conectaremos. Se elige el servidor a conectarse y los nodos del servidor que actuarán como variable de proceso y variable manipulada.

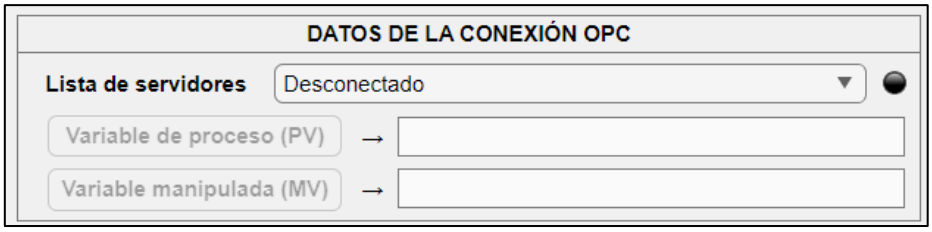


Figura 63: Aspecto del panel "*Datos de la conexión OPC*".

Como vemos en la Figura 63, tenemos los siguientes componentes:

- Un desplegable con la lista de servidores disponibles dentro del '*localhost*'. Al seleccionar un servidor determinado el sistema se conectará de forma automática y notificará al usuario cuando la conexión esté completada.
- Un botón para la selección de la variable de proceso. Al ejecutarlo se nos abrirá una ventana con los nodos disponibles dentro del servidor elegido. Una vez elijamos uno se mostrará en el campo informativo de la derecha, indicado con una flecha al lado del botón.
- Un botón para la selección de la variable manipulada. Mismo funcionamiento que el anterior.
- Una lampara indicadora del estado de la conexión con el servidor y de la configuración de las variables del sistema.

Cabe destacar que el sistema está pensado para evitar los fallos humanos que puedan surgir durante la configuración de la conexión. Entre estos errores más importantes destacamos los siguientes:

- Intento de conexión a un servidor no existente. Si el usuario se intenta conectar a un servidor que no permite conexión, por la razón que sea, el sistema avisará al usuario del fallo en la conexión y recuperará la conexión con el servidor anterior, en caso de haber estado conectado a uno previamente.
- Selección de dos o más nodos para una única variable. Bien sea por error o intentando buscar un funcionamiento erróneo, el sistema cancela la conexión con ese nodo y notifica al alumno de su falla en la selección.

- Selección de una variable de tipo distinto a *DOUBLE*. Para evitar problemas a la hora de escribir en un nodo, se ha optado por prescindir de los nodos que transmitan datos de tipo entero *INTEGER*. Esta restricción permite escribir en la variable manipulada valores con decimales logrando una precisión superior que usando enteros.

Además, el sistema contempla también que el usuario pueda realizar un cambio de servidor o variables, en caso de que fuera necesario.

Panel: *Información sobre el modelo*.

En este panel cargaremos el modelo sobre el que trabajará nuestro control y se mostrarán sus datos más relevantes. Recordemos que el control predictivo se debe hacer a partir de un modelo y en este panel cargaremos su archivo de identificación. El proceso de identificación y la generación del archivo a través de *HIDEN* se explican en el apartado 7.1 EXPERIMENTO DE IDENTIFICACIÓN.

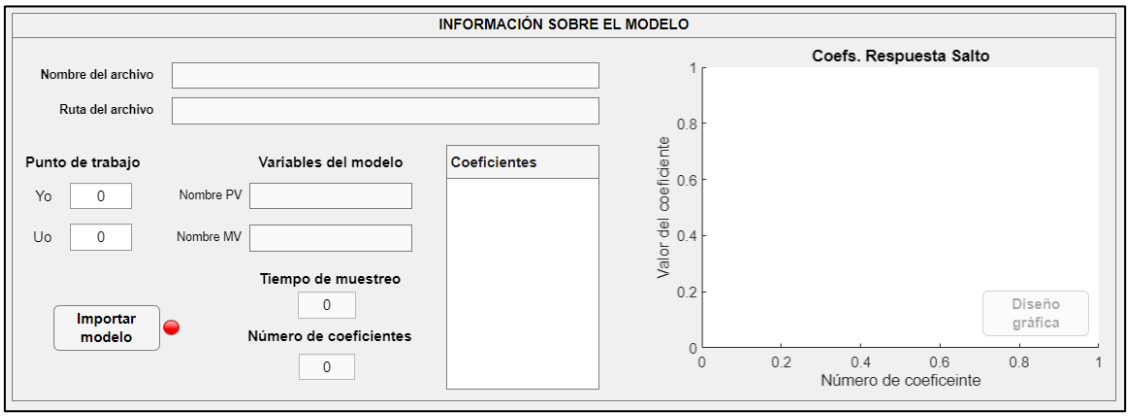


Figura 64: Aspecto del panel "*Información sobre el modelo*" en la vista de datos del modelo.

Básicamente, los datos del modelo que se muestran son: el nombre y la ruta donde está el archivo, el nombre de las variables que se han dado en el proceso de identificación y los coeficientes de la respuesta salto con su representación. Con esto, tenemos los siguientes elementos:

- Dos campos de texto que muestran el nombre y la ruta completa del archivo.
- Dos campos de texto que nos permiten introducir el punto de trabajo, necesario para que el sistema funcione correctamente.
- Dos campos de texto que muestran el nombre de las variables que se han establecido en *HIDEN*.
- Dos campos de texto numéricos que muestran el tiempo de muestreo y el número de coeficientes.
- Una tabla donde se muestran los coeficientes de la respuesta salto generados en el proceso de identificación.
- Una gráfica donde se representan estos coeficientes.
- Un botón (situado en la gráfica) para acceder a la personalización de la gráfica (véase Figura 65).

- Un botón que importa el archivo con el modelo (formato HIDEN).
- Una lámpara que nos indica el estado de la carga.

A la hora de cargar un archivo tenemos dos opciones: importar un modelo que no ha sido tratado por la interfaz aún o cargar la configuración de un archivo ya tratado por la aplicación. El procedimiento para los dos es el mismo: se nos abre una ventana de carga, seleccionamos el archivo deseado y el sistema carga los datos que éste tenga. Si queremos cargar un modelo no tratado, haremos clic en el botón "Importar modelo" de la Figura 64. Para cargar una configuración, deberemos acceder a la sección "Archivo" de la barra de herramientas o a través del primer icono del panel de accesos directo (véase Panel: Accesos directos.).

Cabe resaltar que el sistema contempla que el usuario intente cargar un archivo no compatible a través de la opción de cargar la configuración, en cuyo caso aparecerá una venta emergente avisando del fallo. Sin embargo, en caso de que se intente cargar una configuración a través del camino de importar un archivo sin tratar, el sistema importará los datos necesarios para empezar un control desde cero. Ante esta posible situación, el sistema muestra una ventana informativa avisando que se ha cargado un archivo de este tipo y que será necesario establecer aquellos campos que faltan.

Como valor añadido, se ha optado por añadir una pestaña de personalización de gráfica como se muestra en la Figura 65.

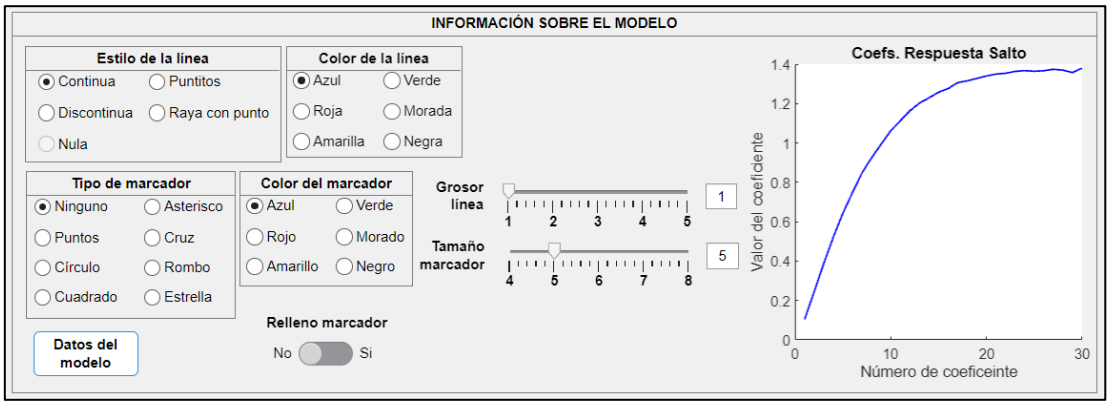


Figura 65: Aspecto del panel "*Información sobre el modelo*" en la vista del diseño de la gráfica.

Básicamente se nos cambia el aspecto visual del panel, ocultando los elementos mencionados anteriormente y mostrando unos nuevos que permiten tocar detalles como el color de la gráfica, poner marcadores en los coeficientes… Para regresar a la vista anterior haremos clic en el botón "Datos del modelo" ubicado en la parte inferior izquierda.

Panel: *Accesos directos.*

Para facilitar al usuario algunas acciones básicas (como guardar y cargar una configuración) se ha añadido este último panel del módulo de "Ajustes del modelo" una serie de iconos que hacen más visual este tipo de funciones.

Figura 66: Aspecto del panel "*Accesos directos*".

Básicamente está compuesto por cuatro imágenes que al accionarlas ejecutan una función. Estas imágenes son, de izquierda a derecha:

- Primer icono: cargar configuración. Como se indicó en el apartado Panel: Información sobre el modelo, este botón ejecuta el proceso de carga de una configuración de un archivo ya tratado.
- Segundo icono: guardar configuración. Además de poder guardar la configuración a través de la sección "Archivo" de la barra de herramientas, se ha dispuesto este icono para facilitar el proceso al usuario. Una vez iniciemos el proceso, se abrirá la ventana emergente de guardado donde podremos darle un nombre y guardarlo en la ruta deseada. Por defecto, el sistema asigna el siguiente nombre: "*Conf_DMC_fecha-actual_hora-actual*", mas el usuario puede modificarlo si lo considera necesario.
- Tercer icono: información del desarrollador. Al hacer clic en esta imagen, se abrirá una pestaña emergente con la información de la escuela, universidad y el desarrollador de la interfaz: Saúl Antolín.
- Cuarto icono: manual de usuario. Al hacer clic en esta última imagen, se abrirá otra pestaña emergente con las instrucciones de funcionamiento y manipulación del sistema. Este añadido es muy útil para que el alumno no se quede atascado en ningún punto de la aplicación.

6.1.2 DISEÑO DEL MÓDULO DE CONTROL DE PLANTA

Una vez tenemos cargada la configuración del módulo anterior ya podremos acceder a esta pestaña donde, como indica su nombre, trataremos los aspectos relacionados con el control de la planta. En la Figura 67 se muestra el aspecto general que tendrá la aplicación al acceder por primera vez a este módulo.

Al igual que en el módulo anterior, su diseño consta de tres paneles en este caso:

- Control de planta.
- Parámetros DMC.
- Representación de las señales.

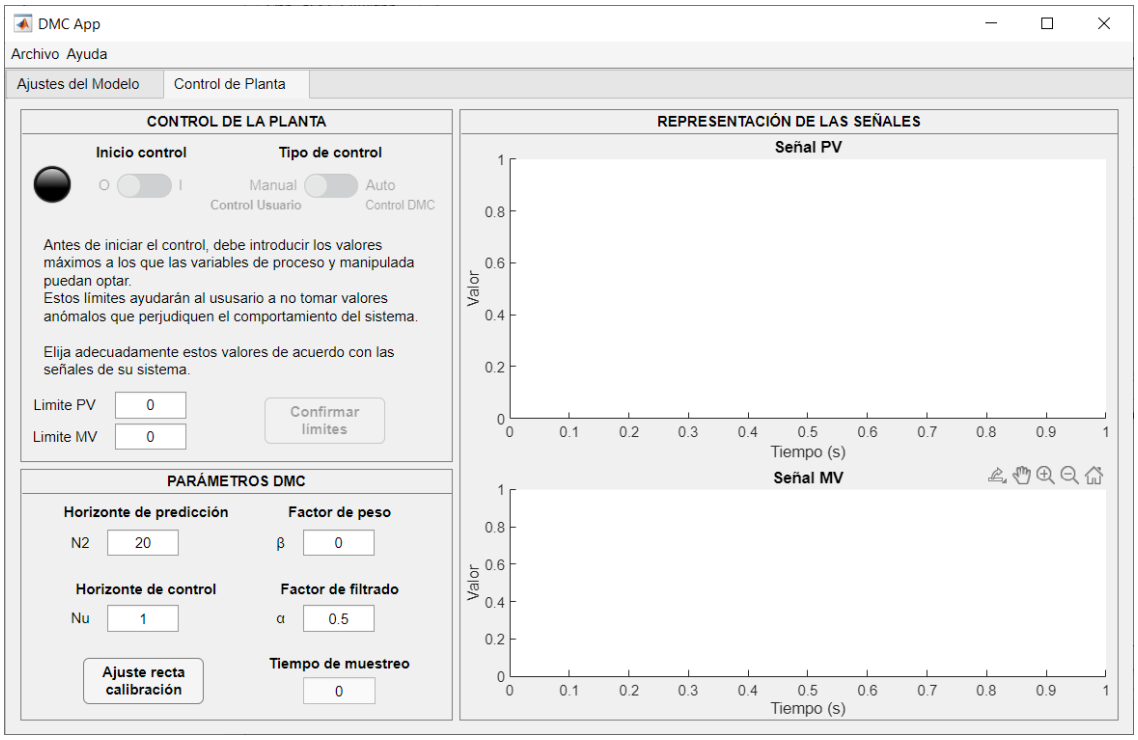


Figura 67: Diseño de la pestaña "*Control de planta*".

Antes de explicar cada conjunto, es conveniente aclarar que el usuario puede entrar a este módulo en cualquier momento, solo que no será capaz de manipular esta parte del interfaz al completo si no ha realizado la configuración previamente.

Panel: *Control de planta*.

En este bloque se agrupan las funciones primordiales para iniciar y detener el control, escoger un control manual o automático, establecer los valores de la variable manipulada, la referencia (o *setpoint*) en lazo cerrado…

Para empezar el aspecto del panel es el mostrado en la Figura 68.

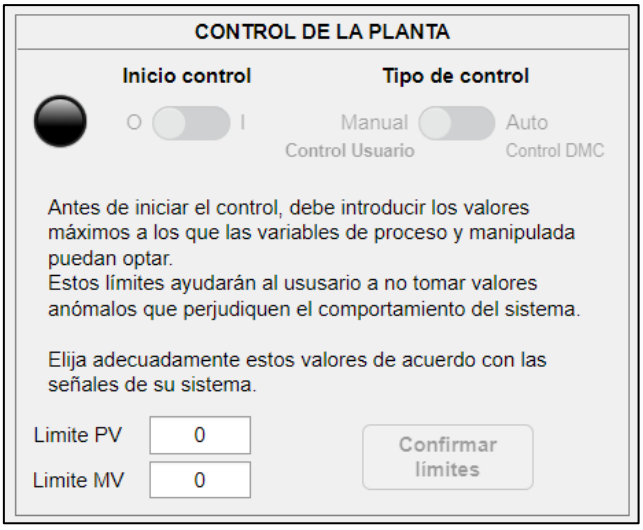


Figura 68: Aspecto inicial del panel "*Control de la planta*".

Como podemos ver tenemos varios elementos:

- Lámpara indicadora del estado del control del sistema. Se iluminará con diferentes colores para avisar al usuario de su estado.
- Interruptor para el inicio/pausa del control.
- Interruptor de selección de control manual/automático.
- Texto informativo del establecimiento de los límites de las variables.
- Dos campos numéricos para establecer el límite máximo de las variables.
- Botón de confirmación de límites del sistema.

Para que el sistema pueda establecer los cambios de manual a automático y viceversa, lo primero que se demanda al usuario es establecer los valores de los límites de las variables del sistema en los campos numéricos de la parte inferior izquierda del panel. Más adelante, en el apartado Panel: Parámetros DMC, se explica con detalle el funcionamiento de las unidades.

Una vez establecidos los límites, el botón se habilitará para, como su nombre indica, confirmar que los límites escritos son los deseados. El aspecto del panel cambia por completo, obteniendo la configuración mostrada en la Figura 69.

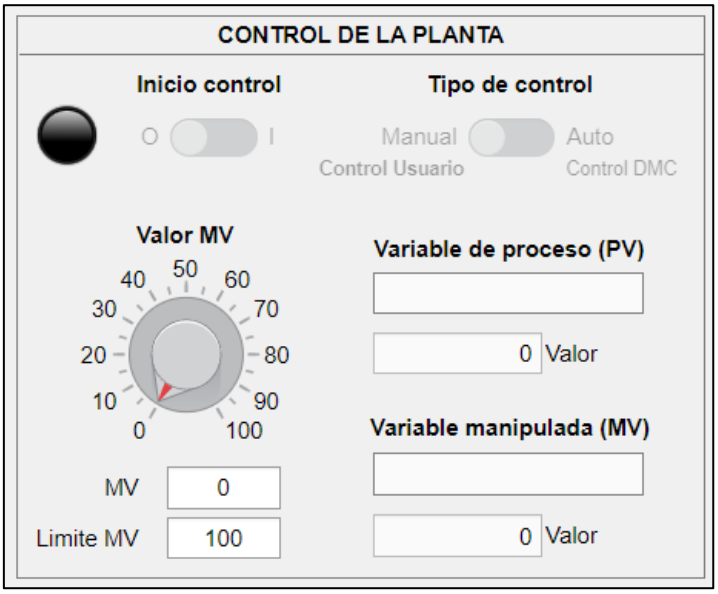


Figura 69: Aspecto del panel "*Control de la planta*" tras confirmar los límites de las variables.

Ahora, los elementos visibles son los siguientes:

- Lámpara indicadora del estado del control del sistema. Se iluminará con diferentes colores para avisar al usuario de su estado.
- Interruptor para el inicio/pausa del control.
- Interruptor de selección de control manual/automático.
- Cuadros de texto que informan del nombre de las variables del sistema y de su valor.
- Ruleta para establecer el valor de la variable manipulada en manual y de la consigna en automático.

- Cuadro numérico para ejercer la misma función que la ruleta, de manera más exacta.
- Campo numérico para cambiar el límite de la variable manipulada en caso necesario.

El funcionamiento ahora es muy sencillo. Lo primero es iniciar el control desde el interruptor "Inicio de control" y el sistema comenzará a tomar datos en manual a través de un temporizador que hace la gestión de lectura de nodos. Además, el temporizador de lectura realiza las siguientes gestiones: escribe los valores de las variables en los campos numéricos, representa las gráficas de estas señales y envía el histórico de las señales al *workspace* de *MATLAB* por si el usuario quisiera acceder a dichos valores a posteriori. Estas acciones se ejecutan cada segundo. Para explicar con brevedad, los temporizadores son funciones que se ejecutan en paralelo con el código normal, son como los hilos en programación.

Al cambiar a control automático, se activa además el temporizador del cálculo DMC. Simplemente reúne los datos necesarios para el cálculo de la señal de control y ejecuta la función vista en el apartado 4.1 CÓDIGO DEL ARCHIVO 'dmc.m'. Tras calcularlo lo escribe en el nodo. Este temporizador se ejecuta con un periodo igual al periodo de muestreo obtenido del archivo de datos.

Cabe destacar que solo se escriben los valores en los nodos cuando el control está encendido; si está apagado, no se modificará nada, aunque se nos permita manipular los elementos. Este valor se vuelve a calcular, si es necesario, y escribir una vez reanudemos el control.

Para el cambio de manual a automático se toma el último valor leído de la variable de proceso y se escribe como consigna. Para el cambio inverso, el último valor de la variable manipulada se mantiene como valor fijo, hasta que se regrese a automático o el usuario introduzca un nuevo valor. El objetivo de este funcionamiento busca suavizar el cambio de tipo de control (automático/manual) del sistema, evitando cambios bruscos en las señales.

Panel: *Parámetros DMC.*

Como el panel indica, este bloque permite al usuario establecer los valores de los parámetros para el cálculo del control predictivo (para más información ver el apartado 2.3.4 Parámetros del control DMC). Además, en este panel, el usuario podrá introducir los valores de las rectas de calibración de las señales de proceso. El diseño se muestra en la Figura 70. Podemos distinguir seis componentes:

- Cuatro campos numéricos editables correspondientes a los parámetros básicos del cálculo DMC (horizontes de predicción y control, y factores de peso y suavidad/filtrado)
- Un campo numérico, no editable, mostrando el tiempo de muestreo leído del fichero.

- Un botón para acceder a la ventana de ajuste de las rectas de calibración.

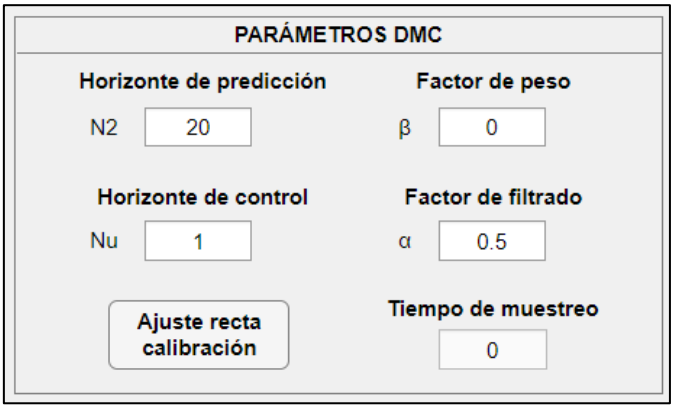


Figura 70: Aspecto del bloque *"Parámetros DMC"*.

Una vez el usuario cambie uno de los parámetros del control, el sistema lo escribe en la variable correspondiente y automáticamente se aplica en los siguientes cálculos. Aunque el sistema esté en manual o apagado, el alumno puede cambiar los valores de estos parámetros, ya que no se reflejarán en el sistema hasta que el temporizador del cálculo DMC actúe.

Dando clic al botón para el cambio de vista accedemos a la gestión de los parámetros de las rectas de ajuste (véase Figura 71).

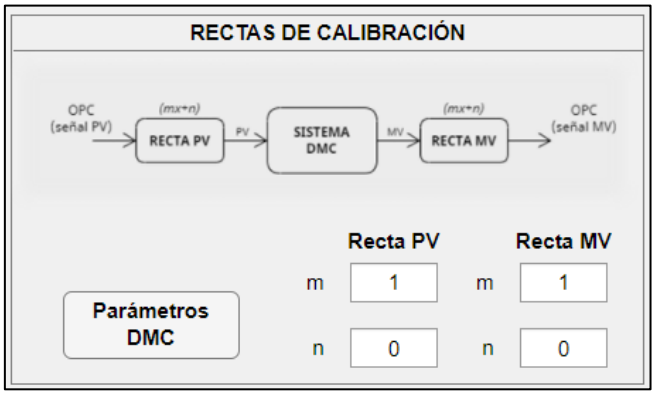


Figura 71: Aspecto del panel *"Rectas de calibración"*.

Ahora diferenciamos los siguientes elementos:

- Campo numérico con el valor de la pendiente de la recta para la variable de proceso.
- Campo numérico con el valor para la ordenada en el origen de la recta para la variable de proceso.
- Los dos elementos anteriores también para la variable manipulada.
- Botón de regreso a la vista de los parámetros DMC.
- Imagen explicativa de la gestión de las rectas.

Al igual que en los parámetros, los campos son editables en todo momento y los valores se aplican de manera instantánea.

Para entender el comportamiento analizaremos la imagen mostrada en el panel (ver Figura 72).

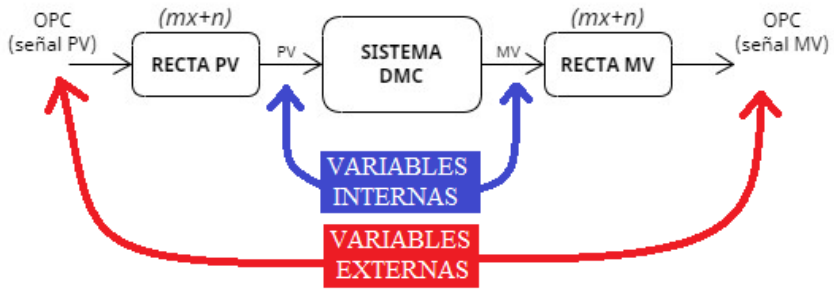


Figura 72: Esquema de tratamiento de señales del sistema.

Como podemos ver, distinguimos dos grupos de señales: internas en azul y externas en rojo. Las variables externas son los nodos del OPC, de donde leemos y escribimos los datos (pueden estar en voltios, miliamperios, etc.). Las internas son la adaptación que hacemos en el sistema para que funcione de manera que el usuario sea capaz de entender de una manera más sencilla desde el punto de vista de unidades de ingeniería (altura del depósito, temperatura, potencia de la bomba en %, etc.).

Los datos leídos de la variable de proceso del servidor se pasan por una recta de calibración para que el sistema trate con ellos, obteniendo así la variable de proceso interna con las unidades del sistema. Este valor interno pasa por el sistema y devuelve un valor para la variable manipulada:

- En **manual**, el usuario establecería el valor de la variable manipulada en unidades internas.
- En **automático**, el valor de la variable manipulada es calculado a través del control DMC.

Para escribir esta variable en el nodo, lo tenemos que pasar por otra recta de calibración, puesto que el sistema trata con las unidades internas. Como resultado obtenemos el valor que se va a escribir en el nodo con las unidades adecuadas al servidor.

Cabe resaltar que esta programación se ha llevado a cabo para que el usuario entienda de manera sencilla el sistema. Por ejemplo, basándonos en la planta real donde se realizarán los experimentos, suponemos que nuestros nodos son el nivel del segundo depósito como variable de proceso y la potencia de la bomba como variable manipulada.

El usuario podría trabajar con las unidades de los nodos (por ejemplo, voltios) y establecer el límite máximo en 5, pero a la hora de controlar la planta se estaría trabajando en unidades brutas y no se podría saber a ciencia cuanto se ha llenado el nivel (tendríamos la señal en voltios) ni cuanta potencia se está suministrando a la bomba. En cambio, si establecemos los límites en 100, entenderíamos mejor tanto el nivel del depósito como la potencia de la bomba, facilitando la comprensión del usuario al manejar unidades en

porcentaje. De la misma forma, se podría ajustar los parámetros de la recta para que la señal de nivel que ve el usuario esté en centímetros.

Es muy importante que el experimento de identificación (ver 7.1.2 PROCESO DE IDENTIFICACIÓN) se haga con las unidades con las que se quiere trabajar en el control predictivo, ya que deben coincidir las unidades del modelo con las unidades que maneje el usuario en la interfaz. Así lograremos un funcionamiento correcto del control.

Panel: *Representación de las señales*.

En este bloque se grafican las señales del sistema: valores de los nodos del servidor y el valor de la consigna, esta última solamente cuando se está en control automático.

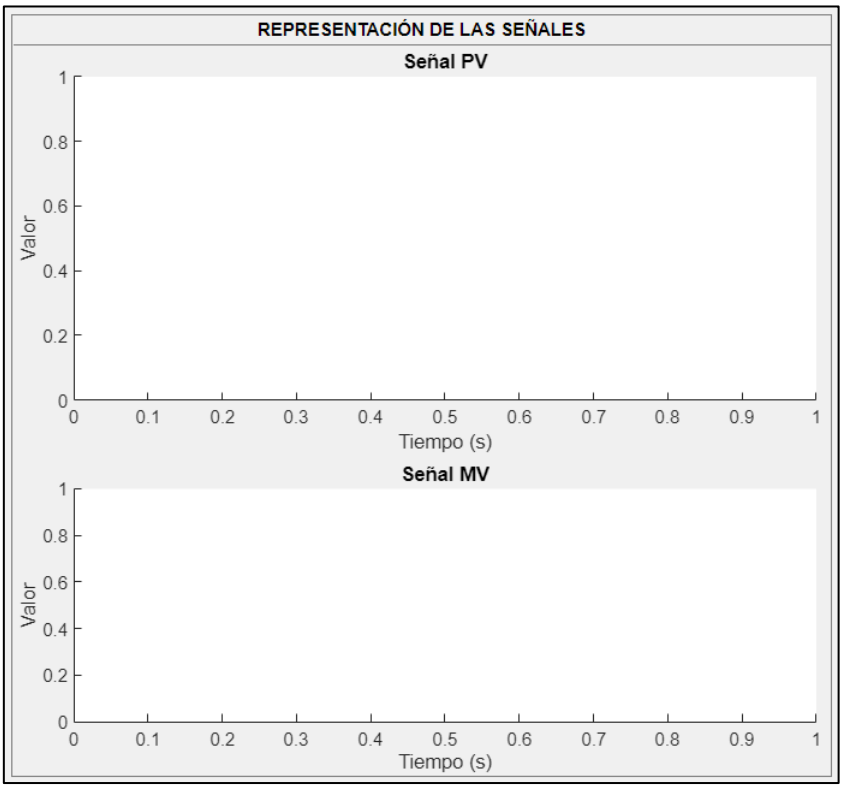


Figura 73: Panel "*Representación de las señales*".

Este panel está compuesto de dos gráficas:

- La gráfica superior representa los valores de la variable de proceso. En modo automático también representa la consigna y el usuario puede distinguir de manera más sencilla cuánto se separa la variable de proceso de la referencia.
- La gráfica inferior representa los valores de la variable manipulada.

Para la señal de la referencia en control manual, se otorga el valor de la variable de proceso, así se logra saber en qué instante se hizo el cambio del tipo de control. En automático, se muestra en los ejes superiores la leyenda con los nombres de las señales para facilitar la comprensión.

Para evitar cargar las gráficas con un número elevado de mediciones, las gráficas representan los últimos cien valores leídos. El histórico de estas señales se guardan en variables que se envían al *workspace* (en el apartado Panel: Control de planta se explica este temporizador con más detalle). Con este comportamiento, logramos además que el usuario sepa el tiempo de simulación observando el eje horizontal de cualquiera de las gráficas.

CAPÍTULO 7: VALIDACIÓN DE LA APLICACIÓN

Antes de iniciar la evaluación de la interfaz y comprobar que funciona correctamente, necesitamos hacer una identificación del sistema a controlar. A continuación, desarrollaremos la metodología para realizar un experimento de identificación [4].

7.1 EXPERIMENTO DE IDENTIFICACIÓN

7.1.1 INTRODUCCIÓN

La identificación de sistemas trata de los estudios de las técnicas que buscan la obtención de modelos matemáticos de sistemas dinámicos a base de realizar mediciones en las variables de control, manipuladas y perturbaciones del proceso. El enfoque se puede realizar en función de la estructura del modelo y su comportamiento. Podemos distinguir varios modelos:

- **Modelos de caja negra o Black-box**. Los parámetros del modelo no tienen interpretación física. Se postula una relación entre entrada/salida que depende de unos parámetros que deben ser estimados mediante datos experimentales, sin que dicha relación deba estar fundamentada en leyes. No se necesita un conocimiento del proceso.
- **Modelos de caja gris o Gray-box**. Son modelos de conocimiento en los que algunas partes del sistema son modeladas basándose en principios funcionales y otras como modelo de caja negra, la cual representa ciertos fenómenos difíciles de modelar de otra forma.
- **Modelo de caja blanca o White-box**. Los parámetros tienen una interpretación física. La estructura del modelo se obtiene a partir de leyes fundamentales.

En cuanto a los modelos matemáticos, realizaremos la siguiente clasificación:

- **Modelos lineales**. El sistema se puede definir por una función matemática lineal. La salida depende linealmente de las entradas y salidas en instantes anteriores. Las variables de proceso y manipulada son cambios sobre un punto de trabajo y el rango de validez está limitado en torno a ese punto. Dentro de este grupo podemos distinguir modelos paramétricos (función de transferencia, ecuaciones diferenciales y de diferencias…) y no paramétricos (respuesta salto, respuesta en frecuencia…).

