I

UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO

FACULTAD DE INGENIERÍA ELECTRICA

MODELADO Y SIMULACION DE SISTEMAS DISCRETOS EN 20-SIM

Tesis para obtener el título profesional de

Ingeniero en computación

Presenta

MARTIN CORONA QUIÑONES

Asesor

Dr. Gilberto González Avalos

Noviembre 2020

Todo lo que un día comiences, debes de terminar, no importa si te demoras poco o mucho tiempo, realízalo sin importar los obstáculos de la vida.

-Martin Corona Quiñones

Agradecimientos

Agradezco a mi mama y hermanos (Joan Robles Corona y Ezequiel Araujo Corona) por la ayuda con sus consejos a que siguiera adelante con mis estudios acompañarme en los buenos y malos momentos vividos durante mi carrera universitaria, quiero agradecer a mi hermano Nazario por apoyarme con mi laptop. También agradezco a todos esos amigos que hicieron que mi estadía en la facultad fuera más divertida y sencilla. Quiero agradecer a mi asesor de tesis Dr. Gilberto Gonzales Avalos por el tiempo, dedicación y paciencia para la elaboración de este documento, dándome las herramientas necesarias para culminar con mi investigación y plasmar mis ideas. Le doy gracias a mi tío Juan Arzate por su ayuda.

Y más que nada le doy las gracias a dios y a la virgen por culminar mi carrera profesional.

Dedicatoria

Le dedico este trabajo con todo mi amor y esfuerzo a mi mama, Audi Corona Quiñones. Ella es la razón por la que concluyo mis estudios universitarios, desde el inicio me a apoyado. Gracias mama por esforzarte para que no me hiciera falta nada en mis estudios, tu esfuerzo y sacrificio no fue en vano. Gracias por educarme y enseñarme a trabajar honradamente. Gracias por todo el cariño que me has dado incondicional y consejos, más en los momentos que los necesitaba. Y sobre todo muchas gracias por creer en mí para lograr este sueño que tanto anhelabas.

Tú eres mi más grande tesoro, te amo mama.

Por ultimo le dedico este trabajo a mi esposa Herlinda Rojas Reyes y mi hijo, por el apoyo que me ha dado, así como a darme ánimos día a día para terminar mi carrera. Le doy gracias a dios por haberte puesto en mi camino te amo.

Resumen

Este proyecto tiene como objetivo la importancia del modelado y simulación de sistemas discretos en 20-sim, ya que es un software con el cual es fácil familiarizarse y cumple este requisito que es para modelar y simular sistemas. En la actualidad existen muchos proyectos los cuales conllevan grandes inversiones de dinero así como de personal, ya que dichos proyecto pueden tener resultados negativos al realizarlos o llevarlos a cabo (ejecución), ocasionado la perdida de dinero y lo más importante vidas humanas al realizar la ejecución, por eso es muy importante el modelado y la simulación de dicho proyecto, para así darnos cuenta como se comportara la ejecución del proyecto, y así mismo saber que valores introducir. Esto es una gran ventaja para el desarrollo del proyecto, con la simulación nos damos cuenta que debemos realizar que no. Para si terminar mucho más pronto, y ahórranos dinero y vidas humanas.

A manera de conclusión sobre el presente trabajo y evaluando las ventajas y desventajas del tema, es muy importante el modelado y simulado antes de realizar dicha tarea porque te estarás ahorrando mucho dinero y vidas humanas, porque en el software se pueden cometer errores pero ya de manera física sería una catástrofe cometerlo.

Palabras clave

- 20 SIM
- Sistema
- Modelado
- Simulación

Abstract

Objective the importance of modeling and simulation of discrete systems in 20-sim, since it is a software with which it is easy to become familiar with and meets this requirement that is to model and simulate systems. Currently there are many projects which involve large investments of money as well as personnel, since these projects can have negative results when carrying them out or carrying them out (execution), causing the loss of money and most importantly human lives when carrying out the execution, that is why the modeling and simulation of said project is very important, in order to realize how the execution of the project will behave, and also to know what values to enter. This is a great advantage for the development of the project, with the simulation we realize that we must do not. In case it ends much sooner, and save us money and human lives.

By way of conclusion about the present work and evaluating the advantages and disadvantages of the topic, modeling and simulation is very important before carrying out this task because you will be saving a lot of money and human lives, because errors can be made in the software but already physically it would be a catastrophe to commit it.

Keywords

- 20 SIM
- Sistema
- Modelado
- Simulación

Índice

Lista de figuras

Lista de tablas

Capítulo 1 Introducción

Desde hace épocas atrás se han ido creando sistemas que han ayudado con el avance de la tecnología ya que han facilitado el crecimiento de los mismos, el modelado de sistemas es muy importante debido a que ayuda a visualizar de cómo será el sistema ya funcionando en tiempo real, se aprecian aún más sus características cualitativas y cuantitativas, ya que nos ayudan a prevenir la perdida de recurso económico al simularlo mediante un software, existen varios tipos de software que nos facilita el modelado y simulación de sistemas, ya que es una herramienta muy útil en la actualidad y es muy recomendable, por varias razones de seguridad.

Los avances en la capacidad de computación permiten hoy en día tratar un gran número de sistemas de simulación con muchas variables lo que lleva a la posibilidad de simular sistemas más complejos.

Para que un software sea del agrado de los usuarios así como de la empresa debe de cumplir ciertas expectativas o necesidades de quien o quienes lo estén utilizando, funciones especiales que sean atractivas para los usuarios. Existen distintos software de modelado y simulación que se ejecutan en distintos sistemas operativos ya que para cada usuario así como para cada empresa utiliza distinto sistema operativo como la versión, en este caso algunas versiones de este tipo de software pude tener conflicto con el sistema operativo en el cual se esta ejecutando o corriendo el software, y por lo tanto el software no se estaría aprovechando al 100% de su rendimiento, lo recomendable es revisar la compatibilidad de dicho sistema operativo con la versión del software de simulación para no cometer ese error y aprovechar al máximo el rendimiento del software.

1.1 La importancia en el modelado

El moldeado es una técnica que trata de enseñar a través de sucesiones o aproximaciones, el modelado está basado en el aprendizaje por observación y no se divide la actividad de forma gradual.

Un sistema se puede definir como un conjunto de elementos unidos por relaciones de interacción o interdependencia. En el ámbito de los sistemas productivos estos elementos normalmente tienen un objetivo común. Los elementos que forman parte del sistema vienen condicionados por el objetivo del estudio que se pretende realizar, ya que un sistema definido para un estudio determinado puede ser una parte de un sistema más amplio definido para otro estudio particular. Por ejemplo, si se quiere determinar cuál es el número más adecuado de operarios y máquinas en la sección de mecanizado de una empresa que tiene una determinada cartera de pedidos, estos elementos serán los que formen parte del sistema a analizar, mientras que, si lo que se desea es estudiar la capacidad productiva de la empresa, los elementos mencionados anteriormente sólo serán una parte del sistema. A ellos habrá que añadir montaje, embalaje, almacenaje, etc.

Por qué modelar, se construye modelos para:

- Comunicar y saber la estructura deseada y el comportamiento de nuestro sistema.
- Visualizar y controlar la arquitectura de nuestro sistema.
- Comprender que estamos construyendo, muchas veces descubriendo oportunidades para la simplificación y reutilización.
- Controlar el riesgo

La importancia de modelar: la simulación se compone de un conjunto de sistemas de hardware y software que se utilizan para imitar el comportamiento de una entidad o fenómeno. Por lo general, la entidad o fenómeno que se simula es del dominio de lo material que van desde la simulación de una placa de circuitos integrados, a la simulación del comportamiento de una avioneta en situaciones de mala meteorología. Un simulador, puede ser utilizado para analiza y verificar modelos teóricos que son demasiado difíciles de entender desde un nivel puramente conceptual. Es por esto que la simulación desempeña un papel crucial tanto en la industria como en la investigación.

A pesar del creciente reconocimiento de la simulación como herramienta de investigación viable y necesaria, se debe ser consciente de los problemas potenciales que puede representar la simulación. Muchos de estos problemas se encuentran relacionados con las limitaciones computacionales de hardware existentes. No obstante, este hecho está siendo rápidamente superado por la introducción de plataformas de mayor potencia.

De acuerdo al tipo de sistema, tanto en su tamaño como en características se necesitará de distintas herramientas, procesos, arquitectura, recursos humanos y las tecnologías. El modelado es una técnica de ingeniería probada y bien aceptada. Nos ayuda a:

- Comprender mejor el sistema.
- Comunicar las ideas a otros.

¿Qué ES ENTONCES MODELAR?

"UN MODELO ES UNA SIMPLIFICACIÓN DE LA REALIDAD".

Pueden involucrar planos detallados como planos más generales que ofrecen una visión global del sistema en consideración.

¿POR QUÉ MODELAMOS?

Construimos modelos para comprender mejor el sistema que estamos desarrollando.

A través del modelado se consiguen cuatro objetivos:

- Nos ayuda a visualizar como es o queremos que sea un sistema.
- Nos permite especificar la estructura o el comportamiento de un sistema.
- Nos proporcionan plantillas que nos guían en la construcción de un sistema.
- Documentan las decisiones que hemos tomado.

