Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 17 (2020) 1–9

www.revista-riai.org

Generación automática de problemas de diseño de controladores para sistemas lineales autoevaluables con Doctus

C. Sánchez[a,*], D. Muñoz de la Peña[b], F. Gómez-Estern[c]

[a] *Universidad Loyola Andalucía. Avda. de las Universidades s/n. 41704 Dos Hermanas, Sevilla*
[b] *Departamento de Ingeniería de Sistemas y Automática. Universidad de Sevilla. Camino de los Descubrimientos s/n. 41013 Sevilla*
[c] *Universidad Loyola Andalucía. Avda. de las Universidades s/n. 41704 Dos Hermanas, Sevilla*

Resumen

En este trabajo se presenta una aplicación que permite a un profesor generar de forma automática ejercicios de diseño de controladores para sistemas lineales para la plataforma de e-learning Doctus. Doctus es una aplicación que permite automatizar la recogida, almacenamiento y evaluación de ejercicios de contenido científico-técnico. En general, para utilizar Doctus es necesario escribir scripts de MATLAB, tanto para la evaluación automática de la respuesta de los alumnos como para la generación de enunciados personalizados. La aplicación que se presenta es una extensión de Doctus destinada a simplificar este trabajo, permitiendo generar ejercicios personalizados mediante la aplicación de un escalado temporal y de ganancia a partir de un problema semilla que proporciona el profesor. La aplicación presentada resuelve, por lo tanto, dos problemas: primero el de facilitar el uso de Doctus a profesores no expertos, y segundo el de proporcionar a cada estudiante un ejercicio individualizado con solución y dificultad garantizadas.

Palabras Clave:
Educación en control, Herramientas de ayuda al aprendizaje, Enseñanza en control, Control PID, Sistemas de control lineales, Benchmarks, Herramientas software, Evaluación automatizada.

Automated generation of control design benchmark problems for computer-assessed education with Doctus.

Abstract

This paper presents an application that enables the teacher to automatically generate control design exercises for linear systems for the Doctus e-learning platform. Doctus is an application that automates the collection, storage and evaluation of scientific and technical education exercises. In general, the use of Doctus requires writing MATLAB scripts, both for the automatic evaluation of student responses and for the generation of personalized wordings. The application presented is an extension of Doctus designed to simplify this work, allowing the generation of personalized exercises by applying time and gain scalings based on a seed problem provided by the teacher. As a consequence, the application presented solves two problems: first, to facilitate the use of Doctus to non-expert teachers, and second, to provide each student with an individualized exercise with guaranteed solution and equal difficulty.

Keywords:
Control education, Educational aids 2, Teaching, PID Control, Linear control systems, Benchmark examples, Software tools, Automated assessment.

1. Introducción

En los últimos años, el ámbito académico de la comunidad de la Automática ha acometido numerosos proyectos destinados a dotar a los estudiates de herramientas que faciliten un aprendizaje activo con un grado creciente de interactividad. En este contexto, han abundado los trabajos sobre nuevas herramientas físicas de bajo coste (Valera et al., 2014), y plataformas online de experimentación basadas en laboratorios virtuales (Ayas and Altas (2016), Méndez et al. (2006)), que extien-

*Autor para correspondencia: cjsanchez@uloyola.es
To cite this article: C. Sánchez, D. Muñoz de la Peña, F. Gómez-Estern. 2020. Automated generation of control design benchmark problems for computer-assessed education with Doctus. Revista Iberoamericana de Automática e Informática Industrial 17, 1-9. https://doi.org/10.4995/riai.2019.11243
Attribution-NonCommercial-NoDerivatives 4,0 International (CC BY-NC-ND 4,0)

den el aprendizaje más allá de las horas presenciales de laboratorio. Este último grupo se apoya en tecnologías muy diversas, entre las que destacamos (Martínez et al., 2017), que emplea Labview para contribuir a la enseñanza del alumno con un laboratorio virtual, (Sánchez-Alonso et al., 2017), donde se incorpora el modelado físico de sólidos rígidos en realidad virtual para la formación en robótica, (Cerezo and Sastrón, 2015), que aborda la enseñanza de la Automática en educación secundaria con el apoyo de laboratorios virtuales, y Ruano Ruano et al. (2016), que enseña a a diseñar controladores de tipo PID mediante laboratorios virtuales integrados en sistemas de gestión del aprendizaje (LMS, módulos SCORM), incluyendo elementos de evaluación en línea.

Salvo escasos ejemplos, como el último, el caso de (Tartaglia and Tresso, 2002), y el enfoque innovador de (Méndez and González, 2013) y (Mendez and Gonzalez, 2011), el resto prescinde de mecanismos de evaluación, dejando al lector la tarea de desarrollarlos. A juicio de los autores, la evaluación automática debe ser un elemento esencial de cualquier plataforma de aprendzaje en línea, no solo por la necesidad del docente de calificar en un entorno de evaluación continua, en línea con las directrices del Espacio Europeo de Educación Superior, sino por la motivación que supone para los estudiantes a) saber que van a ser calificados (recompensados) por su esfuerzo, y b) el feedback que les proporciona toda evaluación temprana en el curso, automática o no. A esto cabe añadir que la evaluación automática está en la base de sistemas más avanzados capaces de adaptar los contenidos al grado de dominio del estudiante (Chih-Ming et al., 2005).

Sin duda alguna, la corrección automática ha recibido una gran atención en la literatura. La generación de enunciados de ejercicios, sin embargo, no ha sido tan estudiada. En general, en la disciplina de programación, la individualización de problemas se consigue mediante la asignación aleatoria de problemas diferentes de igual dificultad de entre un conjunto de problemas que sea al menos un orden de magnitud superior al número de alumnos a los que hay que asignar el problema. En (Ala-Mutka, 2005) se hace una revisión de diferentes metodologías para corregir problemas de programación (siendo el análisis de respuestas frente a conjuntos de datos de entrada la más extendida), pero no se menciona la generación de enunciados personalizados aunque sí se menciona el problema de plagio cuando los enunciados son comunes a los alumnos. El trabajo (Douce et al., 2005) tampoco trata el problema de la generación de problemas. Presenta igualmente una revisión de plataformas basadas en entrada salida y pone como desarrollo futuro un sistema de recogida, almacenaje y evaluación que pueda ser utilizado por Universidades, como lo es el sistema presentado en (Pieterse, 2005). En dicho trabajo se presenta una arquitectura para evaluar programas usando baterías de datos de entrada/salida, y aunque se menciona el problema del plagio, no proporciona ningún resultado sobre la generación automática de enunciados personalizados. (Prados et al., 2005) también presenta una plataforma basada en Mathematica y C++, con la particularidad de que la respuesta del alumno no se entrega en formato texto, sino a partir de un interfaz dedicado. La generación de enunciados personalizados se realiza mediante la elección de parámetros aleatorios.