- **Modelos no lineales**. El sistema no puede definirse por una función matemática lineal. Dentro de este grupo tenemos modelos como el modelo de Hammerstein, redes neuronales, modelos basados en leyes fisicoquímicas en general…

7.1.2 PROCESO DE IDENTIFICACIÓN

A. Toma de datos.

Antes de comenzar, veo conveniente aclarar que el proceso de identificación se va a explicar para el modelo simulado, para el real se realizará siguiendo el mismo procedimiento. Por tanto, las variables se presentarán en porcentaje.

La planta sobre la que se realizarán los procesos se detalla en el apartado 1.1 Introducción.

Para realizar el experimento, podemos seguir los siguientes pasos:

1. **Punto de trabajo (U_0, Y_0)**. Establecemos un estado de la planta como origen del experimento, en el cual los niveles estén estables para un determinado porcentaje de potencia entregada a la bomba. En el caso de nuestra planta se ha considerado establecer este punto alejado de los extremos para evitar la saturación de las variables. Entonces, para un caudal de entrada de 0.35 m^3/s, el nivel del depósito se ha establecido en 0.2907 *cm* aproximadamente.
2. **Periodo de muestreo a partir del tiempo de establecimiento**. El tiempo de establecimiento es el tiempo que tarda el modelo en estabilizarse dentro de unos márgenes ante un salto en la entrada (concretamente $\pm$ 2% del valor en estacionario). En nuestro caso el tiempo de establecimiento es de 380 segundos (ver Figura 74), por lo que podemos tomar el periodo de muestreo T como una veinteava parte del anterior. Por lo tanto, T = 19 segundos.

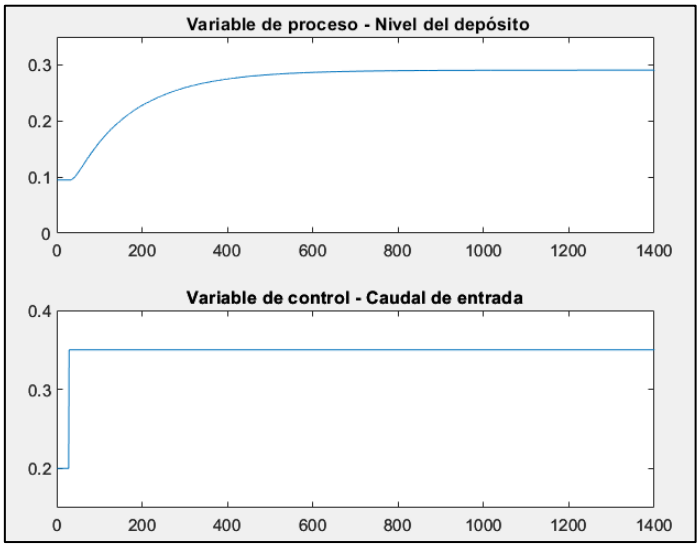


Figura 74: Respuesta del sistema simulado ante un salto.

3. **Amplitud de los saltos**. Una vez es estable el nivel de los depósitos con un determinado caudal de entrada, hay que aplicarles saltos más rápidos y lentos para excitar todas las frecuencias. Estos saltos deben de ser entre los mismos valores para que el estudio sea correcto, pero no debe de haber ni mucha ni poca diferencia para que los cambios sean correctos y notables. En nuestro caso hemos decidido que exista una amplitud de 0.15 m^3/s en el caudal suministrado por la bomba. Por lo tanto, oscilará entre valores de 0.5 m^3/s y el 0.2 m^3/s .
4. **Protocolo de velocidad de cambios**. Como se ha comentado, el objetivo es excitar todas las frecuencias con cambios de la amplitud establecida con tiempos cada vez más pequeños. Se ha comenzado con saltos de 760 segundos, disminuyendo a la mitad los tiempos cada dos saltos hasta llegar a 12 segundos, donde el sistema ya no se es capaz de responder a los cambios demandados.

Una vez realizado el experimento, obtenemos la gráfica de la Figura 75 donde podemos ver que los niveles del depósito han variado entre un 0.5 y un 0.095 acercándose cada vez más estos valores a 0.25 a altas frecuencias de cambio.

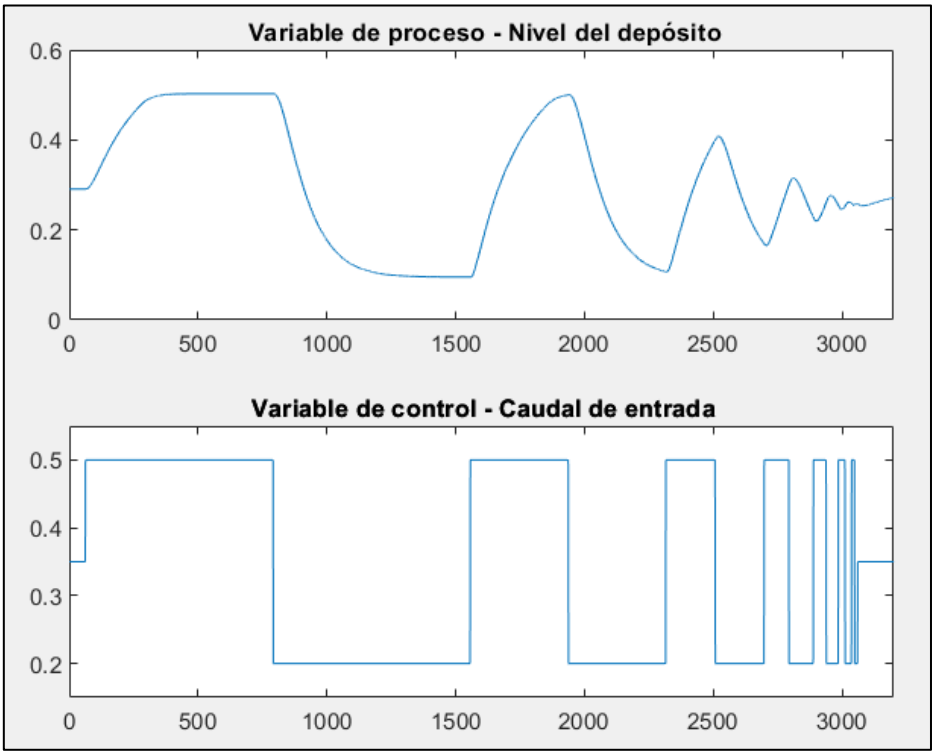


Figura 75: Resultado del experimento de identificación de la planta simulada.

Lo siguiente que hay que hacer es sacar el modelo del sistema para nuestro control DMC. Para ello usaremos *HIDEN*. Recordemos que para el control predictivo buscamos el modelo de incrementos respecto al punto de trabajo, por eso primero lo restamos a nuestras señales. Tras esta operación las señales quedan de la siguiente manera:

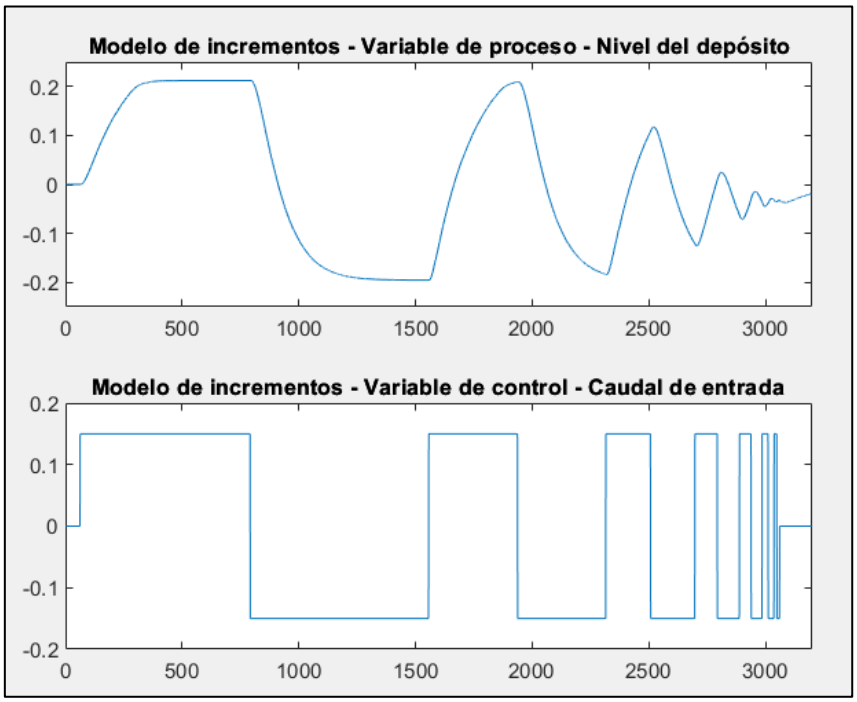


Figura 76: Modelo de incrementos del experimento de identificación.

HIDEN trata con archivos *ASCII* (entre otros), así que preparamos el resultado de la identificación en este formato y lo abriremos en la nueva herramienta (véase Figura 77 y Figura 78).

Una vez incorporadas, HIDEN nos muestra las señales como se muestra en la Figura 79.

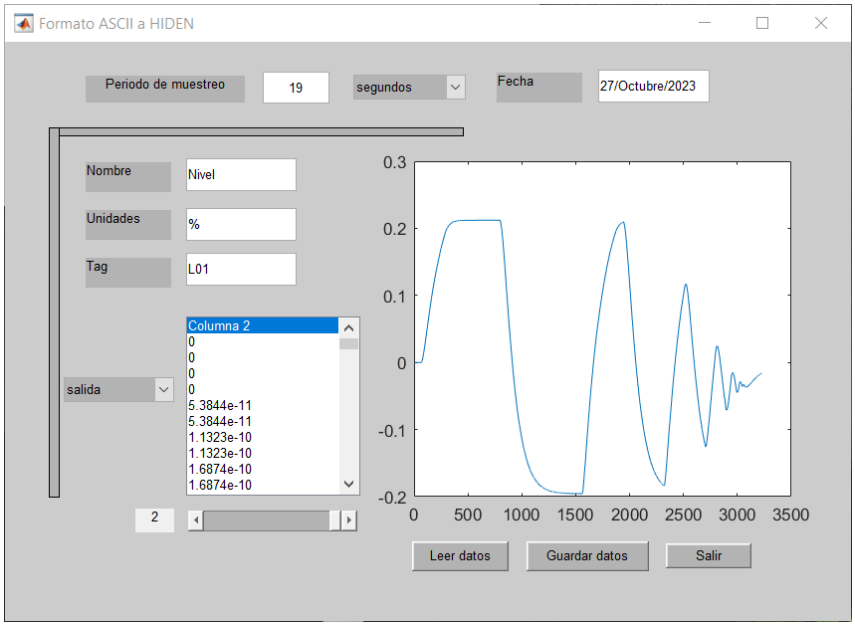


Figura 77: Incorporación de la señal del nivel a *HIDEN*.

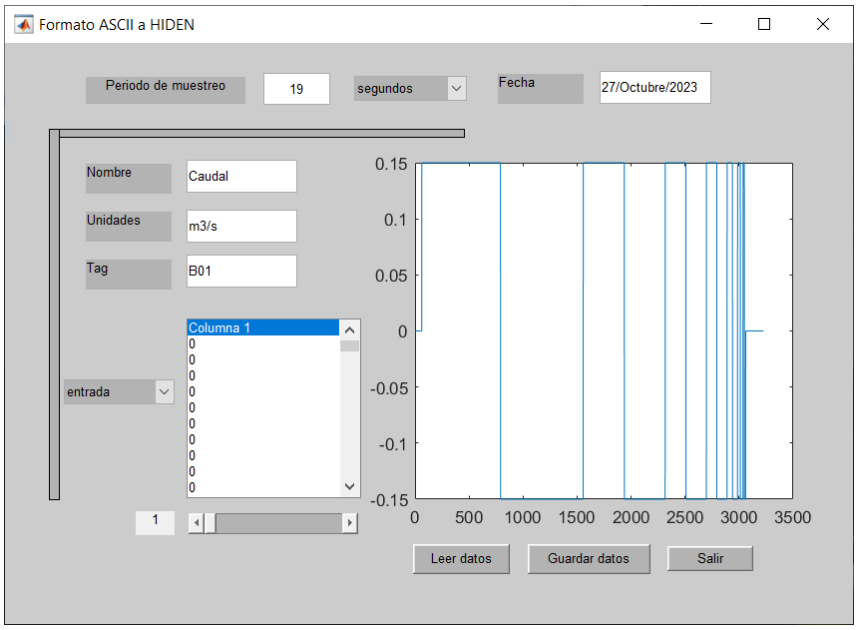


Figura 78: Incorporación de la señal de caudal de entrada a *HIDEN*.

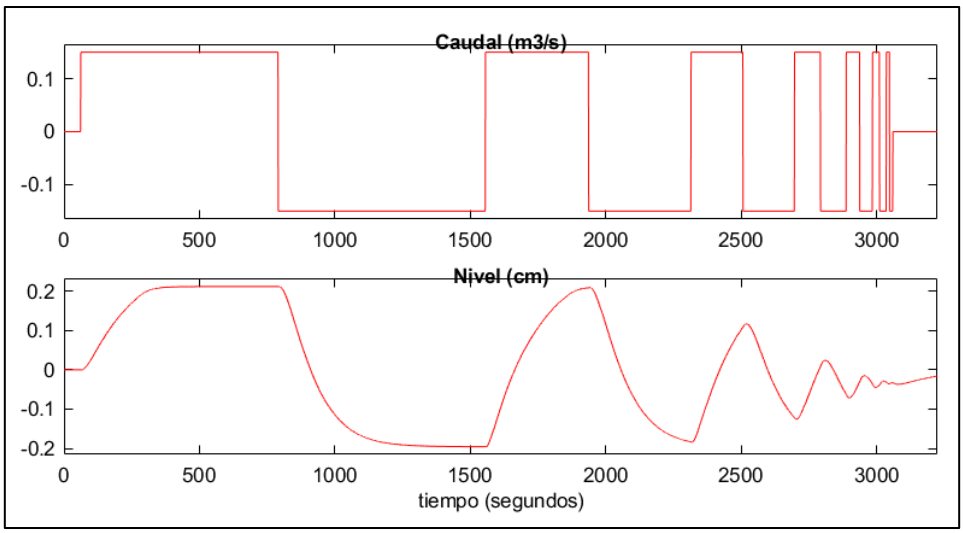


Figura 79: Señales del experimento de identificación en *HIDEN*.

B. Tratamiento de señales.

Una vez incorporadas las señales, tenemos que tratarlas para acondicionarlas para la identificación. En este caso he decidido aplicarle los siguientes tratamientos:

- **Nuevo tiempo de muestreo**. Con el objetivo de reducir ruido en las señales y hacer que control actúe con más frecuencia, podemos aplicar al experimento un nuevo tiempo de muestreo. Estableciendo un nuevo periodo de 10 segundos conseguimos una mejora en la identificación sin perder información (véase Figura 80). Lograremos así reducir el tiempo de respuesta en lazo cerrado.

Cabe destacar que *HIDEN* ofrece también otros tratamientos útiles como el filtrado pasa alta (que no se ha usado porque ya eliminamos el nivel de continua al restar el punto de

trabajo a las señales tras la toma de datos) o a pasa baja (tampoco se ha usado porque, al ser una simulación, los datos no contienen ruido de medida).

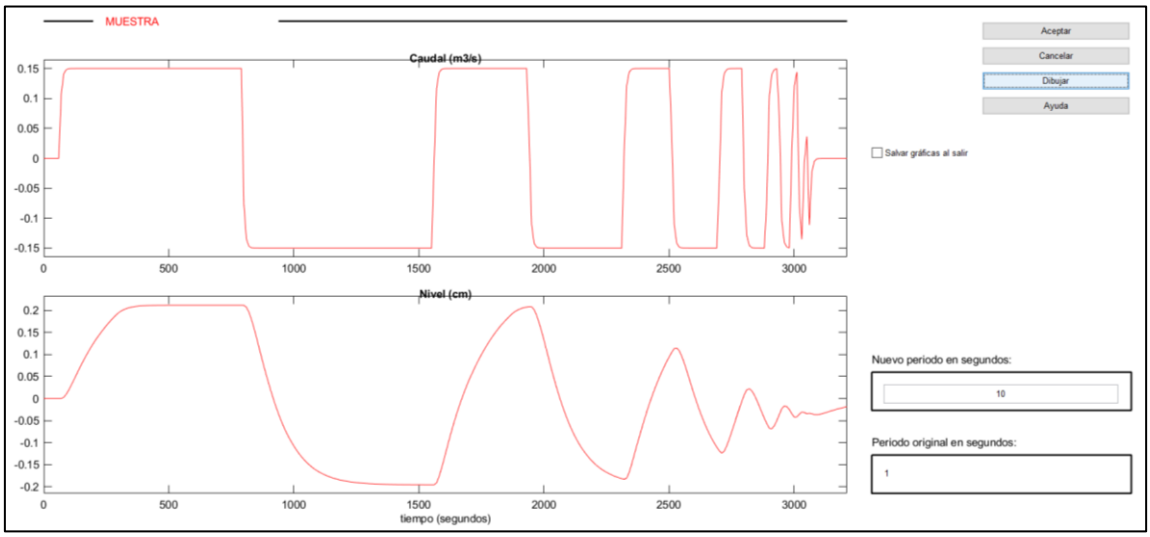


Figura 80: Señales tras aplicarles un nuevo periodo de muestreo.

C. Identificación del sistema.

Para acabar solo nos faltaría elegir la estructura del modelo y su método de identificación. En este caso elegiremos la estructura de respuesta escalón con 30 coeficientes (Figura 81) y el método de mínimos cuadrados como identificación (Figura 82).

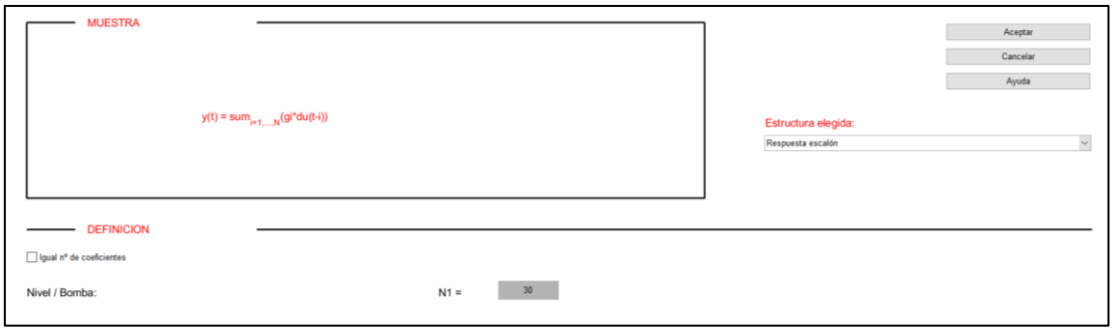


Figura 81: Ventana de selección de estructuras de *HIDEN*.

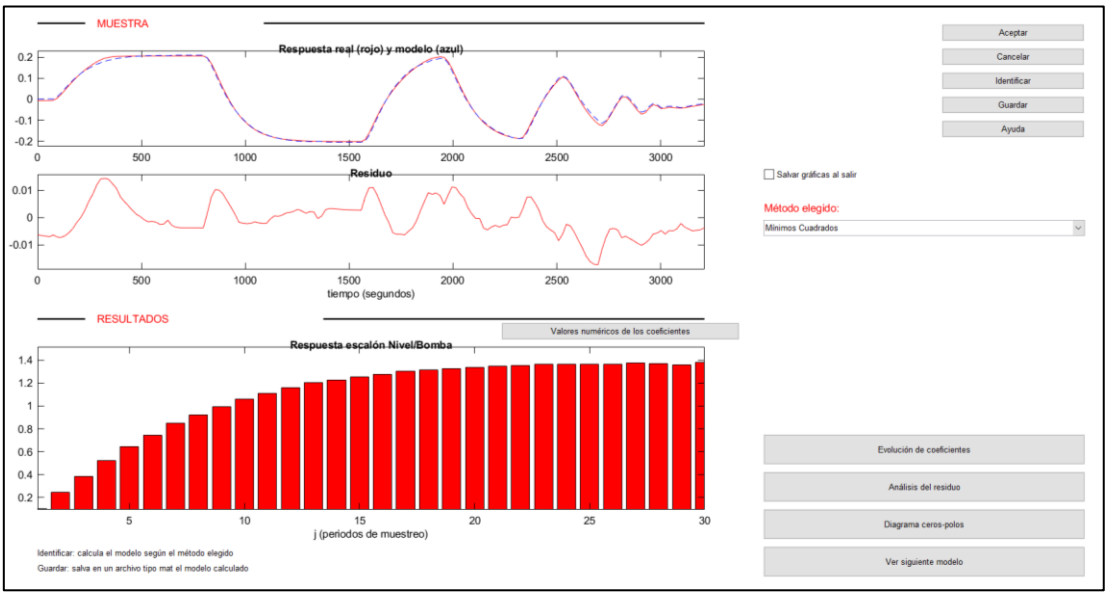


Figura 82: Ventana de selección de método de identificación.

Como podemos ver, el sistema se real se ajusta a la perfección al modelo sin apenas error, por lo que la identificación es correcta. No obstante, faltaría validar el modelo sobre otro conjunto distinto de datos. Guardamos el archivo y lo llevamos a *MATLAB*. El resultado es un archivo formato *MATLAB* que contiene toda la información que luego usará la aplicación, como el número de coeficientes y su valor, el periodo de muestreo, el nombre las variables y sus unidades…

7.2 VERIFICACIÓN DEL FUNCIONAMIENTO

Para comprobar el funcionamiento de las plantas, se realizarán varios saltos en las entradas y veremos cómo responde el sistema; también se cambiarán los valores de los parámetros del control predictivo DMC para analizar los cambios en la respuesta de salida.

Primero empezaremos por el sistema simulado y después la planta real.

7.2.1 MODELO DE SIMULACIÓN

Comenzaremos realizando varios saltos con los siguientes valores para los parámetros DMC (véase Figura 83):

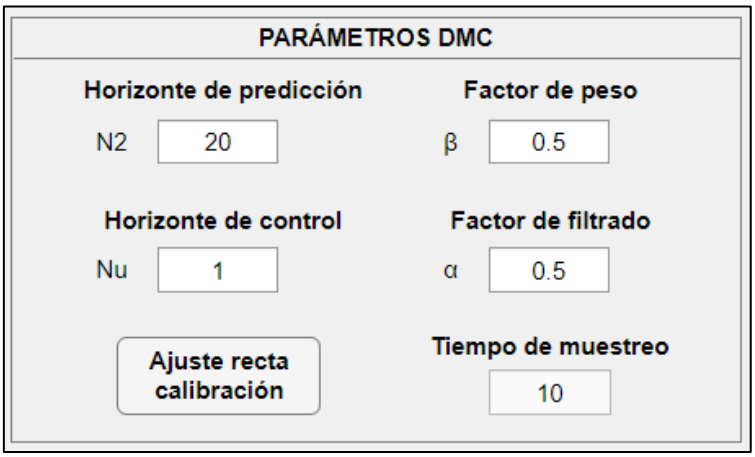


Figura 83: Valores de los parámetros DMC para el experimento de verificación de la planta simulada.

A continuación, se muestran varias imágenes con los diferentes saltos: Figura 84, Figura 85 y Figura 86.

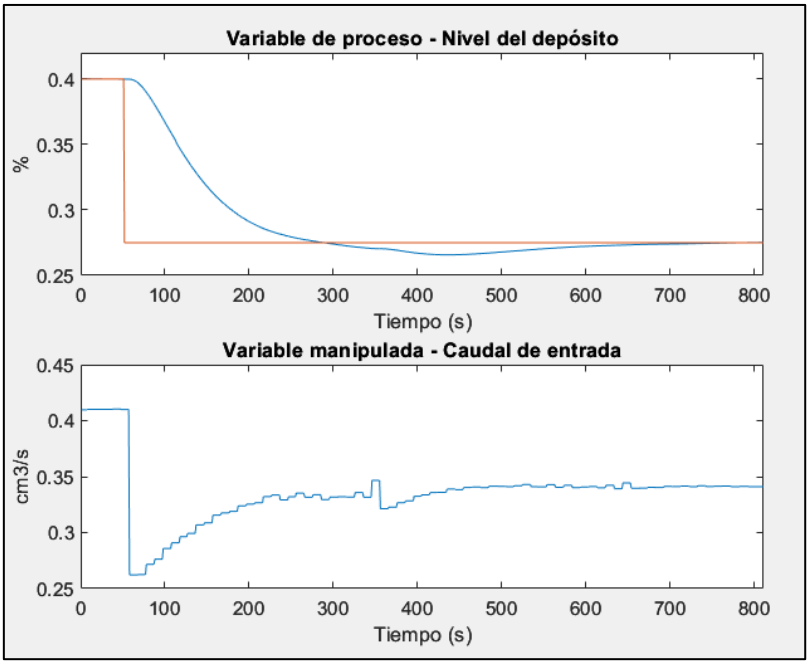


Figura 84: Salto del sistema ante un cambio en la referencia de 0.4 *cm* a 0.275 *cm*.

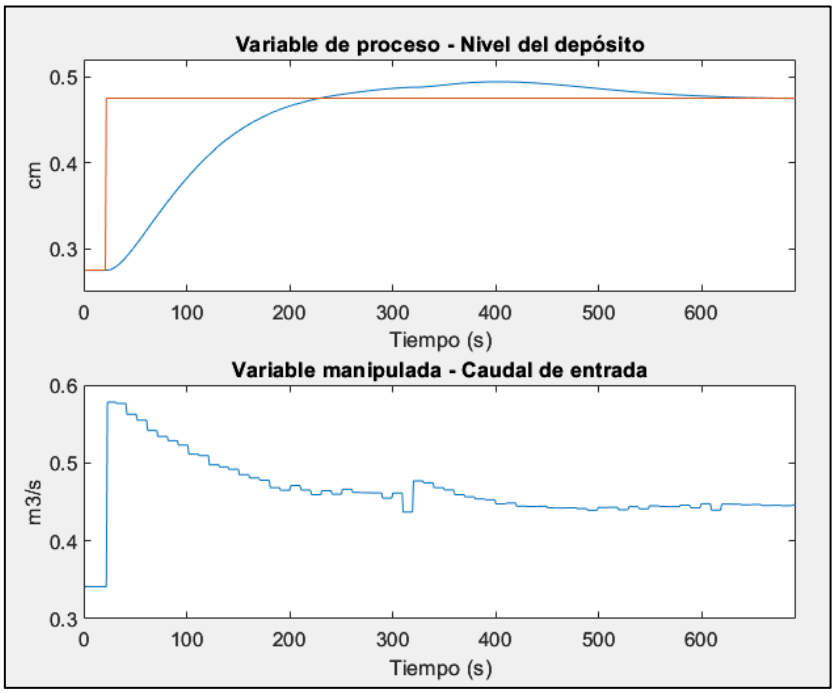


Figura 85: Salto del sistema ante un cambio en la referencia de 0.275 *cm* a 0.475 *cm*.

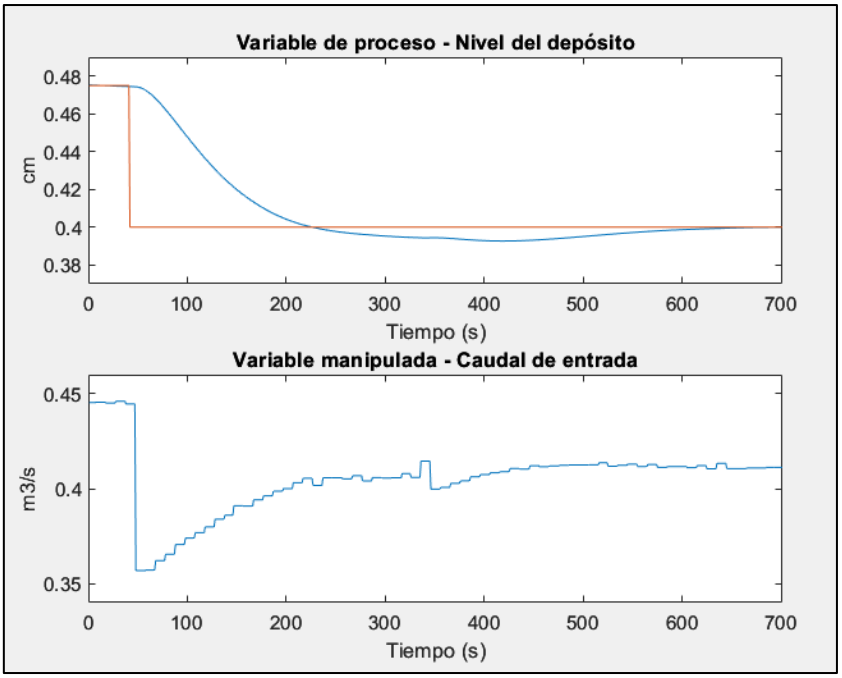


Figura 86: Salto del sistema ante un cambio en la referencia de 0.475 *cm* a 0.4 *cm*.

Como podemos ver, el sistema responde bastante bien. Es lento, pero acaba llegando a la referencia sin problema.

Se ha logrado el objetivo de control. Veremos ahora qué sucede al usar distintos valores en los parámetros del control DMC, primero aplicaremos un valor superior y luego uno inferior a cada parámetro. Para poder comparar resultados, se toma como base el salto de la Figura 84.

Cambios en el parámetro: HORIZONTE DE PREDICCIÓN *N2*.

Aumentando el valor del horizonte de predicción a **50** (véase Figura 87), obtenemos la respuesta vista en la Figura 88.

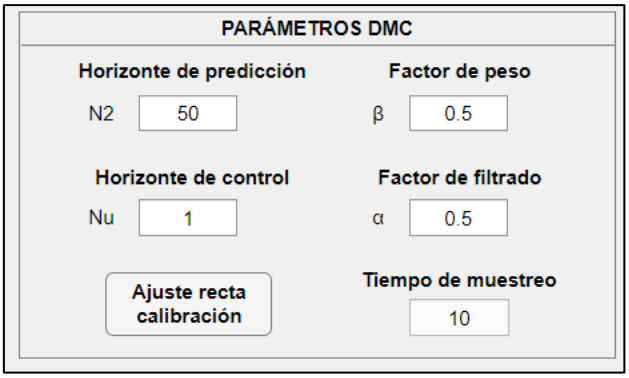


Figura 87: Cambio en el horizonte de predicción.

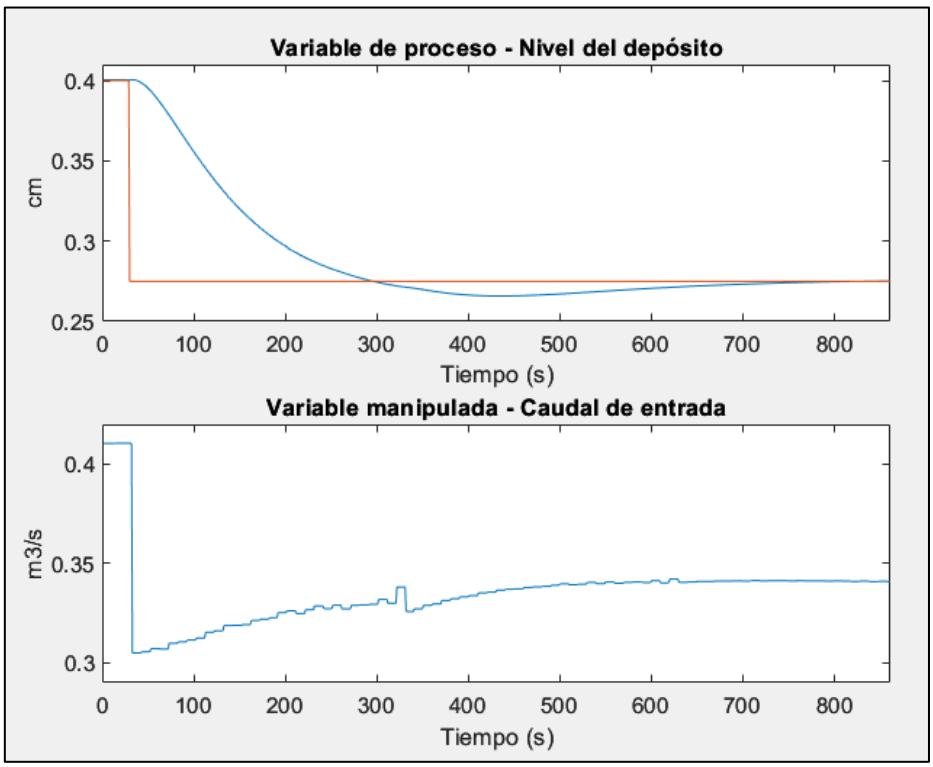


Figura 88: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 50 para *N2*.

