XLIV Jornadas de Automática 2023

https://jautomatica.es

Creación de ejercicios con corrección automática con MATLAB Grader para asignaturas de control automático

Tejado, I.*, Nuevo-Gallardo, C., Pérez, E.

Universidad de Extremadura, Escuela de Ingenierías Industriales, Avenida de Elvas s/n, 06006, Badajoz, España.

To cite this article: Tejado, I., Nuevo-Gallardo, C., Pérez, E. 2023. Creating auto-grading exercises with MATLAB Grader for automatic control courses.
XLIV Jornadas de Automática, 271-276. https://doi.org/10.17979/spudc.9788497498609.271

Resumen

Recientemente, el desarrollo de recursos interactivos está recibiendo gran atención en titulaciones de ingeniería para apoyar la enseñanza, especialmente tras la pandemia derivada por el coronavirus. Entre estos recursos, los que tienen la capacidad de corrección y/o calificación automática son los preferidos tanto por los estudiantes como el profesorado. Asimismo, la implementación de una evaluación formativa eficaz es uno de los principales retos de los docentes para mejorar realmente el proceso de enseñanza-aprendizaje. Este trabajo presenta un ejemplo de aplicación de MATLAB® Grader™, una herramienta de autor basada en web que permite a los instructores proporcionar una evaluación automática de diferentes tipos de problemas de ingeniería, no solo los clásicos ejercicios de programación, y, en consecuencia, informar a los estudiantes sobre su rendimiento de una manera rápida y eficaz. En concreto, se presentan una serie de problemas de diseño clásicos de control automático con fines ilustrativos.

Palabras clave: Educación en control, Herramientas software, Control automático, Diseño de sistemas, Controladores en lazo cerrado.

Creating auto-grading exercises with MATLAB Grader for automatic control courses

Abstract

Recently, the development of interactive resources is receiving great attention in engineering degrees to support teaching, especially after coronavirus pandemic. Among such resources, those that have the ability of automatic correction and/or grading are preferred for both students and faculty. Likewise, the implementation of an effective formative assessment is one of the main challenges for academy in order to actually improve the teaching-learning process. This paper presents an example of application of MATLAB® Grader™, a web-based authoring tool that allows instructors to provide automatic evaluation of different kinds of engineering problems, not only classic coding exercises, and, consequently, inform students about their performance in a quick and effective manner. Specifically, a series of classical design problems of automatic control are given for illustrative purposes.

Keywords: Control education, Software tools, Automatic control, System design, Closed-loop controllers.

1. Introducción

Ante el creciente interés por el uso de recursos interactivos como apoyo a la docencia en las titulaciones de ingeniería, uno de los grandes retos en el contexto educativo actual es el desarrollo de herramientas que incluyan evaluación automática, en un sentido amplio del término, es decir, incluyendo también la corrección. Esto se debe a dos razones principales: por un lado, poder proporcionar a los estudiantes realimentación formativa efectiva (en tiempo y forma) y, por otro, reducir el tiempo dedicado por los profesores de las asignaturas a corregir el gran número de ejercicios que se proponen a los estudiantes

*Autor para correspondencia: itejbal@unex.es

en el caso de evaluación continua. Esta situación es especialmente atractiva en asignaturas de control automático, ya que la enseñanza de esta materia necesita un equilibrio entre una profunda comprensión de las matemáticas que subyacen a los conceptos básicos y la experiencia práctica. Asimismo, muchos de los problemas clásicos de control tienen múltiples soluciones, por lo que su corrección puede resultar un proceso engorroso y laborioso para los profesores.

En sentido estricto, uno de los principales beneficios de los recursos antes mencionados es promover el aprendizaje autónomo de los estudiantes, permitiéndoles adquirir una mayor responsabilidad y un papel más activo en el proceso de enseñanza-aprendizaje. Teniendo en cuenta esto, parece evidente que la evaluación automática debería ser una característica inherente a este tipo de recursos. Por otra parte, la evaluación formativa, es decir, el seguimiento del aprendizaje de alumnado (comprensión, necesidades y progreso académico) para proporcionarle una realimentación continua y en el momento adecuado, desempeña un papel fundamental a la hora de ayudar a los estudiantes a identificar sus puntos fuertes y débiles, así como las cuestiones que necesitan trabajar y, en consecuencia, a mejorar su aprendizaje. Del mismo modo, puede ayudar al profesorado a reconocer las dificultades de los estudiantes y abordarlas de inmediato, así como a mejorar la enseñanza.