En este trabajo proponemos dar un uso especial a la plataforma de evaluación automática Doctus. El sistema de e-learning Doctus es una herramienta consolidada y popular en el área de Ingeniería de Sistemas y Automática en España y el extranjero. Ha sido traducida al francés y al inglés, y actualmente ha sido empleado en más de 100.000 pruebas de evaluación, según reflejan sus registros. Inicialmente desarrollada en 2007 en la arquitectura PHP-MySQL-Apache por profesores del área, ofrecía un interfaz sencillo en el que se podía evaluar código compatible MATLAB escrito por los alumnos como solución a un problema técnico o científico planteado. El código MATLAB de cada alumno era anexado a un código suministrado por el profesor, capaz de verificar si la solución proporcionada por el estudiante era correcta, mediante la observación de las variables creadas en MATLAB (que se ejecutaba en una instancia de servidor) como consecuencia de la ejecución del código del alumno. El desarrollo de esta herramienta ha dado lugar a múltiples resultados (Gómez-Estern et al., 2010), (López-Martínez et al., 2010), (Muñoz de la Peña et al., 2012a), (Muñoz de la Peña et al., 2012b).

En general, el motor de evaluación de Doctus no es más que un continente, o *sandbox*, destinado a que los profesores desarrollen sus propias estrategias de evaluación. El docente ha de programar herramientas y algoritmos sofisticados para la evaluación y la generación de enunciados, que pueden incluir simulaciones, análisis de series de datos, verificación de fórmulas, etc. La dificultad de esta tarea se ve compensada por los esfuerzos orientados a la reutilización de ejercicios, de manera que tras años de uso se ha creado una importante librería de ejercicios autoevaluados en áreas como programación, control automático, ingeniería química o mecánica.

En este trabajo se presenta una extensión de Doctus que permite a un profesor generar de forma automática ejercicios de diseño de controladores para sistemas lineales para esta plataforma de e-learning. La nueva herramienta permite al docente crear ejercicios de control individualizados y evaluados automáticamente con solo seleccionar qué aspectos del curso de control han de ser evaluados (dentro de una lista de especificaciones). Este sistema genera automáticamente los enunciados y los evaluadores, ambos individualizados, y garantiza la existencia de solución del problema en todos los casos empleando la misma técnica de control y manteniendo la dificultad uniforme entre los alumnos. Además, se permite una evaluación en modo *ranking*, donde la puntuación máxima del ejercicio se asigna al alumno que haya diseñado el mejor controlador para el sistema. Es decir, se realiza una evaluación competitiva entre los alumnos.

En este artículo se describirá inicialmente la arquitectura de evaluación automática estándar de Doctus que subyace al sistema propuesto. A continuación se describirá la clase de problemas de control que se desea automatizar, se ilustrará el nuevo interfaz de usuario, y se discutirá la lógica de generación de enunciados y evaluadores bajo la cual se garantiza la equivalencia de los ejercicios. Finalmente, se narra una experiencia práctica llevada a cabo en el aula universitaria.

2. Evaluación y personalización automática con Doctus

Actualmente, Doctus se usa de forma habitual en numerosas asignaturas, especialmente en la Escuela Técnica Superior de Ingeniería de nuestra Universidad. La naturaleza abierta de la aplicación permite ser utilizada como parte de los sistemas de evaluación de diferentes modos. La idea básica, es que el profesor puede crear ejercicios que posteriormente asigna a los alumnos de una asignatura. En algunos casos se utiliza para evaluar el trabajo práctico de las sesiones de laboratorio, teniendo un gran peso en la nota del alumno, en otros se utiliza para evaluar ejercicios complejos que los alumnos realizan en casa. El profesor tiene el control del proceso de evaluación, pudiendo evaluar una única vez los trabajos asignados o incluso múltiples veces a lo largo del periodo de entrega para que los alumnos puedan corregir sus errores. Aunque Doctus permite trabajar con diferentes motores de evaluación como Excel o C++, en el contexto de evaluación de problemas de diseño de controladores, usaremos MATLAB.

Para usar Doctus, el profesor es el que debe generar los códigos de generación y evaluación, los cuales a partir de cierta información, como el DNI del alumno, y de la respuesta de alumno, se encargan de devolver cierta información como la nota, los comentarios o el enunciado personalizado. La Figura 1 muestra el esquema de evaluación de Doctus. La clave de este sistema es que las tareas del servidor están delimitadas a gestionar la información proporcionada por el alumno y por el profesor. Es el profesor el que dota de capacidad de evaluación con su código. Cuando el alumno entra en el sistema se le presenta una plantilla, elaborada previamente por el profesor, para que introduzca los valores correctos de la solución al problema. La solución que proporciona el alumno al problema es código MATLAB, por lo que se puede anexar al final una porción de código proporcionada por el profesor al diseñar el ejercicio, conocida como evaluador, para recuperar los valores creados tras la ejecución de dicho código.

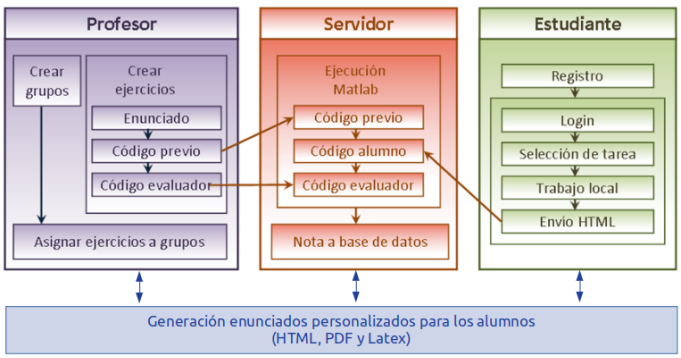


Figura 1: Esquema de evaluación de Doctus.

