Evaluación automática para problemas de control

por

Carlos Sánchez Cazorla

Ingeniero de Telecomunicación por la Escuela Superior de Ingenieros de la Universidad de Sevilla

Presentada en la

Escuela Técnica Superior de Ingeniería

de la

Universidad de Sevilla

para la obtención del

Grado de Doctor Ingeniero de Telecomunicación

Sevilla, Marzo de 2015

Agradecimientos

Ante todo, agradecer a mis padres su apoyo y cariño incondicional durante todos estos años. Sin ellos nada de esto sería posible. En especial a ti, Papá, por todo lo que me has enseñado. A mi hermano, por todo lo que hemos compartido. A Fabio, no sólo por haber dirigido esta tesis, sino por la confianza depositada en mí, por ser un amigo y por haberme guiado en numerosas aventuras. A mi tutor, David, le debo su enorme dedicación, trabajo y esfuerzo durante esta investigación. Ha sido el motor que ha impulsado esta tesis en los momentos de flaqueza. Estoy igualmente agradecido a Sebastián Dormido, Gonzalo Farias y Luis de la Torre por su colaboración en diversas publicaciones. Finalmente, a Lety, por estar siempre a mi lado y ayudarme en los momentos que más lo necesitaba.

Gracias a todos,

Sevilla, Marzo de 2015
Carlos Sánchez Cazorla

Índice general

Glosario

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CMS	Content Management System
EJS	Easy Java Simulations
EEA	Elemento de evaluación automática
FAQ	Frequently Asked Questions
GUI	Graphic User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IT	Information Technologies
IAE	Integral Absolute Error
ISE	Integral Square Error
ISNL	Interfaz de Sistemas No Lineales
JSON	JavaScript Object Notation
LVyR	Laboratorios virtuales y remotos
LCMS	Learning Content Management System
LMS	Learning Management System
MVC	Modelo-vista-controlador
PHP	PHP Hypertext Pre-processor
PMRs	Preguntas de múltiple respuesta
SAC	Sistema algebraico computacional
TICs	Tecnologías de la Cnformación y Comunicación
XML	Extensible Markup Language

Capítulo 1

Introducción

1.1. Sistemas de gestión de aprendizaje

En los últimos años, Internet ha sido el medio por excelencia que ha permitido hacer realidad un paradigma de trabajo en el que diferentes personas de diversas partes del mundo pueden colaborar en un proyecto común. Las redes plenamente integradas en la sociedad, conectan grupos de personas a escala global. Esto abre oportunidades de ofrecer recursos de aprendizaje a los estudiantes en cualquier momento y lugar.

La comunidad educativa ha dedicado un gran esfuerzo para sacar partido a estas nuevas posibilidades, aplicándolas a varios campos de educación en disciplinas científico-tecnicas, y de manera especial en ingeniería de control (B.S. Heck and Poindexter, S.E. and García R. , 2000; Dormido, 2002; Gomes, L. y Bogosyan, S., 2009). Por otro lado, la visualización e interactividad se han empleado con éxito para añadir una dimensión pedagógica y humana a los laboratorios virtuales y remotos (Dormido, R., Vargas, H., Duro, N., Sánchez, J., Dormido-Canto, S., Farias, G., Esquembre, F., Dormido, S., 2008; Farias G., De Keyser R., Dormido S., Esquembre F., 2010; De la Torre, L., Heradio, R., Sanchez, J., Dormido, S., Sanchez, J.P., Carreras, C., Yuste, M, 2011; De la Torre, L., Sanchez, J.P., Sanchez, J., Dormido, S., Yuste, M., Carreras, C., 2010; De la Torre, L., Sanchez, J., Dormido, S., Sanchez, J.P., Yuste, M., Carreras, C., 2011).

Este paradigma de trabajo ha sido posible no sólo por la capacidad de compartir información de forma eficiente y fácil, sino también por las posibilidades de colaboración que ofrecen nuevas herramientas como las redes sociales, servicios de streaming, videoconferencias, foros, blogs y chats.

Todas estas herramientas están englobadas en un mismo concepto genérico denominado sistema de gestión de contenidos o CMS (*Content Management System*). Se entiende por sistema de gestión de contenidos, aquella herramienta que posee un interfaz que controla una o varias bases de datos donde se almacena el contenido del sitio web al que se está accediendo. Estos sistemas permiten, a los diversos editores, manejar de manera independiente el contenido y el diseño. Existen diferentes tipos de CMS, pudiéndose clasificar según según su funcionalidad (blogs, foros, wikis, e-learning, e-commerce,...), el lenguaje de programación empleado (Java, PHP, ASP.NET,...) o la propiedad del código (abierto o propietario).

Atraídos por las posibilidades que ofrecen, las instituciones de enseñanza comienzan a usar estos CMS como herramientas de aprendizaje a través de la web (e-learning). De esta forma, surgen los llamados sistemas de gestión de contenidos para el aprendizaje (LCMS) o simplemente sistemas de gestión del aprendizaje (LMS). Un LMS genérico permite registrar usuarios, organizar cursos, almacenar información de los usuarios y generar informes de gestión para el profesor. Suelen incluir también herramientas para la comunicación entre participantes de un mismo curso.

Las mejoras obtenidas en los últimos años en el manejo y la accesibilidad de dichos sistemas, hacen que se hayan ampliado las posibilidades de formación a todos los niveles, superándose la barrera tecnológica que en un principio era el principal inconveniente del e-learning. De esta forma, los sistemas LMS son muy efectivos para la distribución de material, la recolección de asignaciones a alumnos y para ofrecer la oportunidad de un intercambio de conocimientos entre diferentes miembros de la comunidad educativa a través de foros, blogs y herramientas similares.

El empleo de sistemas de enseñanza on-line permite una reducción de los requisitos de movilidad de los alumnos, con las consiguientes ventajas para las empresas con empleados en procesos de formación, una interacción profesor-alumno más estrecha y con más recursos (foros, e-mail, FAQ, disponibilidad inmediata de apuntes, avisos, notas, planificación, etc) y por último, la introducción de nuevos elementos de evaluación (herramientas de test automático, sistemas de entrega y seguimiento de trabajos de curso, etc).

De esta forma, las principales motivaciones para emplear sistemas de enseñanza on-line vienen dadas por:

- una mejora de las tecnologías disponibles y aparición de potentes sistemas colaborativos gratuitos;
- una reducción de los requisitos de movilidad de los alumnos, con las consiguientes ventajas para las empresas con empleados en procesos de formación;
- una interacción profesor-alumno más estrecha y con más recursos (foros, e-mail, lista de preguntas frecuentes ,disponibilidad inmediata de apuntes, avisos,notas, planificación, etc);
- y por último, la introducción de nuevos elementos de evaluación (herramientas de test automático, sistemas de entrega y seguimiento de trabajos de curso, etc).

En el sistema de educación superior Europea (van der Wende, M. C., 2000), se alienta a los profesores a proporcionar atención personalizada a sus alumnos. Sin embargo, profesores de cursos universitarios a menudo emplean considerables esfuerzos y grandes dosis de tiempo para diseñar y corregir los ejercicios de sus alumnos, en vez de emplear este valioso tiempo para analizar y reforzar contenidos teóricos con ellos.

En relación con este aspecto, es frecuente pensar que la introducción de sistemas web de evaluación automática redunda exclusivamente en una descarga de la tarea del profesor, a costa de emplear métodos de evaluación más rudimentarios que la corrección personalizada de trabajos y exámenes. Existe el temor añadido de que la evaluación en línea abra la puerta a la suplantación del autor del trabajo y por tanto pierde la fiabilidad del examen presencial. Es

por ello que la evaluación automática es un campo desafiante que ha sido objeto de estudio por la comunidad académica desde diferentes puntos de vista.

Posiblemente, las preguntas de múltiple respuesta (PMRs) constituyen el enfoque más popular, y están implementadas en la mayoría de plataformas de e-learnings (Vician, C.; Charlesworth, P., 2003; Penn, J. H.; Nedeff, V. M., 2000; Stewart, B.; Kirk, R.; LaBrecque, D.; Amar, F. G.; Bruce, M. R. M., 2006). Aunque las PMRs son una herramienta útil, no son apropiadas para evaluar ejercicios complejos como los que caracterizan a la mayoría de asignaturas de titulaciones de ingeniería o de ciencia. Sin embargo, en facultades técnicas se consigue una gran mejora si el profesor es capaz de realizar cálculos complejos, tales como simulaciones, para validar las respuestas de los alumnos. Hay varios sistemas web basados en el uso de un sistema algebraico computacional (SAC) con sistemas de entrega a través de Internet que proporcionan un sofisticado mecanismo matemático con el que evaluar el trabajo de los alumnos (Petridis, V.; Kazarlis, S.; Kaburlasos, V. G., 2003; Tartaglia, A.; Tresso, E., 2002; Gentil, S.; Exel, M., 2004; HuaQiang, J.; Liang, Z.; WangQiong, Y., 2009). En general, estos sistemas están dirigidos a diseñar cursos online basados en una secuencia de reducidas y concisas preguntas que pueden no ser apropiadas para cursos científico técnicos generales.

1.2. Laboratorios virtuales y remotos

En el seno de las escuelas de Ingeniería, los LMS pueden ser de especial utilidad, como se indica en (Dormido and Torres, 2005). En ellas, un gran número de profesores trabaja en el desarrollo de herramientas con nuevas características adaptadas a este campo. En particular, se han desarrollado un gran número de trabajos basados en laboratorios remotos que permiten a los alumnos realizar prácticas con equipos reales a través de Internet. Cabe mencionar los trabajos (Sanchez et al., 2004, 2006), donde los autores usan programación en Java (Easy Java Simulations) para crear laboratorios de control virtuales y remotos. Otra plataforma para la creación de laboratorios virtuales se presenta en (Cedazo et al., 2007), bautizada con el nombre *Ciclope*. Otros tele-laboratorios de control automático conocidos son los de (Casini, M., Prattichizzo, D., Vicino, A., 2003, 2004; Torres et al., 2006) a los que cabe añadir los presentados en la revista RIAI (Zuluaga et al., 2005; Guzmán et al., 2005).

Por otro lado, la tendencia actual es la de hacer una extensión del concepto de laboratorio virtual al de tele-laboratorios distribuidos para su uso conjunto por alumnos pertenecientes a diferentes universidades, como se hace en (Kerer et al., 2005; Domínguez et al., 2005; Jiménez et al., 2005). La experimentación en laboratorios tradicionales es esencial para los estudiantes de ingeniería, que necesitan comprender conceptos fundamentales desde dos perspectivas: teórica y práctica. Sin embargo, los elevados costes asociados con equipamiento, espacio, y personal de mantenimiento imponen ciertas restricciones sobre los recursos. Los laboratorios virtuales y remotos (LVyR en adelante) pueden beneficiarse en gran medida de los avances en las TICs. Hay tres características particulares que pueden ser aplicadas a la educación en ingeniería (Dormido, S., Esquembre, F., Farias, G., y Sánchez, J., 2005; Farias, 2010): comunicación por red, visualización e interactividad.

El objetivo último a la hora de usar laboratorios virtuales y remotos es la realización de experimentos con ellos. Tales experimentos se llevan a cabo por parte de los estudiantes con el fin de comprender los conceptos prácticos y teóricos de los sistemas en estudio. Normalmente,

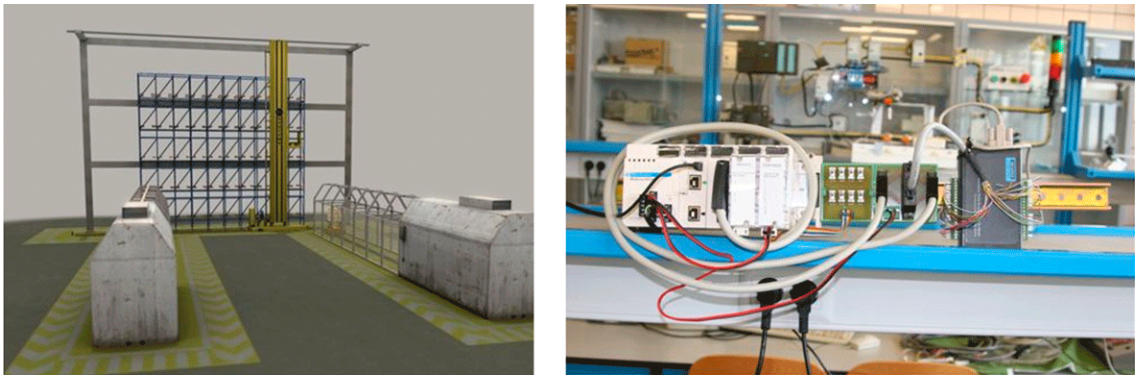

Figura 1.1: Simulador industrial ITS PLC y conexionado del autómata para el control del mismo (E.T.S. de Ingeniería, Sevilla)

(a) Cinta transportadora (b) Grúa (c) Sensor de barrera

Figura 1.2: Simulador para control y automatización utilizando un entorno 3D interactivo

en LVyR es preciso realizar una serie de actividades para llevar a cabo los experimentos. El docente puede evaluar los resultados alcanzados en la experimentación preguntando a los alumnos determinados valores resultado.

Otro ejemplo de uso de los laboratorios virtuales es el de usar herramientas de realidad virtual y modelado físico para sistemas de producción. Este paradigma es reciente, de gran atractivo para los alumnos y presenta un potencial enorme, incluso con interesantes perspectivas comerciales. Los simuladores físicos emplean tecnología propia de videojuegos para la simulación, lo más realista posible (incluyendo colisiones, gravedad...), de entornos de producción. En el caso particular de la ingeniería de sistemas y automática destacan las herramientas que simulan sistemas de producción industrial automatizados. Estos sistemas gestionan una gran cantidad de información visual: movimiento de material, cintas transportaras, vehículos autónomos, grúas, robots manipuladores, etc.

Si las señales que gobiernan los sistemas móviles se pueden extraer de la computadora donde se ejecuta la simulación, y conectar a un ordenador industrial, autómata o controlador lógico programable (PLC del inglés *Programmable Logic Controller*), entonces dispondremos de una herramienta de enseñanza y experimentación sumamente estimulante en la que los alumnos desarrollan los algoritmos de control en los PLCs y ven el resultado de sus acciones en el espacio 3D del simulador con el máximo realismo. Cualquier error en la secuencia de manipulación del material involucrado en la producción dará lugar a escenas de rotura de la producción, provocando caídas de pallets, acumulación de cajas, desorden, etc.

Dicho escenario no es nuevo, está presente en herramientas comerciales como ITS-PLC

(*www.realgames.pt*) empleada en la docencia en asignaturas de automatización en el seno del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla, y refrendada por los alumnos mediante encuestas de satisfacción. La figura 1.1 muestra una captura de pantalla del dicho simulador 3D de un almacén industrial automático, y el conexionado de un autómata industrial Telemecanique Modicon M340 a dicho simulador, presente en los laboratorios el mencionado Departamento. Cabe destacar también la realización de un simlulador programable en (Sánchez, A., Escaño, J., Muñoz de la Peña, D., Gómez-Estern, F., 2013) mostrado en la figura 1.2.

1.3. Evaluación automática

En las pasadas décadas la tecnología de la información están ocasionando un profundo impacto en la educación, y las plataformas web colaborativas son ahora herramientas comunes en instituciones educativas a todos los niveles. Entre ellas cabe mencionar Moodle y WebCT por lo extendido de su uso, aunque hay muchas otras como por ejemplo Blackboard, eCollege, Desire2Learn, Skillfactory, Dokeos, SakaiProject, Atutor, Ilias, Claroline, Saba-Learning, etc. En (Poindexter, S. E.; Heck, B. S., 1999; Kapur and Stillman, 1997; B.S. Heck and Poindexter, S.E. and García R. , 2000) se muestra un resumen sobre qué puede hacerse en un curso utilizando la web, y en (Vician, C.; Charlesworth, P., 2003) para una evaluación del sistema de gestión de aprendizaje empleado en primer curso de química.

Aunque el uso de LVyR puede ser de gran utilidad, numerosas actividades de este tipo pueden acarrear un esfuerzo excesivo por parte de los profesores a la hora de calificar el trabajo realizado, especialmente en cursos numerosos. Precisamente en estos casos la evaluación automática es especialmente valiosa.

Esta problemática también está presente en los sistemas LMS. A pesar de su eficiencia en la recolección y almacenamiento del trabajo del alumno, en general ofrecen una capacidad de evaluación automática limitada a la evaluación de cuestionarios de respuesta múltiple (en diferentes modalidades). En particular, los mecanismos de evaluación ofrecidos no permiten la evaluación personalizada y además no están diseñados para disciplinas de ingeniería con una fuerte carga matemática.

En el campo de la algoritmia y la programación, hay incluso menos aplicaciones y herramientas disponibles aunque cabe destacar que se han presentado diversos laboratorios virtuales específicos que sí disponen de herramientas de autoevaluación en (Rodríguez et al., 2007c,b,a; García et al., 2009), los tres primeros para programar periféricos y el cuarto para programar un microprocesador virtual basado en el *MC88110*. Sin embargo, en estos casos, las herramientas son específicas y están completamente cerradas, no disponiendo de la flexibilidad necesaria para ser fácilmente modificadas y empleadas en otras asignaturas.

No obstante, el mayor inconveniente en todos ellos es que el proceso de evaluación en ellos está apenas automatizado, por lo que la tarea de evaluación es realizada en su totalidad por el profesor. Esto hace que estos laboratorios no sean de utilidad para asignaturas de ingeniería con un gran número de estudiantes. En el caso de trabajos prácticos y exámenes, tal y como demuestra la escasa literatura disponible, apenas se ha profundizado en el estudio de sistemas de evaluación flexibles y automáticos para cursos con un gran número de estudiantes.

1.4. Goodle GMS

Teniendo en cuenta las limitaciones de los sistemas actuales, se ha desarrollado un sistema de e-learning denominado Goodle GMS específico para ingeniería, que permite automatizar la recogida y evaluación de ejercicios prácticos de diferentes disciplinas con la complejidad típica que la formación técnica requiere y con la posibilidad de personalizarlos para cada alumno. El uso de la herramienta presentada permite, entre otras cosas, incrementar la periodicidad de los exámenes y pruebas, ya que en la Universidad, la masificación y la escasez de recursos evaluadores (profesores) suelen ser los principales obstáculos para aumentar dicha frecuencia. Del mismo modo, también permite establecer un criterio único y homogéneo a la hora de evaluar al alumnado que se puede mantener invariable con el paso de los años y los cambios de profesorado.

En el marco de esta tesis se expone que la plataforma Goodle GMS no sólo facilita el trabajo de los docentes permitiéndoles centrarse en tareas más creativas, sino también puede mejorar significativamente el aprendizaje del alumno mediante la presentación de ejercicios sucesivos con informes inmediatos sobre su rendimiento. Si el grupo de alumnos es numeroso, este tipo de tareas se hace inasequible a un solo profesor sin la ayuda de estos medios.

Aunque Goodle GMS está diseñado como una plataforma genérica, inicialmente se aplicó de forma específica en asignaturas de Teoría de Sistemas y Control Automático. Posteriormente se extendió a asignaturas de programación en C, Matlab y Easy Java Simulations (EJS), ampliando drásticamente el abanico de potenciales usuarios.

1.5. Fundamentos de control automático

Un sistema de control automático se define como un conjunto de dispositivos capaces de ajustar el funcionamiento de un sistema manipulable para unos determinados fines; por ejemplo, mantener la temperatura de una caldera, el rumbo de un barco o la velocidad de un automóvil en un valor establecido. En este caso, las decisiones las toma un dispositivo denominado controlador, como por ejemplo un sistema de climatización o el control de los motores de un coche híbrido. El control automático se ha convertido en una pieza fundamental para el avance de la ingeniería y la ciencia. La mayor parte de los sistemas modernos (rocesos de fabricación, vehículos, operaciones industriales, sistemas robóticos, etc.) no podrían funcionar sin la ayuda de sofisticados sistemas de control. En general, los sistemas de control se pueden aplicar a cualquier tipo de operación que comprenda una actuación a partir de unos datos recogidos por un sensor.

Un ejemplo sencillo es el de un lector de CD o DVD. Al introducir un CD en un lector, se espera que la información contenida en él pueda ser extraída para ser tratada posteriormente por un computador o un equipo de música. Para poder acceder a dicha información es necesario gestionar la tensión de un motor eléctrico para situar una lente con mucha rapidez y precisión en una posición determinada. Obviamente, sería imposible para un operador humano realizar esta operación con tantas prestaciones, por lo que esto no sería posible sin un control automático. Obsérvese además que en este caso es muy importante que el sistema de control sea capaz de hacer su labor lo más rápidamente posible. Así disminuirá el tiempo de espera y el producto tendrá mejores prestaciones.

Un concepto clave en el control (ya sea manual o automático) es el de realimentación, que consiste en el uso de la medida de la variable a controlar para la toma de decisiones. En el caso del control manual de la velocidad del coche, el conocimiento de la velocidad instantánea del vehículo se utiliza para decidir si se debe pisar más el acelerador (cuando la velocidad del vehículo sea inferior a la deseada) o lo contrario. Obsérvese cómo para realizar la comparación entre la velocidad deseada y la real se necesita una medida de esta última, la cual se obtiene a partir del velocímetro del coche.

Mediante la realimentación se pueden conseguir objetivos que pueden parecer asombrosos o misteriosos para los no iniciados en la teoría del control. Por ejemplo, en 1934 H. Black aplicó la realimentación al amplificador con lo que consiguió unas extraordinarias características de linealidad necesarias para poder conectar varios amplificadores en serie en largas conexiones de telefonía sin que se produjeran distorsiones. La linealidad del amplificador realimentado era muy superior a la que se podía conseguir con la tecnología existente en esa época en amplificadores sin realimentar. La realimentación introdujo mejoras significativas con un coste muy bajo.

Obsérvese el porqué del término realimentación: la orden que se le da a un actuador provoca cambios en el sistema y, por tanto, en la variable que se mide. Con el sensor se miden estos cambios y, tras un procesamiento de esta información, se decide cuál es la siguiente orden que dar al actuador. Se puede entender este proceso como que la información se ha vuelto a alimentar, se ha realimentado, en el sistema. Se habla entonces de control en *bucle cerrado* en contraposición a los sistemas sin realimentación, a los que se les denomina en *bucle abierto*. Un sistema en bucle cerrado (manual o automático) consta siempre de algún tipo de sensor; en el ejemplo del control manual de velocidad de un vehículo el sensor sería, entre otros, el sentido de la vista, mientras que en el control automático de velocidad el vehículo necesita de un dispositivo para medir la velocidad, un velocímetro.

La ingeniería de control estudia como diseñar los sistemas de control automático. Como puede imaginar el lector, el diseño de éstos abarca un problema muy amplio, y a menudo, interdisciplinar: describir los objetivos de control; modelar matemáticamente el sistema para definir las variables manipulables, las perturbaciones, las variables internas y las salidas de interés; uso de instrumentación para medir las variables de interés del sistema; etc.

La dinámica de muchos sistemas, ya sean mecánicos, eléctricos, térmicos, económicos, biológicos, etc., se describe en términos de ecuaciones diferenciales. Dichas ecuaciones diferenciales se obtienen a partir de leyes físicas que gobiernan un sistema determinado (las leyes de Newton para sistemas mecánicos o las leyes de Kirchhoff para sistemas eléctricos).

En la teoría de control automático a menudo se utiliza el concepto de función de transferencia para definir la relación entre las entradas y salidas de un sistema y se describe mediante ecuaciones diferenciales lineales invariantes en el tiempo. Así, la función de transferencia se define como el cociente entre la transformada de Laplace de la salida (función de respuesa) y la transformada de Laplace de la entrada (función de excitación) bajo la suposición de que todas las condiciones iniciales son cero.

A partir del concepto de función de transferencia, es posible representar la dinámica de un sistema mediante ecuaciones algebraicas en s. Si la potencia más alta de s en el denominador de la función de transferencia es igual a n, el sistema se denomina *sistema de orden n-ésimo*.

Los sistemas de control actuales son, por lo general, no lineales. Sin embargo, si es posible

aproximarlos mediante modelos matemáticos lineales, podemos usar uno o más métodos de diseño bien desarrollados, como por ejemplo el lugar de las raíces, la respuesta en frecuencia, la aproximación en el espacio de estados o el basado en compensadores proporcional-integral-derivativo.

Por último, es importante señalar que un problema de diseño de controladores por lo general no admite una solución única, y además es necesario realizar una serie de simulaciones para comprobar si una solución es o no apropiada. Esto hace a estos tipos de problemas particularmente complicados de evaluar por parte de un profesor. Sin embargo, es muy importante que los alumnos resuelvan diferentes problemas para desarrollar esta competencia, por lo que la evaluación automática es particularmente apropiada para ser aplicada a esta disciplina.

1.6. Resultados principales

Esta tesis presenta una plataforma de apoyo al profesor para poder ofrecer un sistema de recogida, almacenamiento y evaluación automática de problemas personalizados en asignaturas científico-técnicas. Esta plataforma además ha sido probada y utilizada durante varios cursos en asignaturas de control automático y química analítica de diversas universidades. La evaluación automática constituye un campo de investigación desafiante que ha recibido mucha atención estos últimos años y ha sido objeto de estudio por la comunidad académica de muchas maneras.

Los principales resultados obtenidos en esta tesis son los siguientes:

1. **Desarrollo de una nueva plataforma e-learning Goodle GMS basado en ejecución de código**: La premisa en la que se basa esta plataforma es que los alumnos deben presentar la solución a los ejercicios prácticos siguiendo una cierta sintaxis de un lenguaje de programación (Matlab entre otras posibilidades). El código del estudiante se ejecuta en el servidor junto con un código diseñado por el instructor para evaluar el trabajo del estudiante.

2. **Sistema de generación de enunciados personalizados integrado en Goodle GMS**: La implementación de un nuevo módulo en Goodle GMS permite que un profesor puede generar enunciados individualizados de forma aleatoria.

3. **Desarrollo de una plataforma de evaluación basada en modelo de caja negra**: Esta plataforma incluye, entre otras, funcionalidades tales como competitividad entre alumnos, personalización de enunciados y generación de informes.

4. **Extensión de los anteriores elementos a laboratorios virtuales y remotos**: La integración entre Goodle GMS y la herramienta Easy Java Simulations permite que profesores sin conocimientos de programación puedan realizar simulaciones interactivas en Java y evaluar a sus alumnos de forma automática.

5. **Aplicación de la evaluación automática en cursos de control de sistemas no lineales**: Como aplicación específica a control automática se ha desarrollado un módulo de EJS para sistemas no lineales. Su objetivo es la resolución de problemas de control no lineal, empleando una herramienta gráfica que permita el análisis y visualización de diferentes controladores para sistemas de hasta orden cuatro. A partir de diferentes parámetros intrínsicos del alumno (por ejemplo su DNI), este módulo permite al profesor la creación de ejercicios personalizados. De esta forma, el alumno puede experimentar con diferentes controladores y visualizar la dinámica del sistema completo.

6. **Aplicación en una asignatura de análisis instrumental**: El empleo de la metodología de Goodle GMS se ha aplicado en el diseño de ejercicios de análisis instrumental de química analítica que pueden ser evaluados y personalizados para cada alumno de forma automática.

7. **Desarrollo de una interfaz gráfica para el profesor**: Esta interfaz abarca todo el ciclo de trabajo del profesor, desde la definición del problema básico hasta la generación de los ejercicios personalizados para cada alumno. Esta herramienta permite generar de

forma simple, y sin necesidad de tener conocimientos de programación, ejercicios individualizados auto-evaluables garantizando la existencia de solución y la homogeneidad en la dificultad y procedimientos de resolución de los mismos.

8. **Puesta a prueba en cursos reales**: Utilización de esta plataforma en cursos reales con numerosos alumnos y donde era necesario realizar una recogida sistemática de datos. Elaboración y análisis de cuestionarios de satisfacción de alumnos y profesores.

1.7. Esquema de la Tesis

El resto de capítulos de esta tesis se encuentra organizada de la siguiente forma:

Capítulo 2

En este capítulo se describe la plataforma Goodle GMS de evaluación desarrollada en 2009 a partir de una primera versión de 2007. Se presentan sus principales propiedades, a la vez que la complejidad asociada al problema de recogida, almacenaje y evaluación automática de ejercicios de alumnos.

Capítulo 3

Este capítulo se centra en el elemento de evaluación automática (EEA), que dota de funcionalidades de calificación automática a aplicaciones preexistentes creadas mediante Easy Java Simulations (EJS). Esto permite realizar tareas de calificación automática en laboratorios virtuales remotos. EJS es una herramienta de software libre desarrollada en Java, diseñada especialmente para la creación de simulaciones por computador en tiempo discreto dotadas de interfaces gráficas interactivas.

Capítulo 4

El capítulo presenta una intefaz de sistemas no lineales (ISNL) para la resolución de problemas de control no lineal, empleando el la plataforma Goodle GMS y el elemento de evaluación automática presentado en los capítulos anteriores.

Capítulo 5

Este capítulo aborda el problema de diseñar ejercicios de análisis instrumental de química analítica que pueden ser evaluados y personalizados para cada alumno de forma automática. La problemática estriba en la dificultad de parametrizar de forma personalizada un problema definido a partir de datos experimentales, ya que no se puede emplear la técnica descrita en capitulos anteriores donde se relacionan parámetros con funciones sencillas a partir de un dato característico del alumno.

Capítulo 6

El capítulo presenta una herramienta web para la generación de ejercicios individualizados auto-evaluables garantizando la existencia de solución y la homogeneidad en la dificultad y procedimientos de resolución de los mismos. Mediante esta herramienta se evita que el profesor tenga conocimientos de programación para el diseño de ejercicios.

Capítulo 7

El último capítulo hace un resumen de los resultados obtenidos y propone posibles líneas de investigación futuras.

1.8. Publicaciones

Los resultados presentados en esta tesis han dado lugar a una serie de publicaciones en revistas especializadas y en congresos internacionales. Los detalles de las publicaciones que tienen una relación directa con este trabajo se detallan a continuación:

- G. Farias, F. Gomez-Estern, L. De la Torre, D. Muñoz de la Peña, C. Sánchez, S. Dormido. Tecnología para dotar a los Laboratorios Virtuales y Remotos de funcionalidades de Evaluación Automática. Jornadas de Automática, 2012.
- G. Farias, F. Gomez-Estern, L. De la Torre, D. Muñoz de la Peña, C. Sánchez, S. Dormido. Enhancing Virtual and Remote Labs to Perform Automatic Evaluation. 9th IFAC Symposium Advances in Control Education. 2012
- C. Sánchez, F. Gómez-Estern, D. Muñoz de la Peña. A virtual lab with automatic assessment for nonlinear controller design exercises. 9th IFAC Symposium Advances in Control Education. 2012.
- D. Muñoz de la Peña, F. Gómez-Estern, S. Dormido. A new Internet tool for automatic evaluation in Control, Systems and Programming, IEEE Computers & Education Vol. 59, Pag. 535-550, 2012.
- Arsenio Muñoz de la Peña, David Muñoz de la Peña, María P. Godoy-Caballeroa, David González-Gómezc, Fabio Gómez-Esternd and Carlos Sánchez. Automatic evaluation and data generation for analytical chemistry instrumental analysis exercises. Química Nova.2014
- G. Farias, D. Muñoz de la Peña, F. Gómez-Estern, L. De la Torre, S. Dormido, C. Sánchez. Adding automatic evaluation to remote experimentation. Interactive Learning Environments.2014
- C. Sánchez, F. Gómez-Estern, D. Muñoz de la Peña. Plataforma para la formación práctica individualizada en fundamentos de control automático. Jornadas de Automática, 2014.

- Sánchez Cazorla, C., Gómez-Estern F., Muñoz de la Peña, D. Plataforma para la formación práctica individualizada en Fundamentos de Control Automático. Revista Iberoamericana de Automática e Informática industrial (Enviado a RIAI)

Capítulo 2

Evaluación automática vía web

2.1. Introducción

En este capítulo presentamos la nueva herramienta web de evaluación desarrollada en 2009 a partir de una primera versión de 2007. Vamos a hablar de la metodología empleada en la evaluación automática, los diferentes paradigmas de evaluación y la arquitectura de la aplicación propuesta. Además se presentarán los resultados de dos casos de estudio de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Los contenidos de este capítulo fueron publicados en (Muñoz de la Peña, D. y Gomez-Estern, F., 2009), donde se describe el uso de la evaluación automática vía web la asignaturas Teoría de Sistemas y Fundamentos de Informática de la titulación de Ingeniero Industrial, y se ofrece una descripción detallada del motor interno del sistema.

2.2. Paradigmas de evaluación automática

La mayoría de las plataformas de educación virtual incluyen herramientas de evaluación automática. La herramienta más común es una prueba de selección múltiple, donde el estudiante debe responder a algunas preguntas eligiendo una o más opciones de una lista de posibles respuestas. Existen diferentes variaciones en torno a este esquema. Por ejemplo, las preguntas se pueden elegir al azar de una base de datos, de manera que ningún par de exámenes es idéntico, o incluso adaptarse a las últimas respuestas del estudiante como se propone en M. Barla,M. Bieliková,A. B. Ezzeddinne, T. Kramár, M. Simko,O. Vozár (2010). Además, los sistemas de puntuación también ofrecen un cierto grado de personalización. Algunas herramientas de evaluación disponibles están dotadas de cierta capacidad aritmética básica, con el fin de comprobar que la solución numérica proporcionada por el estudiante es precisa.

La plataforma que se presenta aquí se basa en un método de evaluación totalmente diferente, que se inspira en la arquitectura en la figura 2.1. La idea clave es que los estudiantes deben presentar la solución a los ejercicios prácticos siguiendo una cierta sintaxis de un lenguaje de programación (MATLAB entre otras posibilidades). El código del estudiante se ejecuta en el servidor junto con un código diseñado por el instructor para evaluar el trabajo del estudiante. Denominaremos a este marco de trabajo evaluación basada en modelo de caja negra.