En la bibliografía especializada, es posible encontrar diversas herramientas que permiten crear recursos educativos interactivos con evaluación automática, como el elemento de evaluación automática presentado basado en el Goodle (GMS) *grading management system* en Farias et al. (2016), el laboratorio virtual de programación de del Pino (2023), Nbgrader (Land, 2019), Web-CAT (Edwards and Perez-Quinones, 2008), Doctus (Gómez-Estern and Muñoz de la Peña, 2010) y MATLAB® Grader™ (The MathWorks, Inc., 2023), entre otros. Aunque la mayoría de ellas están pensadas para comparar código para asignaturas en la que se lleven a cabo ejercicios de programación, las dos últimas son las que más versatilidad pueden ofrecer de cara a su uso para otro tipo de asignaturas, especialmente del área de Automática. De hecho, Doctus es una versión renovada de Goodle GMS que tiene ciertas similitudes con MATLAB® Grader™ (consúltese (Gómez-Estern et al., 2010) para más información).

En este contexto y con la motivación de poder implementar una evaluación formativa efectiva, este trabajo pretende explorar las posibilidades de la herramienta Grader™ de MATLAB®, un entorno de autor basado en navegador que permite desarrollar y compartir problemas de codificación MATLAB en cualquier entorno de aprendizaje con tareas interactivas y calificar automáticamente el trabajo de los estudiantes, en asignaturas de control automático. El objetivo planteado es doble. Por un lado, crear recursos interactivos que proporcionen, de manera automática, una realimentación formativa oportuna a los estudiantes, en tiempo y forma, de manera que les permita identificar posibles errores de comprensión y/o resolución con el fin último de mejorar su rendimiento académico en la asignatura. Y respecto al profesorado, la idea es poder reducir el tiempo empleado en la corrección y calificación de tareas. Con fines ilustrativos, se presenta un ejercicio de diseño de controladores para la asignatura "Regulación Automática", asignatura obligatoria del tercer curso del Grado en Ingeniería Eléctrica que se imparte en la Universidad de Extremadura.

El resto del documento está estructurado como sigue. El apartado 2 ofrece una visión general de los principales aspectos involucrados en el uso de MATLAB® Grader™ para desarrollar recursos educativos con calificación automática. El apartado 3 describe brevemente el contexto educativo en el que se enmarca la asignatura "Regulación Automática". El apartado 4 presenta el ejemplo de aplicación de MATLAB® Grader™ para la asignatura "Regulación Automática". Finalmente, el apartado 5 resume las principales conclusiones de este trabajo.

2. MATLAB® Grader™

Este apartado contiene la información principal que se debe conocer de MATLAB® Grader™ para el desarrollo de recursos educativos.

2.1. Visión general

Muchos profesores de disciplinas de Ciencias e Ingeniería recurren a MATLAB® para enseñar conceptos asociados a sus materias e inculcar el pensamiento cuantitativo a sus alumnos. Para realizar este tipo de simulaciones es necesario contar con un sólido conjunto de habilidades de programación, que, en el caso de muchas ingenierías, se adquieren desde el primer año de carrera. Esta es la base, en diferentes campos, para comprobar soluciones de ecuaciones, integraciones, derivaciones, etc., optimizar resultados en problemas de diseño que implican múltiples parámetros variables, realizar gráficas para comparar resultados tanto de experimentos como de simulaciones y extraer conclusiones, entre otros.

MATLAB® Grader™ es una herramienta de autor basada en web que permite crear y compartir problemas de programación en MATLAB con calificación automática, es decir, proporcionando una realimentación instantánea a los estudiantes, en cualquier entorno de aprendizaje (consúltese The MathWorks, Inc. (2023) para todos los detalles).

