PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Desarrollo de un Entorno Virtual para Simulación de un Proceso Hidráulico de 4 Tanques Acoplados

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGISTER EN INGENIERIÍA DE CONTROL Y AUTOMATIZACIÓN

AUTOR

Ing. CRUZ OLANO, Willians Cristhian

ASESOR:

Ph. D. SOTOMAYOR MORIANO, Juan Javier

Setiembre, 2017

Dedicatoria:

El presente trabajo está dedicado a mi familia y amigos, en especial a mis queridos padres por su apoyo constante y motivación para ser mejor a lo largo de mi vida.

Agradecimientos:

Quiero expresar mi agradecimiento a todos los profesores de la Maestría de Ingeniería de Control y Automatización por sus lecciones profesionales y personales, en especial a mi asesor el Dr. Javier Sotomayor. Así como también, y no menos importante, a cada una de las personas que en determinados momentos de este proceso colaboraron con sus consejos y apoyo a mi trabajo.

Agradecimiento especial:

El autor de este trabajo de tesis, “Willians Cristhian Cruz Olano”, agradece la subvención de FONDECYT-CONCYTEC a través del convenio 2015-034 FONDECYT, en el marco del cual se desarrolló la presente tesis: “DESARROLLO DE UN ENTORNO VIRTUAL PARA SIMULACIÓN DE UN PROCESO DE 4 TANQUES ACOPLADOS”.

RESUMEN

El objetivo principal es desarrollar un Entorno Virtual de Simulación para un proceso MIMO no lineal de 4 tanques acoplados con fines educativos para realización de experiencias de laboratorio prácticas, el cual posibilite flexibilidad de configuraciones y se encuentre basado en software libre.

Para cumplir con este objetivo, en primer lugar se realiza el estudio de la simulación de sistemas en lo relacionado a procesos con tanques acoplados, esto con la finalidad de definir las herramientas software adecuadas para el desarrollo del Entorno Virtual de Simulación.

Luego, se hace una revisión teórica del modelado para procesos no lineales de 4 tanques acoplados en distintas configuraciones para la elección del modelo a implementar en la simulación.

El entorno de Simulación se desarrolla en lenguaje Java bajo el esquema MVC (modelo – vista - controlador) que incluye la implementación del modelo del proceso elegido, y de la interfaz visual agregándole flexibilidad en las configuraciones a obtener mediante la modificación de parámetros. Se desarrolla un controlador clásico PID para el control de las alturas de los tanques inferiores con la variación de la apertura de las bombas. Finalmente se proporciona al entorno de la capacidad de integración con Matlab/Simulink como una forma de potencializar las capacidades del mismo. Para tal fin se implementa un controlador en Matlab integrándolo con el entorno de simulación.

Para validar el funcionamiento del desarrolla en Java se realizaron pruebas de comparación con datos de simulación de una implementación en Matlab/Simulink, obteniéndose resultados muy similares. Adicionalmente se realizan pruebas con el entorno desarrollado para comprobar su correcto funcionamiento ante diversas situaciones.

ÍNDICE

INTRODUCCIÓN 1
CAPÍTULO 1. HERRAMIENTAS DE SIMULACIÓN Y EL PROCESO DE 4 TANQUES ACOPLADOS . 3
1.1 Introducción 3
1.2 Herramientas de Simulación enfocadas a Ingeniería de Control 3
1.2.1 Generalidades 3
1.2.2 Software Comercial 3
1.2.3 Software Libre 5
1.3 JAVA / EJS 7
1.4 El proceso de 4 tanques acoplados 9
1.5 Trabajos relacionados 11
1.5.1 The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero. 11
1.5.2 Modeling of the Modified Quadruple-Tank Process. 11
1.5.3 The Quadruple-Tank Process: An Interactive Tool for Control Education. 11
1.5.4 El sistema de tres tanques: un laboratorio virtual y remoto usando Easy Java Simulations. 12
1.5.5 An Educational Plant Based On the Quadruple-Tank Process. 13
1.5.6 Plataformas Interactivas de Experimentación Virtual y Remota: Aplicaciones de Control y Robótica 14
1.5.7 Modeling and Simulation of a Modified Quadruple Tank System 14
1.6 Objetivos 15
Objetivo General 15
Objetivos Específicos 15
1.7 Conclusiones Parciales 15
CAPÍTULO 2. MODELO MATEMÁTICO DEL PROCESO 17
2.1 Introducción. 17
2.2 Configuraciones del Sistema. 17
2.2.1 Generalidades 17
2.2.2 Configuración básica 17
2.2.1 Configuración modificada 18
2.3 Modelado Físico del Sistema 19
2.3.1 Modelo a implementar. 19
2.3.2 Modelado matemático del caudal de fuga en un tanque de agua por un orificio. 19

2.3.3 Modelado de la Configuración básica 21
2.3.4 Modelado Configuración modificada 23
2.4 Linealización del Sistema 24
2.4.1 Finalidad 24
2.4.2 Modelo linealizado de la configuración básica 24
2.4.3 Modelo linealizado de la configuración modificada 25
2.5 Conclusiones Parciales 26
CAPÍTULO 3. DESARROLLO DEL ENTORNO DE SIMULACIÓN 27
3.1 Introducción. 27
3.2 Generalidades. 27
3.3 Características y requerimientos básicos de trabajo para la ejecución del Entorno de Simulación. 28
3.4 Proceso de desarrollo. 29
3.5 Funcionamiento. 30
3.6 Características de la Implementación. 31
3.6.1 Elemento Modelo de MVC 31
3.6.2 Elemento Controlador de MVC 34
3.6.3 Elemento Vista de MVC 35
3.7 Comunicación con MATLAB. 37
3.8 Implementación de una estrategia de control basada en el control clásico PID 38
3.9 Flexibilidad de Configuraciones. 41
3.10 Conclusiones Parciales 42
CAPÍTULO 4. PRUEBAS DE FUNCIONAMIENTO Y VALIDACION DEL DESARROLLO 43
4.1 Introducción 43
4.2 Simulación en MATLAB/Simulink para obtención de data de validación del Entorno desarrollado 43
4.3 Validación del Entorno de Simulación implementado en Java. 46
4.4 Pruebas de funcionamiento del sistema 47
4.4.1 Pruebas de mecanismos de seguridad de cierre de bombas y apertura de orificios de fuga en caso de activación de algún sensor de nivel alto (40cm) 47
4.4.2 Pruebas de efectividad del controlador en función de la apertura de las válvulas de 3 vías. 48
4.4.3 Pruebas de funcionamiento del controlador y sistema para la configuración modificada de 4 tanques acoplados 50
4.5 Análisis de resultados 52

CONCLUSIONES. 53
RECOMENDACIONES 54
BIBLIOGRAFÍA. 55
ANEXO A: MANUAL DE USUARIO i
A.1. INSTALACIÓN: i
A.1.1. JAVA i
A.1.2. API EJS (Easy Java Simulations) ii
A.1.3. API SoftwareLinks (RPC Matlab Client y Server) iii
A.2. USO DEL ENTORNO DE SIMULACIÓN COMO USUARIO: v
A.3. USO DEL ENTORNO DE SIMULACIÓN COMO DESARROLLADOR: vi
A.3.1. Adición / Modificación de variables del desarrollo. vii
A.3.2. Inicialización de variables, parámetros o conexiones. viii
A.3.3. Modificación de la lógica de control o comportamiento dinámico del proceso. ix
A.3.4. Modificación de restricciones. xi
A.3.5. Modificación de la configuración de conexión con Matlab. xii
A.3.6. Modificación de la Vista e interacciones de usuario. xii

INTRODUCCIÓN

En el campo de la educación en ingeniería, en especial en control automático, la parte práctica juega un papel importante en el aprendizaje, usualmente se da en laboratorios, ya sea en plantas piloto físicas implementadas con fines educativos, o en entornos de simulación que buscan asemejarse lo mejor posible a una planta real.

Este tipo de plataformas (plantas piloto o entornos de simulación) facilita la aplicación de los conocimientos teóricos adquiridos y permite el desarrollo de habilidades cognitivas de reforzamiento y metodológicas de la aplicación real práctica. Es innegable el hecho de que los laboratorios convencionales (plantas piloto) constituyen una excelente herramienta que permite a los estudiantes la experimentación con equipamiento real. Por otro lado, estos laboratorios en comparación con un Entorno virtual requieren equipamiento, mantenimiento constante, consumo de energía, espacio; tienen las limitaciones del tiempo de disponibilidad y la incapacidad de simultaneidad de uso por varios estudiantes (Fabregas, 2013).

En relación al párrafo anterior, si bien los Entornos de Simulación no son capaces de brindar todas las capacidades de una planta piloto pues existen situaciones de aprendizaje que solo podemos experimentar al trabajar con instrumentación real, podemos y debemos considerar a los entornos de simulación y las plantas piloto como herramientas complementarias para el aprendizaje.

Respecto a los sistemas o procesos elegidos para implementación práctica, suelen elegirse procesos ampliamente conocidos, que sean a su vez interesantes por sus cualidades inherentes, adecuadas para la aplicación en este caso de determinadas estrategias de control y/o diagnóstico de fallos, o para la observación de determinados comportamientos teóricos. Es así que se trabaja con un proceso de 4 tanques acoplados, un proceso MIMO (múltiples entradas, múltiples salidas) con comportamiento no lineal y con fuerte acoplamiento entre sus variables.

Actualmente, a mediados del año 2017, en la Pontificia Universidad Católica del Perú (PUCP) se viene llevando a cabo el proyecto de implementación de una planta piloto del proceso de 4 tanques acoplados en el Laboratorio de Control y Automatización. Se ha formado un grupo de trabajo alrededor de dicho tema de estudio, enfocado principalmente en el Control Automático y Diagnóstico de Fallas. Hecho que incrementa la importancia de un entorno virtual de pruebas para los trabajos a realizarse, pues aún no se contará con la planta piloto totalmente funcional y probada en un breve tiempo. No obstante, se enfoca el desarrollo del Entorno de Simulación propuesto como complementario a la planta piloto de futura implementación.

En el capítulo 1, se estudian las herramientas de simulación de sistemas y/o procesos en el área del control, enfocando principalmente en Java y el conjunto de librerías

para simulaciones de control EJS, asimismo se muestran los trabajos relacionados de mayor relevancia que sirven de base para el enfoque del presente desarrollo.

En el capítulo 2, se revisan las principales configuraciones existentes para la implementación de un proceso no lineal de 4 tanques acoplados, adicionalmente se estudia el modelado teórico de las mismas, con la finalidad de elegir el modelo a implementar en la simulación.

En el capítulo 3, se desarrolló el entorno de Simulación en lenguaje Java bajo el esquema MVC (modelo – vista - controlador) que incluye la implementación del modelo del proceso elegido, y de la interfaz visual agregándole flexibilidad en las configuraciones a obtener mediante la modificación de parámetros. Se desarrolló un controlador clásico PID en Java para el control de las alturas de los tanques inferiores con la variación de la apertura de las bombas. Finalmente se proporcionó al entorno de la capacidad de integración con Matlab/Simulink como una forma de potencializar las capacidades del mismo. Para tal fin se implementó un controlador en Matlab integrándolo con el entorno de simulación.

Finalmente, para validar el funcionamiento del desarrollo en Java se realizaron pruebas de comparación con datos de simulación de una implementación en Matlab/Simulink, obteniéndose resultados muy similares. Adicionalmente se realizaron pruebas con el entorno desarrollado para comprobar su correcto funcionamiento ante diversas situaciones.

CAPÍTULO 1. HERRAMIENTAS DE SIMULACIÓN Y EL PROCESO DE 4 TANQUES ACOPLADOS

1.1 Introducción

En este capítulo se realiza el estudio teórico de las herramientas de simulación factibles de utilizar para el desarrollo del entorno virtual de simulación; además, se estudia el modelo físico-matemático del proceso de 4 tanques acoplados a implementar. Posteriormente, se muestran algunos de los trabajos relacionados más relevantes por su relación con el presente. Finalmente, se presentan los objetivos a alcanzar en el presente trabajo.

1.2 Herramientas de Simulación enfocadas a Ingeniería de Control

1.2.1 Generalidades

Existe multitud de herramientas utilizadas para el desarrollo de simulaciones en el campo de la Ingeniería de Control y la Robótica, entre lenguajes de programación de propósito general, software específico para simulación, programas de diseño, entre otros; no obstante, existen dos grandes grupos: aquellas desarrolladas con fines comerciales, cuyas licencias son por lo general bastante costosas, y otras desarrolladas en base a software libre. Son estas últimas las de prioridad en el desarrollo del presente trabajo debido al ahorro significativo que proporcionan y a que no imponen restricción alguna para su uso.

1.2.2 Software Comercial

Claramente proporcionan mayores capacidades para la construcción de simulaciones con sistemas o estrategias de control complejas respecto de las basadas en software libre. Son las más conocidas, desarrolladas con propósitos específicos en su mayoría y poseen gran cantidad de información disponible. A continuación se mencionan brevemente dos de los programas más utilizados y conocidos para simulación en el campo del control automático.

MATLAB / Simulink.

Desarrollado por MathWorks, Simulink® es un entorno de diagramas de bloque para la simulación multi-dominio y el diseño basado en modelos. Admite el diseño y la simulación a nivel de sistema, la generación automática de código, así como prueba y verificación continuas de los sistemas embebidos. Simulink ofrece un editor gráfico, bibliotecas de bloques personalizables y solvers para modelar y simular sistemas dinámicos. Se integra con MATLAB®, lo que permite incorporar algoritmos desarrollados en el mismo en los modelos y exportar los resultados de la simulación

a su espacio de trabajo para llevar a cabo análisis a mayor profundidad (es.mathworks.com).

Es una de las herramientas software de mayor uso en el ámbito del control automático, se basa en el uso de una gran variedad de toolbox o conjuntos de herramientas relacionadas para determinados fines. Control System Toolbox™ es el relacionado al campo de la ingeniería de control, proporciona algoritmos y apps para analizar, diseñar y ajustar sistemas de control de forma metódica. Puede especificar su sistema como una función de transferencia, como un sistema de espacio de estados, ceros, polos y ganancia o modelo de respuesta en frecuencia. Las apps y las funciones, tales como los diagramas de respuesta escalón y los diagramas de Bode, permiten analizar y visualizar el comportamiento del sistema en los dominios del tiempo y la frecuencia. (es.mathworks.com).

A continuación en la Figura 1.1 se observa el logo de la conocida herramienta.

Figura 1.1: Logo Matlab / Simulink

Labview.

Desarrollado por National Instruments. LabVIEW es un entorno de desarrollo integrado y diseñado específicamente para ingenieros y científicos. Nativo de LabVIEW es un lenguaje de programación gráfica que utiliza un modelo de flujo de datos en lugar de líneas secuenciales de código de texto, lo que le permite escribir código funcional utilizando un diseño visual que se asemeja al proceso de pensamiento.

Como se resume en su web (http://www.ni.com/es-cr/shop/labview/): “El entorno de programación de LabVIEW simplifica la integración de hardware para aplicaciones de ingeniería, así usted tiene una manera consistente de adquirir datos desde

hardware de NI y de terceros. LabVIEW reduce la complejidad de la programación, así usted puede enfocarse en su problema de ingeniería. LabVIEW le permite visualizar resultados inmediatamente con la creación integrada de interfaces de usuario de clic-y-arrastre y visualizadores de datos integrados. Para convertir sus datos adquiridos en resultados del negocio reales, usted puede desarrollar algoritmos para análisis de datos y control avanzado con IP de matemáticas y procesamiento de señales o reutilizar sus propias bibliotecas desde una variedad de herramientas. Para garantizar la compatibilidad con otras herramientas de ingeniería, LabVIEW puede interactuar o reutilizar bibliotecas de otros software y lenguajes de fuente abierta".

En la Figura 1.2 se observa el logo característico del mencionado software de ingeniería.

Figura 1.2: Logo Labview

1.2.3 Software Libre

Las herramientas pertenecientes a esta clasificación presentan como su principal ventaja no requerir de costosas licencias, aunque muchas veces al costo de mayor complejidad para el desarrollo con las mismas. No obstante, muchos de estos ofrecen también en la actualidad interesantes prestaciones para la simulación de Sistemas de Control.

Dentro de las opciones encontradas en la bibliografía, se optó por revisar a mayor detalle dos de las más utilizadas, además de viables para el sistema particular a implementar, en la mayor cantidad de trabajos encontrados para simulación de sistemas similares. Es el caso en primer lugar del lenguaje de programación Java en conjunto con el API (Interfaz de programación de aplicaciones) EJS (Easy Java Simulations), y en segundo lugar del lenguaje Phyton en conjunto con el programa de diseño 3D Blender, los mismos que se mencionan a continuación, una tercera opción estudiada fue VRML (Virtual Reality Modeling Language) la cual fue descartada por ser únicamente para desarrollo de aplicaciones web, hecho que limita las capacidades a poder obtener en lo que respecta a poder de procesamiento.

Java / EJS

Desarrollado originalmente por Sun Microsystems, actualmente propiedad Oracle. Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible (java.com). No obstante, a pesar de ser un lenguaje de propósito general, cuenta con diversidad de APIs o conjuntos de librerías desarrolladas para facilitar el desarrollo en diversas áreas específicas, tal es el caso de EJS para la realización de simulaciones.

EJS (Easy Java Simulations) desarrollado por Francisco Esquembre, es un conjunto de herramientas desarrollado íntegramente en Java orientadas a facilitar el desarrollo de Simulaciones con Java, simplificando algunas de las tareas requeridas, nos facilita la construcción de la parte gráfica de la simulación, llamada vista en el marco de trabajo definido en EJS. Se basa para ello en la noción de configuración arbórea o padre-hijo. Además nos proporciona una estructura simplificada para la creación de la lógica de la simulación, esto es, de la descripción física del comportamiento dinámico del sistema. Evidentemente, lo esencial de esta tarea es necesario programarla en Java. Se requieren definir y diseñar las variables y algoritmos que describen el modelo de la simulación. (García. 2011)

En la Figura 1.3 se muestra el logo del conocido lenguaje de programación.

Figura 1.3: Logo Java

Blender / Phyton

Blender es un programa de diseño 3D de código abierto, software libre de gran éxito en su área que viene adquiriendo bastante protagonismo en la actualidad por la simplicidad que ofrece a la hora de modelar objetos, así como por su potencia, ya que aparte de las herramientas de diseño presentes en otros programas de diseño 3D, posee 2 tipos de renderizado: Blender internal y Cycles, que permite obtener resultados mucho más realistas en el renderizado y con mayor facilidad que con el anterior.

Blender también posee un motor de juegos (Blender Game Engine o BGE), que permite interactuar con los diseños realizados usando periféricos como el teclado o el ratón, así como a través de la programación de scripts que gobiernan el comportamiento de los objetos presentes en el entorno de simulación. Blender utiliza Python como lenguaje de programación por lo que lo ideal para el desarrollo de la lógica de simulación es utilizar el mismo. (Sánchez et al. 2012)

Las principales ventajas de Blender se obtienen en el aspecto visual, lo cual explica su creciente uso, pero principalmente para la creación de videojuegos; y en lo que respecta al control automático, principalmente en simulaciones de robótica con brazos articulados.