Como era de esperar, el sistema funciona correctamente. Aumentando el valor por encima del predeterminado, no mejora la respuesta (ver apartado 2.3.4 Parámetros del control DMC) porque el control ya tiene en cuenta la referencia y estos nuevos valores que se añaden no aportan información nueva. La respuesta se vuelve más lenta ya que se considera toda la dinámica del sistema y, por tanto, los errores mínimos de los últimos coeficientes.

Ahora, se cambiará el valor de *N2* a 5 (ver Figura 89) y analizamos qué ocurre para el mismo salto.

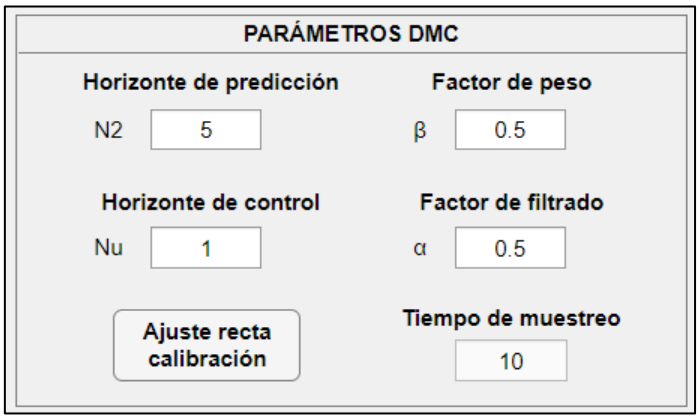


Figura 89: Cambio de valor en el horizonte de predicción a 5.

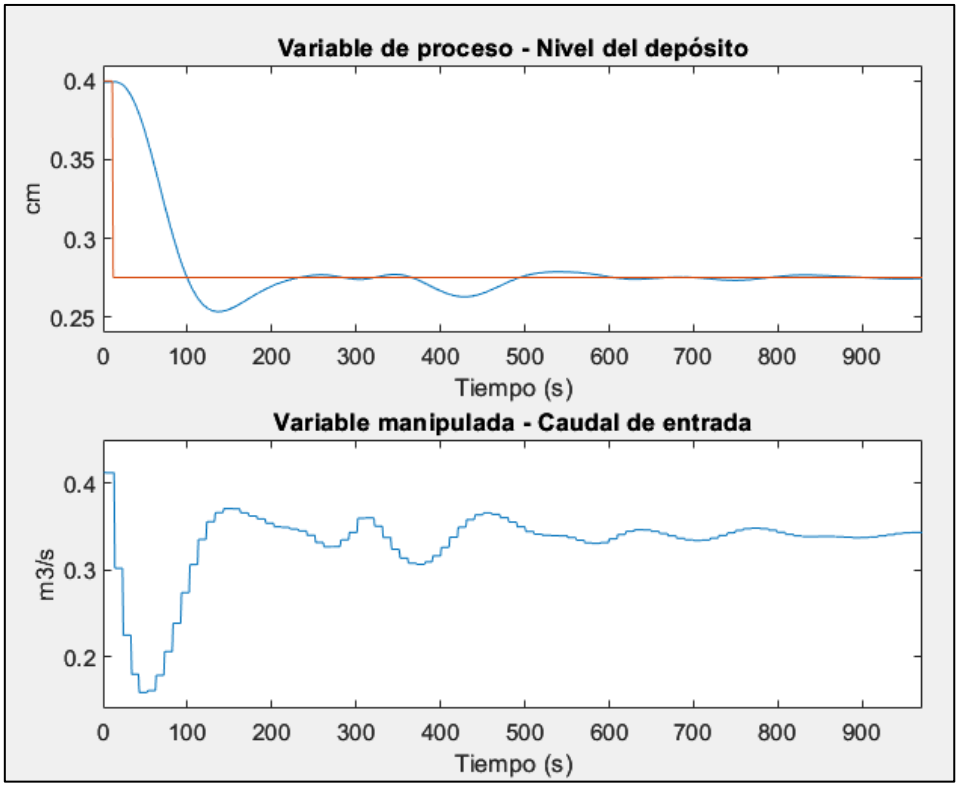


Figura 90: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 5 para *N2*.

Observando la Figura 90, el sistema responde como se esperaba, el control es capaz de seguir a la referencia volviendo la salida más oscilante poniendo en riesgo la estabilidad. Al reducir el valor del horizonte de predicción solo tenemos en cuenta los errores de los primeros coeficientes de la respuesta salto, que son los de mayor valor. Por lo que el sistema se vuelve más rápido pero bastante más oscilante, alejándose de la referencia en ocasiones.

Cambios en el parámetro: HORIZONTE DE CONTROL *Nu*.

En este caso, como es un sistema de primer orden sin integradores, el valor normal que tendrá el horizonte de control es de 1. Por tanto, solo se harán saltos con valores superiores al predeterminado.

Comenzaremos probando con un valor de 2 para el horizonte de control (ver Figura 91).

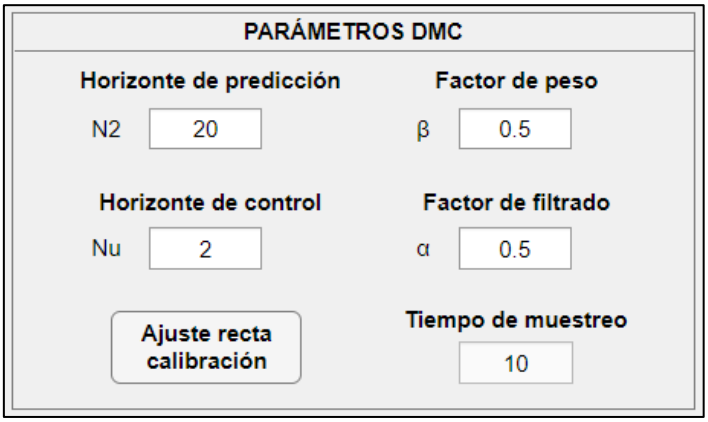


Figura 91: Cambio de valor en el parámetro del horizonte de control a 2.

Repetimos el salto en la referencia de 0.4 a 0.275. Los resultados se muestran en la siguiente figura:

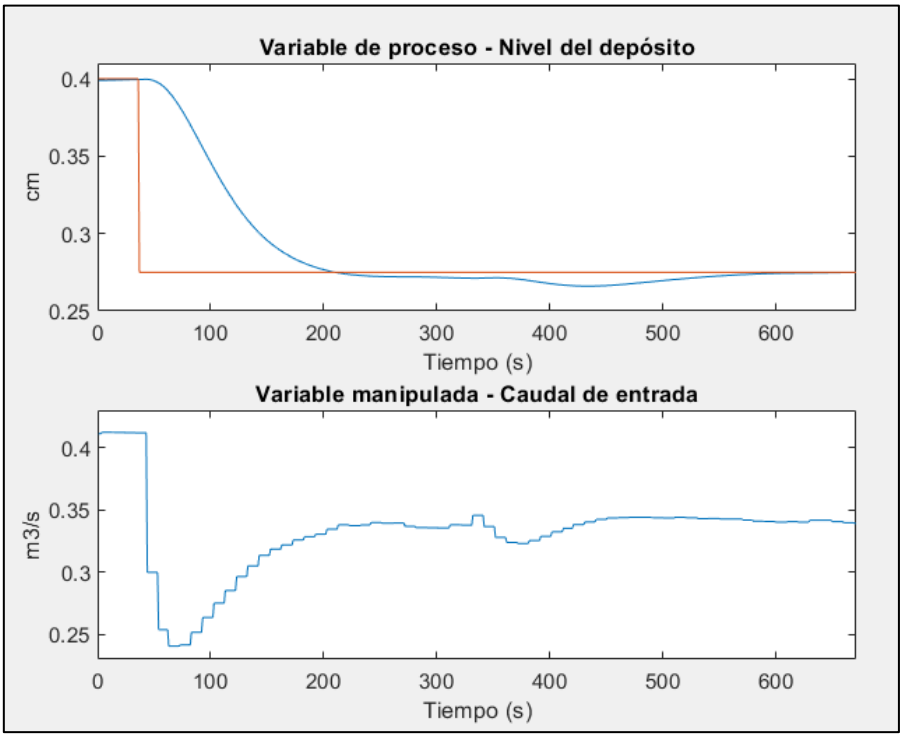


Figura 92: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 2 para *Nu*.

El control es bueno, el sistema sigue correctamente a la referencia. Subir el valor del horizonte de control, permite al control adoptar mayores grados de libertad, por ende, el sistema es algo más rápido.

Se probará a continuación a subir el valor de este parámetro a 10, una cifra bastante mayor, para ver qué tal responde el sistema (ver Figura 93). La respuesta del sistema se muestra en la Figura 94.

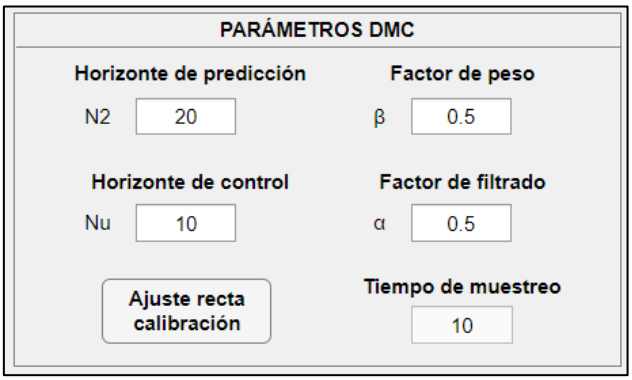


Figura 93: Cambio de valor en el parámetro de horizonte de control a 10.

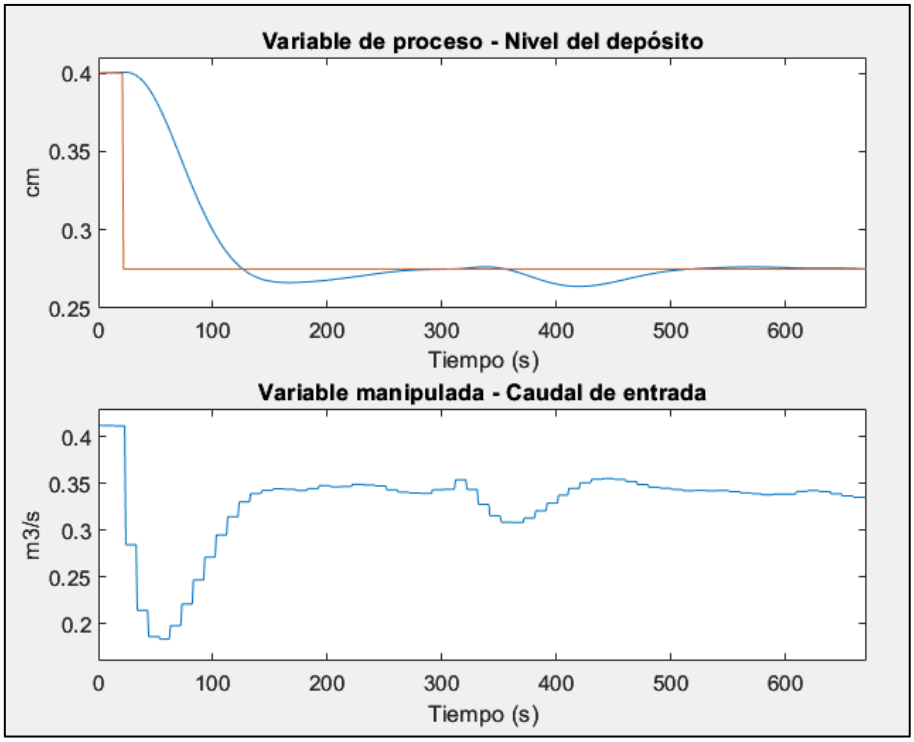


Figura 94: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 10 para *Nu*.

El control sigue siendo bueno, el sistema responde correctamente. Como era de esperar, como el control tiene mayor grado de libertad, los cambios en su valor son mayores (antes el valor mínimo era de 0.25 m^3/s aproximadamente y ahora es de menos de 0.2) y el sistema se ha vuelto más rápido. Sin embargo, con un valor menor se lograría la misma respuesta, ya que el control no necesitaría tantos grados de libertad para actuar.

Si bien es cierto que el control responde mejor subiendo el valor del horizonte de control, la salida en el estacionario oscila más que con un valor menor.

Cambios en el parámetro: FACTOR DE PESO β.

Recordemos que el factor de peso actúa sobre los cambios en la señal de control, haciendo que la variable de proceso alcance más rápido o lento a la referencia.

Comenzamos con un valor de 0.9 para el factor de peso (ver Figura 95).

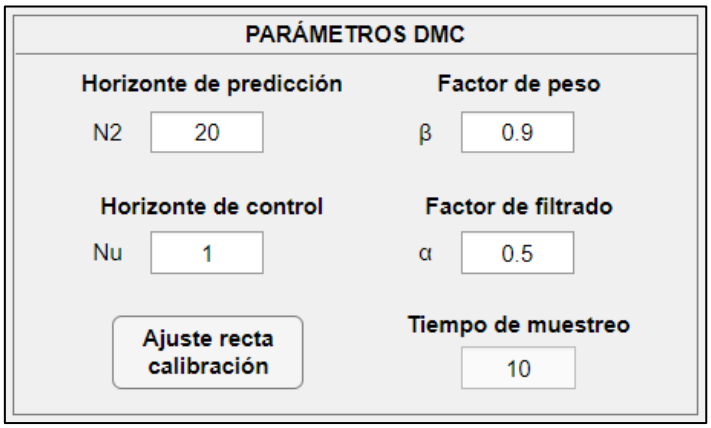


Figura 95: Cambio en el valor del parámetro de factor de peso a 0.9.

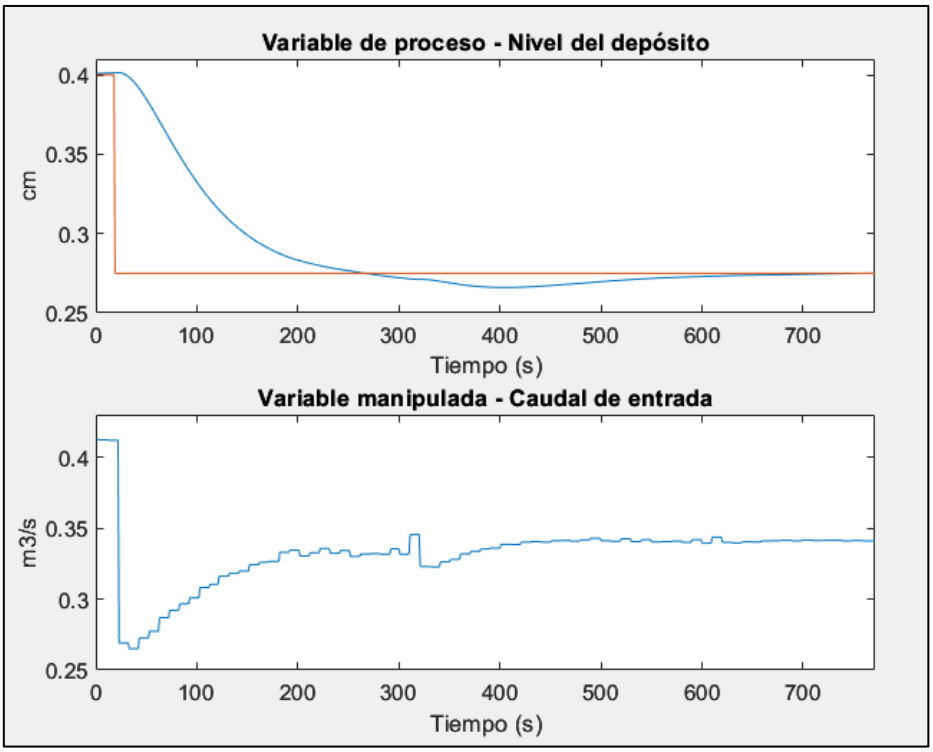


Figura 96: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 0.9 para *Beta*.

Aumentando el valor del factor de peso, el sistema tiene más en cuenta los cambios en el controlador y los penaliza, haciendo que la respuesta sea algo más lenta. Como con este valor no se aprecia este efecto, se aumenta significativamente el valor de este parámetro a 100 (véase Figura 97), para ver cómo afecta a la respuesta.

La respuesta del sistema ante este valor se muestra en la Figura 98. Se aprecia que los cambios en el control son más pequeños. Con los parámetros predeterminados, el primer cálculo de la señal de control tras el cambio en la referencia era de $0.41 m^3/s$ a $0.26\ m^3/s$, sin embargo, con estos parámetros el control realiza cambios más reducidos hasta alcanzar un valor mínimo de $0.31\ m^3/s$. Por tanto, la salida es más lenta.

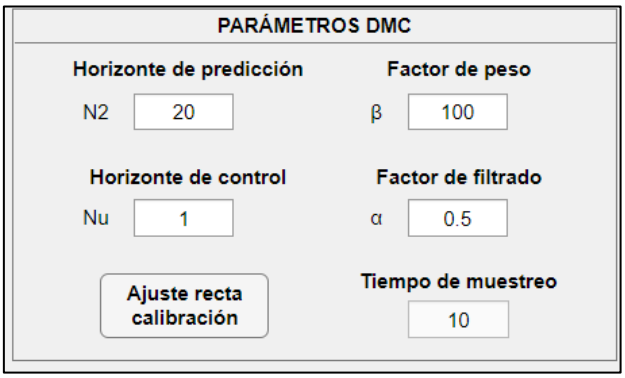


Figura 97: Cambio en el valor del parámetro de factor de peso a 100.

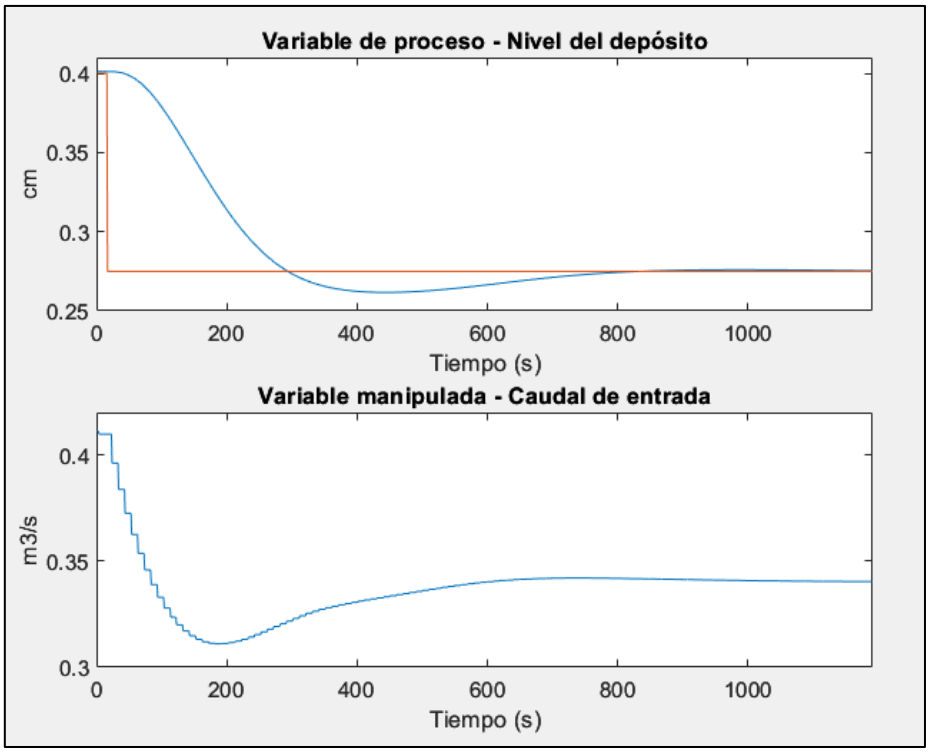


Figura 98: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 10 para *Beta*.

A continuación, repetiremos el salto para un valor de 10^{-5} para el valor del factor de peso, Figura 99.

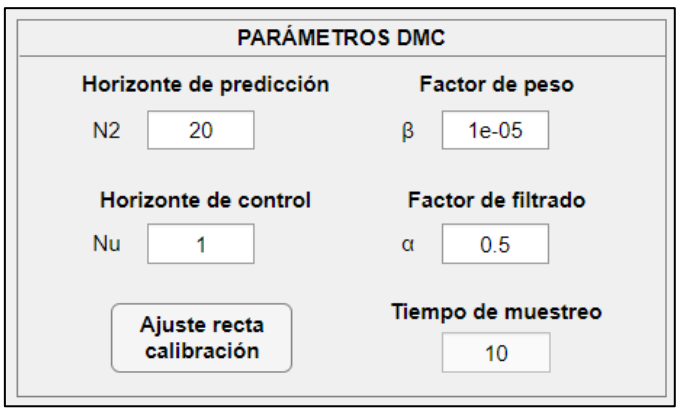


Figura 99: Cambio en el valor del parámetro de factor de peso a 1e-5.

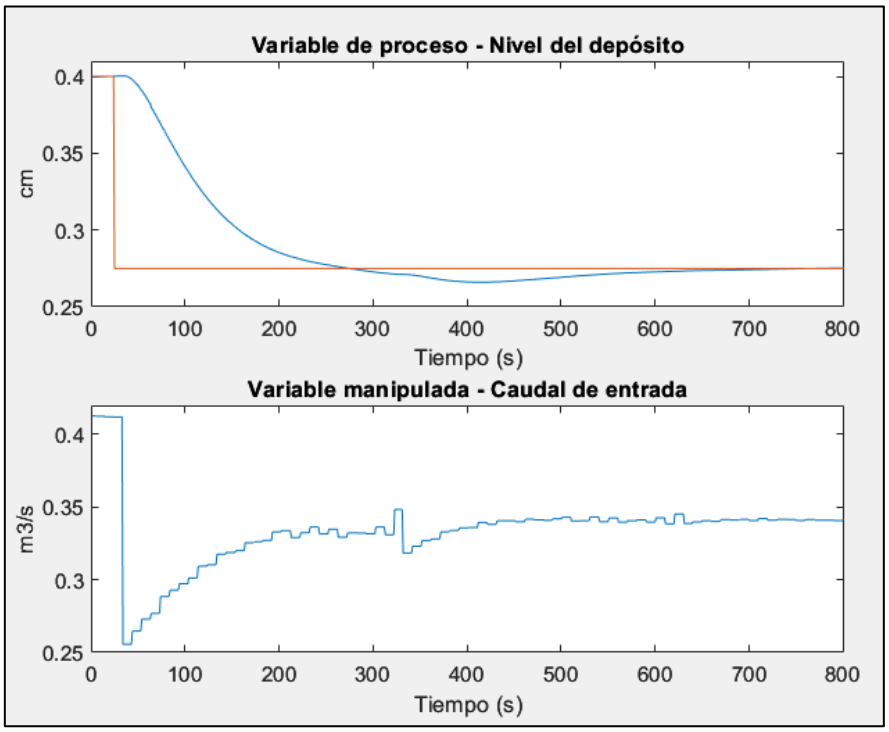


Figura 100: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 1e-5 para *Beta*.

Con valores más reducidos del factor de peso, se permite al control dar saltos más grandes en su valor. Como resultado, el sistema ofrece una respuesta algo más rápida y pronunciada.

Cambios en el parámetro: FACTOR DE FILTRADO α.

Al igual que para el parámetro anterior, el factor de filtrado provoca una respuesta del sistema más suave cuanto mayor sea su valor y más abrupta cuanto menor sea este. Su valor no suele superar la unidad, así que comenzaremos con un valor de 0.99.

Los resultados se muestran en la Figura 102. Como era de esperar, el control realiza cambios muy suaves en la variable manipulada, lo que hace que la respuesta sea muchísimo más lenta y suave, aumentando el tiempo de establecimiento a unos 2000 segundos.

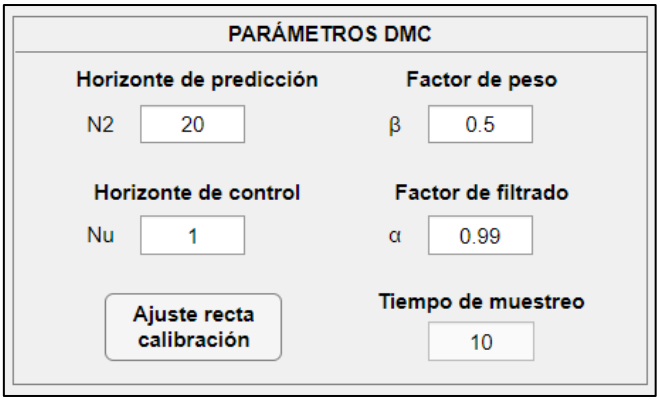


Figura 101: Cambio en el valor del parámetro de factor de filtrado a 0.99.

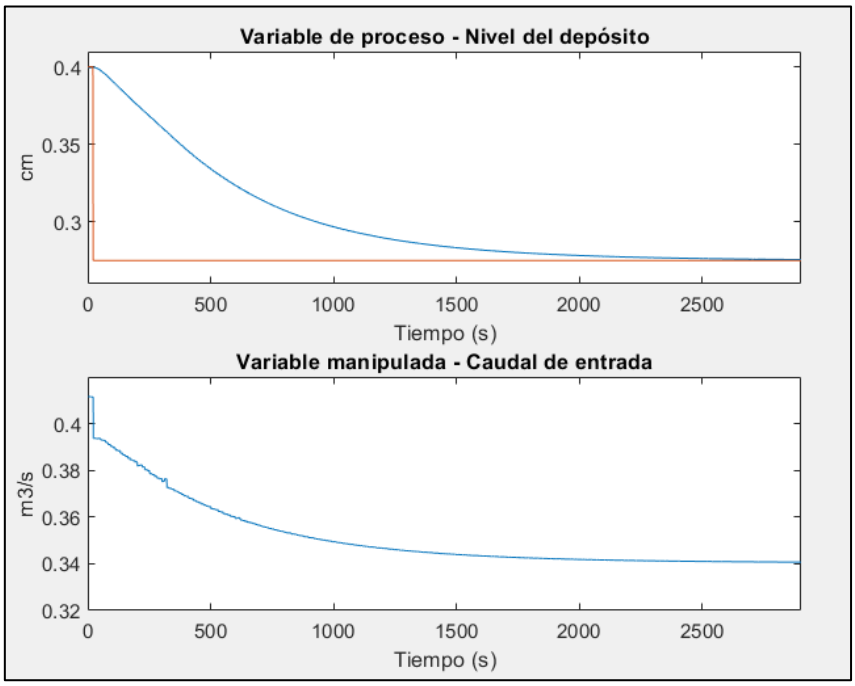


Figura 102: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 0.99 para Alfa.

A continuación, se repite el salto en la referencia para un valor de 10^{-5} para el factor de filtrado (ver Figura 103).

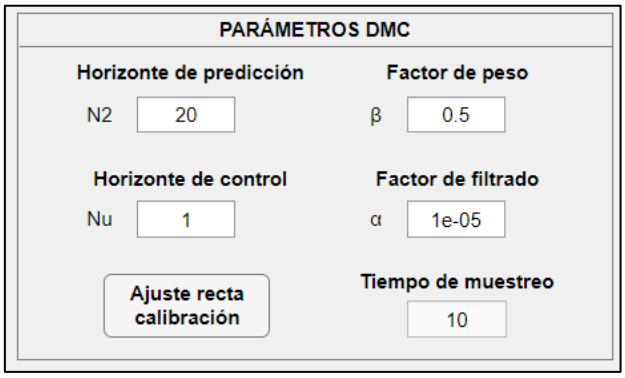


Figura 103: Cambio en el valor del parámetro de factor de filtrado a 1e-5.

Los resultados se muestran en la Figura 104. Como era de esperar, la señal de control tiene más libertad a la hora de hacer cambios de valor en la variable manipulada, realizando saltos mayores disminuyendo la respuesta del sistema. La salida entonces se vuelve más rápida a cambio de convertirla en algo más abrupto.

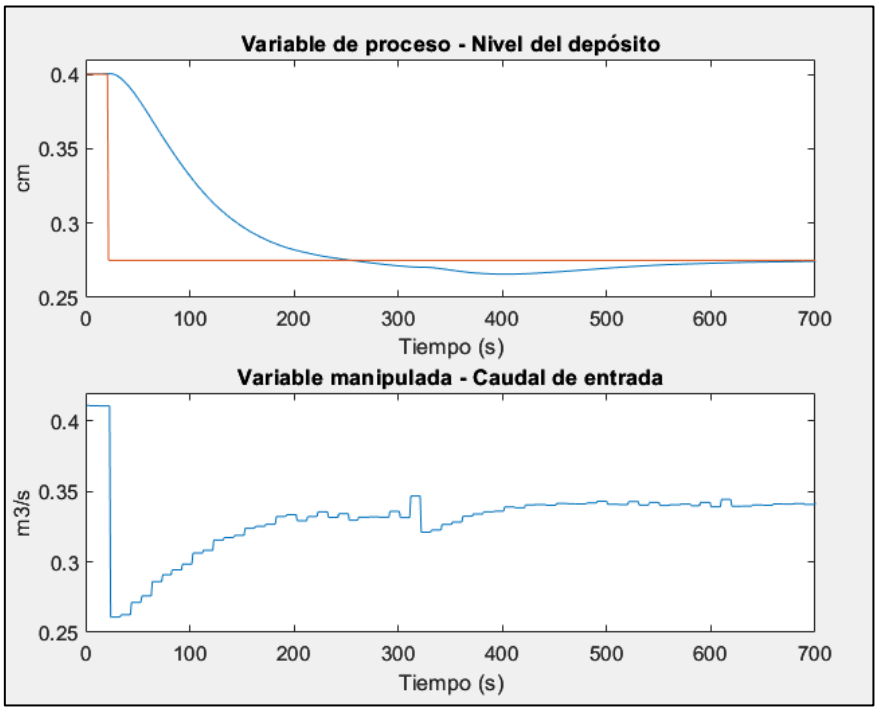


Figura 104: Respuesta del sistema ante un salto de 0.4 *cm* a 0.275 *cm* en la referencia, con un valor de 1e-5 para Alfa.

En resumen, para obtener un sistema que responda de manera rápida, nos interesaría tener un valor bajo para los parámetros de factor de peso y filtrado. Se podría también aumentar en una unidad o dos el horizonte de control y mantener el horizonte de predicción en un valor relativamente alto. Para un sistema estable, se aumentará el horizonte de predicción y los factores de peso y filtrado, a su vez, se reducirá el horizonte de control al mínimo.

7.2.2 MODELO DE LA PLANTA REAL

Al igual que para la planta simulada, se realiza el proceso de toma de datos. El resultado tras importar las señales a HIDEN es el siguiente (véase Figura 105).

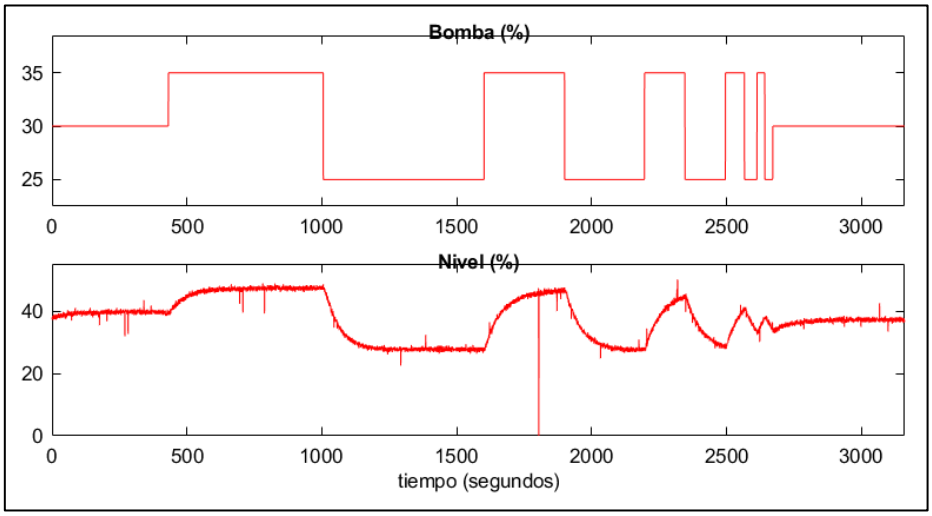


Figura 105: Señales del experimento de identificación de la planta real en *HIDEN*.