Los recursos interactivos desarrollados con Grader™ permiten poner a prueba la asimilación de contenidos y comprensión por parte de los estudiantes, así como contribuir a que aprendan de sus errores y revisen su solución y, en consecuencia, mejoren su aprendizaje mediante la realimentación automatizada en tiempo real a partir de sus soluciones. De hecho, los estudiantes pueden realizar múltiples, e incluso ilimitadas, propuestas de resolución de un problema. Esto puede ser beneficioso para animar a los estudiantes a aprender de sus errores e intentar mejorar sus conocimientos y, por tanto, sus soluciones. Asimismo, este tipo de recursos es útil para los docentes porque pueden obtener información sobre el proceso de enseñanza-aprendizaje también de manera automática. En concreto, es posible crear gráficos con el número de intentos, el tiempo empleado y la diferencia entre la solución dada por los alumnos y la correcta, que pueden ser utilizados como información sobre el grado de comprensión de cada uno de los estudiantes y su evolución, así como para conocer qué conceptos han sido los más difíciles de entender por parte de ellos y controlar su progreso.

2.2. *Creación de recursos*

La creación de un recurso interactivo con MATLAB® Grader™ implica principalmente los siguientes pasos (véase la ventana principal en la Figura 1):

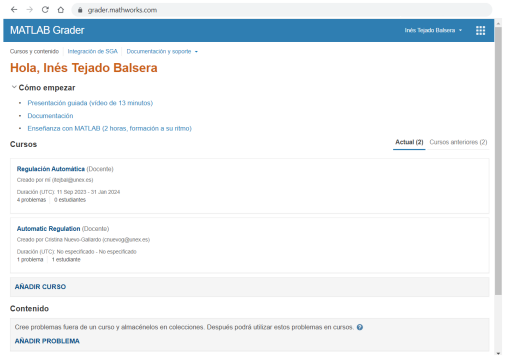

Figura 1: Ventana principal de MATLAB® Grader™.

1. Crear un curso, haciendo clic en el botón 'AÑADIR CURSO' en la sección 'Cursos', para proporcionar el título y la descripción del recurso y establecer la fecha de inicio, la de finalización y la zona horaria. Si fuera necesario, en este momento habría que seleccionar alguna herramienta o función particular de MATLAB necesaria para resolver los problemas del curso (en una pestaña desplegable denominada 'Productos').
2. Crear una tarea, haciendo clic en el botón 'AÑADIR TAREA' del menú de la izquierda e indicando título y descripción. Distingue entre dos tipos de tareas según el número de intentos (o envíos) permitidos a los estudiantes: ilimitado o limitado. También es posible restringir la disponibilidad de la tarea fijando una fecha y hora para hacerla visible y la fecha límite para la resolución.
3. Crear un problema, pulsando el botón 'AÑADIR PROBLEMA' (en el menú de la izquierda o debajo de la descripción de la tarea), con su título y descripción entre las siguientes opciones: un problema en blanco (opción con el mismo nombre), uno de una colección de ejemplos (opción *Getting Started with MATLAB Grader*) o un problema creado previamente (opción *Cursos y colecciones de MATLAB Grader*), siempre y cuando el usuario haya creado contenido fuera de un curso o hayan compartido contenido con él. Además, hay que seleccionar la forma de la solución del problema, es decir, como script o como función.
4. Establecer una solución de referencia (correcta) para el problema en forma de código de MATLAB (en la pestaña 'Solución de referencia'), invisible para los estudiantes, que permitirá compararla con la solución enviada por el estudiante. A ese respecto, se recomienda proporcionar a los estudiantes una plantilla de solución (en la pestaña 'Plantilla de estudiante'), también en forma de código, para que preparen la solución en la forma indicada. En ella, solo se mostrarán las líneas de código que no estén bloqueadas (aparecen con un candado en tono gris, desactivado); para bloquear líneas, únicamente habría que hacer clic sobre su candado, cambiando el color a negro.
5. Crear la evaluación para ese problema, haciendo clic en 'Añadir evaluación', que tiene dos cuestiones asociadas. Por un lado, el método de evaluación, entre *Correcto/Incorrecto* o *Ponderada*, a través de un menú desplegable. Y, por otro, el tipo de prueba, a elegir entre cuatro opciones de otro menú desplegable: *Variable igual a la solución de referencia*, *Consta una función o palabra clave*, *Falta la función o palabra clave*, o *Código MATLAB*). La descripción de cada una de ellas se encuentra en la Figura 2. Dependiendo de la opción elegida, el profesor deberá rellenar un campo obligatorio, así como los comentarios de correción que quiera que se le muestren al estudiante. Es importante resaltar que es posible: 1) añadir tantas evaluaciones como se desee por cada problema, es decir, permite poner el foco en varias de las cuestiones a considerar en un problema (se podrá elegir un tipo de prueba en cada caso); y 2) proporcionar a los estudiantes una descripción detallada del primer error que aparezca en su solución (mediante la opción *Mostrar comentarios solo para el error inicial*), así como permitirles realizar una primera prueba de su solución antes del envío (opción *Prueba previa*).
6. Validar la solución de referencia antes de publicar el problema, pulsando sobre el botón del mismo nombre, para asegurarse de que el código preparado por el instructor es correcto y no hay errores. Una vez validado, el problema será visible para los estudiantes durante la fecha que haya sido establecida.