El modelado es útil tanto en pequeños como en grandes sistemas. Mientras más grande y complejo sea el sistema el modelado se hace importante por una simple razón:

CONSTRUÍMOS MODELOS DE SISTEMAS COMPLEJOS PORQUE NO PODEMOS COMPRENDER EL SISTEMA EN SU TOTALIDAD.

A través del modelado, reducimos el problema que se está estudiando, centrándonos en un solo aspecto a la vez. Se puede modelar formal e informalmente, pero este último no proporciona un lenguaje común que se pueda compartir fácilmente con otros. Mientras más complejo sea el sistema, requiere modelaje. Si se construye un sistema simple y este es sencillo al principio no se piensa que este necesite de modelaje, pero si este evoluciona y crece, se lamentará no haberlo realizado.

VENTAJAS DE LA SIMULACIÓN

Los procesos de simulación ayudan a los sistemas a predecir, comparar y optimizar los resultados de un proceso sin el coste y los riesgos que suponen. Su importancia radica en su utilidad para plantear la estrategia de un sistema desde el punto de vista experimental, para generar observaciones en las variables clave y el análisis estadístico de los datos resultantes.

Razones para utilizar la simulación en un sistema como herramienta de apoyo al modelado y simulación de sistemas.

- La simulación anticipa cómo un sistema puede responder a los cambios: Esto permite analizar si la infraestructura existente puede manejar la nueva simulación a emplear.

- La simulación permite un análisis de las variaciones del sistema desde una perspectiva más amplia: Los métodos convencionales de análisis, como los modelos estadísticos matemáticos, no pueden dirigir eficientemente las variaciones pues los cálculos se derivan de valores constantes. Mediante un sistema que incorpora interdependencia, la simulación tiene en cuenta las variaciones, así como la interacción entre los componentes y el tiempo.

- La simulación promueve soluciones totales: Ya que permite modelar sistemas completos.

- La simulación procura un enfoque cuantitativo para medir la actividad: La simulación puede ayudar a cuantificar las medidas de actividad del sistema.

- Permite cuantificar el impacto sobre el tiempo total del proceso de las actividades que no generan valor añadido.

- Permite a las organizaciones estudiar y reducir las oscilaciones de los procesos definiendo las actividades de mayor impacto en la variación total de cada proceso.

- Permite evaluar posibles cambios en el sistema. La modelización de recursos y jerarquía de procesos permite la visualización y evaluación de las posibles alternativas antes de tomar decisiones arriesgadas sobre cambios en el sistema.

Usos generales de la simulación

Hoy en día la simulación ofrece un amplio rango de usos en áreas como el análisis, diseño, investigación, educación, formación y entretenimiento. Antes de la llegada de los ordenadores con recursos gráficos, los resultados obtenidos a través de la simulación eran únicamente interpretables por el propio modelador y la disciplina requería de un experto programador con amplios conocimientos matemático.

Una vez el hardware mejorado permitió una mejor visualización de los datos y resultados, nuevos grupos de expertos (gerentes de empresas, personal educativo

etc.) Fueron capaces de explorar de una manera fácil los datos y soluciones obtenidos a través de las simulaciones haciendo que la fase experimental del proceso de simulación fue además accesible.

Con los últimos avances en hardware y la disponibilidad de ordenadores, la simulación ha empezado a ser utilizada cada vez más en educación, formación e investigación donde la atención se centra en el uso de los modelos para manejar sistemas o para explorar procesos.

Actualmente, se pueden dividir los usos generales de la simulación en las siguientes categorías:

1. **Investigación.** La investigación mediante el uso de simuladores, es importante para explorar la precisión y utilidad de nuevas técnicas analíticas que implican la derivación y verificación de los modelos de los sistemas. En este caso, las simulaciones se utilizan como

herramientas de investigación que permiten establecer tendencias, demostrar la relación entre los diferentes parámetros del sistema o realizar predicción es sobre aspectos futuro sin ciertos.

2. Diseño. Los diseñadores utilizan la simulación como herramienta para caracterizar o visualizar un sistema que todavía no existe y del que se pretende extraer una solución óptima. Por ejemplo, utilizando la simulación para modelar una instalación de fabricación y de esta manera tener la capacidad y los medios para experimentar con diferentes diseños, distribuciones, capacidades, maquinaria, Cadena de subministro etc. Con el objetivo de mejorar la eficiencia del sistema o proceso.

3. Análisis. El análisis hace referencia a los procesos en los cuales la simulaciones utilizada para determinar el comportamiento o capacidad o la exactitud de un sistema en uso. Puede ser también utilizado para examinar el comportamiento de sistema real es bajo condición es extremas o imposibles.

En estos casos, el comportamiento del modelo es establecido mediante la recolección de los datos que genera el sistema. Por ejemplo: La optimización de la gestión de un hospital, la simulación de los horarios del personal, equipamientos y pacientes etc.

4. Formación. La simulación formativa se utiliza para recrear situaciones reales en las que la experimentación humana es difícil o peligrosa. De esta manera la simulación permite entrenar individuos evitando cualquier tipo de daño o perjuicio causado por esta experimentación y reducciones los costes. Un amplio rango de simulaciones pueden ser llevadas a cabo, desde situaciones de alta complejidad como por ejemplo la simulación de vuelo o el diseño de centrales nucleares.

5. Educación. En el ámbito educativo, están importante saber cómo realizar algo, como saber por qué o las consecuencias. La simulación permite representar los modelos necesarios para llevar acabo la experimentación de hipótesis y la comprensión de un sistema. Esto es debido a que la simulación proporciona herramientas que explican el comportamiento de sistemas dinámicos complejos. Potencialmente, cualquier simulador puede ser utilizado en niveles educativos simples.

6. Entretenimiento. La simulación aplicaba al ámbito del entretenimiento por ordenador (juegos "árcade" o juegos de rol) requiere de un modelo consistente sobre una realidad ficticia. Muchos de ellos utilizan las técnicas de entrenamiento, diseño y análisis (por ejemplo las de optimización y control) explicadas anteriormente.

De manera específica y concreta, tomando como referencia los artículos publicados durante la Winter Simulation Conference 20113, realizada en Phoenix (Arizona) se puede tener una visión específica de los diferentes campos de aplicación de la simulación digital durante el último año:

• Modelado de procesos de negocios (Bussines proceding):

Soporte a la toma de decisiones en el sector automovilístico, gestión de flujos de trabajo, soporte de tareas administrativas etc.

• Aplicaciones medio ambientales y de sostenibilidad: Uso de las simulación para la reducción de los gases de efecto invernadero, simulación de la rapidez de expansión de incendios forestales, reducción de costes mediante el uso de sensores de luz etc.

• Aplicaciones para el sector sanitario: detección de enfermedades a través de la identificación de síntomas, optimización de la gestión de recursos médicos (ambulancias, personal, materiales etc.)

• Procesos de fabricación: fabricación sostenible, soporte para la toma de decisiones, planificación y optimización.

• Aplicaciones militares: simulación de combate, logística y movilidad.

• Emergencias: gestión de emergencias y operaciones humanitarias.

• Aplicaciones para el transporte: por ferrocarril (simulación de redes de vías y efectos), por carretera (gestión de la logística del transporte), transporte intermodal.

• Logística y supply chain management Gestión de RFID, control de inventarios, eficiencia en la distribución de materiales, minimización de la incertidumbre.

• Análisis del riesgo: simulación de eventos poco comunes.

• Calidad, estadística y fiabilidad: Diseño de experimentos de optimización.

• Gestión de proyectos: gestión de proyectos de construcción, animación 3D de proyectos.

• Educación: La simulación como herramienta para el soporte a la formación.

TIPOS DE SISTEMAS DE SIMULACIÓN

Antes de introducir el concepto de modelo de un sistema y definir los principales modelos de simulación existentes, se debe definir el concepto de sistema.

Se puede definir un sistema como un conjunto de componentes que son

Interdependientes o interactúan entre sí formando una unidad integrada o bien como un conjunto de elementos (componentes) y sus relaciones. Es decir, son entidades que interactúan entre sí para alcanzar un objetivo.

Por ejemplo, si se considera un proyecto que analiza la longitud de la cola de los mostradores de check-in de un aeropuerto dependiendo de la cantidad de mostradores operativos, los componentes principales de este sistema serían los pasajeros que esperan su turno para realizar la facturación y el personal de tierra que realiza el proceso. Se podrían considerar otros componentes secundarios como serían las maletas facturadas. En estos casos, el número de componentes considerados puede variar dependiendo del objetivo del análisis.

El estado del sistema está caracterizado por el valor que tengan las variables de estado en un instante de tiempo dado. El conjunto de variables de estado debe ser suficiente para el propósito de estudio, y puede diferir en el número y tipo de variables si los objetivos de la simulación cambian. En el ejemplo anterior, podríamos definir como variables de estado, el estado de cada uno de los mostradores de check –in (ocupado, en uso, disponible) o el número de clientes que están haciendo cola en cada mostrador.

Por lo tanto, si se considera que la finalidad de los análisis o experimentos es el estudio de los diferentes comportamientos y procesos de unos sistemas, se pueden clasificar los sistemas en las siguientes categorías (teniendo en cuenta la evolución de las variables de estado en relación al tiempo):

• **Sistemas continuos:** Las variables de estado evolucionan de forma continuada con el tiempo. Por ejemplo, el nivel de agua en un tanque evoluciona continuamente a lo largo del tiempo tomando cualquier valor de un rango continuo.