Al igual que en la planta de simulación, se aplicará a continuación el tratamiento de señales para su posterior identificación. En este caso se han aplicado dos procedimientos:

- **Nuevo tiempo de muestreo**. En este caso, también se ha optado con un nuevo periodo de muestreo de 10 segundos, mostrado en la Figura 106. Con esto lograremos que el control responda rápidamente. Además, tampoco perdemos información con este valor.
- **Filtrado pasa alta**. Para este caso, sí que es necesario aplicar el filtro pasa alta a frecuencia ω=0, mostrado en la Figura 107, ya que no se ha restado el punto de trabajo en la preparación del archivo (como se hizo en el modelo en simulación). Con esto eliminamos el nivel de continua.

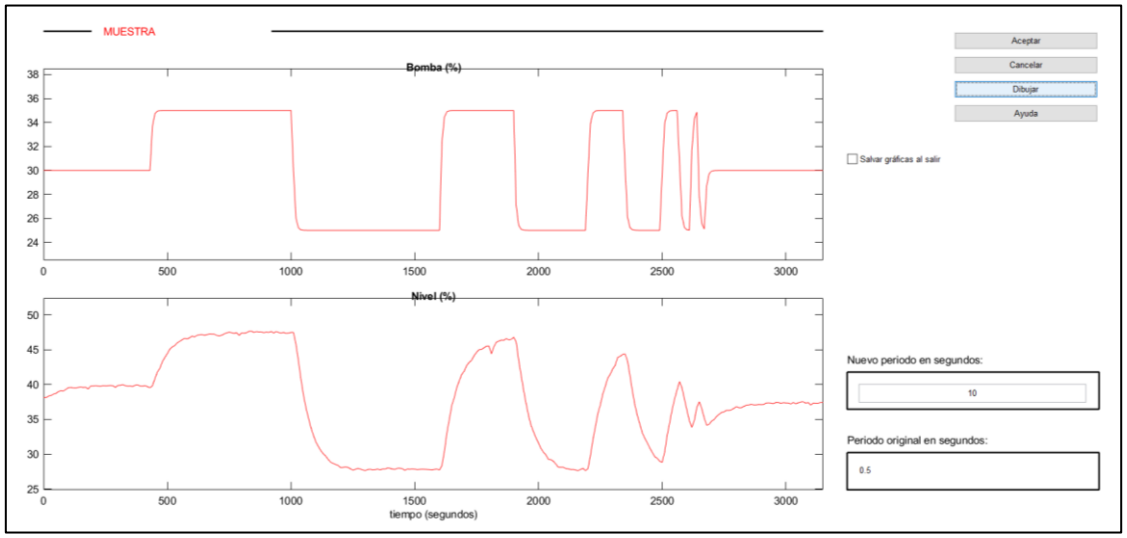


Figura 106: Representación de las señales tras aplicarles un nuevo periodo de muestreo.

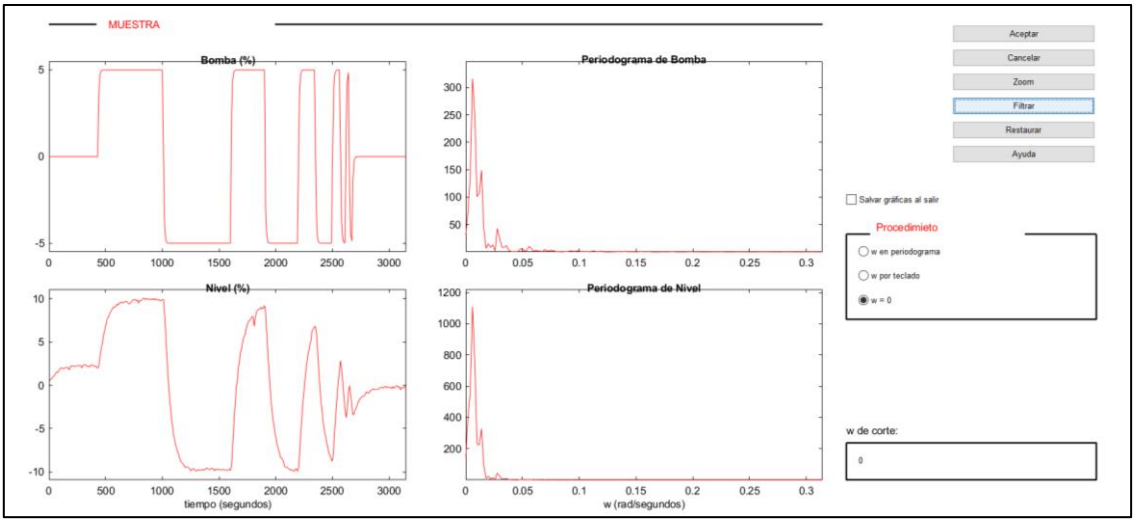


Figura 107: Representación de las señales tras aplicarles un filtro de pasa alta.

Se procede ahora con la identificación. Al igual que en el otro modelo, se escogerá el modelo de respuesta escalón con 30 coeficientes y el método de identificación de mínimos cuadrados. El resultado se muestra en la Figura 108.

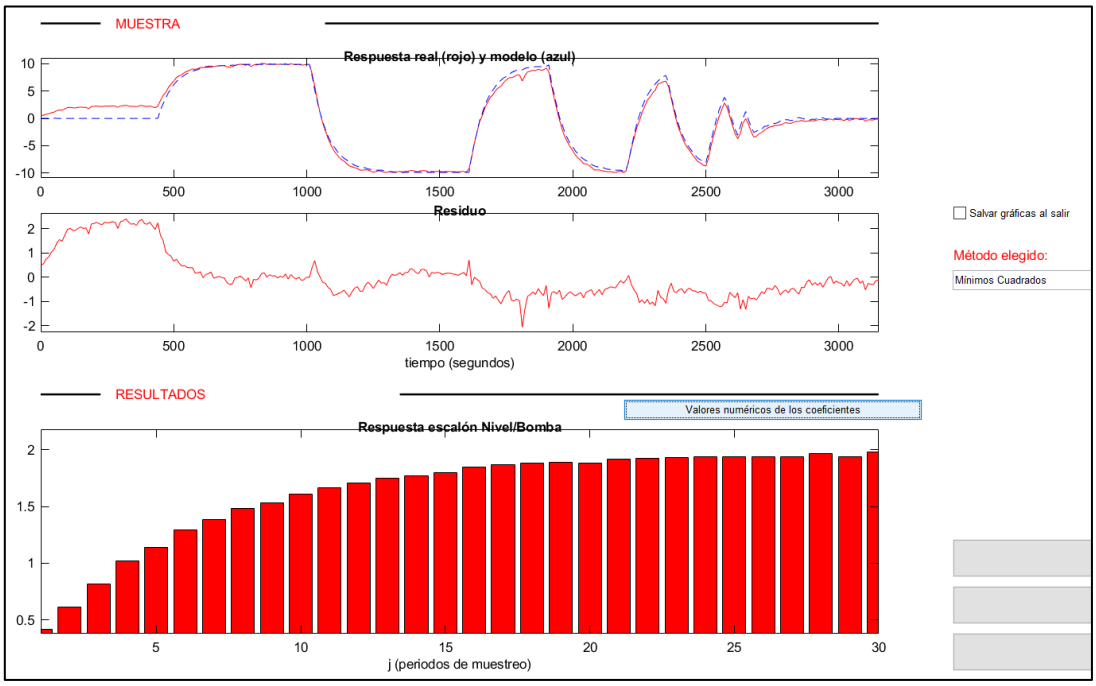


Figura 108: Identificación final de la planta real.

Como se aprecia, el sistema real se ajusta al modelo. Lo guardamos y lo incorporamos en la interfaz para ver cómo responde.

Se realizarán varios saltos de diferentes magnitudes con los parámetros que se muestran en la Figura 109 para ver cómo se comporta el sistema. Dichos saltos se muestran en las imágenes de Figura 110 a Figura 113.

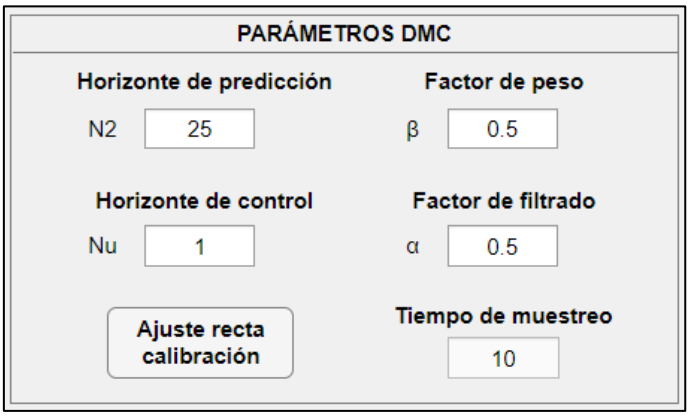


Figura 109: Valores de los parámetros DMC para el experimento de verificación de la planta real.

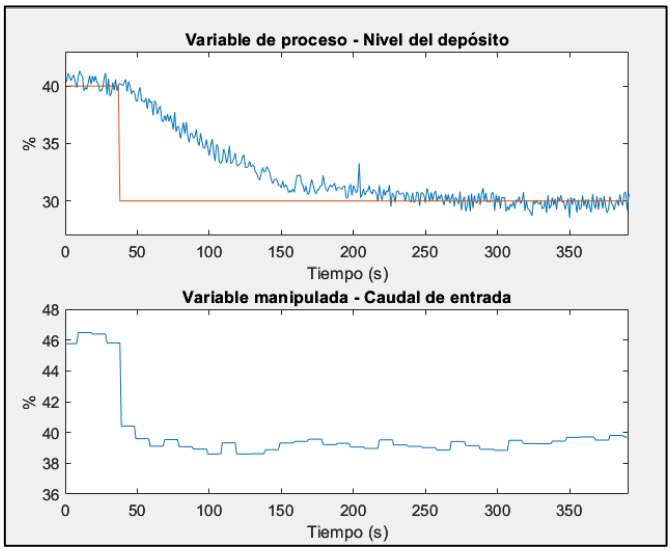


Figura 110: Salto del sistema ante un cambio en la referencia de 40% a 30%.

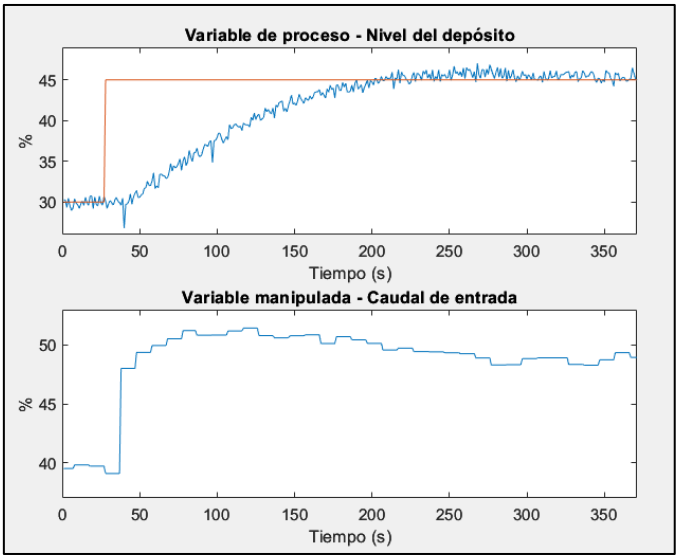


Figura 111: Salto del sistema ante un cambio en la referencia de 30% a 45%.

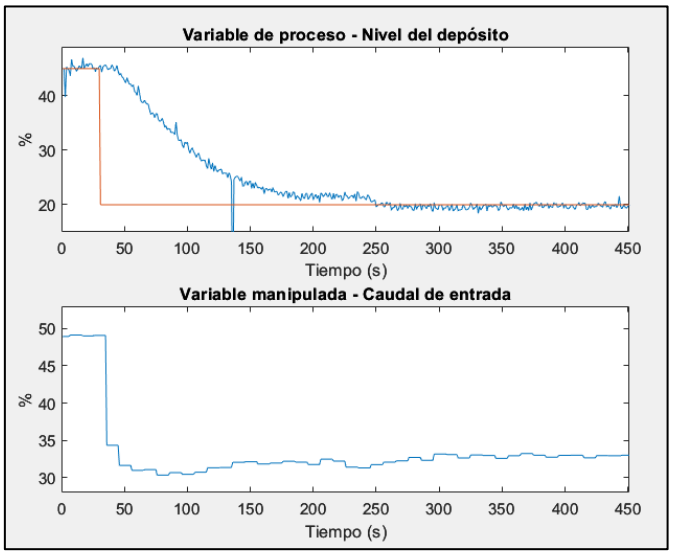


Figura 112: Salto del sistema ante un cambio en la referencia de 45% a 20%.

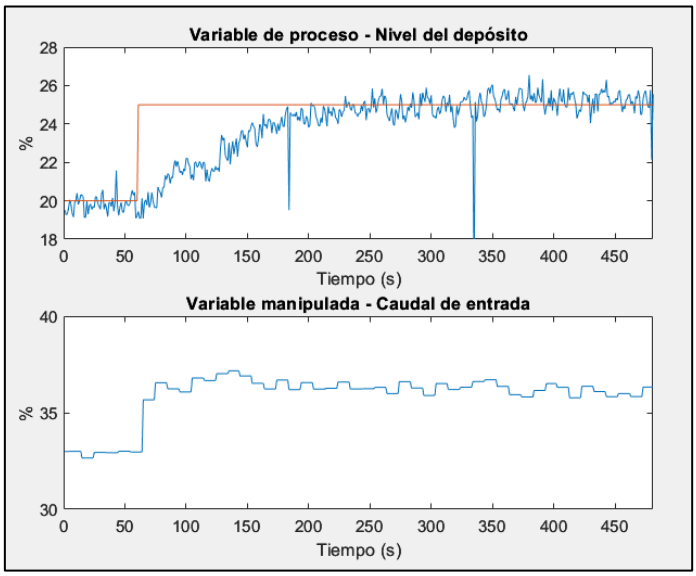


Figura 113: Salto del sistema ante un cambio en la referencia de 20% a 25%.

Como podemos ver, el sistema responde bien y es capaz de seguir la referencia sin problemas. La respuesta es sobreamortiguada con un tiempo de establecimiento de 200 segundos aproximadamente, más rápido que la planta simulada. Es conveniente tener en cuenta que el ruido proveniente de los sensores entorpece levemente la calidad del control y genera ciertos picos en la señal que no se corresponden con medidas reales.

Se ha logrado el objetivo de control para la planta real también. Al igual que en el apartado 7.2.1 MODELO DE SIMULACIÓN procederemos a comprobar cómo actúan los parámetros modificando sus valores. Las definiciones y cómo afectan los parámetros se trata en el apartado 2.3.4 Parámetros del control DMC.

Para poder comparar resultados, se toma como base el salto de la Figura 110: de 40 % del nivel al 30 %.

Cambios en el HORIZONTE DE PREDICCIÓN *N2*.

Se recuerda que el horizonte de predicción es el horizonte en el tiempo futuro de predicción de la salida para minimizar la función objetivo. Se busca minimizar/eliminar el error entre la referencia y las predicciones durante este horizonte temporal. Si su valor es pequeño, solo se consideran errores anteriores a la referencia, por tanto, el control será rápido, pero más oscilante. En cambio, si es muy grande, no mejora el resultado; a partir de cierto valor ya estaríamos en el valor de referencia y los errores serán prácticamente nulos, se logra un control más estable pero más lento.

En la primera prueba, se emplea un valor de *N2* de 50, superior al predeterminado (véase Figura 114). La respuesta del sistema ante el salto de referencia se muestra en la Figura 115.

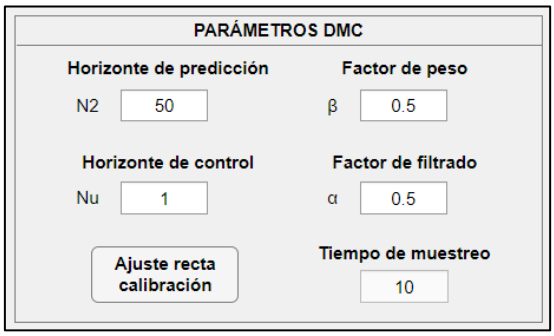


Figura 114: Cambio en el valor del parámetro de horizonte de predicción a 50.

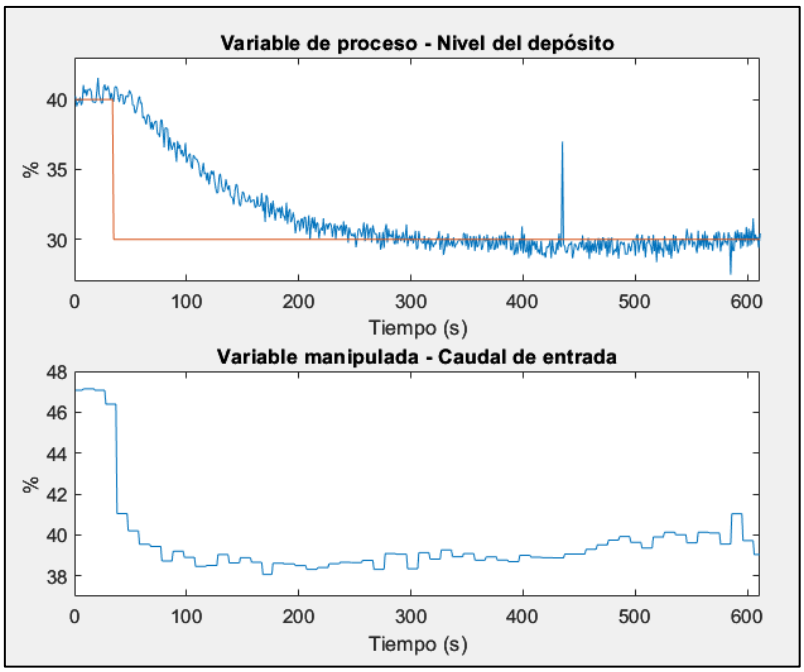


Figura 115: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 29 para *N2*.

El sistema es capaz de alcanzar la referencia en aproximadamente 250 segundos. Como se teorizó previamente, el control se ha vuelto más lento, se tiene en cuenta toda la dinámica del sistema y, por tanto, los últimos coeficientes (errores muy pequeños). En cuanto a las oscilaciones, no se puede apreciar bien ya que partimos de un sistema sobreamortiguado.

A continuación, se comprueba, para un valor de 5 para el horizonte de predicción (véase Figura 116), como responde el sistema. El resultado se muestra en la Figura 117.

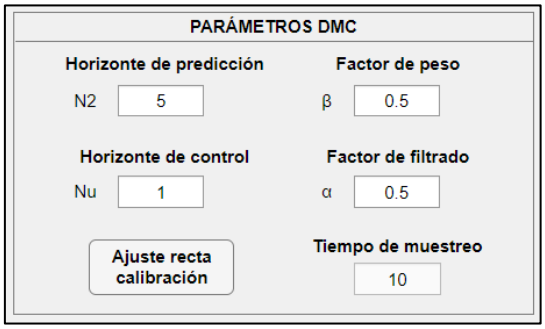


Figura 116: Cambio en el valor del parámetro de horizonte de predicción a 5.

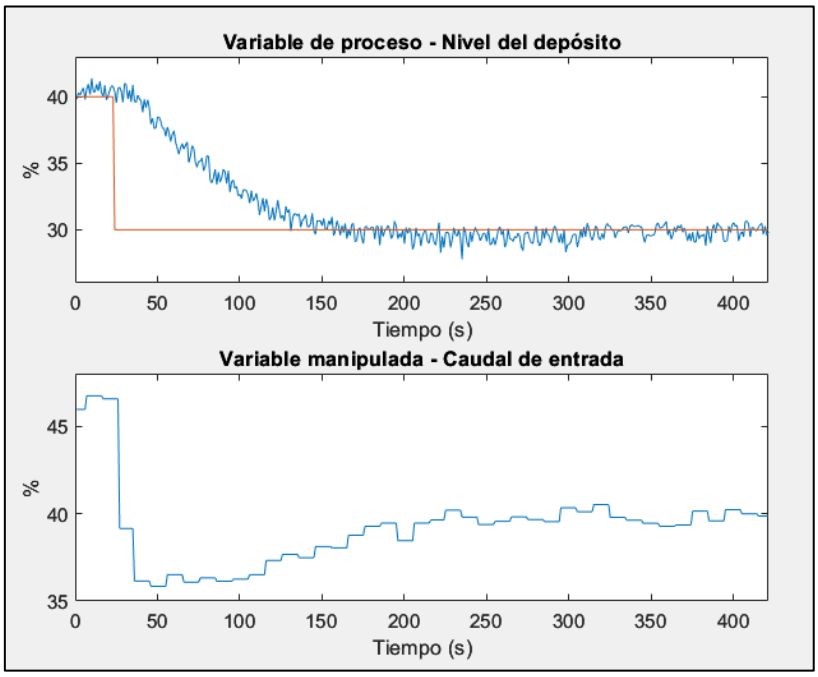


Figura 117: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 5 para *N2*.

El sistema se vuelve más rápido, alcanzando la referencia en unos 130 segundos. El control se aprecia más oscilante, llegando a un valor mínimo de un 36 % (comparado con un 38 – 39 % con los valores predeterminados) y luego se va recuperando el valor normal.

Como se ha previsto, el sistema se vuelve más rápido y oscilatorio con un menor valor del horizonte de predicción y viceversa. A partir de cierto valor, aumentar el horizonte de predicción no mejora la respuesta del sistema, puesto que ya abarca toda la dinámica y vuelve al control más estable pero lento.

Cambios en el HORIZONTE DE CONTROL *Nu*.

El horizonte de control es el número de acciones de control permitidas. A más número de acciones (más grados de libertad), tendremos mejores soluciones porque el control tendrá más libertad para cambiar su acción, sacrificando la carga computacional.

Al igual que en la planta simulada, se realizará el mismo salto para valores de 2 y 10 en *Nu*. Comenzamos con el valor más pequeño (véase Figura 118).

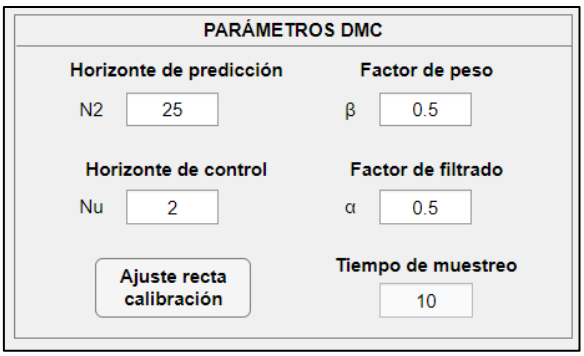


Figura 118: Cambio en el valor del parámetro de horizonte de control a 2.

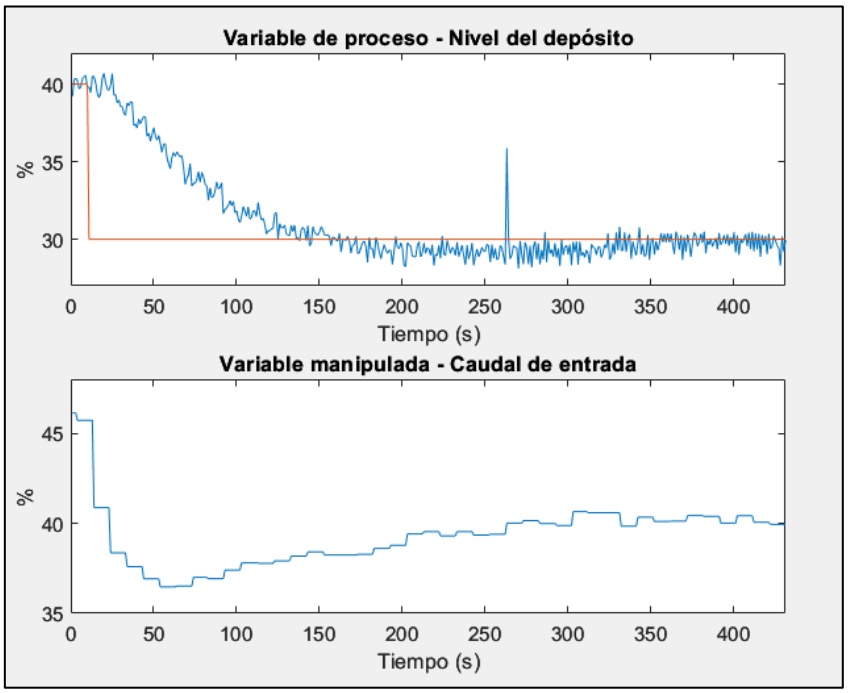


Figura 119: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 2 para *Nu*.

A partir de la respuesta del sistema de la Figura 119, se ve que el control es bueno, el sistema sigue correctamente a la referencia. Subir el valor del horizonte de control permite a este adoptar mayores grados de libertad, logrando reducir el valor de la señal de control a 37% generando así un ligero sobrepico en la salida. A pesar de esto, el sistema no se ha vuelto más rápido, el tiempo de establecimiento es similar al salto realizado con los parámetros predeterminados. Probaremos a continuación qué efectos tiene subir el horizonte de control a un valor más elevado (concretamente 10, Figura 120).

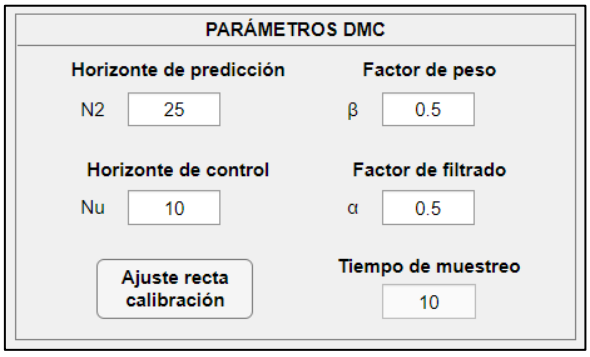


Figura 120: Cambio en el valor del parámetro de horizonte de control a 10.

Analizando la respuesta del sistema mostrada en la Figura 121, vemos el sistema sigue a la referencia sin problemas, el control es bueno. La respuesta es similar a la obtenida para un valor de 2 en *Nu*.

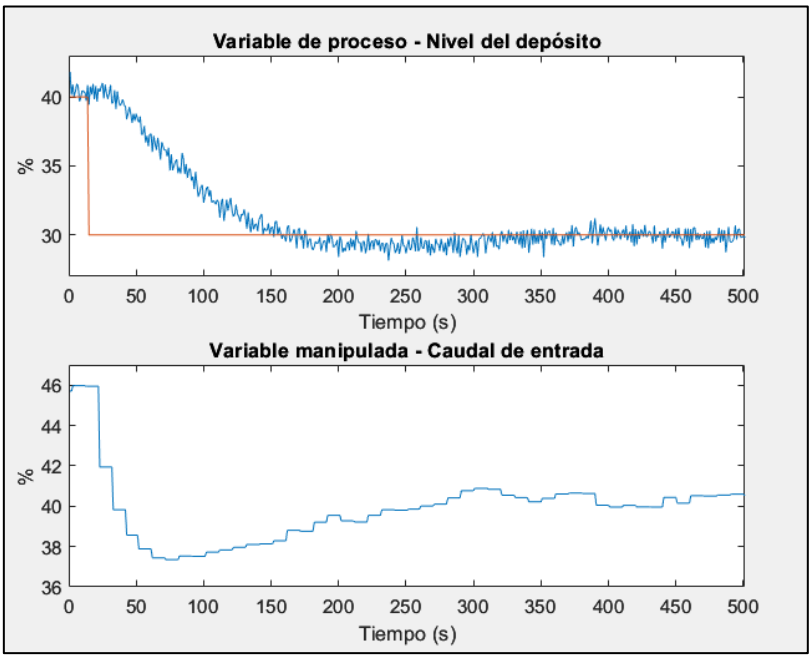


Figura 121: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 10 para *Nu*.

Al igual que se ha mencionado en el apartado Cambios en el parámetro: HORIZONTE DE CONTROL *Nu* del modelo de simulación, la planta es un sistema de primer orden sin integradores, el valor normal que tendrá el horizonte de control es de 1. Por esto, el hecho de subir el valor del horizonte de control no mejora la respuesta, con 1 el sistema es capaz de controlar de manera adecuada. Subir los grados de libertad del control no le ayuda a tomar mejores valores, por eso las respuestas son prácticamente idénticas.

Cambios en el parámetro FACTOR DE PESO β.

El coeficiente de supresión de movimiento, o factor de peso, es la importancia del esfuerzo de control en la función objetivo. Penalizará los cambios en la acción de control, por lo que, cuanto mayor sea, el controlador hará menos cambios y pequeños, provocando así un control más lento y suave. Por el contrario, un menor valor favorecerá una respuesta más rápida ya que no se penalizan cambios grandes en el control.

Comenzaremos con un valor de 1 para β (véase Figura 122) obteniendo así la respuesta de la Figura 123.

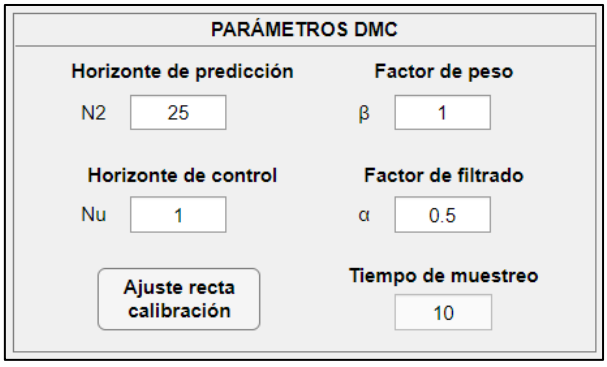


Figura 122: Cambio en el valor del parámetro de factor de peso a 1.

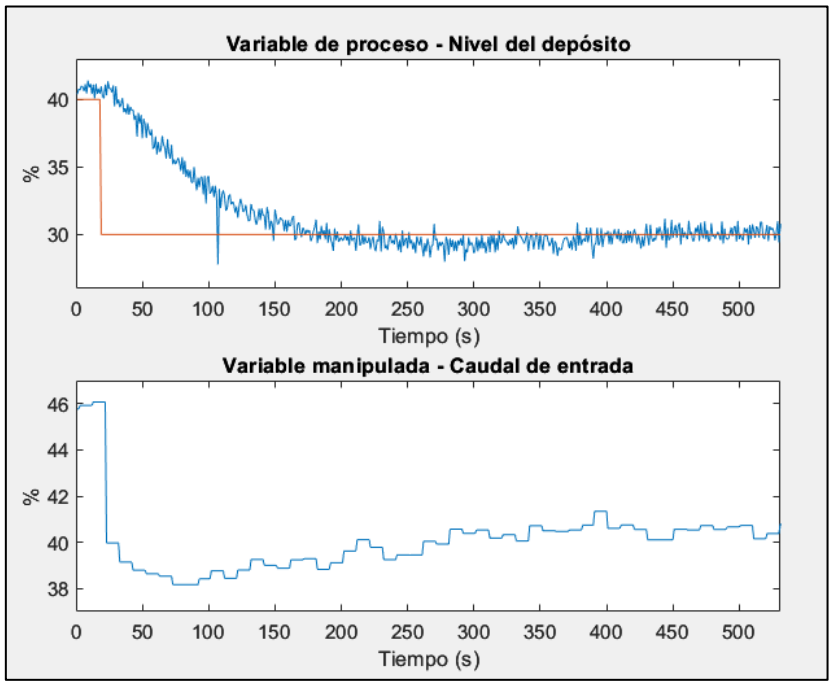


Figura 123: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 1 para *Beta*.