Tipo de prueba	Descripción
La variable es igual a la solución de referencia	Comprueba si una variable en la solución de su estudiante es igual a la misma variable en la solución de referencia (dentro de la tolerancia). También comprueba si hay una variable, el tamaño y el tipo de datos.
Consta una función o palabra clave	Comprueba la presencia de funciones o palabras clave específicas en la solución de su estudiante.
Falta la función o palabra clave	Comprueba si determinadas funciones o palabras clave no están presentes en la solución de su estudiante.
Código de MATLAB	Escriba sus evaluaciones utilizando código de MATLAB. Para ver ejemplos de código, haga clic en ? situado junto a **Código de MATLAB**.

Figura 2: Tipos de prueba que pueden seleccionarse para definir la evaluación de una tarea en Grader.

2.3. *Otras cuestiones importantes*

Aparte de lo anterior, también es importante tener en cuenta las siguientes cuestiones a la hora de utilizar Grader™:

- La inscripción de los participantes debe realizarse una vez creado el curso, pulsando el botón 'Gestionar personas' (en el menú de la izquierda), utilizando sus correos electrónicos, separados por comas, seleccionando la opción *Estudiante* para el caso de los estudiantes y *Docente* para otros profesores o colaboradores (estos últimos tendrán permiso de edición). Los participantes registrados recibirán una notificación por correo electrónico con el enlace para acceder al curso. Para ello, conviene señalar que todos los participantes deben disponer de cuenta MathWorks.

- El acceso al curso como estudiante permite no solo resolver los problemas, sino también visualizar los resultados. Para ejecutar el código de la solución, el alumno debe hacer clic en el botón 'Ejecutar script', que podrá probar,

incluso varias veces, antes del envío final haciendo clic en 'Ejecutar prueba previa' (si está disponible). Una vez enviada la solución (mediante el botón 'Enviar'), le aparecerán los comentarios de realimentación escritos por el profesor para cada problema. En el caso de que se permitan múltiples envíos, los estudiantes podrán mejorar su solución seleccionando el botón 'Corregir esta solución' hasta que se alcance el límite configurado de envíos o hasta que se encuentre la solución correcta. Después de enviar la solución de un problema, se puede visualizar el progreso general desde la página principal de una tarea. En particular, mediante la opción *Ver mis soluciones*, los estudiantes pueden comprobar sus soluciones enviadas en diferentes formatos, así como los comentarios de realimentación recibidos.

- El acceso al curso como instructor permite supervisar el progreso de los estudiantes, también desde la página principal de una tarea y con diferentes formatos, como el progreso global de los estudiantes para cada problema incluido en la tarea (opción *Estado de la tarea* o mediante *Informe* en el menú 'Acciones'), así como un resumen de la clase mediante diferentes tipos de gráficos (opción *Visión general de la clase*) y estadísticas de los resultados (opción *Soluciones de estudiantes*), como se muestra en la Figura 3.
- El curso puede integrarse en cualquier sistema de gestión del aprendizaje siempre que la institución disponga de al menos una de las siguientes licencias: *Licencia de Enseñanza Académica*, *Licencia Campus* o *Licencia de Comunidad y Colegio Técnico*. En caso contrario, puede utilizarse el enlace generado al registrar a los participantes. En el caso de Moodle, además se requiere incluir la librería denominada 'MathJax' para soportar MATLAB® Grader™ y mostrar fórmulas y notaciones matemáticas en navegadores web.

Figura 3: Análisis de estudiantes en Grader: opciones de visualización para el docente del nivel de progreso respecto a un problema.

3. Contexto educativo

Este apartado describe brevemente el contexto educativo en el que se enmarca la asignatura "Regulación Automática".