En la Figura 3 se muestra un breve código MATLAB evaluador que verifica que los resultados proporcionados por el alumno usando la plantilla mostrada en la Figura 2 son correctos. La ejecución de este código evaluador, diseñado previamente por el docente, tiene como objetivo generar una variable *nota*, que reflejará la puntuación obtenida por el alumno. La unión de esta lógica y un conjunto de sistemas informáticos (servidor web, bases de datos, sistema de almacenamiento de ficheros, comentarios, generación de informes, interfaces gráficas, etc.) constituyen el núcleo de la plataforma Doctus.

```
J=0; % Introduzca el momento de  inercia en Kg·m
Vfinal=0; % Introduzca la velocidad final en m/s
```

Figura 2: Formato de código entregado por el alumno como solución a un problema.

```
J_correcta = 23.5;
if abs(J - J_correcta) < 0.01
    nota = nota + 1;
end
```

Figura 3: Código MATLAB para evaluar la entrega del alumno.

2.1. *Generación de enunciados personalizados*

Durante el proceso de evaluación, Doctus crea en el espacio de trabajo de MATLAB una cadena de caracteres con el DNI del alumno que está siendo evaluado. Esto permite de una forma sencilla, personalizar los parámetros de los diferentes problemas en función de este dato. En el ejemplo presentado anteriormente, el cálculo de la velocidad final, suponiendo un móvil en caida libre, dependería de su velocidad inicial y el tiempo final. En el enunciado del problema, el profesor podría indicar que la velocidad inicial y el tiempo final se calculan como 10 más el primer y segundo dígito del DNI respectivamente. Así, cada alumno, obtiene unos parámetros diferentes. El código evaluador debe, por lo tanto, generar primero estos parámetros a partir de DNI y posteriormente calcular la solución correcta. La Figura 4 muestra el código evaluador.

```
V0 = 10 + str2num(dni(1));
Tfinal = 10 + str2num(dni(2));
V_final_correcta = V0+g*Tfinal;
if abs(V_final - V_final_correcta) < 0.01
    nota = nota + 1;
end
```

Figura 4: Código MATLAB para evaluar la entrega personalizada del alumno.

Este paradigma de personalización, sin embargo, tiene la desventaja de que el profesor debe proporcionar al alumno en el enunciado la relación explícita entre los parámetros del problema y su DNI. Esto dificulta las posibilidades de personalización, además de permitir posibles errores por parte del alumno al obtener los parámetros. Doctus proporciona diferentes herramientas para poder generar de forma automática enunciados personalizados usando un modelo de ejecución de código similar al modelo de evaluación. En particular, en esta sección vamos a presentar el modelo de generación de enunciado basado en ejecución de un código generado de MATLAB.

Doctus permite al profesor proporcionar un código en MATLAB que al ejecutarse construya, a partir de la cadena de caracteres con el DNI de un alumno, otra cadena de caracteres, denominada `htmlautowording` que se mostrará al alumno en formato HTML cuando acceda a la plataforma a entregar al

alumno. Cabe recordar que esta cadena se muestra conjuntamente con cualquier otro enunciado estático no personalizado proporcionado por el profesor en formato texto o PDF, por lo que puede limitarse a visualizar únicamente los parámetros del problema.

```
V0 = 10 + str2num(dni(1));
Tfinal = 10 + str2num(dni(2));
htmlautowording = 'Parámetros personalizados:<br>';
htmlautowording=strcat(htmlautowording,...
sprintf('V0=%d<br>',V0));
htmlautowording=strcat(htmlautowording,...
sprintf('Tfinal=%d<br>',Tfinal));
```

Figura 5: Código MATLAB para generar el enunciado personalizado.

Antes de publicar el ejercicio, el profesor da la orden a Doctus para generar los enunciados de todos los alumnos de un grupo. Esto evita posibles sobrecargas del servidor MATLAB al entrar múltiples alumnos simultáneamente en la aplicación. Para que el código de evaluación funcione correctamente, es importante que genere los parámetros de los problemas de forma consistente al enunciado.

Este paradigma de generación de enunciados mediante la ejecución de un código tiene varias ventajas sobre proporcionar la relación explícita de los parámetros con el DNI. Por ejemplo, permite la generación determinista de enunciados aleatorios mediante la generación de una secuencia de números aleatorios a partir de una semilla fija para cada alumno. Esto permite al profesor de una forma sencilla definir un rango de valores mínimos y máximos entre los que los parámetros de cada alumno se generaran, simplificando el proceso de diseño de ejercicios. En la Figura 6 se muestra un código que genera la velocidad inicial y el tiempo final entre 10 y 20 para cada alumno.

```
DNI = str2num(dni);
rand('seed',DNI);
V0 = 10 + 10*rand(1);
Tfinal = 10 + 10*rand(1);
htmlautowording = 'Parámetros personalizados:<br>';
htmlautowording=strcat(htmlautowording,...
sprintf('V0=%d<br>',V0));
htmlautowording=strcat(htmlautowording,...
sprintf('Tfinal=%d<br>',Tfinal));
```

Figura 6: Código MATLAB para generar el enunciado personalizado, con los parámetros del problema delimitados por intervalos.

En este caso, el evaluador también tiene que modificar las líneas de código que calculan los parámetros iniciales. La clave reside en que, aunque se invoca a una función para generar números aleatorios en dos instantes de tiempo diferentes (en la generación del enunciado y en el momento de evaluar), la secuencia para cada alumno es determinista por haber sido generada a partir de la misma semilla (definida en este caso por su DNI). Esta metodología de diseño es muy versátil, y es la que utiliza la aplicación presentada en este trabajo.

3. Problema básico de diseño de controladores

Este paradigma de definición de ejercicios individualizados ha sido de gran utilidad en materias como ingeniería mecánica, química o electrónica. En primer lugar, por la relación explícita entre parámetros y soluciones (lo cual no hay que vincular con una menor complejidad del ejercicio, ya que dicha relación explícita puede ocultarse detrás de un complejo análisis matemático). En segundo lugar porque en los ejercicios como el del ejemplo la dificultad no se ve afectada con la variación de los parámetros.