No se aprecia un cambio significativo en la respuesta, por lo que se procederá a subir el factor de peso a un valor superior de 50 (Figura 124).

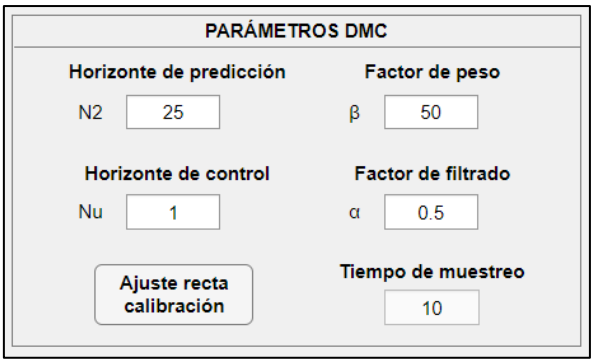


Figura 124: Cambio en el valor del parámetro de factor de peso a 50.

La respuesta para estos valores se muestra en la Figura 125. Sí que se aprecia una ralentización de la salida, teniendo un tiempo de establecimiento de unos 230 segundo (frente a los 200 con los parámetros predeterminados). En cuanto a la penalización de cambios de valor en la señal de control, se puede observar que (para un valor de 50 para el factor de peso) los saltos son más reducidos teniendo un cambio máximo de 47% a 44%, comparado con el salto de 46% a 40% con los valores predeterminados.

Para hacer aún más evidente el efecto en el sistema de subir el factor de peso, solo habría que aumentar su valor más (por ejemplo 100 como en la simulación).

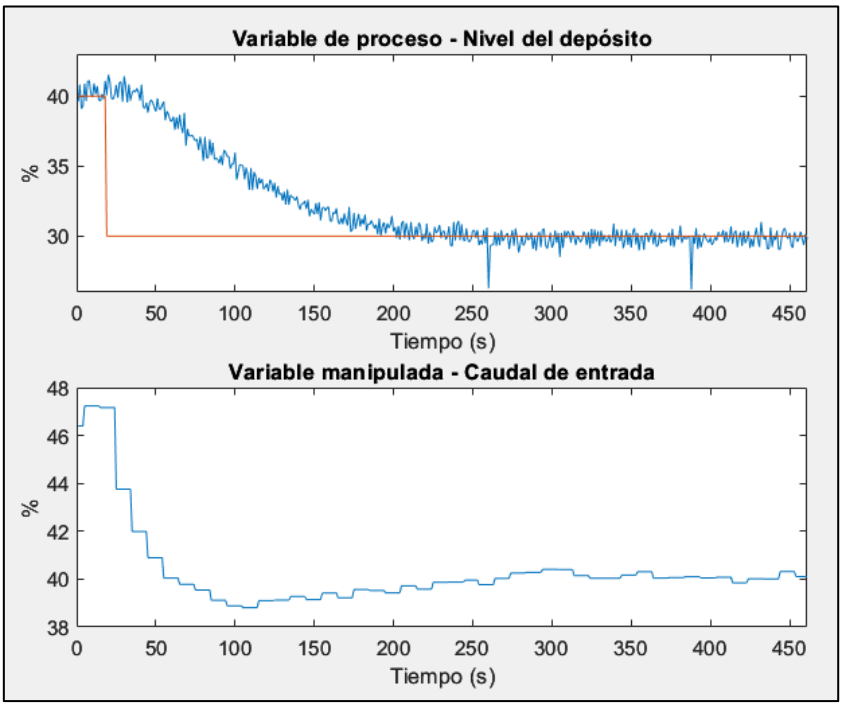


Figura 125: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 50 para Beta.

Cambios en el parámetro FACTOR DE SUAVIDAD *α*.

Se recuerda que el factor de suavidad se considera como el coeficiente de un filtro de primer orden de tal forma que se considera una trayectoria de referencia a seguir desde el valor actual de la salida hasta el valor deseado de la referencia; valores próximos a uno implica un acercamiento suave y lento a la referencia y viceversa.

Primero, se probará el efecto en el sistema de subir este parámetro a 0.99 (Figura 126).

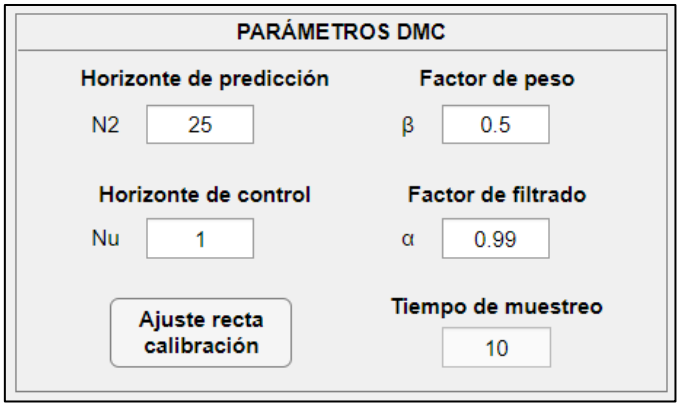


Figura 126: Cambio en el valor del parámetro de factor de filtrado a 0.99.

A continuación, se muestra la respuesta del sistema para estos parámetros (Figura 127).

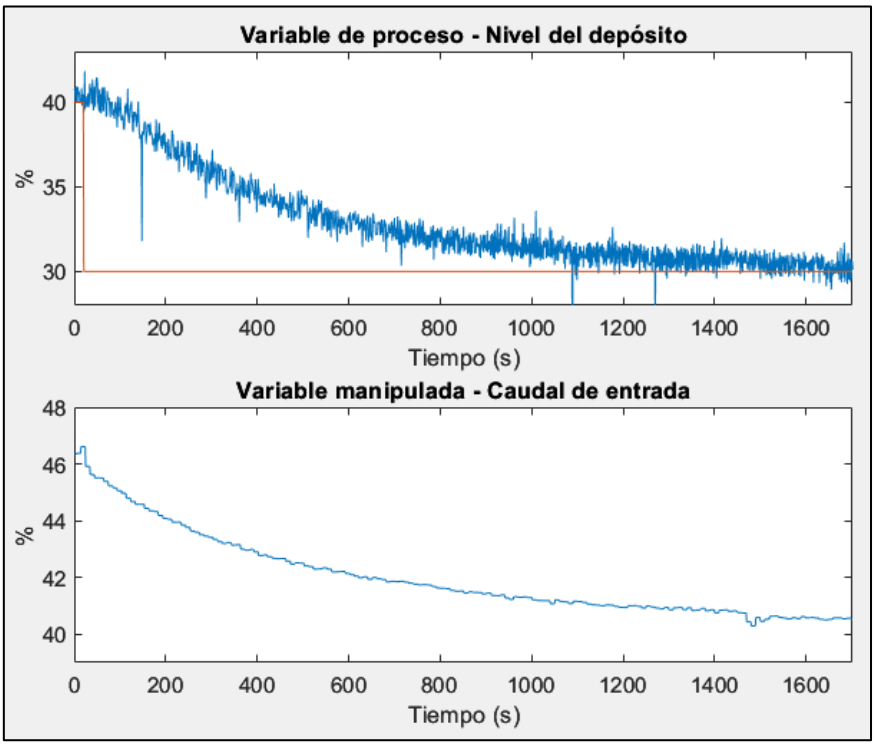


Figura 127: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 0.99 para Alfa.

Como es evidente, el sistema se ha vuelto muchísimo más lento, generando un tiempo de establecimiento de unos 1600 segundos. Los cambios en el control son mínimos para suavizar la salida, con lo que se logra el parecido de la forma de ambas señales.

Se probará con un valor de 10^{-8} de factor de suavidad para ver cómo responde el sistema ante este cambio (véase Figura 128).

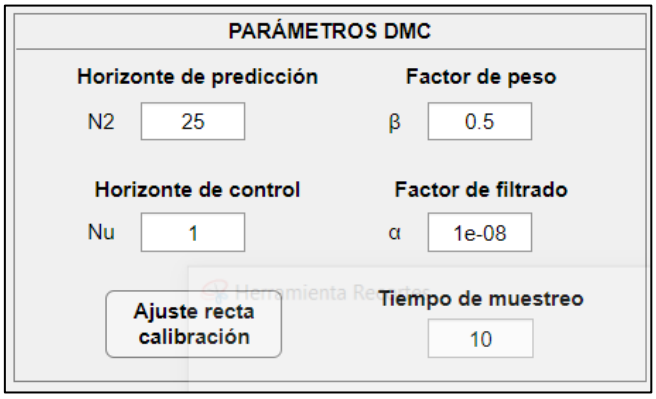


Figura 128: Cambio en el valor del parámetro de factor de filtrado a 1e-8.

La respuesta se muestra en la Figura 129. Al disminuir el valor de este parámetro, se puede apreciar que el sistema se ha vuelto un poco más rápido, ofreciendo una velocidad de respuesta de 170 segundos aproximadamente. El control sacrifica ligeramente la suavidad para ofrecer una respuesta más rápida.

Comparando las señales de las variables manipuladas, se puede ver que, en este caso, los cambios de valor son algo superiores lo que hace que se alcance la referencia más rápido que con los valores predeterminados.

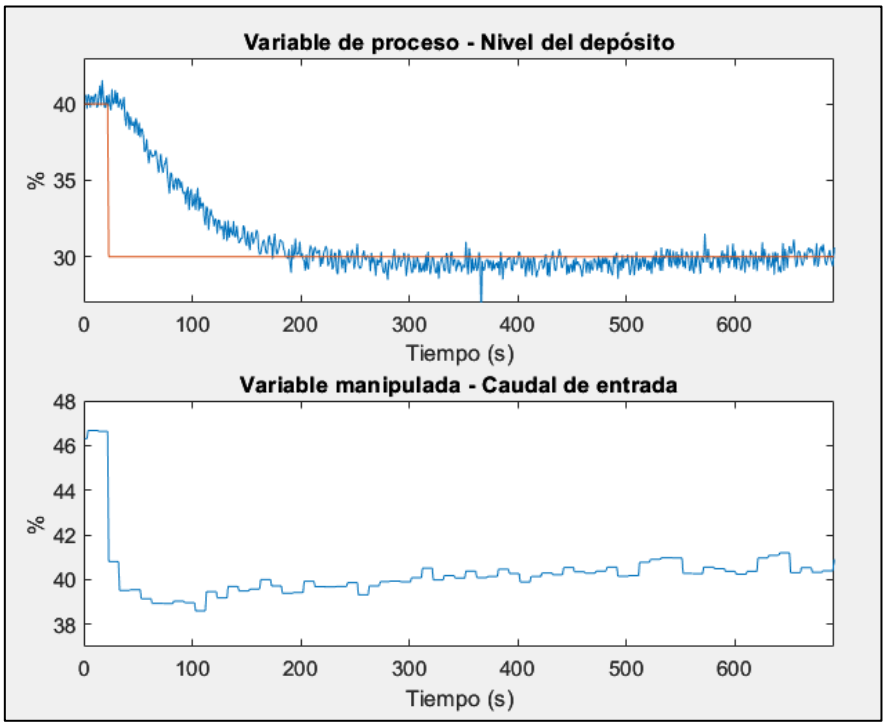


Figura 129: Respuesta del sistema ante un salto de 40% a 30% en la referencia, con un valor de 1e-8 para Alfa.

En resumen, al igual que las conclusiones obtenidas en la planta de simulación, si queremos un sistema rápido tendremos que bajar el horizonte de predicción y los factores de supresión y filtrado. En cambio, es conveniente aumentar el horizonte de control. Por el contrario, para un sistema estable (pero lento) ajustaremos los parámetros DMC de forma inversa: subiremos horizonte de predicción y los factores de supresión y filtrado; y reduciremos el horizonte de control.

En función de lo que se busque en la planta, se ajustarán los parámetros al tipo de control óptimo.

Para acabar la validación de funcionamiento, veremos cómo reacciona el sistema ante un cambio en la perturbación.

Validación del sistema ante una perturbación.

Recordando el apartado 1.1 OBJETIVOS, las perturbaciones no medibles son corregidas por el control predictivo. En el caso del sistema real, la válvula de salida del segundo depósito es manual, pudiendo así generar una perturbación en la salida (véase 1.1 INTRODUCCIÓN).

En este apartado se comprueba el funcionamiento del control DMC ante un cambio en esta perturbación, en este caso abriendo y cerrando la válvula manual de salida. Los parámetros del control predictivo empleados pueden observarse en la Figura 130. Las respuestas del sistema se muestran en la Figura 131 y Figura 132. Recordemos que la apertura de válvula es inversamente proporcional al nivel del depósito: si abrimos la válvula, aumenta el caudal de salida y por tanto se reduce el nivel del depósito.

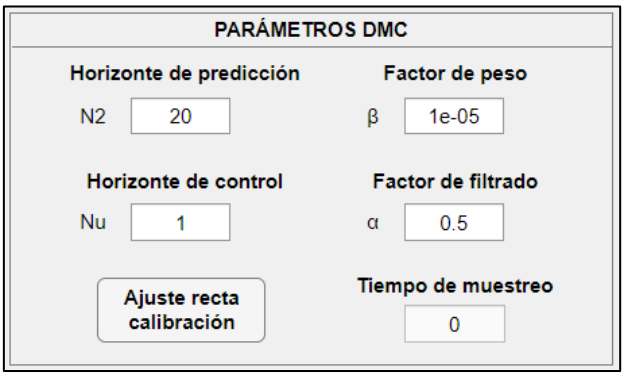


Figura 130: Parámetros del control para la validación ante un cambio en la perturbación.

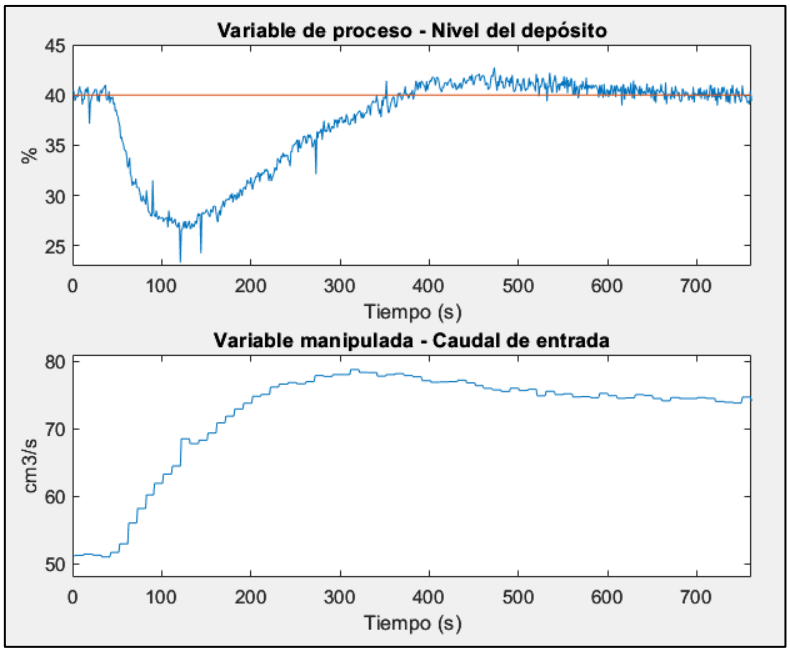


Figura 131: Respuesta del sistema ante un cambio en la perturbación: aumentar caudal de salida.

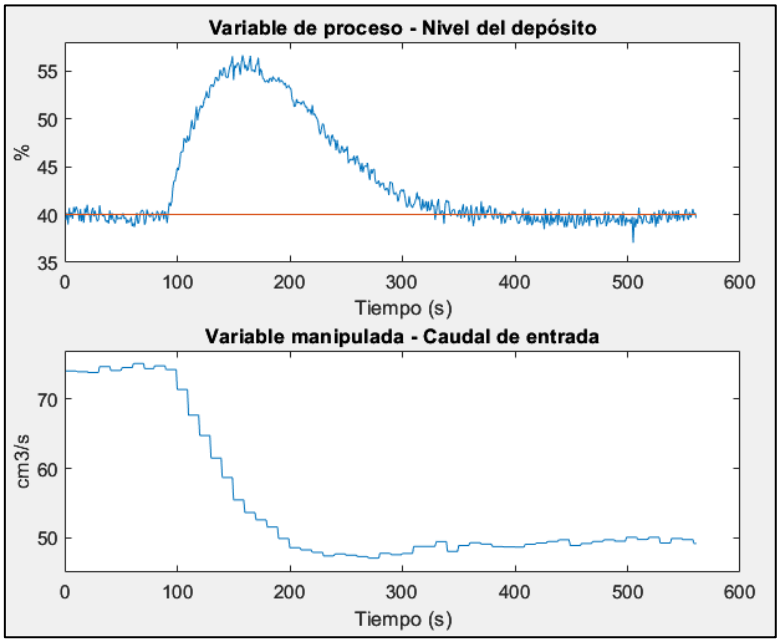


Figura 132: Respuesta del sistema ante un cambio en la perturbación: reducir el caudal de salida.

Ante unos cambios en la perturbación, el sistema responde adecuadamente y logramos recuperar la referencia. En ambos casos (abrir/cerrar la válvula) el control actúa de manera efectiva y volvemos a estado estacionario en unos 250 segundos aproximadamente.

Al igual que se ha explicado al final del apartado Cambios en el parámetro FACTOR DE SUAVIDAD, los parámetros se pueden definir en función de lo que busquemos en la respuesta del sistema (para los saltos realizados en este caso, simplemente se quería comprobar su correcto funcionamiento ante un cambio en la perturbación).

CAPÍTULO 8: CONCLUSIONES Y TRABAJO FUTURO

8.1 CONCLUSIONES

El objetivo principal de este proyecto es la creación de una interfaz que ofrezca al usuario una manera intuitiva y amena de controlar una planta a través del control predictivo de matriz dinámica DMC. A la vista de los resultados obtenidos en el proceso de validación del funcionamiento, se puede confirmar que esta aplicación ha sido creada en función a las necesidades buscadas con éxito. Gracias al manual de usuario que incluye la interfaz, el usuario puede manipular el programa sin inconvenientes, brindando una experiencia intuitiva y fácil de entender.

El diseño se ha fundamentado en la programación de la interfaz, validación y optimización del programa. Se tomó un cambio de rumbo a la hora de abordar la información con el servidor OPC prescindiendo del uso del modelo en *SIMULINK*. Esto hizo modificar el diseño por completo hasta llegar a la versión que se ofrece en la memoria.

En cuanto al aprendizaje requerido, se han rescatado conocimientos de Fundamentos de Automática, Fundamentos de Informática, Control de Procesos (principalmente) y Taller de Robótica (que permite al alumno trabajar con *App Designer*). Ha sido necesario investigar y profundizar en el uso del lenguaje M y *App Designer*.

A continuación, se exponen las conclusiones que se han obtenido durante el desarrollo.

Para empezar, la mayor dificultad que puedes encontrarte en este trabajo es la curva de aprendizaje que hay a la hora de aprender el lenguaje propio de *MATLAB* (lenguaje M), cómo funciona el intercambio de datos a través de un servidor OPC en este programa y el funcionamiento de *App Designer*, sus elementos y las interacciones que estos pueden tener (entre sí y con elementos externos).

Tras conseguir familiarizarte un poco con estos conceptos, se ofrecen multitud de opciones para las diferentes funciones que el programador quiera implementar. Además, gracias al foro de *mathworks* (foro donde los usuarios y desarrolladores de *MATLAB* comparten información) es más sencillo salir de los baches que puedes encontrarte, ya que muchos de los problemas que pueden surgir ya los han tenido otros usuarios y ofrecen sus soluciones. Con la comunidad que tiene a la espalda, existe una gran variedad de campos que se han abarcado dentro de este programa. También cabe destacar el hecho de que puedes comparar este lenguaje con otros existentes (como C++ o Python) y tratar de abordar las tareas de manera similar a como lo harías con otros lenguajes.

La interfaz es capaz de ofrecer los valores reales de la planta, facilitando su control. Como era de esperar, la planta simulada no presenta unos valores similares a la planta real. Sin embargo, sus dinámicas y la respuesta que ofrecen ambos sistemas son similares.

En cuando al código, cabe destacar que probablemente se pueda optimizar mucho más. Aunque se nos ofrezca un corrector de sintaxis, la estructura y metodología a la hora de abordar los intereses de la interfaz quedan en manos del desarrollador y su conocimiento, lo cual hace que el camino elegido no pueda ser el mejor.

Para acabar, *App Designer* resulta una buena herramienta para desarrollar aplicaciones de interés académico de manera muy intuitiva. Si bien es cierto que no presenta una gran cantidad de elementos programables, esto hace que el aprendizaje y programación sean más entretenidos, haciendo una experiencia agradable. Además, gracias a *MATLAB*, las aplicaciones creadas pueden compartirse en su plataforma para que otros usuarios puedan acceder a estas sin complicaciones. Esto permite que la comunidad mejore y aprenda las interfaces alrededor del mundo.

8.2 TRABAJO FUTURO

La primera línea para continuar con este proyecto es básicamente ampliar el entorno de trabajo actual e incluir el tratamiento de perturbaciones medibles. Actualmente solo se abordan las perturbaciones no medibles (por ejemplo, la válvula de apertura a la salida del segundo depósito de la planta real), pero gracias al modelo en *SIMULINK* de Yi Cao se podría preparar al control para que actúe ante las perturbaciones medibles. El ingeniero deja preparada la planta, lo cual también abre el camino a abordar esta modificación sin necesidad de usar la herramienta *SIMULINK* (al igual que se ha hecho con este proyecto).

En el presente, los sistemas con los que puede trabajar la aplicación son de tipo SISO (*Single Input Single Output*), por lo que otra posible línea de trabajo podría ser el abordamiento de sistemas tipo MIMO (*Multiple Input Multiple Output*).

Además, de optimizar el código de la interfaz para así intentar reducir el costo computacional de la aplicación, permitiendo a esta una mejora en los cálculos internos.

CAPÍTULO 9: BIBLIOGRAFÍA

[1] Control Predictivo basado en Modelo – DMC. Disponible en: https://controlautomaticoeducacion.com/control-predictivo/dmc/#Que_es_un_Control_Predictivo [Consulta: 29/01/2024].

[2] Desarrollo de apps mediante *App Designer*. Disponible en: https://es.mathworks.com/help/matlab/app-designer.html [Consulta: 19/01/2024].

[3] Code Analyzer: Identify and address code issues (https://es.mathworks.com/help/matlab/ref/codeanalyzer-app.html). Recuperado January 19, 2024.

[4] Yi Cao (2024). MPC Tutorial IV: DMC Simulink Block and Example (https://www.mathworks.com/matlabcentral/fileexchange/25412-mpc-tutorial-iv-dmc-simulink-block-and-example), MATLAB Central File Exchange. Recuperado January 19, 2024.

[5] Yi Cao (2024). MPC Tutorial I: Dynamic Matrix Control (https://www.mathworks.com/matlabcentral/fileexchange/19479-mpc-tutorial-i-dynamic-matrix-control), MATLAB Central File Exchange. Recuperado January 19, 2024.

[6] About the Maintenance of Level-1 MATLAB S-Functions (https://es.mathworks.com/help/simulink/sfg/maintaining-level-1-matlab-s-functions.html). Recuperado January 19, 2024.

[7] Industrial Communication Toolbox - Conexión con servidores OPCDA. Disponible en: https://es.mathworks.com/help/icomm/ug/opcda.html [Consulta: 19/01/2024].

[8] Introducción a la identificación de sistemas https://www.tecnicaindustrial.es/introduccion-a-la-identificacion-de-sistemas/ [Consulta: 19/01/2024].

CAPÍTULO 9: BIBLIOGRAFÍA

ANEXO A1: MANUAL DE USUARIO

A1.1 INTRODUCCIÓN

Tras iniciar la aplicación, el aspecto que presenta la aplicación se muestra en la Figura 133. Distinguimos un menú y justo debajo dos pestañas: “Ajustes del modelo” y “Control de planta”.

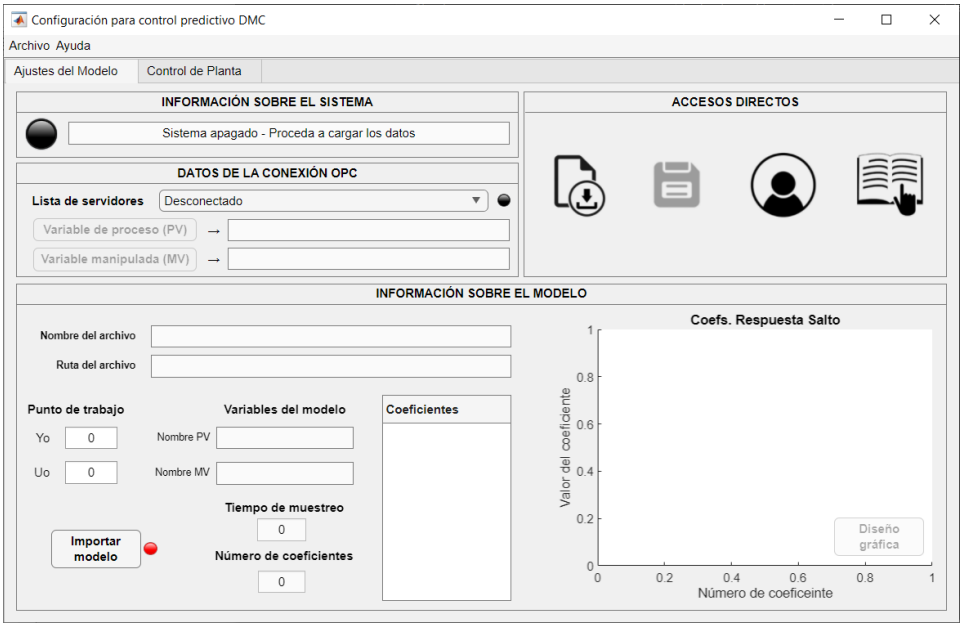


Figura 133: Aspecto de la interfaz en la pestaña "Ajustes del modelo".

Comenzaremos explicando los distintos elementos del interfaz. La pestaña “Ajustes de modelo” se compone de cuatro paneles:

A. **INFORMACIÓN SOBRE EL SISTEMA**. Como el nombre indica, se muestra el estado de la app, qué acción se está llevando a cabo, qué necesita la interfaz para seguir su funcionamiento, etc.
B. **ACCESOS DIRECTOS**. Contiene iconos que permiten el acceso rápido a ciertas funciones (más adelante se indican estas funciones).
C. **DATOS DE LA CONEXIÓN OPC**. Muestra la información del servidor OPC y los nodos elegidos como variables del sistema.
D. **INFORMACIÓN SOBRE EL MODELO**. Muestra la información sobre el modelo/configuración cargada como el nombre del archivo, la respuesta salto con sus coeficientes, los nombres de las variables y resto de parámetros relevantes.

A continuación, se irán explicando los distintos paneles.

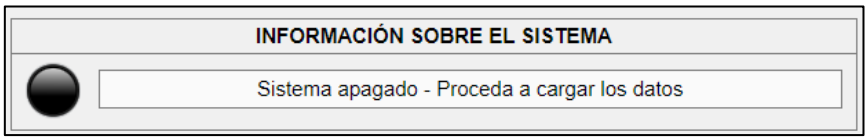


Figura 134: Aspecto del panel "Información sobre el sistema".

El panel "Información sobre el sistema" tiene básicamente dos elementos (véase Figura 134):

- **Campo de texto** que muestra información sobre en qué está trabajando la interfaz o qué necesita para poder iniciar el control.
- **Lámpara** que indica en qué estado se encuentra el sistema. Puede presentar cuatro colores:
  - Negro (sistema apagado, hace falta cargar todo).
  - Rojo (carga parcial, falta importar/cargar un modelo, configurar el servidor o introducir los límites de las variables).
  - Verde (carga lista, se puede iniciar el control).
  - Azul cian (proceso de guardado finalizado).

Figura 135: Aspecto del panel "Accesos directos".

El panel "Accesos directos" (Figura 135) facilita al usuario cuatro acciones a través de cuatro iconos. De izquierda a derecha:

- Primer icono: **CARGAR CONFIGURACIÓN**. Importa un archivo que contiene una configuración previa, es decir, un archivo que ha sido guardado por esta interfaz. En la pestaña 'Proceso de carga' se detalla en profundidad los aspectos relacionados con este proceso.
- Segundo icono: **GUARDAR CONFIGURACIÓN**. Guarda los últimos valores de los parámetros y campos de la aplicación, es decir, la configuración de la interfaz. En la pestaña 'Proceso de guardado' se detalla en profundidad los aspectos relacionados con este proceso.
- Tercer icono: **INFORMACIÓN DEL DESARROLLADOR**. Abre una ventana que muestra la información sobre el desarrollador de la aplicación Saúl Antolín, su contacto y enlaces de las páginas de la Universidad de Valladolid y la Escuela de Ingenierías Industriales.
- Cuarto icono: **MANUAL DE USUARIO**. Abre una ventana que expone las instrucciones sobre la interfaz (información mostrada en este documento).

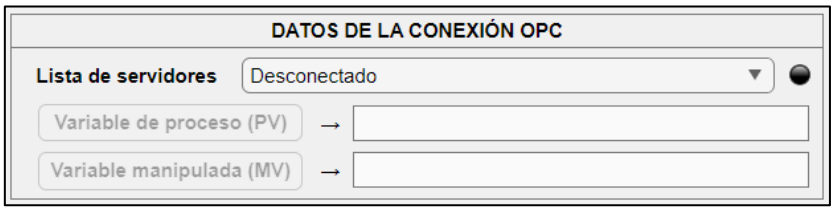


Figura 136: Aspecto del panel "Datos de la conexión OPC".

El panel "Datos de la conexión OPC" (Figura 136) abarca los elementos relacionados con la conexión del servidor y elección de nodos. Básicamente tiene cuatro elementos:

- El desplegable "**Lista de servidores**" realiza la conexión con el servidor OPC.
- El botón "**Variable de proceso (PV)**" abre la ventana de selección del nodo que actuará como variable de proceso.
- El botón "**Variable manipulada (MV)**" abre la ventana de selección del nodo que actuará como variable manipulada.
- Una **lámpara** que nos indicará el estado de la conexión con el servidor.

En la pestaña "**Proceso de carga**" se detalla en profundidad los aspectos relacionados con este proceso.

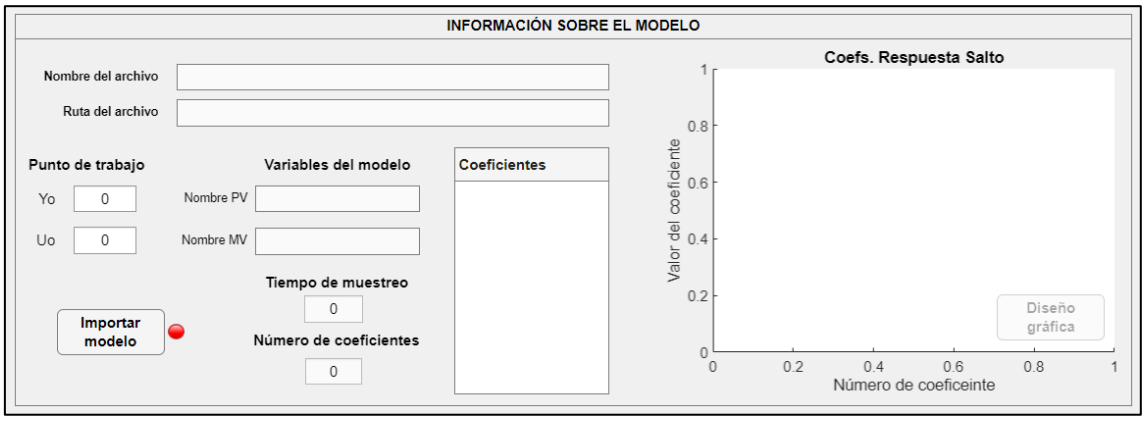


Figura 137: Aspecto del panel "Información sobre el modelo" en la vista de los datos del modelo.