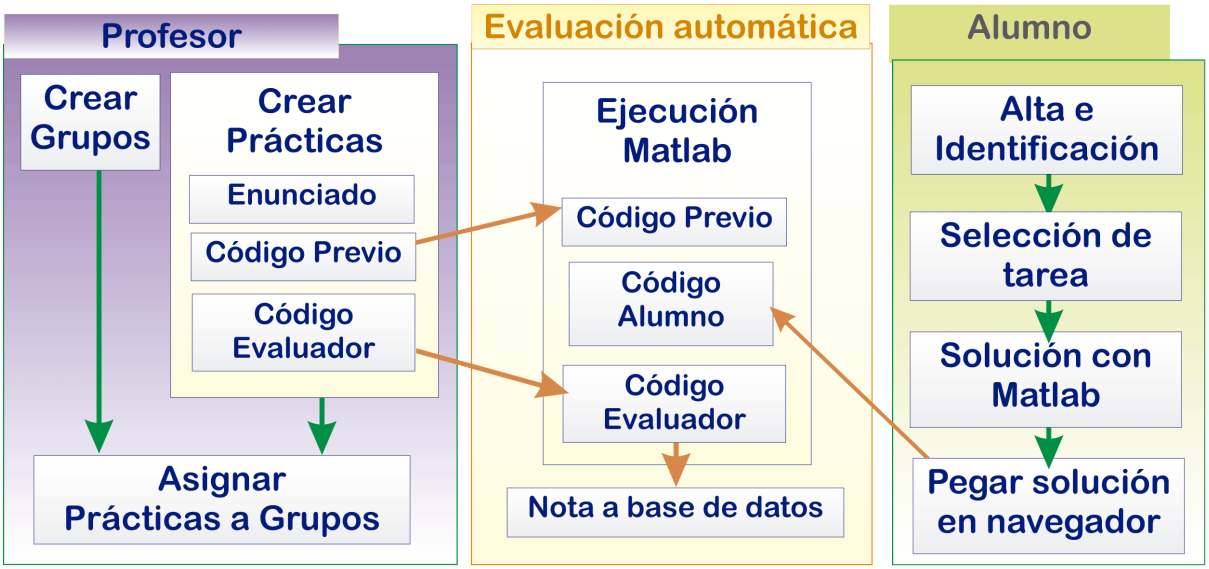

Figura 2.1: Funcionamiento del sistema de evaluación automática

El código de evaluación se genera por el profesor satisfaciendo algunas restricciones de diseño, en particular, el orden en el que el código de los alumnos y su código son ejecutados/compilados, y la manera en la que la nota del alumno y los comentarios tienen que ser almacenados para que el servidor pueda recuperarlos y grabarlos en la base de datos. Estas restricciones de diseño dejan libertad al profesor para diseñar ejercicios complejos y su correspondiente evaluación, utilizando todas las posibilidades que el lenguaje de programación proporcione, por ejemplo bibliotecas de código C y MATLAB.

Bajo esta estructura general, se pueden implementar diferentes paradigmas de evaluación que sobrepasen los límites conceptuales de los ejercicios de selección múltiple. En las siguientes secciones se describirán los siguientes paradigmas:

- **Coincidencia exacta en el resultado**: En este paradigma la solución implica dar valores a un conjunto específico de variables siguiendo las reglas de sintaxis del lenguaje de programación. Se puede aplicar a
  - Problemas individuales.
  - Tareas cooperativas.
- **Evaluación basada en el rendimiento**: En este otro paradigma la solución propuesta por el alumno no es la única, por lo que la evaluación se puede realizar de diferentes maneras:
  - Comprobación específica.
  - Evaluación competitiva.
- **Ejercicios de diseño de algoritmos**: En este modelo los alumnos deben diseñar un programa en un lenguaje de programación concreto

Con el fin de acomodar las diferencias entre estos paradigmas, se han implementado pequeñas variaciones en la secuencia de ejecución representada en la figura 2.2, a través de

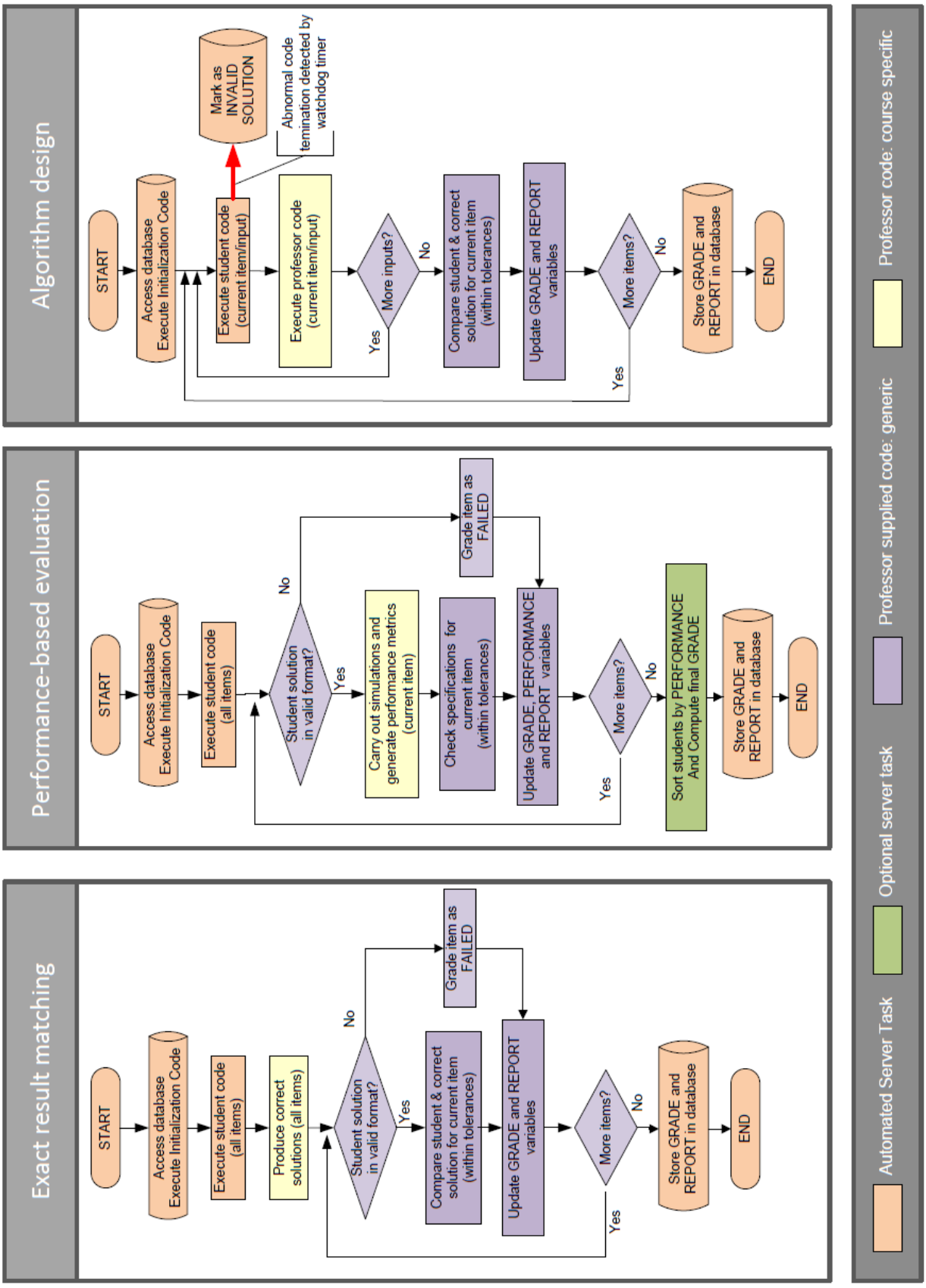

Figura 2.2: Diagramas de flujo de diferentes tipos de paradigmas de evaluacion

un conjunto de opciones de usuario para la personalización de los ejercicios de los estudiantes. Esto se discutirá más adelante en la sección 2.5 donde se presentarán implementaciones específicas de evaluadores.

La afirmación de que esta arquitectura, que conlleva sus propios riesgos y dificultades, es segura y eficaz, se basa en hechos experimentales: lleva puesto en marcha cinco años en seis cursos diferentes en el departamento de Ingeniería de Sistemas y Automática, dando resultados precisos y fiables de evaluación que afectan a más de 2000 estudiantes (ver información en la sección 2.6). Debe observarse que, durante este período, se ha realizado una cantidad considerable de ajustes hasta la presente implementación.

2.2.1. Coincidencia exacta en el resultado

En este paradigma, los estudiantes envían una serie de soluciones numéricas a un conjunto de problemas propuestos por el profesor. Para este fin, la solución implica dar valores a un conjunto específico de variables siguiendo las reglas de sintaxis del lenguaje de programación. Hay que destacar que los estudiantes no necesitan conocer dicha sintaxis si se proporciona una plantilla.

Para poder establecer una puntuación, el ejercicio debe contener un código de evaluación automática generado por el profesor. En principio, este paradigma se puede implementar utilizando cuestionarios de selección múltiple, sin embargo, utilizando el enfoque propuesto de caja negra es posible personalizar los parámetros numéricos de los ejercicios como una función de los dígitos de un identificador personal único o número de inscripción (que puede ser proporcionado por el profesor). En ese caso, cada estudiante recibe un problema exclusivo y le impedirá compartir sus soluciones. Esta arquitectura requiere un intercambio entre la base de datos, que contiene el número de identificación del estudiante, y el código de evaluación. Con el fin de ilustrar esto, se propone el siguiente problema de ecuaciones de segundo orden.

Pregunta: Proporcionar las soluciones de la ecuación de segundo orden $ax^2 + bx + c = 0$ donde (a, b, c) se definen por los tres primeros dígitos de su DNI. Asigne las soluciones a las variables $sol1$ y $sol2$.

Para definir la solución el alumno debe rellenar la siguiente plantilla:

```
sol1 = 0; % Introduzca la primera raíz
sol2 = 0; % Introduzca la segunda raíz
```

Con el fin de comprobar que este resultado es correcto, el servidor accederá a la base de datos para recuperar el identificador del estudiante, definirá (a, b, c), generará las soluciones correctas, ejecutará el código del estudiante para obtener su solución, y comprobará si ambas soluciones son iguales (en este caso, una tolerancia debe ser aceptada con el fin de aceptar diferentes maneras de calcular el resultado).

En nuestro ejemplo, suponiendo que el identificador ID se almacena en un vector, y que el alumno entrega las soluciones en el mismo orden que MATLAB calcula las raíces del polinomio, el código evaluador creado por el profesor podría ser:

```
sol = roots(ID(1:3));
nota = 0;
if(norm(sol(1)-sol1) < eps)
    nota = nota + 1;
end
if(norm(sol(2)-sol2) < eps)
    nota = nota + 1;
end
```

En este código de evaluación, *nota* tomaría un valor entre 0 y 2 y eps es un error de tolerancia. El servidor recupera automáticamente el valor de la variable nota y la almacena en la base de datos. Hay que tener en cuenta que con el fin de ejecutar este código, la variable de identificación del alumno se debe crear en el espacio de trabajo de MATLAB. Esto se realiza por parte del servidor, que accede a la base de datos, recupera el identificador y agrega automáticamente el siguiente código:

```
ID = [x x x x x x x];
```

En este paradigma, la solución se puede dividir en partes interdependientes (por ejemplo, código fuente C++ dividido en módulos), cada una de ellas enviada por un estudiante diferente en un grupo. A continuación, el servidor debe fusionar todas las partes con el fin de evaluar los resultados. Esto abre la posibilidad de implementar esquemas de colaboración.

2.2.2. Evaluación basada en el rendimiento

Otro paradigma, que abre un interesante conjunto de posibilidades, es aquel donde *la solución propuesta por el alumno no es la única*. En este caso, los estudiantes producen una solución que de nuevo se ejecuta en el servidor como código fuente (sin que los alumnos tengan conocimiento de este hecho). Sin embargo, en lugar de comparar las variables del espacio de trabajo generadas con una respuesta correcta pre-especificada, se procesarán por un conjunto de simulaciones parametrizadas a partir de su solución. Finalmente los resultados de la simulación servirán para medir la calidad de la solución. La evaluación se puede realizar de diferentes maneras:

Comprobación específica

Los resultados (los vectores de salida del sistema) se analizan con el fin de comprobar que los requisitos especificados en el enunciado se cumplen (por ejemplo, en un sistema de bucle de cerrado). Esto resulta en una evaluación binaria independiente para cada estudiante.

Evaluación competitiva

Además del procedimiento anterior, para cada alumno se almacena en la base de datos una figura de mérito obtenida a partir de simulaciones (por ejemplo, el tiempo de subida en

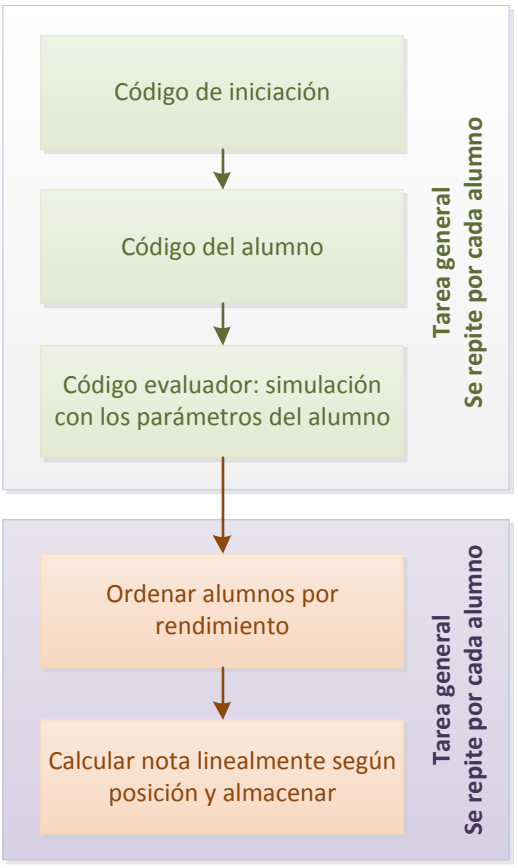

Figura 2.3: Secuencia de ejecución del servidor en evaluación competitiva

bucle cerrado en el diseño del sistema de control). Luego, los estudiantes se ordenan según ese valor, y se puntúan en función de su posición dentro del grupo. Esto se implementa variando la secuencia de ejecución estándar (figura 2.1) ilustrada en la figura 2.3.

Un escenario en el que consideramos que es muy conveniente hacer uso de la arquitectura propuesta para la evaluación (ya sea en forma competitiva o no) sería en el diseño de controladores y los ejercicios de sintonización. En este contexto, se propone el siguiente modelo interactivo:

1. El profesor del curso propone una función de transferencia $G(s)$ (con parámetros individualizados para cada alumno) y una estructura de controlador, por ejemplo un PID o una red de avance-retraso. Tenga en cuenta que la complejidad del modelo del sistema y la estructura del controlador no están de ninguna manera delimitadas por las especificaciones de la plataforma de evaluación, que puede utilizarse para enseñar técnicas avanzadas de control en los cursos de postgrado.

2. El profesor propone algunas especificaciones en circuito cerrado que se deben alcanzar con el diseño de control, tales como el error en estado estacionario, los márgenes de

ganancia, rechazo de perturbaciones, etc.

3. El alumno emplea los métodos explicados en las clases teóricas para diseñar un conjunto de parámetros de un controlador (por ejemplo, K_p, T_d, T_i en el caso de sintonización de un PID) para satisfacer estas especificaciones. El estudiante puede comprobar el resultado de esta solución simulando en bucle cerrado a través de Simulink, con el comando **lsim** o mediante los comandos del *Control System Toolbox* (por ejemplo **margin**).

4. El alumno envía los parámetros del controlador al profesor.

5. El profesor evalúa si las especificaciones se cumplen con las herramientas antes mencionadas.

Pregunta: Dada la función de transferencia $G(s)$ correspondiente a su identificación, diseñe un Controlador PI para garantizar que la respuesta ante escalón del sistema de bucle cerrado resultante tiene una sobreoscilación inferior a $0{,}2$ y un tiempo de subida inferior a 10 segundos.

Para definir la solución el alumno debe rellenar la siguiente plantilla:

```
Kp = 0; % Introduce la ganancia propocional
Ti = 0; % Introduce el tiempo integral
```

Suponiendo que la función de transferencia se genera previamente a partir de la identificación, el código MATLAB del profesor podría ser:

```
c = tf(Kp*[Ti 1],[Ti 0]);
gbc = g*c/(1+g*c);
[Y,T] = step(gbc);
yrp = Y(lenght(Y));
ymax = max(Y);
SO = (ymax-yrp)/yrp;
ts = T(min(find(Y>yrp)));
if(SO<0.2)
    nota = nota + 1;
end
if(ts<10)
    nota = nota + 1;
end
```

Es imporntate destacar que la forma de generar la función de transferencia a partir del número de identificación es una cuestión clave en el desarrollo de este ejercicio, y, en general, depende de las especificaciones del sistema en bucle cerrado. Una opción puede ser definirlo a partir de los dígitos de la identificación, por ejemplo:

```
g = zpk([],[-ID(1) -ID(2) -ID(3)],10);
```

Considere que en este caso, dependiendo del número de identificación, podría no ser posible cumplir con las especificaciones. Otra opción, es tener una lista de posibles parámetros de problemas, que se asignan por el profesor a cada alumno. En ese caso el profesor tendría que agregar esta asignaciones al código de evaluación, por ejemplo:

```
if ID = 10230405
    g = tf([2,[1 4 5 1]);
else if ID = 19471453
    g = tf([2,[1 2 4 2]);
else if ...
```

Esta lista tendría que tomar en cuenta todos los estudiantes. Observe que hay muchas maneras diferentes de resolver el problema de definición de parámetros a partir del identificador obtenido procedente de la base de datos. Sólo hemos presentado dos posibles ejemplos.

2.2.3. Ejercicios de diseño de algoritmos

Por último, el procedimiento de evaluación de *caja negra* propuesto permite al profesor evaluar problemas de diseño de algoritmos donde los estudiantes deben diseñar un programa en un lenguaje de programación concreto. En algunos cursos, no basta con evaluar sólo algunos parámetros de un controlador estructurado, sino todo un algoritmo implementado por un alumno.

Este caso es especialmente importante para los cursos de programación general y se ha aplicado en cursos de lenguaje C y programación MATLAB de la Universidad de Sevilla. Además, procedimientos de diseño de controladores complejos pueden ser evaluados para cursos de postgrado de mayor complejidad. La idea para evaluar automáticamente un ejercicio de programación tiene diferentes enfoques; el más sencillo, pero probablemente el más eficaz es simplemente ejecutar el código del estudiante. Si su algoritmo es visto como una función de procesamiento de información, y en la mayoría de los casos así es, el evaluador automático debe proporcionar datos de entrada al azar a la función, ejecutar el código, y comparar la salida con una versión proporcionada por el profesor. Para ilustrar este paradigma, se propone el siguiente problema en lenguaje C.

Pregunta: Definir una función que evalúe el valor absoluto de un número real. El prototipo de la función es **float absolute(float x);**.

Para definir la solución el alumno debe rellenar la siguiente plantilla:

```
float absolute(float x)
{ // Comienzo de la solución


} // Fin de la solución
```

Para comprobar que este resultado es correcto, el código del profesor ejecutará secuencialmente el código del estudiante y el código correcto, y comprobará si ambas soluciones son iguales. El servidor compila el código del profesor junto con la solución del alumno en un

programa único que escribe la nota en la salida estándar, que es leída directamente por el servidor para recuperar la nota.

En nuestro ejemplo, el siguiente código sería suficiente.

```
#include <stdio.h>
#include <math.h>
float absolute(float x);
void main(void)
{
    int nota = 1;
    float x;
    for(x=-10;x<10;x++)
        if(absolute(x)!=fabs(x))
            nota = 0;
    printf("%d",nota);
    return;
}
```

En este paradigma, la personalización de los ejercicios de los alumnos es un asunto mucho más complejo que en los paradigmas anteriormente mencionados donde los parámetros numéricos pueden definir un problema de análisis o de diseño.

2.3. Metodología

La herramienta presentada en este trabajo es un sistema de gestión de contenidos para el aprendizaje que además ofrece la funcionalidad de evaluar automáticamente ejercicios prácticos de diferente naturaleza. Para ello, la herramienta dispone de las siguientes características:

- Existe una metodología sencilla y flexible para la creación y almacenado de nuevos ejercicios.
- Existe un mecanismo simple y robusto para la entrega de los trabajos de los alumnos, así como de gestión de los mismos.
- Existe un método automático para compilar, ejecutar y evaluar los trabajos de los alumnos (calcular la nota).

Los dos primeros puntos (recogida y almacenamiento de información) son comunes a los sistemas de LMS existentes. Aun así, se ha desarrollado una plataforma particular utilizando tecnología estándar con características que la hacen atractiva desde el punto de vista de la gestión administrativa, tales como:

- Creación de grupos.
- Gestión de usuarios: identificación segura de alumnos y profesores.

- Creación, coordinación y almacenamiento de bibliotecas de ejercicios para posteriores cursos académicos, que pueden ser usados en diferentes asignaturas y por distintos profesores.
- Asignación de ejercicios a grupos.
- Control automático de las fechas límite de entregas de los ejercicios.
- Acceso a las soluciones y calificaciones de cada alumno para una revisión manual posterior, en caso de que el alumno lo requiriera.

La principal aportación del sistema desarrollado se encuentra en el campo de la automatización de la evaluación. Se ha desarrollado un sistema con dos objetivos fundamentales: personalizar los ejercicios para cada alumno introduciendo variabilidad en las preguntas a partir de un identificador personal (como el número de matrícula), y evaluar ejercicios basándose en cálculos matemáticos complejos, simulaciones numéricas y comprobación de algoritmos. El proceso de evaluación, lo más importante de este sistema, emplea el principio de evaluación funcional, o de "caja negra", consistente en coordinar los siguientes elementos:

- La solución del alumno, enviada en formato de texto a través de Internet. Dicho texto debe redactarse conforme a unas normas sintácticas que permitan ser tratadas *como instrucciones de ordenador* escritas en algún lenguaje de programación determinado (C, MATLAB, Ensamblador,...). El alumno no tiene por qué ser consciente de este hecho, ni conocer el lenguaje en su totalidad, sólo las reglas básicas que se explicarán en el enunciado.
- El algoritmo del profesor, escrito en el mismo lenguaje de programación, que es capaz de producir una entrada de información para el código (el conjunto de instrucciones) del alumno, ejecutar éste, y observar la información de salida tras el procesado del alumno. Dicha información permitirá calcular si la solución del alumno es correcta funcionalmente, es decir, si como objeto de proceso de información funciona correctamente.

En resumen, el sistema presentado es capaz de recoger código desarrollado por el alumno para después compilarlo y ejecutarlo junto con código desarrollado por el profesor. De esta forma, el profesor diseña el código de evaluación de la práctica, el alumno inserta su solución, y el sistema compila y ejecuta el programa resultante, obteniendo y almacenando la nota del alumno. Esta operación se realiza para cada una de las soluciones propuestas por los alumnos de un grupo determinado que está realizando una práctica concreta de una asignatura definida y empleando un lenguaje de programación específico.

2.3.1. Ventajas del sistema de evaluación

La posibilidad de diseñar un código que se compila junto con el código del alumno ofrece la flexibilidad necesaria para personalizar prácticas y para poder evaluar el trabajo de cualquier disciplina con una base matemática o algorítmica. Aunque a primera vista cabe pensar que este sistema se restringe a asignaturas relacionadas con la programación y la informática, nada más lejos de la realidad.

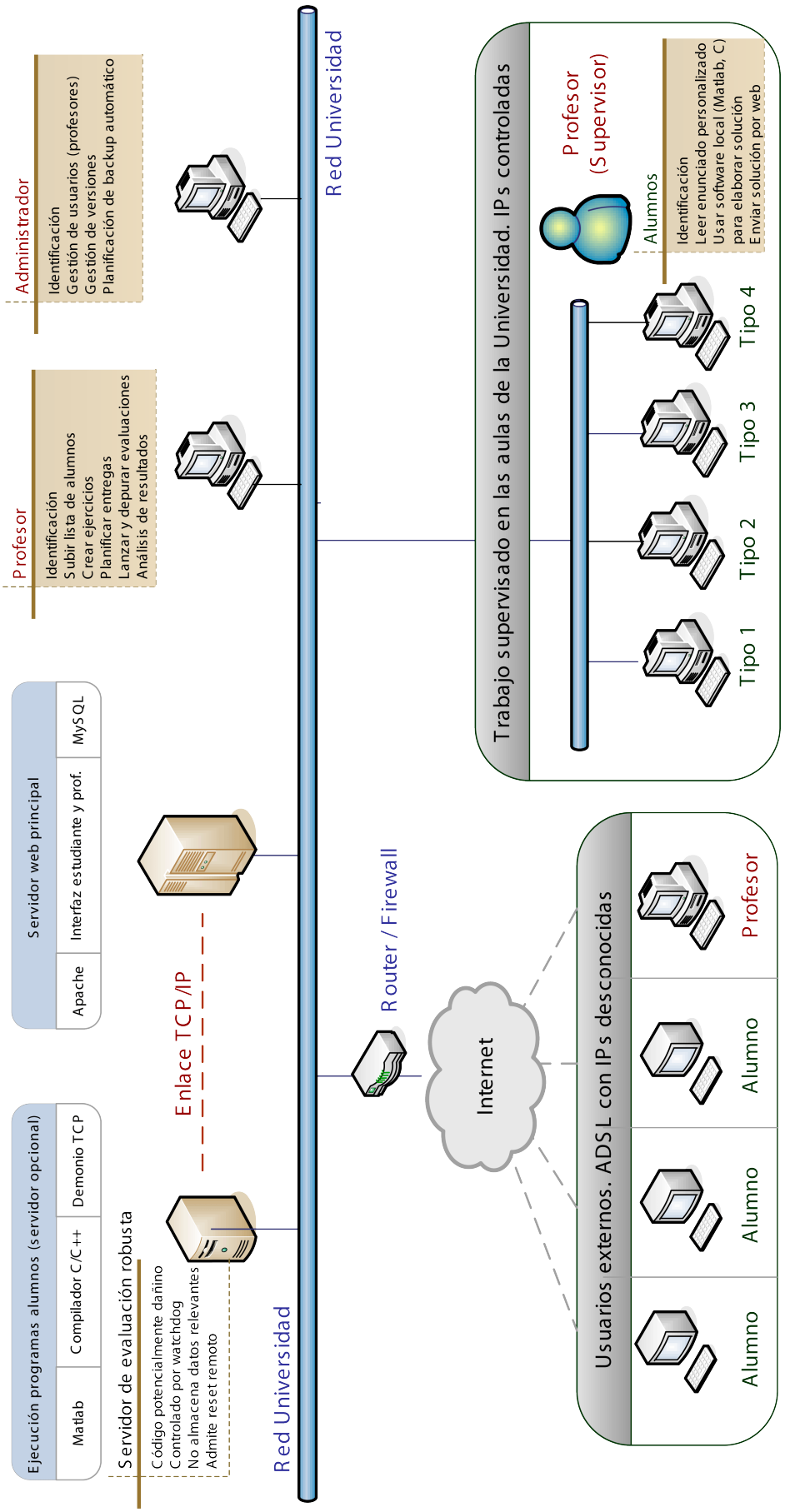

Figura 2.4: Arquitectura del sistema Goodle GMS

En el caso de un problema de análisis, la respuesta del alumno puede consistir en la definición de unas variables que almacenan los resultados de un determinado problema. El programa desarrollado por el profesor puede calcular los resultados que deberían obtenerse para el conjunto de parámetros personalizados del alumno y compararlo con la solución entregada.

En el caso de problemas de diseño, es posible incluso comprobar si la solución presentada por el alumno cumple o no un conjunto de restricciones mediante simulación (por ejemplo en problemas de diseño de controladores para sistemas no lineales).

En los casos de estudio se ilustrará cómo se ha empleado intensivamente la herramienta en asignaturas de fuerte componente matemático y teórico, como la Teoría de Sistemas. En la figura 2.1 se muestra la estructura lógica del sistema de evaluación en el caso en el que se emplee MATLAB (lenguaje matemático, de cálculo numérico y simulación empleado en numerosos campos de ingeniería).

El proceso de corrección es instantáneo y se puede repetir cuantas veces se desee. Es particularmente útil el hecho de que se puede cambiar el código de corrección en línea para determinar por ejemplo la sensibilidad de las notas con un margen de error α. Este hecho también facilita el desarrollo del código de corrección. Además, se ha diseñado un procedimiento robusto de evaluación, por un lado prohibiendo el uso de funciones potencialmente peligrosas para la integridad del servidor, y por otro lado, detectando posibles errores y bucles infinitos en el código del alumno.

Respecto a las dificultades encontradas, cabe reseñar que el servidor desarrollado proporciona una nota numérica que no tiene en cuenta posibles errores tipográficos, confusión en los parámetros u otros factores que puedan adulterar los resultados. Esto significa que una cierta información se pierde en el proceso de evaluación. Sin embargo es una herramienta muy adecuada para evaluar a una gran cantidad de alumnos bajo un único criterio. Esto es muy difícil de conseguir con un método tradicional de evaluación.

2.3.2. Evaluación competitiva

Existe la posibilidad de calificar el trabajo del alumno no por su calidad individual, sino tras compararlo con el resto. Este paradigma se ilustra en la figura 2.5 y funciona del siguiente modo: se ejecuta el código del alumno y el código evaluador del profesor, obteniéndose una figura de mérito que se almacena en la base de datos como nota provisional. En un problema de diseño de controladores, esta figura podría ser el tiempo de subida en bucle cerrado. Una vez calculada la nota provisional para todos los alumnos, el sistema filtra los resultados inválidos, ordena las calificaciones y calcula la nota final para cada alumno según su posición en la lista ordenada, dentro de un intervalo de notas configurado por el profesor. Esta funcionalidad aparece como una opción en el interfaz de creación de prácticas.

2.3.3. Protección frente al fraude

Una dificultad del sistema es la posibilidad de ayuda no autorizada de alumnos más aventajados al resto. Los riesgos son mayores cuando el trabajo se realiza en casa del estudiante y sin control del profesorado. Se trata de una limitación inherente al aprendizaje basado en pro-

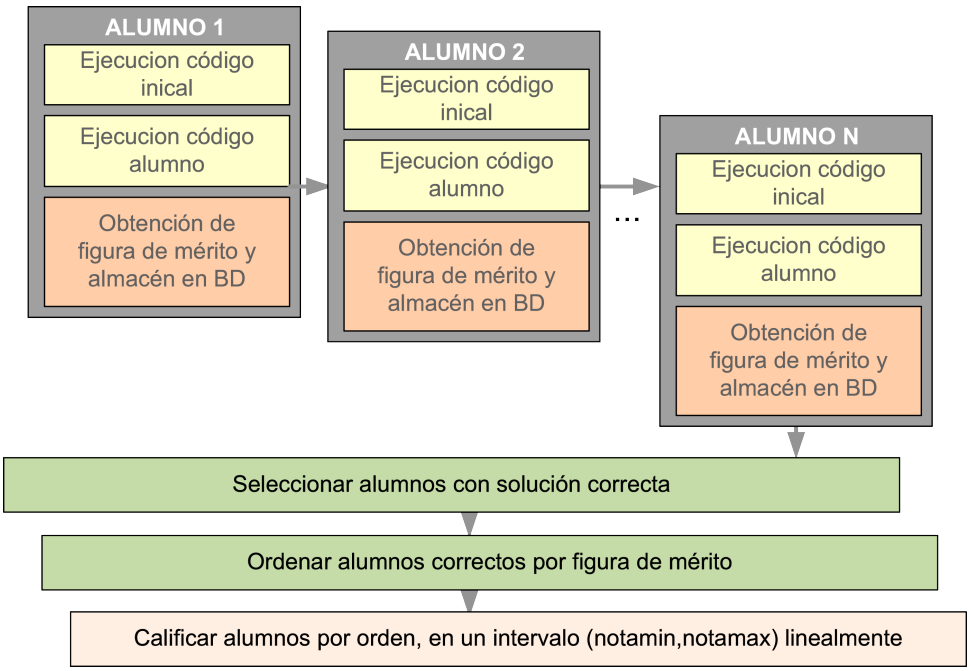

Figura 2.5: Funcionamiento del sistema de evaluación competitiva

yectos, y es común a las herramientas de e-learning mencionadas. Nuestra aplicación, mitiga este riesgo mediante la personalización de ejercicios, como es el caso de la parametrización individual de las funciones de transferencia $G(s)$ del apartado 2.6.2. Dicha parametrización es posible gracias al acceso al número de matrícula o ID del alumno desde el Código Previo y el Código Evaluador del Profesor (ver figura 2.4). A este mecanismo se añade la posibilidad de supervisar el trabajo en el aula, según se ilustra en la figura 2.4. En este escenario el sistema permite filtrar los envíos en función de la dirección IP de origen, garantizando la presencia de los alumnos en las aulas, mientras que el profesor se encarga de verificar la identidad del alumno, que se muestra en pantalla en todo momento.

En las siguientes secciones se describe la arquitectura del sistema junto con una breve descripción de los interfaces de usuario.

2.4. Arquitectura del sistema

La herramienta consta de dos procesos independientes, un programa de ordenador, que se ejecuta en un servidor web dedicado situado en la Escuela Técnica Superior de Ingeniería, y un servidor independiente no visible desde la red, de apoyo a la evaluación automática (aunque, a costa de una ligera pérdida de seguridad, ambos procesos se pueden unificar en una única máquina). El hardware, sistema operativo y aplicaciones empleadas son:

- Servidor de rack HP Proliant DL-1400, 2 GB de memoria RAM, 160Gb de disco duro.
- Sistema operativo Windows 2003 Server R2.
- Apache 2.2.6, PHP 5 y MySQL Server 5.0.

- MATLAB 7.3.
- Dev-C++.