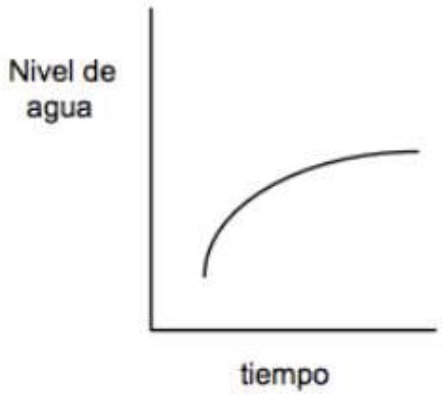


Fig. 1.1 Gráfica que presenta un ejemplo de sistema continúo.

• **Sistemas discretos:** Es un sistema en que las variables de estado cambian en ciertos instantes de tiempo y permanecen constantes durante un periodo de tiempo. Los cambios en los sistemas obedecen un patrón. Por ejemplo, el número de pedidos realizados en un restaurante de comida rápida y que deben ser completados.

Fig. 1.2. Gráfica que presenta un ejemplo de un sistema discreto.

Antes de proceder con la descripción de los sistemas orientados a eventos discretos es necesario introducir el concepto de evento en simulación. Un evento puede de finirse como:

"Una acción que ocurre de forma instantánea y que causa la transición de un estado discreto a otro"

• **Sistemas orientados a eventos discretos:** En un sistema orientado a eventos discretos, las variables toman valores de un conjunto discreto de valores en instantes concretos de tiempo que no obedecen ningún patrón estático. Por ejemplo, continuando con el proceso de check-in el siguiente gráfico representa el estado de los mostradores (ocupado=1 y libre=0) a lo largo del tiempo.

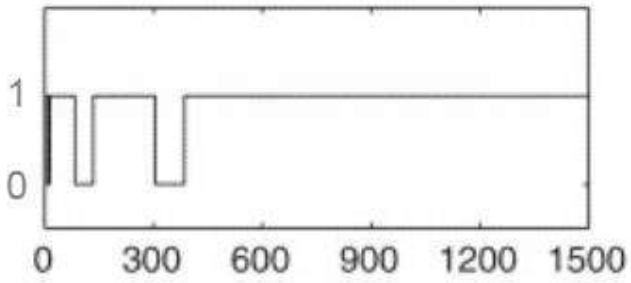


Fig. 1.3. Gráfica que representa un ejemplo de un sistema orientado a eventos discretos

TIPOS DE MODELOS DE SIMULACIÓN

Un modelo es cualquier representación, lo más simplificada posible, de las dinámicas de interés de un sistema, que permita, comprender, explicar, cambiar, prever y posiblemente controlar el comportamiento de un sistema.

Existen diferentes metodologías para el desarrollo de modelos matemáticos de sistemas físicos. Con el fin de garantizar una representación eficiente del sistema real se deben tener en cuenta un conjunto de consideraciones:

- Un modelo es siempre una aproximación del sistema real basado en hipótesis y aproximaciones. En consecuencia, nunca se podrá considerar una representación perfecta del sistema real.
- Un modelo siempre es construido para un objetivo específico. Por lo tanto debe ser formulado de manera que sea útil para responder a este.
- Un modelo es siempre el resultado de un compromiso entre simplicidad y la necesidad de incorporar todos los aspectos relevantes del sistema.

Un modelo por lo tanto, debe asegurar que cumple con la propiedad que se exponen a continuación:

• Debe ser capaz de representar las características del sistema que se pretendan analizar.

• Debe ser una representación abstracta de alto nivel, con el objetivo de facilitar su comprensión, mantenimiento, adaptación y reutilización.

Teniendo en cuenta las consideraciones y propiedades descritas anteriormente se pueden clasificar los modelos en las siguientes aproximaciones:

Modelos estáticos vs Modelos dinámicos

Se entiende por modelo de simulación estático, la representación de un sistema en un instante particular en el tiempo o bien la representación de un sistema, en el que el avance en el tiempo no es una variable considerada. Por ejemplo, la simulación de Montecarlo.

En cambio un modelo de simulación dinámico representa un sistema en el que el tiempo es una variable de interés, dado que permiten deducir como estas variables evolucionaran respecto al tiempo. Un ejemplo podría ser un modelo que simule el avance de un incendio forestal en relación al transcurso del tiempo.

Modelos deterministas vs Modelos estocásticos

Un modelo determinista es aquel en el que un nuevo estado puede ser definido de manera completa a partir de los estados previos y sus entradas. Estos modelos no se comportan siguiendo ninguna ley probabilística, por lo que cada simulación, retorna los mismos resultados y no es necesario repetir esta simulación repetidamente.

Sin embargo, muchos sistemas se modelan tomando en cuenta algún componente probabilístico aleatorio de entrada (modelos estocásticos).En consecuencia, durante la fase de experimentación, el modelo genera diferentes conjuntos de salidas, por lo que será necesario realizar repetidas simulaciones para estimar los valores de las variables del sistema. Si tomamos

como ejemplo de modelo estocástico, un sistema de inventarios de una fábrica, veremos como la salida que presenta el sistema dependerá de variables de carácter aleatorio (demanda, estacionalidad etc.). Por consiguiente la información obtenida únicamente podrá ser utilizada a modo de estimación del modelo real.

Modelos continuos vs Modelos discretos.

Los modelos de simulación discretos y continuos, se definen de manera análoga a los sistemas discretos y continuos explicados anteriormente.

Los modelos continuos se caracterizan, por representar la evolución de las variables de interés de forma continua. Por el contrario los modelos discretos, se caracterizan por representar la evolución de las variables de interés de forma discreta.

No obstante, se debe tener presente que un modelo discreto de simulación no siempre se usa para modelar un sistema discreto. La decisión de utilizar un modelo discreto o continuo para simular un sistema en particular, depende de los objetivos específicos de estudio y no de las características del sistema.

Poniendo como ejemplo, un modelo de flujo de tráfico en una carretera, puede ser discreto si las características y el movimiento de los vehículos en forma individuales importante. En cambio si los vehículos pueden considerarse como un agregado en el flujo de tráfico, entonces se puede usar un modelo basado en ecuaciones diferenciales presentes en un modelo continuo.

A modo de conclusión del concepto de modelo, tal y como describen los autores en la publicación “Modelado y simulación. Aplicación a procesos logísticos de fabricación y servicios” a continuación se presentan algunas definiciones generales válidas y algunas características que debe cumplir un buen modelo:

• Un modelo es un objeto o concepto que utilizamos para representar cualquier otra entidad compleja (un sistema). Así pues, mediante un proceso de abstracción, se muestran en un forma to adecuado las características de interés de un objeto (sistema) real o hipotético.

• Un modelo es una representación simplificada de un sistema que nos facilitará explicar, comprender, cambiar, preservar, prever y posiblemente, controlar el comportamiento del mismo.

• Un modelo puede ser el substituto de un sistema físico concreto.

• Un modelo debe representar el conocimiento que se tiene de un sistema de modo que facilite su interpretación, formalización tan sólo los factores relevantes para los objetivos del modelado.

• Un modelo debe ser tan sencillo como sea posible (ya que el desarrollo de los modelos universales es impracticable y poco económico) siempre y cuando represente adecuadamente los aspectos de interés.

1.2 Objetivo

1.2.1 Objetivo general

El objetivo de esta tesis consiste en presentar la importancia del modelado y simulación de sistemas en el software 20-SIM, para así evitar pérdidas económicas y humanas al momento de desarrollar los sistemas de manera física.

1.2.2 Objetivos particulares

Para lograr de manera eficiente el trabajo en esta tesis se requiere de los siguientes objetivos particulares.

- La instalación correcta del software 20-SIM.
- Manejo adecuado para realizar el modelado y la simulación del sistema que se desea implementar.

1.3 Justificación

Al elegir el tema de tesis que es el MODELADO Y SIMULACION DE SISTEMAS DISCRETOS EN 20-SIM, es porque es un útil el modelado de cualquier sistemas ya que nos ayuda a comprender su comportamiento al ingresarle las variables que deseamos para así obtener el resultado que deseamos, ya que con el modelado se estará evitando la perdida de máquinas, sistemas de manera física, perdida de dinero. Porque ya una vez realizado el modelado ya tenemos conocimiento de lo que pasara si se alteran las variantes, una vez ya echo el modelado sabremos con certeza que variables debemos de ingresar para así no dañar el sistema, así mismo tener ya planificado el sistema que deseamos presentar de manera física, sin tener ningún riego ya sea económico y de personal.

1.4 Metodología

Tipo de investigación:

El presente trabajo de tesis se basa en investigaciones teóricas, de campo y bibliográficas.

Investigación teórica: el trabajo de investigación comienza con la recolección de información, que permiten tener una visión clara del problema que se resolverá en el presente

trabajo y seleccionar las herramientas más adecuadas para resolver el problema de manera satisfactoria.

1.5 Contenido de la tesis

A continuación, se aborda la forma de la clasificación de los capítulos y un pequeño resumen de lo que contienen.

CAPITULO 1 INTRODUCCIÓN: En este capítulo se explica sobre la importancia del proyecto de tesis, así como los antecedentes que dieron origen a la idea.