3.1. Descripción general

La asignatura "Regulación Automática" es una materia obligatoria del primer semestre del tercer curso del Grado en Ingeniería Eléctrica (Rama Industrial) que se imparte en la Universidad de Extremadura. Más concretamente, la asignatura pertenece al módulo de *Tecnología Específica Electricidad* de esta titulación, y a la materia *Electrónica de Potencia y Automatización Industrial*. Tiene una carga lectiva total de 150 horas por alumno, distribuidas de la siguiente forma: 30 horas de clases teóricas (incluyen también sesiones de resolución de problemas y exámenes), 22,5 horas de clases prácticas (programadas en sesiones de 2 horas), 3 horas de tutorías programadas para el seguimiento del trabajo de los estudiantes, y las 94,5 horas restantes corresponden al trabajo no presencial del alumno.

El objetivo principal de esta asignatura es que los estudiantes se familiaricen con los conceptos básicos de teoría de control y regulación y el diseño de reguladores monovariables para abordar el control de sistemas eléctricos. El contenido se organiza en cinco temas: 1) Introducción al control realimentado, 2) Comportamiento dinámico de sistemas lineales e invariantes en el tiempo, 3) Control proporcional-integral-derivativo (PID), 4) Diseño de compensadores, y 5) Consideraciones prácticas en sistemas de control.

3.2. Observaciones sobre el contenido

En las asignaturas de teoría de control, los problemas clásicos tratan de controlar un sistema (o planta) en lazo cerrado para que su salida siga una referencia deseada, que puede ser fija o variable en el tiempo, cumpliendo un conjunto de requisitos de control deseados o especificaciones de diseño. Para ello, suele ser necesario un esquema de control con un controlador (también conocido como regulador o compensador), normalmente en lazo cerrado, capaz de medir la salida del sistema y compararla con la referencia. La diferencia entre la salida real y la deseada, conocida como señal de error, se aplica a la entrada del sistema a través del controlador, para hacer que el error sea igual a cero, o lo que es lo mismo, que la salida sea igual a la referencia. Por su parte, en este curso las especificaciones de diseño pueden referirse a la respuesta transitoria del sistema, ya sea en forma de características de dicha respuesta (como sobreimpulso, tiempo de establecimiento y tiempo de pico) o como polos dominantes de lazo cerrado deseados (en adelante, PDD), así como al error de estado estacionario de la respuesta del sistema, que se refiere a la precisión de la respuesta.

Los ejercicios de control mencionados suelen ser problemas de solución múltiple, ya que las especificaciones de diseño que deben alcanzarse se establecen normalmente como desigualdades. Esto significa que se pueden diseñar diferentes controladores, e incluso estructuras de control, para resolver el problema cumpliendo las especificaciones dadas. Debido a este hecho, la corrección de este tipo de ejercicios se convierte en un reto para los profesores del curso con el fin de reducir al máximo el tiempo invertido en su corrección, pero tratando de llevar a cabo una evaluación formativa eficaz para mejorar el aprendizaje de los alumnos en las primeras etapas de la enseñanza. Estas circunstancias hacen que los cursos de teoría de control sean firmes candidatos para aplicar con éxito los recursos interactivos creados con MATLAB® Grader™ que permiten a los alumnos no sólo saber si sus soluciones a los problemas son correctas, sino también obtener comentarios de realimentación de forma automática al enviarlas. Sin embargo, a diferencia de las asignaturas de informática, los problemas de programación tradicionales no son útiles para este caso, aunque los docentes enseñan muy a menudo control con MATLAB y su paquete

Simulink. Más concretamente, *Control System Toolbox* proporciona la mayoría de las herramientas necesarias para abordar los problemas de control clásicos. En consecuencia, la aplicación eficaz de Grader™ en los cursos de control exige un uso muy distinto de la mera comparación de códigos.

4. Ejemplo de recurso creado con Grader™

A continuación, se presenta un ejercicio de diseño de controladores PID a modo de ejemplo de aplicación de Grader™ para el Tema 3 de la asignatura "Regulación Automática". El recurso creado, denominado 'Relación de ejercicios de ejemplo', requiere para la resolución del problema la herramienta *Control System Toolbox* de MATLAB; como duración del recurso se ha elegido el primer semestre del próximo curso académico. Conviene señalar que, aunque el ejercicio se resolverá en este documento aplicando un método de diseño determinado, esta circunstancia no afecta en nada a la creación del recurso con MATLAB® Grader™, tal y como se ha ideado el recurso, por lo que este podría servir para otros métodos.