Los usuarios (profesores y alumnos) acceden mediante un navegador convencional de Internet. En la figura 2.4 se ilustra la arquitectura de red del sistema. Los alumnos se encuentran en las aulas de prácticas de la escuela o facultad (en el caso en que se desee controlar la identidad del alumno), o en el exterior.

Respecto de las tecnologías software empleadas, destacamos el empleo de MATLAB. Como apoyo a la docencia, se emplea principalmente en el análisis de sistemas, la síntesis de controladores (con el soporte del *Control System Toolbox*), y dentro de las asignaturas de primeros cursos de Informática y Programación. Como herramienta de realización de las prácticas su efectividad es innegable y a menudo se emplea el propio MATLAB para evaluar la validez de las soluciones de los alumnos.

Además de MATLAB, en el servidor se ofrece la posibilidad de realizar prácticas basadas en el lenguaje de programación C, usando el compilador de libre distribución Dev-C++. Cabe destacar, que es posible desarrollar módulos que permitan la realización de prácticas para cualquier otro lenguaje de programación.

Para desarrollar el servidor se ha utilizado ActiveX. ActiveX emplea el modelo de objetos distribuidos COM de Microsoft para la automatización de aplicaciones. En el presente proyecto se utiliza esta tecnología de manera similar a (Zuluaga et al., 2005) con el fin de que el servidor web pueda realizar operaciones en el espacio de trabajo de MATLAB y compilar código C a través de llamadas a Dev-C++.

La implementación software del servidor se basa en las siguientes tecnologías

- Script de servidor PHP (PHP, 2014). Herramienta de software libre que permite la ejecución de programas en un servidor web en respuesta a peticiones HTTP emitidas por un navegador.
- Servidor de base de datos MySQL (MySQL, 2014). Servidor en red de base de datos relacional para la gestión de los alumnos, las bibliotecas de prácticas y las calificaciones.
- Apache (Apache Open Software Foundation, 2014). Servidor web que atiende a las consultas HTTP de los navegadores y genera las respuestas a través de la ejecución de scripts PHP.

Los servicios Apache–PHP–MySQL forman la arquitectura de servidores web dinámicos más empleada actualmente en el mundo (65 % de penetración frente al 35 % de ASP de Microsoft) y es reconocido por su estabilidad y portabilidad a distintos sistemas operativos.

2.4.1. Interfaz del alumno.

El flujo de trabajo del alumno aparece esquematizado en la parte derecha de la figura 2.1. Inicialmente, el alumno debe identificarse a través de la página web del servidor de docencia, **http://bono.us.es/sdocencia**, mediante nombre de usuario y clave (figura 2.6).

Figura 2.6: Ventana de entrada al sistema

Figura 2.7: Ventana de selección de trabajos

A continuación se muestra una lista con los trabajos pendientes de entrega (que pueden pertenecer a distintas asignaturas). La lista depende de los grupos en los que aparece dicho alumno y las asignaciones (asociaciones) de prácticas que estén activas en cada uno de estos grupos (figura 2.7).

El alumno seleccionará uno de estos trabajos, y a continuación se le mostrará una página con un enunciado en PDF, y un campo de texto sobre HTML para incorporar su solución al problema (figura 2.8). Dicha solución debe estar escrita usando una sintaxis compatible con el lenguaje que se emplee para corregir las prácticas (C, MATLAB,...) y cumplir unas normas unívocamente definidas en el enunciado de la práctica.

Un ejemplo práctico sería la realización de una función en lenguaje C que determine si una matriz de enteros dada es un Sudoku. La solución presentada por el alumno se muestra en la figura 2.8.

Figura 2.8: Ventana de envío de soluciones.

2.4.2. Interfaz del profesor.

El flujo de trabajo del profesor se muestra en la figura 2.1 (izquierda). El interfaz consiste en un menú que permite introducir todos los parámetros para que se lleven a cabo las prácticas, y una serie de opciones para calcular la evaluación y mostrar los resultados. Las opciones principales son:

- Crear nuevo grupo. Para dar de alta nuevos alumnos, grupos y asignaturas.
- Listar alumnos y grupos.
- Crear prácticas, definiendo el programa evaluador y una serie de opciones (ver figura 2.9).
- Listar y modificar prácticas.
- Lanzar evaluación fuera de línea. Realiza la conexión con MATLAB o Dev-C++ y evalúa todos los trabajos entregados correspondientes a una asignatura seleccionada.
- Visualizar calificaciones. En distintos formatos, permite la descarga de un fichero Excel que puede ser complementado con otro tipo de calificaciones. Además, permite el acceso al código entregado por cada alumno y realizar modificaciones sobre él (por ejemplo, para recuperar errores de sintaxis).

En todo momento, el profesor puede volver a lanzar evaluaciones de un grupo. Si se observa que las calificaciones no son coherentes o se desea mejorar el proceso, se podrá optar por refinar el programa que calcula la nota o revisar y en su caso corregir a mano los errores particulares de los alumnos, para finalmente volver a lanzar la evaluación.

El sistema es también capaz de rechazar aquellos envíos con errores de sintaxis, o que usan funciones potencialmente peligrosas para la integridad del servidor. La arquitectura del sistema ha sido desarrollada para permitir que se asignen diversos ejercicios en cada sesión con el fin de evitar el fraude.

Figura 2.9: Ventana de creación y configuración de prácticas

2.5. Directrices para la elaboración de una secuencia de evaluación automática

En esta sección se presenta la estructura básica del código de evaluación que un profesor debe diseñar con el fin de utilizar la aplicación propuesta. Aunque en general cada ejercicio es único, la mayoría de los ejercicios pertenecen a uno de los paradigmas de evaluación que se han presentado en la Sección 2.2. A continuación revisamos estos casos.

2.5.1. Coincidencia exacta en el resultado

La figura 2.2 muestra el diagrama de flujo del código de evaluación para el paradigma de coincidencia exacta en el resultado. Los pasos del algoritmo son los siguientes:

- **Código de inicialización**. Su objetivo es generar las variables en el espacio de trabajo de MATLAB (workspace) previas a la ejecución del código del alumno. Al ser variables personalizadas para cada alumno el algoritmo accede a la base de datos para recuperar toda la información necesaria.

- **Ejecutar la solución del alumno**. En este paradigma, la solución proporcionada por el alumno se compone principalmente de un conjunto de definiciones de variables utilizando la sintaxis apropiada. Goodle GMS debe ser capaz de ejecutar o incluir estas líneas de código en una manera apropiada en función del lenguaje utilizado.

- **Comprobar la coherencia de la solución**. Aunque la aplicación propuesta implementa un filtro de sintaxis que rechaza las soluciones que no son correctas desde un punto de vista sintáctico, las soluciones todavía pueden contener errores. La robustez del código de evaluación respecto a estos errores en las soluciones de los alumnos es importante con el fin de proporcionar una puntuación precisa al trabajo de los alumnos. Si un error

ocurre, el código de evaluación no puede proporcionar una puntuación y el profesor debe evaluar personalmente ese alumno. Muchos errores se corrigen implementando un control de coherencia del nombre, la dimensión y límites de las diferentes variables que la solución del alumno puede crear. Si una variable no existe, hay un error de tipos o la respuesta está fuera de los límites se sustituye por una respuesta incorrecta del tipo apropiado. La robustez mejora sustancialmente si la secuencia de comandos se incluye en una estructura de manejo de excepciones *try-catch*, disponible en la mayoría de los lenguajes de programación modernos.

- **Generar las soluciones correctas**. Usando la información recuperada de la base de datos, el algoritmo debe generar las soluciones correctas para el ejercicio y los parámetros correspondientes al alumno en particular. Para este fin, el profesor aprovecha la posibilidad de ejecutar complejas operaciones matemáticas usando MATLAB, C o cualquier otro lenguaje implementado en la aplicación. Las soluciones se almacenan en un conjunto diferente de variables de las generadas por el alumno.

- **Evaluar**. Para cada respuesta, el código de evaluación compara el valor de las variables creadas por el alumno y el valor correcto que genera el código de evaluación. Si el error es inferior a un umbral, la puntuación se incrementa en un valor determinado (diferentes preguntas pueden tener diferentes ponderaciones). En algunos casos también es importante generar un comentario de texto para proporcionar información alfanumérica para el alumno y el profesor.

Tenga en cuenta que este código se puede generar de forma automática (a través de plantillas) para una amplia clase de problemas, en particular, los pasos de comprobación de la coherencia y la comparación de las soluciones son independiente del tipo de ejercicio. La parte del algoritmo que tiene que ser diseñada de forma específica es el código que genera las soluciones correctas en función del ejercicio y los parámetros.

2.5.2. Evaluación basada en el rendimiento

La figura 2.2 muestra el diagrama de flujo del código de evaluación para el paradigma de evaluación basada en el rendimiento. Los pasos del algoritmo son similares al caso anterior, sin embargo en este caso, en vez de generar soluciones correctas, el código de evaluación debe correr una serie de simulaciones para comprobar el diseño del alumno.

- Código de inicialización.

- Ejecutar la solución del alumno.

- Comprobar la consistencia de la solución.

- Análisis de la solución de los alumnos. En este paso se ejecutan un conjunto de simulaciones basadas en la solución del alumno. La salida de estas simulaciones es analizada, y se generan un conjunto de variables con varias medidas de rendimiento.

- Evaluar. Para cada medida de rendimiento, el evaluador comprueba si se satisfacen las especificaciones de diseño y actualiza la puntuación y los comentarios apropiadamente. Observe que es posible proporcionar una puntuación a partir de una única medida de desempeño.

Parte de este algoritmo se puede generar de forma automática para una amplia clase de problemas, por ejemplo, ejercicios en los que la puntuación depende de la cantidad de especificaciones (es decir, medidas de desempeño que satisfacen condiciones) cumplidas por el diseño del alumno.

En general, la principal diferencia entre la coincidencia en el resultado exacto y la evaluación basada en el rendimiento se encuentra en el análisis de la solución del alumno. Cómo diseñar las entradas de las simulaciones y la forma de medir los diferentes índices de rendimiento dependen de cada ejercicio en particular.

2.5.3. Diseño de algoritmos

En este paradigma de evaluación el alumno debe diseñar un algoritmo para resolver un problema utilizando un lenguaje de programación determinado. Esto puede conducir a una amplia gama de ejercicios. Dedicamos esta sección a los ejercicios en los que el algoritmo debe implementar una función con especificaciones de entrada / salida específicas. La aplicación presentada es capaz de incluir la solución aportada por el alumno para que el evaluador puede llamar a las funciones desarrolladas por los alumnos varias veces tanto para MATLAB y C. Para ello, el código del alumno debe satisfacer varios requisitos de diseño en función del ejercicio, por ejemplo las funciones C deben implementar un prototipo particular. La figura 2.2 muestra el diagrama de flujo del código de evaluación para el paradigma de diseño de algoritmos.

- Código de inicialización.
- Incluir/compilar la solución del alumno.
- Incluir/compilar la solución del profesor.
- Análisis de la solución de los alumnos. Por cada función, se aplican diferentes entradas tanto a la solución del profesor como a la del alumno. Las salidas resultantes se almacenan en un conjunto de variables.
- Evaluar. Para cada respuesta, el código evaluador compara la salida de las variables creadas por el alumno y el valor correcto generado por el código de evaluación. Este paso es similar al paso de evaluación en el paradigma de coincidencia en el resultado exacto.

El esquema propuesto se basa en la idea de que los alumnos proporcionan una solución que utiliza la sintaxis de un lenguaje de programación determinado que implica que el programa resultante puede fallar. La aplicación propuesta ha sido diseñada con el fin de ser robusto con respecto a los errores del código del alumno. Si el programa de evaluación falla (por ejemplo, cuelgues detectados a través de tiempos de espera), el servidor debe ser capaz de detectar

este error y simplemente anotarlo en la base de datos de manera que el profesor sepa que hay un error de programación en la solución particular del alumno. Además, es importante que estos fallos no detengan el resto de las evaluaciones. Esto es muy importante cuando un gran número de alumnos están siendo evaluados.

2.6. Casos prácticos

En esta sección se presentan los resultados de utilizar el sistema de evaluación en dos cursos, uno de teoría de sistemas y otro de fundamentos de informática. Para los casos prácticos presentados, el procedimiento de evaluación se realizó en menos de tres minutos.

2.6.1. Caso práctico 1: Fundamentos de informática.

Una situación donde la automatización de la evaluación resulta efectiva es en las asignaturas de introducción a la informática y la programación, que en numerosas escuelas se adscriben al área de Automática. En este contexto nos centraremos en las asignaturas en las que se imparte programación en entorno MATLAB (como es el caso de la asignatura Informática de Ingeniería Aeronáutica en la Universidad de Sevilla y las asignaturas Informática y Métodos numéricos en ingeniería en la Universidad Loyola Andalucía) y en lenguaje C (Fundamentos de Informática de Ingeniería Industrial y de Ingeniería Química). El esquema de evaluación de un trabajo de programación se basa en la comprobación de que una función programada por un alumno realiza el proceso especificado por el profesor en forma de "caja negra". Para ello se plantea el siguiente esquema:

- El profesor propone una función a programar por el alumno, especificando sus parámetros de entrada y salida, y el procesado que ha de hacerse con los mismos. Ejemplo: cálculo de la media de un vector que se ha pasado como parámetro.
- El profesor implementa la función o algoritmo que resuelve el problema, que denominaremos S1.
- El alumno implementa su propia versión de su solución, S2. Esta función debe contener un encabezado estándar definido por el profesor con el fin de que la llamada y el paso de parámetros sean inequívocos.
- El profesor ejecuta un programa que genera aleatoriamente unos datos de entrada (en el ejemplo, el vector cuya media se desea calcular). El programa llama a S1 y S2 con dichos datos y compara los resultados.
- Se genera una calificación en función de la coincidencia de resultados.

Nótese que para la evaluación puede que no haga falta que el profesor implemente la solución del problema. Basta con que pruebe con unos valores de entrada cuya salida es conocida. Esto es especialmente cierto en el caso de la media del vector que se puede calcular a mano o con una sola linea de MATLAB.

A continuación se presenta el enunciado de la octava práctica que fue realizada en el curso académico 2008-2009, en la asignatura de Fundamentos de Informática de 1^o de Ingeniería Industrial en la Universidad de Sevilla. Dicha asignatura es troncal y dicho curso constaba de 615 alumnos divididos en 12 grupos diferentes. En esta asignatura se realizaron 5 prácticas diferentes gestionadas a través del servidor. Las prácticas constan de una serie de funciones que el alumno ha de diseñar y codificar en lenguaje C. Al final de la clase de laboratorio el alumno envía el código fuente diseñado a través de la aplicación.

Enunciado de ejemplo

Determine si una matriz de enteros 9x9 dada, es un *Sudoku*. *Sudoku* es un pasatiempo matemático, donde hay que rellenar una matriz 9x9 con enteros del 1 al 9. La matriz está divida a su vez en 9 submatrices 3x3. Dada una matriz, se dice que ésta es un *Sudoku* si para cada posición de la matriz, el número correspondiente no se repite ni en la misma fila, ni en la misma columna ni en la misma submatriz 3x3.

- Cuestión 1: Realice una función tal que dada una matriz y una posición específica (fila y columna), compruebe si el número en esa posición de la matriz se repite en la misma fila. La función devuelve 0 si hay repetición. Si no, devuelve 1.

```
int sud_row(int * M, int i, int j);
```

- Cuestión 2: Realice una función tal que dada una matriz y una posición específica (fila y columna), compruebe si el número en esa posición de la matriz se repite en la misma columna. La función devuelve 0 si hay repetición. Si no, devuelve 1.

```
int sud_col(int * M, int i, int j);
```

- Cuestión 3: Realice una función tal que dada una matriz y una posición específica (fila y columna), compruebe si el número en esa posición de la matriz se repite en la submatriz 3x3 a la que pertenece. La función devuelve 0 si hay repetición. Si no, devuelve 1.

```
int sud_sub(int * M, int i, int j);
```

- Cuestión 4: Realice una función tal que dada una matriz y una posición específica (fila y columna), compruebe si el número en esa posición de la matriz verifica las tres propiedades anteriores. La función devuelve 1 si las tres propiedades se cumplen. Si no, devuelve 0.

```
int sud_cas(int * M, int i, int j);
```

- Cuestión 5: Realice una función que compruebe si una matriz dada es un *Sudoku*. La función devuelve 1 si la matriz es un Sudoku. Si no, devuelve 0.

```
int sud(int * M);
```

El código de evaluación del profesor para este ejercicio práctico, viene principalmente dado por la siguiente función.

```
int evaluacion(void) {
    int nota=0;
    int sudoku[81] =
               {1,2,3,4,5,6,7,8,9,
                4,5,6,3,2,7,8,9,6,
                7,8,9,3,2,7,8,9,6,
                2,4,5,3,2,7,8,9,6,
                3,4,5,3,2,7,8,9,6,
                5,4,5,3,2,7,8,9,6,
                6,4,5,3,2,7,1,4,3,
                8,4,5,3,2,7,4,4,6,
                9,4,5,3,2,7,7,8,9};

    int sudoku1[81]=
               {7,9,8,6,3,2,5,1,4,
                5,3,1,8,7,4,6,9,2,
                4,6,2,1,9,5,7,8,3,
                2,4,6,5,8,1,9,3,7,
                8,1,7,9,6,3,4,2,5,
                3,5,9,4,2,7,8,6,1,
                6,2,4,3,5,9,1,7,8,
                1,8,3,7,4,6,2,5,9,
                9,7,5,2,1,8,3,4,6};

    /* Cuestion 1 */
    if((sud_row(sudoku,0,0)==1) && (sud_row(sudoku,7,7)==0)) nota++;

    /* Cuestion 2 */
    if((sud_col(sudoku,0,0)==1) && (sud_col(sudoku,7,7)==0)) nota++;

    /* Cuestion 3 */
    if((sud_sub(sudoku,0,0)==1) && (sud_sub(sudoku,7,7)==0)) nota++;

    /* Cuestion 4 */
    if((sud_cas(sudoku,0,0)==1) && (sud_cas(sudoku,7,7)==0)) nota++;

    /* Cuestion 5 */
    if((sud(sudoku)==0) && (sud(sudoku1)==1)) nota++;

    return nota;
}
```

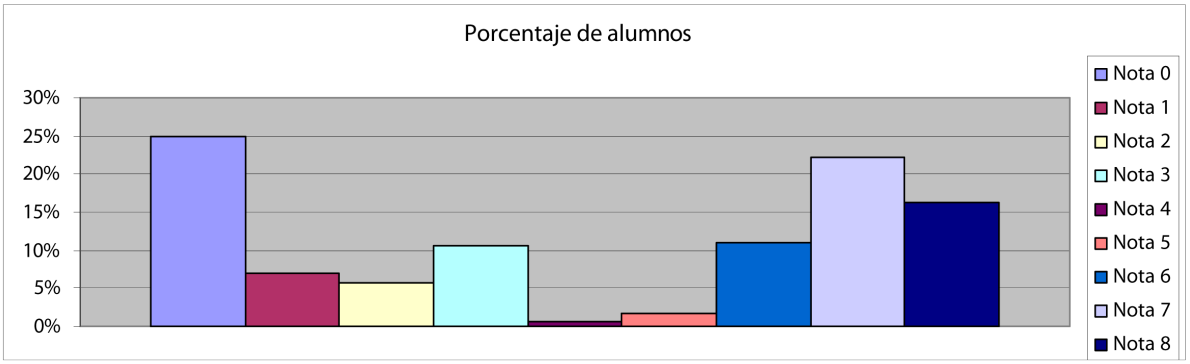

Figura 2.10: Distribución porcentual de las notas obtenidas.

Esta función de evaluación simplemente verifica si cada una de las cinco funciones propuestas al alumno, funciona correctamente con dos matrices diferentes. En caso de que una función sea correcta, la nota se incrementa en un punto.

Calificaciones de los alumnos.

En la figura 2.10 se presenta la distribución porcentual de las notas obtenidas, evaluadas entre 0 y 8. En el ejercicio práctico había tres cuestiones más, que se han omitido por motivos de claridad en la exposición. Al analizar los resultados obtenidos, se puede observar que hay una clara separación entre los ejercicios suspensos (nota de 0 a 3) y los aprobados (nota de 6 a 8). Particularmente, el 48 % de las calificaciones corresponde a ejercicios suspensos. Sin embargo, este porcentaje no es realista en absoluto, ya que el evaluador asigna 0 al ejercicio del alumno cuando el proceso de su evaluación dura más de cinco segundos, lo cual implica que el código de alguna función del alumno es incorrecto (probablemente un bucle infinito). La imposibilidad de continuar la evaluación hace que no se tenga en cuenta la nota de las restantes funciones. Esto se ha solventado sacando notas parciales para cada función[1].

2.6.2. Caso práctico 2: Teoría de sistemas

Otro escenario de docencia donde resulta muy conveniente el empleo de MATLAB sería en la calificación de prácticas de diseño de controladores. En este contexto, se propone el siguiente esquema:

- El sistema propone una función de transferencia G(s) (puede ser individualizada para cada alumno) y una estructura de controlador, por ejemplo PID o red de avance.
- El sistema propone al alumno unas especificaciones a cumplir por el controlador en bucle cerrado: error en régimen permanente, márgenes de fase y ganancia, etc.
- El alumno diseña el controlador empleando los métodos explicados en clase, obteniendo las constantes del controlador (Kp, Td, Ti en el caso de un controlador PID).

[1]Concretamente se programaron varios evaluadores que hacían llamadas selectivas a una u otra función, de manera que el bloqueo de una de ellas no imposibilitara la ejecución de las demás.

- El alumno verifica que el controlador cumple las especificaciones, mediante comandos de simulación ("*lsim*") o con instrucciones específicas del Control System Toolbox, (p. ej. "*margin*"). A continuación, entrega el trabajo.
- El sistema evalúa automáticamente si se cumplen las especificaciones empleando las mismas herramientas de MATLAB (principalmente simulaciones).

A continuación se presenta el enunciado de una de las prácticas que fueron realizadas en el curso académico 2007-2008, en la asignatura de Teoría de sistemas de 2^o de Ingeniería Industrial en la Universidad de Sevilla. Dicha asignatura es troncal y dicho curso constaba de 366 alumnos divididos en cuatro grupos diferentes.

La práctica consta de una serie de ejercicios que el alumno ha de resolver usando Simulink durante la clase de laboratorio. Al final de la clase de laboratorio el alumno entrega una memoria con los resultados. Los parámetros que definen el ejercicio que debe realizar el alumno dependen de su número de DNI. La mayor parte de las respuestas de las prácticas son numéricas (como por ejemplo el tiempo de subida de un sistema o la amplitud de la salida en régimen permanente frente a una entrada escalón). La corrección manual de un gran número de prácticas con resultados numéricos diferentes para cada una de ellas conllevaba una gran carga de trabajo. Además, tan sólo se distinguían cuatro posibles notas, *suspenso*, *mal*, *regular* y *bien*. El uso de la aplicación en esta asignatura proporciona una valoración más precisa del trabajo del alumno.

Enunciado de la práctica.

Considere el siguiente sistema lineal:

$$Y(s) = \frac{k}{1+\tau s} U(s)$$

dónde los parámetros k y τ toman los siguientes valores para cada alumno en función de sus dos últimos dígitos del DNI (siendo D7 el penúltimo y D8 el último por la derecha):

- $k = (D7 + 1) \times 10$
- $\tau = D8 + 1$

Evalúe utilizando Simulink la respuesta en régimen permanente del sistema frente a una serie de señales de entrada senoidales de diferente amplitud y frecuencia, y rellene la tabla 2.1.

Cada respuesta correspondiente a una cuestión de trabajo en el centro de cálculo tiene asignado un nombre. En este módulo, se han agrupado las respuestas en vectores.

Ejemplo: La respuesta **p1** corresponde a la frecuencia de entrada, amplitud de salida, desfase temporal y ángulo de desfase de la señal de salida correspondiente a una frecuencia de entrada de $0.01/\tau$.

Para las respuestas **p1...p7** se pide escribir una línea con el siguiente formato:

```
nombre = [frec amp des_temp des_ang];
```

ω (rad/s)	Y_0 (dB)	Δ (s)	ϕ (rad)
p1 $0.01/\tau=$			
p2 $0.1/\tau=$			
p3 $0.3/\tau=$			
p4 $1/\tau=$			
p5 $3/\tau=$			
p6 $10/\tau=$			
p7 $100/\tau=$			

Tabla 2.1: Resultados a entregar de la práctica

Ejemplo: Para responder que para una señal de entrada de frecuencia de 0.1 rad/s (correspondiente a $0.3/\tau$) la señal de salida tiene una amplitud de 3dB, un desfase temporal de 2 segundos y un desfase angular de 0.3 radianes tenemos que escribir las siguiente líneas:

```
p3 = [0.1 3 2 0.3];
```

Ejemplo de formulario relleno.

```
p1 = [0.15 10 2 0.3]; p2 = [1.5 10 0.2 0.3];

p3 = [4.5 5.7 0.01 0.3]; p4 = [15 -5 0.01 0.3];

p5 = [45 -10 0.01 0.3]; p6 = [150 -30 0.010.3];

p7 = [1500 -50 0.001 0.3];
```

Para evaluar cada una de las prácticas se ha escrito un código en MATLAB que realiza las siguientes operaciones:

- Genera los parámetros k y τ a partir de el DNI del alumno.
- Genera un conjunto de vectores solución **s1...s7** usando la fórmula explícita de la respuesta frecuencial en régimen permanente de un sistema lineal.
- De cada uno de los elementos los vectores de respuesta se evalúa el error normalizado como
$$ei(j) = \left| \frac{pi(j) - si(j)}{si(j)} \right|$$
- Fijado un porcentaje de error α, la nota entre 0 y 1 de una práctica se evalúa como el numero de respuestas con un margen de error menor de α partido por el número total de respuestas.

Nota	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Alumnos	9	2	2	4	3	11	43	118	60	7

Tabla 2.2: Resultados de la evaluación de la práctica

Calificaciones de los alumnos.

La práctica se corrigió con un margen de error del 10 %. En la tabla 2.2 se muestra la distribución de notas obtenida. Se puede observar que la práctica presenta unas notas bastante bajas. Esto se debe a que, al tratar la práctica del estudio de la respuesta frecuencial de un sistema de primer orden, la medidas tomadas por los alumnos del retraso y la amplitud de señales de alta frecuencia eran poco precisas.

2.7. Encuestas de satisfacción de usuarios

Se ha empleado la plataforma con más de 400 alumnos con resultados diversos en cuanto a calificaciones y mejoras con respecto a años anteriores. En la figura 2.11 se ilustran los resultados de una encuesta masiva realizada entre los alumnos de Fundamentos de Informática.

Por último, en la tabla 2.3 se muestra el resultado de una encuesta realizada entre los profesores involucrados en asignaturas de la Escuela Superior de Ingenieros donde se ha empleado la herramienta. La muestra es pequeña (7 profesores), ya que se restringe al profesorado de 5 asignaturas, aunque de alumnado numeroso. Sin embargo, se puede apreciar un consenso en cuanto a la convenien cia de uso de la herramienta para profesores y alumnos, mientras se atisba cierta desconfianza a la hora de depositar completamente la responsabilidad de la evaluación en el sistema.

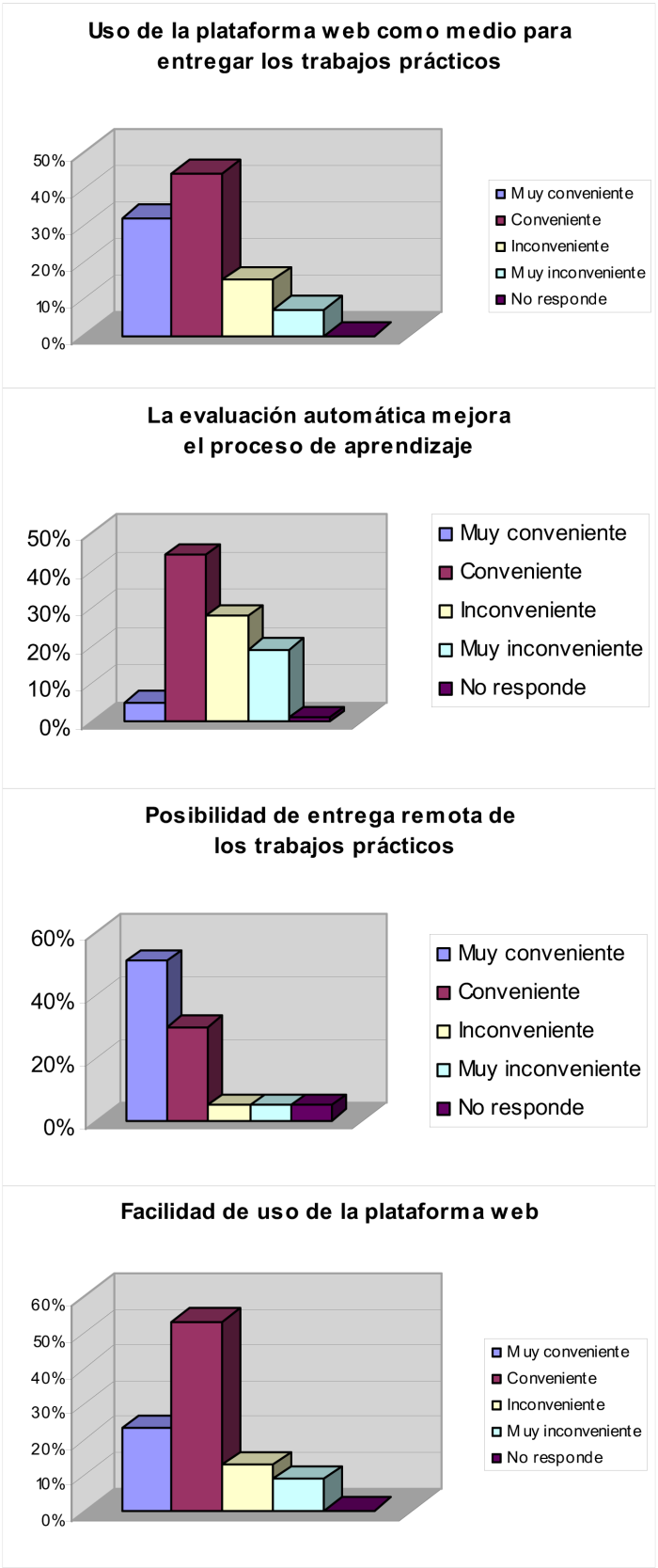

Figura 2.11: Resultado de la encuesta realizada en Fundamentos de Informática

Cuestión	**1**	**2**	**3**	**4**	**5**	**NS/NC**
Utilidad de la herramienta para tu trabajo como profesor (economía de esfuerzo, organización, rapidez, información disponible)	0	0	0	1	6	0
Utilidad de la herramienta para el aprendizaje del alumno	0	0	0	4	3	0
¿Se logra mayor implicación del alumno al emplear la herramienta y su metodología de trabajo?	0	0	1	2	4	0
Calidad técnica de la aplicación (robustez, opciones disponibles)	0	1	1	3	2	0
Calidad del interfaz gráfico (uso amigable, estética, orden en la presentación de datos)	0	1	0	4	2	0
Confianza que le inspira la nota calculada automáticamente	0	1	1	4	1	0
Facilidad de diseño de nuevas prácticas (creación de evaluadores, selección de opciones y filtros)	0	1	1	3	1	1
Facilidad de gestionar el desarrollo de la práctica en el aula del Centro de Cálculo cuando se emplea la herramienta	0	0	1	2	2	2
Facilidad para ejecutar la evaluación automática para calificar un grupo (gestión de bucles infinitos, errores de problemas sintaxis de las entregas)	0	1	1	3	2	0
¿Utilizarías la herramienta, con todos sus filtros y supervisión del profesor, como único medio para calificar su asignatura?	2	1	1	1	2	0
Satisfacción general del alumno con la herramienta (tanto por su comodidad de uso como por la metodología en general)	0	0	2	4	0	1

Tabla 2.3: Resultados de la encuesta al profesorado (Votos totales. 1=Desfavorable, 5=Muy favorable)

Capítulo 3

Tecnología para dotar a los laboratorios virtuales y remotos de funcionalidades de evaluación automática

En este capítulo se presenta el llamado elemento de evaluación automática (EEA), que permite realizar tareas de calificación automática con LVyR. El EEA permite dotar de funcionalidades de calificación automática a aplicaciones preexistentes creadas mediante Easy Java Simulations (EJS). EJS es una herramienta gratuita de desarrollo de aplicaciones interactivas con componentes matemáticos, de análisis y de simulación sin conocimientos avanzados de programación, o en muchos casos sin programación alguna (Esquembre, 2004; Christian, 2007).

3.1. Laboratorios virtuales y remotos con EJS

EJS es una herramienta de software libre desarrollada en Java, diseñada especialmente para la creación de simulaciones por computador en tiempo discreto dotadas de interfaces gráficas interactivas.

EJS se desarrolló originalmente para usuarios con escasos conocimientos de programación. El usuario sólo tiene que conocer el modelo analítico del proceso y diseñar el interfaz gráfico en detalle. La arquitectura de EJS se deriva del paradigma modelo-vista-controlador (MVC), cuya filosofía sostiene que las simulaciones interactivas deben contener tres partes:

- El modelo, que describe el proceso en estudio en función de 1) variables, que registran los diferentes estados posibles del proceso, y 2) relaciones entre estas variables, expresadas como algoritmos.
- El control, que define ciertas acciones que el usuario puede hacer sobre la simulación.
- La vista, que muestra una representación gráfica (realista o esquemática) de los estados del proceso.