CAPITULO 2 ANTECEDENTES DE SOFTWARE DE SIMULACIÓN: En este capítulo se da la información básica para entender la simulación de sistemas que se implementa en el actual proyecto. El capítulo comienza con un poco de historia sobre la evolución de software de simulación. Se menciona los tipos de simulación. Después se mencionan algunos softwares de simulación.

CAPÍTULO 3 MODELADO DE SISTEMAS DISCRETOS: Se da una explicación que es el control digital, además se expone una figura esquemática del control por computadora para un entendimiento mejor. Se detallan las características del control por computadora. Se mencionan los elementos básicos de un sistema de control digital. Los conceptos de frecuencia en señales en el tiempo continuo y en tiempo discreto. Se detallan las características de conversión Analógico-Digital y Digital-Analógico.

CAPÍTULO 4 SIMULACIÓN DE SISTEMAS DISCRETOS EN 20-SIM: Se explica la simulación en el software 20-Sim, todo lo que se puede realizar con 20-Sim, ya que es una

infinidad de formas para el modelado y simulación de sistemas. Como es su editor, crear ecuaciones, las herramientas del modelado así como animaciones en 3D y mucho más.

CAPÍTULO 5 SIMULACIONES DE SISTEMAS DISCRETOS: En este capítulo como su nombre lo dice se realizan simulaciones de casos de estudio, se puede observar en las imágenes, en la graficas se puede observar la respuesta del sistema que se está simulando.

CAPÍTULO 6 CONCLUSIONES Y RECOMENDACIONES: Los sistemas discretos han sido un desafío en el modelado y simulación, ya que se consideran sistemas que no son fácilmente manejables. También, las herramientas matemáticas en su modelado, análisis y control genera dificultades a los estudiantes de ingeniería.

Por lo tanto, en este trabajo de tesis, se presenta algunas de las herramientas del modelado de sistemas en tiempo discreto y su aplicación en el software 20-sim

Este trabajo de tesis es básico, por lo que puede ser fácilmente extendido a sistemas con varias funciones de transferencia y varias retroalimentaciones.

Se puede aplicar controladores en el dominio discreto P, PI o PID.

Capítulo 2. Antecedentes de Softwares de Simulación

2.1 Introducción

Por *simulación* se entiende la emulación del comportamiento e un sistema real por otro artificia, haciendo especial referencia a la posibilidad de realizar experimentos manipulando y controlando variables del sistema artificial (construido en nuestro ordenador o algún programa que ara dicha tarea), que ha sido modelado previamente para que se comporte de forma análoga al sistema real. En el supuesto de que el modelo se comporte adecuadamente, las simulaciones que se efectúen sobre aspectos desconocidos del sistema, resultaran plausibles, es decir, las novedades que se revelen mediante experimentación en el sistema artificial, caben esperarse también en la realidad.

Una **simulación por computadora**, un **modelo de simulación por computador** o un **modelo informatizado** es un programa informático o una red de ordenadores cuyo fin es crear una simulación de un modelo abstracto de un determinado sistema. Las simulaciones por computadora se han convertido en una parte relevante y útil de los modelo matemático de muchos sistemas naturales de ciencias como la física, geofísica, astrofísica, química y la biología; así como de sistemas humanos de economía, psicología y ciencias sociales. Además, se emplea en el diseño de nueva tecnología para llegar a comprender mejor su funcionamiento.

Las simulaciones por computadora abarcan desde programas informáticos cuya ejecución dura unos minutos hasta conjuntos de ordenadores conectados en red cuya ejecución dura horas, e

incluso hay simulaciones que se extienden varios días. La variedad de acontecimientos que se pueden recrear mediante simulaciones por computadora ha superado con creces las posibilidades del modelo matemático tradicional de lápiz y papel. Especialmente en aquellos contextos de investigación donde los sistemas son demasiado complejos para la experimentación tradicional, o con dificultades prácticas insuperables.

2.2 HISTORIA DE LA SIMULACION

La historia y la evolución de la simulación por ordenador han ido paralelas a la evolución de la Informática. Se podría considerar que la simulación nace en 1777 con el planteamiento del problema de "La aguja de Buffon", un método matemático sencillo para ir aproximando el valor del número "π" a partir de sucesivos intentos. El planteamiento matemático de este problema se basa en: teniendo una aguja de longitud determinada que es lanzada sobre un plano segmentado por líneas paralelas separadas en unidades. ¿Cuál es la probabilidad de que la aguja cruce alguna línea?

En 1812 Laplace mejoró y corrigió la solución de Buffon y desde entonces se conoce como solución Buffon? Laplace. Posteriormente, el estadístico William Sealy Gosset, que trabajaba en la destilería de Arthur Guinness, ya aplicaba sus conocimientos estadísticos en la destilería y en su propia explotación agrícola. El especial interés de Gosset en el cultivo de la cebada le llevo a especular que el diseño de experimentos debería dirigirse no solo a mejorar la producción media, sino también a desarrollar variedades de cebada cuya mayor robustez permitiese que la producción no se viese afectada por las variaciones en el suelo y el clima.

Para evitar futuras filtraciones de información confidencial, Guinness prohibió a sus empleados la publicación de cualquier tipo de artículo independientemente de su contenido, de

ahí el uso que hizo Gosset en sus publicaciones del seudónimo "Student", para evitar que su empleador lo detectara. Es por esta razón que su logro más famoso se conoce como la "distribución t de Student", que de otra manera hubiera sido conocida como la "distribución t de Gosset".

Este hito histórico abrió las puertas a la aplicación de la simulación en el campo del proceso de control industrial así como a las sinergias que generaba esta simulación basada en la experimentación y técnicas de análisis para descubrir soluciones exactas a problemas clásicos de la industria y la ingeniería.

Durante la segunda Guerra Mundial dos hechos sentaron las bases para la rápida evolución del campo de la simulación. En primer lugar, la construcción de los primeros computadores de propósito general como el ENIAC (Electronic Numerical Integrator And Computer).

La simulación por computadora se desarrolló a la par que se produjo el vertiginoso progreso del ordenador. Su primer despliegue a gran escala fue en el Proyecto Manhattan, durante la Segunda Guerra Mundial, para recrear una detonación nuclear. Se empleó el Método de Montecarlo o Simulación Monte Carlo" se agrupan una serie de procedimientos que analizan distribuciones de variables aleatorias usando simulación de números aleatorios. El Método de Monte Carlo da solución a una gran variedad de problemas matemáticos haciendo experimentos con muestreos estadísticos en una computadora. El método es aplicable a cualquier tipo de problema, ya sea estocástico o determinístico. Las simulaciones por computadora a veces complementan o incluso sustituyen a los sistemas de modelización para los que no es posible hallar soluciones analíticas de forma cerrada. Existen muchos tipos de simulación por computadora, pero todos ellos comparten una característica común: tratan de generar una muestra de escenarios representativos para un modelo en el que una relación completa de todos los estados posibles de

este sería muy costoso o imposible. Los modelos informatizados se emplearon inicialmente como suplemento de otros parámetros, pero más adelante su uso se extendió a otros ámbitos.

La simulación computarizada fue utilizada por primera vez en la industria de Defensa en los años 50. Los primeros modelos de simulación fueron construidos utilizando lenguajes de programación tales como FORTRAN y Run On Mainframes. En los años 80, otro gran desarrollo de la simulación fue la explosión de los usuarios de ordenadores personales que llevó la empresa Microsoft en el entorno operativo. Instrumentos de simulación como Writness, Pro-model, Arena y Ithink mostraron con los menú-driver interfaces modelos visuales interactivos y posibilidades de animación impresa. Estos desarrollos afectaron significativamente a la extensión del uso de los lenguajes de simulación. La primera mitad de los noventa aportó otro interesante desarrollo de la simulación. Existe una relación natural entre los modelos objeto orientado y la simulación. Los lenguajes de simulación tales como Modsim y Simple ++ representaron otro hito en la simulación, aprovechando la tecnología del objeto y permitiendo bibliotecas de "objeto reutilizable". Este desarrollo está favoreciendo el desarrollo de soluciones en el campo específico de la simulación, haciendo así la simulación útil para un mayor número de usuarios finales.

Independientemente de los diferentes desarrollos del pasado y del presente, es evidente que la simulación ha sido siempre un poderoso instrumento. Creemos que el futuro de la simulación se basa en la posibilidad de distribuir modelos en una red. La aplicación más pragmática de la misma sería su uso en la red mundial. Existen ya muchas investigaciones y desarrollos que se están realizando en esas áreas. En la próxima década por tanto, un incremento en el uso de la simulación, la tecnología orientada al objeto y la web cambiarán dramáticamente la forma de utilizar la simulación en el futuro.

A partir de la década de los 60 empiezan a aparecer en el mercado programas de simulación de sistemas de acontecimientos discretos que poco a poco se empezaron a utilizar para resolver problemas de ámbito civil. Los más destacables fueron el GPSS de IBM (General Purpose System Simulator) y el SIMSCRIPT. Los modelos de acontecimientos discretos son muy utilizados en la actualidad para estudiar problemas de fabricación de procesos, logística, transporte, comunicaciones y servicios. Estos problemas se caracterizan por centrar su interés en los cambios que hay en el sistema como consecuencia de los acontecimientos y en su capacidad para modelar los aspectos aleatorios del sistema. Este simulador se utilizó para analizar el diseño de la terminal (Barcelona) en lo que respecta a los distintos espacios, el movimiento de las personas en situaciones normales y el análisis del plan de evacuación.