Sin embargo, en control automático no se cumplen esas premisas: los parámetros de entrada de los problemas clásicos (coeficientes de la planta a controlar y especificaciones de comportamiento) no tienen una vinculación directa (matemáticamente explícita) con la solución al problema, dada por las constantes de ajuste de controladores. Ni siquiera es de esperar que se cumplan las especificaciones mediante una coincidencia numérica, sino mediante una superación de las mismas, lo que admite la existencia de infinitas soluciones a un único ejercicio. Además, existe una dificultad adicional en la extensión de esta arquitectura a la disciplina control automático, y reside en la dificultad de generar enunciados individualizados. En efecto, si hacemos depender de un número identificativo único para cada alumno los coeficientes de la función de transferencia en bucle abierto, o las especificaciones, es probable que la dificultad o la propia existencia de solución al problema varíen, y por tanto no se esté garantizando la homogeneidad en la dificultad y procedimientos de resolución del problema.

En este trabajo nos centramos en el problema de diseño de controladores para sistemas lineales, propio de asignaturas de Fundamentos de Control (Dorf and Bishop, 2005), (Guzmán et al., 2012), (Ogata, 2011). La plataforma diseñada se centra en una clase de ejercicios muy concreta, el diseño de controladores para sistemas lineales descritos mediante función de transferencia. Este problema también es relevante para el control de sistemas no lineales en torno a un punto de operación de los que se dispone de un modelo linealizado del mismo en torno a ese punto. El objetivo es diseñar un controlador lineal descrito por la función de transferencia $C(s)$, de forma que el sistema en bucle cerrado cumpla usa serie de especificaciones, tanto en el dominio del tiempo como en el dominio de la frecuencia. En esta sección presentamos el procedimiento seguido para generar de forma automática el enunciado individualizado garantizando la existencia de solución y la homogeneidad en la dificultad y procedimientos de resolución de los mismos. Es El procedimiento de generación de enunciados no es extensible a sistemas/leyes de control no lineales.

Un ejercicio está definido por el modelo de los sistemas, las especificaciones, la estructura del controlador y opcionalmente el índice de desempeño objetivo. En general, la generación aleatoria de sistemas y/o especificaciones no puede garantizar ninguna propiedad respecto a la existencia de solución del problema, ni respecto a los procedimientos para obtenerla. Una variación acotada suficientemente pequeña de los parámetros de los sistemas y de las especificaciones puede dar como resultado problemas equivalentes, pero no ofrece ninguna garantía en la existencia de soluciones y además, los problemas pueden resultar demasiado parecidos.

El método propuesto consiste en definir problemas a partir de un problema base proporcionado por el profesor, de forma que sean equivalentes en dificultad y procedimiento de solución. Este procedimiento permite utilizar los ejercicios disponibles en los libros de texto de la materia. En la herramienta desarrollada, hemos seguido otro procedimiento basado en aplicar dos transformaciones que no afecten la naturaleza del problema base, pero sí sus parámetros, en particular, un escalado temporal y un escalado de la ganancia del sistema en bucle abierto. En particular, para generar el sistema de un ejercicio, se definen dos parámetros individualizados de escalado, A y τ. El sistema escalado se define de la siguiente forma:

$$G(s) = A \cdot G_b(\tau s) \tag{1}$$

siendo G(s) la función de transferencia generada a partir de la función de transferencia del problema base Gb(s).

Teniendo en cuenta las propiedades de la transformada de Laplace, esta transformación es equivalente a un cambio en la escala de tiempo de factor τ, es decir,

$$t = t_b/\tau, \tag{2}$$

siendo t la escala de tiempo del problema generado y t_b la escala de tiempo del problema base.

Un escalado temporal no modifica la naturaleza del problema de diseño, y el escalado de la ganancia del sistema en bucle abierto puede ser compensado por la ganancia del controlador. En particular, se puede demostrar que la ley de control

$$C(s) = C_b(\tau s)/A, \tag{3}$$

siendo $C_b(s)$ una ley de control solución del problema base, garantiza que el sistema en bucle cerrado resultante es igual al sistema base en bucle cerrado con el controlador base, pero con un cambio en la escala temporal. Esto implica que si las especificaciones han sido escaladas de forma apropiada, el controlador $C(s)$ cumple todas las especificaciones para la planta modificada.

En la aplicación se han considerado un conjunto concreto de especificaciones. En particular, en el domino del tiempo se han considerado especificaciones de sobreoscilación, tiempo de subida y tiempo de establecimiento frente a una referencia en escalón de amplitud unidad. También se ha considerado el valor del error en régimen permanente frente a referencias en escalón, rampa y parábolas de amplitud unidad. En el domino de la frecuencia se ha considerado límites en la frecuencia de corte, el margen de fase y el margen de ganancia del sistema compensado. Esta lista no es extensiva, y puede ser ampliada con cualquier tipo de especificación que pueda ser evaluada a partir del diagrama de bloques y de las funciones de transferencia implicadas, tanto de forma explícita como por simulación. La Tabla 1 muestra la operación escalada para cada una de las especificaciones consideradas.

En el campo del control resulta de particular interés la posibilidad de comparar diferentes controladores en función de algún índice de desempeño. Mediante la evaluación competitiva es posible usar tres índices, que además cumplan una serie de restricciones, para comparar las soluciones de los controladores. Los índices de desempeño disponibles son la integral del error al cuadrado, la integral del valor absoluto del error y el tiempo de subida frente a un escalón en la referencia. Con respecto a la evaluación de los índices de desempeño, es posible comparar los controladores diseñados para diferentes plantas, normalizando los índices con respecto a la escala de tiempo del problema base. Esto permite un alto grado de personalización, ya que cada alumno se enfrenta a un sistema diferente, pero a la vez está compitiendo con el resto de alumnos del curso. La Tabla 2 muestra los índices de desempeño normalizados.

Tabla 1: Escalado de especificaciones

Especificación	Base	Escalada
Sobreoscilación	SO	SO
Tiempo de subida	Ts	Ts·τ
Tiempo de establecimiento	Te	Te·τ
Frecuencia de corte	Wc	Wc/τ
Margen de fase	Mf	Mf
Margen de ganancia	Mg	Mg
Error en posición	Erp	Erp
Error en velocidad	Erv	Erv·τ
Error en aceleración	Era	Era·τ^2

Tabla 2: Índices de desempeño base normalizado

Especificación	Base	Escalada
Tiempo de subida	Ts	Ts·τ
Integral del error al cuadrado	ISE	ISE/τ
Integral del error absoluto del error	IAE	IAE/τ
Tiempo de integración (ISE e IAE)	Tf	Tf·τ

3.1. Ejemplo

Para generar un ejercicio personalizable se necesita partir de un problema base. En este caso, consideramos el problema de diseñar un controlador para el sistema

$$G_b(s) = \frac{2}{(s+0{,}1)(s+1)(s+10)},$$

de forma que el sistema en bucle cerrado presente, frente a una referencia en escalón, una respuesta subamortiguada con una sobreoscilación menor del 30 % y un tiempo de subida menor de 0.3s.