El panel "**Información sobre el modelo**" (Figura 137) recoge aquellos elementos que contienen la información importante del archivo importado. Dichos componentes son:

- Campos de texto "**Nombre del archivo**" y "**Ruta del archivo**". Muestran el nombre y ruta completa respectivamente.
- Campos numéricos "**Punto de trabajo**". Permiten al usuario introducir el punto de trabajo del archivo importado.
- Campos de texto "**Variables del modelo**", "**Tiempo de muestreo**" y "**Número de coeficientes**". Muestran esos tres parámetros al usuario.
- Tabla "**Coeficientes**". Presenta los coeficientes de la respuesta salto.
- Gráfica "**Coefs. Respuesta Salto**". Dibuja los coeficientes de la tabla anterior.
- Botón "**Importar Modelo**". Abre la ventana de carga de archivo. En la pestaña "Proceso de carga" se detalla en profundidad los aspectos relacionados con este proceso.

- Botón "**Diseño gráfica**". Cambia la vista de este panel para acceder a la configuración de la gráfica. En la Figura 138 se muestra esta vista.

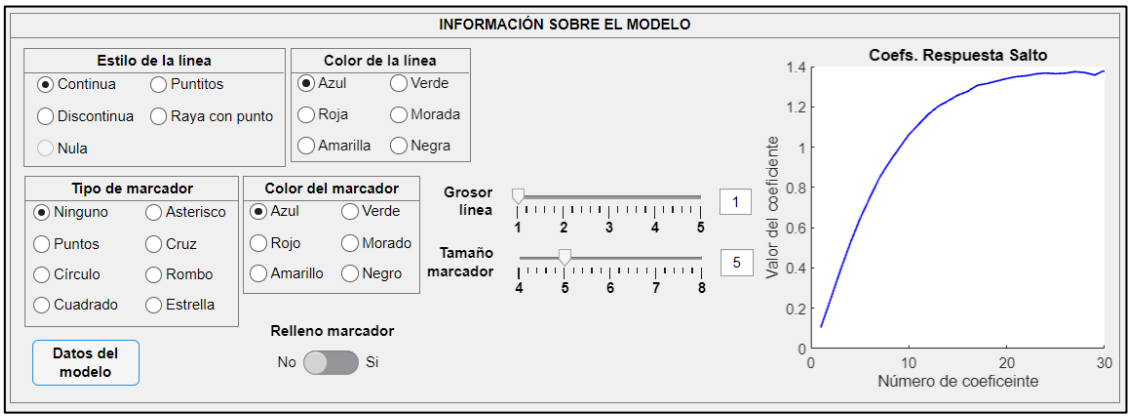


Figura 138: Aspecto del panel "Información sobre el modelo" en la vista del diseño de la gráfica.

En esta parte del panel podremos cambiar la configuración de la gráfica. Se puede cambiar el tipo y color de línea que une los puntos; el tipo, color y relleno de marcador de los puntos; y los grosores de la línea y el marcador.

El botón "**Datos del modelo**" nos regresa a la vista con la información del modelo.

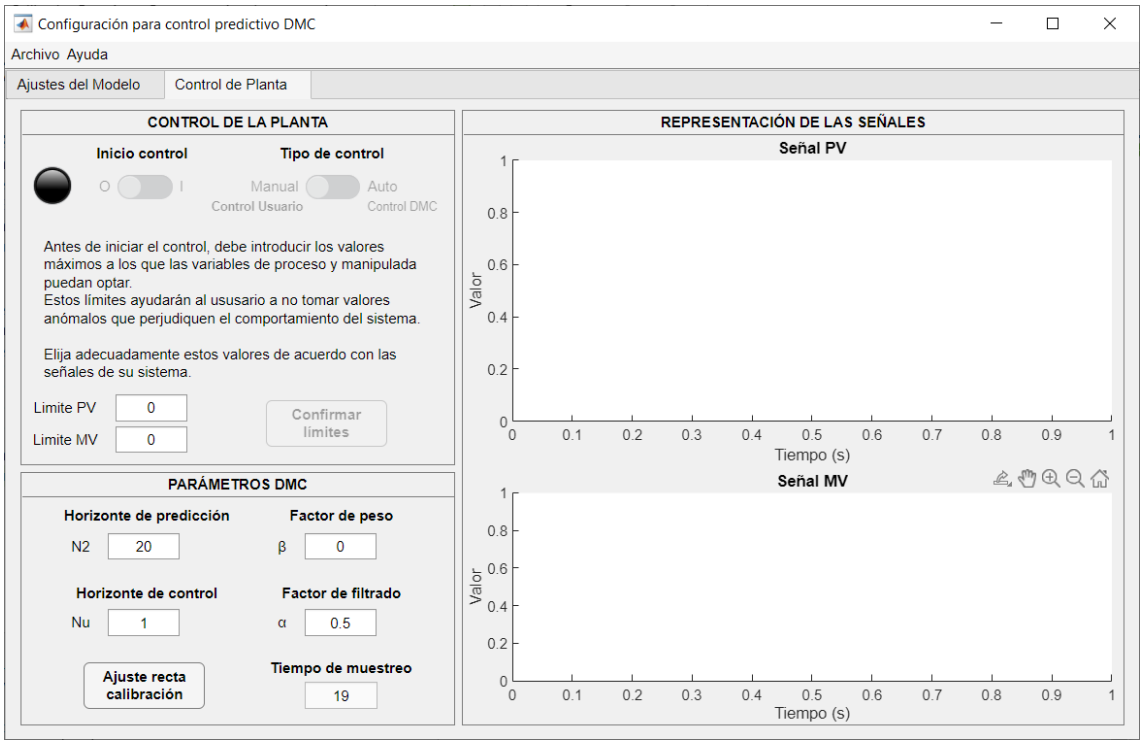


Figura 139: Aspecto de la interfaz en la pestaña "Control de planta".

El aspecto inicial de la pestaña "Control de Planta" se muestra en la Figura 139. Los tres paneles que la componen son:

A. CONTROL DE LA PLANTA. Permite al usuario introducir los límites de las señales y (una vez introducidos) gestionar el control (iniciar, parar, control automático...).

B. PARÁMETROS DMC. Contiene los campos con los parámetros del control DMC y rectas de calibración para que el usuario pueda editarlos.

C. REPRESENTACIÓN DE LAS SEÑALES. Contiene las gráficas que representarán los valores de las variables del sistema.

A continuación, se irán explicando los distintos paneles.

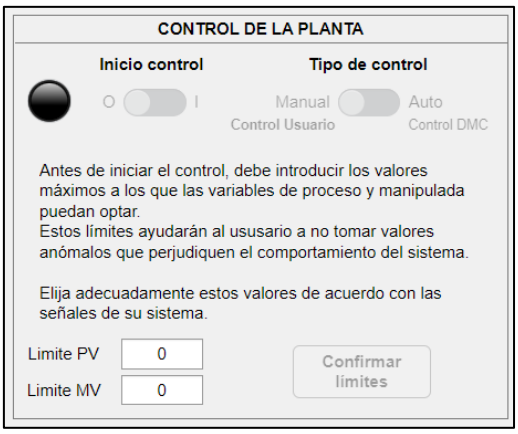


Figura 140: Aspecto inicial del panel "Control de la planta".

En primera instancia, el panel "**Control de la planta**" tiene el siguiente aspecto mostrado en la Figura 140. Básicamente tenemos dos cuadros numéricos en los que el usuario podrá introducir los valores de los límites de las variables que haya elegido.

Una vez introducidos se habilitará el botón "**Confirmar límites**" y los límites quedarán guardados (se pueden modificar a posteriori).

Una vez los límites se han introducido el aspecto es el siguiente:

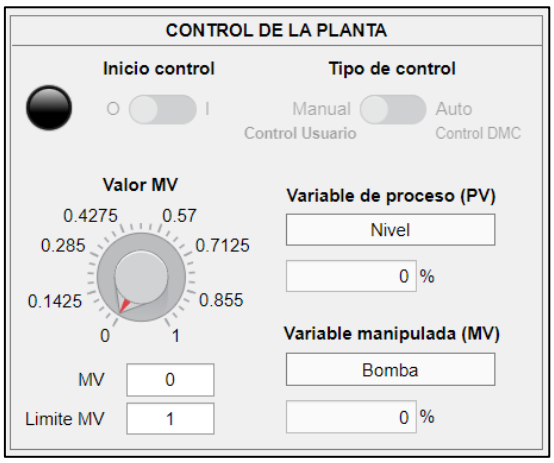


Figura 141: Aspecto final del panel "Control de la planta".

Los campos ahora son los siguientes:

- Ruleta "**Valor MV**". Permite al usuario introducir el valor de la variable manipulada (en manual) o la consigna (en automático, además cambiará su nombre para mejorar el entendimiento de su función).
- Campo numérico "**MV**". Misma función que la ruleta.

- Campo numérico "**Límite MV**". Permite al usuario modificar el valor del límite de la variable manipulada (en manual) o la variable de proceso (en automático).
- Campos de la "**Variable de proceso (PV)**". Muestran el nombre que el usuario ha asignado a la variable de proceso y el valor leído del nodo.
- Campos de la "**Variable manipulada (MV)**". Muestran el nombre que el usuario ha asignado a la variable manipulada y el valor asignado al nodo o calculado por el control.

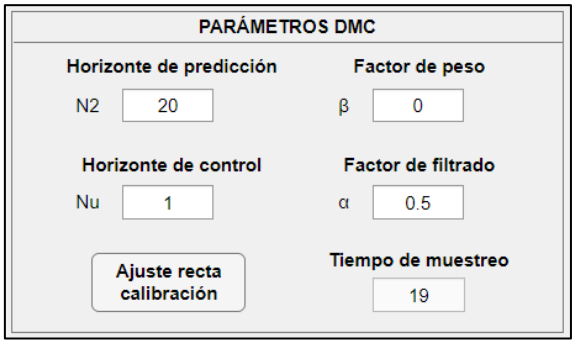


Figura 142: Aspecto del panel "Parámetros DMC".

En el panel "**Parámetros DMC**" (Figura 142) básicamente tenemos 5 campos numéricos y un botón:

- Los cuatro campos superiores están habilitados para que el usuario introduzca el valor del parámetro correspondiente.
- El campo "**Tiempo de muestreo**" muestra el periodo de muestreo del archivo importado. No es editable ya que se toma del archivo importado.
- El botón "**Ajuste recta calibración**" cambia la vista del panel a la mostrada en la Figura 143.

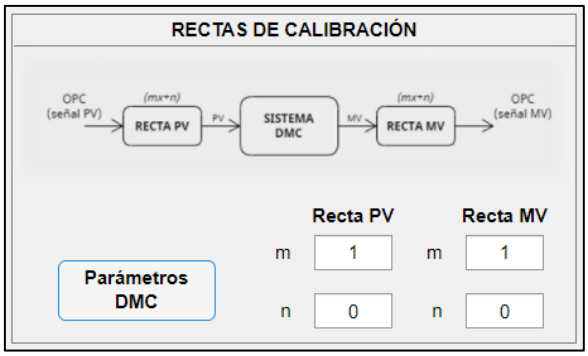


Figura 143: Aspecto del panel "Rectas de calibración".

Ahora, los componentes son los siguientes:

- **Imagen** explicativa del proceso de calibración.
- Cuatro campos numéricos para introducir los valores de las rectas de calibración.
- El botón "**Parámetros DMC**" regresa a la vista de la Figura 142.

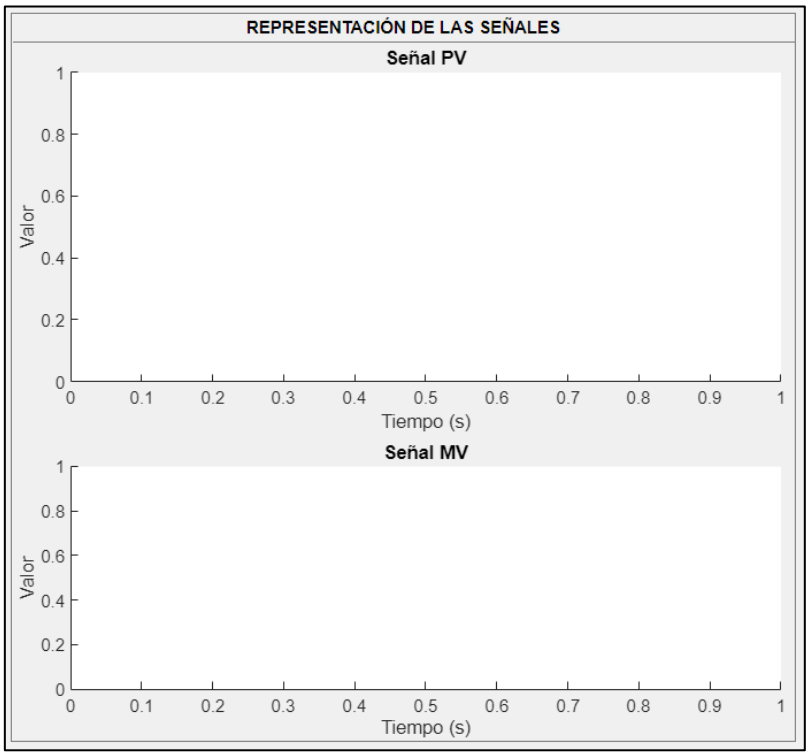


Figura 144: Aspecto del panel "Representación de las señales".

El panel "**Representación de señales**" (Figura 144) se encarga de graficar las señales del sistema, las lecturas de los nodos que hemos elegido previamente. Básicamente son dos gráficas:

- La gráfica "**Señal PV**" mostrará los datos del nodo elegido como variable de proceso. En automático también grafica la consigna y una leyenda para distinguir ambas señales.
- La gráfica "**Señal MV**" mostrará los datos del nodo elegido como variable manipulada.

A1.2 PROCESO DE CARGA

Actualmente, a la interfaz se le pueden incorporar dos tipos de archivos:

I. **Modelo nuevo**: se trata de la identificación del sistema que incorporamos directamente de HIDEN. Solo contiene los parámetros referentes al modelo, no tiene ninguna característica propia de esta aplicación.
II. **Rescatar configuración**: se trata de un archivo generado por esta app, es decir, una configuración con la que ya se ha trabajado. Contiene los parámetros referentes al modelo y también características de esta interfaz.

El primer tipo de archivo se importa a través del panel "Información sobre el modelo" (recuadros rojos). El segundo tipo se incorpora a través del acceso directo señalado o a través del menú "Archivo → Cargar config. DMC" (recuadros azules). Véase Figura 145.

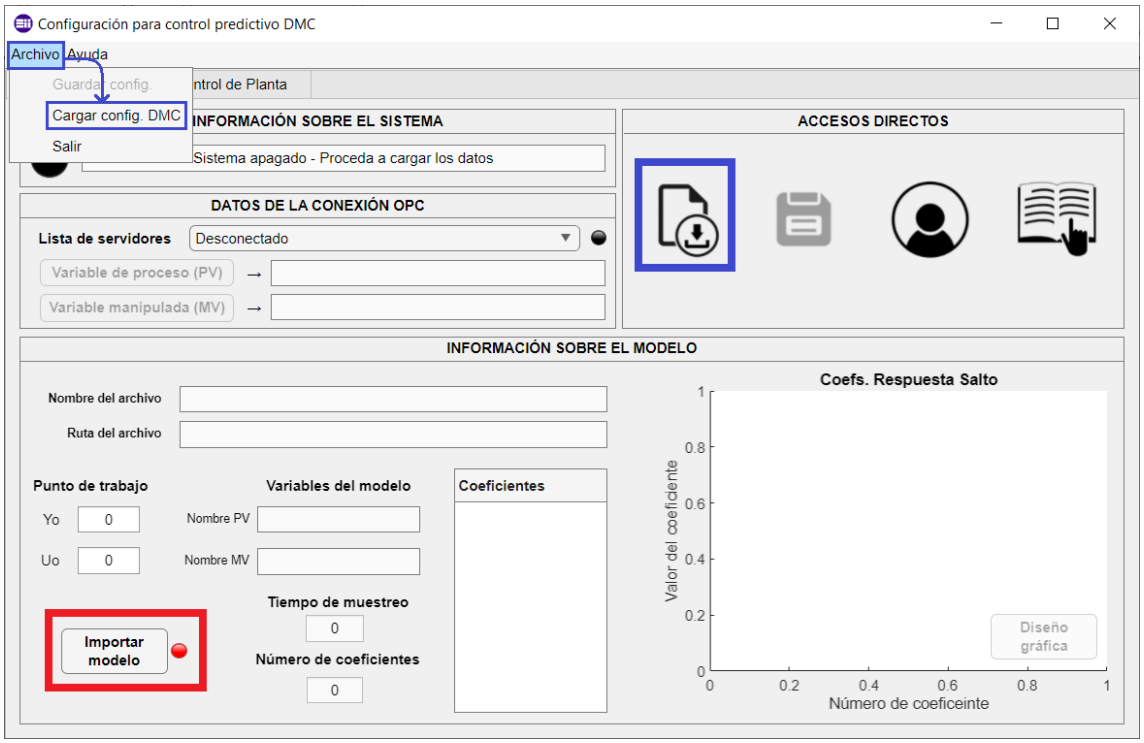


Figura 145: Distintos métodos para importar un archivo al sistema.

Importar un nuevo modelo.

Para importar un modelo nuevo seguiremos el siguiente procedimiento:

1. Pulsamos el botón “Importar modelo” rodeado de un recuadro rojo en la Figura 145.
2. Se abrirá una ventana emergente, como la de la Figura 146, que nos pedirá (literalmente) que seleccionemos el archivo del modelo. Seleccionamos nuestro archivo y hacemos clic en “Abrir” (situado en la zona inferior de la ventana).

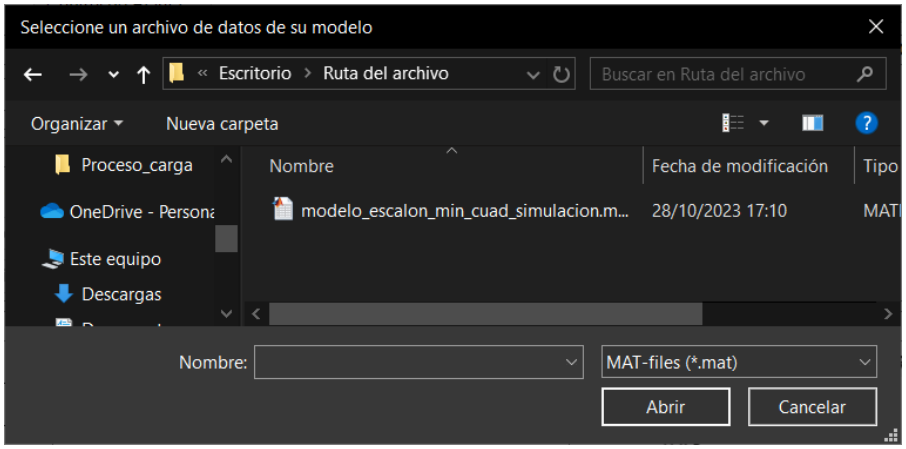


Figura 146: Ventana emergente para la selección y carga de un nuevo modelo.

Es muy importante elegir un archivo de *MATLAB*, esta ventana no nos dejará elegir un archivo que no tenga la extensión *.mat* en un principio. Si se elige otro tipo de archivo el programa notificará de este fallo y cancelará el proceso de carga.

Una vez abrimos el modelo, tenemos varias opciones:

- **Carga correcta** (Figura 147). Se cargan los datos de nuestro modelo. El sistema muestra una ventana emergente con un *tic* verde donde notifica que la carga se ha realizado satisfactoriamente. También avisa de los campos que tiene que introducir el usuario una vez se ha cargado el archivo.
- **Carga incorrecta** (Figura 148). Es común confundirse a la hora de cargar el archivo correcto. En este caso el sistema notifica que el modelo no se corresponde con el archivo buscado con una exclamación roja y cancelará la carga.
- **Carga cancelada** (Figura 149). Si por algún casual cancelamos la carga, el sistema también notificará del fallo en el proceso con una exclamación amarilla.

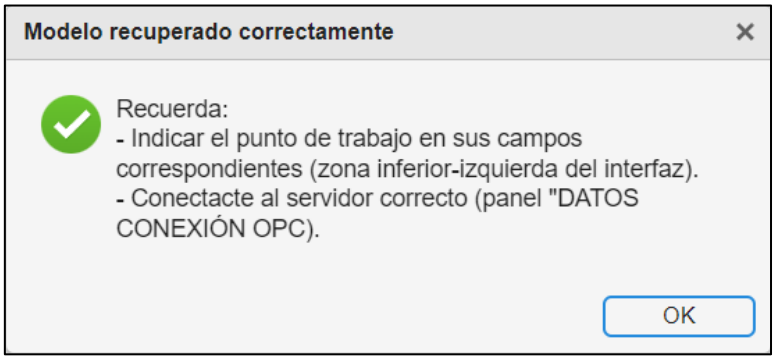


Figura 147: Mensaje de carga de nuevo modelo correcta.

Figura 148: Mensaje de error en carga de nuevo modelo.

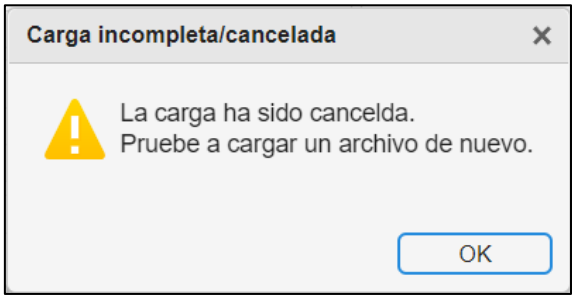


Figura 149: Mensaje de carga de nuevo modelo cancelada.

Si tenemos un modelo/configuración cargada previamente e importamos un modelo nuevo, el sistema borrará todos los datos que tenía y cargará los nuevos (en caso de que el archivo sea correcto). En el caso de que tengamos una conexión con un servidor (dentro de la interfaz), el sistema mantendrá esta conexión, por lo que no es necesario cargar el archivo primero, nos podemos conectar al servidor previamente.

Si el modelo cargado es correcto, el aspecto del interfaz será algo similar a la Figura 150.

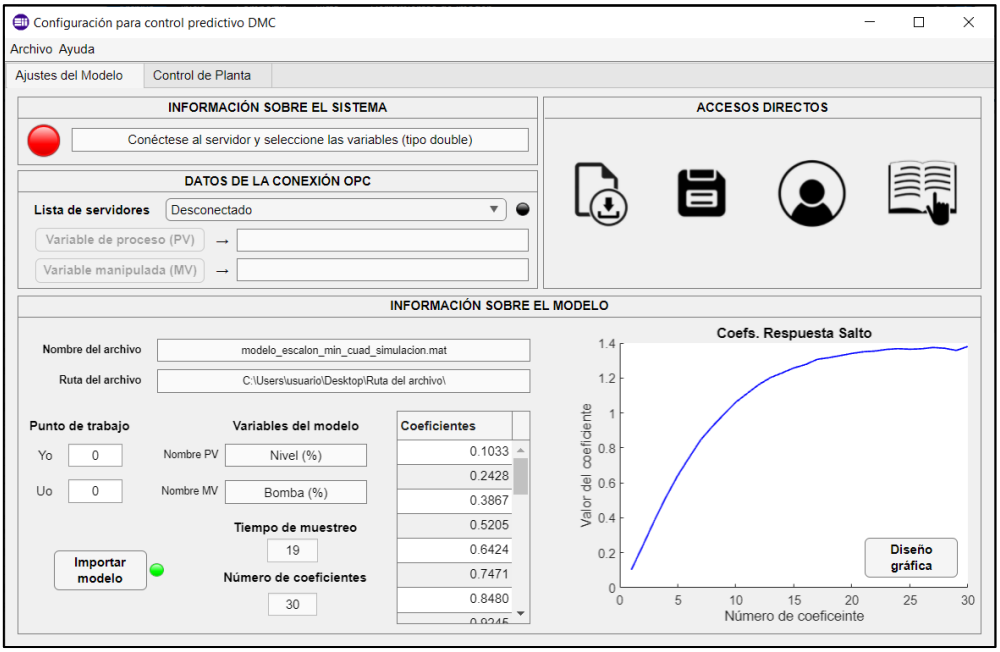


Figura 150: Aspecto del interfaz tras cargar un nuevo modelo con éxito.

Los elementos del panel "Información sobre el modelo" muestran los datos del archivo (es importante rellenar los dos campos numéricos del punto de trabajo, que están inicializados a cero). Además, podemos acceder a la vista de la personalización de la gráfica a través del botón "Diseño gráfica" que se habilita tras acabar el proceso de carga.

El sistema también notifica de las acciones restantes que faltan para iniciar el control a través del panel "Información sobre el sistema".

Importar una configuración DMC.

Para cargar una configuración existente, seguiremos el siguiente procedimiento:

1. Pulsamos el primer icono del panel "Accesos directos". También se puede a través del menú "Archivo → Cargar config. DMC" (recuadros azules de la Figura 145).
2. Se abrirá una ventana emergente que nos pedirá que recuperemos la configuración (Figura 151). Seleccionamos nuestro archivo y hacemos clic en "Abrir".

Es muy importante elegir un archivo de *MATLAB*. En un principio, esta ventana no nos dejará elegir un archivo que no tenga la extensión *.mat*, si se elige otro tipo de archivo el programa notificará de este fallo y cancelará el proceso de carga (además, el sistema guarda las configuraciones con el siguiente formato: "*Conf_DMC_fecha-actual*", de esta manera es más fácil de reconocer el archivo que contiene una configuración).

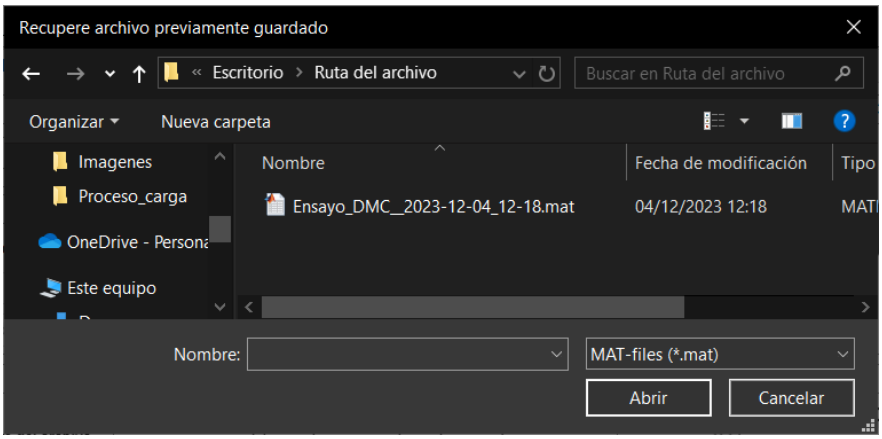


Figura 151: Ventana emergente para la selección y carga de una configuración ya existente.

Una vez se abre el archivo, se pueden dar las siguientes situaciones en función de los datos de nuestro archivo:

- **Carga correcta completa** (Figura 152). El archivo cargado contiene los parámetros de la configuración DMC y los datos de la conexión con el servidor OPC. El sistema recupera los datos de los parámetros y se intentará conectar al servidor. En función de esta conexión podemos encontrarnos las siguientes situaciones:
  - Tanto el servidor como los nodos están presentes. El sistema se conecta sin problemas y notifica al usuario a través de una ventana emergente con un *tic* verde.
  - Solo el servidor está presente. El sistema solo se conecta al servidor y notifica al usuario de la inexistencia de los nodos. También puede ocurrir que solo uno de los nodos esté presente, también es notificado al usuario si fuese el caso.
- **Carga correcta parcial** (Figura 153). El archivo cargado solamente contiene los parámetros de la configuración DMC. El sistema cargará estos valores y notificará al usuario de que falta la conexión con el servidor a través de una ventana emergente con una exclamación amarilla.
- **Carga incorrecta** (Figura 154). Es común confundirse a la hora de cargar el archivo correcto. En este caso el sistema notifica que el modelo no se corresponde con una exclamación roja y cancelará la carga.
- **Carga cancelada** (Figura 155). Si por algún casual cancelamos la recuperación, el sistema también notificará del fallo en el proceso con una exclamación amarilla.

Es importante resaltar que, en el proceso de importación de recuperación, el sistema se deshace la conexión con el servidor anterior (si existe conexión) e intenta conectarse al nuevo servidor. En caso de que no se pueda darse la conexión, el sistema quedaría desconectado de cualquier servidor, permitiendo al usuario elegir una nueva conexión para el nuevo archivo. Con esto se logran disminuir los errores en la configuración del servidor.

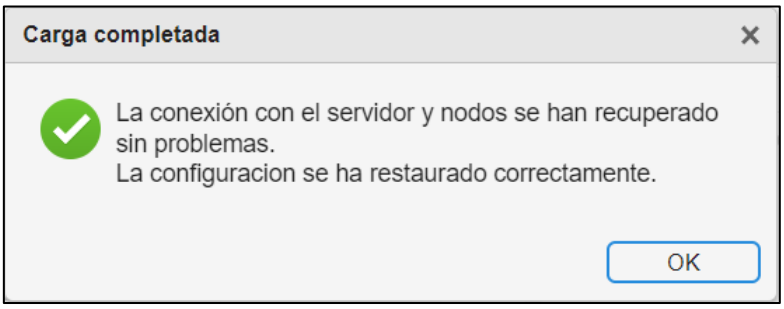


Figura 152: Mensaje de carga correcta de configuración DMC.

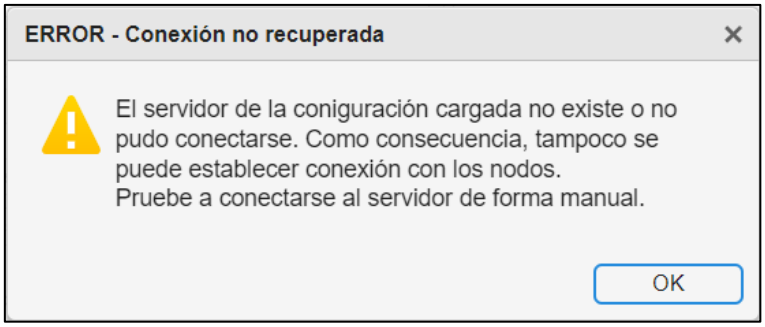


Figura 153: Mensaje de aviso de servidor no conectado al cargar una configuración DMC.

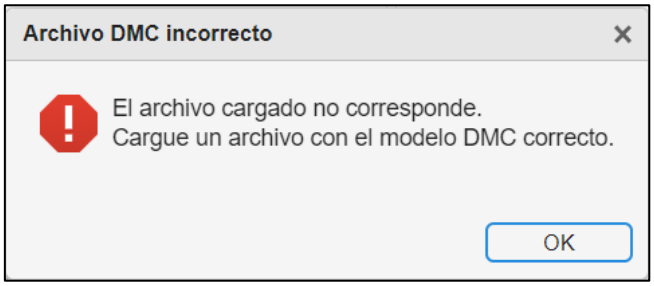


Figura 154: Mensaje de carga incorrecta de configuración DMC.

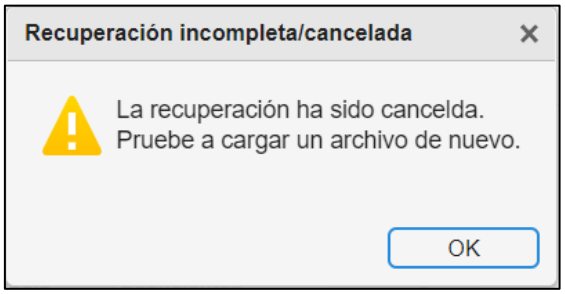


Figura 155: Mensaje de carga cancelada de configuración DMC.