Complementariamente a los desarrollos llevados a cabo por RAND e IBM, el Royal Norwegian Computing Center inició en1961 el desarrollo del programa SIMULA con ayuda de Univac. El resultado fue SIMULA I, probablemente el lenguaje de programación más importante de toda la historia.

En 1967 se fundó la WSC (Winter Simulation Conference), lugar donde desde entonces y hasta ahora se archivan los lenguajes de simulación y aplicaciones derivadas, siendo en la actualidad el referente en lo que a avances en el campo de los sistemas de simulación se refiere.

En los años posteriores (1970 [1981), el campo de la simulación sobrellevo un periodo de expansión durante el cual se desarrollaron avanzadas herramientas de modelado y de análisis de resultados. Gracias también a los desarrollos obtenidos en la generación de datos y a las técnicas de optimización y representación de datos, la simulación llega a su fase de expansión donde comienza a aplicarse en múltiples campos.

En esos momentos la simulación era todavía un aspecto utilizado en las escuelas de Ingeniería pero raramente aplicado. El éxito y la popularidad de la simulación como una herramienta potente incremento a raíz de las diversas conferencias que se realizaron durante los siguiente años. Aún y este hecho durante los primeros años de los 80 se continuaba viendo a la simulación como una herramienta extremadamente complicada que solo podía ser utilizada por expertos.

No obstante, el número de sistemas computarizados aumentó de los cuatro existentes en 1970 a más de doscientos a principios de 1980. Durante ese periodo, la mayor parte de los software de simulación existentes se concentraban en el la planificación de necesidades de materiales (MRP).

A finales de los ochenta se vieron nuevos desarrollos como SIMANIV y CINEMAIV, que incorporaban software de uso fácil con menús y pestañas así como mejoras en la animación. De esta manera se llegó en 1984 al desarrollo del primer lenguaje de específicamente orientado a la simulación de procesos de producción.

En estos últimos años han surgido muchos tipos de lenguajes de programación para diseñar software de simulación de facilidad de uso para los usuarios. Antes, la aleatoriedad de los coeficientes en los modelos de simulación hacia su resolución inviable, actualmente con el apoyo de la Informática, la resolución de estos modelos resulta muy sencilla. Los paquetes de software y lenguajes de programación han proliferado.

Es importante señalar que, en general, las características que tiene que tener un software de simulación son de funcionalidad, utilidad, bajo coste y buena calidad de mantenimiento del proveedor.

2.3 TIPOS

Los modelos computacionales pueden clasificarse atendiendo a distintos pares de atributos, a saber:

- Estocástico o determinista
- Estático o dinámico
- Continuo o discreto
- Local o distribuido

Las ecuaciones definen las relaciones existentes entre los elementos del sistema modelado y tratan de encontrar un estado en el que el sistema esté en equilibrio. Esta clase de modelos se emplean habitualmente para simular sistemas físicos, esto es, a modo de modelaje más sencillo antes de pasar al modelado dinámico.

Los modelos de simulación dinámica cambian en un sistema en respuesta a señales de entrada.

Los modelos estocásticos emplean generadores de números aleatorios para simular el azar o una serie de acontecimientos aleatorios.

Una simulación de un acontecimiento discreto (DES, del inglés *Discrete event simulation*) manipula acontecimientos en el tiempo. La mayoría de las simulaciones por computadora de tests de lógica y arborigramas de fallos son de este tipo. En este tipo de simulación, el simulador tiene una lista de acontecimientos ordenados por el tiempo al que deberían suceder. El simulador lee la lista y activa nuevos acontecimientos a medida que se procesa otro. No es importante ejecutar la simulación en tiempo real, sino que normalmente se le da más importancia al poder acceder a los

datos producidos por la simulación para descubrir defectos lógicos en el diseño o en la sucesión de acontecimientos.

Una simulación de movimiento continuo proporciona una solución numérica a ecuaciones diferenciales algebraicas o ecuaciones diferenciales (tanto ecuaciones diferenciales en derivadas parciales como ecuaciones diferenciales ordinarias). A intervalos regulares, el programa de simulación resuelve todas las ecuaciones y utiliza los números para cambiar el estado y la salida de la simulación. Entre las aplicaciones se incluyen simuladores de vuelo, videojuegos de construcción y gestión, modelados de procesos químicos y simulaciones de circuitos eléctricos. En un principio, este tipo de simulaciones se ejecutaban en ordenadores analógicos, en los que se podían representar las ecuaciones diferenciales mediante distintos componentes eléctricos como amplificadores operacionales. Sin embargo, a partir de finales de los años 80, la mayor parte de las simulaciones analógicas se ejecutaban en ordenadores digitales que emulaban a los ordenadores analógicos.

Un tipo especial de simulación discreta que no se sustenta en un modelo basado en una ecuación, pero que, no obstante, puede representarse formalmente, es la simulación*agent-based*. En esta simulación, las entidades individuales (como, por ejemplo, moléculas, células, árboles o consumidores) del modelo se representan directamente (en lugar de por su densidad o concentración) y poseen un estado interno y un conjunto de comportamientos o reglas que determinan cómo se actualiza el estado del «agente»*(agent)* de un salto de tiempo al siguiente.

Los modelos distribuidos se ejecutan en una red de ordenadores interconectados, posiblemente a través de Internet. A este tipo de simulaciones dispersas en distintos ordenadores centrales se las conoce con el nombre de «simulaciones distribuidas». Existen diversos standars para las

simulaciones distribuidas, entre los que se encuentran el Aggregate Level Simulation Protocol (ALSP), el Distributed Interactive Simulation (DIS),el High level architecture (simulation)(HLA) y el Test and Training Enabling Architecture (TENA).

La SCS (Society for Computer Simulation) publica cada año una lista actualizada del software disponible.

En la tabla 2.1 se ha pretendido resumir los hitos más importantes. En la tabla 2.2 se hace referencia de algunos de los programas informáticos comercializados que tiene relación directa con la simulación de sistemas oleo hidráulicos.

En la tabla 2.1 se han incluido también algunos de los avances tecnológicos más significativos habida cuenta que en algunos periodos la tecnología ha sido un factor clave en el desarrollo de la simulación. Sin embargo, es interesante observar que las ideas cambian más lentamente que la tecnología.

Es solamente a partir de los 90 que se hace necesario un cambio de paradigma.

Este cambio ha sido motivado por los usuarios.

Los usuarios requieren:

1. la simulación de sistemas complejos multidisciplinares

2. la programación avanzada orientada al objeto,

3. software para la resolución de sistemas diferenciales algebraicos,

4. la computación simbólica y

5. métodos gráficos avanzados

Las metodologías modernas se construyen sobre la base de una modelización NO causal con ecuaciones matemáticas y el empleo de construcciones orientadas al objeto para facilitar la reutilización de conocimiento modelado.

2.3.1 Simulación Analógica

Los primeros simuladores fueron analógicos. La idea es modelar un sistema en términos de ecuaciones diferenciales ordinarias y después hacer un dispositivo físico que obedezca a las ecuaciones. El sistema físico se inicializa con valores iniciales apropiados y su desarrollo en un cierto plazo que simula la ecuación diferencial. Inicialmente se desarrollaron analizadores diferenciales mecánicos como herramienta de propósitos generales para simular sistemas dinámicos los cuales fueron reemplazados por sistemas electrónicos.

La simulación analógica no puede tratar con ecuaciones diferenciales algebraicas (EDAs), sólo con ecuaciones diferenciales ordinarias (EDOs), lo cual no es muy grave, según Broenink J. F (1999), las EDAs se dan cuando se simplifica demasiado el modelo.

2.3.2 Simulación Numérica

La solución numérica de una ecuación diferencial es un esencial ingrediente de la simulación numérica.

Hay varias maneras de encontrar soluciones de aproximación numérica para las ecuaciones diferenciales. Los métodos son basados en la idea de reemplazar las ecuaciones diferenciales por una ecuación de diferencia. El método de Euler es basado en aproximación de la derivativa por una diferencia de primer orden. Hay técnicas más eficientes tales como Runge-Kutta y métodos de múltiple pasos. Estos métodos fueron muy conocidos cuando emergieron los simuladores

digitales en el año de 1960. El campo de las matemáticas numéricas experimentó un renacimiento debido al impacto de las computadoras digitales.

La integración numérica de ODEs y DAEs son campos muy activos de la investigación que continúan teniendo impacto fuerte en el modelado y simulación, considera Hairer y Wanner (1991). Entre el desarrollo interesante está el algoritmo mejorado, una estructuración mejor del código en donde se separan los algoritmos y el error de control. Los algoritmos para ecuaciones algebraicas todavía no están muy desarrollados como los algoritmos para ecuaciones diferenciales ordinarias.