Este problema de diseño de controladores se puede resolver de múltiples formas (tiene infinitas soluciones). En particular, con un controlador proporcional derivativo con una ganancia proporcional de 60 y un tiempo derivativo de 0.3, dando como resultado la siguiente función de transferencia del controlador

$$C_b(s) = 18s + 60.$$

Se puede obtener un ejercicio que en esencia es de la misma naturaleza mediante los escalados propuestos. Escogiendo $A = 0{,}5$ y $\tau = 2$ se obtiene el ejercicio de diseñar un controlador para el sistema

$$G_p(s) = \frac{0{,}125}{(s+0{,}05)(s+0{,}5)(s+5)},$$

de forma que el sistema en bucle cerrado presente frente a una referencia en escalón una respuesta subamortiguada con una sobreoscilación menor del 30 % y un tiempo de subida menor de

0.6s. Obsérvese que cambian la función de transferencia y el tiempo de subida, pero que la sobreoscilación es invariante al escalado.

Este problema tiene solución garantizada, pues podemos observar que si aplicamos las transformaciones al controlador solución del problema base, con un tiempo derivativo de 0.6 y una ganancia de 72, obtenemos una solución válida que cumple las dos especificaciones:

$$C_p(s) = 72s + 120$$

En la Tabla 3 se muestran el valor de la sobreoscilación, el tiempo de subida y el tiempo de establecimiento de la respuesta frente a un escalón de los sistemas en bucle cerrado resultando en ambos casos junto con sus correpondientes frecuencia de corte y margen de fase de la cadena directa.

Tabla 3: Especificaciones del ejemplo

Especificación	Base	Escalada
Sobreoscilación	28.87	28.87
Tiempo de subida	0.2948	0.5897
Tiempo de establecimiento	1.1302	2.2604
Frecuencia de corte	4.1466	2.0733
Margen de fase	43.6235	43.6235

Las Figuras 7 y 8 muestran los diagramas de Bode y las respuestas al escalón unitario de los sistemas en bucle abierto sin compensar, para distintos valores de escalado A y τ, partiendo del problema base $G_b(s)$. En estas figuras se observa que los problemas son diferentes aparentemente, objetivo principal de la personalización.

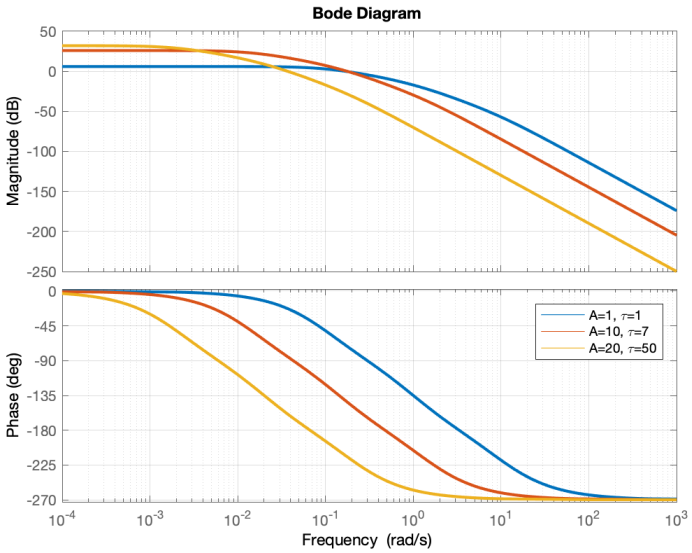


Figura 7: Diagramas de Bode para distintos ejercicios equivalentes, obtenidos con distintos valores de escalado en amplitud (A) y frecuencia (τ).

4. Arquitectura de la plataforma

La plataforma está formada por dos componentes. El primero es la interfaz gráfica mediante la cual se introducen los parámetros que definen el ejercicio. El segundo es el motor que recoge esta información y genera la plantilla de entrega del alumno, las variables iniciales y el código evaluador.

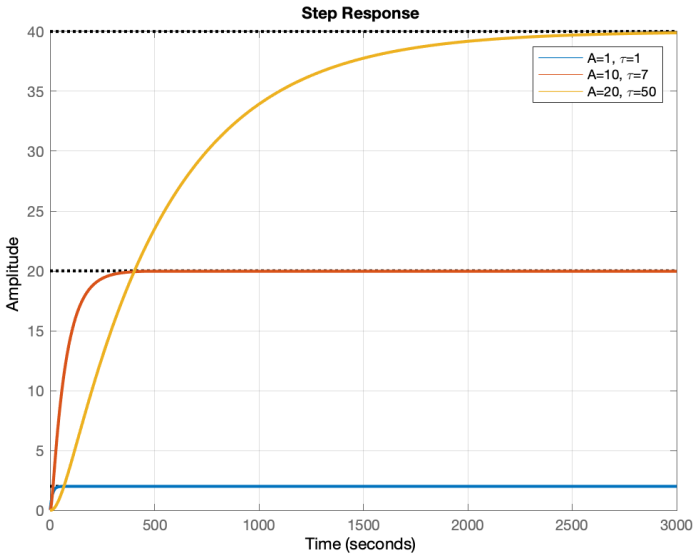


Figura 8: Gráficos de respuesta al escalón unitario obtenidos los mismos valores de escalado en amplitud y frecuencia de la Figura 8.

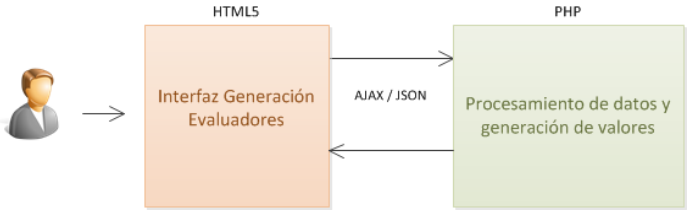


Figura 9: Arquitectura HTML/AJAX/PHP de la interfaz

La interfaz gráfica está dividida en dos zonas. La superior está formada por un sistema de navegación por pestañas que permite al profesor introducir toda la información necesaria para la generación del problema: sistema nominal a partir del cual se van a generar las plantas, especificaciones que debe cumplir el sistema en bucle cerrado, el tipo de controlador que debe diseñar el alumno, evaluación competitiva y exportar la práctica. En la Figura 10 se muestra esta interfaz gráfica.