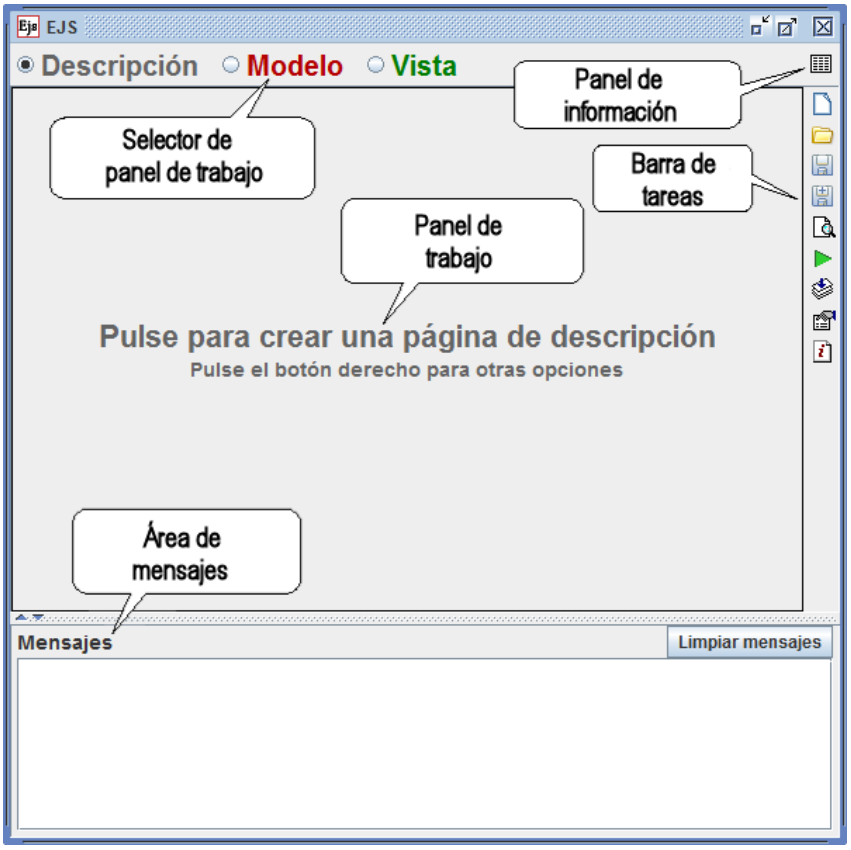

Figura 3.1: Interfaz de usuario EJS

EJS simplifica las tareas eliminando la necesidad de implementar el elemento control del paradigma MVC, definiendo la física del sistema en el modelo, la interacción del usuario en la Vista y enlazando ambas partes de manera transparente y automatizando las funciones de control de simulación (ver figura 3.1).

Por tanto, las aplicaciones se crean en dos pasos: 1) la construcción del modelo a simular usando el mecanismo integrado de simulación de EJS y 2) la construcción de la vista para mostrar para mostrar el estado del modelo y sus reacciones a los cambios hechos por los usuarios. Los laboratorios remotos se construyen mediante EJS y pueden ser distribuidos mediante una arquitectura cliente servidor.

El interfaz de usuario (GUI) del lado del cliente se diseña y construye mediante EJS, mientras que una aplicación de servidor, que actúa como pasarela entre la planta real y el laboratorio, también es necesaria.

Se pueden encontrar ejemplos y más información sobre esta arquitectura en (Dormido, R., Vargas, H., Duro, N., Sánchez, J., Dormido-Canto, S., Farias, G., Esquembre, F., Dormido, S., 2008; De la Torre, L., Sanchez, J.P., Sanchez, J., Dormido, S., Yuste, M., Carreras, C., 2010), y algunos ejemplos de simulaciones EJS se pueden descargar gratuitamente en el siguiente enlace:

http://www.um.es/fem.

La Figura 3.2 muestra la estructura básica de estas aplicaciones, donde un cliente remoto manipula un proceso localizado en el laboratorio a través de una computadora servidora tra-

Figura 3.2: Control remoto y manipulación de un equipo físico a través de Internet.

bajando como capa de comunicación middleware. La realimentación visual del equipo remoto se obtiene mediante una webcam enfocada hacia el equipo real.

3.2. Evaluación automática con Goodle GMS

Con el fin de encajar los laboratorios EJS en la arquitectura Goodle GMS, se han introducido nuevas opciones en el último para un nuevo tipo de ejercicio específico denominado EJS. El lenguaje de base es MATLAB, y la idea principal es la de configurar de manera especial las dos partes principales del código del profesor: código de pre-evaluación y código de post-evaluación.

El código de pre-evaluación específico para EJS consiste en una serie de instrucciones destinadas a generar un conjunto de variables en el espacio de trabajo MATLAB, que serán transmitidas por red a los interfaces EJS de los estudiantes, con anterioridad al uso del laboratorio. Estas variables parametrizan la simulación EJS del alumno, posiblemente con valores únicos para cada uno en función de su DNI u otro ID, razón por la cual es preciso registrarse al inicio.

Tras la ejecución, las variables personalizadas se crean en el espacio de trabajo de MATLAB y se transmiten al cliente EJS tras el registro (login) del estudiante (detallado en una sección posterior). La segunda parte principal de la configuración del ejercicio EJS consiste en el código de evaluación (post-evaluación). Este código ha de diseñarse para calificar al alumno en función de las respuestas recibidas desde la aplicación cliente EJS.

En los ejercicios específicos de EJS, el alumno no proporcionará código fuente puro. En su lugar, al realizar el trabajo sobre el laboratorio, el alumno debe rellenar los campos de texto del interfaz EJS y fijar los valores de los slider bars (controles de tipo barras de desplazamiento). Cuando el alumno termina su tarea, pulsará un botón de envío (Submit), y estos valores se transforman automáticamente en código fuente MATLAB (sentencias de asignación con los valores de los campos de texto). Este código se transmite por red al servidor para su

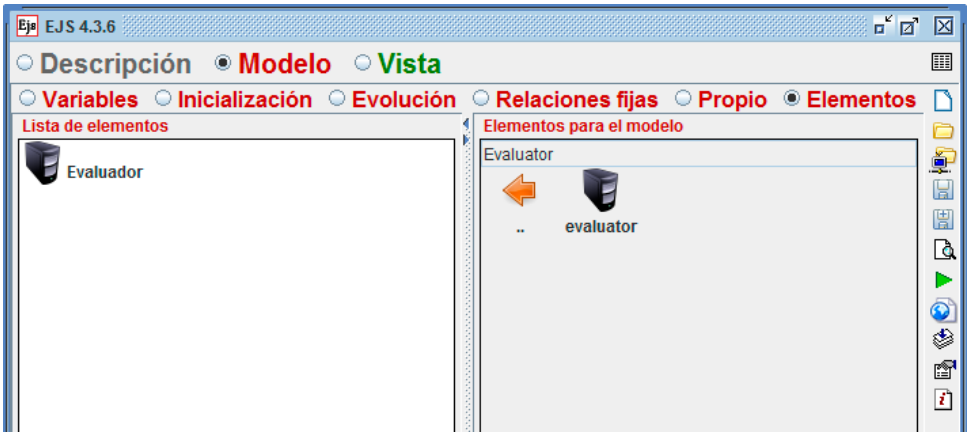

Figura 3.3: El elemento de evaluación automática

almacenamiento en la base de datos de entregas, y para la posterior evaluación automática, que será lanzada por el profesor cuando éste lo desee.

3.3. El elemento de evaluación automática: EEA

El EEA simplifica la calificación de las sesiones de laboratorio realizadas mediante simulaciones EJS. Este elemento puede descargarse en el siguiente enlace:

http://lab.dia.uned.es/evaluator.

El EEA divide la evaluación de los usuarios de laboratorios EJS (estudiantes) en dos pasos: registro (login) y entrega. El paso de registro o login es necesario para adaptar o personalizar la simulación a cada usuario. Ello implica que las variables de la simulación dependen del DNI o número de matrícula del alumno. Por tanto, las simulaciones de los alumnos pueden estar basadas en el mismo sistema físico, pero distintos parámetros (ejemplo, masas, constantes de fricción, etc.).

Los estudiantes usan la simulación personalizada para llevar a cabo un conjunto de tareas o actividades predefinidas (protocolo de actividades). Una vez terminadas las actividades, los alumnos deben transmitir sus soluciones al servidor por el procedimiento descrito para su posterior evaluación. Además, EJS permite a los alumnos realizar todas las simulaciones que deseen en un modo off-line, sin que los resultados tengan que enviarse al servidor.

3.3.1. Cómo usar el EEA

El uso del EEA es muy sencillo. Su filosofía de diseño está orientada a su inclusión en LVyR preexistentes sin apenas modificaciones. Los profesores (creadores de ejercicios) tienen que añadir el EEA a la lista de elementos (plugins) de la simulación en su aplicación EJS (ver figura 3.3). Esto se logra a través de una de las opciones mostradas en el panel Model del editor EJS. Los elementos en EJS son muy simulares a las librerías Java. Una vez que se han importado en una aplicación, un conjunto de métodos queda a la disposición del desarrollador.

A continuación, se configura el elemento. Esto se logra mediante una ventana de diálogo

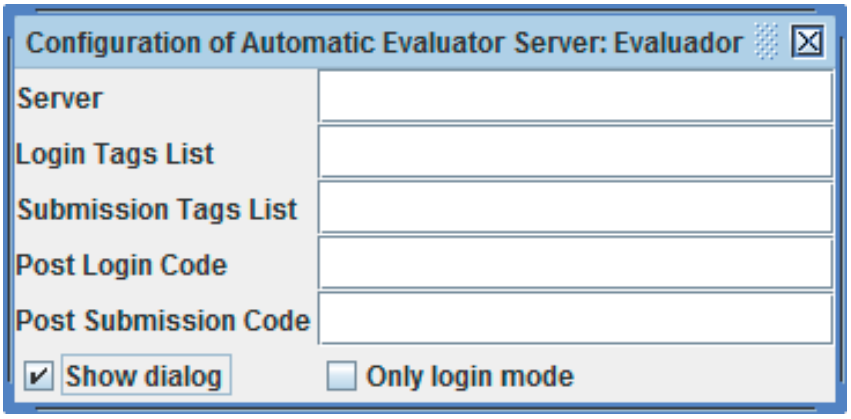

Figura 3.4: Configuración de elemento de evaluación automática

(ver figura 3.4), que aparece al hacer doble click en el Nuevo elemento dentro de la ventana "List of elements".

Para configurar el AEE, el autor debe especificar como mínimo los siguientes datos:

- **Server**: La dirección web del servidor remoto Goodle GMS.
- **Login Tags List**: Una lista de variables que se solicitan al alumno para hacer login en el servidor y que serán enviadas al mismo tras el registro inicial. El carácter doble "##" se emplea para separar las etiquetas que identifican estas variables. El carácter especial "*" indica que una etiqueta particular representa una contraseña. Un ejemplo de esta lista de variables es "dni##*contrasena##grupo##practica"
- **Submission Tags List**: Lista de datos requeridos como resultados del trabajo con el Laboratorio, previos al envío final de los alumnos. Estas etiquetas se traducen en campos de texto a rellenar por el alumno con resultados que se enviarán al servidor para ser transformados en asignaciones MATLAB y ser evaluados. De nuevo, el carácter doble "##" se emplea parta separar las etiquetas de la lista. Un ejemplo de esta lista es "kp##ti"

Basándose en esta sencilla configuración, el EEA crea automáticamente una ventana de diálogo que permite a los usuarios finales registrase al inicio y enviar los resultados al final de protocolo de actividades.

Nótese que ambas listas de variables se organizan como etiquetas (Tags), que se mostrarán junto a un campo de texto por rellenar durante la ejecución del applet EJS en los instantes de registro y envío final. Un botón de OK confirma la validación de los datos y el envío, tras lo cual se muestra la respuesta del servidor en el campo Server Response. De este modo se puede personalizar el diálogo del EEA para interactuar con cualquier tipo de servidor, no sólo con Goodle GMS.

Aparecen en este diálogo de configuración los siguientes parámetros adicionales:

- **Post login code**: opcional, sirve para definir el nombre del método Java incluido en el applet EJS al que hay que llamar tras recibir los parámetros iniciales desde el servidor

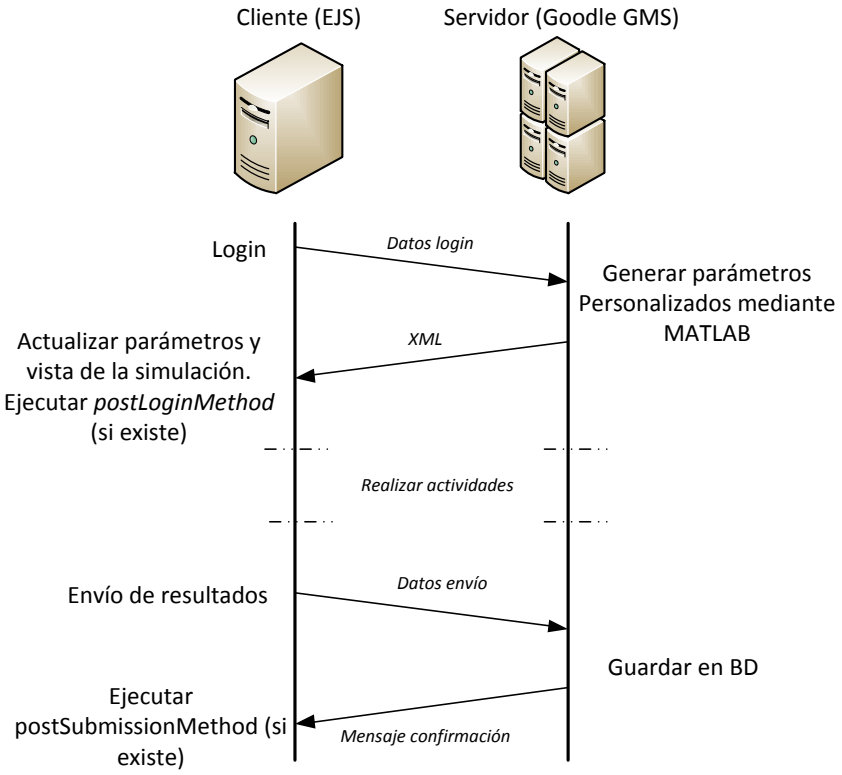

Figura 3.5: Secuencia del protocolo de comunicaciones cliente-servidor del EEA

Goodle, para tareas específicas de inicialización. Mediante esta función se logran simulaciones muy dependientes del ID del alumno, como el caso de sistemas dinámicos cuyo orden no es fijo para todo el alumnado, o el de variaciones en el aspecto del interfaz.

- **Post submission code**: opcional, método que se ejecutará tras el envío con éxito de los resultados de la práctica, para diversos usos que permitan al alumno dar por cerrada la tarea (por ejemplo generación de un comprobante de entrega).
- **Show dialog**: Si se selecciona esta opción, la caja de diálogo de registro se mostrará en forma modal al inicio del applet (funcionamiento por defecto).

3.3.2. Fundamentos de la evaluación automática

El sistema se ejecuta en una arquitectura cliente servidor. La figura 3.5 muestra la secuencia del protocolo de comunicaciones entre el Laboratorio EJS y el servidor de evaluación. La interacción se divide en las siguientes fases: Login, Realización del protocolo de actividades, Entrega de Resultados y Evaluación.

Login: esta fase lanza una consulta HTTP al servidor, con los datos del login del alumno y un identificador del ejercicio del que se trate. Si el registro es correcto (existe el alumno y la práctica especificada), el servidor responde con un paquete de datos XML, o con un

mensaje de error en caso contrario. El mensaje XML contiene un conjunto de pares etiqueta-valor generados en MATLAB mediante el código de pre-evaluación específico de la práctica, posiblemente dependiente del ID del alumno.

Estas asignaciones XML se reciben por el EEA y se emplean para personalizar la simulación EJS (los pares etiqueta-valor se transforman en valores iniciales de los elementos del interfaz gráfico, por ejemplo campos de texto, gracias a la coincidencia con las etiquetas dadas a dichos elementos al diseñar el applet). Los autores pueden usar esta personalización no sólo para definir valores iniciales de simulación, sino también para modificar elementos del interfaz gráfico.

La información codificada en el mensaje XML contiene los valores iniciales y parámetros específicos de la simulación. El EEA usa la API reflection para actualizar las variables del applet y refrescar la vista de la simulación. Existen tres tipos de variables en este intercambio: double, array de doubles, y string. Con este tercer caso se pueden incluir expresiones matemáticas simbólicas en el intercambio. Para cada variable del mensaje XML, el EEA busca una variable del mismo nombre en el applet EJS (como parte del interfaz o como variable interna), y si encuentra la coincidencia, fija la variable local al valor recibido.

Tras un login con éxito, la pestaña submission se activa, y se ejecuta el post login code si se hubiera definido.

Realización del protocolo de actividades: Aquí los estudiantes deben usar localmente la simulación personalizada para realizar las tareas y problemas a resolver descritas en el protocolo de actividades. En este paso no hay comunicación con el servidor.

Entrega: Una vez terminadas las actividades, los alumnos deben enviar sus respuestas para ser evaluadas. Para ello sólo deben rellenar los campos de la pestaña Submission con los valores calculados (opcionalmente) y pulsar OK. Estos campos y el resto de los campos de entrada del interfaz EJS (texto, slider bars, etc.) se codifican por el EEA y se empaquetan en un mensaje XML con pares etiqueta-valor recogidos del interfaz gráfico. Una vez recibido el mensaje en el servidor, los pares se transforman automáticamente en asignaciones MATLAB como se ilustra en el siguiente ejemplo:

```
<variables>
    <a>10.0</a>
    <f1>cos(x1)+x2^2</f1>
</variables>
```

Nótese la posibilidad de incluir expresiones simbólicas en el intercambio de datos, que pueden ser procesadas mediante el MATLAB Symbolic Toolbox en la evaluación. El cliente EJS también contiene un parser capaz de interpretar estas expresiones. Estas características se han empleado para desarrollar un laboratorio virtual simbólico para cursos de control no lineal (Sánchez, C., Gomez-Estern, F. y Muñoz de la Peña, D., 2012) que se describirá en el siguiente capítulo.

Evaluación: Las respuestas enviadas (código MATLAB) se almacenan en el servidor de evaluación automática Goodle. Se organizan en una base de datos que contiene código pre y post evaluación específicos de cada ejercicio. Una vez terminado el plazo de entrega, el profesor lanzará una evaluación de todos los ejercicios entregados para una práctica concreta.

Para cada envío de estudiante, el código de envío se fusiona con los códigos de pre y post evaluación como se mostró en la Figura 2.1, con el fin de obtener automáticamente la nota y almacenarla en la base de datos.

3.4. Aplicación de servomotor en laboratorio virtual

En esta sección presentamos un caso de estudio de un servomotor en un laboratorio virtual diseñado a partir del elemento de evaluación automática. Este laboratorio virtual se ha utilizado en dos diferentes cursos básicos de control, primero en la Universidad de Almería (UAL, España) y después en Potificia Universidad Católica de Valparaíso (PUCV, Chile).

La aplicación de servomotor se basa en un laboratorio remoto desarrollado bajo el proyecto Automatl@bs de la Universidad Española de Educación a Distancia (H. Vargas, J. Sánchez, N. Duro, R. Dormido, S. Dormido-Canto, G. Farias, S. Dormido, F. Esquembre, C. Salzmann, D. Gillet, 2008). En este laboratorio, los alumnos pueden controlar en lazo abierto y cerrado le velocidad de giro de un motor de corriente continua y su posición. El motor se modela como un sistema lineal de primer orden con una zona muerta y saturación de entrada que relaciona la tensión de entrada con la velocidad de rotación del motor. El laboratorio implementa un control de bucle abierto en el que la tensión se ajusta manualmente y tanto la posición como la velocidad se controlan en bucle cerrado usando un controlador proporcional integral derivativo (PID).

La interfaz empleada que se muestra en la figura 3.6 se divide en diferentes secciones. En la esquina superior izquierda se muestra un círculo conectado al eje del motor. Este círculo gira durante la simulación y, además, es posible cambiar la referencia de bucle cerrado de una manera interactiva. Debajo de la imagen, están los paneles de control del laboratorio, incluidos los controles de reproducción de la simulación, que permiten iniciar, pausar y reiniciar la simulación en cualquier momento, un botón selector para cambiar entre control de velocidad y posición (que modifica los gráficos mostrados), los parámetros PID y dos controles para establecer la referencia (en la simulación en bucle cerrado) o la entrada de tensión (en la simulación en bucle abierto). Con el fin de elegir la simulación en bucle abierto o bucle cerrado, el usuario tiene que acceder a los botones del menú superior.

En el lado derecho hay dos gráficos que muestran las simulaciones de velocidad y tensión o posición y tensión. Es importante remarcar que el alumno debe tomar una serie de medidas con el fin de llevar a cabo el ejercicio. Para este fin, los gráficos disponen de un indicador que muestra los valores exactos del punto seleccionado. También es posible guardar cualquier experimento de trayectorias para procesar los datos con otras herramientas, como MATLAB o Excel. En este caso particular, el ejercicio fue diseñado de modo que era posible hacer todas las mediciones con el indicador de EJS, teniendo en cuenta su correspondiente precisión.

El laboratorio virtual original de EJS se actualizó con el EEA para proporcionar una personalización y conexión a Goodle GMS. EEA añade estas funcionalidades de manera transparente por lo que es posible actualizar cualquier implementación de EJS de una manera fácil (aunque un diseño integrado ofrece una mejor experiencia de usuario en general). La figura 3.7 muestra la ventana EEA incluido en el laboratorio de servomotor. El alumno debe incluir el ID y la contraseña proporcionada por el profesor a fin de conectarse a Goodle GMS, recuperar los

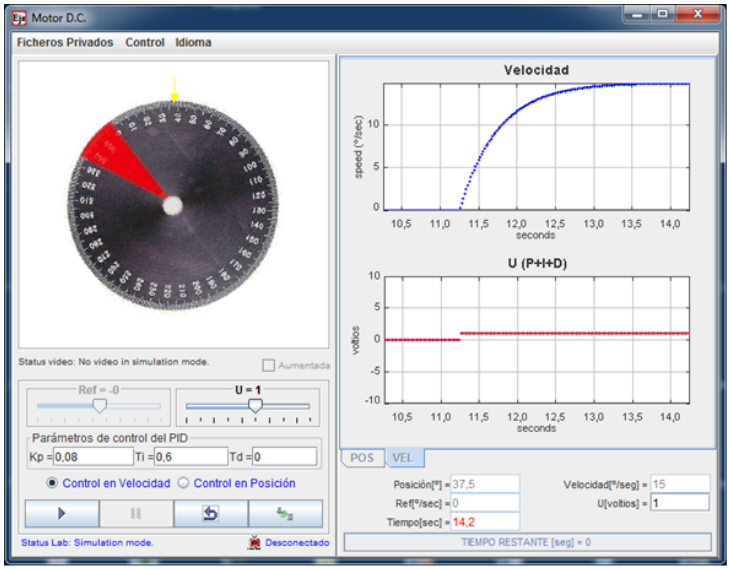

Figura 3.6: Interfaz principal de la aplicación de servomotor del laboratorio virtual

parámetros del motor y enviar la solución del ejercicio. En este caso, el laboratorio recibe de Goodle GMS tras el inicio de sesión los parámetros que definen cada motor, en particular, la ganancia, el tiempo característico y el ancho de banda de la zona muerta (la saturación de tensión es la misma para todos los estudiantes). Los parámetros se obtienen a partir de la identificación del alumno con el código de inicio de sesión.

El ejercicio se define en un documento PDF proporcionado a los alumnos. Este documento describe una serie de experimentos que los estudiantes tienen que llevar a cabo con el fin de responder a una serie de preguntas. En particular, el ejercicio incluía la identificación de los parámetros del motor incluyendo la ganancia estática, el tiempo característico y la zona muerta, y el diseño de controladores P y PI para el esquema de control de velocidad. El ejercicio tuvo un total de 20 preguntas, y para cada pregunta el alumno tenía que proporcionar una solución numérica proporcionando los vectores y valores escalares. En cada cuadro de texto, el alumno introduce los resultados obtenidos a partir de sus experimentos. Por ejemplo, en *p1* los alumnos proporcionan la velocidad en régimen permanente del motor para las entradas de tensión constante de 0 V a 3 V en incrementos de 0.5 V, mientras que en *p2*, los alumnos proporcionan el valor de la ganancia estática del motor evaluado a partir de estos datos.

Los experimentos se describen con detalle y las preguntas han sido diseñados de una manera tal que es posible medir directamente en la pantalla de datos del laboratorio. Por ejemplo, en *p15*, los alumnos deben diseñar analíticamente un controlador P usando el teorema de valor final tal que el error en estado estacionario cuando se aplica un escalón en la referencia es exactamente el 30 % de la amplitud del escalón. A continuación, en *p16* los alumnos deben llevar a cabo un experimento en el que la referencia de velocidad va desde 0 grados/segundo a 30 grados/segundo y medir el tiempo de establecimiento, la sobreoscilación y la velocidad en estado estacionario en el sistema de bucle cerrado.

Además de las pautas del ejercicio, se proporcionó material adicional a los alumnos para que pudieran relacionar la teoría con la práctica. Este material incluía teoría de motores de corriente continua, información sobre la respuesta transitoria de sistemas lineales de primer y segundo orden y un esquema de control en bucle cerrado de la posición y velocidad en este

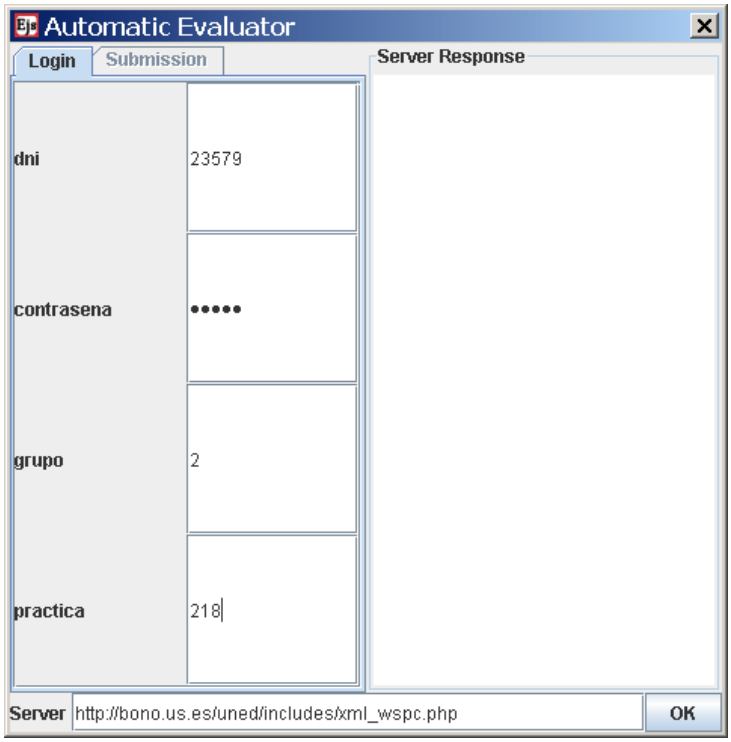

Figura 3.7: Elemento EEA para el laboratorio de servomotor

sistema.

3.4.1. Evaluación educacional

El laboratorio virtual de servomotor con evaluación automática ha sido probado con dos grupos de alumnos de cursos de control automático de las universidades UAL y PUCV. En ambos casos, la evaluación automática se empleó al final del semestre para evaluar la comprensión del alumno de los conceptos teóricos y prácticos en la ingeniería de control. La práctica o el trabajo requiere contestar 20 preguntas, que incluyen desde el modelado hasta el control de velocidad de un servomotor. Hay que tener en cuenta que cada alumno tiene una versión personalizada, y por lo tanto diferentes parámetros (ganancia, tiempo de reacción, zona muerta), del modelo de servomotor. Los alumnos podrían utilizar la simulación implementada en EJS para responder cada pregunta. Para enviar las respuestas, los estudiantes rellenaron el formulario proporcionado por el laboratorio virtual. Al final de la práctica, se invitó a los estudiantes a evaluar su experiencia con el laboratorio virtual empleando una encuesta *on-line* (lo que permitió a los alumnos votar de forma anónima). Se explicó a los alumnos que el objetivo de la encuesta era para obtener sus opiniones sobre la contribución de la experimentación en evaluación automática a su desarrollo como ingenieros.

Es importante remarcar que el objetivo principal de nuestro trabajo no es demostrar que los alumnos o los dos casos de estudio obtuvieron mejores resultados de aprendizaje, sino proporcionar una herramienta para que los profesores incluyan laboratorios virtuales con evaluación automática, lo que permite desarrollar diferentes metodologías de aprendizaje que pueden beneficiarse de una mayor cantidad de ejercicios y los comentarios generados por el

Valor en el aprendizaje
P4: ¿Cree que el uso de evaluaciones automáticas mejora su aprendizaje de la materia?
Valor añadido
P2: ¿Piensa que el uso del laboratorio virtual ha sido útil para evaluar su tarea? P5: En general, ¿está satisfecho con la experiencia de la práctica?
Usabilidad
P1: ¿Su experiencia con la página web ha sido sencilla para enviar la prática? P3: ¿Ha sido sencilla su experiencia con el servidor Goodle GMS?

Tabla 3.1: Cuestionario presentado a los alumnos para la evaluación de la plataforma

evaluador automático. Realizar un análisis comparativo del conocimiento de estos alumnos respecto a cursos anteriores que no incluían el laboratorio o la evaluación automática no se han llevado a cabo, ya que no eran el objetivo principal. Los elementos del cuestionario, que se basaban en trabajos anteriores (C. Jara, F. A. Candelas, F. Torres, S. Dormido, F. Esquembre, O. Reinoso, 2009),(Dormido, R., Vargas, H., Duro, N., Sánchez, J., Dormido-Canto, S., Farias, G., Esquembre, F., Dormido, S., 2008),(E. Fabregas, G. Farias, S. Dormido-Canto, S. Dormido, F. Esquembre, 2011), se organizaron en tres subescalas:

1. El valor de aprendizaje se centra en la percepción de los alumnos de la eficacia sobre cómo el laboratorio virtual (con evaluación automática) les ayuda a aprender los contenidos relevantes.

2. El valor añadido por la evaluación automática al laboratorio virtual, junto con sus ventajas frente a los informes tradicionales y la usabilidad del laboratorio se centra en la percepción de los estudiantes sobre la facilidad y claridad con la que son capaces de navegar a través de la plataforma de aprendizaje.

3. La usabilidad del diseño, que evalúa la facilidad de uso del laboratorio virtual para los usuarios finales (alumnos).

La tabla 3.1 muestra las preguntas de la encuesta presentada a un grupo de 28 (14 + 14) alumnos de los cursos de control de UAL y PUCV. Las respuestas se evaluaron utilizando cinco clasificaciones desde "muy de acuerdo" (MA) a "muy en desacuerdo" (MD). La figura 3.8 muestra el porcentaje de acuerdo a cada pregunta de los estudiantes UAL y PUCV.

Ambos grupos de estudiantes (UAL y PUC) han mostrado una respuesta positiva sobre su experiencia con la práctica. Estudiantes de la UAL parecen tener una mejor opinión general respecto a la evaluación automática. Esto podría explicarse por el hecho de que el grupo de estudiantes de la UAL ha estado trabajando con experiencias virtuales y remotos en los cursos anteriores. Las mejores percepciones de cada grupo se corresponden con la usabilidad de la página web para enviar la práctica (P1). Los peores comentarios provienen de la percepción de los alumnos de PUCV sobre la evaluación automática como una forma de evaluar su práctica.

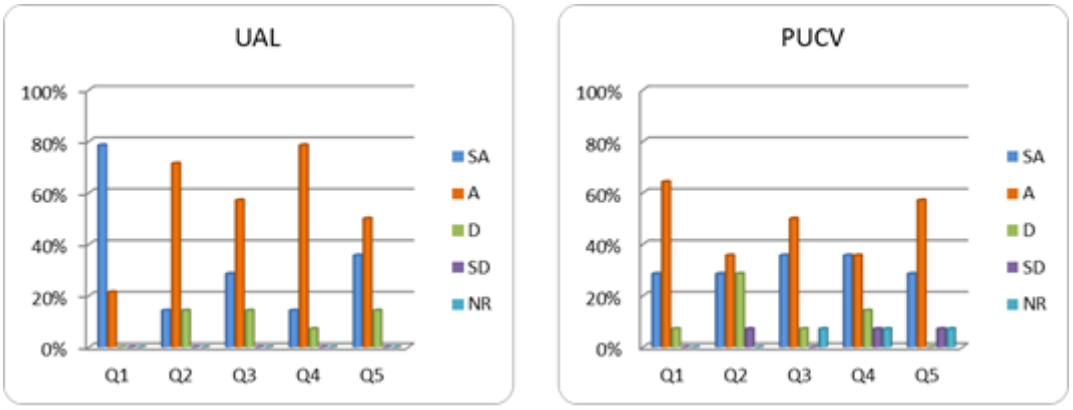

Figura 3.8: Resultado del cuestionaro de los alumnos de UAL y PUCV. Las respuestas están divididas desde "Muy de acuerdo" (SA) hasta "Muy en desacuerdo" (SD). NR significa que no hubo respuesta

Subescala	**MA (%)**	**DA (%)**	**DE (%)**	**MD (%)**	**NC (%)**
Valor de aprendizaje	25 %	57 %	11 %	4 %	4 %
Valor añadido	27 %	54 %	14 %	4 %	2 %
Diseño de usabilidad	43 %	48 %	7 %	0 %	2 %

Tabla 3.2: Resultado en porcentaje de acuerdo por subescala de los cuestionarios presentados a los alumos de UAL y PUCV (MA = Muy de acuerdo, DA = De acuerdo, DE = Desacuerdo, MD = Muy en desacuerdo, NC = No contesta)

Esto podría indicar la necesidad de introducir el uso de laboratorios virtuales y remotos en el inicio del curso en la PUCV.