En la Figura 1.4 se muestran los logos tanto de Blender como de su lenguaje de programación asociado Phyton.

Figura 1.4: Logo Blender / Phyton

1.3 JAVA / EJS

En base a las generalidades mencionadas y la popularidad de los mismos, reflejada en la cantidad de trabajos relacionados desarrollados en cada una de las alternativas mencionadas se eligió la opción de Java + EJS por compromiso entre posibilidades en el aspecto visual y principalmente, en comparación con Blender, por las facilidades para la implementación del modelado y las estrategias de control, esto último algo más complicado de manejar en Blender. Aunque el último ofrece mejores posibilidades visualmente, es más utilizado en simulaciones de robótica y animación de videojuegos.

EJS permite desarrollar simulaciones interactivas de forma completa en tres pasos, utilizando una simplificación del paradigma arquitectónico Modelo – Vista – Controlador o MVC (Dormido et al. 2004). Esto es una arquitectura que implementa de manera independiente el aspecto visual, la lógica del sistema o proceso a implementar, y las relaciones e interacciones entre componentes y acciones de usuario.

- **Modelo**: descripción del modelo matemático en términos de variables y relaciones, es aquí donde se implementan las ecuaciones diferenciales y/o relaciones que definen al proceso estudiado.

- **Vista**: elementos gráficos de visualización de la aplicación que permiten generar una interfaz de comunicación con el usuario de las aplicaciones desarrolladas.

- **Controlador**: encargado de vincular los diversos componentes de la aplicación y de dirigir las acciones a realizarse definidas por acciones del usuario a través de la vista o por la misma lógica del modelo.

En la Figura 1.5 se observan las posibles interacciones entre el usuario y cada uno de los componentes del paradigma MVC utilizado por EJS, se observa como el usuario básicamente interactúa haciendo peticiones al controlador (a través de la vista), la misma que intercambia datos con el modelo para luego enviarlos a la Vista que finalmente muestra la respuesta a la petición inicial del usuario.

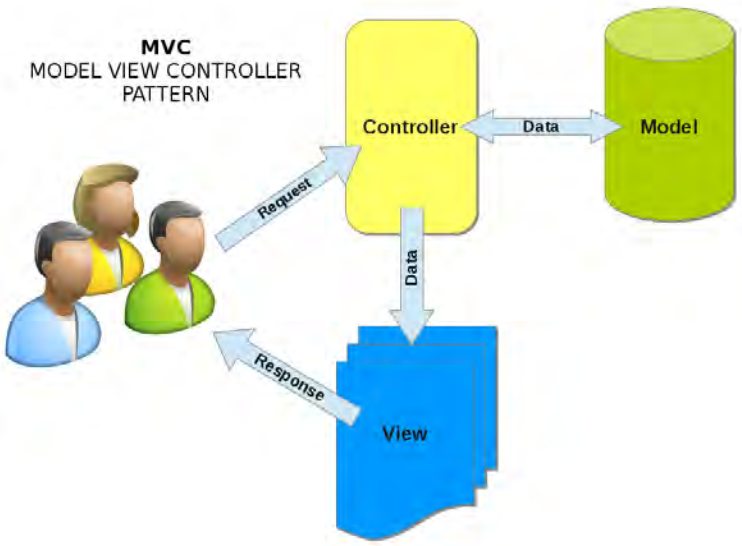


Figura 1.5: Arquitectura base Modelo – Vista – Controlador (MVC) de EJS (perl-diving).

Un aspecto importante es que Java puede integrarse con MATLAB/Simulink, usándolos como el motor numérico interno que describe y resuelve el modelo matemático o la lógica del controlador o el sistema de detección y diagnóstico de fallas, esto mediante el uso de APIs que utilizan el protocolo de comunicación TCP/IP basadas en una arquitectura cliente servidor. Para este fin es requisito tener instalado MATLAB en una máquina de forma local o remota. (Torres. 2006).

En la Figura 1.6 podemos apreciar la interfaz básica de EJS, el mismo que posee en el lado derecho una barra de tareas con iconos para actividades rápidas iniciar un nuevo proyecto, abrir un proyecto existente, guardar los avances actuales o iniciar/detener una simulación. En la parte inferior se observa un área informativa de mensajes de la ejecución tales como advertencias o errores producidos al ejecutar determinadas acciones. Finalmente vemos en la parte principal 3 pestañas. La pestaña Descripción donde se muestra toda la información relacionada al proyecto desarrollado, el Modelo donde se describe toda la lógica del proceso a implementar así como donde se finen las variables utilizadas y la Vista donde se arma toda la parte visual del desarrollo. La lógica del control no posee una pestaña particular, sino que se define a lo largo de estas dos últimas pestañas mencionadas.

Figura 1.6: Interfaz de desarrollo de EJS.

1.4 El proceso de 4 tanques acoplados

Una de las primeras propuestas del Sistema de 4 tanques orientada al control fue la desarrollada por Johansson en el artículo "The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero" publicado el año 2000. La configuración propuesta es la más utilizada en la serie de trabajos posteriores desarrollados al respecto de Sistemas de 4 tanques interconectados, sea directamente o como la base de otras.

Se define en dicha publicación la configuración mostrada a continuación en la Figura 1.7 con los tanques 1 y 2 en la parte inferior y los tanques 3 y 4 en la parte superior y sobre los tanques 1 y 2 respectivamente, el sistema tiene como entradas los voltajes en dos bombas y como salidas las alturas del nivel de líquido en los tanques 1 y 2. Una bomba 1 que alimenta los tanques 1 y 4, repartiendo el flujo por medio de una válvula de 3 vías; y una bomba 2 que hace lo análogo para los tanques 2 y 3 por medio de otra válvula de 3 vías.

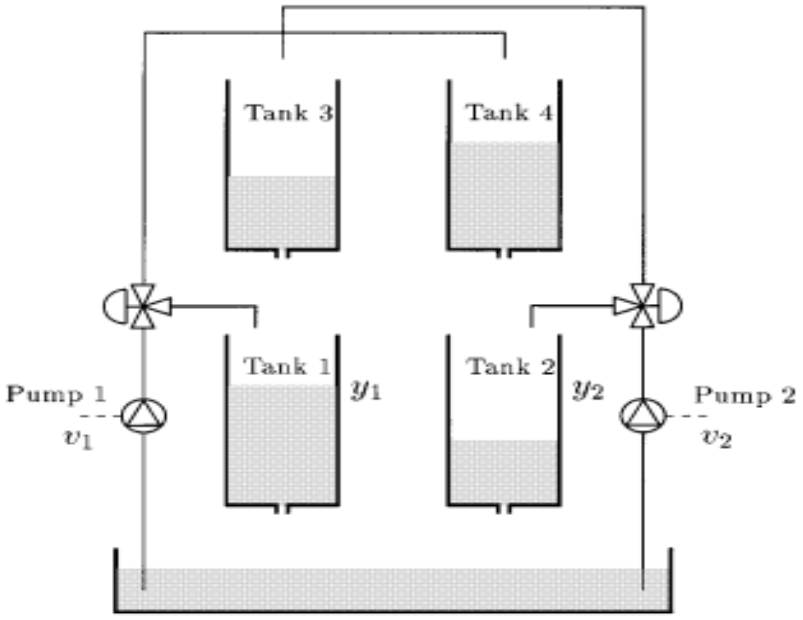


Figura 1.7: Diagrama esquemático del Sistema de 4 tanques (Johansson. 2000)

Posteriormente se propusieron diversas versiones modificadas de esta configuración tales como las de Alvarado et al en 2006 y Numsomran en 2008. Una de las más conocidas e interesantes es la propuesta de Numsomram mostrada a continuación que incluye válvulas de dos vías a la salida de los orificios de fuga de cada uno de los tanques para regular la salida del líquido, así como una conexión entre los tanques inferiores con su respectiva válvula de dos vías para regular el flujo como se observa a continuación en la Figura 1.8:

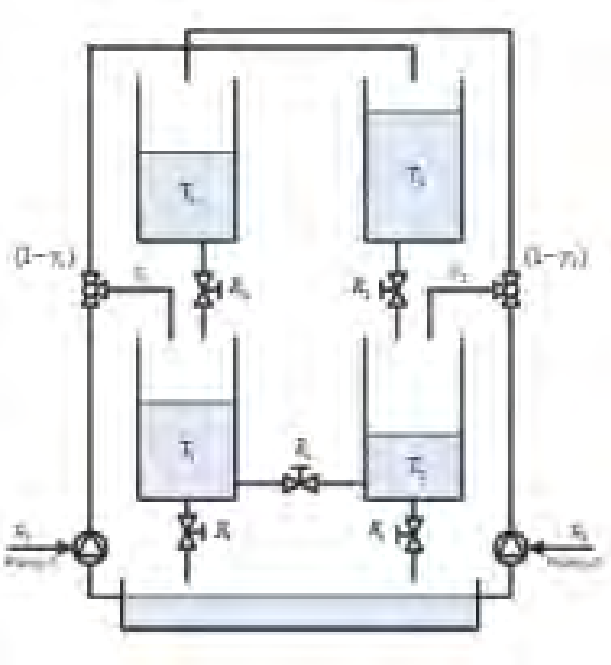


Figura 1.8: Diagrama Sistema de 4 tanques modificado (Numsomram et al. 2008)

1.5 Trabajos relacionados

1.5.1 The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero.

Este trabajo es una de las primeras implementaciones del proceso multivariable, consistente en 4 tanques de agua interconectados o acoplados. En el mismo se muestra que el sistema linealizado tiene un cero multivariable con posibilidad de moverse a lo largo del eje real de las abscisas variando una válvula. Ilustrando con dicho proceso muchos conceptos relacionados al control multivariable, principalmente las limitaciones de desempeño causadas por los polos ubicados en el semiplano derecho. Linealización del modelo y aplicación de control PI. (Johansson, 2000).

1.5.2 Modeling of the Modified Quadruple-Tank Process.

Aquí se presenta el modelado del proceso de 4 tanques modificado. La planta modelo desarrollada es altamente flexible para poder ser ajustada a múltiples configuraciones que pueden ser usados para distintos objetivos de experiencias de control multivariable. Se describe de manera clara la estructura y propiedades físicas del Sistema de 4 tanques modificado y el modelado matemático de la planta. (Numsomran et al. 2008)

1.5.3 The Quadruple-Tank Process: An Interactive Tool for Control Education.

En este trabajo se desarrolla una simulación y animación interactiva por computadora del Sistema de tanques acoplados propuesto por Johansson utilizando EJS y Java. Dicho entorno virtual ilustra los principios básicos del control. Desarrollado con el objetivo de: proveer a estudiantes motivaciones en la práctica de los principios de la Ingeniería de Control, mostrando de modo simple algunas características del control multivariable. Integra habilidades y principios adquiridos por el autor en la solución de problemas prácticos de ingeniería y de este modo motiva a estudiantes a promover una actitud activa usando aprendizaje basado en problemas vía problemas de control abiertos. No obstante, presenta como principal limitación que es una implementación básica del modelo que no permite mayor grado de personalización de la configuración. En la Figura 1.9 se observa la interfaz del desarrollo con los 4 tanques y las tuberías de interconexión entre las mismas, y donde además se observa que se permite variar los voltajes aplicados a las bombas y la apertura de las válvulas de tres vías. (Dormido et al. 2003)

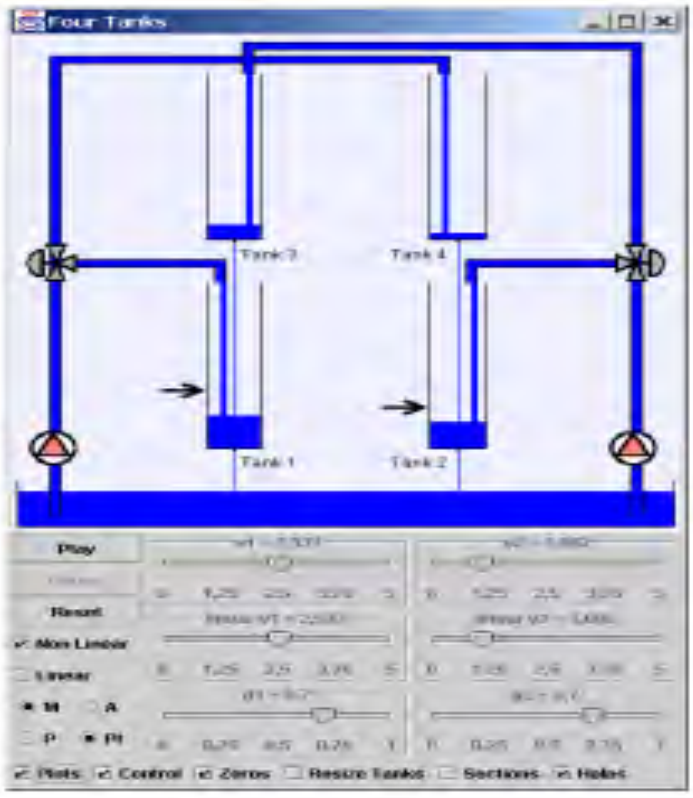


Figura 1.9: Vista principal de la simulación en EJS para el modelo de Johansson (Dormido et al. 2003)

1.5.4 El sistema de tres tanques: un laboratorio virtual y remoto usando Easy Java Simulations.

Este artículo fue publicado en Duro et al. 2005, en el mismo se presenta un desarrollo completo de un laboratorio virtual y remoto para la realización de experiencias sobre un sistema MIMO no-lineal de tres tanques en serie. Permite trabajar en modo simulación y modo remoto con la misma interfaz gráfica y la misma batería de controladores. La interfaz del cliente desarrollada íntegramente usando el sistema de desarrollo conocido como EJS, una herramienta de código abierto (software libre) para generar potentes aplicaciones y applets.

En la figura 1.10 podemos apreciar la interfaz de la aplicación desarrollada se tienen 3 tanques interconectados en serie con el tanque 1 conectado con el tanque 3 y el 3 con el 2, así como un orificio adicional de salida en el tanque 2, se tienen además 2 bombas que alimentan a los tanques 1 y 2. En este sistema se controlan los voltajes de las bombas para alcanzar las alturas deseadas en los tanques 1 y 2.

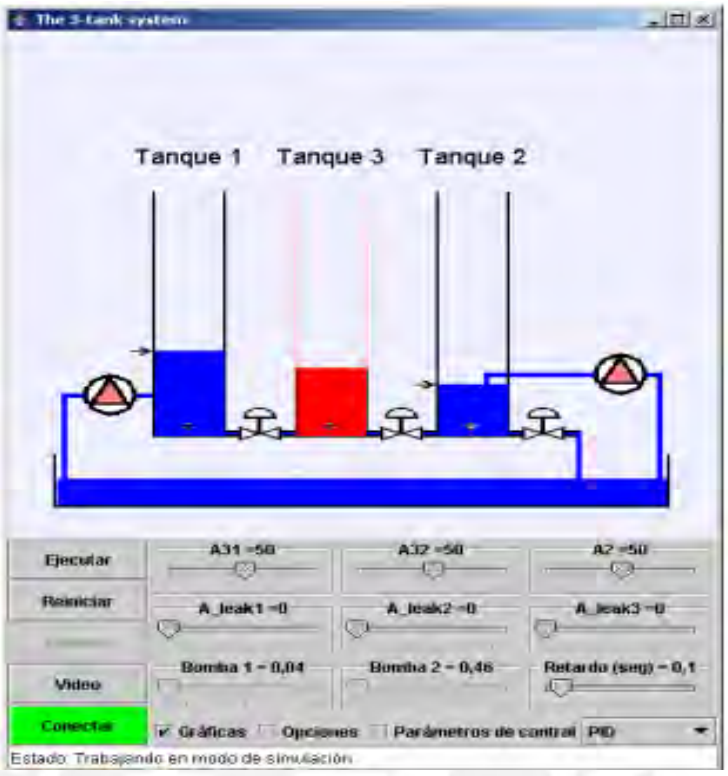


Figura 1.10: Interfaz gráfica con EJS del Sistema de 3 tanques (Duro et al. 2005)

1.5.5 An Educational Plant Based On the Quadruple-Tank Process.

Este trabajo (Alvarado et al. 2006) presenta el desarrollo de un Sistema de 4 tanques desarrollado en la Universidad de Sevilla para educación en Control de Procesos. Esta planta está basada en el modelo de Johansson con algunas modificaciones con objetivo de obtener un amplio rango de aplicaciones.

Consiste en una implementación real donde la estructura original del proceso fue modificada para ofrecer una amplia variedad de configuraciones usadas para educación e investigación. Pudiéndose formar además configuración de un solo tanque, dos o tres tanques interconectados, y en general una mezcla de procesos y dinámicas mixtas. Además, los parámetros de cada tanque pueden ser configurados ajustando la sección transversal del orificio de fuga de cada tanque.

En la Figura 1.11 podemos apreciar los planos de configuración donde apreciamos una gran cantidad de válvulas de dos vías que proporcionan flexibilidad a la configuración al abrirse/cerrarse para permitir/restringir respectivamente el flujo de agua en determinados sentidos.

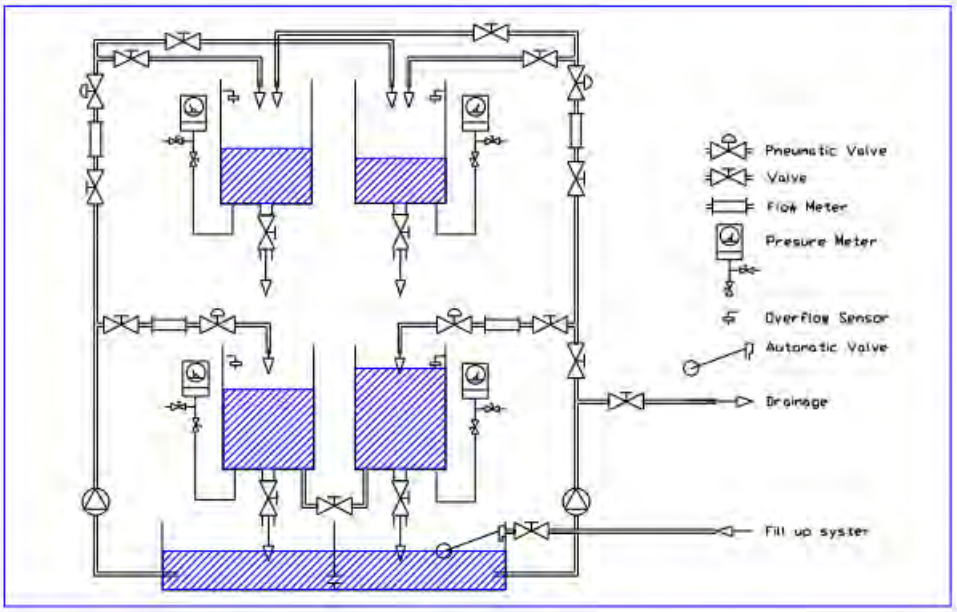


Figura 1.11: Planos de configuración de plantas (Alvarado et al. 2006)

1.5.6 Plataformas Interactivas de Experimentación Virtual y Remota: Aplicaciones de Control y Robótica

El objetivo general de este trabajo (Fabregas, 2013) fue diseñar e implementar plataformas interactivas de experimentación para el desarrollo de prácticas de laboratorio de forma virtual y remota en el ámbito de la enseñanza de la Ingeniería de Control y la Robótica Móvil.