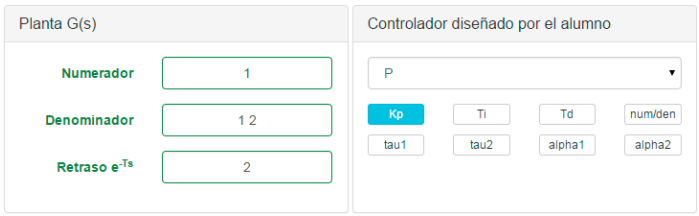
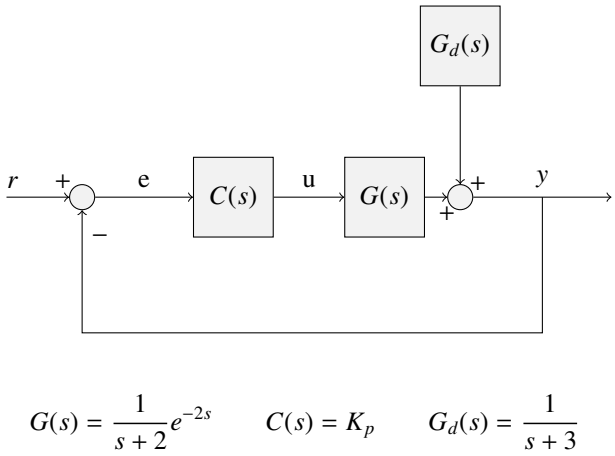


Figura 10: Definición de la planta G(s) y el controlador

En la parte inferior se muestran los resultados generados por la plataforma: representación gráfica del sistema base, estructura del controlador requerido y su plantilla de entrega, el código MATLAB que inicializa las variables y el código evaluador. La información generada por la plataforma se actualiza de forma automática cada vez que se modifica algún dato en

la interfaz. Esto se consigue a través del envío de peticiones asíncronas AJAX (acrónimo de Asynchronous JavaScript And XML) a un *script* PHP encargado de procesar la información y generar los datos. De esta forma, es posible realizar cambios sobre el diseño del evaluador sin necesidad de recargar la interfaz web, como ocurre en el tradicional modelo de formularios HTML a través de POST y GET, mejorando la interactividad, velocidad y usabilidad en la plataforma.

La zona superior de la plataforma está dividida en cinco menús: sistema base, generación de plantas, especificaciones, objetivo competitivo y exportar. En el menú *Sistema base*, mostrado en la Figura 11, el profesor define las funciones de transferencia del problema base introduciendo un vector de coeficientes para cada polinomio en s. Desde aquí puede describir la información de la planta G(s), la perturbación Gd(s) y el controlador que tiene que diseñar el alumno entre controladores P, PD, PI, PID, red de avance, red de retraso, red mixta y controlador libre.



$$G(s) = \frac{1}{s+2}e^{-2s} \qquad C(s) = K_p \qquad G_d(s) = \frac{1}{s+3}$$

Figura 11: Representación gráfica del sistema base seleccionado por el usuario

En el menú *Generación de plantas*, mostrado en la Figura 12, el profesor define los valores mínimos y máximos de los factores de escalado A y τ. Para cada alumno, el valor concreto de A y τ se genera de forma aleatoria a partir de su DNI.

Figura 12: Menú *Generación de plantas*

En el menú *Especificaciones*, el profesor selecciona de una tabla las especificaciones del problema base, indicando los valores máximo y mínimo de una determinada propiedad y el peso en la nota del ejercicio. La nota del alumno se calcula como la suma del peso de aquellas especificaciones que cumple su sistema en bucle cerrado.

En el menú *Objetivo competitivo* el profesor puede seleccionar una evaluación competitiva entre los alumnos. Esto se consigue a través del cálculo de una figura de mérito que caracteriza el rendimiento del controlador diseñado por cada alumno, estableciendo una clasificación de puntuaciones.

Entre los criterios disponibles se encuentra minimizar la integral del error al cuadrado (ISE), minimizar la integral del valor absoluto del error (IAE) y minimizar el tiempo de subida. El tiempo de integración determinado por t_f depende de la dinámica de la planta, y su valor se puede especificar en la herramienta o dejar libre para que el sistema lo calcule automáticamente. El cálculo del objetivo competitivo sólo se realiza cuando la nota obtenida por los alumnos que han diseñado un controlador que cumple todas las especificaciones está situada entre 5 y 10. Si alguna de las especificaciones no se cumple, la nota se establece en un valor -1.

Por último el menú *Exportar* permite generar de forma automática un archivo compatible con la aplicación Doctus. El profesor puede importar este archivo en la plataforma, creando un ejercicio con los parámetros definidos en la interfaz.

La plataforma permite añadir diferentes idiomas que se utilizan para la información mostrada en la interfaz gráfica y en la generación de los comentarios de MATLAB. Para añadir un idioma nuevo sólo hay que incluir un fichero de texto plano con las traducciones pertinentes. Cada vez que se accede a la plataforma web automáticamente se analizan los ficheros de idioma instalados y se muestra un menú en la zona superior de la interfaz para seleccionar el idioma.

Una vez proporcionada la información del problema base y de los límites de escalado, la aplicación devuelve, además de una representación gráfica del sistema, tres fragmentos de código MATLAB: la plantilla de solución, el código generador de parámetros y el código evaluador. Estos códigos MATLAB están diseñados para ser utilizados junto con la aplicación Doctus.

Cada vez que el *script* recibe una petición AJAX se inicia el diagrama de flujo reflejado en la Figura 13 y se realizan cuatro operaciones. La primera permite representar gráficamente los datos introducidos por el profesor para definir el sistema nominal, la estructura del controlador escogida y la perturbación Gd(s). El resultado de esta operación se muestra en la Figura 11.

El segundo bloque de operaciones se encarga de generar el enunciado personalizado del alumno. Para ello se analizan las especificaciones que ha seleccionado el profesor y se realiza su escalado. El tercer y cuarto bloque son los encargados de generar el sistema en bucle cerrado personalizado para el alumno y el objetivo competitivo. Toda esta información sirve como punto de entrada al bloque de construcción del evaluador, que es el encargado de simular el controlador diseñado por el alumno y comprobar si cumple las especificaciones indicadas.