La tabla 3.2 muestra los resultados de los dos grupos de estudiantes de acuerdo a las subescalas de la tabla 3.1. Los resultados indican que alrededor del 82 % de los alumnos piensa que el laboratorio con evaluación automática les ayudó a entender los conceptos relevantes (valor de aprendizaje), mientras que sólo el 15 % está en desacuerdo o muy en desacuerdo. En cuanto a valor añadido, el 80 % de los alumnos encuentran que el laboratorio con evaluación automática tiene ventajas sobre las prácticas tradicionales; sin embargo alrededor del 18 % de los alumnos encontraron lo opuesto. Este hecho puede indicar el valor que los alumnos dan a las prácticas tradicionales en el laboratorio, lo que significa que las actividades remotas o virtuales deben considerarse como un complemento (y no un sustituto) en la enseñanza de la ingeniería de control. En cuanto al diseño de usabilidad, aproximadamente el 91 % de los alumnos no tuvo dificultad para usar el laboratorio virtual o la plataforma. Estos resultados son muy satisfactorios, ya que los estudiantes adquieren una nueva visión cualitativa y práctica de sus conocimientos teóricos.

En resumen, ni los alumnos ni los profesores encontraron ninguna dificultad al utilizar los laboratorios virtuales y el elemento EEA para comunicarse con Goodle GMS. La mayoría de los alumnos entendieron el concepto de personalización y evaluación automática cuando se introdujeron las simulaciones y las tareas de trabajo. El principal reto era diseñar el laboratorio antes mencionado, que consumió un tiempo considerable. Sin embargo, es importante

destacar, que debido a los parámetros personalizados, será utilizado en cursos futuros, lo que justifica el tiempo invertido por el profesor.

Capítulo 4

Interfaz de sistemas no lineales

A continuación se describe la interfaz de sistemas no lineales (ISNL) desarrollada para el sistema Easy Java Simulations (EJS) empleando el elemento evaluador (EEA) y la plataforma web Goodle GMS. El objetivo de esta interfaz es la resolución de problemas de control no lineal, empleando una herramienta gráfica que permita el análisis y visualización de diferentes controladores para sistemas de hasta orden cuatro. A partir de parámetros que caracterizan de forma unívoca al alumno (por ejemplo su DNI), ISNL permite al profesor la creación de ejercicios personalizados. De esta forma, el alumno puede experimentar con diferentes controladores y visualizar la dinámica del sistema completo.

La novedad fundamental del paradigma del ISNL es que el alumno puede introducir en el laboratorio virtual expresiones matemáticas de tipo simbólico que serán posteriormente procesadas por el servidor con la ayuda del *Matlab Symbolic Toolbox* instalado. Los resultados de este capítulo están publicados en (Sánchez, C., Gomez-Estern, F. y Muñoz de la Peña, D., 2012).

4.1. Descripción de la interfaz

El objetivo de esta interfaz es la simulación de sistemas de control no lineal en espacio de estados con estructura afin

$$\dot{x}_i = f_i(x) + g_i(x) \cdot u_i \tag{4.1}$$

y que permite a los estudiantes simular diferentes leyes de control para sistemas no lineales de forma gráfica.

Esta interfaz trabaja de forma conjunta con el sistema Goodle GMS, lo que permite a los alumnos autentificarse en el sistema, escoger un ejercicio definido previamente por el profesor, calcular $u(x)$ de forma simbólica y simular el sistema, proporcionar la función de Lyapunov $V(x)$, enviar la solución que consideren más apropiada y su consecuente evaluación.

El primer paso para que un profesor cree un ejercicio es acceder al portal web Goodle GMS y definir el orden y las ecuaciones del sistema a través de *Matlab Symbolic Toolbox*. Por

ejemplo, si estamos interesados en crear un sistema de orden dos definido por

$$\begin{aligned} \dot{x}_1 &= \underbrace{x_1^2 + x_2}_{\text{f}} + \underbrace{(x_1 \cdot x_2)}_{\text{g}} \cdot u_1 \\ \dot{x}_2 &= \underbrace{x_2^2 + x_1}_{\text{f}} + \underbrace{x_2}_{\text{g}} \cdot u_2 \end{aligned}$$

debemos introducir en Goodle la siguiente expresión:

```
syms x1 f1 g1 u1 x2 f2 g2 u2;
ordenSistema = 2;
x1 = 0;
x2 = 0;
f1 = x1^2+x2;
g1 = x1*x2;
f2 = x2^2+x1;
g2 = x2;
```

Estas instrucciones nos permiten definir las ecuaciones de un sistema no lineal y sus condiciones iniciales. Una vez que el alumno se autentifique en el sistema, la interfaz EJS generará las expresiones apropiadas de acuerdo a estos datos. Es importante destacar, que este sistema debe ser escrito en lenguaje Matlab y el orden no debe ser superior a cuatro.

Una característica importante de esta arquitectura es su flexibilidad. El orden del sistema no lineal no está fijado de antemano. Éste se define en la plataforma Goodle GMS, y el sistema ISNL se encargará de generar los diferentes elementos visuales, las diferentes señales de control y las gráficas. Por ejemplo, si en el panel de Goodle se define un sistema de orden tres, ISNL mostrará las tres ecuaciones, sus condiciones iniciales, las gráficas de su dinámica de orden tres y las tres señales de control $u_1(x)$,$u_2(x)$ y $u_3(x)$.

4.2. Conexión entre ISNL y Goodle GMS

Tal y como se ha descrito inicialmente, el envío y recepción de datos hacia Goodle GMS se realiza a partir del elemento EEA descrito en el capítulo 2. Este componente permite a los alumnos identificarse en el sistema mediante usuario y contraseña, y escoger un ejercicio. Tras conectar con el sistema Goodle GMS y escoger un ejercicio, los estudiantes podrán simular diferentes señales de control sobre el sistema no lineal antes de proceder al envío de resultados.

4.3. Estructura de ISNL

La pantalla que se muestra al alumno viene representada en la figura 4.1. La parte superior de la pantalla contiene los controles de reproducción de la simulación, que permiten comenzar, pausar y reiniciar la simulación en todo momento. Mediante estos controles podemos variar también la velocidad de la simulación.

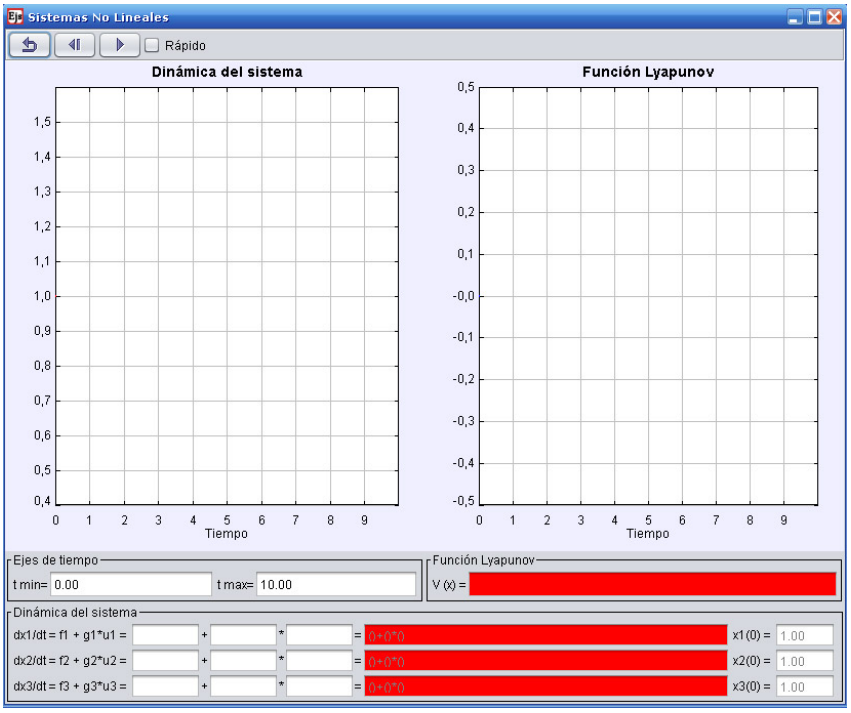

Figura 4.1: Interfaz de sistemas no lineales (ISNL)

Las diferentes gráficas del sistema se representan en la zona central de la pantalla (figura 4.2). La gráfica de la izquierda muestra la dinámica del sistema, donde cada estado del sistema está representado por una curva diferente. El comportamiento de estas curvas dependerá de la ley de control introducida por el alumno, por lo que se pueden simular diferentes controladores antes de enviar la solución al sistema de evalución.

La gráfica de la derecha muestra la evolución de la función de Lyapunov introducida por el alumno. Mediante esta gráfica podemos comprobar si el comportamiento de la función es monótono decreciente y asegurar así la estabilidad del sistema.

La parte inferior de la interfaz permite a los alumnos introducir diferentes leyes de control. Inicialmente, estos recuadros aparecen coloreados en rojo, avisando al usuario que están vacíos antes de ejecutar la simulación. A medida que los datos se van introduciendo, la señal de aviso desaparece.

El primer grupo de controles permite ajustar la escala de tiempo de las gráficas, permitiendo hacer zoom en un instante determinado. En cualquier caso, el sistema está preparado para realizar un autoescalado a medida que la gráfica evoluciona.

El siguiente control se utiliza para introducir la expresión de la función de Lyapunov que se mostrará a medida que el sistema evoluciona.

Finalmente, se muestran las ecuaciones del sistema. El contenido de estos recuadros se rellena automáticamente cuando el alumno se autentifica en el sistema y escoje un ejercicio. El alumno sólo puede introducir valores para las señales u_1, u_2, u_3 y u_4, ya que los otros controles están bloqueados y no permiten su modificación porque referencian a la dinámica del sistema y sus condiciones iniciales. f_1, g_1, f_2, g_2, etc.

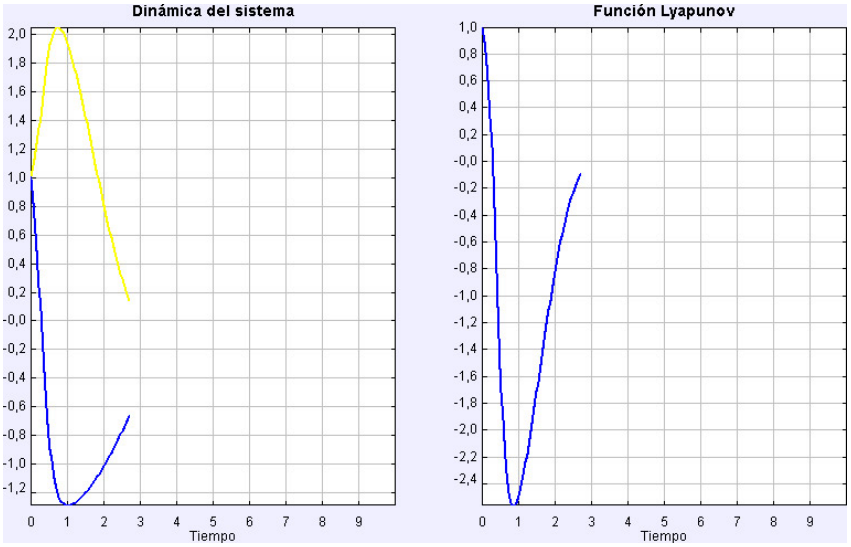

Figura 4.2: Dinámica del sistema y función de Lyapunov

Figura 4.3: Sistema completo

Para iniciar la simulación, el alumno debe introducir valores para las señales de control (en nuestro ejemplo, sólo u_1 y u_2), presionar ENTER para validar la expresión y pulsar sobre el botón PLAY de la parte superior.

4.4. Diseño de la interfaz ISNL

Para desarrollar la interfaz ISNL se ha utilizado el editor gráfico EJS. Esta herramienta permite, sin necesidad de tener conocimientos de programación, diseñar la interfaz gráfica de la simulación. Su funcionamiento es muy simple. Por un lado se define cuántos *frames* [1] va a tener la interfaz y por otro se añaden diferentes controles (texto, gráficas, botones, etc.) que queremos utilizar. En el caso de la intefaz ISNL se ha dividido la pantalla en tres zonas bien diferenciadas: la zona superior con los controles de reproducción, la zona central con las gráficas, y la zona inferior con la dinámica del sistema. En la figura 4.4 se puede la arquitectura de este diseño.

Para poder generar el modelo EJS que define las ecuaciones del sistema y poder vincularlo a la interfaz gráfica se hace uso de las variables recibidas desde el servidor. Estas variables se convierten en texto simbólico y se enlazan a las variables *derx1*, *derx2*, etc. asociadas a los controles de texto de la vista. En la figura 4.5 se muestra este proceso.

Es importante mencionar que EJS impone una restricción importante a la hora de vincular

[1] Marco o cuadro que aglutina una serie de elementos.

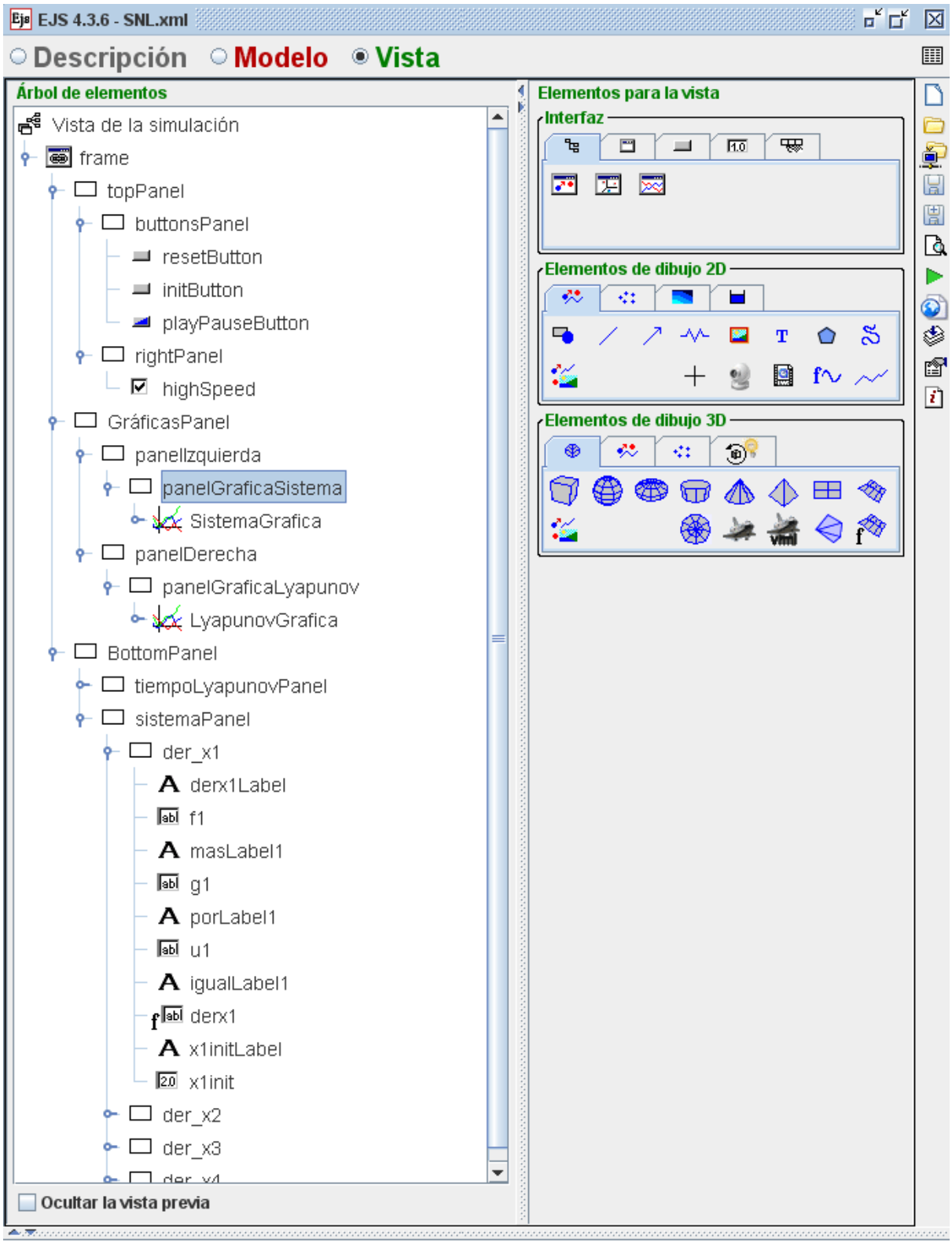

Figura 4.4: Estructura de la interfaz ISNL en EJS

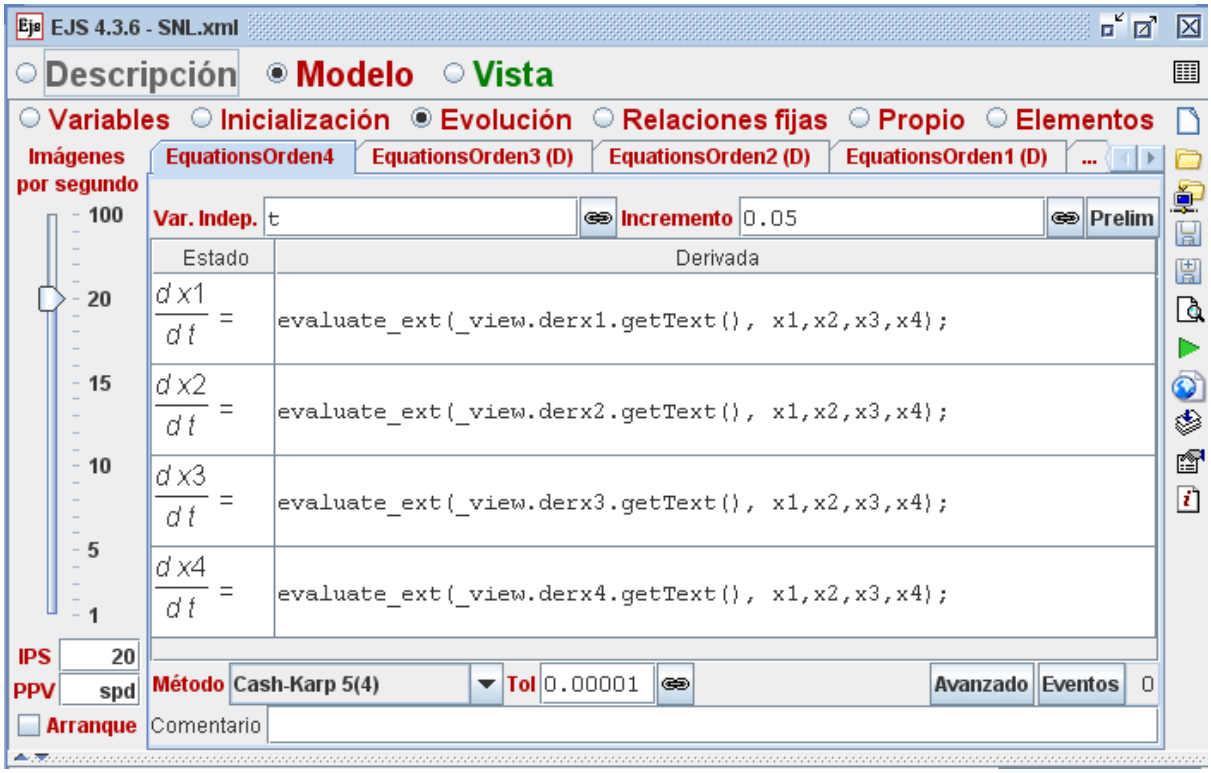

Figura 4.5: Ecuaciones que definen la evolución del sistema

las ecuaciones del sistema al modelo. Es preciso definir un interfaz específico según el orden del sistema, que se ha definido en la configuración del ejercicio (a través de variable OrdenSistema de MATLAB), dado que no es posible crear controles dinámicamente mediante el interfaz EJS. La figura 4.5 muestra las cuatro pestañas que se han definido para el sistema de orden cuatro de ejemplo. Si en un futuro se desea añadir un quinto orden, sólo hace falta añadir una pestaña al menú *Modelo* - *Evolución* y llamarlo *EquationsOrden5*. Ahí se incluirán, además de la evolución de los estados x_1,x_2,x_3 y x_4, la ecuación en derivadas de x_5.

Para evaluar la función de Lypunov sólo tenemos que introducir la siguiente expresión como una *Relación Fija* dentro de EJS:

```
V = evaluate_ext(_view.lyapunov.getText(),x1,x2,x3,x4);
```

La función "evaluate_ext" está definida dentro de EJS y se ejecuta en la propia interfaz.

4.5. Evaluación del alumno

ISNL permite al alumno la posiblidad de simular diferentes leyes de control de una forma gráfica y realizar el envío de la ley de control que se considera apropiada, tras haber trabajado sobre el sistema. La estabilidad de este sistema vendrá sujeta a la condición propuesta por la teoría Lyapunov para la estabilidad local de un sistema.

4.5.1. Teoría de estabilidad de Lyapunov

Dado un sistema de control, la más básica de sus propiedades es su estabilidad. La aproximación más general al estudio de la estabilidad de sistemas no lineales fue introducida a finales del siglo XIX por el matemático ruso Alexandr Mikhailovich Lyapunov, quien en 1892 publicó su tesis doctoral titulada *El Problema General de la Estabilidad del Movimiento*. Este trabajo presenta dos métodos para analizar la estabilidad de sistemas no lineales: el **método de linealización** y el así llamado segundo método o **método directo**.

El método de linealización establece conclusiones sobre la estabilidad local de sistemas no lineales alrededor de los puntos de equilibrio, tomando como base las propiedades de estabilidad de sus aproximaciones lineales. El método directo determina la estabilidad de un sistema no lineal en general, y no tiene carácter local. Este método se basa en la construcción de una función escalar de *energía generalizada* que describe al sistema, y cuya evolución en el tiempo permite deducir algunas de sus propiedades de estabilidad.

4.5.2. Definiciones de estabilidad

El punto de quilibrio $x = 0$ es **estable en el sentido de Lyapunov**, si para cualquier $R > 0$ existe un $r(R) > 0$ tal que si el estado en el instante inicial cumple que $\|x(0)\| < r$, entonces $\|x(t)\| < R$ para todo t. En caso contrario el punto de equilibrio es *inestable*.

Un punto de equilibrio $x = 0$ es **asintóticamente estable**, si es estable y además existe algún $r > 0$ tal que $\|x(0)\| < r$ implica que $x \to 0$ cuando $t \to \infty$. La estabilidad asintótica es una propiedad siempre más fuerte que la simple estabilidad, porque el hecho de que un punto de equilibrio sea asintóticamente estable implica que las trayectorias que comienzan suficientemente cerca de él, tienden hacia él.

Un punto de equilibrio $x = 0$ es **exponencialmente estable** si existen dos números positivos α y δ tales que:

$$\forall t > 0 \qquad \|x(t)\| \leq \alpha \|x(0)\| e^{-\delta t}$$

para cualquier valor inicial $x(0)$ dentro de la bola B_r en torno al origen. La convergencia exponencial implica que el vector de estado converge al origen más rápido que una función exponencial, y la constante δ se llama *ritmo de convergencia exponencial*. La estabilidad exponencial implica estabilidad asintótica, pero no a la inversa.

Un punto de equilibrio es globalmente asintóticamente (o exponencialmente) estable si la estabilidad asintótica (o exponencial) se mantiene para cualquier condición inicial.

4.5.3. Método directo de Lyapunov

La filosofía básica de este médoto está en la extensión matemática de una observación física básica: si la energía total de un sistema físico (mecánico, eléctrico, etc.) se disipa continuamente, entonces el sistema, sea lineal o no lineal, debe llegar en algún momento a algún punto de equilibrio estable. De esta forma, puede concluirse la estabilidad de un sistema no lineal simplemente examinando la variación de una única función escalar.

El método directo de Lyapunov nos indica que dado un sistena $\dot{x} = f(x)$, y siendo $x = 0$ un equilibrio de dicho sistema, si existe una función escalar $V : D \subset R^n \to R$, con $x = 0 \; \epsilon \; D$, y si se cumple

$$\begin{aligned} \dot{V}(x) &= \frac{\partial V}{\partial x}[f(x) + g(x) \cdot u] \leq 0 \quad x \in D \\ V(x) &> 0 \quad \forall x \in D - \{0\} \\ V(0) &= 0 \end{aligned}$$

Entonces el origen $x* = 0$ es estable en sentido de Lyapunov. Además, si se cumple

$$\dot{V}(x) = \frac{\partial V}{\partial x}[f(x) + g(x) \cdot u] < 0 \quad \forall x \in D - \{0\}$$

el origen $x* = 0$ es asintoticamente estable.

Esta definición de estabilidad es local, ya que comprende un dominio $D \subset R^n$ y por lo tanto $||x(0)|| < \delta \Rightarrow \lim_{t \to +\infty} x(t) = 0$

Con la estabilidad asintótica global $\forall x(0) \Rightarrow \lim_{t \to +\infty} x(t) = 0$ con lo que

$$\begin{aligned} V(0) &= 0 \\ V(x) &> 0 \quad \forall x \in R^n - \{0\} \\ \|x\| \to \infty &\Rightarrow \quad V(x) \to \infty \\ \dot{V}(x) &< \quad 0 \; \forall x \in R^n - \{0\} \end{aligned}$$

Es decir, hace falta que las curvas de nivel sean todas cerradas, lo que implica que $V(x)$ sea radialmente no acotada.

4.5.4. Cuencas de atracción mediante Lyapunov y LaSalle

La cuenca de atracción se define como el conjunto de condiciones iniciales cuyas trayectorias terminan en el origen. En los casos de estabilidad global, la cuenca de atracción es R^n.

El método de Lyapunov consiste en buscar una función escalar $V : R^n \to R$ tal que podamos demostrar que es Lyapunov (tal y como hemos definido en el apartado anterior) a partir de unas restricciones que nos delimitan una zona del espacio. Esa zona será la estimación de la cuenca de atracción, ya que en el caso de estabilidad asintótica, las curvas de nivel siempre se cruzan hacia dentro. El teorema de LaSalle nos permite demostrar estabilidad asintótica cuando $\dot{V} \leq 0$ y calcular la cuenca de atracción.

Sea $\Omega \subset D$ un conjunto compacto (cerrado y acotado) y positivamente invariante respecto al sistema $\dot{x} = f(x)$. Sea $V : D \to R$ una función continua diferenciable tal que $\dot{V}(x) \leq 0$ en Ω. Sea E el conjunto de todos los puntos donde $\dot{V}(x) = 0$ y M el mayor conjunto invariante en E. Entonces, toda trayectoria que comience en Ω, tiende a M cuando $t \to \infty$.

El teorema de LaSalle extiende el teorema de Lyapunov en tres direcciones. Primero, proporciona una estimación de la cuenca de atracción, la cual no tiene que ser de la forma

$$\Omega_c = \{x \in R^n | V(x) \leq c\}$$

Segundo, el teorema de LaSalle puede usarse en casos donde el sistema tenga un conjunto de puntos de equilibrio en vez de un punto de equilibrio aislado. Tercero, la función $V(x)$ no tiene que ser definida positiva.

4.5.5. Evaluación en ISNL

Para poder realizar la evaluación de este envío, se ha modificado Goodle GMS para transformar las leyes de control ($u_i \;\; i = 1, 2, 3, 4$) y la función de Lyapunov enviadas por el alumno, en expresiones simbólicas en lenguaje MATLAB (compatible con el Symbolic Math Toolbox). A continuación se muestra un fragmento del evaluador que define el profesor en la interfaz Goodle GMS:

```
% Radio elipse
radio = 1;
paso = 0.5;

% Sistema
derx1 = f1+g1*u1;
derx2 = f2+g2*u2;
derx3 = f3+g3*u3;
derx4 = f4+g4*u4;
derVx = diff(Vx,x1)*(derx1) + diff(Vx,x2)*(derx2)
        + diff(Vx,x3)*(derx3) + diff(Vx,x4)*(derx4)
```

Para comprobar la estabilidad del sistema, se ha realizado un algoritmo genérico que realiza un barrido en una región determinada por una elipse, y que comprueba, a partir de una serie de iteraciones, el cumplimiento, para cada valor x en dicha elipse, de la condición expresada en (4.2).

$$\dot{V}(x) \;\; = \;\; \frac{\partial V}{\partial x}[f(x) + g(x) \cdot u] \leq 0 \tag{4.2}$$

Si en algún momento de la iteración, el cálculo de la parte izquierda de la expresión (4.2) devuelve un valor positivo, la condición de estabilidad no se cumpliría y el sistema no sería estable en el sentido de Lyapunov. En el código de ejemplo que se muestra a continuación, se ha utilizado como región la esfera mostrada en la figura 4.6

```
% Definimos el radio de la elipse y el tamaño de paso
radio = 1;
paso = 0.5;

% Construimos la elipse
x1min = -radio;
x1max = +radio;
x2min = -radio;
```

```
x2max = +radio;
x3min = -radio;
x3max = +radio;
x4min = -radio;
x4max = +radio;

% Bucle de iteración
% En función del orden del sistema tendremos que realizar uno,
% dos, tres o cuatro bucles
switch ordenSistema
    case 1
        % Definición del sistema de orden 1
        derx1 = f1+g1*u1;
        derVx = diff(Vx,x1)*(derx1);
        % Barrido en una sóla región
        for indice1 = x1min:paso:x1max
            resultado = subs(derVx,{x1},{indice1});
            % Si la condición de estabilidad no se cumple
            if resultado > 0
                nota = 0;
                return;
            end
        end
    case 2
        % Definición del sistema de orden 2
        derx1 = f1+g1*u1;
        derx2 = f2+g2*u2;
        derVx = diff(Vx,x1)*(derx1)
                + diff(Vx,x2)*(derx2);
        % Barrido en dos regiones
        for indice1 = x1min:paso:x1max
            for indice2 = x2min:paso:x2max
                resultado = subs(derVx,{x1,x2},
                            {indice1,indice2});
                % Si la condición de estabilidad no se cumple
                if resultado > 0
                    nota = 0;
                    return;
                end

            end
        end
    case 3
        % Definición del sistema de orden 3
        derx1 = f1+g1*u1;
```

```
        derx2 = f2+g2*u2;
        derx3 = f3+g3*u3;
        derVx = diff(Vx,x1)*(derx1)
                + diff(Vx,x2)*(derx2)
                + diff(Vx,x3)*(derx3);

        % Barrido en tres regiones
        for indice1 = x1min:paso:x1max
            for indice2 = x2min:paso:x2max
                for indice3 = x3min:paso:x3max
                    resultado = subs(derVx,{x1,x2,x3},
                                {indice1,indice2,indice3});
                    % Si la condición de estabilidad no se cumple
                    if resultado > 0
                        nota = 0;
                        return;
                    end
                end
            end
        end
    case 4
        % Definición del sistema de orden 4
        derx1 = f1+g1*u1;
        derx2 = f2+g2*u2;
        derx3 = f3+g3*u3;
        derx4 = f4+g4*u4;
        derVx = diff(Vx,x1)*(derx1)
                + diff(Vx,x2)*(derx2)
                + diff(Vx,x3)*(derx3)
                + diff(Vx,x4)*(derx4);

        % Barrido en cuatro regiones
        for indice1 = x1min:paso:x1max
            for indice2 = x2min:paso:x2max
                for indice3 = x3min:paso:x3max
                    for indice4 = x4min:paso:x4max
                        resultado = subs(derVx,{x1,x2,x3,x4},
                        {indice1,indice2,indice3,indice4});
                        % Si la condición de estabilidad no se cumple
                        if resultado > 0
                            nota = 0;
                            return;
                        end
                    end
                end
```

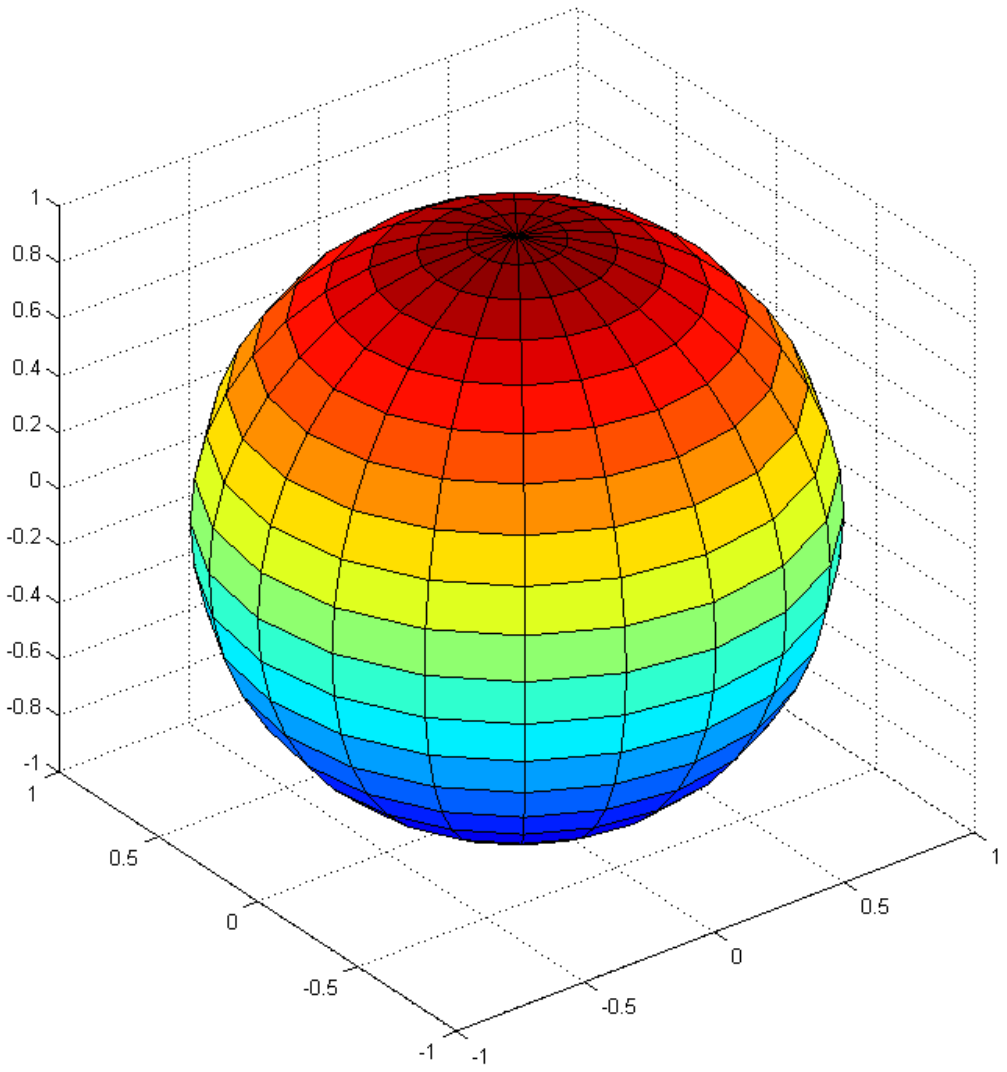

Figura 4.6: Dominio de exploración de la ecuación (4.2) en el algoritmo de evaluación

```
            end
        end
end

% Depuración para introducir comentarios
% comentarios = "Texto comentario";
```

Capítulo 5

Evaluación y generación de datos de forma automática para ejercicios de química analítica

En este capítulo se presenta una novedosa metodología para diseñar ejercicios de análisis instrumental de química analítica que pueden ser evaluados y personalizados para cada alumno de forma automática; esto es, a cada alumno se le asigna un conjunto diferente de datos experimentales generados de forma aleatoria y que satisfacen un conjunto de restricciones. El reto consiste en la dificultad de parametrizar de forma personalizada un problema definido a partir de datos experimentales, ya que no se puede emplear la técnica descrita en capitulos anteriores donde se relacionan parámetros con funciones sencillas a partir del DNI del alumno. Es necesario utilizar un método más sofisticado que permita generar los enunciados personalizados de forma automática con un componente aleatorio pero repetible. Esto permite al profesor proveer a los estudiantes con un conjunto de datos que complementan el trabajo del laboratorio y su evaluación. Para este fin se ha incluido un paso adicional en el proceso de generación de datos experimentales mediante un script de MATLAB. Las técnicas descritas en este capítulo serán de utilidad para la generación de ejercicios de diseño de controladores que se describe en el siguiente capítulo.