Si la configuración cargada es correcta, el aspecto del interfaz será algo similar a la Figura 156.

Los elementos del panel “Información sobre el modelo” muestran los datos del modelo. Los del panel “Datos de la conexión OPC” muestran la información sobre la conexión con el servidor.

También, podemos acceder a la vista de la personalización de la gráfica a través del botón “Diseño gráfica” que se habilita tras acabar el proceso de carga.

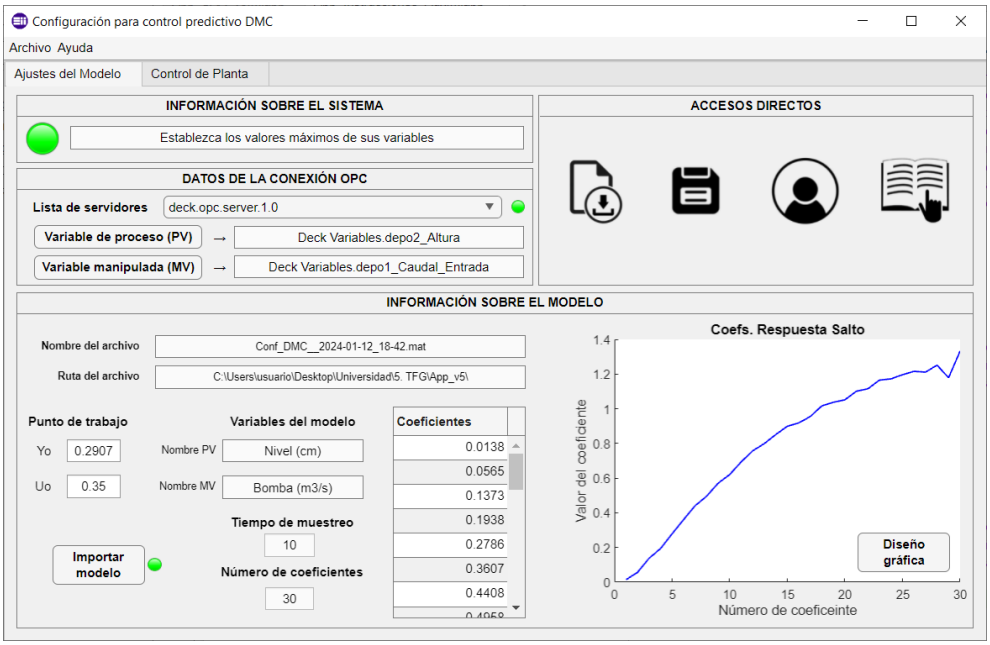


Figura 156: Aspecto del interfaz tras cargar una configuración DMC completa con éxito.

El sistema notifica de las acciones restantes que faltan para iniciar el control a través del panel "Información sobre el sistema".

Recordemos que el archivo de configuración contiene todos los valores de los parámetros, entonces si accedemos a la pestaña "Control de planta" veremos también los valores de los límites de las variables, los parámetros DMC y las rectas de calibración (véase Figura 157).

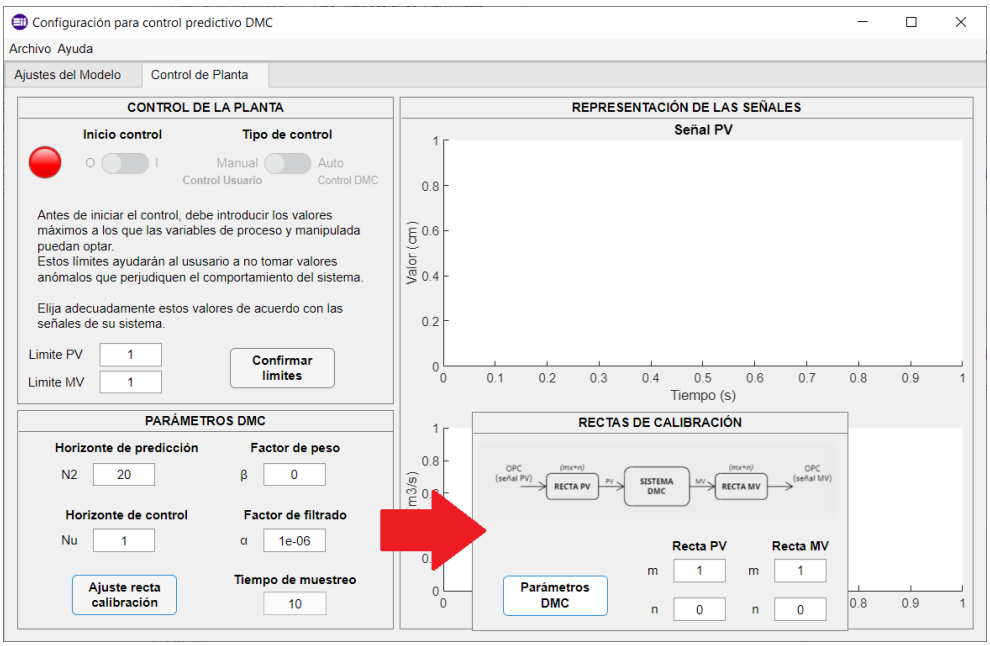


Figura 157: Aspecto de la pestaña "Control de planta" tras cargar una configuración DMC completa con éxito.

Estos son los dos métodos de carga de modelos. En el apartado 10.1.4 INICIO DE CONTROL se detalla el proceso para poder iniciar el control correctamente.

A1.3 PROCESO DE GUARDADO

En este apartado se explica el proceso de guardado, qué se guarda y los requisitos para poder llevar a cabo el proceso.

Para guardar tenemos dos opciones (indicadas en la Figura 158):

1. Icono de guardado en el panel "Accesos directos".
2. A través de la ruta "Archivo → Guardar config.".

Figura 158: Métodos para iniciar el proceso de guardado de configuración.

En un inicio se encuentran bloqueadas. Para habilitarlas tenemos que cumplir los siguientes requisitos:

I. Importar correctamente un modelo o cargar una configuración DMC.
II. El control debe estar detenido.

Si el archivo es correcto y no estamos controlando la planta, ambos botones estarán disponibles para su uso.

Para realizar el proceso de guardado, realizaremos los siguientes pasos:

1. Clicamos en cualquiera de las dos opciones expuestas en la Figura 158. Se nos abrirá una ventana emergente (Figura 159).
2. En la ventana elegimos la ruta y nombre deseados. El sistema ofrece un nombre predeterminado.
3. Damos clic en "Guardar".

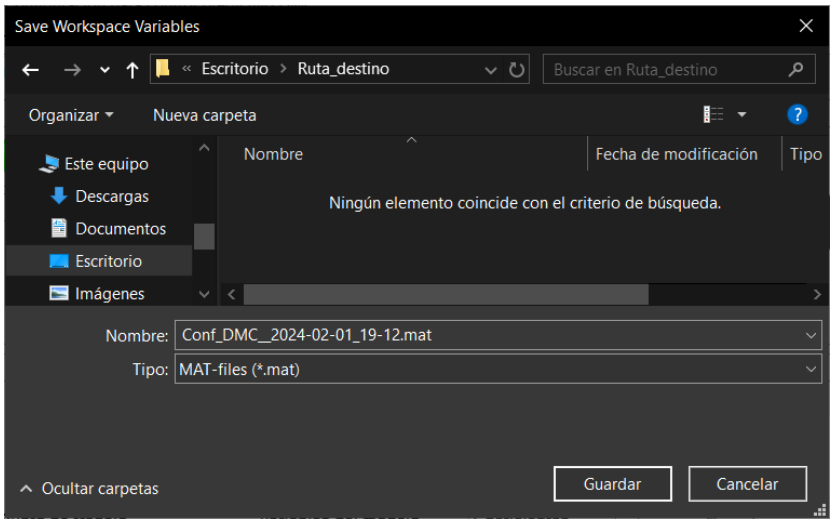


Figura 159: Ventana emergente para guardar la configuración DMC de la aplicación.

Si el proceso de carga se cancela el sistema no avisará, tampoco lo hará si el proceso se completa con éxito. Para ambos casos el panel "Información sobre el sistema" aparecerá un mensaje que indica el fin del proceso de guardado y la lámpara se pondrá de un color azul cian por unos instantes (ver Figura 160).

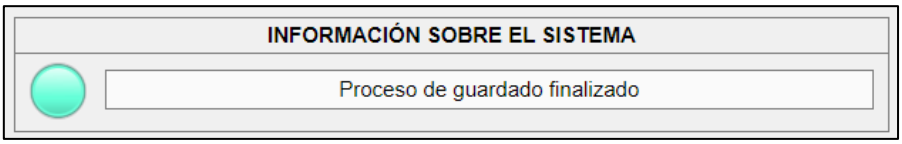


Figura 160: Aviso del sistema sobre la finalización del proceso de guardado.

A continuación, se detallan los parámetros y variables de la configuración que se guardan.

Comenzaremos con las variables y parámetros que se guardarán en el archivo de configuración de la pestaña "Ajustes del modelo".

Con respecto al panel "Información sobre el modelo" se guardan la información resaltada en la Figura 161.

- Nombre y ruta del archivo (recuadro rojo).
- Nombres de las variables del modelo con sus unidades (recuadro azul).
- Tiempo de muestreo y número de coeficientes (recuadro verde).
- Coeficientes de la respuesta salto (recuadro naranja).
- Punto de trabajo (recuadro morado).

Todos los elementos nombrados, a excepción del punto de trabajo, viene dado por el modelo importado de *HIDEN*. Es decir, el punto de trabajo se puede guardar sin haberse manipulado, ambos con valor cero. Es conveniente echar un vistazo a estos parámetros para evitar guardar unos valores erróneos.

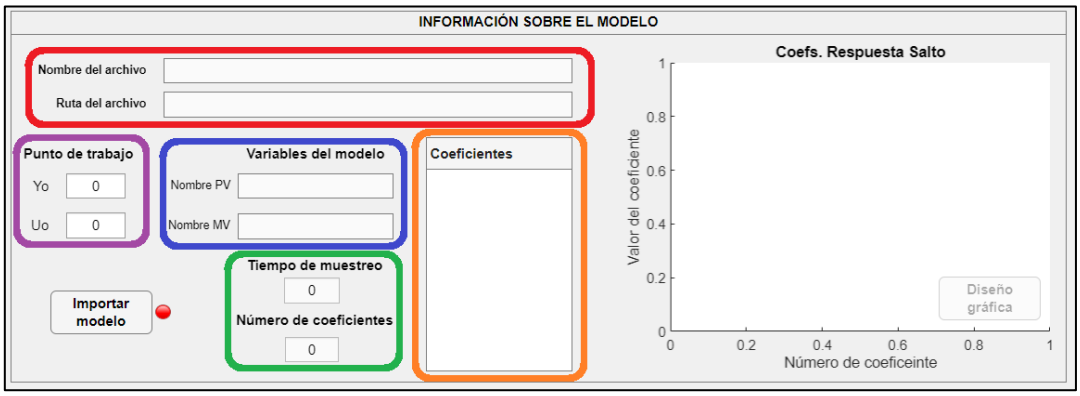


Figura 161: Campos del panel "Información sobre el modelo" que se guardan en el archivo de configuración DMC.

Del panel "Información sobre el modelo" se recogen los siguientes parámetros ():

- Nombre del servidor (recuadro rojo).
- Nombres de los nodos que actúan como variables del sistema (recuadro azul).

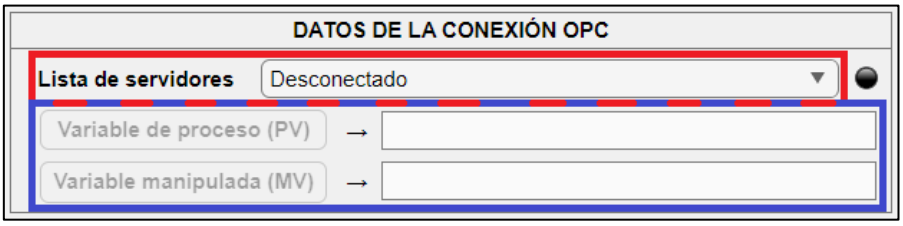


Figura 162: Campos del panel "Datos de la conexión OPC" que se guardan en el archivo de configuración DMC.

No es necesario realizar la configuración con el servidor para poder guardar el archivo. En este caso, se guardan estas variables vacías. A la hora de recuperar la configuración el sistema no podrá conectarse al servidor, pero el resto se carga sin problemas (más información en el apartado 10.1.2 PROCESO DE CARGA).

En cuanto a la pestaña "Control de planta" se guardan las variables de dos paneles.

Figura 163: Campos de la pestaña "Control de planta" que se guardan en el archivo de configuración DMC.

Del panel “Control de la planta” tenemos dos parámetros:

- Límites de las variables del sistema (recuadro rojo de la Figura 163). Al igual que la configuración del servidor, si estos campos no se completan, se guardará ‘cero’ en sus valores. No supone un problema ya que el sistema no dejará iniciar el control (más información en el apartado 10.1.4 INICIO DE CONTROL).

Del panel “Parámetros DMC” y “Rectas de calibración” se guarda lo siguiente:

- Valores de los parámetros DMC: horizontes de predicción y control y factores de peso y filtrado (recuadro azul de la Figura 163).
- Parámetros de las rectas de calibración para las variables del sistema (recuadro verde de la Figura 163).

Si estas variables no se modifican se guardan los valores predeterminados que da la interfaz.

A1.4 INICIO DE CONTROL

Para poder iniciar el control, debemos tener los siguientes requisitos señalados en la Figura 164:

I. Modelo o configuración cargada.
II. Conexión con un servidor OPC disponible y nodos elegidos (estos últimos hacen de las variables del sistema).
III. Límites de las variables establecidos.

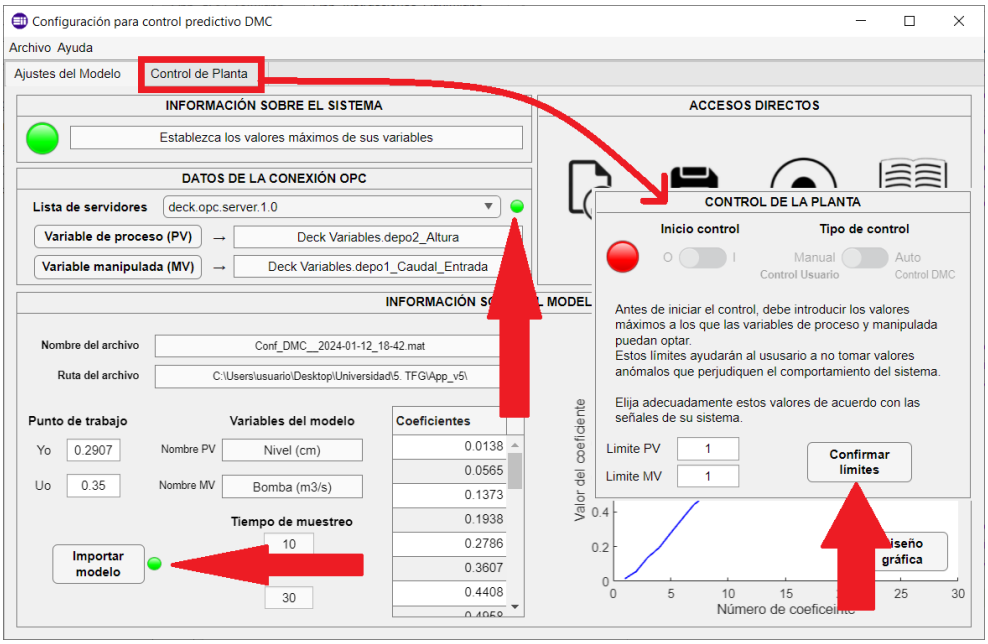


Figura 164: Requisitos necesarios para iniciar el control de la planta.

Hasta que no se den estas tres condiciones, el interruptor para iniciar el control no se habilitará (el campo de texto del panel "Información sobre el sistema" indicará qué tarea le falta por completar).

Como la interfaz puede cargar dos tipos de archivo, tendremos dos "caminos" para iniciar el control: uno para el modelo nuevo y otro para el archivo de configuración (el camino del segundo tipo de archivo es una derivación del primero).

Inicio de control para un nuevo modelo.

Una vez hemos importado el archivo correcto, es muy importante establecer los valores del punto de trabajo en el panel "Información sobre el modelo" (recuadro azul de la Figura 165).

Lo siguiente será conectarnos a un servidor. Desde el panel "Datos de la conexión OPC" abriremos el desplegable "Lista de servidores" (recuadro rojo de la Figura 165). Se mostrarán los servidores disponibles dentro de la red ***localhost***.

Hasta que no estamos conectados a un servidor, no se habilitarán los botones para selección de variables (recuadro verde de la Figura 165).

Se indica la conexión correcta con un servidor a través del color rojo en la lámpara del panel "Datos de la conexión OPC" (ver Figura 165).

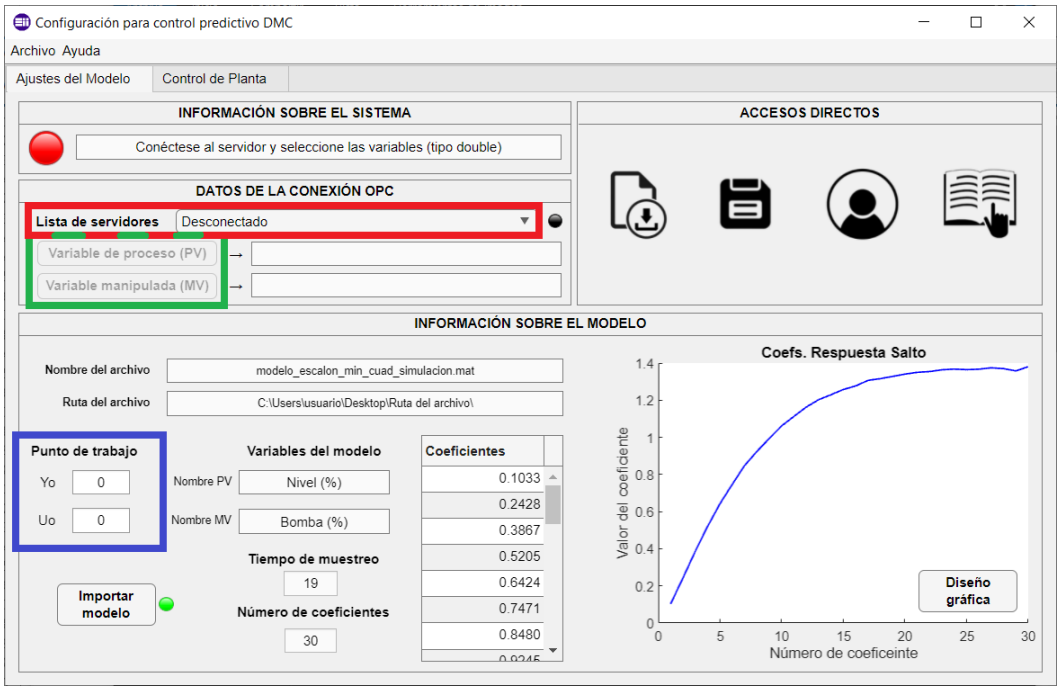


Figura 165: Tareas a completar a partir de un nuevo modelo importado.

Dentro de los distintos servidores seleccionables, podemos encontrarnos con dos tipos:

- Los que no es posible establecer conexión. El sistema avisará al usuario del error (con el mensaje de la Figura 167) y no se conectará.

- Los que sí es posible establecer conexión. El sistema avisará del éxito de la conexión (con el mensaje de la Figura 166).

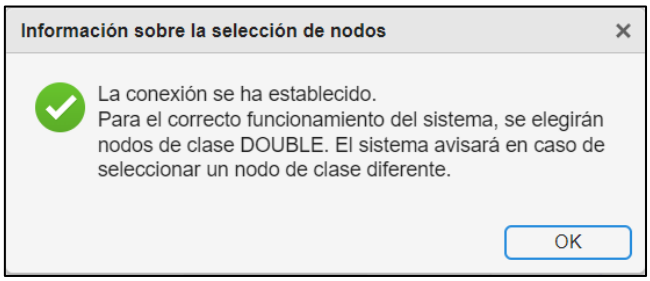


Figura 166: Mensaje informativo sobre éxito en conexión con servidor seleccionado.

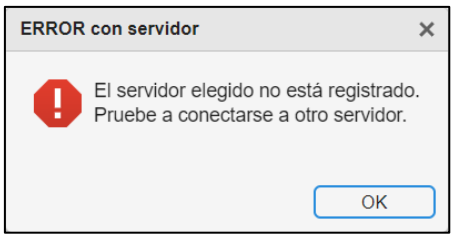


Figura 167: Mensaje informativo sobre fallo en conexión con servidor seleccionado.

Existe la opción de que intentemos cambiar de servidor una vez nos hayamos conectado a alguno. Si no podemos conectarnos al nuevo servidor, el sistema avisará del error y recuperará la conexión con el servidor anterior. Si sí podemos conectarnos, el servidor se conectará y borrará la información de la antigua conexión.

Una vez nos hayamos conectado, los botones para elegir las variables del sistema estarán habilitados (recuadro azul de la Figura 168).

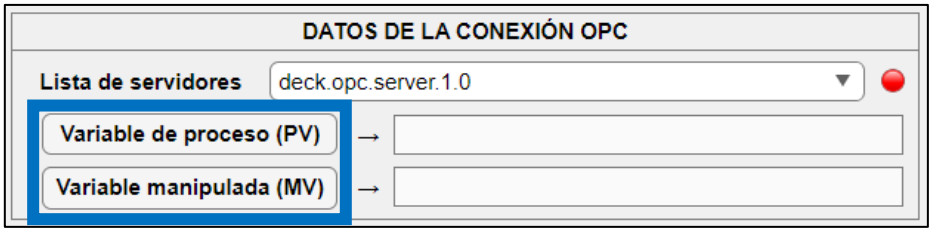


Figura 168: Aspecto del panel "Datos de la conexión OPC" tras la conexión con un servidor.

A partir de este punto, en el momento en el que se cierre la ventana de la conexión con el servidor OPC (mientras seguimos conectados), el sistema avisará del error y cerrará el programa. Para más información, ir al apartado 10.1.5 ERRORES.

Lo siguiente es la selección de las variables del sistema. Estas variables son los nodos disponibles dentro del servidor. Para establecer una variable el proceso es el siguiente:

1. Clicamos el botón de la variable deseada. Se abrirá una ventana emergente con el nombre "*Browse Name Space*" (Figura 169).
2. Seleccionamos **un único nodo de tipo *double***. En caso contrario el sistema avisará del error específico cometido en la selección (Figura 170 y Figura 171).
3. Clicamos el botón "OK" de la ventana emergente de la Figura 169.

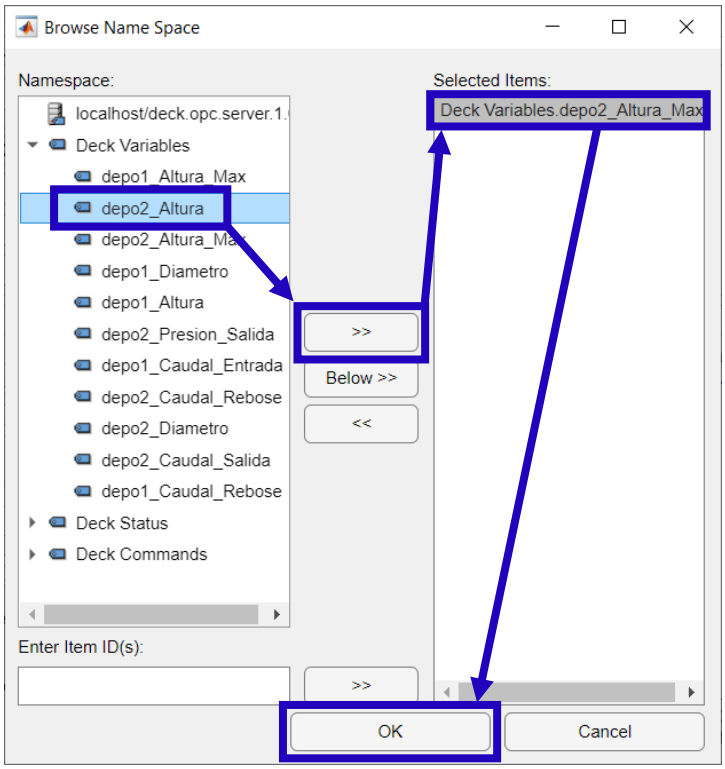


Figura 169: Ventana emergente de selección de nodo.

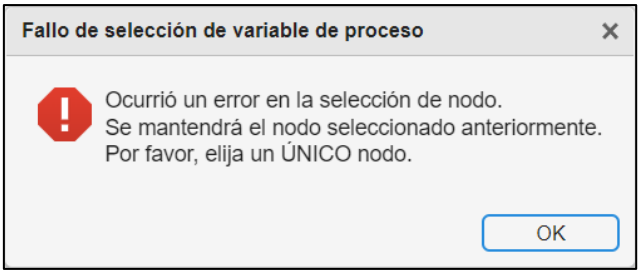


Figura 170: Error en la selección de nodo por selección múltiple.

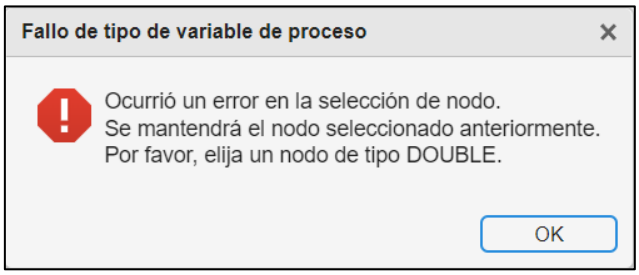


Figura 171: Error en la selección de nodo por no escoger uno de tipo *double*.

Si todo va bien, los nombres de los nodos aparecerán en los campos de texto de la variable correspondiente (Figura 172). Cuando ambos nodos hayan sido seleccionados, el color de la lámpara cambiará a verde. Esto indica que ya hemos finalizado la configuración del servidor y podemos pasar al siguiente paso.

En caso de querer cambiar una variable existente, el usuario debe repetir los pasos explicados y el nodo se sustituirá.

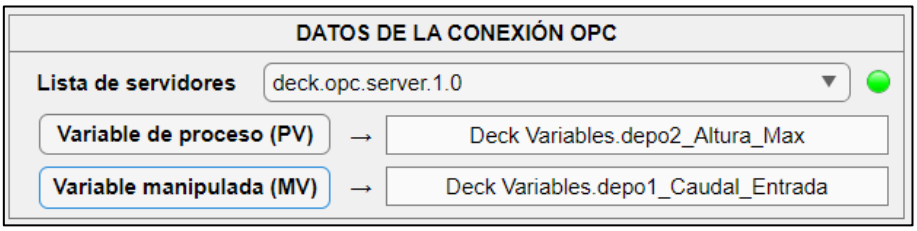


Figura 172: Aspecto del panel "Datos de la conexión OPC" tras la selección de ambas variables.

Una vez cargados estos parámetros, el aspecto del interfaz debería ser similar al mostrado en la siguiente figura:

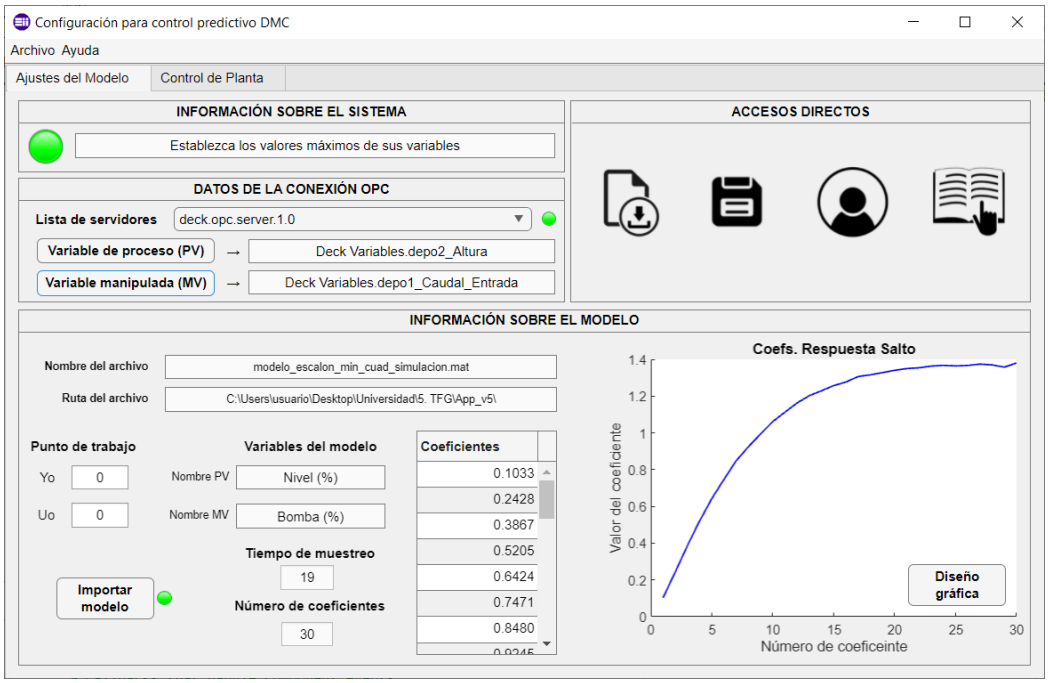


Figura 173: Aspecto del interfaz tras completar las tareas para iniciar el control de la pestaña "Ajustes del modelo".

Esta pestaña ya tiene toda la información necesaria, todos los campos requeridos ya están completos. Si nos fijamos en el panel "Información sobre el sistema" la lámpara está en color verde (quiere decir que ya hemos finalizado la carga de esta pestaña) y se muestra un texto que indica: "Establezca los valores máximos de los límites". Entonces, lo siguiente es acceder a la pestaña "Control de planta" y continuar el proceso.

Lo último que nos restaría para poder iniciar el control es la configuración de los límites.

Inicialmente, el aspecto del panel "Control de la planta" se muestra en la Figura 174. Como se puede ver, no es posible iniciar el control, el interruptor está bloqueado. Para desbloquearlo hay que pulsar el botón "Confirmar límites", que también se encuentra inhabilitado. Para desbloquear el botón debemos introducir los valores de los límites de los nodos que hacen de nuestras variables. Dichos valores deben ser distintos de cero, si se intenta meter esta cifra, el botón se bloqueará impidiendo establecer los límites.

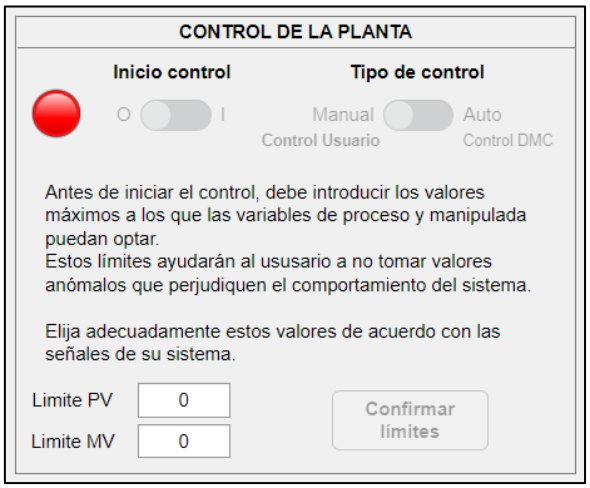


Figura 174: Aspecto inicial del panel "Control de la planta".

Una vez se introducen unos valores correctos, el botón se habilitará y podremos confirmar los límites (Figura 175). Si lo pulsamos el aspecto del panel debería cambiar al de la Figura 176. Además, el interruptor para iniciar el control quedará desbloqueado.