Con el objetivo de aumentar el número de especificaciones disponibles en la plataforma (sobreoscilación, tiempo de subida, etc.), se ha desarrollado un mecanismo modular donde se agrupan los cálculos requeridos para cada especificación.

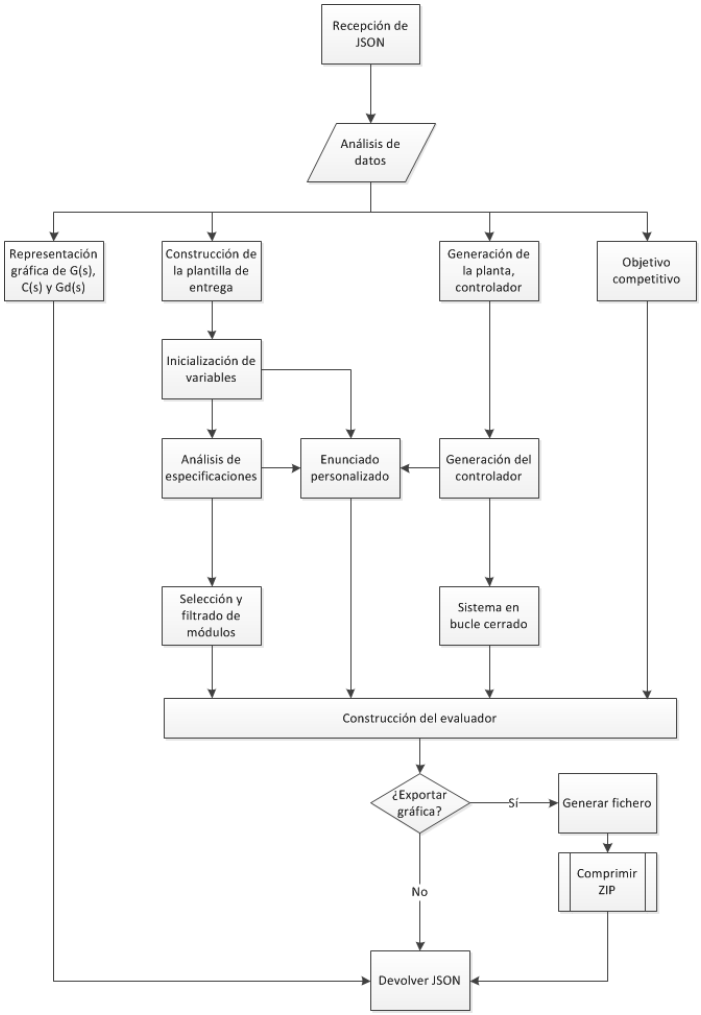


Figura 13: Diagrama de flujo para la generación de evaluadores

Los bloques *Análisis de especificaciones* y *Selección de filtrado de módulos* son los responsables de analizar qué operaciones son necesarias para la comprobación de cada especificación. Inicialmente se han definido tres módulos:

1. Parámetros subamortiguados.
2. Parámetros de frecuencia.
3. Error en régimen permanente.

El primero permite calcular los parámetros subamortiguados tales como tiempo de subida y tiempo de establecimiento. El segundo obtiene los parámetros de márgen de ganancia y frecuencia de corte en el dominio de la frecuencia. El último cálcula los diferentes errores en régimen permanente en función del tipo de sistema. En función de las especificaciones incluidas en el ejercicio por el profesor y del índice de desempeño, el código generado por la interfaz contendrá o no estos módulos.

5. Experiencias en el aula

La plataforma desarrollada ha sido aplicada con éxito en el curso Fundamentos de Control Automático de segundo del Grado en Ingeniería de las Tecnologías Industriales de la Universidad de Sevilla durante el curso 2017-18, con 453 estudiantes. En dicha asignatura se estudian por primera vez los sistemas dinámicos, las funciones de transferencia, las estructuras de realimentación y el análisis frecuencial. Este tipo de curso posee una importante carga conceptual, pero el reto más importante se presenta a la hora de lograr que los estudiantes resuelvan problemas por sí mismos y adquieran habilidades para adaptar los métodos a la gran diversidad de situaciones prácticas.

El uso de la plataforma de generación de enunciados durante el curso 2017-2018 no tenía como objetivo modificar la experiencia del alumno. Desde el año 2007 los alumnos han realizado prácticas y proyectos con enunciados personalizados como se indica en las referencias de los trabajos (Muñoz de la Peña et al., 2012b) y (Gómez-Estern et al., 2010). En cursos anteriores, la personalización de los problemas de diseño de controladores se realizaba por ensayo y error modificando los parámetros de los problemas de forma aleatoria entre unos valores máximos y mínimos. Este proceso en general necesitaba de mucho trabajo y tiempo por parte del profesor y además, en determinadas ocasiones, daba como resultado problemas que no tenían solución, lo cual podía perjudicar a determinados alumnos.

En general, para los profesores involucrados en la asignatura el resultado de la experiencia ha sido muy positivo. La posibilidad de generar de forma automática enunciados a partir de un ejercicio base permite reutilizar colecciones de ejercicios ya diseñados para el curso concreto, de forma que cada alumno los afronte de forma individual, permitiendo una renovación de los contenidos de la asignatura.

6. Conclusiones

En este trabajo se presenta una aplicación que permite a un profesor generar de forma automática ejercicios de diseño de controladores para sistemas lineales para la plataforma de e-learning Doctus.

La aplicación presentada genera scripts MATLAB de automatización tanto para la evaluación automática de la respuesta de los alumnos, como para la generación de enunciados personalizados.

El reto de generar problemas de forma aleatoria para que sean diferentes para un gran número de alumnos es una tarea difícil que, desde el conocimiento de los autores, no ha sido tratado con anterioridad en la literatura. En el proceso de generación automática de ejercicios se ha tenido especial cuidado con dos aspectos: el grado de diferenciación entre distintos ejercicios, que debe ser alto para que las soluciones no sean intercambiables entre estudiantes (en términos probabilísticos), y la homogeneidad en la dificultad de resolverlos. La solución propuesta logra diferenciar sensiblemente las respuestas temporales y frecuenciales de los sistemas en bucle abierto, al tiempo que mantiene invariable la dificultad de resolución.