La metodología propuesta ha sido utilizada para evaluar 3 ejercicios personalizados de más de 200 alumnos procedentes de los cursos de Química Analítica Ambiental del primer curso del Máster de Ciencias en Contaminación Ambiental y del curso de Química Analítica Avanzada del Grado de Química impartida por el Departamento de Química Analítica de la Universidad de Extremadura. Los resultados han sido presentados en (Muñoz de la Peña, A., Muñoz de la Peña, D., Godoy-Caballero, M., González-Gómez, D., Gómez-Estern, F., Sánchez, C., 2014).

5.1. Parte experimental

A continuación se describe cómo se ha empleado Goodle GMS para diseñar ejercicios de análisis instrumental de química analítica. La diferencia respecto a otro tipo de problemas estriba en la dificultad de parametrizar de forma personalizada un problema definido a partir de datos experimentales, ya que no se puede emplear la técnica descrita en capitulos anteriores

donde se relacionan parámetros con funciones sencillas a partir de un dato característico del alumno. Para solventar esto se incluye un paso adicional en el flujo de trabajo de Goodle GMS para permitir una personalización más compleja.

5.1.1. Métodos y materiales

Enunciados personalizados para cada estudiante

Cada alumno recibe un enunciado personalizado para cada ejercicio que describe el problema objetivo de química analítica particular y el método que debe ser usado para resolverlo; sin embargo, el enunciado no incluye los datos experimentales que serán utilizados. En su lugar, los alumnos reciben un script MATLAB (fichero .p con el código oculto) que al ejecutarlo e introducir su número de DNI genera el conjunto de datos a utilizar en el ejercicio. Esto permite una fácil integración con el sistema Goodle GMS. Es importante destacar que para poder llevar a cabo la evaluación el sistema debe poder generar el mismo conjunto de datos para cada alumno. Para solventar esto la generación de datos se realiza a través de una función aleatoria de MATLAB donde la semilla inicial se obtiene a partir del DNI de cada alumno. Esto garantiza la repetibilidad y la obtención de resultados deterministas para cada alumno.

Ejercicios de química analítica considerados

A continuación se describen tres ejemplos de enunciados diferentes que ilustran cómo el sistema puede ser utilizado para generar datos y evaluaciones automáticas de ejercicios basados en datos complejos o resultados experimentales. En concreto, se muestra un análisis de regresión lineal y el cálculo de parámetros de calidad de un método analítico, junto a dos pruebas comparativas.

Ejercicio 1: Para determinar la concentración del molibdeno en una muestra, empleando una regresión lineal, se ha realizado una calibración estándar usando cinco niveles de concentración diferentes de una solución estándar y una luminiscencia molecular como señal instrumental. Todas las medidas se realizaron por triplicado. A partir de los datos obtenidos por el script de generación de datos; los alumnos deben determinar la concentración de la muestra desconocida y los parámetros de calidad del método analítico (linealidad, sensibilidad analítica y detección del límite).

Ejercicio 2: Dos métodos han sido empleados para analizar el contenido total de hierro de una roca de silicio. Un script de generación de datos proporciona la información numérica de ambos métodos. Los alumnos deben determinar si hay diferencias significativas entre ambos métodos considerando un nivel de confianza del 95 %, mediante una prueba comparativa de los resultados de los análisis de ambos métodos sobre la misma muestra.

Ejercicio 3: Se ha desarrollado un nuevo método basado en la espectroscopia de absorción atómica para detectar antimonio en muestras atmosféricas urbanas. Para poder valorar el rendimiento de la nueva metodología, se comparan sus resultados con los obtenidos por un método oficial de espectrofotometría. Un script de generación de datos proporciona la información numérica de ambos métodos. Se pide a los alumnos determinar si hay diferencias entre ambas metodologías para un nivel de confianza del 95 %, mediante una prueba t de

Student de los resultados de los análisis de ambos métodos sobre la misma muestra.

Consideraciones teóricas asociadas a cada ejercicio

Ejercicio 1

El uso de una regresión lineal sin ponderar para el establecimiento de una curva de calibración lineal implica la aceptación de cuatro suposiciones (Muñoz de la Peña, D.; Gómez-Estern, F.; Dormido, S., 2012):

- El eje Y contiene la respuesta instrumental y el eje X la concentración.
- Sólo aparecen errores en las medidas de la señales instrumentales.
- Los errores están distribuidos normalmente.
- La amplitud de los errores de medida son independientes de la magnitud de la concentración de las soluciones estándar (homocedasticidad).

Para cada estándar tenemos una señal analítica. El siguiente paso es ajustar estos resultados a una ecuación lineal (5.1.1):

$$y = mx + b$$

donde y es la señal analítica, x es la concentración, y m y b son la pendiente y el término independiente de la regresión lineal, respectivamente. Esta relación se obtiene utilizando el modelo estadístico de una regresión lineal (mínimos cuadrados). La desviación estándar de la regresión y la pendiente se pueden determinar también. La pendiente y el término independiente se calculan de acuerdo a:

$$\begin{aligned} m &= \frac{\sum_{i=1}^{n}[(x_i - \bar{x})(y_i - \bar{y})]}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ b &= \bar{y} - m\bar{x} \end{aligned}$$

donde y_i representa el valor de la señal instrumental obtenida para una concentración x_i de analito estándar, y $\bar{x}$ e $\bar{y}$ son los valores medios del conjunto de concentraciones y señales utilizados en la calibración.

La desviación estándar de la regresión de y sobre x, $s_{y/x}$, se calcula como:

$$S_{y/x} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y})^2}$$

donde $\hat{y}$ es el valor de la señal calculada de la ecuación de regresión, correspondiente a una concentración x_i, y n es el número total de puntos utilizados para calcular la regresión lineal; $s_{y/x}$ es una medida de la fidelidad de la aproximación lineal del conjunto de datos a la recta.

En la predicción de una muestra utilizando el método de calibración externo, la expresión general para calcular la desviación estándar de la predicción, en unidades de concentración, viene dado por:

$$S_x = \frac{S_{y/x}}{m}\sqrt{\left(\frac{1}{M}+\frac{1}{n}\right)+\frac{(y-\bar{y})^2}{m^2\sum_{i=1}^{n}(x_i-\bar{x})^2}}$$

donde M es el número de repeticiones del análisis de las muestras del problema, e y es la media aritmética de los valores de señal obtenidos en los M análisis.

Un número importante de los parámetros de análisis estadísticos necesarios para expresar los resultados analíticos se calculan a mano normalmente, por ejemplo, las figuras de mérito. Para expresar linealidad, el coeficiente de determinación que se usa frecuentemente es:

$$r^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2}{\sum(y_i-\bar{y})^2}$$

En un análisis de regresión el cuadrado del coeficiente de correlación entre x e y, r^2, que se conoce como coeficiente de determinación, expresa la proporción de la variación total explicada en términos la regresión lineal. Si $r^2 = 1$, todas las observaciones encajan perfectamente en una línea recta, y, consecuentemente, la variación total en y puede ser explicada en términos de una línea de regresión. Si, por otro lado, $r^2 = 0$, no existe correlación entre x e y. Sin embargo, la línea de regresión, que es paralela al eje x, no puede explicar ninguna variación en el eje y (Massart, D. L.; Vandeginste, B. G. M.; Buydens, L. M. C.; De Jong, S.; Lewi, P. J.; Smeyers-Verbeke, J., 1998). Aún así, valores de r^2 cercanos a 1 no implican necesariamente una buena relación de los datos analíticos (Miller, N. J.; Miller, J. C., 2009). En la literatura se propone una expresión alternativa de linealidad, la relación entre la desviación estándar de la pendiente y la pendiente, que es, la desviación estándar relativa de la pendiente, expresada en % (Cuadros-Rodríguez, L.; García Campaña, A. M.; Jiménez Linares, C.; Román Ceba, M., 1993):

$$Linealidad(\,\%) = \left(1-\frac{S_m}{m}\right)\times 100$$

Existen diferentes definiciones de la sensibilidad de un método analítico. La IUPAC recomienda la relación entre señales instrumentales y la concentración analítica. Esto puede ser medido por el análisis de regresión ya que la relación entre el error estándar de la regresión y la pendiente, expresada en unidades de concentración, establece la mínima diferencia de concentración que es estadísticamente discernible en cualquier punto en la línea de calibración. Este parámetro se conoce como la sensibilidad analítica (o resolución analítica) (Cuadros-Rodríguez, L.; García Campaña, A. M.; Jiménez Linares, C.; Román Ceba, M., 1993):

$$\gamma^{-1} = \frac{S_{y/x}}{m}$$

En términos generales, el límite de detección de un método puede ser descrito como la concentración que proporciona la señal de un instrumento significativamente diferente del

blanco o señal de fondo (Miller, N. J.; Miller, J. C., 2009). En la década de los 70, IUPAC (Cuadros-Rodríguez, L.; García Campaña, A. M.; Jiménez Linares, C.; Román Ceba, M., 1993) estipuló que el límite de detección, expresado como la concentración, x_L, viene derivado de la medida más pequeña, y_L, donde $y_L = \bar{y}bL + kS_{bL}$; $\bar{y}_{bL}$ es la media de las respuestas en blanco, S_{bL} es la desviación estándar de las respuestas en blanco, y k es una constante. El límite de detección, x_L, se obtiene como $x_L = \frac{kS_{bL}}{m}$ siendo m la pendiente de la línea de calibración. IUPAC recomienda encarecidamente usar un valor de constante $k = 3$.

El procedimiento de IUPAC para calcular el límite de detección ha sido analizado por muchos autores. Long y Winefordner (Massart, D. L.; Vandeginste, B. G. M.; Buydens, L. M. C.; De Jong, S.; Lewi, P. J.; Smeyers-Verbeke, J., 1998), en un estudio crítico, propusieron una modificación de la definición de IUPAC basado en la teoría de la propagación del error. La teoría de la propagación del error considera errores en los valores de m y b:

$$x_L = 3x\frac{\sqrt{(S_{bL}^2 + S_b^2) + \left(\frac{b}{m}\right)^2 S_m^2}}{m}$$

Clayton et al. (1987) estipularon en su trabajo que los métodos tradicionales para determinar los límites de detección están basados en conseguir protección ante las conclusiones de falso-positivo. Un mejor enfoque consiste en definir el límite de detección tal que se asegure la protección ante falsos positivos y negativos. Clayton propuso el uso de un parámetro $\Delta(\alpha, \beta)$ en vez de $k = 3$. Este factor depende de α, β y de los grados de libertad $(n-2)$ donde n es el número total de pares de datos utilizados para construir la línea de calibración y α y β están relacionados con la probabilidad de errores de falso positivo y falso negativo, respectivamente. Una buena aproximación se considera cuando $\alpha = \beta = 0{,}05$.

$$x_L = \Delta(\alpha, \beta)\frac{S_{y0}}{m}$$

donde S_{y0} se obtiene extrapolando el valor de la desviación estándar a una concentración cero, utilizando la desviación estándar de la señal en el punto, Y_0:

$$S_{y0} = S_{y/x}\sqrt{\frac{1}{m} + \frac{1}{n} + \frac{(-\bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

Ejercicio 2

Cuando no existen materiales de referencia para la evaluación de un método y éste debe ser validado comparando sus resultados frente a los obtenidos con un método estándar, o cuando se quieren comparar los resultados proporcionados por dos laboratorios diferentes, es necesario realizar una comparación entre las medias de dos conjuntos de resultados ($\bar{X}_1$ y $\bar{X}_2$) (Ramis-Ramos, G.; García Alvarez-Coque, M. C., 2001; Miller, N. J.; Miller, J. C., 2009). Esta comparación se realiza utilizando pruebas de comparación, que son pruebas estadísticas empleadas para comparar la precisión entre dos conjuntos de datos por medio de la probabilidad de que la hipótesis nula (H0) no haya sido rechazada. La hipótesis nula consiste en que la

diferencia entre las desviaciones estándar y la media se debe únicamente a errores aleatorios. La prueba se lleva a cabo en dos pasos. Primero, se comparan las varianzas de ambos métodos a través de la prueba F de Fisher. A continuación, dependiendo de los resultados de esta prueba, se comparan las medias a través de dos procedimientos. Para calcular la prueba F de Fisher se calcula el valor de Fcal a través de la ecuación:

$$F_{cal} = \frac{S_1^2}{S_2^2}$$

Donde S_1^2 y S_2^2 son las varianzas de dos conjuntos de datos. Hay que destacar que la mayor varianza de ambos métodos recae en el numerador ya que $Fcal > 1$. El valor obtenido se compara con el valor tabulado $Ftab$ obtenido de la distribución estadística F, para el valor de importancia (α) escogido y los grados de libertad $(n1 - 1)$ y $(n2 - 1)$ donde $n1$ y $n2$ son el número de medidas realizadas con los métodos 1 y 2 respectivamente.

Si $Fcal < Ftab$, entonces no existen diferencias significativas entre las varianzas de los dos conjuntos de datos. En este caso, el siguiente paso consiste en realizar una prueba t de Student. Si $n1$ o $n2$ son inferiores a 30, considerando una función de densidad Gaussiana y que S_1^2 y S_2^2 no son significativamente diferentes, el valor t_cal se calcula a través de la siguiente expresión

$$t_{cal} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{S^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

siendo S^2 la llamada *varianza común* que es calculada como:

$$S^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

donde t tiene $(n_1 + n_2 - 2)$ grados de libertad. Finalmente, el valor de t_cal obtenido se compara con el valor $t_tab\ (\alpha, (n_1 + n_2 - 2))$. Si $t_cal < t_tabl$ se acepta la hipótesis nula, por lo que no hay diferencias significativas entre los dos conjuntos de resultados.

Si $Fcal > Ftab$, no es posible calcular la varianza común S^2. En ese caso, la prueba necesaria para la comparación entre los dos conjuntos de datos es la prueba t de Cochran donde t_cal es:

$$t_{cal} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\left(\frac{S_1^2}{n_1}\right) + \left(\frac{S_2^2}{n_2}\right)}}$$

Los grados de libertad de t_tab se calculan de acuerdo a la siguiente ecuación:

$$g.d.l = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{\frac{S_1^2}{n_1}}{n_1+1} + \frac{\frac{S_2^2}{n_2}}{n_2+1}} - 2$$

Entonces, si $t_cal < t_tab$ se acepta la hipótesis nula y se puede concluir que no hay diferencias significativas entre los dos conjuntos de datos para el nivel de confianza seleccionado.

Ejercicio 3

En este ejercicio los alumnos tienen que comparar dos métodos a través del estudio de muestras que contienen diferentes cantidades de analito. La prueba de comparación de las medias no es adecuada en este caso, ya que ninguna variación relativa a los métodos se verá afectada por las diferencias entre las muestras comparadas (Ramis-Ramos, G.; García Alvarez-Coque, M. C., 2001; Miller, N. J.; Miller, J. C., 2009). En ese caso, la hipótesis nula establecida es que las diferencias significativas entre las concentraciones determinadas por ambos métodos no existe. La hipótesis no es cierta si la media de las diferencias es significativamente diferente de cero. La ecuación de t_cal para comparar la media con un valor conocido es:

$$t_{cal} = \frac{(\bar{X} - \mu)\sqrt{n}}{S}$$

De acuerdo a la hipótesis nula, $\mu = 0$, por lo que (17) se traduce en:

$$t_{cal} = \frac{(\bar{X}_d)\sqrt{n}}{S_d}$$

Donde $\bar{X}_d$ indica la media de las diferencias y S_d la desviación estándar de las diferencias y, para t_cal, los grados de libertad son $n - 1$. Finalmente, si $t_{cal} < t_{tab}$ la hipótesis nula se acepta y, por lo tanto, los dos métodos no son significativamente diferentes.

5.1.2. Diseño del evaluador

Los resultados enviados por los alumnos a través del formulario de texto pueden ejecutarse como código MATLAB para generar un conjunto de variables en el espacio de trabajo (workspace).

En este trabajo se presentan dos tipos de ficheros de MATLAB. Los ficheros encriptados de MATLAB, conocidos como archivos "p", se utilizan para codificar las instrucciones necesarias para generar los datos de los alumnos, bloqueando así cualquier alteración de este código. Los ficheros abiertos de MATLAB, también conocidos como archivos "m", se utilizan para almacenar las instrucciones necesarias para evaluar a los alumnos. El servidor ejecuta este código, y a continuación el código evaluador diseñado por el profesor.

En la figura 5.1 se muestra el flujo de información. El enunciado y el fichero "p" se los proporciona el profesor al alumno de forma independiente del servidor (usando un LMS por

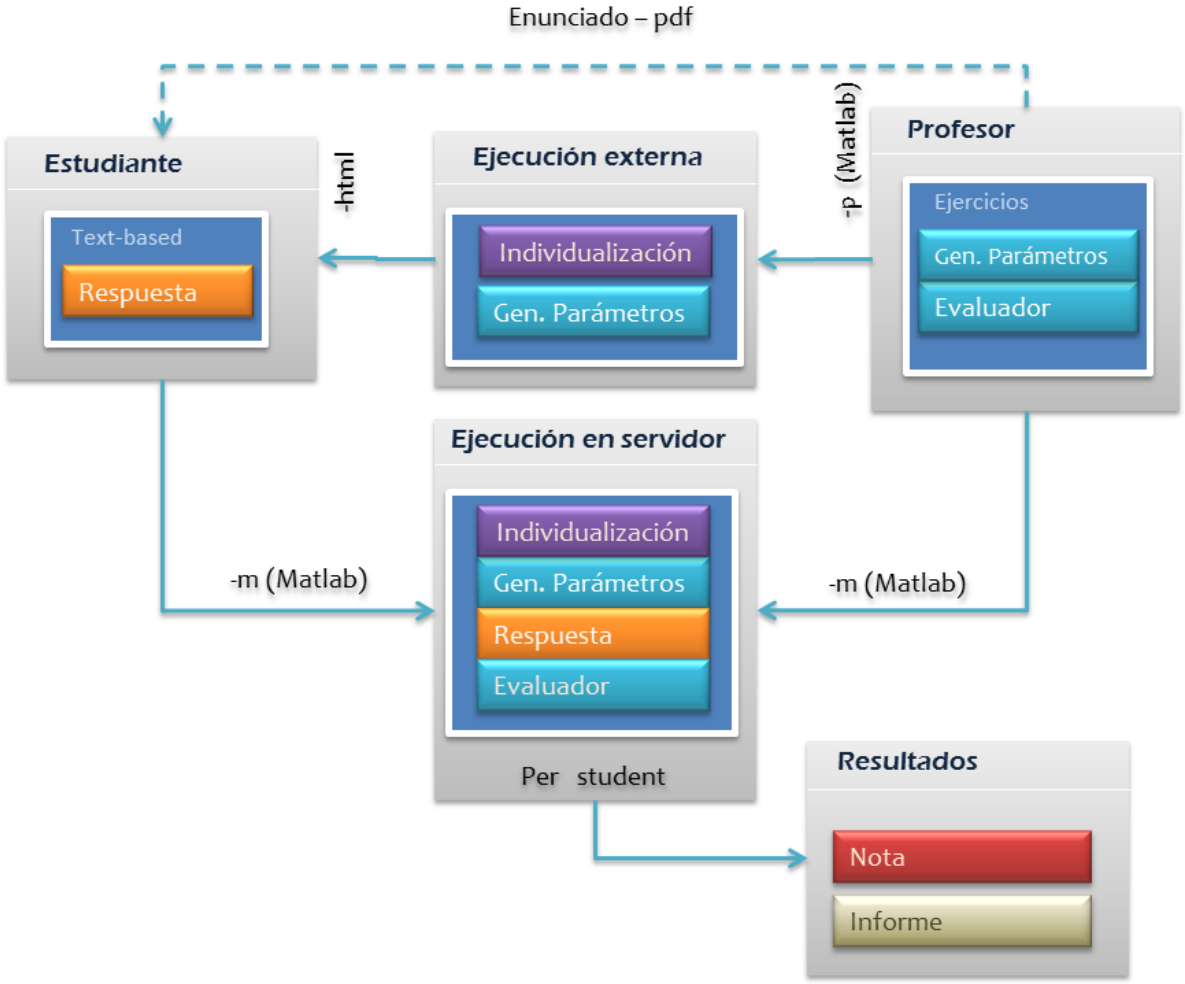

Figura 5.1: Flujo de información entre el profesor, el alumno y el servidor.

ejemplo). El alumno debe ejecutar el fichero "p" en un MATLAB propio para obtener los datos de su enunciado en formato "html". El código evaluador realiza los cálculos con los datos entregados por el alumno para establecer las respuestas correctas. Se utilizan diferentes rangos de notas para cada tipo de ejercicio. Hay que destacar que el criterio de evaluación es fácilmente modificable por el profesor. El código de evaluación no sólo establece una nota, sino que también informa al alumno con un informe detallado de sus resultados. En la sección 5.3 se muestran los códigos de evaluación empleados en cada uno de los ejercicios.

5.2. Resultados

A continuación se exponen los resultados obtenidos en los tres ejercicios y el análisis de la encuesta de satisfacción que se realizó a los alumnos.

5.2.1. Ejercicio 1

La primera parte del ejercicio consiste en generar los datos a través del entorno de MATLAB. Las concentraciones y las señales de datos de los métodos empleados, así como las muestras analíticas. se personalizan para cada alumno a través de su DNI. Las medidas de cada alumno se obtienen mediante una regresión lineal con valores aleatorios para la pendiente y el término independiente, añadiendo un error de medida del 5 % en los métodos y las medidas del problema. Las concentraciones estándar permanecieron fijas entre los valores de 0.2 y 3.2. La concentración del problema se escogió de forma automática en el rango entre 0.5 y 3. Este proceso se lleva a cabo a través de un fichero MATLAB, exercise1data.p, que puede ser descargado por los alumnos en la página web de la asignatura (Moodle). Al ejecutar este fichero se genera un fichero HTML que muestra al alumno los datos necesarios para trabajar y que se obtienen a partir de su DNI.Como ejemplo, se muestra la información generada en el fichero HTML para el DNI 80071067:

```
EXERCISE DATA
ID number 80071067

The three measures for the standard 1 (x = 0.2) are:
y1 = 4.25154, y2 = 3.93557, y3 = 4.30114

The three measures for the standard 2 (x = 0.4) are:
y1 = 4.2702, y2 = 4.21205, y3 = 4.25277

The three measures for the standard 3 (x = 0.8) are:
y1 = 4.73237, y2 = 4.96964, y3 = 4.78168

The three measures for the standard 4 (x = 1.6) are:
y1 = 5.46605, y2 = 5.5696, y3 = 5.4491

The three measures for the standard 5 (x = 3.2) are:
y1 = 7.45906, y2 = 7.40427, y3 = 7.12714

The three measures of the sample are:
y1 = 4.65451, y2 = 4.91009, y3 = 4.89413
```

El problema propuesto se resuelve de acuerdo a esta información. El usuario debe utilizar el software ACOC (MacDougall, D.; Crummett, W. B., 1980; Long, G. L.; Winefordner, J. D., 1983) para obtener los parámetros de las curvas de calibración y las figuras de mérito analíticas, así como la concentración calculada con su correspondiente incertidumbre. Los resultados se envían a la plataforma Goodle GMS. Para ello, cada alumno debe registrarse en el servidor con su DNI y contraseña.

Los alumnos envían la linealidad (lin), la sensivilidad analítica (res), los límites de detección de acuerdo a los criterios de Long y Winefordner (LODWL) (Massart, D. L.; Vandeginste, B. G. M.; Buydens, L. M. C.; De Jong, S.; Lewi, P. J.; Smeyers-Verbeke, J., 1998) y Clayton (LODC) (Clayton, C. A.; Hines, J. W.; Elkins, P. D., 1987), y, finalmente, la concentración

de molibdeno calculada en la muestra del problema (X_p), así como la incertidumbre asociada (E_p). La incertidumbre se proporciona a través de un valor de t-Student correspondiente a n-2 grados de libertad y a un nivel de significación del 95 %. Los alumnos deben proporcionar la respuesta con un único dígito de incertidumbre. Los resultados se envían a través de un formulario de texto con la siguiente estructura:

```
Quality_parameters = [lin res LODWL LODC];
Concentration = [xp ep];
```

La calificación del alumno se calcula comparando su resultado con el resultado correcto calculado por MATLAB. La puntuación máxima es 10. Si la concentración real está dentro del rango de valores estimado por los alumnos, obtienen 2.5 puntos. Si la incertidumbre asociada es correcta, obtienen 2.5 puntos. Por cada parámetro de calidad correcto, obtienen 1.25 puntos.

5.2.2. Ejercicio 2

En este ejercicio los alumnos tienen que comparar los resultados conseguidos a través de dos métodos de acuerdo a las indicaciones teóricas que se muestran a continuación. Al igual que el ejercicio anterior, cada alumno dispone de datos personalizados de acuerdo a su número de DNI. Utilizando el mismo DNI de ejemplo, el alumno obtendrá, tras ejecutar el script de MATLAB, exercise2data.p, un fichero HTML con la información numérica. En este caso, la generación de datos se ha diseñado de tal forma que las soluciones de los ejercicios generados se distribuyen en cuatro posibles casos, dependiendo de si la media o la desviación estándar mostraban diferencias significativas. Cada alumno tiene un conjunto de datos comprendido entre un valor máximo y mínimo de valores, y se le asigna uno de los cuatro posibles casos de forma aleatoria. Dependiendo del caso, los dos conjuntos de datos se generaron con diferentes valores de errores y media. Los valores específicos de cada caso fueron ajustados a través de una extensiva simulación hasta que la distribución se consideró aceptable.

Como ejemplo, para un DNI 80071067, el script generó un fichero llamado "80071067.html" con los siguientes datos:

```
NUMERICAL DATA FOR THE EXERCISE
ID 80071067

Method A: Atomic absorption spectroscopy
Fe2O3(percentage)
36.8961
36.6263
36.1783
34.7433
35.8801

Method B: Molecular spectroscopy
```

```
Fe2O3(percentage)
36.1608
36.0275
36.1827
35.7914
35.8459
36.1211
36.2558
35.8745
35.8983
```

Con esta información los alumnos deben aplicar la prueba estadística adecuada y enviar los resultados a Goodle GMS. Precisamente, tienen que enviar los valores calculados y tabulados de F, los valores de T y los resultados de la comparación. En este caso, el alumno tiene que indicar la presencia de diferencias significativas con el número 1 y su ausencia con el número 0. Así, los resultados se envían en un formulario de texto con la siguiente estructura:

```
test_parameters = [Fcal Ftab Tcal Ttab];
results = res;
```

Por ejemplo, para el DNI mostrado los resultados serán:

```
test_parameters = [24.2764 5.0530 0.1249 4.3030];
results = 0;
```

Si los resultados y parámetros son correctos la puntuación es 10, en caso contrario la puntuación es 0.

5.2.3. Ejercicio 3

El objetivo de este ejercicio es validar un nuevo método analítico a través de la comparación de los resultados proporcionados por el nuevo método con los obtenidos de una referencia. Para resolver este ejercicio los alumnos tienen que aplicar las ecuaciones descritas anteriormente en la sección Materiales y Métodos, teniendo en cuenta que en este caso tienen que comparar dos métodos a través del estudio de muestras que contienen diferentes cantidades de analito. Como en el ejercicio anterior, cada alumno tiene datos personalizados de acuerdo a su número de DNI siguiendo el mismo procedimiento mencionado anteriormente. En este caso, los datos pertenecen únicamente a dos posibles casos, dependiendo de si hay diferencias significativas o no. El diseño de los parámetros también se realizó a través de una extensiva simulación como en el Ejercicio 2.

Usando el mismo DNI de ejemplo, el alumno obtiene, tras ejecutar el script generador de MATLAB, "exercise3data.p", un fichero HTML con la información numérica. Como ejemplo, para el DNI 80071067, el script genera un fichero 80071067.html con la siguiente información:

```
NUMERICAL DATA FOR THE EXERCISE
ID 80071067

Method A
New method
Antimony (mg/m3)
86.22
40.48
37.15
32.49
66.75
20.47
66.96
44.35
54.63

Method B
Standard official method
Antimony (mg/m3)
83.62
45.05
39.77
37.11
72.21
22.84
68.59
47.45
52.51
```

Con esta información los alumnos deben aplicar la prueba estadística adecuada y enviar los resultados a Goodle GMS. Los alumnos envían los valores de T calculados (T_{cal}) y tabulados (T_{tab}), y los resultados de la comparación (res), si existen diferencias o no entre los métodos A y B. Como en el ejemplo anterior, el alumno tiene que indicar la presencia de diferencias significativas con el número 1 y su ausencia con el 0. Así, los resultados se envían en un formulario con la siguiente estructura:

```
test_parameters = [Tcal Ttab];
results = res;
```

Por ejemplo, para el DNI mostrado anteriormente los resultados serán:

```
test_parameters = [2.2954 2.3060];
results = 0;
```

Si los resultados y parámetros son correctos la puntuación es 10, en caso contrario la puntuación es 0.

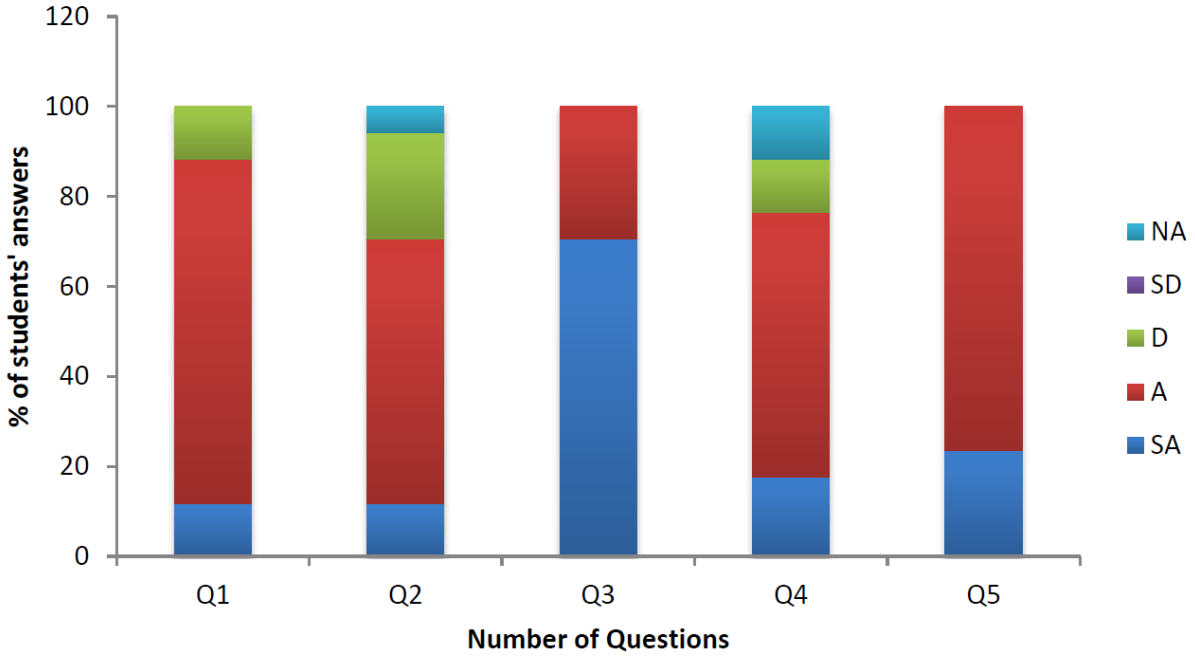

Figura 5.2: Resultados de la encuesta de satisfacción a los alumnos. Los resultados se expresan mediante el porcentaje (%) de respuestas de acuerdo a la siguiente escala: SA: Fuertemente de acuerdo; A: De Acuerdo; D: Desacuerdo; SD: Fuertemente en desacuerdo; NA: Sin respuesta.

5.2.4. Análisis tras la evaluación

Tras realizar los ejercicios los alumnos fueron preguntados por la idoneidad de esta aplicación y su satisfacción a través de una encuesta. La encuesta se realizó a la totalidad de alumnos de un curso de analítica química en diciembre de 2012. Ésta incluye un total de cinco preguntas en las que los alumnos indican su grado de conformidad o disconformidad sobre una declaración. Las preguntas del cuestionario fueron las siguientes:

1. Emplear una página web para enviar los resultados de los ejercicios.
2. Emplear un servidor para calcular automáticamente la nota de los ejercicios.
3. Goodle GMS es fácil de utilizar.
4. La evaluación automática mejora el proceso de aprendizaje porque proporciona una evaluación continua.
5. En general, ¿está usted satisfecho con el curso?