Para ello se desarrolló un entorno virtual en cada uno de los casos, lo cual constituye una simulación de cada proceso que permite realizar experimentos de forma interactiva y sencilla

1.5.7 Modeling and Simulation of a Modified Quadruple Tank System

En este trabajo (Nazrah, 2015) describe el Sistema de 4 tanques modificado, todas las variables importantes y el modelo matemático. Se desarrollan modelos determinísticos y estocásticos usando ecuaciones diferenciales y simulando los modelos con Matlab. Se realiza el análisis de estado estacionario para obtener una ventana de operación para la elección del punto de referencia para una óptima operación.

1.6 Objetivos

Los Entornos Virtuales de Simulación facilitan la aplicación práctica de conocimientos teóricos. En comparación con una planta física presentan como ventajas el no requerir de equipamiento, mantenimiento constante, elevado consumo de energía, grandes espacios; ni tener las limitaciones del tiempo de disponibilidad o la incapacidad de simultaneidad de uso. Si bien los Entornos Virtuales no son capaces de brindar todas las capacidades de una planta física, pues existen situaciones de aprendizaje que solo podemos experimentar al trabajar con instrumentación real, podemos y debemos considerar a los entornos virtuales y las plantas físicas como herramientas complementarias y necesarias en el campo del control automático y en particular en el estudio del proceso de 4 tanques acoplados.

Objetivo General

Realizar el desarrollo e implementación de un Entorno Virtual de Simulación con fines educativos para la realización de experiencias de laboratorio prácticas con un proceso MIMO no lineal de 4 tanques acoplados en distintas configuraciones, basado en software libre.

Objetivos Específicos

Para poder alcanzar este objetivo es necesario realizar los siguientes trabajos de investigación y desarrollo:

- Estudio del modelado teórico del proceso de 4 tanques acoplados.
- Análisis y elección de herramientas de simulación a utilizar.
- Implementación del Entorno Virtual de Simulación con funcionamiento íntegramente en un lenguaje de programación libre.
- Generación de una interfaz visualmente amigable para el entorno propuesto.
- Validación del funcionamiento del desarrollo propuesto con datos de simulación reales.

1.7 Conclusiones Parciales

Se determinó la elección de JAVA por ser una herramienta de software libre con amplias capacidades, además de contar con el conjunto de librerías EJS que facilitan el trabajo de simulaciones en el ámbito del control y por existir inclusive librerías

para integración de JAVA con otras herramientas software especializadas como MATLAB/Simulink.

Del estudio de trabajos relacionados se determinó su capacidad o viabilidad para implementar tanto la lógica de control de procesos MIMO, como de implementar una interfaz visual amigable para procesos similares al de los 4 tanques acoplados.

CAPÍTULO 2. MODELO MATEMÁTICO DEL PROCESO

2.1 Introducción.

En el presente capítulo se hace un estudio del modelado teórico del Proceso o Sistema a implementar, de 4 tanques acoplados. Inicialmente se revisan las principales configuraciones del sistema según las conexiones y válvulas presentes en la implementación, enfocándose finalmente en resumen en 2 variantes generales.

Posteriormente, se explican los fundamentos del modelado físico/matemático para el trabajo con sistemas de tanques y fluidos. Detallando finalmente el modelo obtenido para cada una de las configuraciones estudiadas previamente.

2.2 Configuraciones del Sistema.

2.2.1 Generalidades

El sistema de 4 tanques interconectados es un proceso ampliamente estudiado, a lo largo del tiempo han sugerido modificaciones en lo que respecta a la configuración inicial del modelo de Johansson presentado el año 2000. El presente trabajo busca abarcar una amplia gama de configuraciones a través del uso de opciones de configuración. No obstante, la base para el desarrollo y diseño serán dos de las más estudiadas y completas. Una primera que llamaremos la configuración básica (modelo de Johansson, 2000) y una segunda que denominaremos configuración modificada (utilizada en "Modeling of the Modified Quadruple-Tank Process" Numsomram et al. 2008). A partir de estas, con el cierre y/o apertura de algunas de las válvulas implementadas, es posible obtener otras configuraciones. En este punto en particular solo nos referimos a la distribución de los elementos de la configuración, mas no a sus dimensiones y/o capacidades.

2.2.2 Configuración básica

Compuesta por un reservorio amplio en su base y otros cuatros tanques de menores dimensiones distribuidos en pares en dos niveles, con cada tanque del nivel superior (tanques 3 y 4) encima de su correspondiente en el nivel inferior (tanques 1 y 2). Una bomba 1, que alimenta del reservorio a los tanques 1 y 4, con su respectiva válvula de 3 vías para regular el caudal distribuido a cada tanque; de modo similar una bomba 2 que alimenta los tanques 2 y 3, también con su respectiva válvula de 3 vías. Podemos observar esta configuración en la Figura 2.1.

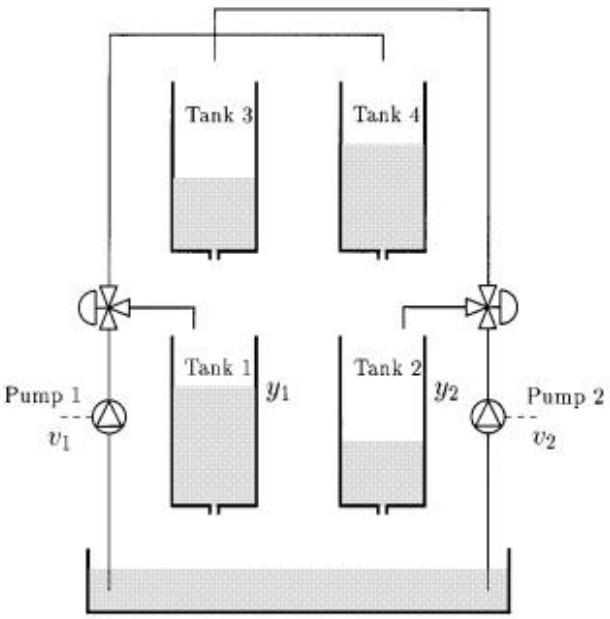


Figura 2.1: Configuración básica Sistema de 4 tanques (Johansson. 2000)

2.2.1 Configuración modificada

Partiendo de la configuración básica, incrementa una conexión a nivel entre los tanques 1 y 2 en la parte más baja de los mismos, regulable mediante la apertura de una válvula de 2 vías. Adicionalmente, agrega una válvula regulable en los orificios de fuga de cada tanque con la finalidad de graduar el flujo a través de los mismos. Podemos observar la configuración mencionada en la Figura 2.2.

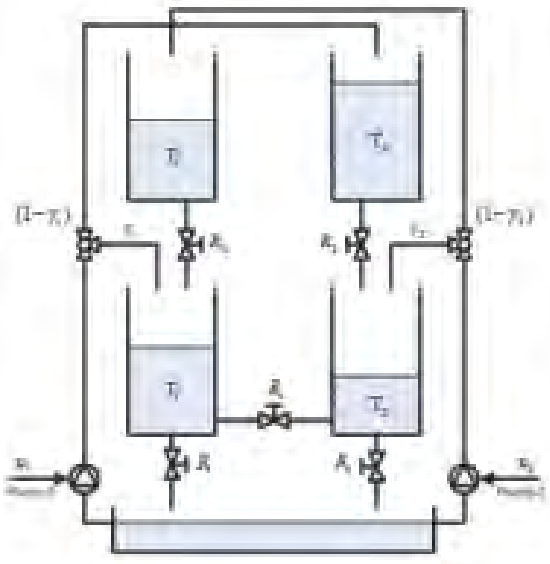


Figura 2.2: Configuración modificada Sistema de 4 tanques (Numsomram et al. 2008)

2.3 Modelado Físico del Sistema

2.3.1 Modelo a implementar.

El modelo de la planta que se implementará en la Simulación será la del modelo físico no lineal obtenido a partir de las leyes físicas de la dinámica de fluidos, pues lógicamente será el que entregue resultados lo más cercanos posibles a una implementación física real. Asimismo se verán los modelos linealizados con fines básicamente de implementación de controladores.

2.3.2 Modelado matemático del caudal de fuga en un tanque de agua por un orificio.

Como se explica al detalle en el trabajo "Modelado y simulación de un sistema interconectado de cuatro tanques, en Labview" (Arias. 2013) Partimos de la ecuación de continuidad basada en la ley de conservación de la masa, asumiendo la incompresibilidad del fluido a causa de las presiones a las que es sometido (agua).

$$q_i = A_i v_i = A_1 v_1 = A_2 v_2$$

La ecuación presentada expresa que el caudal en cualquier punto del tanque (u zona del orificio) es igual a la sección transversal de dicho punto del flujo de agua (A_i) por la velocidad de la corriente en el mismo punto (v_i). Es así que el caudal de descenso en el interior del tanque ($A_1 v_1$) es igual al caudal de salida por el orificio de fuga ($A_2 v_2$). A continuación, podemos apreciar de manera más clara lo descrito previamente con la Figura 2.3.

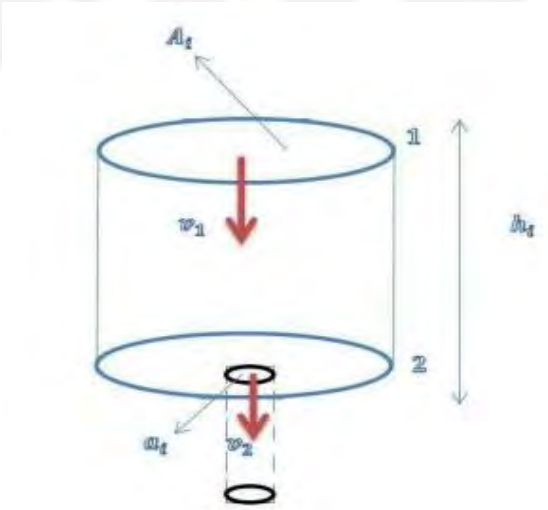


Figura 2.3: Diagrama para obtener la ecuación de Bernoulli (Arias. 2013)

Continuamos con la aplicación de la ecuación de Bernoulli basada en el principio de conservación de la energía indica que:

$$\Delta K + \Delta U = \Delta W$$

La variación de la energía (cinética y potencial) es igual a la variación del trabajo realizado por la fuerza de la presión del líquido.

La variación de la energía cinética viene dada por:

$$\Delta K = \frac{Mv_2^2}{2} - \frac{Mv_1^2}{2}$$

donde M: masa del líquido y v_i velocidad del líquido en el punto i.

La variación de energía potencial por:

$$\Delta K = Mgz_2 - Mgz_1$$

donde g es la aceleración de la gravedad y z_i la altura en el punto i.

Finalmente, el cambio de trabajo (fuerza por distancia) viene dado por

$$\Delta W = p_1 A_1 v_1 t - p_2 A_2 v_2 t$$

Donde p_i es la presión en el punto i (presión igual a fuerza entre área).

Reemplazando en la ecuación de Bernoulli:

$$\frac{Mv_1^2}{2} + Mgz_1 + p_1 A_1 v_1 t = \frac{Mv_2^2}{2} + Mgz_2 + p_2 A_2 v_2 t$$

Por la ley de conservación de la masa, el volumen:

$$V = A_1 v_1 t = A_2 v_2 t$$

Reemplazando V y la densidad del fluido $\rho = M/V$:

$$\frac{\rho v_1^2}{2} + \rho g z_1 + p_1 = \frac{\rho v_2^2}{2} + \rho g z_2 + p_2$$

Para el caso del tanque abierto se considera $p_1 = p_2$= presión atmosférica, considerando $z_1 = h_i$ (altura del fluido en el tanque), $z_1 = 0$ (orificio al ras del tanque) y $v_1 = 0$ (despreciando la velocidad de descenso del líquido en el tanque, orificio muy pequeño respecto de la sección del tanque y para un corto periodo de tiempo), obtenemos despejando el flujo por el orificio de fuga del tanque:

$$q_i = A_v \sqrt{2gh_i}$$

Donde A_v es la sección transversal de la vena contracta formada por la salida del agua, la cual es menor a la sección transversal del orificio de salida a_i y que suele expresarse mediante el Coeficiente de contracción $A_v = C_i a_i$. Para mayor caridad de los términos descritos podemos observar la Figura 2.4.

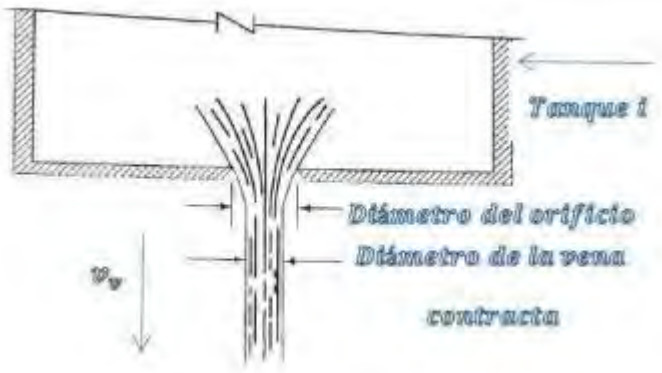


Figura 2.4: Flujo a través de un orificio (Mott. 2006)

2.3.3 Modelado de la Configuración básica

Se aplica para cada tanque, ver figura 2.5 (Alvarado et al. 2006):

$$\Delta Q = \sum Q_{entrantes} - \sum Q_{salientes}$$

$$A_1 \frac{dh_1}{dt} = -q_{fuga\,1} + q_{fuga\,3} + q_1$$

$$A_2 \frac{dh_2}{dt} = -q_{fuga\,2} + q_{fuga\,4} + q_2$$

$$A_3 \frac{dh_3}{dt} = -q_{fuga\,3} + q_3$$

$$A_4 \frac{dh_4}{dt} = -q_{fuga\,4} + q_4$$

$$A_{reservorio} \frac{dh_{reservorio}}{dt} = q_{fuga\,1} + q_{fuga\,2} - q_1 - q_2 - q_3 - q_4$$

Luego en base a lo explicado anteriormente, considerando un coeficiente de contracción igual a la unidad:

$$q_{fuga\,i} = a_i \sqrt{2gh_i}$$

Considerando los flujos entregados en las bombas $(q_a \; y \; q_b)$ y las aperturas de las válvulas respectivas $(\gamma_1 \; y \; \gamma_2)$.

$$q_1 = \gamma_1\, q_a;\; q_2 = \gamma_2\, q_b;\; q_3 = (1-\gamma_2) q_a;\; q_2 = (1-\gamma_1)\, q_b$$

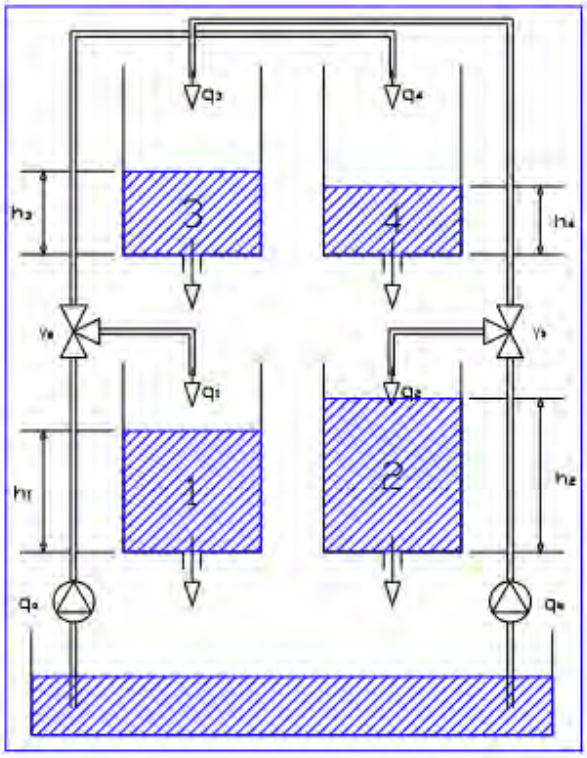


Figura 2.5: Esquema del proceso de 4 tanques (Alvarado et al. 2006)

Finalmente obtenemos el modelo no lineal del sistema (Alvarado et al. 2006):

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1}{A_1} q_a$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2}{A_2}\, q_b$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)}{A_3} q_a$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)}{A_4} q_a$$

Dónde:

A_i: $sección\ transversal\ del\ tanque\ i\ (cm^2)$.

a_i: $sección\ transversal\ del\ orificio\ de\ salida\ del\ tanque\ i\ (cm^2)$

h_i: $nivel\ del\ agua\ en\ el\ tanque\ i\ (m)$

$q_a \, y \, q_b$: $Flujo \; de \; salida \; de \; las \; bombas (m^3/h)$

g : $aceleración \; de \; la \; gravedad \; (m/s^2)$

q_i: $flujo \; de \; entrada \; al \; tanque \; i \; (m^3/h)$

γ_i: $parámetros \; de \; apertura \; de \; las \; válvulas \; de \; 3 \; vías.$

2.3.4 Modelado Configuración modificada

Como se indicó anteriormente básicamente se incluye una conexión "x" entre los tanques 1 y 2 (a nivel y al ras de los mismos) con una válvula reguladora en medio de la conexión. Este modelo modificado además asume que el área de la sección transversal de los tanques es la misma e igual a "A". Además incluye en cada orificio de fuga (incluida la conexión "x") válvulas reguladoras (β_i) Ver figura 2.2

Esto básicamente genera una modificación el comportamiento dinámico del nivel de agua de los tanques 1 y 2:

$$A_1 \frac{dh_1}{dt} = -q_{fuga\,1} + q_{fuga\,3} - q_{fuga\,x} + q_1$$

$$A_2 \frac{dh_2}{dt} = -q_{fuga\,2} + q_{fuga\,4} + q_{fuga\,x} + q_2$$

Dónde:

$$q_{fuga\,x} = \frac{\beta_x a_x}{A} sgn(h_1 - h_2)\sqrt{2g|h_1 - h_2|}$$

Quedando el modelo no lineal resultante (Numsomram et al. 2008):

$$\frac{dh_1}{dt} = -\frac{\beta_1 a_1}{A}\sqrt{2gh_1} + \frac{\beta_3 a_3}{A}\sqrt{2gh_3} - \frac{\beta_x a_x}{A} sgn(h_1 - h_2)\sqrt{2g|h_1 - h_2|} + \frac{\gamma_1}{A} q_a$$

$$\frac{dh_2}{dt} = -\frac{\beta_2 a_2}{A}\sqrt{2gh_2} + \frac{\beta_4 a_4}{A}\sqrt{2gh_4} + \frac{\beta_x a_x}{A} sgn(h_1 - h_2)\sqrt{2g|h_1 - h_2|} + \frac{\gamma_2}{A} q_b$$

$$\frac{dh_3}{dt} = -\frac{\beta_3 a_3}{A}\sqrt{2gh_3} + \frac{(1-\gamma_2)}{A} q_a$$

$$\frac{dh_4}{dt} = -\frac{\beta_4 a_4}{A}\sqrt{2gh_4} + \frac{(1-\gamma_1)}{A} q_a$$

Dónde:

A_i: $sección\ transversal\ del\ tanque\ i\ (cm^2)$.

a_i: $sección\ transversal\ del\ orificio\ de\ salida\ del\ tanque\ i\ (cm^2)$

h_i: $nivel\ del\ agua\ en\ el\ tanque\ i\ (cm)$

$q_a\ y\ q_b$: $Flujo\ de\ salida\ de\ las\ bombas(m^3/h)$

g : $aceleración\ de\ la\ gravedad\ (m/s^2)$

q_i: $flujo\ de\ entrada\ al\ tanque\ i\ (m^3/h)$

γ_i: $parámetros\ de\ apertura\ de\ las\ válvulas\ de\ 3\ vías.$

β_i: $parámetros\ de\ apertura\ de\ las\ válvulas\ de\ salida\ del\ tanque\ i.$

β_x: $parámetros\ de\ apertura\ de\ las\ válvulas\ de\ conexión\ entre\ tanques\ 1\ y\ 2.$

2.4 Linealización del Sistema

2.4.1 Finalidad

El objetivo de linealizar es básicamente para temas de control, pues finalmente en el entorno simulado del sistema en si se trabaja con el modelo no lineal con el objetivo de buscar la mejor aproximación posible al comportamiento real. Para representar el sistema linealizado en forma de ecuaciones de espacio de estado de la forma:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Dado que el sistema es no lineal, es decir $\dot{h} = f(h, u)$ donde f es una función no lineal. Linealizamos el modelo alrededor de un punto de operación (h_i^0, q_j^0).