Para ello, el algoritmo principal de la herramienta parte de un problema semilla, cuyo método y dificultad de resolución son conocidos, y realiza escalados temporal y de ganancia para cada estudiante con el fin de obtener ejercicios personalizados.

La herramienta ha sido probada extensivamente y depurada a través de la experiencia acumulada. Se ha evaluado la eficacia y los riesgos por parte de numerosos docentes del área. El riesgo más reseñable es el que provendría de una labor de ingeniería inversa por parte de los estudiantes, que revirtiera

el doble escalado e hiciera extrapolables las soluciones de un problema a otro. Se ha hecho un especial seguimiento de este riesgo, mediante consultas a los estudiantes y observando las posibles coincidencias numéricas entre las soluciones entregadas tras revertir es escalado, sin encontrar indicios de que esto se produzca. En cualquier caso, la comprensión de las propiedades matemáticas de los sistemas lineales, de sus respuestas y de sus especificaciones que haría falta para este ejercico de ingeniería inversa exceden considerablemente las competencias que se llegan a adquirir en los cursos introductorios de diseño de controladores.

La herramienta se puede emplear en distintos contextos: homework, trabajo en el aula colaborativo y competitivo, exámenes presenciales, benchmarks para concursos, congresos y actividades de CEA/IFAC, aprendizaje por prueba y error y actividades experimentales de análisis del aprendizaje (learning analytics).

En resumen, la aplicación presentada resuelve dos problemas, el primero facilitar el uso de Doctus a profesores, y el segundo proporcionar ejercicios personalizados con solución y dificultad garantizadas.

Agradecimientos

Los autores desean agradecer a Sebastián Dormido, Gonzalo Farias y Luis de la Torre por sus útiles comentarios y colaboraciones en el contexto del desarrollo de esta herramienta. Asimismo desean agradecer la ayuda otorgada por el Vicerrectorado de investigación de la Universidad Loyola Andalucía. Este trabajo se ha financiado parcialmente con los proyectos DPI2016-75294-C2-2-R y DPI2016-76493-C3-1-R.

Referencias

Ala-Mutka, K. M., 2005. A survey of automated assessment approaches for programming assignments. Computer Science Education 15 (2), 83–102.

Ayas, M. S., Altas, I. H., 2016. A virtual laboratory for system simulation and control with undergraduate curriculum. Computer Applications in Engineering Education 24 (1), 122–130.

Cerezo, F., Sastrón, F., 2015. Laboratorios virtuales y docencia de la automática en la formación tecnológica de base de alumnos preuniversitarios. Revista Iberoamericana de Automática e Informática industrial 12 (4), 419–431.

Chih-Ming, C., Hahn-Ming, L., Ya-Hui, C., 2005. Personalized e-learning system using item response theory. Computers & Education 44 (3), 237–255.

Dorf, R., Bishop, R., 2005. Sistemas de Control Moderno 10Ed. Pearson, Inglaterra.

Douce, C., Orwell, J., Livingstone, D., 2005. Automatic test-based assessment of programming: A review. ACM Journal of Educational Resources in Computing 5 (3).

Guzmán, J., Costa, R., Berenguel, M., Dormido, S., 2012. Control automático con herramientas interactivas. Pearson, Inglaterra.

Gómez-Estern, F., López-Martínez, M., Muñoz de la Peña, D., 2010. Sistemas de evaluación automática vía web en asignaturas prácticas de ingeniería. Revista Iberoamericana de Automática e Informática Industrial 7 (3), 111–119.

López-Martínez, M., Gómez-Estern, F., Muñoz de la Peña, D., 2010. Automatic web-based evaluation of C-programming exercises in engineering education. International Journal for Knowledge, Science and Technology 1 (2), 1–6.

Martínez, J., Padilla, A., Rodríguez, E., Jiménez, A., Orozco, H., 2017. Diseño de herramientas didácticas enfocadas al aprendizaje de sistemas de control utilizando instrumentación virtual. Revista Iberoamericana de Automática e Informática industrial 14 (4), 424–433.

Mendez, J. A., Gonzalez, E. J., 2011. Implementing motivational features in reactive blended learning: Application to an introductory control engineering course. IEEE Transactions on Education 54 (4), 619–627.

Méndez, J. A., González, E. J., 2013. A control system proposal for engineering education. Computers & Education 68, 266–274.

Méndez, J. A., Lorenzo, C., Acosta, L., Torres, S., González, E., 2006. A web-based tool for control engineering teaching. Computer Applications in Engineering Education 14 (3), 178–187.

Muñoz de la Peña, A., González-Gómez, D., Muñoz de la Peña, D., Gómez-Estern, F., Sánchez, M., 2012a. Automatic web-based grading system: Application in an advanced instrumental analysis chemistry laboratory. Journal of Chemical Education 90, 308–314.

Muñoz de la Peña, D., Gómez-Estern, F., Dormido, S., 2012b. A new internet tool for automatic evaluation in control, systems and programming. IEEE Computers & Education 59, 535–550.

Ogata, K., 2011. Ingeniería de Control Moderna 5ED. Pearson, Inglaterra.

Pieterse, V., 2005. Automated assessment of programming assignments. Computer Science Education Research Conference.

Prados, F., Boada, I., Soler, J., Poch, J., 2005. Automatic generation and correction of technical exercises. International Conference on Engineering and Computer Education.

Ruano Ruano, I., Gámez García, J., Gómez Ortega, J., 2016. Laboratorio web scorm de control pid con integración avanzada. Revista Iberoamericana de Automática e Informática industrial 13 (4), 472–483.

Sánchez-Alonso, R. E., Ortega-Moody, J., González-Barbosa, J. J., Reyes-Morales, G., 2017. Uso de plataformas para el desarrollo de aplicaciones virtuales en el modelado de robot manipuladores. Revista Iberoamericana de Automática e Informática industrial 14 (3), 279–287.

Tartaglia, A., Tresso, E., 2002. An automatic evaluation system for technical education at the university level. IEEE Transactions on Education 45 (3), 268–275.

Valera, A., Soriano, A., Vallés, M., 2014. Plataformas de bajo coste para la realización de trabajos prácticos de mecatrónica y robótica. Revista Iberoamericana de Automática e Informática industrial 11 (4), 363–376.