4.1. Enunciado

El enunciado del ejercicio es el siguiente: *Diseñar un controlador PID adecuado (P, PD, PI o PID) para el sistema:*

$$G(s) = \frac{1}{s(s+1)} \tag{1}$$

para que el sistema en lazo cerrado responda con error en estado estacionario para una entrada escalón $e_{ss} < 0{,}1$*, sobreimpulso* $M_p \leq 18\,\%$ *y tiempo de establecimiento* $t_s < 4$ *s.*

4.2. Solución de referencia

La primera cuestión que debe tener en cuenta el estudiante es que el sistema a controlar es de tipo 1 (tiene un polo en el origen) y, por tanto, va a responder con error cero en estado estacionario para una entrada escalón (es decir, $e_{ss} = 0$). De aquí se derivan dos cosas: la especificación de error se cumple y, por consiguiente, el término integral del controlador PID no será necesario. Por otro lado, considerando el sistema de segundo orden patrón como referencia para el sistema en lazo cerrado, los PDD vendrán dados como: $PDD = -\delta\omega_n \pm j\omega_n\sqrt{1-\delta^2}$, donde δ es el coeficiente de amortiguamiento y ω_n, la frecuencia natural del sistema que se obtendrán, en este caso, a partir de las especificaciones de diseño (M_p y t_s, en este caso).

Teniendo en cuenta lo anterior, para este ejercicio será necesario un controlador de dos parámetros para poder cumplir las dos especificaciones del enunciado, es decir, un controlador proporcional-derivativo (PD) que, en forma paralelo, vendrá dado por la siguiente función de transferencia:

$$C(s) = K_p + K_d s \tag{2}$$

donde K_p y K_d son las ganancias proporcional y derivativa, respectivamente.

Obviamente, existe una gran variedad de métodos para el diseño de este controlador. La solución que aquí se propone se obtendrá aplicando el método de localización de polos que se explica en el Tema 3 de la asignatura, que consiste en igualar la ecuación característica con la dinámica deseada de segundo orden ($p_d(s) = (s - PDD_1)(s - PDD_2)$ en el caso de conocer los PDD, o bien $p_d(s) = s^2 + 2\delta\omega_n s + \omega_n^2$, donde δ y ω_n se obtendrían a partir de las especificaciones), es decir:

$$1 + (K_p + K_d s)\frac{1}{s(s+1)} = s^2 + 2\delta\omega_n s + \omega_n^2 \tag{3}$$

Desarrollando e igualando los términos en s por un lado y los independientes por otro, se obtiene que el controlador PD vendrá dado por: $K_d = 2\delta\omega_n - 1$ y $K_p = \omega_n^2$. De lo que se trata ahora es de elegir unos valores de M_p y t_s por debajo del límite, determinar δ y ω_n y calcular los parámetros del controlador. Asimismo, será conveniente determinar la respuesta en lazo cerrado del sistema controlado y verificar que se cumplen las especificaciones de diseño. Por ejemplo, eligiendo $M_p = 8\,\%$ y $t_s = 3{,}8$ s, se obtiene $K_p = 2{,}822$ y $K_d = 1{,}105$, que hace que las características de la respuesta en lazo cerrado sean (véase la Figura 4): $M_p = 11{,}1\,\%$, $t_s = 3{,}13$ s y $e_{ss} = 0$. Como puede comprobarse, las especificaciones del enunciado se cumplen.

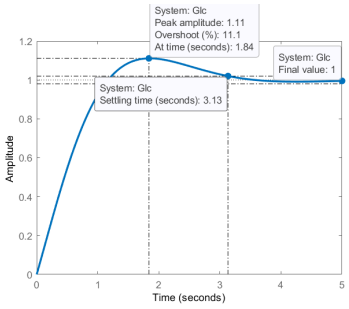

Figura 4: Respuesta al escalón del sistema controlado (lazo cerrado).