2.4.2 Modelo linealizado de la configuración básica

Linealizamos el modelo obtenido en la sección anterior en un punto de operación dado por (h_i^0, q_j^0) donde $j = a, b\ e\ i = 1,2,3,4$. Definiendo las variables:

$$x_i = h_i - h_i^0;\ u_j = q_j - q_j^0$$

El modelo linealizado del sistema para entradas q_a, q_b queda como (Alvarado et al. 2006):

$$\frac{dx}{dt} = \begin{bmatrix} \frac{-1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & \frac{-1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & \frac{-1}{T_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_4} \end{bmatrix} x + \begin{bmatrix} \frac{\gamma_1}{A_1} & 0 \\ 0 & \frac{\gamma_2}{A_2} \\ 0 & \frac{(1-\gamma_2)}{A_3} \\ \frac{(1-\gamma_1)}{A_4} & 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x$$

Dónde:

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}} \geq 0; i = 1,2,3,4\ son\ las\ constantes\ de\ tiempo\ de\ cada\ tanque.$$

El sistema es estable en lazo abierto con dos ceros multivariable. La naturaleza de los ceros es determinada por los parámetros $\gamma_1\ y\ \gamma_2$ de la siguiente manera (Alvarado et al. 2006):

Si $0 \leq \gamma_1 + \gamma_2 < 1$ tiene ceros en el semiplano derecho.

Si $1 \leq \gamma_1 + \gamma_2 < 2$ tiene ceros en el semiplano izquierdo

2.4.3 Modelo linealizado de la configuración modificada

Linealizamos el modelo obtenido en la sección anterior en un punto de operación dado por $(\bar{h}, \bar{u})$ donde u es el vector de voltajes en las válvulas, es decir $q_i = k_{pi}.u_i$ (k_{pi} : $ganancia\ de\ la\ bomba\ i\ en\ cm^3/Volt/s$). Definiendo las variables:

El modelo linealizado del sistema representado en espacio queda como (Numsomran et al. 2008):

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -\frac{1}{T_1} & -\frac{1}{T_x} & \frac{1}{T_x} & \frac{A_3}{A_1 T_3} & 0 \\ \frac{1}{T_x} & -\frac{1}{T_2} & -\frac{1}{T_x} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & & \frac{-1}{T_3} & 0 \\ 0 & 0 & & 0 & \frac{-1}{T_4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{\gamma_1 k_{p1}}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_{p2}}{A_2} \\ 0 & \frac{(1-\gamma_2) k_{p2}}{A_3} \\ \frac{(1-\gamma_1) k_{p1}}{A_4} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Dónde:

$T_i = \frac{A}{\beta_i a_i} \sqrt{\frac{2\overline{h_i}}{g}}$ y $T_x = \frac{A}{\beta_x a_x} \sqrt{\frac{2\left|\overline{h_1} - \overline{h_2}\right|}{g}} 0;$

$i = 1,2,3,4$ *son las constantes de tiempo de cada tanque.*

2.5 Conclusiones Parciales

Se realizó el estudio del modelado con éxito, lográndose entender los fundamentos físicos/matemáticos y el proceso de modelado para las dos principales configuraciones del proceso de 4 tanques acoplados a implementar.

CAPÍTULO 3. DESARROLLO DEL ENTORNO DE SIMULACIÓN

3.1 Introducción.

En el presente capítulo se realiza la implementación del entorno de simulación propuesto, se da inicialmente una visión general del funcionamiento del mismo, se abarcan los requerimientos de ejecución, las principales características y la arquitectura de diseño de software basada en MVC.

Posteriormente, se enfoca la implementación en cada elemento del modelo MVC (Modelo, Vista, Controlador). Asimismo, se explican los detalles utilizados para lograr la comunicación entre el entorno virtual de simulación y MATLAB/Simulink. Finalmente, detallamos la estrategia de control basada en PID desarrollada íntegramente en JAVA y la desarrollada en MATLAB/Simulink.

3.2 Generalidades.

El entorno de simulación se desarrolla íntegramente en lenguaje de programación Java con la ayuda del API EJS, que proporciona un conjunto de librerías e interfaces predefinidas para el trabajo de simulaciones de control como se explica en el capítulo anterior. Bajo estos criterios se implementa el comportamiento interactivo del sistema de 4 tanques bajo sus distintas configuraciones; así como una estrategia de control clásica basada en el PID, el mismo que se empaqueta bajo un archivo ".jar" ejecutable capaz de ser utilizado en cualquier equipo portátil o de escritorio con soporte de Java.

Adicionalmente con el objetivo de potenciar el entorno desarrollado, se provee al desarrollo de la capacidad de trabajo de manera integrada con Matlab/Simulink, haciendo uso del API SoftwareLinks, dándole la posibilidad de escribir o leer variables desde o hacia los mismos. Así como de ejecutar código o funciones de Matlab, como modelos en Simulink de manera local o remota. Logrando de este modo la opción de que a futuro puedan ser probadas estrategias de control más complejas con la planta desarrollada sin requerir de acceso físico a la planta real e incluso de manera remota al laboratorio. Para esto se requiere tener instalado adicionalmente Matlab/Simulink en una estación servidor que podría ser incluso en el mismo computador.

El objetivo del presente desarrollo es su funcionamiento en conjunto con la planta piloto de futura implementación en el laboratorio de la Sección de Electricidad y Electrónica de la PUCP, cuyo esquema propuesto se puede apreciar en la Figura 3.1. Es así que el desarrollo se basa en las características del diseño propuesto para la misma a la fecha. El mismo que presenta los siguientes datos de diseño principales:

- Configuraciones posibles: Tanques acoplados / Tanques en serie
- Caudal máximo entregado por las bombas: 16lt/min (266.7cm^3/s)
- Diámetro de la tubería: ½'' (1.27cm)
- Altura máxima en los tanques: 40cm.

Cabe mencionar que en caso de posibles cambios en este diseño, el entorno programado goza de la flexibilidad necesaria para variar todos y cada uno de los parámetros mencionados entre otros con la menor dificultad.

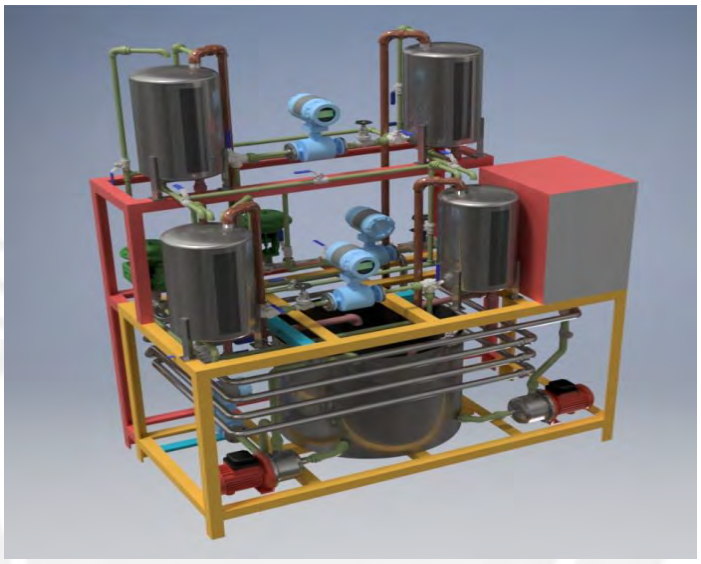


Figura 3.1: Esquema propuesto para la implementación de planta piloto en laboratorio de Control de la Sección de Electricidad y Electrónica de la PUCP.

3.3 Características y requerimientos básicos de trabajo para la ejecución del Entorno de Simulación.

Para la ejecución del entorno de Simulación desarrollado, si se desea trabajar con la versión que integra la funcionalidad de comunicación con Matlab físicamente requerimos de al menos dos computadoras de manera lógica. Una que funcione como equipo cliente y otra como servidor, las mismas que podrían estar instaladas en un mismo equipo físico. Si solo se trabaja con la versión íntegramente en Java solo es necesaria una computadora. A continuación se detallan los requerimientos específicos para cada computadora del primer caso. En el Anexo A (Manual de Usuario) se puede encontrar mayor detalle sobre la instalación de los elementos requeridos en cada una de las computadoras.

Por un lado el cliente requiere contar con (necesario):

- Plataforma Java: JDK O JRE 1.6 o superior.

- RPC Matlab Client (Cliente JIM, solo si se requiere funcionalidad adicional de ejecutar código en Matlab)
- EJS (solo para el desarrollo, mas no para la ejecución de la aplicación desarrollada)

Y en el lado del Servidor se requiere tener instalado (opcional):

- Plataforma Java: JDK O JRE 1.6 o superior
- Matlab / Simulink
- RPC Matlab Server (Servidor JIM)

3.4 Proceso de desarrollo.

El proceso de desarrollo se llevó a cabo de la siguiente manera:

a) Definición de Variables: parámetros del modelo, estados, variables de control, etc.

b) Inicialización de Variables: establecimiento de las condiciones iniciales.

c) Definición de la dinámica del proceso: funciones en ecuaciones diferenciales resueltas con el método de Runge-Kutta de orden 4, a ser resueltas de manera recurrente.

d) Definición de restricciones: tales como las alturas máximas a alcanzar en cada uno de los tanques.

e) Desarrollo de la interfaz gráfica: con ayuda de la interfaz proporcionada por EJS que facilita el mismo, al tener precargados los elementos como tuberías y tanques.

f) Asociación de elementos de la interfaz gráfica con variables o propiedades.

g) Relacionamiento de acciones con elementos de la interfaz gráfica.

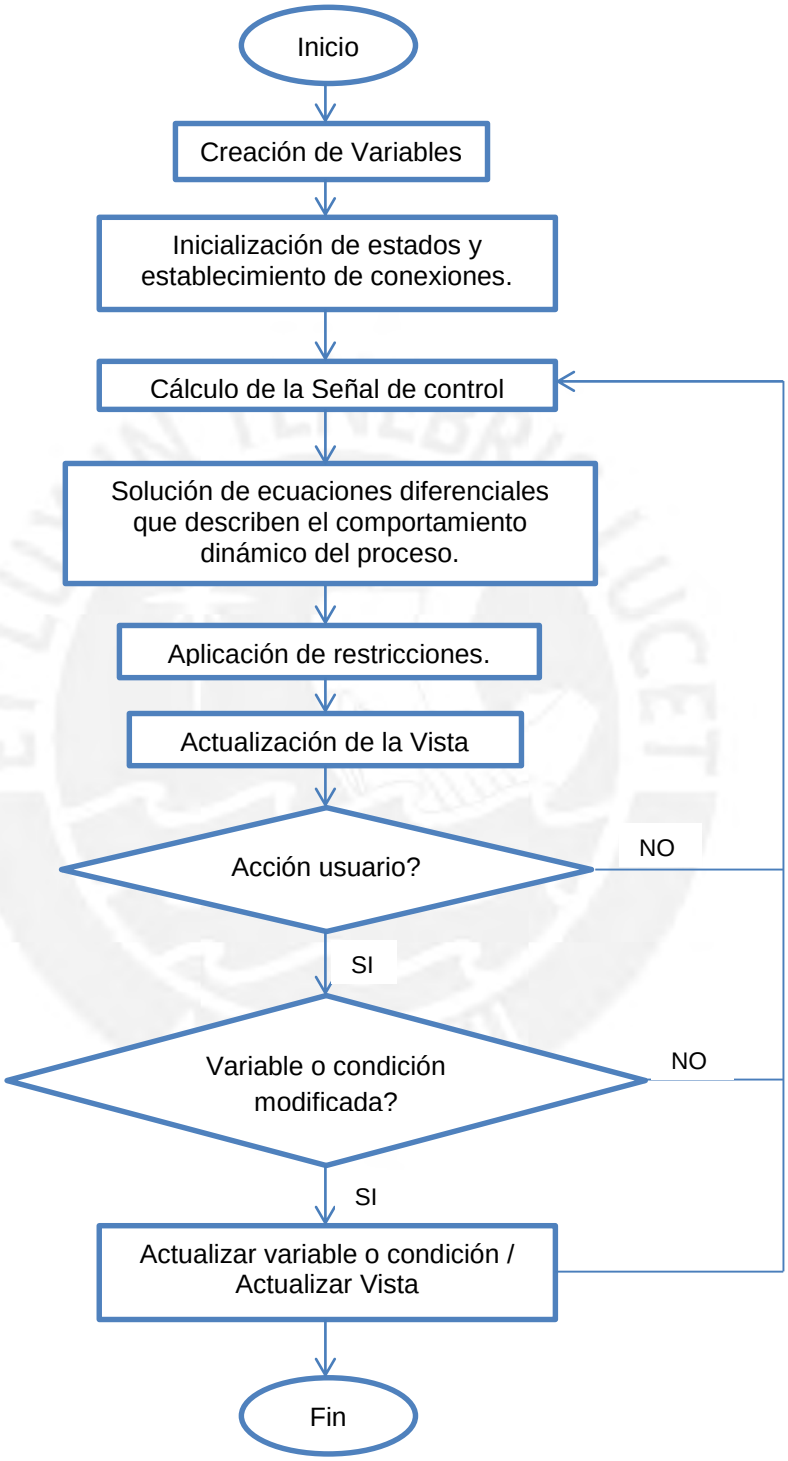
h) Definición de funciones personalizadas para la implementación del PID.

i) Definición de la lógica de control a aplicar de manera recurrente

j) Creación del elemento de conexión con Matlab: para la lectura escritura de variables entre plataformas y el envío remoto de sentencias a ejecutar en Matlab/Simulink.

3.5 Funcionamiento.

El entorno de simulación desarrollado funciona en resumen según el siguiente diagrama de flujo:



3.6 Características de la Implementación.

El entorno es íntegramente desarrollado bajo la arquitectura de EJS que se basa en el patrón MVC (Modelo - Vista - Controlador), de este modo la simulación queda estructurada en tres partes perfectamente diferenciables.

- **El modelo:** que describe el comportamiento del sistema de cuatro tanques implementado en función a variables utilizadas y relaciones de evolución de las mismas.
- **El controlador:** que define las acciones que se pueden ejecutar por parte del usuario como parte de la interactividad provista.
- **La vista:** que proporciona todos los elementos visuales que se presentan al usuario tanto como para su interacción como para representar visualmente el fenómeno estudiado.

3.6.1 Elemento Modelo de MVC

Como parte del modelo se han implementado en Java como elementos principales, las funciones que describen el comportamiento dinámico del Sistema de cuatro tanques.

Las variables utilizadas en la configuración básica (Johansson) son:

h_i: $nivel\ del\ agua\ en\ el\ tanque\ i, donde\ i$: 1,2,3,4 (cm)

$q_a\ y\ q_b$: $Flujo\ de\ salida\ de\ las\ bombas$ (cm^3/s)

γ_i: $parámetros\ de\ apertura\ de\ las\ válvulas\ de$ 3 $vías$. i: 1,2

a_i : $Area\ del\ orificio\ de\ fuga\ del\ tanque\ i, dondei$: 1,2,3,4 (cm^2)

A_i : $Area\ de\ la\ base\ del\ tanque\ i, dondei$: 1,2,3,4 (cm^2)

Logrando con ello representar el comportamiento deseado en cada uno de los tanques tal como se muestra a continuación:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1}{A_1}q_a$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2}{A_2}\,q_b$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)}{A_3}q_a$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)}{A_4}q_a$$

A continuación se observan en la Figura 3.2 a modo de funciones el comportamiento dinámico en ecuaciones diferenciales (las que se presentan previamente) para el cálculo del nivel de agua, se observan las funcione dh1, dh2, dh3, dh4 y dhsumi que describen el comportamiento de los tanques 1, 2, 3, 4 y reservorio respectivamente. Estas se resuelven de manera iterativa mediante el método de Runge-Kutta de orden 4.

```
841   // Evolution of the water level in tank 1  // > Propio.ODEs:1
842   public double dh1(double h1, double h3, double lambda1, double Aper1, double Aper3) {  // > Propio.ODEs:2
843       if (h1<0) h1=0;  // > Propio.ODEs:3
844       if (h1>60) h1=60;  // > Propio.ODEs:4
845       if (h3<0) h3=0;  // > Propio.ODEs:5
846       if (h3>60) h3=60;  // > Propio.ODEs:6
847       double Q1leak = (Aper1)*Sn*coef_leak*Math.sqrt(2*g*h1);  // > Propio.ODEs:7
848       double Q3leak = (Aper3)*Sn*coef_leak*Math.sqrt(2*g*h3);  // > Propio.ODEs:8
849       return (-Q1leak + Q3leak + lambda1*Q14)/A;  // > Propio.ODEs:9
850   }  // > Propio.ODEs:10
851   // Evolution of the water level in tank 2  // > Propio.ODEs:11
852   public double dh2(double h2, double h4, double lambda2, double Aper2, double Aper4){  // > Propio.ODEs:12
853       if (h2<0) h2=0;  // > Propio.ODEs:13
854       if (h2>60) h2=60;  // > Propio.ODEs:14
855       if (h4<0) h4=0;  // > Propio.ODEs:15
856       if (h4>60) h4=60;  // > Propio.ODEs:16
857       double Q2leak = (Aper2)*Sn*coef_leak*Math.sqrt(2*g*h2);  // > Propio.ODEs:19
858       double Q4leak = (Aper4)*Sn*coef_leak*Math.sqrt(2*g*h4);  // > Propio.ODEs:20
859       return (-Q2leak + Q4leak + lambda2*Q23)/A;  // > Propio.ODEs:22
860   }  // > Propio.ODEs:23
861   // Evolution of the water level in tank 3  // > Propio.ODEs:24
862   public double dh3(double h3, double lambda2, double Aper3){  // > Propio.ODEs:25
863       if (h3<0) h3=0;  // > Propio.ODEs:26
864       if (h3>60) h3=60;  // > Propio.ODEs:27
865       double Q3leak = (Aper3)*Sn*coef_leak*Math.sqrt(2*g*h3);  // > Propio.ODEs:30
866        return (-Q3leak + (1-lambda2)*Q23)/A;  // > Propio.ODEs:33
867   }  // > Propio.ODEs:34
868   // Evolution of the water level in tank 4  // > Propio.ODEs:35
869   public double dh4(double h4, double lambda1, double Aper4){  // > Propio.ODEs:36
870       if (h4<0) h4=0;  // > Propio.ODEs:37
871       if (h4>60) h4=60;  // > Propio.ODEs:38
872       double Q4leak = (Aper4)*Sn*coef_leak*Math.sqrt(2*g*h4);  // > Propio.ODEs:41
873        return (-Q4leak + (1-lambda1)*Q14)/A;  // > Propio.ODEs:44
874   }  // > Propio.ODEs:45
875   // Evolution of the water level in the lower deposit  // > Propio.ODEs:46
876   public double dhsumi(double h1, double h2, double h3, double h4, double Aper1, double Aper2, double Aper3, double Aper4){
877       if (h1<0) h1=0;  // > Propio.ODEs:48
878       if (h1>60) h1=60;  // > Propio.ODEs:49
879       if (h2<0) h2=0;  // > Propio.ODEs:50
880       if (h2>60) h2=60;  // > Propio.ODEs:51
881       if (h3<0) h3=0;  // > Propio.ODEs:52
882       if (h3>60) h3=60;  // > Propio.ODEs:53
883       if (h4<0) h4=0;  // > Propio.ODEs:54
884       if (h4>60) h4=60;  // > Propio.ODEs:55
885       double Q1leak = (Aper1)*Sn*coef_leak*Math.sqrt(2*g*h1);  // > Propio.ODEs:56
886       double Q2leak = (Aper2)*Sn*coef_leak*Math.sqrt(2*g*h2);  // > Propio.ODEs:57
887       return (Q1leak+Q2leak-Q23-Q14)/Asumi;  // > Propio.ODEs:61
888   }  // > Propio.ODEs:62
```

Figura 3.2: Código en Java de las funciones que describen el comportamiento del Sistema de cuatro tanques bajo el esquema de la configuración básica.