2.3.3 Simulación Analógica Digital

Cuando las computadoras digitales aparecieron, era natural explorar si pudiesen ser usadas para la simulación. El desarrollo fue iniciado por Selfridge (1955) que demostró como una computadora digital puede emular un analizador diferencial. Este enfoque dio lugar a la aparición de numerosos lenguajes de programación. En este medio se desarrolló CSSL estándar [Strauss (ed) (1967)], fue un mayor hito, puesto que unificó los conceptos y estructuras de lenguaje de los programas disponibles de simulación. ACSL de Mitchell y Gauthier (1967), se basa en CSSL pero con ciertas modificaciones y mejoras. SIMNON fue desarrollado en la universidad de Lund iniciando en 1972 y distanciándose de CSSL estándar [Elmqvist (1975)].

El desarrollo en esta dirección sólo fue posible con la aparición de computadoras con buenas prestaciones en donde el PC con gráficos de tramas llegó a estar generalmente disponible.

Entre las aplicaciones de este tipo están VisSim [Darnell y Kolk (1990)].

Mitchell y Gauthier introdujeron el modelador gráfico ACSL en 1993 o SIMULINK.

BDSP (Block Diagram Simulation Program) es un programa de simulación usado en el entorno Windows desarrollado por la universidad de Gifu. Ofrece una librería que cubre un amplio rango de áreas físicas y aplicaciones que facilitan la construcción de los modelos. Este programa posee una rutina para identificar modelos a partir de la estimación experimental de su función de transferencia [Yamada y Muto (2001)].

SIMULINK (originalmente llamado SIMULAB) que se integra con MATLAB, apareció en 1991 [Grace (1991)]. Es especialmente diseñado para trabajar con diagramas de bloques usando MATLAB para el análisis dinámico del sistema. Los diagramas de bloques de SIMULINK pueden ser definidos como ecuación de estado.

A continuación se puede utilizar las funciones MATLAB para resolver numéricamente o procesarlas de diferentes maneras. De esta forma, es posible obtener la función de transferencia, el diagrama de Bode, margen de ganancia, etc.

Este programa es muy útil en el campo de control automático.

2.3.4 Simuladores Específicos

Es posible diseñar el entorno del modelo, que son muy fáciles de usar, limitando el dominio del modelo. Inicialmente, estos programas solo abarcaban una rama de la ingeniería. Algunas herramientas de este tipo son: el sistema SPICE, el cual fue desarrollado para el modelo analógico de un circuito eléctrico, y DADS, desarrollado en la universidad de Iowa (1984), para la simulación de sistemas mecánicos. Un gran número de herramientas de este tipo ha sido desarrollado en varias ramas de la ingeniería.

El programa Hysis, desarrollado por Mannesmann Rexroth, orientado al cálculo dinámico en el campo de la oleo hidráulica, comprende cuatro subprogramas bien definidos para la simulación no

lineal de circuitos electro-hidráulicos de regulación. Cada subprograma, disponibles para aplicaciones prácticas, permite simular un tipo diferente de sistema:

1) HYVOS: accionamientos que conciernen al actuador lineal controlado por una válvula

2) HYDRA: accionamientos que conciernen al actuador rotativo controlado por una válvula

3) HYSTA: accionamientos hidrostáticos

4) HYSEK: accionamientos con regulación secundaria

La ventaja del programa Hysis, según Murrenhoff (1998) es su notable facilidad de empleo siempre que el modelo desarrollado se pueda clasificar dentro de uno de los cuatros subprogramas.

2.3.5 Simuladores Multi-Disciplinares

El software para dominios específicos es muy útil de usar si el problema es adecuado a la herramienta directamente, siendo muy útiles en su campo de aplicación, pero hay muchos diseños que necesitan de programas de simulación que permitan la colaboración con otros programas.

Como ejemplo esté el programa ITI – SIM (ITI GMBH; Dresden Alemania), ver

Klein y Grätz (2001). Es un software de simulación que cubre un amplio rango de áreas físicas y aplicaciones: hidráulica, neumática, mecánica, térmicas, transmisión electro – mecánica y bloque de señales. Este programa permite definir los modelos para facilitar su utilización. De esta forma la construcción de un modelo es adaptada a la descripción más común para cada disciplina de la ingeniería por medio de diagramas de circuitos hidráulicos, para dispositivos mecánicos y diagramas de bloques para la estructura de control. Incluye una librería con numerosos componentes, razón por la cual, la posibilidad de construir nuevos componentes definidos por el usuario, debe ser considerado como un rasgo importante. ITI – SIM proporciona un fichero DLL

que permite modelar componentes especiales. El archivo DLL puede ser creado por cualquier compilador (Microsoft C++ ó MATLAB / SIMULINK).

2.3.6 Simulación Física

Un procedimiento físico para el modelado físico es dividir un sistema en subsistemas y explicar el comportamiento en los nodos de conexión. Cada subsistema puede tener numerosos niveles, en el cual, el nivel inferior representan elementos básicos que se agrupan para formar un subsistema de nivel superior. El modelo completo se obtiene agrupando los subsistemas.

Según Broenink (1999), se deben cumplir dos condiciones para garantizar que los submodelos sean acoplables:

1. Los puertos de conexión entre submodelos están definidos como pares de variables. El uso de pares de variables conjugadas en potencia hacen que las conexiones sean físicas, por ejemplo: par y velocidad, presión y caudal.

2. Los submodelos han de estar escritos en estilo declarativo, es decir, estableciendo relaciones y no procedimientos para computar. Esta descripción también se llama causal.

Los submodelos se han de describir aplicando ecuaciones de conservación, aunque también puede ser necesario describir las propiedades de los elementos usados. El sistema de ecuaciones se obtiene a través de la combinación de las ecuaciones de los submodelos y de los puertos de conexión. Esto conduce, naturalmente, a las ecuaciones diferenciales algebraicas (EDAs).

Bond Graphs y Modelica estándar son ejemplos de programas que usan este tipo de modelo.

A) Dymola

Dynamic Modeling Languaje (Dymola) de Elmqvist (1978), fue un esfuerzo primitivo para ayudar al modelo físico. Actualmente utiliza el lenguaje Modélica y es deseado para el modelado de muchos dominios tales como circuitos eléctricos, sistemas termodinámicos, procesos químicos, sistemas de control, etc. Ofrece una librería para todos los dominios tecnológicos en lo que es aplicable. Para la mejor comprensión del sistema, los submodelos y sus conexiones están representados por iconos.

B) Bond Graphs

Bond graphs es dirigida a gráficas en donde los subsistemas son nodos y los flujos de energía en el sistema es presentados por ramas [Karnopp y Rosenberg (1968)]. Una de las ventajas de Bond Graphs es que puede pasar fácilmente de un dominio a otro a través de los elementos Transformer y Gyrator. Además se puede modificar fácilmente el modelo de un sistema añadiendo y/o quitando elementos.

Algunos de los programas disponibles son los siguientes:

Bond Graphs Simulation Program (BGSP) del laboratorio de Ingeniería Mecánica del Ministerio de Comercio e Industria de Japón. Es un programa de solución basado en Bond Graphs. Necesita de un programa externo para resolver numéricamente las ecuaciones de estado.

20-SIM fue desarrollado en la universidad de Twente, Holanda. Con este programa se puede simular el comportamiento de sistemas dinámicos tales como sistemas eléctricos, mecánicos e hidráulicos o cualquier combinación de estos sistemas. Se puede entrar al modelo en una forma gráfica por medio de un editor gráfico. Ofrece un modelado, simulación y entorno de análisis para sistemas de ingeniería que apoya a la jerarquía del modelo que utiliza diferentes lenguajes en cada

elemento. Puede trabajar con Bond Graphs multipuerto, con diagrama de bloques, diagrama iconos y con ecuaciones escritas en SIDOPS+. El programa se divide en dos partes, con la primera se introduce el modelo y se compila y con la segunda se realizan experimentos sobre el modelo. En el último caso, el usuario tiene la posibilidad de escoger el algoritmo de integración.

Los sistemas de modelado físico tienen el inconveniente, de que en principio, no se puede aplicar a sistemas con parámetros distribuidos. Sin embargo, en algunos casos, se puede obtener una solución aproximada dividiendo el componente continuo en numerosos elementos a los que se le asignan parámetros concentrados.

Esta solución tiene varias limitaciones:

1. Exige mucha memoria y velocidad de proceso a la computadora

2. La solución obtenida solo es válida para bajas frecuencias. Esto implica que si aparecen modos propios en la simulación, solo son aceptables los que tienen la frecuencia más baja.