4.3. Creación con Grader™

Se configuran las siguientes opciones para el problema: problema tipo script, método de evaluación correcto/incorrecto y número ilimitado de intentos. La solución de referencia y la plantilla del alumno para resolver el problema se muestran en la Figura 5. Como puede observarse, los estudiantes tienen que introducir dos datos (variables `n` y `C`) para validar su solución:

- `n`: número del 1 al 4 que identifica el tipo de controlador PID que ha sido diseñado (1=proporcional, 2=proporcional-derivativo, 3=proporcional-integral y 4=proporcional-derivativo-integral).
- `C`: función de transferencia del controlador diseñado.

Teniendo en cuenta el diseño explicado en el apartado anterior, la solución de referencia para este ejercicio es: `n=2` (controlador PD) y `C=2.822+1.105*s)` (función de transferencia (2)).

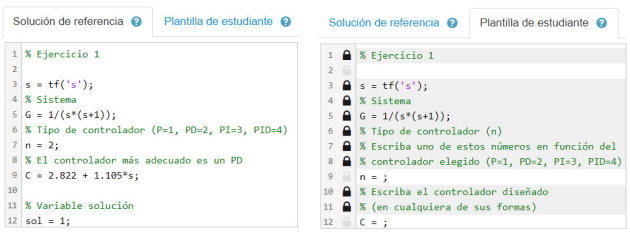

Figura 5: Código de la solución de referencia (izquierda) y la plantilla de estudiante (derecha).

A partir de estas dos variables, se realizan cuatro comprobaciones (evaluaciones), en las que las dos primeras se refieren al controlador, mientras que las otras a cada una de las especificaciones de diseño, es decir: 1) ¿es el tipo de controlador elegido el más adecuado?; 2) ¿la función de transferencia del controlador diseñado se corresponde con el tipo elegido?; 3) ¿se cumple la especificación del sobreimpulso?; y 4) ¿se cumple la especificación del tiempo de establecimiento? En concreto, el tipo de prueba para la primera comprobación se configura como *Variable igual a solución de referencia* para la variable `n`, y como *Código MATLAB* las tres restantes, que se basa en el comando `assessVariableEqual(var,refsol)` (donde `var` es la solución del estudiante a comparar con la solución de referencia `refsol`). Para cada una de estas comprobaciones se establece un comentario de realimentación para el estudiante en caso de fallo, dándole pistas sobre la posible procedencia.

En relación con los códigos desarrollados para las tres últimas pruebas, conviene hacer las siguientes aclaraciones. En la comprobación 2 (Figura 6(a)), la función de transferencia dada por el estudiante en la variable `C` se compara con la correspondiente a cada tipo de controlador mediante el comando `case()` de MATLAB y dependiendo del valor de `n`, proporcionando diferentes comentarios de realimentación para cada caso. Como puede observarse en las Figuras 6(b) y 6(c), el código desarrollado para las comprobaciones 3 y 4 se basan en obtener, primero, la función de transferencia del sistema en lazo cerrado (mediante el comando `feedback`), la respuesta al escalón (comando `step`) y, a partir de ella, determinar las características de respuesta transitoria (comando `stepinfo`). En cada caso se comprueba si se cumple la especificación de diseño del enunciado.

```
switch n
    case 1
        C_type = double(isa(C,'double'))
        assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo P.')
    case 2
        if double(isa(C,'double')) == 0
            C_type = double(C.den{1}(1)==0 && C.den{1}(2)==1 && length(C.num{1})==2);
            assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo PD.')
        else
            C_type = 0;
            assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo PD.')
        end
    case 3
        if double(isa(C,'double')) == 0
            C_type = double(~C.den{1}(1)==0 && C.den{1}(2)==0 && length(C.num{1})==2);
            assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo PI.')
        else
            C_type = 0;
            assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo PI.')
        end
    case 4
        if double(isa(C,'double')) == 0
            C_type = double(C.den{1}(1)==0 && ~C.den{1}(2)==0 && length(C.num{1})==3);
            assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo PID.')
        else
            C_type = 0;
            assessVariableEqual('C_type',referenceVariables.sol,'Feedback','El controlador diseñado no es de tipo PID.')
        end
end
```

(a)

```
Glc = feedback(C*G,1);          % FdT en lazo cerrado
[y,t] = step(Glc);              % Respuesta al escalón
[y,t] = step(Glc,t(end)*1.5);% Respuesta al escalón (ampliación Tfinal)
info = stepinfo(y,t,y(end)); % Características de la respuesta

Mp_alumno = double(info.Overshoot <= 18);
assessVariableEqual('Mp_alumno',referenceVariables.sol)
```