Adicionalmente, para la configuración modificada se agregan las siguientes variables que permiten modificar la configuración de tanques original:

β_i: *parámetros de apertura de las válvulas de salida del tanque* i.

β_x: *parámetros de apertura de las válvulas de conexión entre tanques* 1 *y* 2.

Pudiendo con ello representar el siguiente comportamiento para cada uno de los tanques:

$$\frac{dh_1}{dt} = -\frac{\beta_1 a_1}{A}\sqrt{2gh_1} + \frac{\beta_3 a_3}{A}\sqrt{2gh_3} - \frac{\beta_x a_x}{A} sgn(h_1 - h_2)\sqrt{2g|h_1 - h_2|} + \frac{\gamma_1}{A} q_a$$

$$\frac{dh_2}{dt} = -\frac{\beta_2 a_2}{A}\sqrt{2gh_2} + \frac{\beta_4 a_4}{A}\sqrt{2gh_4} + \frac{\beta_x a_x}{A} sgn(h_1 - h_2)\sqrt{2g|h_1 - h_2|} + \frac{\gamma_2}{A} q_b$$

$$\frac{dh_3}{dt} = -\frac{\beta_3 a_3}{A}\sqrt{2gh_3} + \frac{(1-\gamma_2)}{A} q_a$$

$$\frac{dh_4}{dt} = -\frac{\beta_4 a_4}{A}\sqrt{2gh_4} + \frac{(1-\gamma_1)}{A} q_a$$

En la Figura 3.3 se aprecian las funciones principales que describen las ecuaciones diferenciales, previamente mostradas, del comportamiento dinámico para esta configuración, de manera análoga al caso anterior para las nuevas ecuaciones.

```
932  public double dh1(double h1, double h2, double h3, double lambda1, double Aper1, double Aper3, double AperX) {
933      if (h1<0) h1=0;  // > Propio.ODEs:3
934      if (h1>hmax) h1=hmax;  // > Propio.ODEs:4
935      if (h2<0) h2=0;  // > Propio.ODEs:5
936      if (h2>hmax) h2=hmax;  // > Propio.ODEs:6
937      if (h3<0) h3=0;  // > Propio.ODEs:7
938      if (h3>hmax) h3=hmax;  // > Propio.ODEs:8
939      double Q1leak = (Aper1)*Sn*coef_leak*Math.sqrt(2*g*h1);  // > Propio.ODEs:9
940      double Q3leak = (Aper3)*Sn*coef_leak*Math.sqrt(2*g*h3);  // > Propio.ODEs:10
941      double Q12 = (AperX)*Sn*coef_12*signo(h1-h2)*Math.sqrt(2*g*Math.abs(h1-h2));      // > Propio.ODEs:11
942      return (-Q1leak + Q3leak - Q12 + lambda1*Q14)/A;  // > Propio.ODEs:12
943  }  // > Propio.ODEs:13
944  // Evolution of the water level in tank 2  // > Propio.ODEs:14
945  public double dh2(double h1, double h2, double h4, double lambda2, double Aper2, double Aper4, double AperX){
946      if (h1<0) h1=0;  // > Propio.ODEs:16
947      if (h1>hmax) h1=hmax;  // > Propio.ODEs:17
948      if (h2<0) h2=0;  // > Propio.ODEs:18
949      if (h2>hmax) h2=hmax;  // > Propio.ODEs:19
950      if (h4<0) h4=0;  // > Propio.ODEs:20
951      if (h4>hmax) h4=hmax;  // > Propio.ODEs:21
952        // > Propio.ODEs:22
953      double Q12 = (AperX)*Sn*coef_12*signo(h1-h2)*Math.sqrt(2*g*Math.abs(h1-h2));  // > Propio.ODEs:23
954      double Q2leak = (Aper2)*Sn*coef_leak*Math.sqrt(2*g*h2);  // > Propio.ODEs:26
955      double Q4leak = (Aper4)*Sn*coef_leak*Math.sqrt(2*g*h4);  // > Propio.ODEs:27
956     // return (Q2+Q32-Q20-Q2leak)/A;  // > Propio.ODEs:28
957      return (-Q2leak + Q4leak + Q12 + lambda2*Q23)/A;  // > Propio.ODEs:29
958  }  // > Propio.ODEs:30
959  // Evolution of the water level in tank 3  // > Propio.ODEs:31
960  public double dh3(double h3, double lambda2, double Aper3){  // > Propio.ODEs:32
961      if (h3<0) h3=0;  // > Propio.ODEs:33
962      if (h3>hmax) h3=hmax;  // > Propio.ODEs:34
963      double Q3leak = (Aper3)*Sn*coef_leak*Math.sqrt(2*g*h3);  // > Propio.ODEs:37
964        // > Propio.ODEs:39
965       return (-Q3leak + (1-lambda2)*Q23)/A;  // > Propio.ODEs:40
966  }  // > Propio.ODEs:41
967  // Evolution of the water level in tank 4  // > Propio.ODEs:42
968  public double dh4(double h4, double lambda1, double Aper4){  // > Propio.ODEs:43
969      if (h4<0) h4=0;  // > Propio.ODEs:44
970      if (h4>hmax) h4=hmax;  // > Propio.ODEs:45
971      double Q4leak = (Aper4)*Sn*coef_leak*Math.sqrt(2*g*h4);  // > Propio.ODEs:48
972       return (-Q4leak + (1-lambda1)*Q14)/A;  // > Propio.ODEs:51
973  }  // > Propio.ODEs:52
```

Figura 3.3: Código en Java de las funciones que describen el comportamiento del Sistema de cuatro tanques bajo el esquema de las configuraciones modificadas.

Además, se han implementado restricciones para simular sensores en los topes de los tanques, que al activarse cierren automáticamente la apertura de las bombas respectivas, así como abrir por completo los orificios de fuga que liberan el tanque que alcanzo la altura máxima, el código de implementación de las mismas se puede apreciar en la Figura 3.4 mediante la función en Java _constraints1 sin argumentos.

```
866   public void _constraints1 () { // > Relaciones fijas.Restricciones
867     if (h1>hmax) {  // > Relaciones fijas.Restricciones:1
868       manualQ23=0;  // > Relaciones fijas.Restricciones:2
869       manualQ23aux=0;  // > Relaciones fijas.Restricciones:3
870       manualQ23end=0;  // > Relaciones fijas.Restricciones:4
871       manualQ14=0;  // > Relaciones fijas.Restricciones:5
872       manualQ14aux=0;  // > Relaciones fijas.Restricciones:6
873       manualQ14end=0;  // > Relaciones fijas.Restricciones:7
874       Aper1=1;  // > Relaciones fijas.Restricciones:8
875       Aper3=1;  // > Relaciones fijas.Restricciones:9
876     }  // > Relaciones fijas.Restricciones:10
877     if (h2>hmax) {  // > Relaciones fijas.Restricciones:11
878       manualQ23=0;  // > Relaciones fijas.Restricciones:12
879       manualQ23aux=0;  // > Relaciones fijas.Restricciones:13
880       manualQ23end=0;  // > Relaciones fijas.Restricciones:14
881       manualQ14=0;  // > Relaciones fijas.Restricciones:15
882       manualQ14aux=0;  // > Relaciones fijas.Restricciones:16
883       manualQ14end=0;  // > Relaciones fijas.Restricciones:17
884       Aper2=1;  // > Relaciones fijas.Restricciones:18
885       Aper4=1;  // > Relaciones fijas.Restricciones:19
886     }  // > Relaciones fijas.Restricciones:20
887     if (h3>hmax) {  // > Relaciones fijas.Restricciones:21
888       manualQ23=0;  // > Relaciones fijas.Restricciones:22
889       manualQ23aux=0;  // > Relaciones fijas.Restricciones:23
890       manualQ23end=0;  // > Relaciones fijas.Restricciones:24
891       Aper3=1;  // > Relaciones fijas.Restricciones:25
892     }  // > Relaciones fijas.Restricciones:26
893     if (h4>hmax) {  // > Relaciones fijas.Restricciones:27
894       manualQ14=0;  // > Relaciones fijas.Restricciones:28
895       manualQ14aux=0;  // > Relaciones fijas.Restricciones:29
896       manualQ14end=0;  // > Relaciones fijas.Restricciones:30
897       Aper4=1;  // > Relaciones fijas.Restricciones:31
898     }  // > Relaciones fijas.Restricciones:32
```

Figura 3.4: Código en Java de las funciones que representar sensores en cada uno de los tanques que cierran la apertura de las bombas respectivas en caso de activarse.

3.6.2 Elemento Controlador de MVC

Respecto al controlador, que define las posibles interacciones por parte del usuario a través de la interfaz, en primer lugar está la posibilidad de pasar del modo de control automático (opción por defecto) basado en estrategias de control clásico PID al modo de control manual. Asimismo, la posibilidad de pausar, reanudar o detener la ejecución.

En segundo lugar se tiene la posibilidad de jugar con la apertura y cierre de cada una de las válvulas que modifican la configuración del Sistema.

En el modo de control automático únicamente está permitido poder variar los setpoint (referencia a alcanzar) de los tanques 1 y 2, así como los respectivos parámetros (Kp, Ti, Td) de sus PID correspondientes.

En el modo de control manual tenemos la posibilidad de controlar por completo la apertura de cada una de las válvulas del modelo, así como de las bombas a través de elementos deslizables o por ingreso directo de valores numéricos.

Generación de ventanas desplegables con las características de los principales componentes al hacer clic sobre ellos.

Como se explicó previamente, el proceso de ejecución inicia en primer lugar con la creación (y posible inicialización) de las variables definidas para implementar el modelo, en segundo lugar se ejecutan por primera y única vez los procesos definidos en el apartado inicialización, en tercer lugar se desarrolla, en un bucle continuo, la lógica definida en el apartado evolución junto con el cálculo del estado de las variables estudiadas y el de la señal de control a aplicar en cada instante.

3.6.3 Elemento Vista de MVC

El menú principal del entorno de simulación está formado por los elementos de menú Control y Configuración, el primero de ellos para la selección entre los modos de control manual y automático y el segundo para la selección del tipo de configuración para el sistema, dando la opción de elegir entre la configuración modificada de tanques acoplados por defecto o de tanques en serie. Cabe mencionar que en configuración manual es posible ajustar los parámetros de apertura y cierra de válvulas y bombas para formar mayor variedad de configuraciones personalizadas.

La vista está distribuida en 3 zonas, en la zona superior izquierda podemos apreciar el tipo de configuración actual seleccionada mediante el menú configuración. Adicionalmente, se puede observar una representación visual de los tanques y el nivel de líquido presente en cada uno de ellos para dicha configuración, así mismo se aprecia en verde las bombas cuando están en funcionamiento.

En la zona inferior izquierda podemos ver según el tipo de control, los parámetros modificables por el usuario, así como en otra pestaña los parámetros para el ajuste de los controladores PID en el caso de seleccionar control automático.

Por último en la zona derecha se muestra la gráfica del comportamiento de nivel de cada uno de los 4 tanques junto a los set points actuales. Así como del grado de apertura de las bombas 1 y 2 a través del tiempo. Además, se incluyen los valores actuales exactos en la parte inferior, así como el tiempo de simulación actual en segundos en la parte superior. Podemos observar lo descrito en las Figuras 3.5 y 3.6.

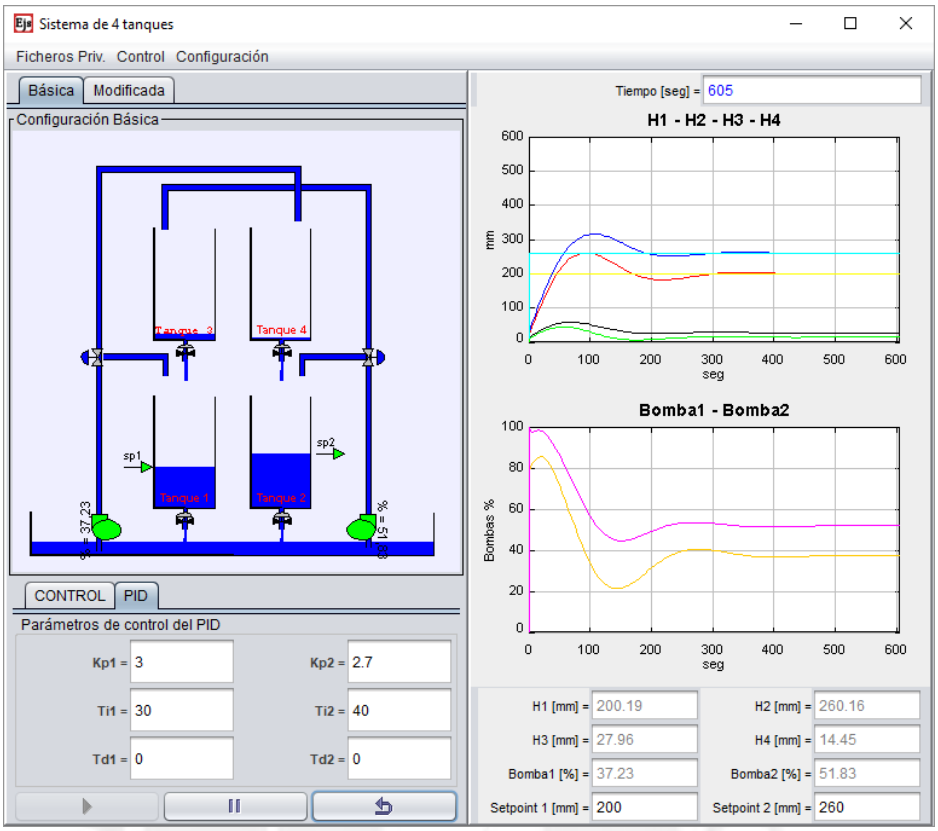


Figura 3.5: Vista actual del entorno de simulación funcionando bajo el modo de control automático.

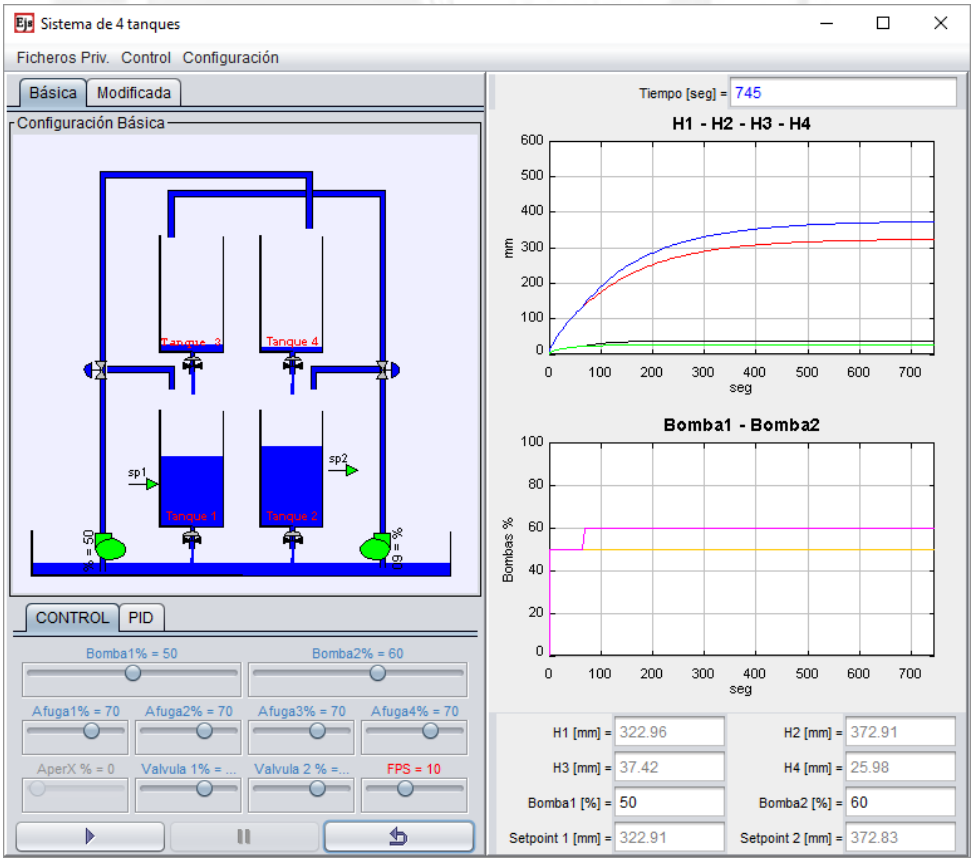


Figura 3.6: Vista actual del entorno de simulación funcionando bajo el modo de control manual.