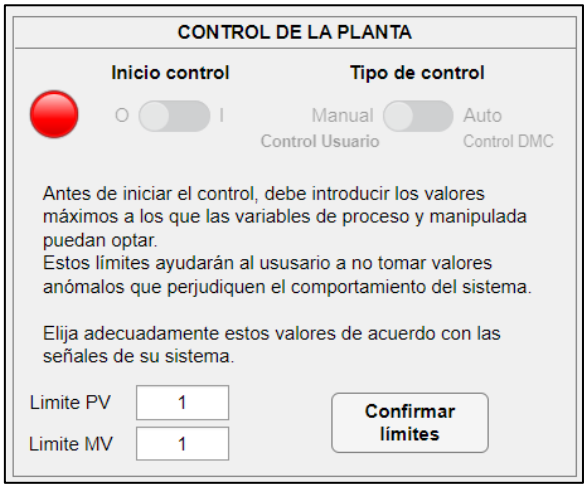


Figura 175: Aspecto del panel "Control de la planta" tras introducir unos límites válidos.

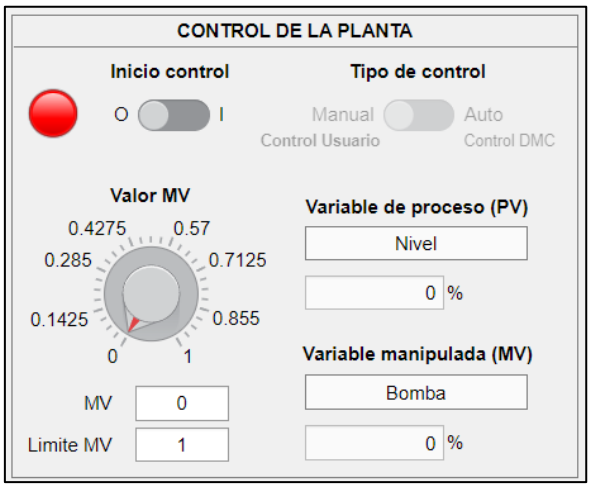


Figura 176: Aspecto del panel "Control de la planta" tras pulsar el botón "Confirmar límites".

Al pulsar el botón, también estamos dando los valores de la ruleta y campo numérico "MV" (en manual) o "PV" (en automático). Con esto, ya habríamos configurado el sistema para controlar la planta.

Los límites se pueden volver a modificar en cualquier momento (siempre con un valor superior a cero). Con el interruptor "Tipo de control" en modo "Manual" podremos establecer el límite de la variable manipulada y en "Automático" el de la referencia.

Inicio de control para una configuración existente.

Iniciar el control a través de una configuración previa es muy sencillo. Supondremos que cargamos un archivo que contiene todos los campos y se puede conectar al servidor sin problemas. Simplemente tendremos que confirmar los límites de las variables y llegaremos al punto de vista de la Figura 176.

En caso de cargar un archivo al que le falte algún campo, basta con repetir los pasos mostrados en el apartado Inicio de control para un nuevo modelo.

Antes de explicar qué sucede al hacer un cambio en el tipo de control, es necesario aclarar lo siguiente:

- Es muy importarte no olvidarse de establecer los parámetros adecuados de las rectas de ajuste (Figura 143), si no la interfaz no funcionará correctamente (de hecho, puede provocar la necesidad de un reinicio de la aplicación).
- De forma predeterminada, los valores se ajustan a los de una planta simulada: '1' para la pendiente y '0' para la ordenada en el origen. Para una planta real es casi imposible que coincida, así que se recomienda cambiar dichos valores.

Inicio de control y cambio de tipo.

Una vez tenemos el sistema preparado podemos iniciar el control. Antes de iniciarlo, es conveniente tener las siguientes cosas en cuenta:

- La primera vez que se arranca el control se hace en modo manual siempre (recuadro rojo de la Figura 177). Una vez iniciado se podrá detener/reanudar el control en el modo que el usuario desee.
- Se pueden cambiar los valores tanto la ruleta como el campo numérico "MV" (manual) o "SP" (automático) con el control apagado (recuadro azul de la Figura 177). Estos valores no se escribirán en la variable hasta que iniciemos el control de nuevo.
- Todos los campos numéricos de este panel están redondeados para facilitar la lectura rápida de los valores (flechas verdes de la Figura 177). A la hora de realizar los cálculos el sistema sí tiene en cuenta el valor completo.
- Como ya se indicó, los valores de los límites de las variables se pueden modificar (recuadro naranja de la Figura 177). Se recomienda (una vez establecidos los

límites) no manipular estos campos, si aun así fuera necesario, se recomienda cambiar los valores antes de iniciar el control y con el control apagado.

Una vez expuestas estas premisas, se continúa con el control manual.

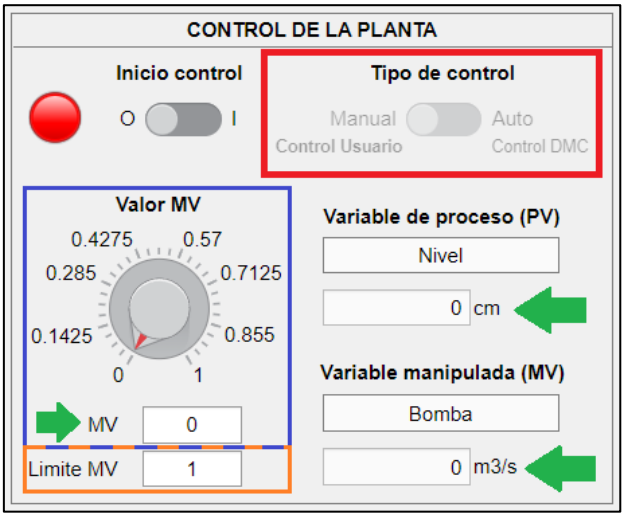


Figura 177: Aclaraciones previas al inicio de control.

Para iniciar el control manual debemos iniciar el control (si es la primera vez en la interfaz que activamos el control) o poner el interruptor "Tipo de control" en "Manual" (si estamos en modo automático). En el momento que se inicia el control el sistema realiza las siguientes acciones:

- Bloquea los elementos cuyas acciones puedan comprometer el funcionamiento del interfaz (como cargar un modelo nuevo, cambiar el servidor o sus variables, etc.).
- Escribe el valor de la variable manipulada en el nodo que actúa como tal.
- La lámpara adopta un color azul cian, indicativa del color manual.
- Comienza a tomar datos del servidor cada segundo de tiempo.

En el panel "Representación de las señales" se graficarán los datos de la variable de proceso en la gráfica superior y los de la variable manipulada en la inferior (ver Figura 178). Las gráficas se ajustan de manera automática mostrando los últimos 100 segundos en el eje del tiempo y los valores lo más ajustados posibles en el eje Y (de manera que dará la sensación que nos estamos desplazando en el tiempo).

Si se quisiera cambiar el valor de la variable manipulada, se puede manipular la ruleta o el campo numérico del panel "Control de la planta" y automáticamente se enviará el valor al nodo (tras pasar por las rectas de ajuste). Los parámetros DMC se pueden manipular, pero no tendrán efecto en el sistema, ya que se usan para los cálculos en automático.

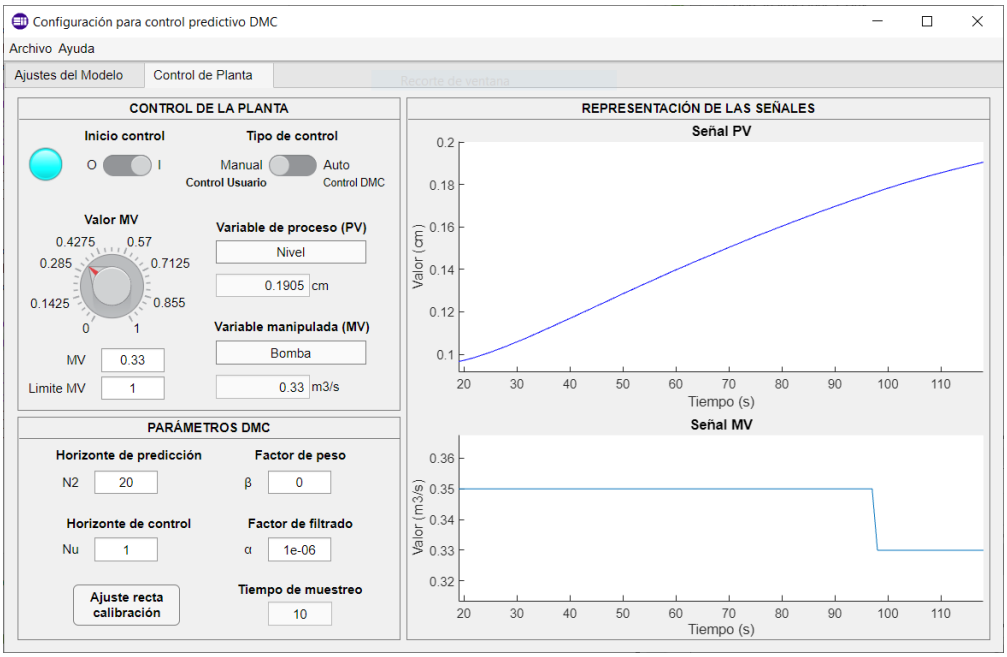


Figura 178: Aspecto de la interfaz tras iniciar el control en modo manual.

Para iniciar el control automático debemos poner el interruptor "Tipo de control" en "Automático" (el control debe estar iniciado). En este caso, el sistema realiza las siguientes acciones:

- Bloquea (o mantiene bloqueados) los elementos cuyas acciones puedan comprometer el funcionamiento del interfaz.
- Establece el valor de la consigna.
- La lámpara adopta un color verde, indicativa del control automático.
- Comienza a tomar datos del servidor cada segundo.
- Comienza a calcular los valores de la variable manipulada a través del control predictivo DMC. La frecuencia de cálculo se establece con el tiempo de muestreo leído del modelo/configuración cargada.

El panel "Representación de las señales" mantendrá las mismas características que en el control manual, a excepción de la gráfica de la señal de la variable de proceso. Ahora se mostrarán dos rectas: la señal de la variable de proceso (azul) y la consigna (roja), también aparece una leyenda en la esquina superior izquierda para facilitar la comprensión (véase Figura 179).

Si se quisiera cambiar el valor de la consigna, se puede manipular la ruleta o el campo numérico y automáticamente se cambiará su valor. Los parámetros DMC se pueden manipular en todo momento y se tendrán en cuenta en la siguiente iteración del control.

Es importante tener en cuenta lo siguiente de cara a los parámetros de los horizontes:

- Los límites del horizonte de predicción son: [2, ∞].
- Los límites del horizonte de control son: [1, horizonte de predicción-1].

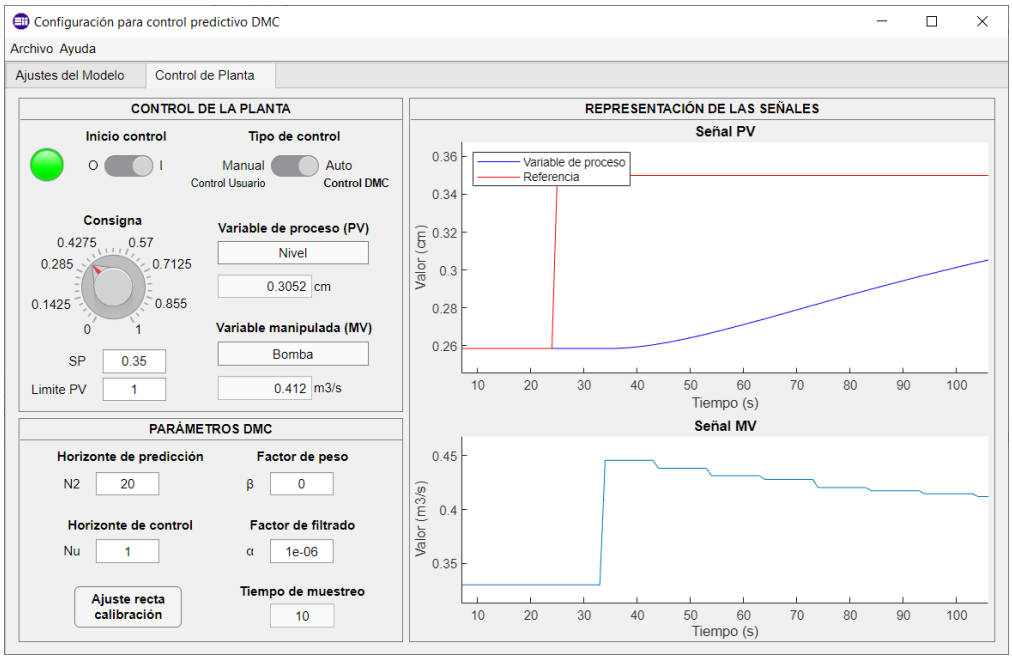


Figura 179: Aspecto de la interfaz tras iniciar el control en modo automático.

Para finalizar la explicación del proceso de inicio de control, se detallan las acciones que realiza internamente la interfaz con respecto a los cambios en el control.

1. **Cambio de manual a automático** (Figura 180). El sistema guarda el último valor leído del nodo de la variable de proceso y lo establece como consigna. Se empieza a mostrar la señal de consigna en la gráfica de "Señal PV" con la leyenda.
2. **Cambio de automático a manual** (Figura 181). El sistema asigna el último valor del control calculado a la variable manipulada. Se dejan de graficar la señal de la consigna y la leyenda.
3. **Reanudación del control en modo manual** (Figura 182). El sistema asigna el valor de la ruleta a la variable manipulada y comienza a leer datos de nuevo.
4. **Reanudación del control en modo automático** (Figura 183). El sistema recalcula el valor a partir de los últimos datos leídos y continua el proceso.

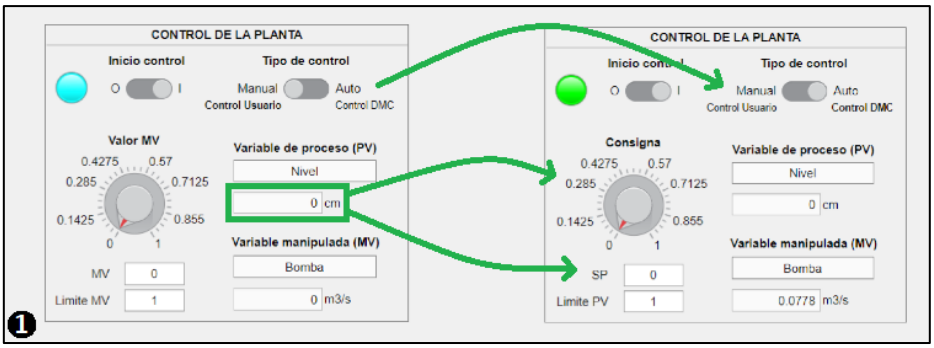


Figura 180: Cambios en la interfaz de manual a automático.

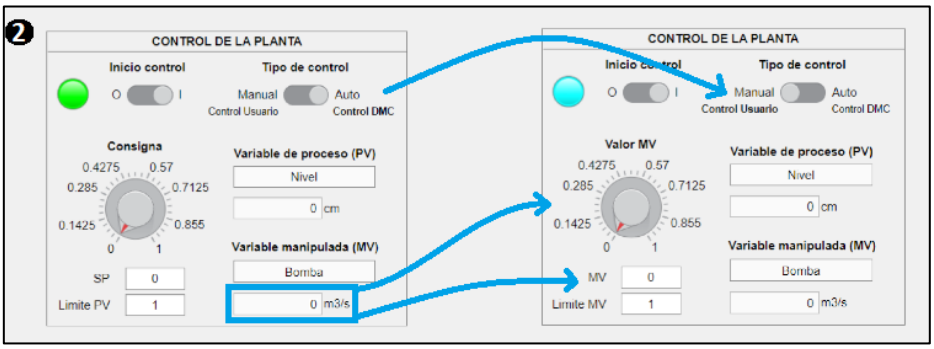


Figura 181: Cambios en la interfaz de automático a manual.

Figura 182: Cambios en la interfaz al reanudar el control manual.

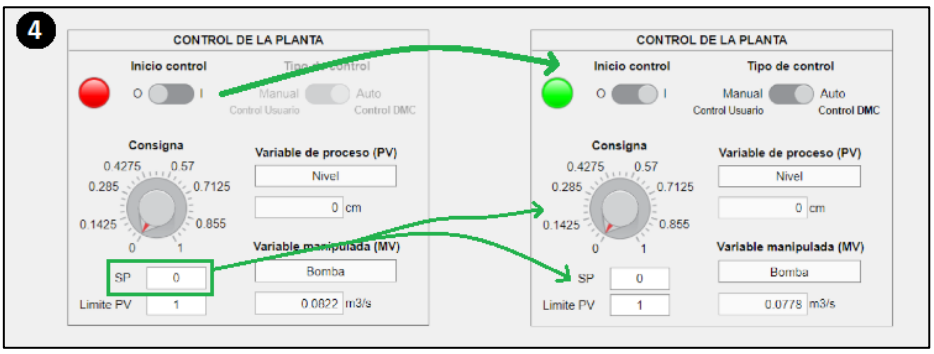


Figura 183: Cambios en la interfaz al reanudar el control automático.

A1.5 ERRORES

Recordemos que, en esta interfaz, podemos obtener cuatro tipos de mensajes (todos mostrados en la Figura 184):

- **Éxito**: círculo verde con una "V" blanca en el centro.
- **Información**: círculo azul con una "i" blanca en el centro.
- **Advertencia**: triángulo amarillo con una exclamación blanca en el centro.
- **Error**: octógono rojo con una exclamación blanca en el centro.

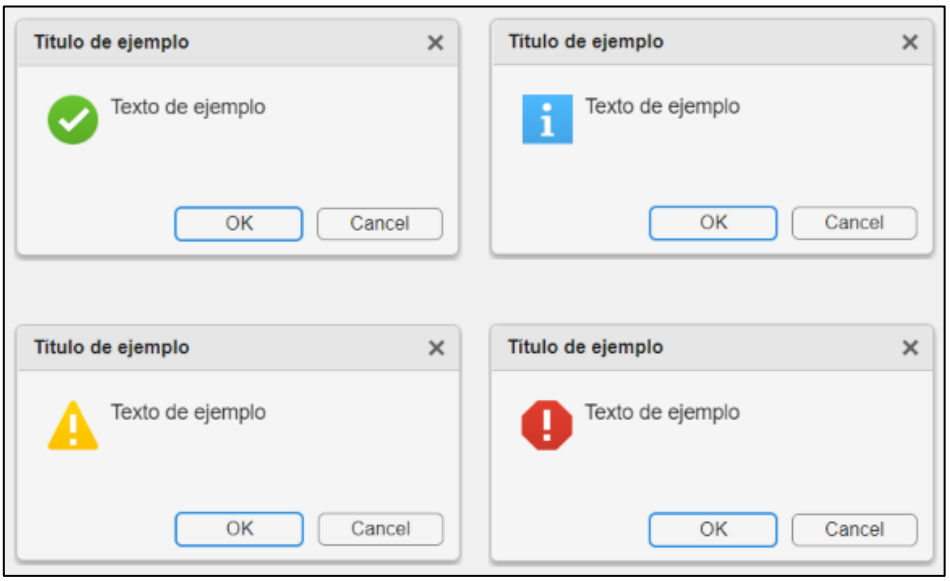


Figura 184: Tipos de mensajes emergentes posibles en esta interfaz.

En otros apartados (como por ejemplo 10.1.2 PROCESO DE CARGA) se han mostrado varias de estas ventanas emergentes. En este apartado veremos aquellas que pueden surgir por aquellos errores ajenos al usuario, es decir, que no se producen por la interacción del usuario con la aplicación. Nos encontraremos con dos tipos de errores: internos y externos.

Errores externos.

Los errores externos ocurren cuando se produce algún fallo con los elementos ajenos a la aplicación. En nuestro caso, solo es posible que ocurra un caso: pérdida de conexión con el servidor OPC. Esto básicamente ocurre si se cierra la ventana de la conexión con el servidor o por un error interno del propio servidor ajeno a la acción del usuario.

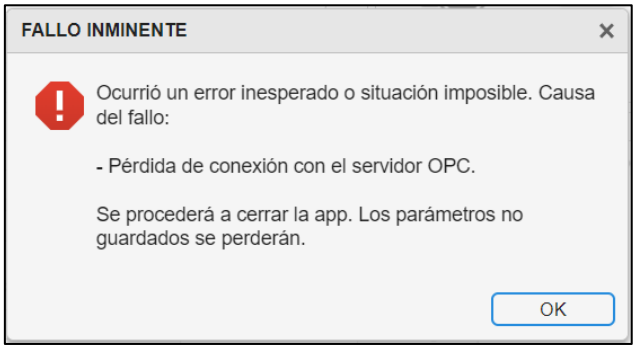


Figura 185: Mensaje de aviso de pérdida de conexión con el servidor.

En este caso el sistema mostrará un mensaje de aviso informando del error y sus consecuencias, como el mostrado en la Figura 185. Una vez cerrado el mensaje, la interfaz iniciará el proceso de cierre y se perderá la configuración que no se haya guardado. Por tanto, es muy importante tener cuidado con la ventana de la conexión y solamente cerrarla cuando se haya acabado de manipular y detener la aplicación.

Errores internos.

Los errores internos se dan cuando la programación de la aplicación falla. Son situaciones que no deberían suceder a no ser que se modifique el código interno del sistema.

Básicamente podremos distinguir los siguientes errores internos:

- Fallo en la variable que actúa como memoria de confirmación de los límites.

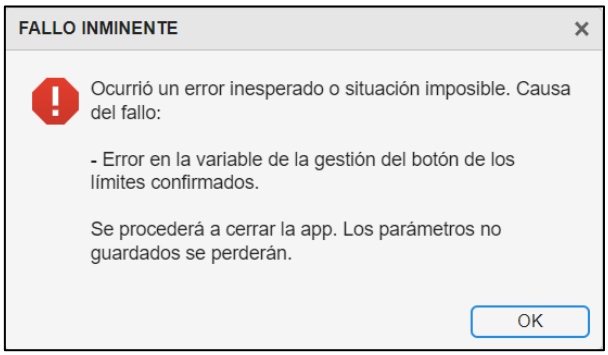


Figura 186: Mensaje de aviso de cierre de la interfaz por fallo de la variable interna de la gestión del botón "Confirmar límites".

- Fallo en la variable que actúa como memoria de la configuración visual del panel "Información sobre el modelo".

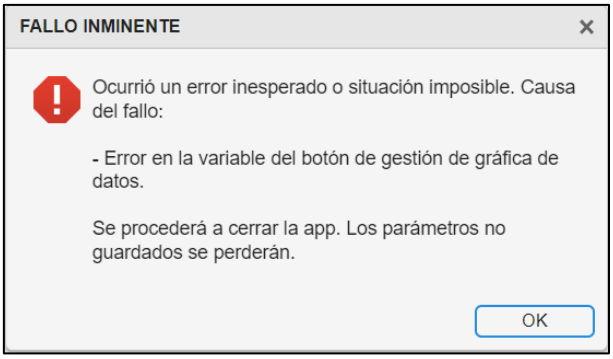


Figura 187: Mensaje de aviso de cierre por error en la variable de la gestión visual del panel "Información sobre el modelo".

- Fallo en la variable de control del estado de habilitación del interruptor "Inicio de control".

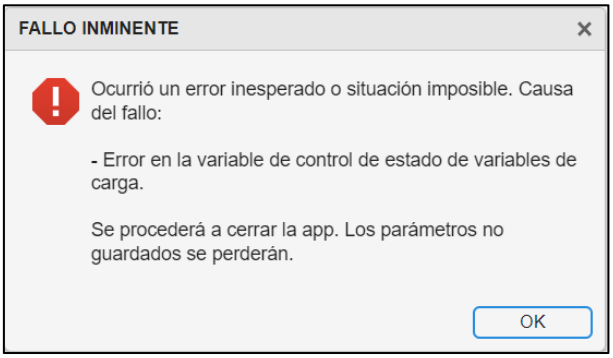


Figura 188: Mensaje de aviso de cierre por error en la variable de la habilitación del interruptor de inicio.

- Fallo en el estado del interruptor “Inicio de control”.

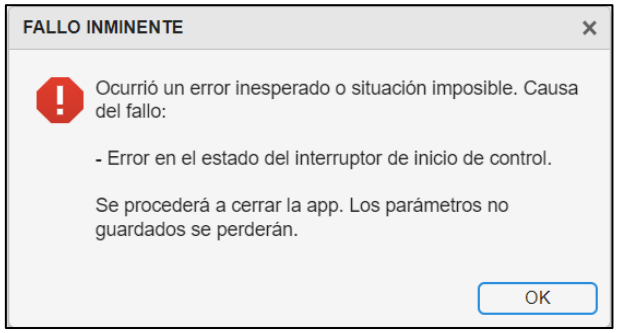


Figura 189: Mensaje de aviso de cierre por error en la variable de la gestión del estado del interruptor de inicio.

- Fallo en el estado del interruptor “Tipo de control”.

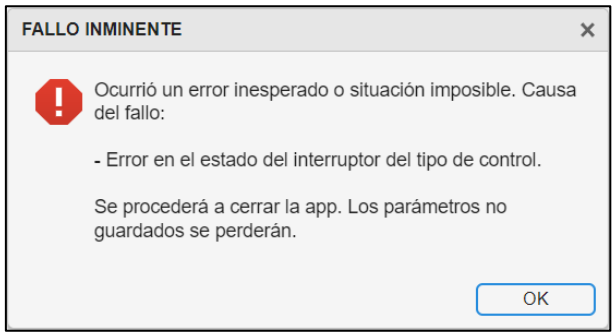


Figura 190: Mensaje de aviso de cierre por error en la variable de la gestión del estado del interruptor del tipo de control.

- Fallo en la variable que actúa como memoria de la configuración visual del panel “Parámetros DMC” y “Rectas de calibración”.

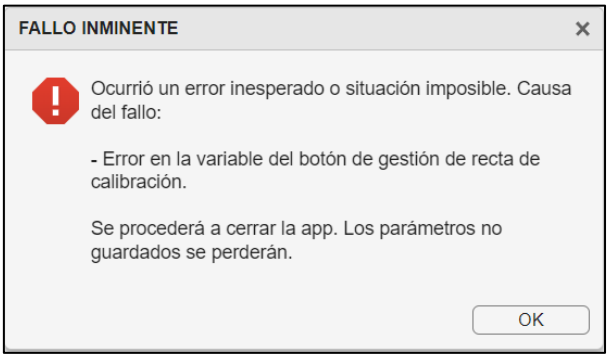


Figura 191: Mensaje de aviso de cierre por error en la gestión visual del panel "Parámetros DMC" & "Rectas de calibración".

Al igual que para los errores externos, una vez se haya eliminado la ventana del mensaje, el sistema iniciará el proceso de cierre de la aplicación.

Como se ha indicado, es muy improbable que estos errores lleguen a suceder, el usuario no debería preocuparse, no se llegarán a estas situaciones.

Para finalizar, a modo de extra, se explicará cómo funcionará el proceso de cierre.

Proceso de cierre.

Básicamente, para cerrar la aplicación tenemos dos caminos posibles:

I. A través del icono de cierre normal (situado en la esquina derecha superior de la ventana mostrada en la Figura 192).
II. A través del menú "Archivo -> Salir" (situado en la esquina izquierda superior de la ventana mostrada en la Figura 192).

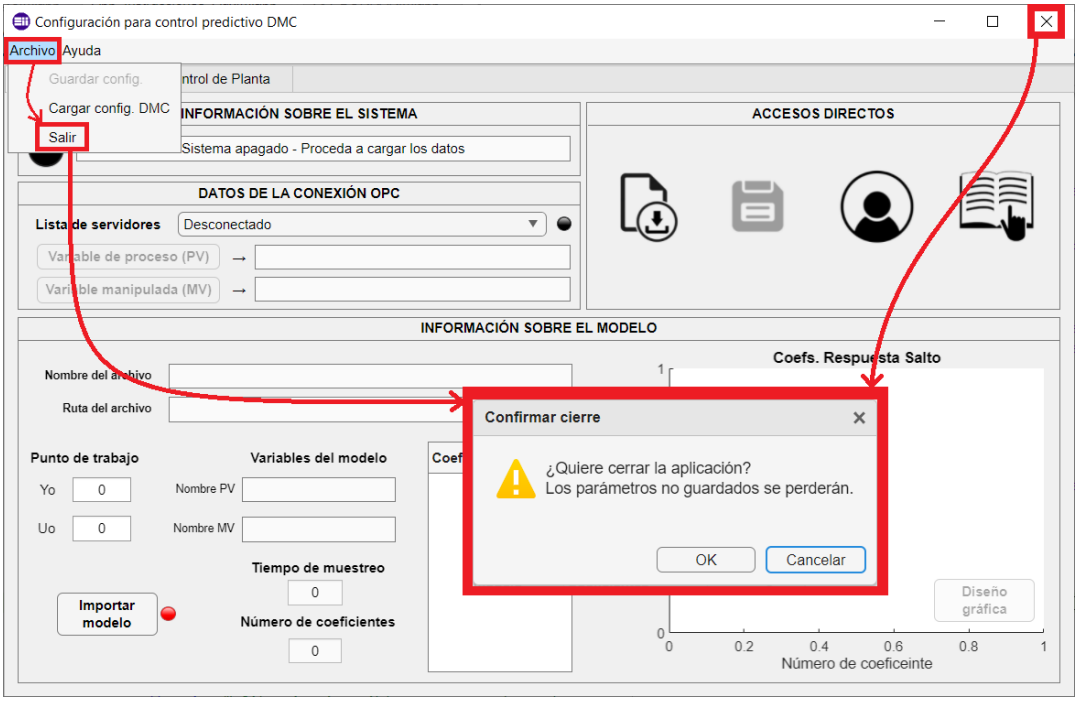


Figura 192: Opciones para iniciar el proceso de cierre de la aplicación.

Con cualquiera, se mostrará un mensaje de advertencia de cierre y el usuario tendrá dos elecciones: "OK" y "Cancelar" (el mensaje se encuentra enmarcado en un recuadro rojo en la Figura 192). Si se pulsa "OK" el sistema comienza el proceso de cierre. Si se pulsa "Cancelar" el sistema vuelve al estado anterior al de ejecutar el proceso de cierre.

ANEXO A2: CONJUNTO DE ARCHIVOS

El conjunto de archivos que compone la aplicación se ha adjuntado en un archivo comprimido en la sección de anejos. Los documentos que contiene son los siguientes:

- "**App_DMC_SaulAntolin.mlapp**": código fuente de la interfaz, es la aplicación principal. Se abre a través de *MATLAB* o *App Designer* y permite la visualización, edición y ejecución de la aplicación a través de estas herramientas.
- "**App_Desarrollador_Aux.mlapp**": código de la ventana emergente que contiene la información del desarrollador de la aplicación y de la escuela de Ingenierías Industriales de la Universidad de Valladolid. Se abre a través de *MATLAB* o *App Designer* y permite la visualización, edición y ejecución de la aplicación a través de estas herramientas. Es llamada a través de la aplicación principal.
- "**App_Instrucciones_Aux.mlapp**": código de la ventana emergente que contiene el manual de usuario. Se abre a través de *MATLAB* o *App Designer* y permite la visualización, edición y ejecución de la aplicación a través de estas herramientas. Es llamada a través de la aplicación principal.
- "**dmc.m**": código que realiza los cálculos del control predictivo DMC. Se abre a través de *MATLAB* y es llamada a través de la aplicación principal.
- "**Imagenes**": carpeta que contiene todas los iconos y fotos que aparecen en la interfaz.
- "**DMC control SISO Systems App.prj**": guarda la información de la aplicación (como el nombre del desarrollador, la versión de la interfaz, un resumen de la misma, etc.) para crear el archivo de instalación para *MATLAB*. Se ejecuta a través de *App Designer* (la aplicación principal debe estar abierta).
- "**Control control SISO Systems App.mlappinstall**": instalador de la aplicación. Para instalarlo en *MATLAB*, simplemente estableceremos la ubicación de este archivo como ruta principal y lo ejecutaremos dentro de *MATLAB*. Se generará un icono (con el mismo nombre del archivo) en la sección de apps. Se inicia haciendo clic en su icono.