(b)

```
Glc = feedback(C*G,1);
[y,t] = step(Glc);              % Respuesta al escalón
[y,t] = step(Glc,t(end)*1.5);% Respuesta al escalón (ampliación Tfinal)
info = stepinfo(y,t,y(end)); % Características de la respuesta

ts_alumno = double(info.SettlingTime < 4);
assessVariableEqual('ts_alumno',referenceVariables.sol)
```

(c)

Figura 6: Códigos de MATLAB para las comprobaciones del ejercicio: (a) comprobación 2 (b) comprobación 3 (c) comprobación 4.

5. Conclusiones

Este trabajo ha mostrado el posible potencial de la herramienta Grader™ de MATLAB® para el desarrollo de recursos con corrección automática para asignaturas de regulación automática. Más concretamente, se ha demostrado que esta herramienta puede ser muy útil para problemas de diseño de controladores o compensadores con soluciones múltiples por dos motivos principalmente. Por un lado, poder proporcionar una realimentación eficaz (en tiempo y forma, ya que se produce de manera automática una vez el estudiante hace el envío de su solución) y, por otro, reducir el tiempo empleado por los docentes en la corrección. Por tanto, Grader™ puede ser una herramienta interesante para favorecer la evaluación formativa.

Aunque se han presentado un par de ejemplos de problemas de diseño, otros muchos también podrían ser implementados. Por ejemplo, con especificaciones de precisión en estado estacionario o incluso ejercicios de tipo verdadero o falso y exámenes. Evidentemente, también pueden añadirse problemas con solución única, eligiendo la opción *Variable igual a la solución de referencia*, que ya viene configurada con cierta tolerancia. La herramienta también ofrece la posibilidad de personalizar los ejercicios para cada estudiante, que puede hacerse introduciendo variabilidad a partir de un indentificador personal del mismo, o incluso mediante datos del problema que se generen de manera aleatoria.

Asimismo, otro de los beneficios que ofrecen los recursos creados con Grader™ tiene que ver con la visión que da, tanto a estudiantes como a profesores, respecto al rendimiento académico, que puede ser monitorizado de diversas maneras. Quizás el inconveniente principal sea el no ser una herramienta libre; requiere tener cuenta MathWorks y ciertas licencias para su integración en sistemas de aprendizaje (aunque podría hacerse únicamente mediante enlace).

En trabajos futuros, se estudiará la posibilidad de comparar las herramientas Grader™ y Doctus en la elaboración de este tipo de recursos para asignaturas de control automático, tanto desde el punto de vista del estudiante como del docente.

Agradecimientos

Los autores agradecen a Rocío de la Encarnación Sama su Trabajo Fin de Grado titulado “Evaluación automática con la herramienta Grader de MATLAB: Manual de uso y aplicaciones en ingeniería”, defendido en la Escuela de Ingenierías Industriales, Universidad de Extremadura, en julio de 2022 y que ha servido de base para el desarrollo de los ejemplos aquí presentados.

Referencias

del Pino, J. C. R., 2023. The Virtual Programming Lab for Moodle. VPL, https://vpl.dis.ulpgc.es/.

Edwards, S. H., Perez-Quinones, M. A., 2008. Web-CAT: automatically grading programming assignments. ACM SIGCSE Bulletin 40 (3), 328.

Farias, G., Muñoz de la Peña, D., Gómez-Estern, F., de la Torre, L., Sánchez, C., Dormido, S., 2016. Adding automatic evaluation to interactive virtual labs. Interactive Learning Environments 24 (7), 1456–1476.

Gómez-Estern, F., López-Martínez, M., Muñoz de la Peña, D., 2010. Sistema de evaluación automática vía web en asignaturas prácticas de ingeniería. Revista Iberoamericana de Automática e Informática Industrial 7 (3), 111–1119.

Gómez-Estern, F., Muñoz de la Peña, D., 2010. Zona de descargas de Doctus. Universidad de Sevilla, http://doctus.us.es/descargas/.

Land, D., 2019. Automatic grading in engineering classes. In: Proceedings of the 10th International Conference on Physics Teaching in Engineering Education. pp. 1–7.

The MathWorks, Inc., 2023. MATLAB® Grader™ . Https://www.mathworks.com/products/matlab-grader.html.