3.7 Comunicación con MATLAB.

Para la comunicación con Matlab/Simulink se utiliza una arquitectura Cliente – Servidor, para lograr la misma se trabaja con un conjunto de librerías que añaden al entorno de simulación de comunicación la posibilidad de trabajar en conjunto con este software de ingeniería de manera externa e incluso de manera remota. Para esto implementamos se instala un servidor JIM para crear la conexión entre Matlab instalado en el mismo y Java instalado en el cliente JIM a través de RPC (remote procedure calls).

El servidor JIM puede ser instalado por medio de un archivo .jar (Java) ejecutable, que luego de iniciado debe mantenerse en ejecución en el servidor, el "RPC Matlab Server". Este se puede descargar desde la web en la siguiente dirección: https://github.com/UNEDLabs/ejs-element_jim.

El cliente JIM denominado EJS Matlab Connector Element es un plugin implementado que únicamente debe ser incluido en la carpeta de ejecución del entorno de simulación, para estar accesible por la aplicación cuando sea requerido. Este archivo se puede descargar desde la web en la siguiente dirección: https://github.com/UNEDLabs/ejs-element_jim.

A continuación en la Figura 3.7 podemos observar la arquitectura utilizada para la comunicación con Java.

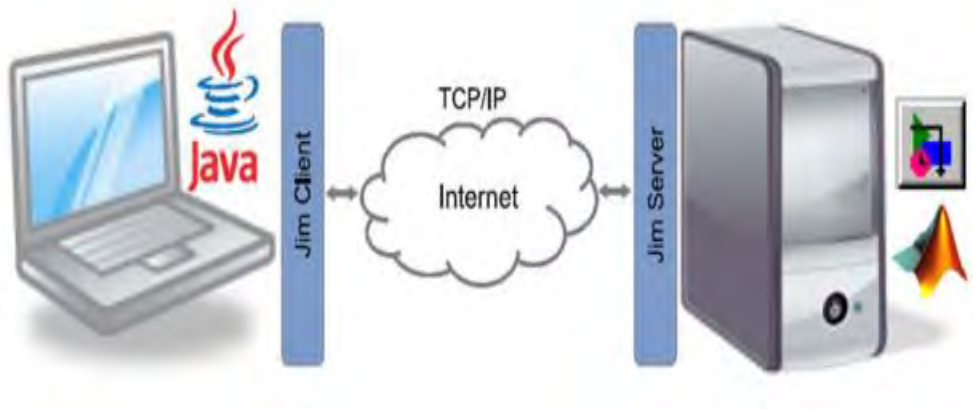


Figura 3.7: Esquema cliente-servidor utilizado para la comunicación entre Java – Matlab/Simulink

Adicionalmente se requiere incluir en el workpath de JAVA el conjunto de librerías matlabcontrol (https://code.google.com/p/matlabcontrol/), el cual proporciona una serie de métodos Java para la comunicación local o remota con MATLAB como son:

- connect(): Crea una nueva conexión con Matlab
- disconnect(): Cierra una conexión existente previamente abierta con el comando connect().
- eval(command): Evalua el código Matlab proporcionado.
- set(variable, value): Define el valor de una variable de Matlab con el valor indicado dentro del workspace de Matlab.
- get(variable): lee el valor de una variable definida en el workspace de Matlab.

En la Figura 3.8. se observa un ejemplo de conexión entre Java y Matlab donde se aprecian las funciones disponibles para utilizar, ver su uso a mayor detalle en el Anexo A.

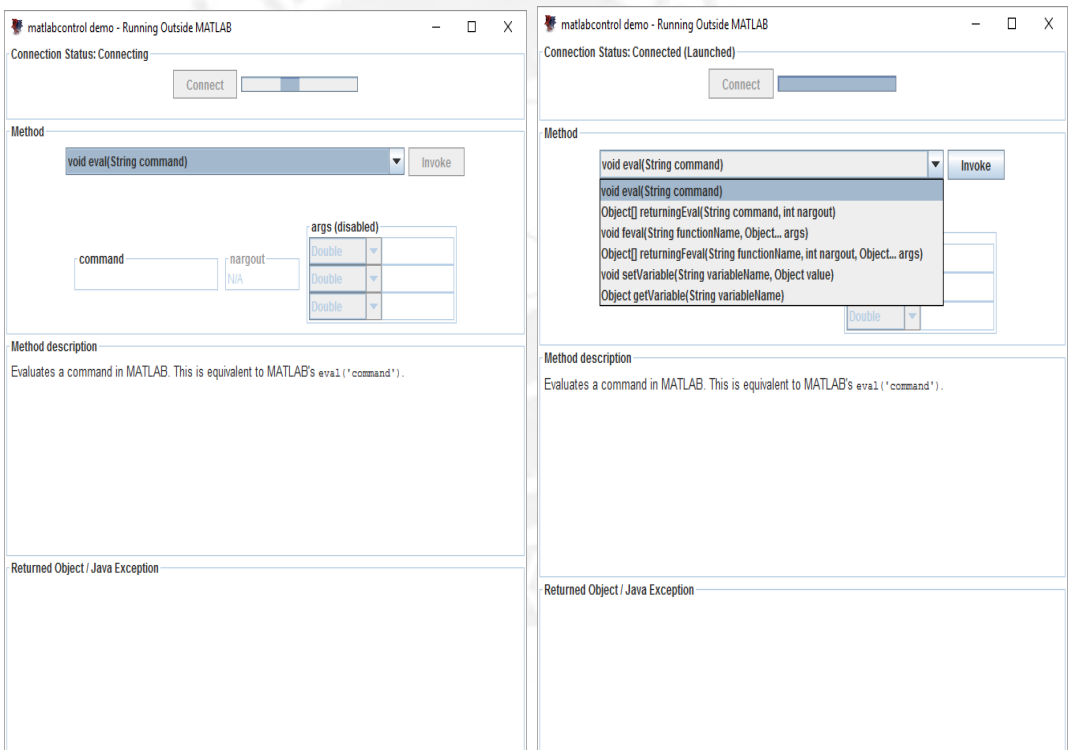


Figura 3.8: Ejemplo de conexión y ejecución remota de Matlab desde Java.

3.8 Implementación de una estrategia de control basada en el control clásico PID.

Para probar la aplicabilidad de estrategias de control en el lenguaje aplicado se diseña una estrategia de control basada en control clásico PID.

El objetivo de control es alcanzar y mantener la altura de los tanques inferiores (tanques 1 y 2) en los setpoint indicados, modificando para ello la apertura de las bombas.

Por tratarse de un sistema MIMO con fuerte acoplamiento son necesarias ciertas consideraciones para el funcionamiento adecuado de una estrategia de control basada en PID.

Para la configuración modificada de 4 tanques acoplados basado en el análisis de Matriz de ganancia relativa y de ubicación de ceros en el sistema (Numsomran, 2008) se concluye que el sistema es de fase mínima cuando $\gamma_1 + \gamma_2 > 1$; así como que la altura del tanque 1 depende principalmente del flujo Q_a y la del tanque 2 de Q_b. Es así que se diseña la estrategia de control como dos PID independientes lógicamente bajo la condición mencionada.

El controlador PID se implementó en base a la propuesta de implementación en "Feedback Systems: An introduction for Scientists and Engineers" (Astrom, 2008).

Se tiene:

$$v(t) = P(t) + I(t) + D(t)$$

Donde $u(t)$ es la saturación de $v(t)$ que modela al actuador.

Considera un peso β en la referencia (se considera $\beta = 1$ en la implementación) para el cálculo de la acción proporcional:

$$P(t_k) = k_p(\beta y_r(t_k) - y(t_k))$$

La acción integral se calcula como:

$$I(t_{k+1}) = I(t_k) + \frac{k_p}{t_i} t_s(r - y) + \frac{t_s}{T_t}(sat(v) - v)$$

Se considera un filtro para la acción derivativa:

$$D(t_k) = \frac{t_d}{t_d + Nt_s} D(t_{k-1}) + k_p N \frac{t_d}{t_d + Nt_s}(y - y\_old)$$

El algoritmo de control basado en una estrategia PID, implementado íntegramente en Java se aprecia en la Figura 3.9:

```
1030    private void initControl (double[] _control,   // > Propio.ControlSignal:1
1031                                                   double _init, double _kp, double _ts, double _ti, double _td,
1032                                                   double _tt, double _n, double _b, double _min, double _max) {
1033      // Input parameters  // > Propio.ControlSignal:4
1034      _control[0] = _init;  // > Propio.ControlSignal:5
1035      _control[1] = _kp;  // > Propio.ControlSignal:6
1036      _control[2] = _b;  // > Propio.ControlSignal:7
1037      _control[3] = _min;  // > Propio.ControlSignal:8
1038      _control[4] = _max;  // > Propio.ControlSignal:9
1039      // Computed variables  // > Propio.ControlSignal:10
1040      _control[5] = _kp * _ts / _ti;                  // Bi = Kp*Ts/Ti;  // > Propio.ControlSignal:11
1041      _control[6] = _ts / _tt;                          // Ar = Ts/Tt;  // > Propio.ControlSignal:12
1042      _control[7] = _td / (_td + _n * _ts);       // Ad = Td/(Td+N*Ts);  // > Propio.ControlSignal:13
1043      _control[8] = _kp * _n * _control[7];     // Bd = Kp*N*Ad;     // > Propio.ControlSignal:14
1044    }  // > Propio.ControlSignal:15
1045    private void init_IDaction (double[] _control, double _Iant, double _Dant) {  // > Propio.ControlSi
1046        _control[9] = _Iant;                            // Integral action previous  // > Propio.Contr
1047        _control[10] = _Dant;                         // Derivative action previous  // > Propio.Contro
1048    }  // > Propio.ControlSignal:19
1049    private double controlSignal (double[] _control, double _value, double _ref, double Iaction_ant) {
1050      // Proportional action : Kp*(b*ref - value)  // > Propio.ControlSignal:21
1051      double P = _control[1]*(_control[2]*_ref - _value);  // > Propio.ControlSignal:22
1052      // Derivative action : D = ad*D - Bd*(h-h_old);  // > Propio.ControlSignal:24
1053      _control[10] = _control[7]*_control[10] - _control[8]*(_value - _control[0]);   // > Propio.Contr
1054      double output;  // > Propio.ControlSignal:27
1055      output = P + _control[9] + _control[10]; // P + I + D  // > Propio.ControlSignal:31
1056      // Saturation of control signal  // > Propio.ControlSignal:32
1057      double outputSat;  // > Propio.ControlSignal:33
1058      if (output<_control[3])  outputSat = _control[3];  // > Propio.ControlSignal:34
1059      else if (output>_control[4])  outputSat = _control[4];  // > Propio.ControlSignal:35
1060      else outputSat = output;  // > Propio.ControlSignal:36
1061        // > Propio.ControlSignal:37
1062      if (manualControl) {  // > Propio.ControlSignal:38
1063      _control[9]=Iaction_ant;  // > Propio.ControlSignal:39
1064      } else {  // > Propio.ControlSignal:40
1065      // Update integral action  :  I = I + Bi*(Hr-h) + Ar*(v-u)  // > Propio.ControlSignal:41
1066      _control[9] = _control[9] + _control[5] * (_ref - _value) + _control[6]*(outputSat-output);  // >
1067      }
1068      // keep the value for the next time  // > Propio.ControlSignal:44
1069      _control[0] = _value;  // > Propio.ControlSignal:45
1070      return outputSat;  // > Propio.ControlSignal:46
1071    }  // > Propio.ControlSignal:47
```

Figura 3.9: Código en Java de las funciones que describen el comportamiento de los controladores PID a utilizar para controlar la apertura de cada bomba.

En la Figura 3.10 se muestra el controlador implementado en MATLAB/Simulink:

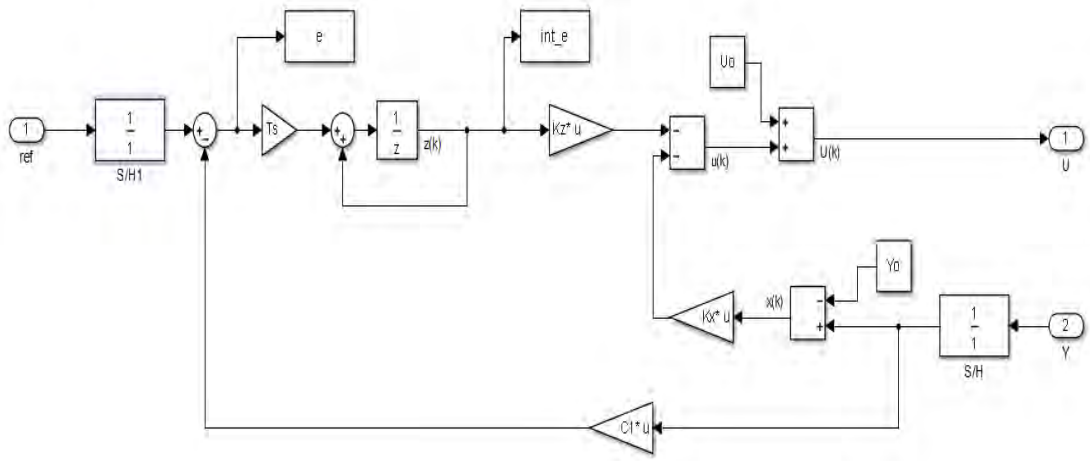


Figura 3.10: Controlador implementado en MATLAB/Simulink para ajustar la apertura de las bombas.

3.9 Flexibilidad de Configuraciones.

Con el desarrollo propuesto se obtiene una variedad de configuraciones, así como la posibilidad de variación de los parámetros de la planta en la simulación. Esto con la finalidad de que este Entorno de Simulación sea utilizado como benchmark de la planta física futura a implementarse en la universidad con otras implementaciones realizadas en distintas universidades. Con esta finalidad, se ha elegido realizar la propuesta de la Planta educativa basada en el Sistema de 4 tanques mencionado previamente en la sección 1.6 (Alvarado et al. 2006), y que puede apreciarse en la Figura 3.11.

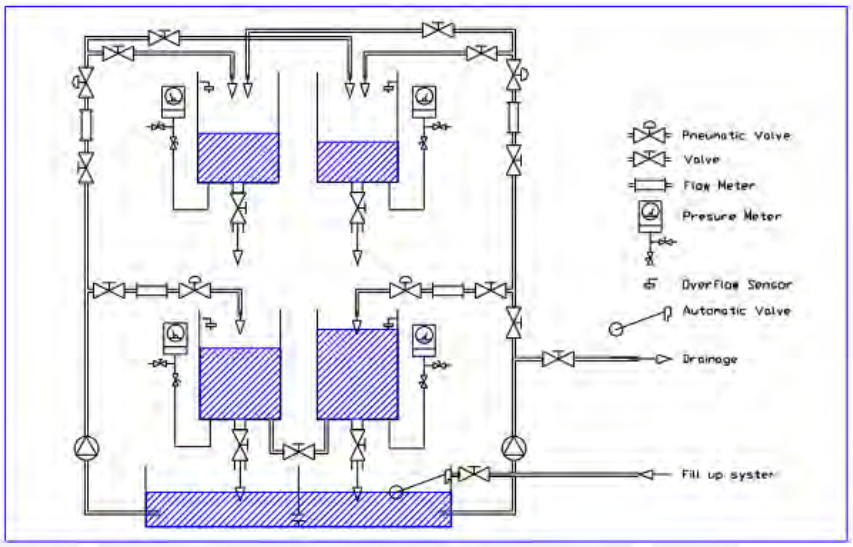


Figura 3.11: Plano de Configuración de Plantas (Alvarado et al. 2006).

Con esta implementación es posible obtener multitud de configuraciones mediante la apertura y cierre de alguna o algunas de las válvulas colocadas en el sistema.

Algunos ejemplos de configuraciones posibles se muestran a continuación en la Figura 3.12:

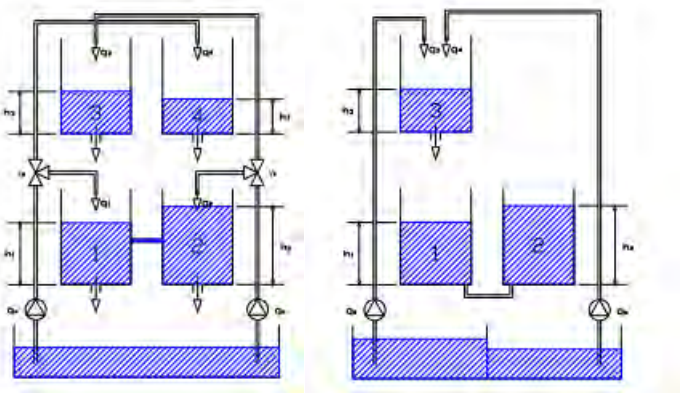


Figura 3.12: Ejemplo de configuraciones de Plantas obtenidas (Alvarado et al. 2006).

3.10 Conclusiones Parciales

Se consiguió una versión del Entorno virtual de simulación íntegramente en Java que adicionalmente tiene capacidad de integración con MATLAB/Simulink para potencializar sus capacidades de ser requerido, tal como el trabajo con controladores más avanzados.

Se obtuvo un entorno de simulación con una interfaz visualmente amigable para el proceso, que brinda la posibilidad de obtener diversas configuraciones mediante la simple modificación de ciertas variables tales como la apertura y cierre de ciertas válvulas y/o bombas a través de la interacción con la misma.

CAPÍTULO 4. PRUEBAS DE FUNCIONAMIENTO Y VALIDACION DEL DESARROLLO

4.1 Introducción

En este capítulo se valida el funcionamiento del desarrollo obtenido en el capítulo 3. Para lograr este objetivo se desarrolló otra simulación en MATLAB/Simulink con la finalidad de obtener datos confiables para la posterior validación. Posteriormente, se muestran pruebas de funcionamiento de algunas de las principales características del Entorno Virtual.

4.2 Simulación en MATLAB/Simulink para obtención de data de validación del Entorno desarrollado

Para la validación del funcionamiento del Entorno de Simulación desarrollado, se ha desarrollado adicionalmente una Simulación del sistema en MATLAB/Simulink para la generación de datos confiables, bajo las mismas condiciones de operación del Sistema.