Periodo	**Concepto**	**Desarrollos significativos**
Años 20-50	Simulación con técnicas analógicas basadas en las ecuaciones diferenciales ordinarias y diagramas de bloques.	
	Simulación con técnicas analógicas basadas en las ecuaciones diferenciales ordinarias y diagramas de bloques.	Analizador diferencial mecánico Bush y otros, en el MIT (1931)
		Analizador diferencial

		Electrónico Ragazzini y otros (1947)
Años 60	Simuladores digitales	Selfridge (1995) demostró que un computador digital podía emular a un simulador Analógico
	Optimización de los métodos numéricos para resolución de las ecuaciones diferenciales ordinarias basados en sustituir las ec. Diferenciales por ecuaciones en diferencias finitas.	
	Estabilidad de las aproximaciones de diferencias finitas	
	Ajuste automático de la longitud del paso de integración	Fehlberg (1964)
	Eclosión de los programas Comerciales	MIMIC (Wright Patterson-1965) DYNASAR (General Electric-1965) DSL/90 (IBM-1965) CSMP (IBM-1968)
	Unificación de los conceptos y estructuras de los lenguajes de los	CSSL report (Strauss ed.-1967)

	programas de simulación comerciales	
Años 70	Métodos numéricos para ecuaciones diferenciales algebraicas (EDA)	Métodos numéricos de integración de EDA, p.e Gear (1971)
	Nota: Los algoritmos de las ecuaciones diferenciales algebraicas aún no están tan bien desarrollados como las ecuaciones diferenciales ordinarias. Sigue siendo un campo de frenética Investigación	Hairer and Wanner (1991)
	Eclosión de programas comerciales basados en el CSSL Nota: en ellos es posible trabajar con la modelización de sistemas discretos y sistemas continuos en el tiempo	Ejemplos: ACSL (Mitchell y Gauthier- 1976) SIMNON (Elmqvist-1975)
	Desarrollo de entornos gráficos con el usuario. Empleo de tubos de rayos catódicos (CRT) y lápices luminosos para dibujar los diagramas de bloques. Nota: limitado por tecnología.	Van der Bosch and Bruijin (1977) EASY5 (1976)
	Manipulación simbólica de formulas	Tarjan (1972) Wiberg (1977)

	Se utiliza para la convertir las ecuaciones EDA en EDO. Aplicación de las teorías de grafos	
	Primeras ideas sobre lenguajes orientados al objeto y descomposición por niveles jerárquicos	Simula (Birtwistle y otros-1973)
Años 80	Desarrollo de entornos matriciales	SystemBuild integrado con MATRIX (1985) SIMULINK integrado con MATLAB (1991) VisSim (1990) ACSL Graphics (1993)
	Primeros pasos hacia la modelización física (se requiere un nuevo paradigma para la modelización)	Tutsim (Bond Graph) 20Sim (version actualizada del Tutsim, Broenink-1997) Dymola (Elmqvist –1978, primera versión comercial 1992)
	Desarrollo de lenguajes orientados al objeto	Omola (Andersson-1988)
	Disponibilidad de máquinas con altas prestaciones: WorkStation y PC con potentes entornos gráficos	

	Disponibilidad de algoritmos numéricos para resolver EDA	Brenan y otros (1989) Hairer y otros (1989)
Años 90	Aparición de una amplia variedad de lenguajes con ideas similares a Dymola Nota: situación similar a los años 60 cuando se acordó definir el CSSL	OmSim (Mattsson-1993) ASCEND (Piela y otros-1991) GPROMS (Sahlin y otros-1996) OjectMath (Fritzson y otros-1995) SIDOPS+ (Broenink-1997) Smile (Kloas y otros-1995) ULM (Jeandel y otros-1996)
	Unificación de técnicas e ideas de los diferentes programas de simulación con el objetivo de alcanzar un alto grado de conocimiento sobre la modelización física lenguajes orientados al objeto aplicación a diferentes dominios (multidisciplinar) empleo de una amplia variedad de formalismos: ODE, DAE, bond graph, redes de Petri, etc.	Modélica(1997) Nota: con el Modelica se pretendió disponer de un formato estándar, tal que los modelos desarrollados endiferentes dominios puedan ser intercambiados entre herramientas y usuarios

Tabla 2.1 – Evolución del desarrollo de los programas de simulación.

Programas	Año	Observaciones
HydCal		Nacido en el entorno científico es un software utilizado para dibujar basado en simbología normalizada ISO. Permite calcular las pérdidas, caudales, pérdidas de energía en las resistencias hidráulicas.
HydrauSim		Permite diseñar esquemas hidráulicos, comprobar su funcionamiento, pero no permite el cálculo en régimen transitorio
Hysys		Es un sistema de programas para simulación no lineal de circuitos electro – hidráulicos de regulación. Para el desarrollo se emplearon las experiencias obtenidas por Mannesmann Rexroth en materia de técnica hidráulica. Es un software limitado al catálogo de la marca Rexroth. Los accionamientos pueden simularse tanto como cadena abierta como en circuito cerrado de regulación. Es un programa configurado por cuatro subprogramas para aplicaciones específicas: HYVOS, HYDRA, HYSTA HYSEV
Bathfp	1990	Es una herramienta de simulación del dominio del tiempo interactivo. Las herramientas de desarrollo del modelo completo permiten la definición de un nuevo modelo y su

		incorporación dentro de la librería estimula una aproximación estructurada del modelado asegurando una futura aplicabilidad de cada modelo. El paquete puede ser usado en la etapa de diseño para evaluar las posibles configuraciones del circuito y optimizar el tamaño de los componentes y seleccionar los parámetros de control. También se usa para optimizar sistemas existentes y sirve como una herramienta de diagnóstico para identificar la causa del problema. La última versión se estableció el 17 de Octubre del 2001 en la Universidad de Bath.
DSH – plus	1995	Creado por la sociedad Fluidon GmbH y tiene como objetivo principal el cálculo dinámico de sistemas complejos hidráulicos y neumáticos. El modelo matemático utilizado es el de sistema de ecuaciones diferenciales de primeros órdenes lineales y no lineales. El esquema hidráulico se construye conectando elementos y líneas cuya descripción está basada en las leyes de Kirchoff, aplicadas sobre un volumen de aceite considerado como nodo. Utiliza el método de integración de Runge Kutta de 4º orden.
HOPSAN	1991	Es un programa de simulación general, creado especialmente para simular sistemas óleo – hidráulicos y neumáticos. La compleja carga dinámica puede ser estudiada

		así como la propagación de las ondas en grandes líneas. Originalmente se usaron diferentes modelos matemáticos 1) Un sistema de ecuaciones algebraicas no lineales para el modelado en régimen permanente 2) Un sistema de ecuaciones diferenciales ordinarias y algebraicas para la simulación del dominio del tiempo 3) Un sistema de ecuaciones diferenciales lineal izadas para las respuestas de las investigaciones de la frecuencia. Posteriormente se incorporó un algoritmo FFT la que permitió el análisis frecuencia les sobre simulaciones en el dominio del tiempo. El régimen permanente puede obtenerse a partir de la simulación en el dominio del tiempo. El modelo se complementa mediante la incorporación de un sistema nodal.
AME – Sim	1995	Fue desarrollado por IMAGINE (Michel Lebrun 1987) el cual ha sido capaz de capitalizar los conocimientos mecánicos e hidráulicos y en el modelado y diseño de sistemas complejos. IMAGINE tomó parte en la introducción de Bond Graphs en Francia. Previo al desarrollo de AME – Sim, IMAGINE usó para modelar y simular sus estudios de consulta usando ACSL (Advanced Continuous Simulation Language), una potente

		pero no muy interactiva herramienta dedicado al modelado de sistemas dinámicos. En 1995, IMAGINE rompe la barrera entre el usuario de software y sus promotores creando AME – Sim. IMAGINE ha sido desde entonces el que proporciona AME – Sim, y desde entonces, la simulación se apega a las medidas experimentales. Se puede comunicar interactivamente con otros programas (ADAMS, MATLAB, SIMULINK, SIMPACK).
ITI – Sim	2001	La forma de construcción del modelo ha sido adaptada a la descripción más común de cada disciplina de la ingeniería; así se dispone: 1) Diagramas de circuitos para los componentes y sistemas oleohidráulicos 2) Diagramas funcionales para los sistemas mecánicos 3) Diagramas de bloques para los sistemas de control El software se complementa con una aplicación que permite al usuario definir su propio submodelo y otra aplicación que permite simular partes del sistema por una herramienta de simulación externa, comunicándose entre sí a ciertos intervalos de tiempo para intercambiar datos,

		ejemplos ADAMS para la modelización y simulación de sistemas mecánicos y MATLAB/SIMULINK para la modelización y simulación de los elementos de control.
ITI – TOOL		Es un software desarrollado para la optimización de accionamientos oleohidráulicos para grúas y está basado en el ODHAMN (Optimal Design of Hydraulically Actuated Manipulators) que permite: 1) Análisis del régimen permanente durante la elevación y bajada 2) Análisis dinámico en la operación de elevación y bajada 3) Trazado del diagrama de Bode de las funciones de transferencia de los lazos abierto y cerrados para la posición más elevada y las condiciones de carga máxima y mínima 4) Animación del sistema mecánico y del movimiento de la corredera de la válvula "Over – Center"
VisSim	1990	Es un programa usado para el modelado y simulación y en la aplicación del diseño de los sistemas de control. Con la interfase del diagrama de bloques se puede construir, y modificar el modelo. Proporciona soluciones para los sistemas lineales, no lineales, tiempo continuo, tiempo discreto.