De acuerdo al resumen de los resultados obtenidos mostrados en la siguiente ilustración (Figura 5.2) se puede destacar el alto grado de satisfacción sobre el prototipo y su simplicidad de uso.

5.3. Material suplementario

A continuación se detallan los scripts de MATLAB (ficheros "p" y "m") de cada uno de los ejercicios.

5.3.1. Ejercicio 1

exercise1data.p

```
DNI=input ('Input ID:');

% A deterministic seed based on the ID is provided
rand('seed',DNI);

% Random slope and bias of the method
dni_pdt = 1 + 4*rand(1);
dni_oor = 1 + 4*rand(1);

% Random data generated for the particular ID
% Each measurement has a 5% error
% Standard measures
Concentraciones=[0.2;0.4;0.8;1.6;3.2];
Abs_dni=[];
Absorbancias=[];
for j=1:5
    Abs=Concentraciones(j)*dni_pdt+dni_oor;
    erel = 0.05;
    for kk=1:3
        aux = (1-erel) + 2*erel*rand(1);
        Abs_dni=[Abs_dni (aux*Abs)];
    end
    Absorbancias=[Absorbancias;Abs_dni];
    Abs_dni=[];
end

% Sample measures
replicas = [];
dni_xu = 0.5 + 2.5*rand(1);
repl=dni_xu*dni_pdt+dni_oor;
for k=1:3
    aux = 0.95 + 0.1*rand(1);
    replicas=[replicas aux*repl];
end
replicas=replicas';
Absorbancias;
```

```
disp(sprintf('File %d.html generated',DNI));
comentarios = '<html lang="es"> <body> EXERCISE DATA <br>';
comentarios = strcat(comentarios,sprintf(' ID %d
<br><br>',DNI));
for i=1:5
    comentarios = strcat(comentarios,sprintf(' The three
    measures for the standard %d (x = %g)
    are:<br>',i,Concentraciones(i)));
    comentarios = strcat(comentarios,sprintf(' y1 = %g, y2
    = %g, y3 =
    %g<br><br>',
    Absorbancias(i,1),Absorbancias(i,2),Absorbancia
    s(i,3)));
end
comentarios = strcat(comentarios,sprintf(' The three
measures of the sample are:<br>'));
comentarios = strcat(comentarios,sprintf(' y1 = %g, y2 =
%g, y3 = %g<br>',replicas(1),replicas(2),replicas(3)));
comentarios = strcat(comentarios,'</body> </html>');
fid = fopen(sprintf('%d.html',DNI),'w');
fprintf(fid,'%s',comentarios);
fclose(fid);
```

exercise1eval.m

```
% The ID is stored in variable dni in text form
automatically by GOODLE GMS
DNI = str2num(dni);

% A deterministic seed based on the ID is provided
rand('seed',DNI);

% Random slope and bias of the method
dni_pdt = 1 + 4*rand(1);
dni_oor = 1 + 4*rand(1);

% Random data generated for the particular ID
% Each measurement has a 5% error
% Standard measures
Concentraciones=[0.2;0.4;0.8;1.6;3.2];
Abs_dni=[];
Absorbancias=[];
for j=1:5
    Abs=Concentraciones(j)*dni_pdt+dni_oor;
    erel = 0.05;
    for kk=1:3
```

```
        aux = (1-erel) + 2*erel*rand(1);
        Abs_dni=[Abs_dni (aux*Abs)];
    end
    Absorbancias=[Absorbancias;Abs_dni];
    Abs_dni=[];
end

% Sample measures
replicas = [];
dni_xu = 0.5 + 2.5*rand(1);
repl=dni_xu*dni_pdt+dni_oor;
for k=1:3
    aux = 0.95 + 0.1*rand(1);
    replicas=[replicas aux*repl];
end
replicas=replicas';
Absorbancias;
patron=[Concentraciones,Absorbancias];

% First, the correct solution is computed
m=15;
ac_file=[];
for i=1:5
    for j=2:4
     ac_file=[ac_file;patron(i,1),patron(i,j)];
    end
end
ac_b=[cov(ac_file(:,1),ac_file(:,2))]/[(std(ac_file(:,1)))^
2];
ac_pdt=ac_b(2,1);
ac_oor=mean(ac_file(:,2))-ac_pdt*(mean(ac_file(:,1)));
ac_ycalajus=ac_oor+(ac_file(:,1)).*ac_pdt;
ac_sysobx=((sum((ac_file(:,2)-ac_ycalajus).^2))/(m-2))^(1/2);
t=2.16;

% Calculated concentration of the molybdenum in the problem
% sample (xp), as well as the associated uncertainty
ac_xu=(mean(replicas)-ac_oor)./ac_pdt;
ac_int_conf=t*((1/3+1/15+((mean(replicas)-
mean(ac_file(:,2))).^2)./(ac_pdt^2*sum((ac_file(:,1)-
mean(ac_file(:,1))).^2))).^(1/2)).*ac_sysobx/ac_pdt;

% The results are rounded with a single uncertain digit
e = 1;
while floor(e*ac_int_conf)<1
```

```
    e=10*e;
end
ep = round(e*ac_int_conf)/e;
xp = round(e*ac_xu)/e;
Resultado_concentracion_P=[xp ep];

% Quality parameters
% Lineality
Tsumxcuad=sum((ac_file(:,1)).^2);
Tmedxexp=mean((ac_file(:,1)));
Tdif=(ac_file(:,1)-Tmedxexp);
Tdifcuadx=Tdif.^2;
Tsumdifcuadx=sum(Tdifcuadx);
Tdesvord=((Tsumxcuad)/(m*(Tsumdifcuadx)))^(1/2)*ac_sysobx;
Tdesvpdt=((1/sum((ac_file(:,1)-
mean(ac_file(:,1))).^2))^(1/2))*ac_sysobx;
L=(1-Tdesvpdt/ac_pdt)*100;

% Analytical resolution
S_a=ac_sysobx/ac_pdt;

% Limits of detection according to the Long and Winefordner
(LODWL) and
% Clayton (LODC) criteria
X_LOD_WL=(3*(0+Tdesvord^2+(ac_oor/ac_pdt)^2*Tdesvpdt^2)^(1/
2))/ac_pdt;
wo_2=1/3+1/15+(mean(ac_file(:,1)))^2*1/sum((ac_file(:,1)-
mean(ac_file(:,1))).^2);
delta= 3.4787;
X_LOD_CLAYTON=delta*(wo_2^(1/2))*ac_sysobx/ac_pdt;

% Correct solution
Resultados_parametros_calidad_P=[L,S_a,X_LOD_WL,X_LOD_CLAYT
ON];
comentarios = '<html lang="es"> <body> RESULTS <br>';
comentarios = strcat(comentarios,sprintf(' ID %s
<br><br>',dni));
comentarios = strcat(comentarios,sprintf(' Calculated
concentration<br><br>'));
comentarios = strcat(comentarios,sprintf(' Results (raw):
%g ppm +-%g ppm <br>',ac_xu,ac_int_conf));
comentarios = strcat(comentarios,sprintf(' Results
(rounded): %g ppm +-%g ppm <br>',xp,ep));
comentarios = strcat(comentarios,sprintf(' Results
(student): %g ppm +-%g ppm
```

```
<br>',Concentracion(1),Concentracion(2)));
comentarios = strcat(comentarios,sprintf('<br> Correct
quality parameters: <br><br>'));
comentarios = strcat(comentarios,sprintf('Lineality (lin):
%g <br>',Resultados_parametros_calidad_P(1)));
comentarios = strcat(comentarios,sprintf('Analytical
resolution (res): %g
<br>',Resultados_parametros_calidad_P(2)));
comentarios = strcat(comentarios,sprintf('(LODWL): %g
<br>',Resultados_parametros_calidad_P(3)));
comentarios = strcat(comentarios,sprintf('(LODC): %g
<br>',Resultados_parametros_calidad_P(4)));
comentarios = strcat(comentarios,sprintf('<br> Students
quality parameters: <br><br>'));
comentarios = strcat(comentarios,sprintf('Lineality (lin):
%g <br>',Parametros_calidad(1)));
comentarios = strcat(comentarios,sprintf('Analytical
resolution (res): %g <br>',Parametros_calidad(2)));
comentarios = strcat(comentarios,sprintf('(LODWL): %g
<br>',Parametros_calidad(3)));
comentarios = strcat(comentarios,sprintf('(LODC): %g
<br>',Parametros_calidad(4)));
nota = 0;
if(abs(Concentracion(1)-ac_xu)<ac_int_conf)
    nota = nota + 2.5;
    comentarios = strcat(comentarios,sprintf('<br>The
    concentration is correct (2.5) <br><br>'));
else
    comentarios = strcat(comentarios,sprintf('<br>The
    concetration is not correct<br><br>'));
end
if(Concentracion(2)==ep)
    nota = nota +2.5;
    comentarios = strcat(comentarios,sprintf('The
    uncertainty is correct (2.5) <br><br>'));
else
    comentarios = strcat(comentarios,sprintf('The
    uncertainty is not correct<br><br>'));
end
k = 0;
for i=1:4
    if(abs(Resultados_parametros_calidad_P(i)-
        Parametros_calidad(i))
        <0.01*abs(Resultados_parametros_calid
        ad_P(i)))
```

```
        nota = nota+1.25;
        k = k+1;
    end
end
comentarios = strcat(comentarios,sprintf(' %d quality
parameters correct (1.25each)<br><br>',k));
comentarios = strcat(comentarios,sprintf('<br>Grade %g
<br>',nota));
comentarios = strcat(comentarios,'</body> </html>');
```

5.3.2. Ejercicio 2

exercise2data.p

```
DNI=input ('Input ID:');

% A deterministic seed based on the ID is provided
rand('seed',DNI);
t_distr=[1,12.706;
2, 4.303;
3, 3.182;
4, 2.776;
5, 2.571;
6, 2.447;
7, 2.365;
8, 2.306;
9, 2.262;
10, 2.228;
11, 2.201;
12, 2.179;
13, 2.160;
14, 2.145;
15, 2.131;
16, 2.120;
17, 2.110;
18, 2.101;
19, 2.093;
20, 2.086;
21, 2.080;
22, 2.074;
23, 2.069;
24, 2.064;
25, 2.060;
26, 2.056;
```

```
27, 2.052;
28, 2.048;
29, 2.045];
test_f5_doble=[647.8, 799.5, 864.2, 899.6, 921.8, 937.1,
948.2, 956.7, 963.3, 968.6;
38.51, 39.00, 39.17, 39.25, 39.30, 39.33, 39.36, 39.37,
39.39, 39.40;
17.44, 16.04, 15.44, 15.10, 14.88, 14.73, 14.62, 14.54,
14.47, 14.42;
12.22, 10.65, 9.979, 9.605, 9.364, 9.197, 9.074, 8.980,
8.905, 8.844;
10.01, 8.434, 7.764, 7.388, 7.146, 6.978, 6.853, 6.757,
6.681, 6.619;
8.813, 7.26, 6.599, 6.227, 5.988, 5.820, 5.695, 5.600,
5.523, 5.461;
8.073, 6.542, 5.890, 5.523, 5.285, 5.119, 4.995, 4.899,
4.823, 4.761;
7.571, 6.059, 5.416, 5.053, 4.817, 4.652, 4.529, 4.433,
4.357, 4.295;
7.209, 5.715, 5.078, 4.718, 4.484, 4.320, 4.197, 4.102,
4.026, 3.964;
6.937, 5.456, 4.826, 4.468, 4.236, 4.072, 3.950, 3.855,
3.779, 3.717];
N = [5 10]; %Minimum and maximum number of samples
X = [10 90]; %Minimum and maximum value of the concetration
of the sample

% Randomly determine one of the four different cases
if rand(1)<0.5
    % Both methods have the same mean
    erelx = 0;
else
    % The means differ 5%
    if rand(1)<0.5
        erelx = 0.05;
    else
        erelx = -0.05;
    end
end
if rand(1)<0.5
    % Both methods have a similar variance
    % Maximum measurement errors of 2%
    erela = 0.02;
    erelb = 0.02;
else
```

```
    % There are significative differences in the variance
    % Maximum measurement errors of 1% and 4%
    if rand(1)<0.5
        erela = 0.01;
        erelb = 0.04;
    else
        erela = 0.04;
        erelb = 0.01;
    end
end

% Determine the number of samples and the problem
concentration
na = N(1)+round((N(2)-N(1))*rand(1));
nb = N(1)+round((N(2)-N(1))*rand(1));
xa = X(1)+round((X(2)-X(1))*rand(1));
xb = (1+erelx)*xa;

% Generate the measurements
metodo_a = [];
for i = 1:na
    e = 2*(0.5-rand(1))*erela*xa;
    metodo_a(i) = xa + e;
end
metodo_b = [];
for i = 1:nb
    e = 2*(0.5-rand(1))*erelb*xb;
    metodo_b(i) = xb + e;
end
disp(sprintf('File %d.html generated',DNI));
comentarios = '<html lang="es"> <body> EXERCISE DATA <br>';
comentarios = strcat(comentarios,sprintf(' ID %d
<br>',DNI));
comentarios = strcat(comentarios,sprintf('<br>Method A
Atomic absorption spectroscopy <br> Fe2O3
(percentage)<br>'));
for i=1:na
comentarios = strcat(comentarios,sprintf('%g
<br>',metodo_a(i)));
end
comentarios = strcat(comentarios,sprintf('<br>Method B
Molecular spectroscopy <br> Fe2O3 (percentage)<br>'));
for i=1:nb
comentarios = strcat(comentarios,sprintf('%g
<br>',metodo_b(i)));
```

```
end
comentarios = strcat(comentarios,'</body> </html>');
fid = fopen(sprintf('%d.html',DNI),'w');
fprintf(fid,'%s',comentarios);
fclose(fid);
```

exercise2eval.m

```
% The ID is stored in variable dni is text form
automatically by GOODLE GMS
DNI = str2num(dni);
% A deterministic seed based on the ID is provided
rand('seed',DNI);
t_distr=[1,12.706;
2, 4.303;
3, 3.182;
4, 2.776;
5, 2.571;
6, 2.447;
7, 2.365;
8, 2.306;
9, 2.262;
10, 2.228;
11, 2.201;
12, 2.179;
13, 2.160;
14, 2.145;
15, 2.131;
16, 2.120;
17, 2.110;
18, 2.101;
19, 2.093;
20, 2.086;
21, 2.080;
22, 2.074;
23, 2.069;
24, 2.064;
25, 2.060;
26, 2.056;
27, 2.052;
28, 2.048;
29, 2.045];
test_f5_doble=[647.8, 799.5, 864.2, 899.6, 921.8, 937.1,
948.2, 956.7, 963.3, 968.6;
38.51, 39.00, 39.17, 39.25, 39.30, 39.33, 39.36, 39.37,
39.39, 39.40;
```

```
17.44, 16.04, 15.44, 15.10, 14.88, 14.73, 14.62, 14.54,
14.47, 14.42;
12.22, 10.65, 9.979, 9.605, 9.364, 9.197, 9.074, 8.980,
8.905, 8.844;
10.01, 8.434, 7.764, 7.388, 7.146, 6.978, 6.853, 6.757,
6.681, 6.619;
8.813, 7.26, 6.599, 6.227, 5.988, 5.820, 5.695, 5.600,
5.523, 5.461;
8.073, 6.542, 5.890, 5.523, 5.285, 5.119, 4.995, 4.899,
4.823, 4.761;
7.571, 6.059, 5.416, 5.053, 4.817, 4.652, 4.529, 4.433,
4.357, 4.295;
7.209, 5.715, 5.078, 4.718, 4.484, 4.320, 4.197, 4.102,
4.026, 3.964;
6.937, 5.456, 4.826, 4.468, 4.236, 4.072, 3.950, 3.855,
3.779, 3.717];
N = [5 10]; %Minimum and maximum number of samples
X = [10 90]; %Minimum and maximum value of the concetration
of the sample

% Randomly determine one of the four different cases
if rand(1)<0.5
    % Both methods have the same mean
    erelx = 0;
else
    % The means differ 5%
    if rand(1)<0.5
        erelx = 0.05;
    else
        erelx = -0.05;
    end
end
if rand(1)<0.5
    % Both methods have a similar variance
    % Maximum measurement errors of 2%
    erela = 0.02;
    erelb = 0.02;
else
    % There are significative differences in the variance
    % Maximum measurement errors of 1% and 4%
    if rand(1)<0.5
        erela = 0.01;
        erelb = 0.04;
    else
        erela = 0.04;
```

```
        erelb = 0.01;
    end
end

% Determine the number of samples and the problem
concentration
na = N(1)+round((N(2)-N(1))*rand(1));
nb = N(1)+round((N(2)-N(1))*rand(1));
xa = X(1)+round((X(2)-X(1))*rand(1));
xb = (1+erelx)*xa;
% Generate the measurements
metodo_a = [];
for i = 1:na
    e = 2*(0.5-rand(1))*erela*xa;
    metodo_a(i) = xa + e;
end
metodo_b = [];
for i = 1:nb
    e = 2*(0.5-rand(1))*erelb*xb;
    metodo_b(i) = xb + e;
end

% Methods A and B are sorted depending on the variance
if std(metodo_a)<std(metodo_b)
    x1 = mean(metodo_b);
    s1 = std(metodo_b);
    n1 = length(metodo_b);
    x2 = mean(metodo_a);
    s2 = std(metodo_a);
    n2 = length(metodo_a);
else
    x1 = mean(metodo_a);
    s1 = std(metodo_a);
    n1 = length(metodo_a);
    x2 = mean(metodo_b);
    s2 = std(metodo_b);
    n2 = length(metodo_b);
end
[x1 s1 n1;x2 s2 n2];

% the variances of the two methods are compared through the
"F-test" or
% "Fischer test".
Ftab=test_f5_doble(n2-1,n1-1);
Fcal=s1^2/s2^2;
```

```
% Fcal and Ftab are compared to determine the appropriate
mean test
if Fcal < Ftab
    % There are not significative differences between the
    variances of the
    % two sets of results
    % The commom variance is used in the test
    s_2 = ((n1-1)*s1^2+(n2-1)*s2^2)/(n1+n2-2);
    t_cal = abs(x1-x2)/sqrt(s_2*(1/n1+1/n2));
    t_tab=t_distr(n1+n2-2,2);
else
    % There are significative differences between the
    variances of the
    % two sets of results
    % The t-test of Cochran is used
    t_cal = abs(x1-x2)/sqrt(s1^2/n1+s2^2/n2);
    gl =
    (s1^2/n1+s2^2/n2)^2/(((s1^2/n1)^2)/(n1+1)+((s2^2/n2)^2)/(n2
    +1))-2;
    gl = round(gl);
    t_tab = t_distr(gl,2);
end

% Correct solution
[Fcal Ftab t_cal t_tab];
comentarios = '<html lang="es"> <body> RESULTS <br>';
comentarios = strcat(comentarios,sprintf(' ID %s
<br><br>',dni));
comentarios = strcat(comentarios,sprintf(' Correct
solution: <br>'));
comentarios = strcat(comentarios,sprintf(' Fcal = %g
<br>',Fcal));
comentarios = strcat(comentarios,sprintf(' Ftab = %g
<br>',Ftab));
comentarios = strcat(comentarios,sprintf(' Tcal = %g
<br>',t_cal));
comentarios = strcat(comentarios,sprintf(' Ttab = %g
<br>',t_tab));
if(t_cal<t_tab)
comentarios = strcat(comentarios,sprintf(' There are
not significative differences <br>'));
else
comentarios = strcat(comentarios,sprintf(' There are
significative differences <br>'));
end
```

```
comentarios = strcat(comentarios,sprintf('<br> Evaluation
criteria:<br>'));
comentarios = strcat(comentarios,sprintf('If the results
and the parameters are correct the grade is 10 <br>'));
comentarios = strcat(comentarios,sprintf('Else the grade is
0 <br>'));
comentarios = strcat(comentarios,sprintf('The maximum error
is 5 percent<br>'));
comentarios = strcat(comentarios,sprintf('<br>'));
nota = 10;
eps = 0.05;
if abs(Fcal-Test_parameters(1))>eps*Fcal
    comentarios = strcat(comentarios,sprintf('Fcal is
    wrong<br>'));
    nota = 0;
end
if abs(Ftab-Test_parameters(2))>eps*Ftab
    comentarios = strcat(comentarios,sprintf('Ftab is
    wrong<br>'));
    nota = 0;
end
if abs(t_cal-Test_parameters(3))>eps*t_cal
    comentarios = strcat(comentarios,sprintf('Tcal is
    wrong<br>'));
    nota = 0;
end
if abs(t_tab-Test_parameters(4))>eps*t_tab
    comentarios = strcat(comentarios,sprintf('Ttab is
    wrong<br>'));
    nota = 0;
end
if (Results == 0)&(t_cal>t_tab)
    comentarios = strcat(comentarios,sprintf('The results
    is wrong<br>'));
    nota = 0;
end
if (Results == 1)&(t_cal<t_tab)
    comentarios = strcat(comentarios,sprintf('The results
    is wrong<br>'));
    nota = 0;
end
comentarios = strcat(comentarios,sprintf('<br>Grade %g
<br>',nota));
```

5.3.3. Ejercicio 3

exercise3data.p

```
DNI=input ('Input ID:');
% A deterministic seed based on the ID is provided
rand('seed',DNI);
t_distr=[1,12.706;
2, 4.303;
3, 3.182;
4, 2.776;
5, 2.571;
6, 2.447;
7, 2.365;
8, 2.306;
9, 2.262;
10, 2.228;
11, 2.201;
12, 2.179;
13, 2.160;
14, 2.145;
15, 2.131;
16, 2.120;
17, 2.110;
18, 2.101;
19, 2.093;
20, 2.086;
21, 2.080;
22, 2.074;
23, 2.069;
24, 2.064;
25, 2.060;
26, 2.056;
27, 2.052;
28, 2.048;
29, 2.045];
N = [5 10]; %Minimum and maximum number of samples
X = [10 90]; %Minimum and maximum value of the concetration
of the sample

% Randomly determine one of the two different cases
% Each case provides a different measurement error for each
method
if rand(1)<1
    erel1 = 0.2;
    erel2 = 0.1;
```

```
else
    erel1 = 0.01;
    erel2 = 0.01;
end

% Number of samples analized
n = N(1)+round((N(2)-N(1))*rand(1));

% Generate the measurements taking into account the error
of each method
dat_met_1 = [];
dat_met_2 = [];
for i = 1:n
    x = X(1)+round((X(2)-X(1))*rand(1));
    e1 = 2*(0.5-rand(1))*erel1*x;
    e2 = 2*(0.5-rand(1))*erel2*x;
    dat_met_1(i) = round(100*(x + e1))/100;
    dat_met_2(i) = round(100*(x + e1 + e2 + 3))/100;
end
disp(sprintf('File %d.html generated',DNI));
comentarios = '<html lang="es"> <body> EXERCISE DATA <br>';
comentarios = strcat(comentarios,sprintf(' ID %d
<br>',DNI));
comentarios = strcat(comentarios,sprintf('<br>Method A <br>
New method<br>Antimony(mg/m3)<br>'));
for i=1:n
comentarios = strcat(comentarios,sprintf('%g
<br>',dat_met_1(i)));
end
comentarios = strcat(comentarios,sprintf('<br>Method B <br>
Standard method<br>Antimony(mg/m3)<br>'));
for i=1:n
comentarios = strcat(comentarios,sprintf('%g
<br>',dat_met_2(i)));
end
comentarios = strcat(comentarios,'</body> </html>');
fid = fopen(sprintf('%d.html',DNI),'w');
fprintf(fid,'%s',comentarios);
fclose(fid);
```

exercise3eval.m

```
DNI = str2num(dni);
rand('seed',DNI);
t_distr=[1,12.706;
2, 4.303;
```

```
3, 3.182;
4, 2.776;
5, 2.571;
6, 2.447;
7, 2.365;
8, 2.306;
9, 2.262;
10, 2.228;
11, 2.201;
12, 2.179;
13, 2.160;
14, 2.145;
15, 2.131;
16, 2.120;
17, 2.110;
18, 2.101;
19, 2.093;
20, 2.086;
21, 2.080;
22, 2.074;
23, 2.069;
24, 2.064;
25, 2.060;
26, 2.056;
27, 2.052;
28, 2.048;
29, 2.045];
N = [5 10]; %Minimum and maximum number of samples
X = [10 90]; %Minimum and maximum value of the concetration
of the sample

% Randomly determine one of the two different cases
% Each case provides a different measurement error for each
method
if rand(1)<1
    erel1 = 0.2;
    erel2 = 0.1;
else
    erel1 = 0.01;
    erel2 = 0.01;
end

% Number of samples analized
n = N(1)+round((N(2)-N(1))*rand(1));
% Generate the measurements taking into account the error
```

```
of each method
dat_met_1 = [];
dat_met_2 = [];
for i = 1:n
    x = X(1)+round((X(2)-X(1))*rand(1));
    e1 = 2*(0.5-rand(1))*erel1*x;
    e2 = 2*(0.5-rand(1))*erel2*x;
    dat_met_1(i) = round(100*(x + e1))/100;
    dat_met_2(i) = round(100*(x + e1 + e2 + 3))/100;
end
d1=dat_met_1;
d2=dat_met_2;

% Calculate the mean of the differences
n=length(dat_met_1);
xd=[];
for i=1:n
    xd_i=((d1(1,i)-d2(1,i)));
    xd=[xd,xd_i];
    xd_i=[];
end
media=mean(xd);

% Calculate T_cal
t_cal=abs(mean(xd))/(std(xd)/sqrt(n));

% Calculate T_tab
t_tab=t_distr(n-1,2);

% Correct solution
[t_cal t_tab];
comentarios = '<html lang="es"> <body> RESULTS <br>';
comentarios = strcat(comentarios,sprintf(' ID %s
<br><br>',dni));
comentarios = strcat(comentarios,sprintf(' Correct
solution: <br>'));
comentarios = strcat(comentarios,sprintf(' Tcal = %g
<br>',t_cal));
comentarios = strcat(comentarios,sprintf(' Ttab = %g
<br>',t_tab));
if(t_cal<t_tab)
comentarios = strcat(comentarios,sprintf(' There are
not significative differences <br>'));
else
comentarios = strcat(comentarios,sprintf(' There are
```

```
significative differences <br>'));
end
comentarios = strcat(comentarios,sprintf('<br> Evaluation
criteria:<br>'));
comentarios = strcat(comentarios,sprintf('If the results
and the parameters are correct the grade is 10 <br>'));
comentarios = strcat(comentarios,sprintf('Else the grade is
0 <br>'));
comentarios = strcat(comentarios,sprintf('The maximum error
is 5 percent<br>'));
comentarios = strcat(comentarios,sprintf('<br>'));
nota = 10;
eps = 0.05;
if abs(t_cal-Parametros_test(1))>eps*t_cal
    comentarios = strcat(comentarios,sprintf('Tcal is
    wrong<br>'));
    nota = 0;
end
if abs(t_tab-Parametros_test(2))>eps*t_tab
    comentarios = strcat(comentarios,sprintf('Ttab is
    wrong<br>'));
    nota = 0;
end
if (Resultado == 0)&(t_cal>t_tab)
    comentarios = strcat(comentarios,sprintf('The result is
    wrong<br>'));
    nota = 0;
end
if (Resultado == 1)&(t_cal<t_tab)
    comentarios = strcat(comentarios,sprintf('The result is
    wrong<br>'));
    nota = 0;
end
comentarios = strcat(comentarios,sprintf('<br>Grade %g
<br>',nota));
```

Capítulo 6

Interfaz web para la generación de problemas de control

En este capítulo se presenta una plataforma de apoyo al profesor para generar de forma sencilla ejercicios de diseño de controladores individualizados auto-evaluables con Goodle GMS a partir de enunciados ya existentes. Mediante un escalado temporal y de ganancia, se garantiza que todos los ejercicios generados son de naturaleza equivalente al problema original. La plataforma permite utilizar material de cursos bien establecidos, de forma individualizada y autoevaluada, facilitando la aplicación de metodologías docentes basadas en la resolución de ejercicios prácticos sin necesidad de tener conocimientos de programación para desarrollar evaluadores.

6.1. Descripción del problema de control

El contenido de una asignatura de fundamentos de control es muy amplio (Dorf and Bishop, 2005), (Guzmán, J.L, Costa, R., Berenguel, M., Dormido, S., 2012), (Ogata, 2011), sin embargo una clase de problemas de particular interés, es el del diseño de controladores para sistemas lineales. La plataforma diseñada se centra en una clase de ejercicios muy concreta, el diseño de controladores para el sistema de control mostrado en la Figura 6.1. La señal $R(s)$ indica el valor deseado para la salida del sistema (referencia), $F(s)$ es un filtro de entrada, $E(s)$ es la señal de error, $C(s)$ es el controlador, $D(s)$ es la señal de perturbación, $U(s)$ es la señal de control, $G(s)$ es la planta del sistema, $Y(s)$ es la señal de salida, $N(s)$ es la salida del ruido y $H(s)$ es la función de transferencia del sensor. Este problema 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 una serie de especificaciones, tanto en el dominio del tiempo como en el dominio de la frecuencia.

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

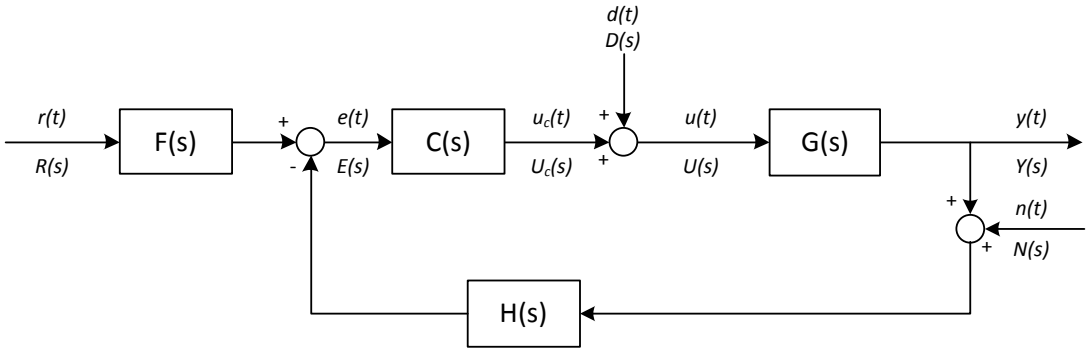

Figura 6.1: Diagrama de bloques del problema básico de diseño de controladores

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.

Respecto a los controladores considerados, puede ser cualquier esquema de control lineal que se pueda definir con una función de transferencia. En particular consideramos controladores PID, aunque se han desarrollado interfaces específicas de redes de adelanto de fase, de retraso de fase y controladores PID.

En la herramienta, también se considera la posibilidad de introducir saturación y retardos de transporte en el sistema en bucle abierto, sin embargo en este caso, la generación de parámetros individualizada no está contemplada.

En el campo del control resulta de particular interés la posibilidad de comparar diferentes controladores en función de algún índice de desempeño. En la herramienta es posible utilizar tres índices, que además cumplan una serie de restricciones, para comparar las soluciones de los controladores, lo que permite abordar el paradigma de evaluación competitiva: evaluar a un alumno considerando los resultados obtenidos por el resto de compañeros. Los índices de desempeño disponibles son:

1. Minimizar la integral del error al cuadrado (ISE): Se minimiza esta integral cuando se desea eliminar errores grandes, ya que estos son los que más contribuyen al valor de la integral.

$$ISE = \int_0^{t_f} e^2(t)dt$$

2. Minimizar la integral del valor absoluto del error (IAE): En este caso se trata de eliminar errores pequeños. Estos errores podrían ser muy dificiles de eliminar minimizando la ISE ya que al elevarlos al cuadrado se hacen todavía más pequeños respecto a los errores

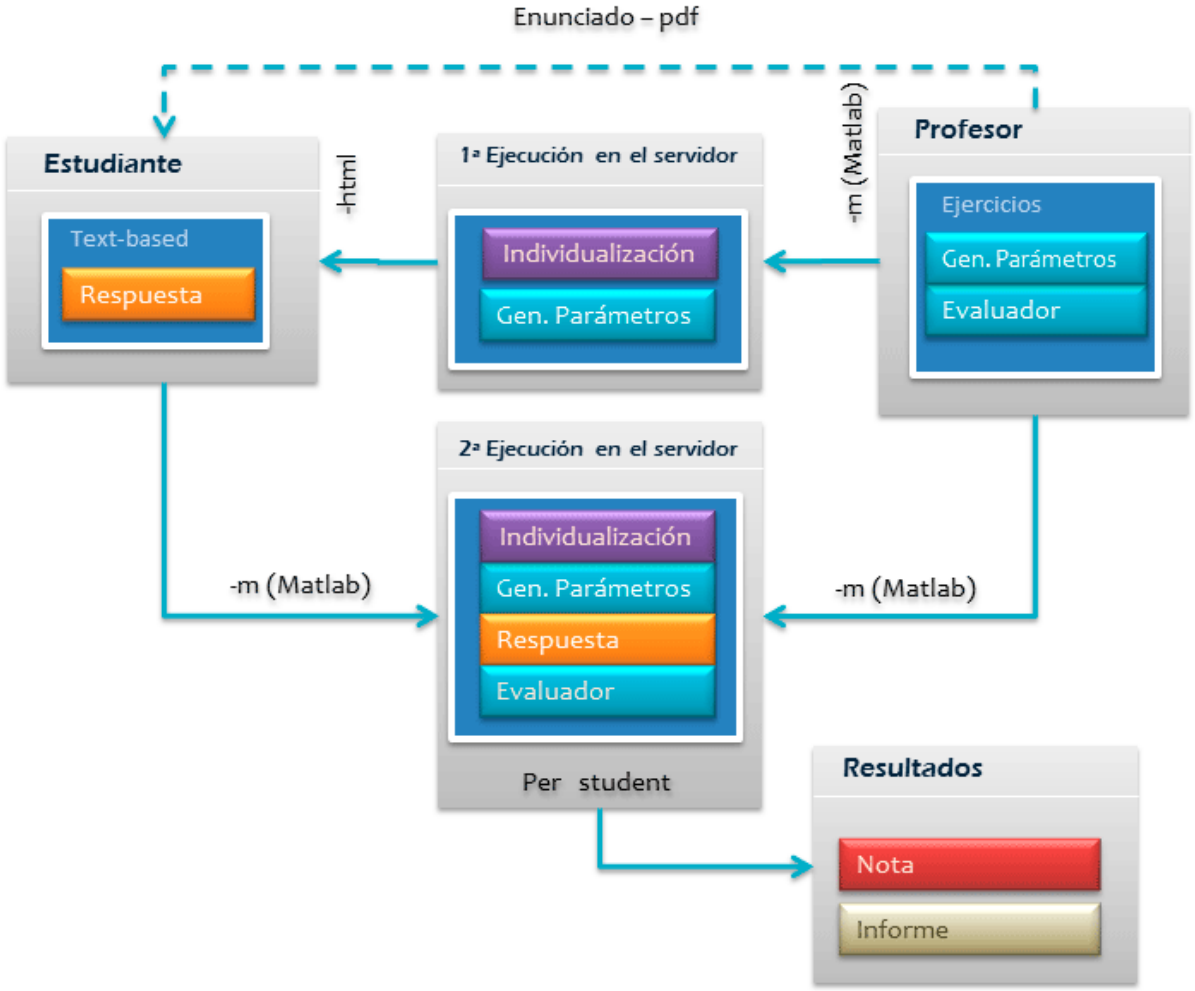

Figura 6.2: Generación de enunciados personalizados en formato HTML

grandes.

$$IAE = \int_0^{t_f} |e(t)|dt$$

3. Minimizar el tiempo de subida.

6.2. Generación de enunciados individualizados

Una característica destacable de la plataforma Goodle GMS es la generación de enunciados personalizados a cada alumno en formato HTML. Esta funcionalidad no existía en la versión original de 2009, por lo que la evaluación y generación de datos de forma automática para los ejercicios de química analítica descritos en el capítulo 5 se realizó mediante la ejecución de un fichero ".p" por parte del alumno.