Se definen el diagrama general en Simulink donde se aprecian subsistemas tanto para el controlador como para la planta

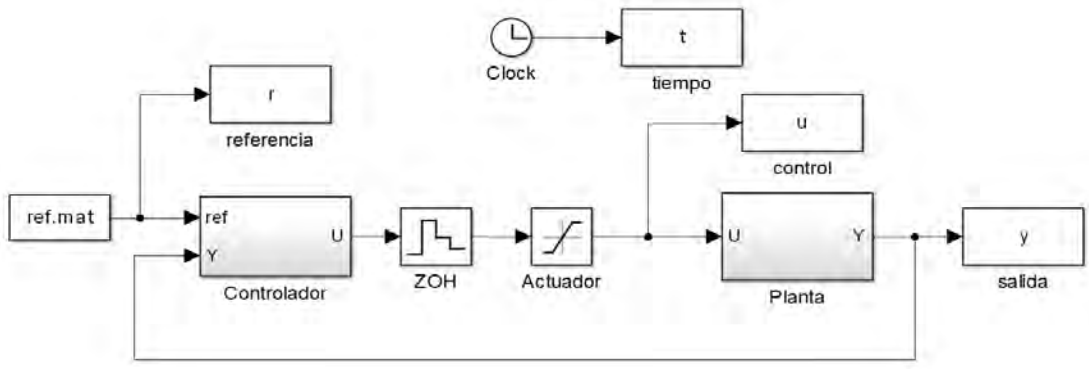


Figura 4.1: Modelo general en Simulink del Sistema de 4 tanques acoplados.

A continuación se detalla el subsistema Planta implementado: En la figura 4.2 podemos apreciar el detalle del subsistema, el mismo que incluye un bloque saturación con valor mínimo en altura 0 y máximo 40cm de altura de los tanques.

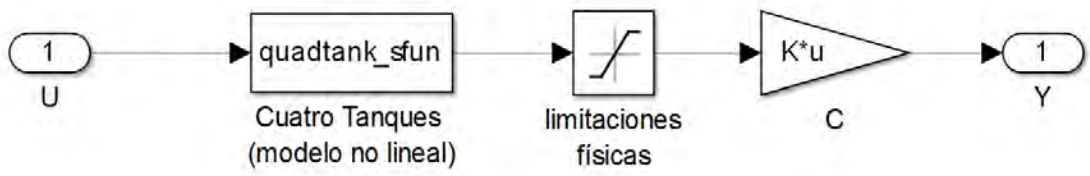


Figura 4.2: Subsistema Planta que representa el proceso de 4 tanques acoplados.

El modelo de la planta se ha implementado a través de un bloque S-function, el cual resuelve las ecuaciones diferenciales del comportamiento de cada tanque como se aprecia en la Figura 4.3:

Tanque 1:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1}{A_1}q_a$$

Tanque 2:

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2}{A_2}\,q_b$$

Tanque 3:

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)}{A_3}q_a$$

Tanque 4:

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)}{A_4}q_a$$

```
function [sys,x0,str,ts] = quadtank_sfun(t,x,u,flag,h0)
switch flag
    case 0 % inicializacion;
        sizes = simsizes;
        sizes.NumContStates = 4;
        sizes.NumDiscStates = 0;
        sizes.NumOutputs    = 4;
        sizes.NumInputs     = 2;
        sizes.DirFeedthrough = 0;
        sizes.NumSampleTimes = 1;
        sys = simsizes(sizes);
        str = [];
        ts = [0 0];
        x0 = h0 ;
```

Figura 4.3: Función en Matlab del proceso no lineal de 4 tanques acoplados (parte 1).

```
        case 1 % derivadas;
        h1 = x(1);
        h2 = x(2);
        h3 = x(3);
        h4 = x(4);
        % entradas
        v1 = u(1); % manipulada
        v2 = u(2); % manipulada

        % área de los tanques
        A1 = 28; % [cm2]
        A2 = 32; % [cm2]
        A3 = 28; % [cm2]
        A4 = 32; % [cm2]
        % áreas de las salidas de los tanques
        a1 = 0.071; % [cm2]
        a2 = 0.057; % [cm2]
        a3 = 0.071; % [cm2]
        a4 = 0.057; % [cm2]
        % ganancia de las bombas
        k1 = 3.33; % cm3/V/s
        k2 = 3.35; % cm3/V/s
        % apertura de las válvulas de 3 vías
        g1 = 0.70;
        g2 = 0.70;
        % gravedad
        g = 981; % cm2/s

        % derivadas de los estados
        if h1 < 0
            h1 = 0;
        end
        if h2 < 0
            h2 = 0;
        end
        if h3 < 0
            h3 = 0;
        end
        if h4 < 0
            h4 = 0;
        end
        dh1dt = -
a1/A1*sqrt(2*g*h1)+a3/A1*sqrt(2*g*h3)+g1*k1*v1/A1;
        dh2dt = -
a2/A2*sqrt(2*g*h2)+a4/A2*sqrt(2*g*h4)+g2*k2*v2/A2;
        dh3dt = -a3/A3*sqrt(2*g*h3)+(1-g2)*k2*v2/A3;
        dh4dt = -a4/A4*sqrt(2*g*h4)+(1-g1)*k1*v1/A4;
        sys = [dh1dt; dh2dt; dh3dt; dh4dt];

    case 3 % salidas;
        sys = [x(1); x(2); x(3); x(4)];

end
```

Figura 4.3: Función en Matlab del proceso no lineal de 4 tanques acoplados (parte 2).

4.3 Validación del Entorno de Simulación implementado en Java.

A continuación, en la Figura 4.4. se presenta la simulación bajo las mismas condiciones y entradas de apertura de las bombas para la simulación en MATLAB/Simulink y para el Entorno de simulación desarrollado en Java.

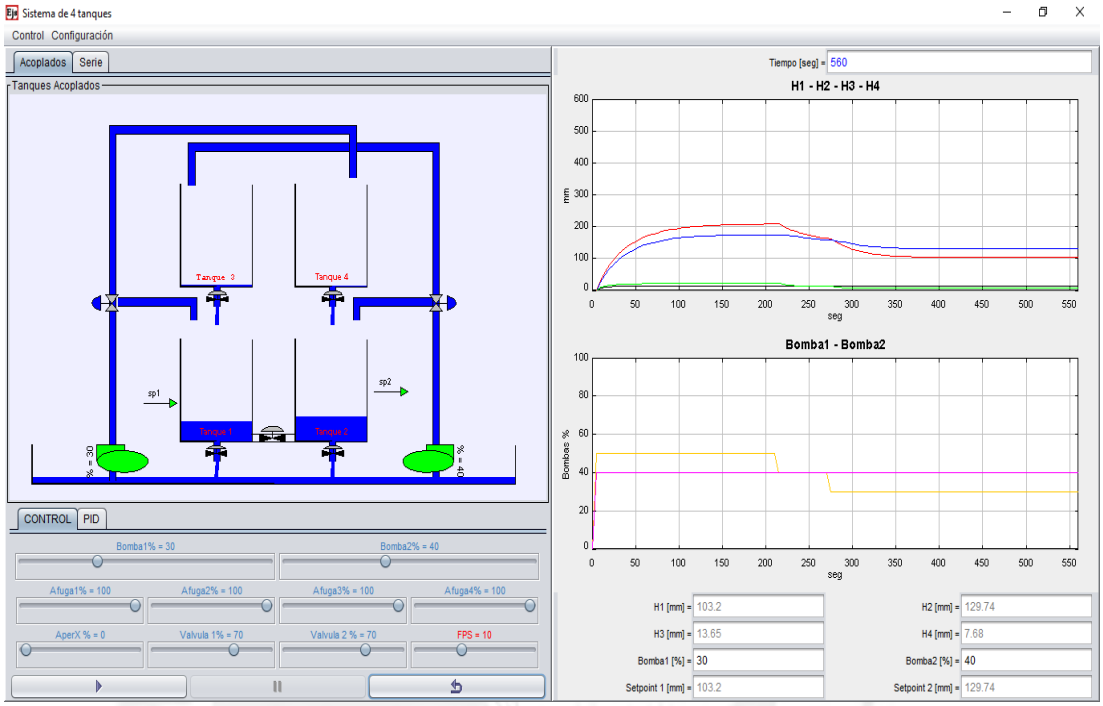


Figura 4.4: Simulación en modo control manual con el Entorno desarrollado.

En la simulación en Simulink se utilizó la misma señal de entrada para la apertura de ambas válvulas como se aprecia en la Figura 4.5.

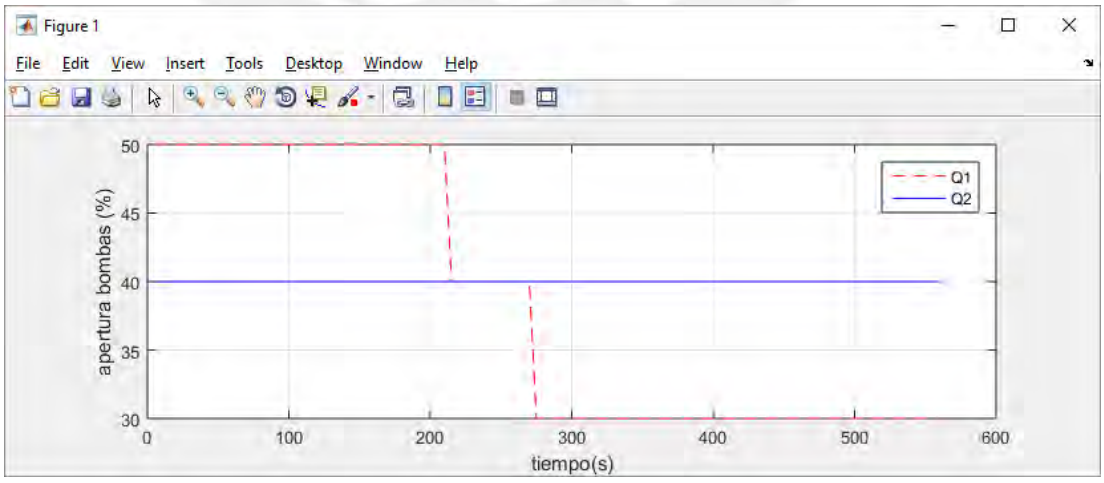


Figura 4.5: Señales de entrada para apertura de las válvulas para simulación en Simulink.

Finalmente, observamos en la Figura 4.6. que se obtiene prácticamente la misma salida para cada uno de los niveles de agua en los 4 tanques.

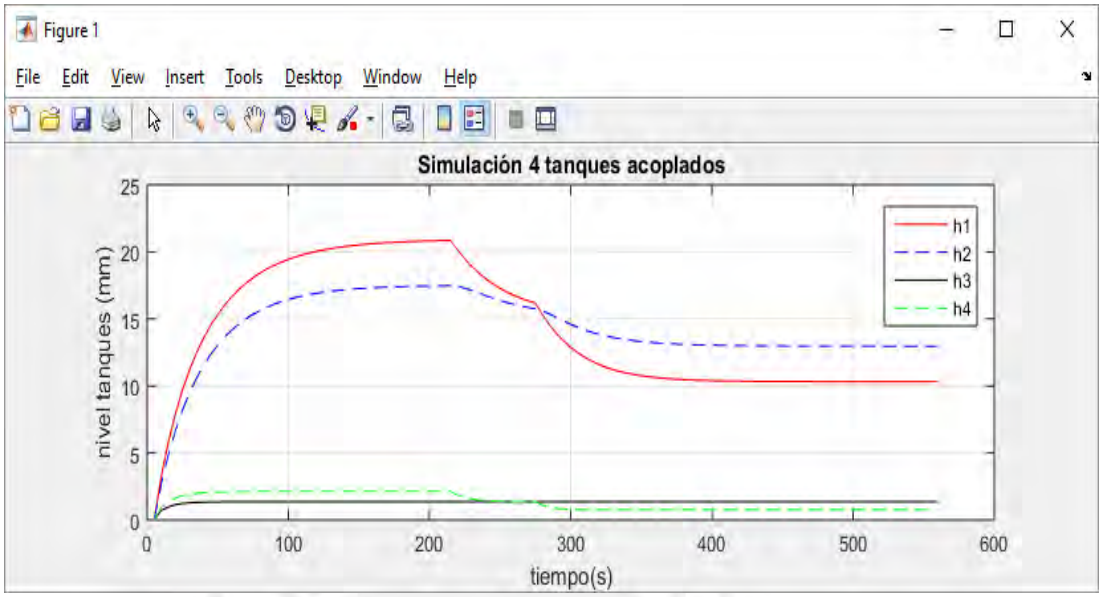


Figura 4.6: Nivel de líquido para los 4 tanques de la simulación para simulación en Simulink.

La experiencia se realizó para distintos valores de los parámetros, obteniéndose en todos los casos resultados prácticamente idénticos. Validando con ello que el desarrollo en Java tiene un funcionamiento correcto.

4.4 Pruebas de funcionamiento del sistema.

Finalmente, se realizaron pruebas bajo distintas variaciones en los parámetros para evaluar el comportamiento del sistema en diferentes aspectos tales como: el grado de eficacia del control PID bajo distintos parámetros de apertura de las válvulas de 3 vías, y la validez de los mecanismos de reacción ante la activación de los sensores de altura máxima en los tanques.

4.4.1 Pruebas de mecanismos de seguridad de cierre de bombas y apertura de orificios de fuga en caso de activación de algún sensor de nivel alto (40cm)

En la simulación se generó alcanzar la altura máxima primero en el tanque 1, disminuyendo el porcentaje de apertura de la válvula en el orificio de fuga del mismo, se observa como al alcanzar la altura máxima permitida y activarse el sensor ambas bombas se apagaron. A continuación se realizó una prueba similar esta vez haciendo que se active el sensor de nivel máximo del tanque 2, esta vez disminuyendo el porcentaje de apertura de la respectiva válvula en su orificio de fuga, produciéndose el mismo efecto que en el primer caso. Comprobándose así el correcto funcionamiento del mecanismo. En la figura 4.7 podemos apreciar la simulación descrita.

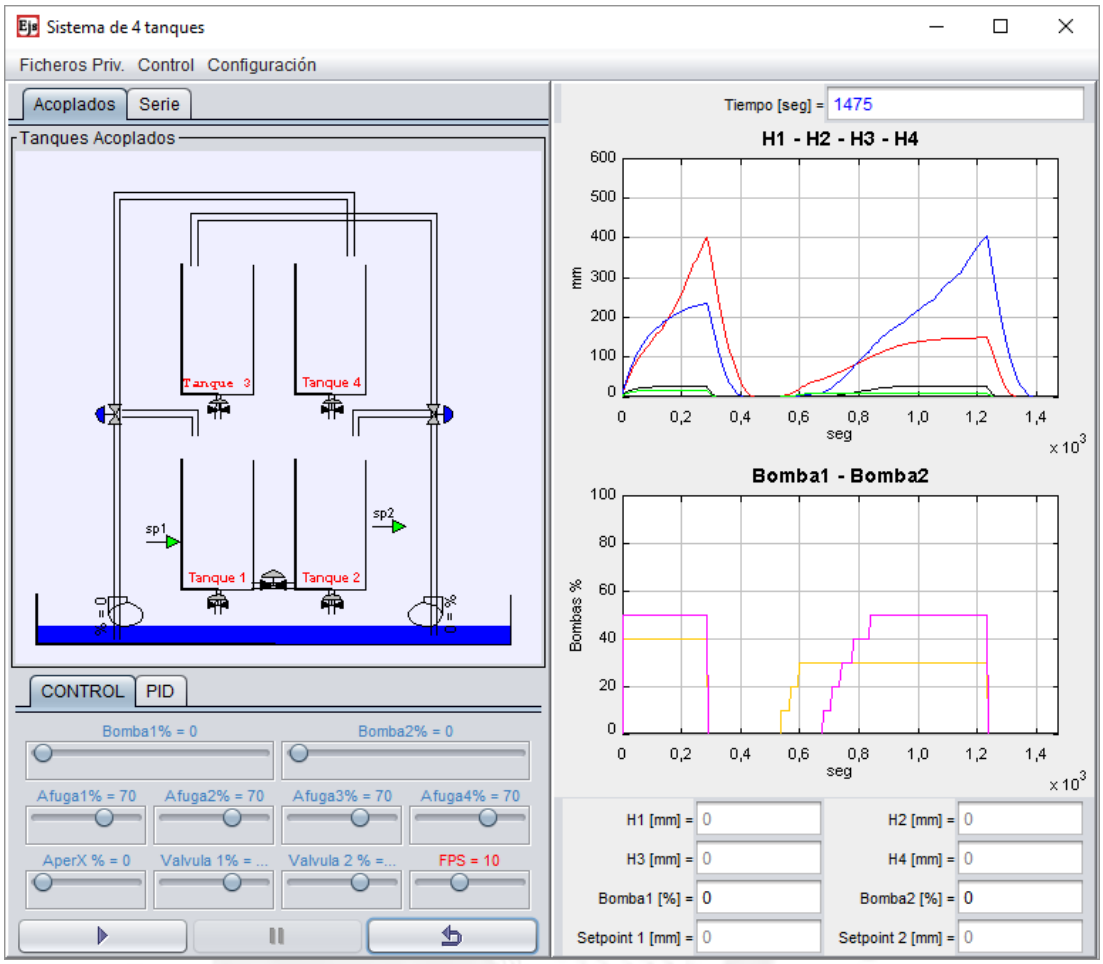


Figura 4.7: Simulación de prueba de los mecanismos de seguridad por activación de sensores de nivel máximo.

4.4.2 Pruebas de efectividad del controlador en función de la apertura de las válvulas de 3 vías.

Como se indicó en la sección 3.5 la estrategia de control se desarrolló a partir del supuesto de que la suma de las aperturas de las válvulas de 3 vías era mayor a 1 ($\gamma_1 + \gamma_2 > 1$). Es así que se evalúan 3 casos para comprobar la efectividad de la estrategia de control en base al comportamiento teórico del proceso estudiado.

Primero ejecutamos la simulación para $\gamma_1 = 0.6$ y $\gamma_2 = 0.7$, observando que efectivamente tal como indicaba el análisis teórico el sistema alcanza los valores deseados de nivel de líquido en ambos tanques. El resultado de esta simulación se aprecia en la Figura 4.8.

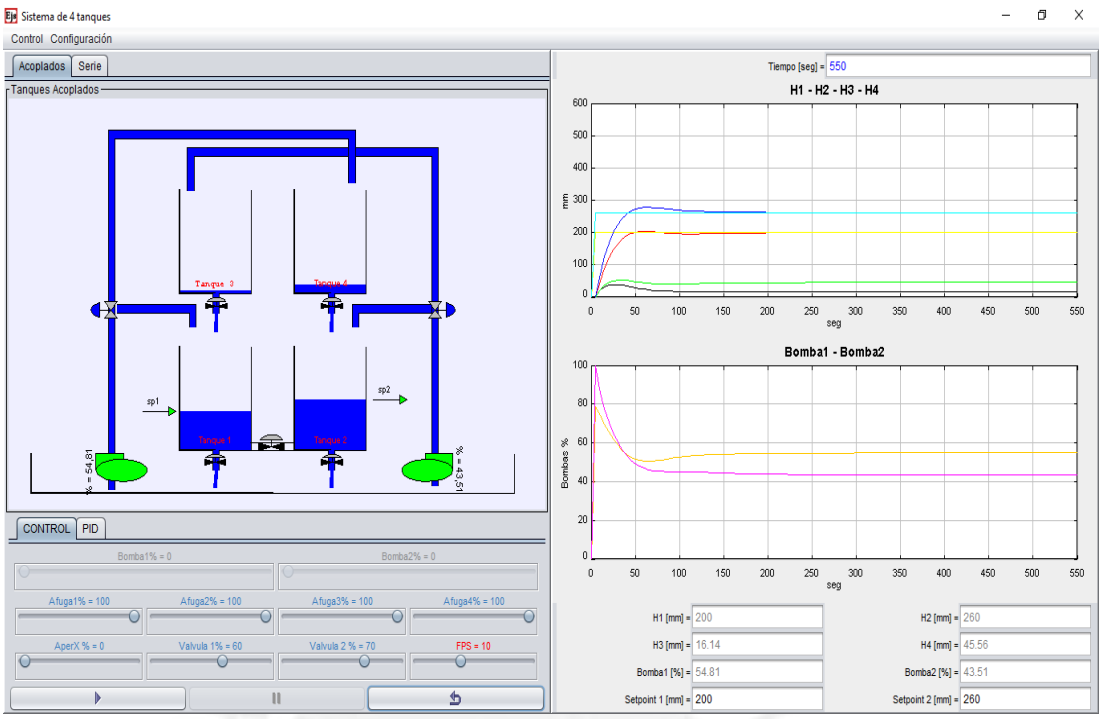


Figura 4.8: Simulación en control automático para $\gamma_1 = 0.6$ y $\gamma_2 = 0.7$.