La actualización efectuada en Goodle GMS permite que el alumno reciba un enunciado personalizado en formato HTML, y se basa en el desarrollo de un cógido MATLAB por parte del profesor (código de generación de parámetros y código evaluador) que se ejecuta al entrar

un alumno en la página de entrega del ejercicio. El código accede a la base de datos, recupera el DNI y genera una única cadena de caracteres que se muestra en el interfaz del alumno como código HTML.

Esto permite personalizar en función del DNI algunos de los parámetros (como por ejemplo la función de transferencia de la planta de un problema de control) utilizando métodos más sofisticados que los explicados previamente, en los que se realizaba una asignación del valor de los parámetros en función de los dígitos del DNI. En particular, en esta sección se explica cómo generar cualquier número de parámetros aleatorios de forma determinista para cada alumno, lo permite su uso en asignaturas de control automático. El funcionamiento de este mecanismo se detalla en la figura 6.2.

Como se ha descrito en la sección anterior, consideramos ejercicios de problemas de diseño de controladores para sistemas lineales definidos por el diagrama de bloques mostrado en la Figura 6.1. Un ejercicio está definido por el modelo de los sistemas, las especificaciones, la estrucutura del controlador y opcionalmente el índice de desempeño objetivo. 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.

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. 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 como por ejemplo (Dorf and Bishop, 2005), (Guzmán, J.L, Costa, R., Berenguel, M., Dormido, S., 2012), (Ogata, 2011).

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 y además, los problemas pueden resultar demasiado parecidos. En la herramienta desarrollada hemos seguido otro procedimiento basado en aplicar dos transformaciones que no afecten la naturaleza del problema base, pero sí a 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:

$$G(s) = A \cdot G_b(\tau s)$$

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

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,

$$t = tb/\tau$$

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

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

Tabla 6.1: Escalado de especificaciones

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

Tabla 6.2: Índices de desempeño Base Normalizado

Un escalado en la escala 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

$$C(s) = C_b(\tau s)/A$$

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. La tabla 6.1 muestra las operación escalada para cada una de las especificaciones consideradas.

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 t_b. 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 6.2 muestra los índices de desempeño normalizados.

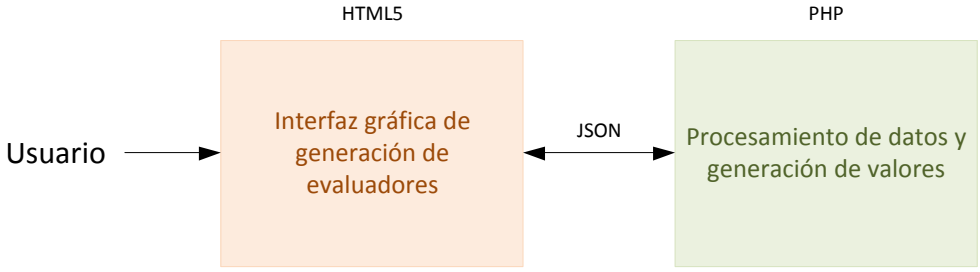

Figura 6.3: Arquitectura HTML/AJAX/PHP de la interfaz

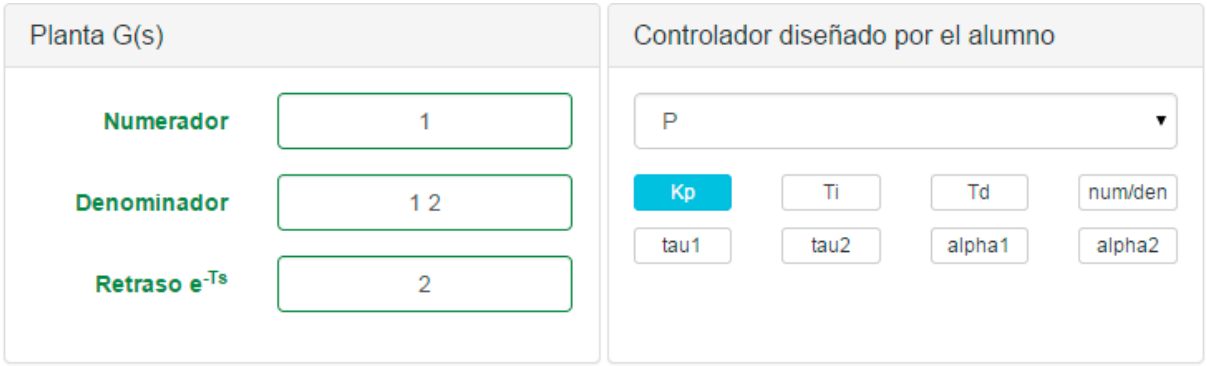

Figura 6.4: Definición de la planta G(s) y el controlador

6.3. 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.

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 6.4 se muestra esta interfaz gráfica.

En la parte inferior se muestran los resultados generados por la plataforma: representación gráfica del sistema nominal, 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

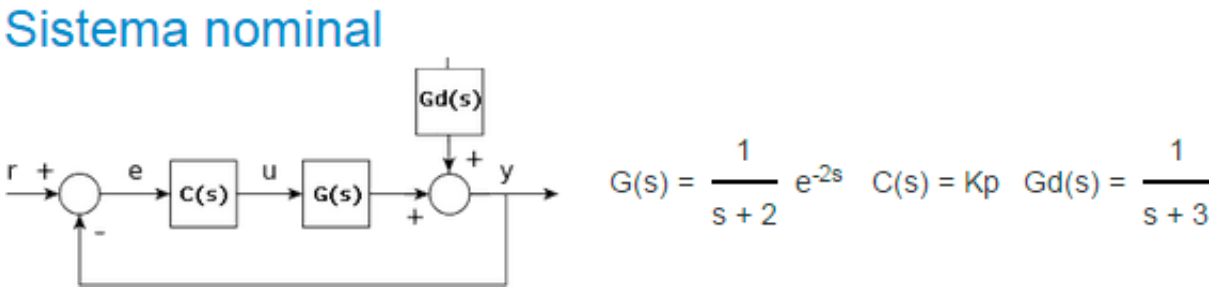

Figura 6.5: Representación gráfica del sistema nominal

Figura 6.6: Menú *Generación de plantas*

plataforma.

La zona superior de la plataforma está dividida en cinco menús: sistema nominal, generación de plantas, especificaciones, objetivo competitivo y exportar.

En el menú *Sistema nominal*, mostrado en la Figura 6.5, 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.

En el menú *Generación de plantas*, mostrado en la Figura 6.6, 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. Esto se consigue generano dos números aleatorios usando como semilla de la función rand de MATLAB el DNI del alumno. Esto permite obtener estos dos números aleatorios de forma determinista, permitiendo que el evaluador pueda generar los mismos parámetros a en el momento de evaluar el trabajo del alumno.

En el menú *Especificaciones*, mostrado en la Figura 6.7, 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 activar la opción de 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 integrar 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 el controlador cumple todas las especificaciones del enunciado. Aquellos controladores que no cumplen las especificaciones se les asigna una figura de mérito negativa para que Goodle GMS los distinga a la hora de calcular la nota final. La nota final se calcula de la siguiente manera. A todos los alumnos que no hayan cumplido las especificaciones se les asigna un 0. El resto de alumnos reciben una nota entre un valor mínimo y máximo definido por el profesor en función de su figura de mérito en relación a la de sus compañeros. El profesor define dos valores identificados por las variables *notamin* y *notamax*. Goodle GMS asigna *notamin* al alumno con una solución con la peor figura de mérito y *notamax* al alumno con la mejor figura de mérito. Al resto se reparten de forma lineal entre ambos valores.

Para ilustrar este concepto pongamos como ejemplo un ejercicio en el que se le exige al alumno diseñar un controlador que minimice el tiempo de subida T_s de un sistema y donde la sobreoscilación máxima permitida $SO(\,\%)$ se establece en 19. La tabla **??** muestra los resultados entregados por 6 alumnos:

Tiempo de subida (s)	**Figura de mérito ($1/T_s$)**	**Sobreoscilación (%)**	**Nota**
1.4	0.71	17	5
1.3	0.77	18	7.5
1.2	0.83	19	10
1.1	0.91	20	0
1.0	1	21	0
1.1	0.91	22	0

Tabla 6.3: Ejemplo cálculo de nota según los resultados de los alumnos

En este ejemplo los tres últimos alumnos no cumplen la especificación exigida, ya que superan el umbral establecido para la sobreoscilación, y se les asigna un 0 como nota. Para los restantes ejercicios se establece la nota máxima (variable *notamax* establecida a 10 en el evaluador) para el alumno con mejor figura de mérito y la nota mínima (variable *notamin* establecida a 5 en el evaluador). La figura de mérito de este ejercicio se define como la inversa del tiempo de subida.

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

6.3.1. Idiomas

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

Nombre	Valor mínimo	Valor máximo	Nota
☑ Sobreoscilación (%)	2	5	1
☐ Tiempo de subida (segundos)			
☑ Tiempo de establecimiento (segundos)	3	4	1
☐ Frecuencia de corte (radianes)			

Figura 6.7: Menú *Lista de especificaciones*

idioma instalados y se muestra un menú en la zona superior de la interfaz para seleccionar el idioma.

6.3.2. Generación del evaluador

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. A continuación se muestra parte del código generado por la plaforma en la inicialización de la semilla para la generación aleatoria de plantas.

```
DNI = str2num(dni);
rand('seed',DNI);
A = Amin + (Amax-Amin)*rand(1);
Tau = Taumin + (Taumax-Taumin)*rand(1);

num_str = '[';
den_str = '[';

for i=1:length(num_pro)
    num(i) = A*num_pro(i)*Tau^(length(num_pro)-i);
    num_str = sprintf('%s %s',num_str,num2str(num(i)));
end
num_str = sprintf('%s %s',num_str,']');

for i=1:length(den_pro)
    den(i) = den_pro(i)*Tau^(length(den_pro)-i);
    den_str = sprintf('%s %s',den_str,num2str(den(i)));
end
den_str = sprintf('%s %s',den_str,']');
```

Estos fragmentos de código MATLAB están diseñados para ser utilizados junto con la aplicación Goodle GMS. Como se ha descrito en la sección de descripción de Goodle GMS, el procedimiento de evaluación consiste en la ejecución sucesiva de código generado por la aplicación, por el alumno y por el profesor. En particular, la aplicación Goodle GMS propor-

ciona de cada alumno, su número de identificación personal (en general el DNI sin letra) en formato cadena de caracteres. Esta es la información utilizada por los códigos de generación de parámetros y de evaluación para individualizar el ejercicio.

Cada vez que el *script* recibe una petición AJAX se inicia el diagrama de flujo reflejado en la Figura 6.8 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 $G_d(s)$. El resultado de esta operación se muestra en la Figura 6.5.

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 abierto 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. Por último, si el profesor ha seleccionado la opción de exportar la práctica el sistema devolverá un fichero en formato *.prac* y comprimido en formato *.zip* compatible con la plataforma Goodle GMS.

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. 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 margen 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.

6.4. 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 tecnologías industriales de la Universidad de Sevilla durante el curso 2013-14 para asignar y evaluar cinco ejercicios diferentes de diseño de controladores PID a los 453 alumnos matriculados en la asignatura. El servidor Goodle GMS se utilizó para corregir de forma automática las más de 1500 entregas recibidas.

El enunciado individualizado para cada alumno definía el numerador y el denominador del sistema en bucle abierto usando sintaxis MATLAB, el tipo de controlador a diseñar y un

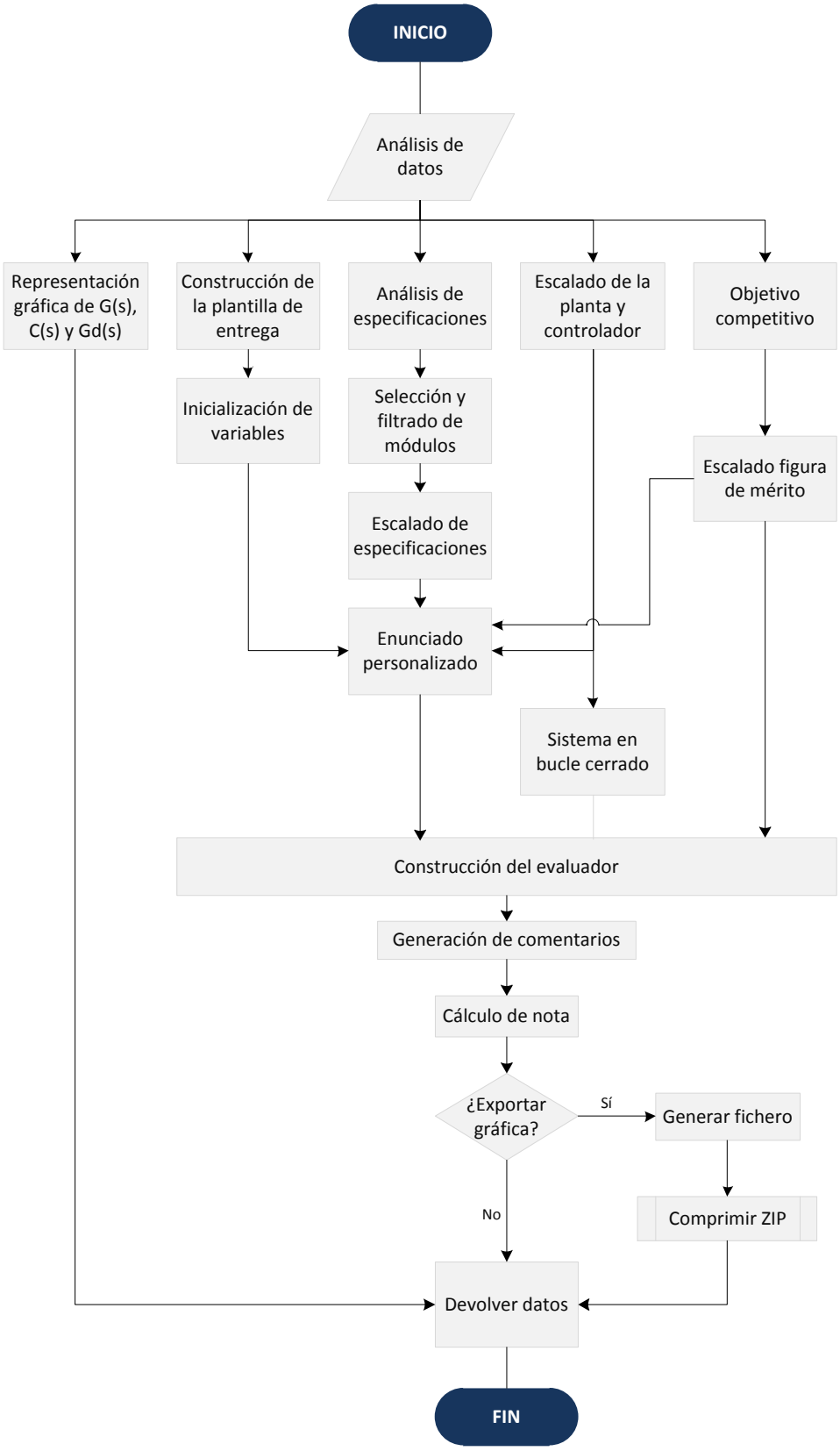

Figura 6.8: Diagrama de flujo para la generación de evaluadores

conjunto de especificaciones. Para el número de identificador 28847022 se generaron de forma automática los siguientes enunciados:

```
Problema 1
Para la planta G1 definida por los siguientes polinomios:
    Numerador = [ 0 0 0 332.8 ];
    Denominador = [ 0.1461 2.774 8.427 0 ];
diseñe un controlador PD que cumpla las siguientes
especificaciones:
    SO < 20
    Ts (10-90%) < 0.1622

Problema 2
Para la planta G2 definida por los siguientes polinomios:
    Numerador = [ 0 0 0 332.8 ];
    Denominador = [ 0.1461 2.774 8.427 0 ];
diseñe un controlador PID que cumpla las siguientes
especificaciones:
    SO < 20
    Ts (10-90%) < 0.1622

Problema 3
Para la planta G3 definida por los siguientes polinomios:
    Numerador = [ 0 0 0 -3.328 ];
    Denominador = [ 0.1461 3.051 13.69 16 ];
diseñe un controlador PD que cumpla las siguientes
especificaciones:
    Mf > 30
    Wc > 11.3925

Problema 4
Para la planta G4 definida por los siguientes polinomios:
    Numerador = [ 0 0 0 -3.328 ];
    Denominador = [ 0.1461 3.051 13.69 16 ];
diseñe un controlador PID que cumpla las siguientes
especificaciones:
    Mf > 40
    Wc > 9.4937

Problema 5
Para la planta G5 definida por los siguientes polinomios:
    Numerador = [ 0 0 20.8 ];
    Denominador = [ 0.2774 5.793 10 ];
diseñe un controlador PI que cumpla las siguientes
especificaciones:
    Mf > 20
```

```
    Evrp < 0.0053
```

Cada problema es resoluble y tiene un procedimiento de solución diferente definido por los cinco problemas base utilizados para definir los ejercicios.

El proceso de asignación, recogida y evaluación de resultados fue llevado a cabo sin ningún problema.

Para cada alumno, el evaluador generó además de la nota, una serie de comentarios con el análisis del controlador entregado. A continuación se muestra el comentario generado para uno de los cinco problemas.

```
Para la planta G2 definida por los siguientes polinomios:

    Numerador = [ 0 0 0 462.3 ];
    Denominador = [ 12.45 53.73 37.09 0 ];

los resultados del controlador diseñado son:

  SO = 17.2497. Sí cumple la especificacion SO
  < 21.00
  Ts(10-90%) = 0.4355. Sí cumple la
  especificacion Ts(10-90%) < 0.75
Nota: 1.0
```

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.

Capítulo 7

Conclusiones

Esta tesis presenta una plataforma de apoyo al profesor para poder ofrecer un sistema de recogida, almacenamiento y evaluación automática de problemas personalizados en asignaturas científico-técnicas. La posibilidad de introducir evaluación automática en un curso tiene un gran potencial desde un punto de vista didáctico. Goodle GMS proporciona una herramienta abierta (sandbox) con el que un profesor puede llevar a cabo desde simples cuestionarios personalizados, hasta la evaluación de complejos problemas de diseño.

Aunque el desarrollo de ejercicios con enunciados y evaluadores automáticos conlleva un gran trabajo por parte del profesor, tanto en formación como en desarrollo, la ventaja de poder reutilizar estos ejercicios para una gran cantidad de alumnos y a lo largo de diferentes cursos, es en general suficiemente para compensarlo. Prueba de esto es que el número de profesores, asignaturas y en resumidas cuentas, ejercicios almacenados en la base de datos de Goodle GMS crece año tras año, habiéndose instalado ya otros dos servidores independientes en la Universidad de Loyola y la Universidad de Lossane, en Francia.

Entre los resultados más señalables merece destacar el desarrollo de una plataforma de evaluación Goodle GMS basada en modelo de caja negra; la implementación de un sistema de generación de enunciados personalizados; la aplicación de la evaluación automática en cursos de control de sistemas no lineales a través de la herramienta Easy Java Simulations; el desarrollo de una interfaz gráfica que abarca todo el ciclo de trabajo del profesor, desde la definición del problema básico hasta la generación de los ejercicios personalizados para cada alumno; y la aplicación de esta plataforma en cursos reales con numerosos alumnos y donde era necesario realizar una recogida sistemática de datos.

7.1. Líneas futuras de investigación

A continuación se enumeran varias posibles líneas de investigación que profundicen el trabajo desarrollado en esta tesis:

1. **Detección de plagio**: Desarrollo de un módulo para Goodle GMS que permita analizar el código fuente entregado por los alumnos y detectar copias. Como primera aproxima-

ción se propone integrar herramientas Open Source tales como *Sherlock* [1], creada por la Universidad de Sydney, o *Tester SIM* [2], desarrollada por la Universidad de Amsterdam.

2. **GNU Octave**: La arquitectura abierta de Goodle GMS permite añadir nuevos componentes a la plataforma. Se propone diseñar una nueva pasarela de comunicación que permita incluir la herramienta *GNU Octave* [3] como un nuevo sistema de evaluación.

3. **Integración con Moodle**: Moodle [4] proporciona a educadores, administradores y estudiantes un sistema integrado único, robusto y seguro para crear ambientes de aprendizaje personalizados. Sus numerosas funcionalidades y la popularidad y difusión que goza en entornos educativos la convierten en un complemento ideal para Goodle GMS.

4. **Históricos**: Implementación de un sistema de históricos que permita archivar información de cursos anteriores y realizar análisis comparativos.

5. **Ayuda al aprendizaje**: Desarrollo de un sistema inteligente de ayuda al aprendizaje, con itinerarios que se ajusten al avance del alumno.

6. **Concurrencia en acceso a MATLAB**: Desarrollo de un mecanismo que garantice el aislamiento de variables en la ejecución de evaluaciones automáticas y la generación automática de enunciados en MATLAB.

7. **Robustez de la plataforma**: Diseño e implementación de un mecanismo que permita reiniciar el servicio de MATLAB y la plataforma Goodle GMS de forma automática.

8. **Escalabilidad de la plataforma**: Goodle GMS está diseñado como un sistema monolítico, por lo que su escalabilidad en momentos de gran actividad se encuentra limitada. Para poder implementar una arquitectura distribuida donde cada componente (servidor web, base de datos, MATLAB, etc.) esté alojado en una máquina diferente (virtual o física) es necesario diseñar una nueva pasarela de comunicación entre la plataforma web y MATLAB.

[1] http://sydney.edu.au/engineering/it/ scilect/sherlock/
[2] http://dickgrune.com/Programs/similarity_tester/
[3] https://www.gnu.org/software/octave/
[4] https://moodle.org

Bibliografía

Apache Open Software Foundation, 2014. Apache server main page.
URL `www.apache.org`

B.S. Heck and Poindexter, S.E. and García R. , 2000. Integrating the web into traditional teaching methods. Proceedings of the American Control Conference 5, 3444–3448.

C. Jara, F. A. Candelas, F. Torres, S. Dormido, F. Esquembre, O. Reinoso, 2009. Real-time collaboration of virtual laboratories through the internet. Computers and Education 52, 126–140.

Casini, M., Prattichizzo, D., Vicino, A., 2003. The automatic control telelab: A user-friendly interface for distance learning. IEEE Transactions on Education 46, 252–257.

Casini, M., Prattichizzo, D., Vicino, A., 2004. The automatic control telelab - a web-based technology for distance learning. IEEE Control Systems Magazine 24, 36–44.

Cedazo, R., Lopez, D., Sanchez, F., Sebastian, J., 2007. Ciclope: Foss for developing and managing educational web laboratories. IEEE Transactions on Education 50 (4), 352 – 359.

Christian, W. y Esquembre, F., 2007. Modelling physics with easy java simulations. The Physics Teacher 45 (8), 475–480.

Clayton, C. A.; Hines, J. W.; Elkins, P. D., 1987. Analytical Chemistry 59 (2506).

Cuadros-Rodríguez, L.; García Campaña, A. M.; Jiménez Linares, C.; Román Ceba, M., 1993. Analytical Letters 26 (1243).

De la Torre, L., Heradio, R., Sanchez, J., Dormido, S., Sanchez, J.P., Carreras, C., Yuste, M, 2011. A thin lens virtual and remote laboratory at the new fislabs portal. MPTL-HSCI.

De la Torre, L., Sanchez, J., Dormido, S., Sanchez, J.P., Yuste, M., Carreras, C., 2011. Two web-based laboratories of the fislabs network: Hookes and snells laws. European Journal of Physics 32, 571–584.

De la Torre, L., Sanchez, J.P., Sanchez, J., Dormido, S., Yuste, M., Carreras, C., 2010. The virtual and remote laboratory for snells law at the fislabs portal. MPTLGIREP-ICPE.

Domínguez, M., Reguera, P., Fuertes, J., 2005. Laboratorio remoto para la enseñanza de la automática en la universidad de león (españa). Revista Iberoamericana de Automática e Informática Industrial 2 (2), 36–45.

Dorf, R., Bishop, R., 2005. Pearson, Inglaterra.

Dormido, S., 2002. Control learning: Present and future. IFAC Annual Control Reviews 28, 115–136.

Dormido, S., Torres, F., 2005. Aplicación de las tic's a la educación en automática. Revista Iberoamericana de Automática e Informática Industrial 2 (2), 3–7.

Dormido, R., Vargas, H., Duro, N., Sánchez, J., Dormido-Canto, S., Farias, G., Esquembre, F., Dormido, S., 2008. Development of a web-based control laboratory for automation technicians: The three tank system. IEEE Transactions on Education 51 (1), 35–44.

Dormido, S., Esquembre, F., Farias, G., y Sánchez, J., 2005. Adding interactivity to existing simulink models using easy java simulations. In Proceedings 44th IEEE CDCECC, 4163–4168.

E. Fabregas, G. Farias, S. Dormido-Canto, S. Dormido, F. Esquembre, 2011. Developing a remote laboratory for engineering education. Computers and Education 57, 1686–1697.

Esquembre, F., 2004. Easy java simulations: A software tool to create scientific simulations in java. Comput. Phys. Commun. 156 (2), 199–204.

Farias, G., 2010. Adding interactive human interfaces to engineering software. Ph. D. Thesis, Computer Science and Automatic Control Dept. UNED.

Farias G., De Keyser R., Dormido S., Esquembre F., 2010. Developing networked control labs: A matlab and easy java simulations approach. IEEE Transactions on Industrial Electronics 57 (10), 3266–3275.

García, M. I., Rodríguez, S., Perez, A., Garcia, A., 2009. p88110: A graphical simulator for computer architecture and organization courses. IEEE Transactions on Education 52 (2), 248 – 256.

Gentil, S.; Exel, M., 2004. Proceedings of 2nd IFAC Conference on IBCE, Grenoble, France.

Gomes, L. y Bogosyan, S., 2009. Current trends in remote laboratories. IEEE Trans. Ind. Electron. 56 (12), 4744–4756.

Guzmán, J., Rodríguez, F., Berenguel, M., Dormido, S., 2005. Laboratorio virtual para la enseñanza de control climático de invernaderos. Revista Iberoamericana de Automática e Informática Industrial 2 (2), 82–92.

Guzmán, J.L, Costa, R., Berenguel, M., Dormido, S., 2012. Pearson, Inglaterra.

H. Vargas, J. Sánchez, N. Duro, R. Dormido, S. Dormido-Canto, G. Farias, S. Dormido, F. Esquembre, C. Salzmann, D. Gillet, 2008. A systematic two-layer approach to develop web-based experimentation environments for control engineering education. Intelligent Automation and Soft Computing 14, 505–524.

HuaQiang, J.; Liang, Z.; WangQiong, Y., 2009. International Conference on Computational Intelligence and Software Engineering, Wuhan, China.

Jiménez, L., Puerto, R., Reinoso, O., Fernández, C., Ñeco, R., 2005. Recolab: Laboratorio remoto de control utilizando matlab y simulink. Revista Iberoamericana de Automática e Informática Industrial 2 (2), 64–72.

Kapur, S., Stillman, G., 1997. Teaching and learning using the world wide web: A case study. Innovations in Education and Training International 34, 316–322.

Kerer, C., Reif, G., Gschwind, T., Kirda, E., Kurmanowytsch, R., Paralic, M., 2005. Shareme: Running a distributed systems lab for 600 students with three faculty members. IEEE Transactions on Education 48 (3), 430 – 437.

Long, G. L.; Winefordner, J. D., 1983. Analytical Chemistry 55 (712A).

M. Barla,M. Bieliková,A. B. Ezzeddinne, T. Kramár, M. Simko,O. Vozár, 2010. On the impact of adaptive test question selection for learning efficiency. Computers and Education 55, 846–857.

MacDougall, D.; Crummett, W. B., 1980. Analytical Chemistry 52 (2242).

Massart, D. L.; Vandeginste, B. G. M.; Buydens, L. M. C.; De Jong, S.; Lewi, P. J.; Smeyers-Verbeke, J., 1998. Handbook of Chemometrics and Qualimetrics: Part A.

Miller, N. J.; Miller, J. C., 2009. Statistics and Chemometrics for Analytical Chemistry. 6th ed. Pearson: Essex.

Muñoz de la Peña, A., Muñoz de la Peña, D., Godoy-Caballero, M., González-Gómez, D., Gómez-Estern, F., Sánchez, C., 2014. Automatic evaluation and data generation for analytical chemistry instrumental analysis exercises. Química Nova.

Muñoz de la Peña, D.; Gómez-Estern, F.; Dormido, S., 2012. Computers & Education 59 (535).

Muñoz de la Peña, D. y Gomez-Estern, F., 2009. A new web-based tool for education and automatic evaluation in control systems engineering. Advances in Control Education ACE'2009. Kumamoto, Japan.

MySQL, 2014. Mysql server main page.
URL `www.mysql.org`

Ogata, K., 2011. Pearson, Inglaterra.

Penn, J. H.; Nedeff, V. M., 2000. Journal of Chemical Education 77 (227).

Petridis, V.; Kazarlis, S.; Kaburlasos, V. G., 2003. IEEE Transactions on Education 46 (102).

PHP, 2014. Hypertext preprocessor main page.
URL `www.php.net`

Poindexter, S. E.; Heck, B. S., 1999. IEEE Control Systems Magazine 19 (83).

Ramis-Ramos, G.; García Alvarez-Coque, M. C., 2001. Quimiometría, Sintesis. Madrid.

Rodríguez, S., Pedraza, J., Dopico, A., Rosales, F., Mendez, R., 2007a. Computer-based management environment for an assembly language programming laboratory. Computer Applications in Engineering Education 15 (1), 41 – 54.

Rodríguez, S., Pedraza, J., Garcia, A., Rosales, F., Mendez, R., 2007b. Computer-assisted assembly language programming laboratory. International Journal of Electrical Engineering Education 44 (3), 216 – 229.

Rodríguez, S., Zamorano, J., Rosales, F., Dopico, A., Pedraza, J., 2007c. A framework for lab work management in mass courses. application to low level input/output without hardware. Computers and Education 48 (2), 153 – 170.

Sanchez, J., Dormido, S., Pastor, R., Morilla, F., 2004. A java/matlab-based environment for remote control system laboratories: Illustrated with an inverted pendulum. IEEE Transactions on Education 47, 321–329.

Sanchez, J., Morilla, F., Dormido, S., Aranda, J., Ruiperez, P., 2006. Virtual and remote control labs using java: a qualitative approach. IEEE Control Systems Magazine 22, 8–20.

Sánchez, A., Escaño, J., Muñoz de la Peña, D., Gómez-Estern, F., 2013. 3d simulator of industrial systems for control education with automated assessment. 10th IFAC Symposium Advances in Control Education, At University of Sheffield, Sheffield, United Kingdom 10.

Sánchez, C., Gomez-Estern, F. y Muñoz de la Peña, D., 2012. A virtual lab with automatic assessment for nonlinear controller design exercises. IFAC Symposyum on Advances in Control Education.

Stewart, B.; Kirk, R.; LaBrecque, D.; Amar, F. G.; Bruce, M. R. M., 2006. Journal of Chemical Education 83 (494).

Tartaglia, A.; Tresso, E., 2002. IEEE Transactions on Education 45 (268).

Torres, F., Candelas, F. A., Puente, S. T., Pomares, J., Gil, P., Ortiz, F. G., 2006. Experiences with virtual environment and remote laboratory for teaching and learning robotics at the University of Alicante. International Journal of Engineering Education 22, 766–776.

van der Wende, M. C., 2000. Higher Education in Europe 25 (305).

Vician, C.; Charlesworth, P., 2003. Journal of Chemical Education 80 (1333).

Zuluaga, C., Sánchez, C., Rodríguez, E., 2005. Laboratorio de automática vía internet (lavi). Revista Iberoamericana de Automática e Informática Industrial 2 (2), 30–35.