Posteriormente se hace la simulación para $\gamma_1 = 0.5$ y $\gamma_2 = 0.5$. Caso límite teórico a partir del cual la estrategia de control deja de ser efectiva pues $\gamma_1 + \gamma_2 = 1$. Observándose también que se cumple lo analizado teóricamente, el controlador no es capaz de alcanzar los valores deseados de nivel de los tanques 1 y 2. No obstante, el sistema alcanza valores estacionarios como se puede apreciar en la Figura 4.9.

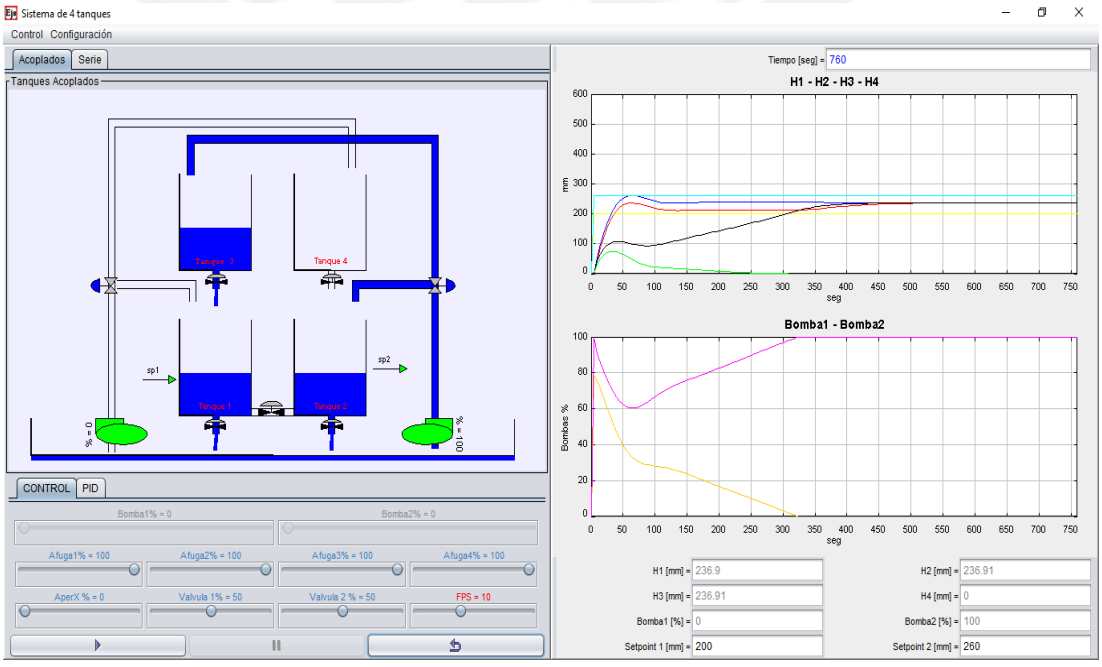


Figura 4.9: Simulación en control automático para $\gamma_1 = 0.5$ y $\gamma_2 = 0.5$.

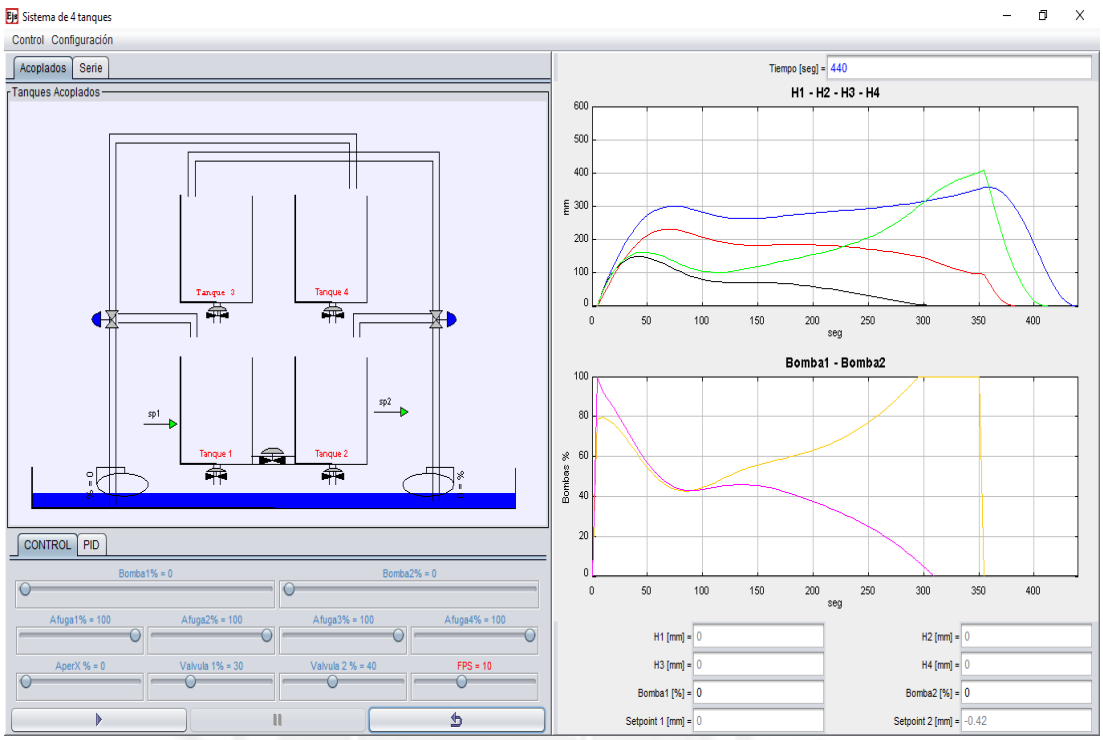


Figura 4.10: Simulación en control automático para $\gamma_1 = 0.3$ y $\gamma_2 = 0.4$.

Finalmente se hace la simulación para $\gamma_1 = 0.3$ y $\gamma_2 = 0.4$. Este caso es aún más extremo pues $\gamma_1 + \gamma_2 < 1$. Se observa también que se cumple lo planteado teóricamente, el supuesto analizado con la matriz de ganancias relativas deja de cumplirse totalmente de modo que el nivel de líquido en el tanque 1 no es principalmente afectada por la bomba1 y similar en caso del nivel en el tanque 2. La estrategia de control es totalmente inadecuada y rápidamente el tanque 4 se llena alcanzando a activarse el mecanismo de seguridad que apaga ambas bombas. Esta simulación se aprecia en la Figura 4.10.

4.4.3 Pruebas de funcionamiento del sistema para la configuración modificada de 4 tanques acoplados

Como se observa en el detalle de los capítulos anteriores; la configuración modificada, a diferencia de la de Johansson, incluye una tubería entre los tanques 1 y 2, la misma que posee una válvula de dos vías para regular el porcentaje de apertura de la misma.

A continuación, se evalúa la efectividad del controlador para distintos valores de apertura de la válvula indicada (AperX).

Para valores bajos de porcentaje de apertura, como es el caso de una apertura del 20%, el sistema aún no presenta problemas para alcanzar los valores de nivel deseados como se observa en la Figura 4.11.

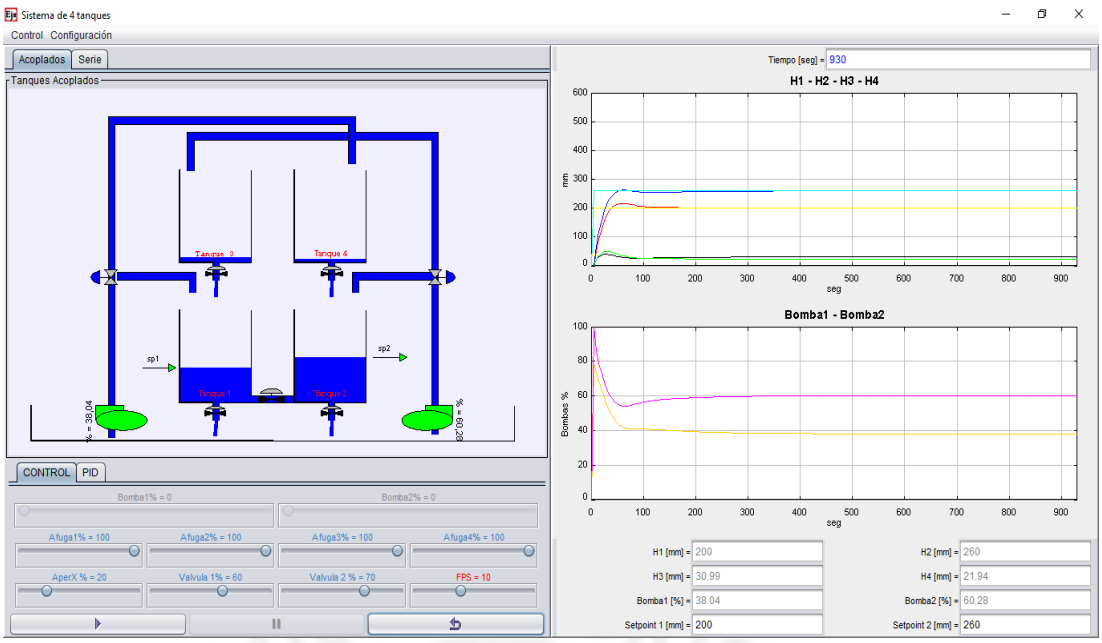


Figura 4.11: Simulación en control automático para configuración modificada con $\gamma_1 = 0.6$ y $\gamma_2 = 0.7$ y AperX =0.2 (bajo).

Conforme los valores de porcentaje de apertura aumentan el sistema presenta cada vez mayores dificultades como una mayor demora para alcanzar los valores de niveles deseados, pero a valores próximos o mayores a 80% por ejemplo, el sistema ya no es capaz para alcanzar los valores de nivel deseados como se observa en la Figura 4.12.

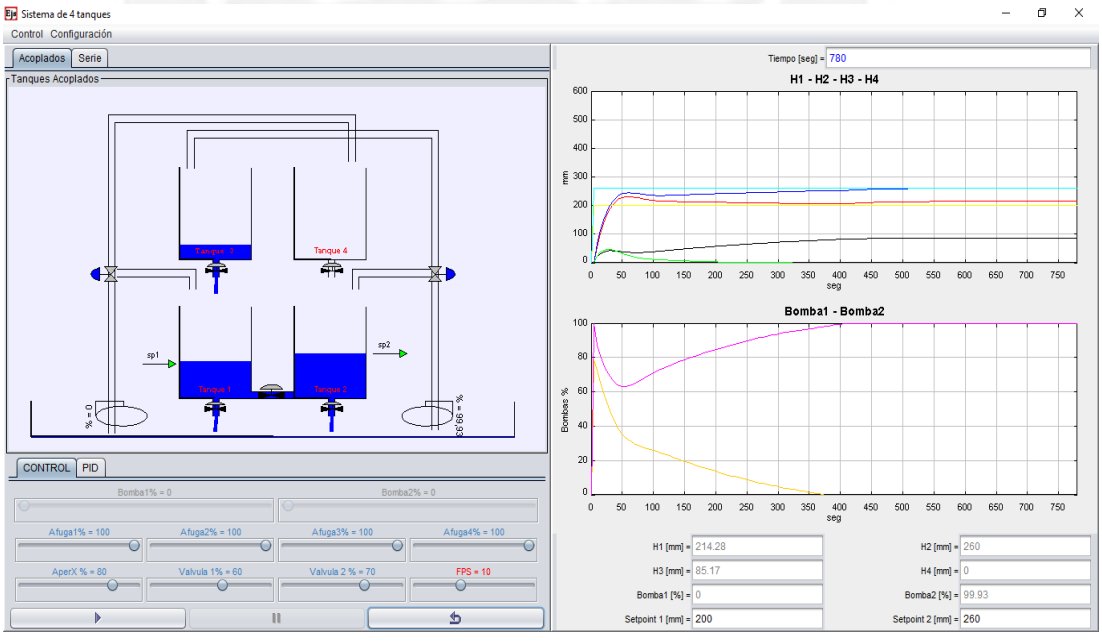


Figura 4.12: Simulación en control automático para configuración modificada con $\gamma_1 = 0.6$ y $\gamma_2 = 0.7$ y AperX =0.8 (alto).

4.5 Análisis de resultados

Se corroboró el correcto funcionamiento del modelo implementado para el proceso de 4 tanques acoplados, en contraste con data de validación de simulaciones desarrolladas en MATLAB/Simulink. La interfaz desarrollada resulto visualmente amigable e intuitiva, lo que proporciona un fácil manejo del Entorno. Asimismo, el Entorno resulta de fácil personalización aún en modo desarrollador (con acceso al código fuente). Para un mejor manejo del Entorno se desarrolló como anexo y aporte adicional el Manual de Usuario e Instalación (Ver Anexo A)

El Entorno Virtual de Simulación se desempeñó correctamente según las consideraciones esperadas para los distintos escenarios de prueba. Se logró la integración con Matlab para ampliar las potencialidades del Entorno Virtual. Se validaron los mecanismos de seguridad de cierre de bombas y apertura de válvulas de fuga en casos de alcanzarse la altura máxima en alguno de los tanques. Se verificó el adecuado desempeño del controlador y el sistema para ambas configuraciones implementadas.

CONCLUSIONES.

Se desarrolló el Entorno Virtual de Simulación para el Sistema de 4 tanques acoplados íntegramente en Java, alcanzando todos los requerimientos planteados como objetivo, y se validó su funcionamiento con una simulación del sistema en MATLAB/Simulink.

Se estudió el modelado teórico del sistema de 4 tanques acoplados utilizado en diferentes trabajos y se logró implementar una versión general del mismo en el Entorno Virtual de Simulación con éxito. Se eligió JAVA/EJS como herramienta de software libre para la implementación obteniéndose resultados satisfactorios que permitieron una correcta implementación

Se logró obtener una versión funcional del Entorno virtual de simulación íntegramente en Java y adicionalmente una versión con capacidad de integración con MATLAB/Simulink para potencializar sus capacidades de ser requerido, tal como el trabajo con controladores más avanzados.

Se corroboró el correcto funcionamiento del modelo implementado para el proceso de 4 tanques acoplados, en contraste con data de validación de simulaciones desarrolladas en MATLAB/Simulink. La interfaz desarrollada resulto visualmente amigable e intuitiva, lo que proporciona un fácil manejo del Entorno. Asimismo, el Entorno resulta de fácil personalización aún en modo desarrollador (con acceso al código fuente). Para un mejor manejo del Entorno se desarrolló como anexo y aporte adicional el Manual de Usuario e Instalación.

RECOMENDACIONES

Se recomienda la mejora del sistema para la implementación de nuevas estrategias de control sean implementadas en JAVA o en MATLAB/Simulink para realizar comparación de las mismas y determinar las ventajas y desventajas de las mismas en este proceso de 4 tanques acoplados.

Asimismo se sugiere posterior a la implementación de la Planta física modelo en laboratorio, la integración con el Entorno de simulación desarrollado para que no solo funcione como herramienta de simulación sino que sea capaz de intercambiar información con la planta y funcione como sistema de supervisión de la misma.

BIBLIOGRAFÍA.

- Nazrah S.; Bagterp J. (2015). **Modeling and Simulation of a Modified Quadruple Tank System,** IEEE International Conference on Control System, Computing and Engineering, Malaysia.
- Dominguez M.; Fuertes J.; Prada M.; Morán A. (2014) **Remote Laboratory of a Quadruple Tank Process for Learning in Control Engineering Using Different Industrial Controllers**. Dpto. de Ingeniería Eléctrica y de Sistemas y Automática, University of León. León, Spain.
- Arias L. (2013) **Modelado y simulación de un sistema interconectado de cuatro tanques** (LabView), Universidad "Rodrigo Franco", Costa Rica.
- Fabregas E (2013). **Plataformas interactivas de experimentación virtual y remota,** Tesis doctoral UNED, Madrid, España.
- Hermosell A.; Fernandez J. (2012). **Modelado y Simulación de Brazos Robóticos con Blender y Python**. XXXIII Jornada de Automática. Vigo, España.
- Numsomran A.; Tipsuwanporn V.; Tirasesth K. (2008). **Modeling of the Modified Quadruple-Tank Process**, SICE Annual Conference, Japón.
- Roinila T., Vilkko M.; Jaatinen A. (2008). **Corrected Mathematical Model of Quadruple Tank Process**. Institute of Automation and Control, Tampere University of Technology. Tampere, Finland.
- Astrom K.; Murray R. (2008). **Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press**, Estados Unidos.
- Mott R. (2006). **Mecánica de Fluídos.** Sexta Edición. Pearson Education, México.
- Alvarado I.; Limon D.; García W.; Alamo T.; Camacho E. (2006) **An Educational Plant Based on the Quadruple-Tank Process**.
- Torres L. (2006) **Simulación de Sistemas de Control en JAVA: Bases Conceptuales y Metodológicas para un Laboratorio Virtual.** Universidad de los Andes, Venezuela
- Duro N.; Vargas H; Dormido S.; Dormido R.; Sanchez J. (2005); **El Sistema de tres tanques: Un laboratorio virtual y remoto usando Easy Java Simulations**. Dpto. de Informática y Automática. UNED. Madrid, España.

- Jimenez E.; Pérez M.; Martinez E.; Sanz F.; Santamaría J.; Blanco J. (2005) **Escenarios virtuales WEB3D: Simulación con VRML, JAVA3D y X3D**. Universidad de la Rioja. España.
- Sánchez. J.; Esquembre, F.; Martin C.; Dormido S.; Pastor R.; Urquía A. (2005) **Easy Java Simulations: an Open-Source Tool to Develop Interactive Virtual Laboratories Using MATLAB/Simulink.** Department of Computer Sciences and Automatic Control, UNED, Madrid, España.
- Dormido, S.; Esquembre, F. (2003) **The Quadruple-Tank Process: An Interactive Tool for Control Education.** European Control Conference (ECC). Cambridge, UK.
- Johansson K.; (2000) **The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero**. IEEE