		El usuario construye su propio modelo seleccionando un bloque predefinido en la librería y realizando la conexión gráfica de los bloques dentro del diagrama. Cada diagrama de bloque realiza una función matemática o función entrada/salida. Estos bloques pueden representar algoritmos complejos, entradas de variables o salidas de las gráficas similares, esquemas, trazados o archivos de datos. El usuario puede crear un bloque en C, FORTRAN o Pascal y añadirlo a la librería de VisSim. Después el modelo es configurado, se pone en marcha la simulación y los resultados son desplegados.
20 – SIM	1995	Desarrollado en el laboratorio de control de la Universidad de Twente, como sucesor del paquete TUTSIM. Después de un extensivo examen, en Agosto de 1995, la versión 1.0 de CAMAS fue comercialmente liberado bajo el nombre de 20 – SIM. La última versión tiene origen en varios prototipos (MAX, CAMAS y TUTSIM). Usa varios algoritmos de integración avanzada. Permite crear submodelos y combinarlos para formar un modelo complejo. Utiliza el sistema de modelado a través de diagramas de iconos, Bond Graphs y ecuaciones. Permite una interacción

		cerrada con MATLAB y SIMULINK. Comprende cajas de herramientas para la optimización de parámetros, generación de código ANSI – C, linealización, animación gráfica y animación 3D.
OHC – SIM	2001	Desarrollado en Japón con el apoyo de JFPS (The Japan Fluid Power Systems Society ó mejor conocido como The Japan Hydraulics and Pneumatics Society) y mejorado en Sakurai (1999 – 2000). Es dedicado al diseño y mejoramiento de circuitos óleo – hidráulicos en la base de los resultados simulados y el análisis de sus características dinámicas. La función "User – Customized" ejecuta la simulación para la cual era necesario conocer de Bond Graphs, pero que en las próximas versiones se proporcionará un entorno en donde un nuevo componente óleo – hidráulico pueda ser registrado sin conocimiento de Bond Graphs.

Tabla 2.2 - Programas disponibles o comerciales

Temiendo en cuenta los aspectos antes mencionados de los programas citados, se ha considerado utilizar 20-SIM para simular el comportamiento dinámico de sistemas, el cual es un programa que permite separar los diferentes dominios tecnológicos en submodelos independientes. Las razones que indujeron a utilizar este programa fueron las siguientes:

En LABSON hay licencia de uso del programa

20-SIM

20-SIM está basado en Bond Graphs, el cual es una herramienta que proporciona un modelo común para los diferentes campos de la técnica mediante la representación gráfica de sus componentes y sus interrelaciones y en LABSON existe experiencia desde inicio de los 90.

Fig. 2.1. Interfaz de usuario de 20-sim.

Con las técnicas de Bond Graphs es muy sencillo añadir complejidad a un modelo mediante la incorporación de nuevos elementos o subsistemas. Una de las virtudes de Bond Graphs es su versatilidad, porque permite modificar la estructura del sistema de forma rápida y certera, el cual es muy útil desde el punto de vista de diseño. Partiendo de esto, es posible tratar un sistema sencillo y refinarlo progresivamente hasta transformarlo en un modelo lo suficientemente exacto del sistema considerado.

20-SIM cuenta con una librería múltiple con un gran sistema de submodelos de dominio orientado.

Ejemplo de algún software que utilizamos en la facultad de ingeniería eléctrica para la simulación de algunas actividades así como procesos que desarrollamos durante los cursos tomados e impartidos por nuestros profesores.

Packet Tracer

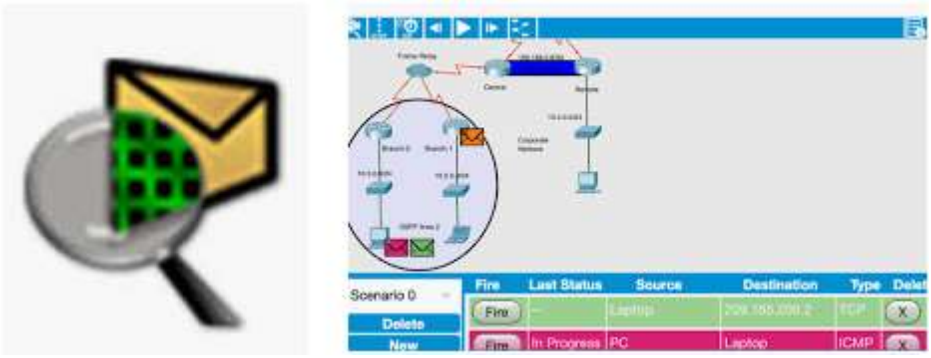


Fig. 2.2. Interfaz de usuario de packet Tracer.

Información general

Tipo de programa	software privativo
Desarrollador	Cisco Systems
Última versión estable	7.2.1
Género	Simulación de estructura de red
Sistema operativo	Linux, Android 4.1+, iOS 8+ , Microsoft Windows and MacOS
Licencia	Propietario
Estado actual	Activo
Idiomas	Inglés, ruso, portugués, alemán, español y francés
En español	✓ **Sí**

Tabla 2.3 - Información general Packet Tracer

Packet Treacer de cisco es un programa de simulación de la estructura de redes que permite a los estudiantes experimentar con el comportamiento de la red y resolver preguntas del tipo

Características de packet Tracer

La versión actual soporta un conjunto de Protocolos de capa de aplicación simulados, al igual que enrutamiento básico con RIP, OSPF, y EIGRP. Aunque Packet Tracer provee una simulación de redes funcionales, utiliza solo un pequeño número de características encontradas en el hardware real corriendo una versión actual del Cisco IOS. Packet Tracer no es adecuado para redes en producción.

En este programa se crea la topología física de la red simplemente arrastrando los dispositivos a la pantalla. Luego haciendo clic sobre ellos se puede ingresar a sus consolas de configuración. Allí están soportados todos los comandos del Cisco IOS e incluso funciona el "tab completion". Una vez completada la configuración física y lógica de la red, también se pueden hacer simulaciones de conectividad (pings, traceroutes) todo ello desde las mismas consolas incluidas.

Una de las grandes ventajas de utilizar este programa es que permite "ver" (opción "Simulation") cómo deambulan los paquetes por los diferentes equipos (switchs, routers, PCs), además de poder analizar de forma rápida el contenido de cada uno de ellos en las diferentes "capas" y "datos".

PuTTY

Diálogo principal de configuración PuTTY 0.58 sobre FVWM

Fig. 2.3. Interfaz de usuario de PuTTY.

Información general

Aplicación informática	Emulador de terminal
Tipo de programa	cliente SSH terminal client software libre y de código abierto
Desarrollador	Simon Tatham
Última versión estable	0.73
Género	Emulador de terminal
Programado en	C
Sistema operativo	Microsoft Windows, Linux/otros Unix
Licencia	Licencia MIT
En español	✗ **No**

Tabla 2.4 - Información general PuTTY

PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Disponible originalmente sólo para Windows, ahora también está disponible en varias plataformas Unix, y se está desarrollando la versión para Mac OS clásico y Mac OS X. Otra gente ha contribuido con versiones no oficiales para otras plataformas, tales como Symbian para teléfonos móviles. Es software beta escrito y mantenido principalmente por Simon Tatham, open source y licenciado bajo la Licencia MIT.

Características

Algunas características de PuTTY son:

El almacenamiento de hosts y preferencias para uso posterior.

Control sobre la clave de cifrado SSH y la versión de protocolo.

Clientes de línea de comandos SCP y SFTP, llamados "pscp" y "psftp" respectivamente.

Control sobre el re direccionamiento de puertos con SSH, incluyendo manejo empotrado de reenvío X11.

Completos emuladores de terminal xterm, VT102, y ECMA-48.

Soporte IPv6.

Soporte 3DES, AES, RC4, Blowfish, DES.

Soporte de autentificación de clave pública.

Soporte para conexiones de puerto serie local.

El nombre **PuTTY** proviene de las siglas **Pu**: Port unique **TTY**: Teletipo (por sus siglas en inglés). Su traducción al castellano sería: Teletipo de puerto único.

Aplicaciones

Las funciones principales están realizadas por los mismos ficheros PuTTY:

PuTTY - los clientes Telnet y SSH

PSCP - un cliente SCP, i.e. copia de ficheros segura por línea de comandos

PSFTP - un cliente SFTP, i.e. sesiones de transferencia de ficheros generales como en FTP

PuTTYtel - un cliente de solo Telnet

Plink - una interfaz de línea de comandos al PuTTY back ends

Pageant - un agente de autenticación SSH para PuTTY, PSCP y Plink

PuTTYgen - una utilidad de generación de claves RSA y DSA.

pterm - un emulador de terminal X.

Capítulo 3. Modelado de sistemas discretos

3.1 ¿Que es el control digital?

Prácticamente, todos los sistemas de control que son implementados hoy están basados en control por computadora. Es por lo tanto importante comprender bien sistemas controlados por computadora.

Un sistema controlado por computadora puede ser descrito esquemáticamente en la figura 3.1.

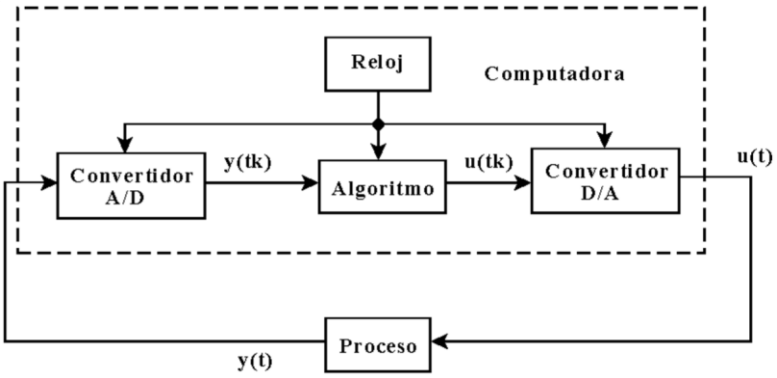


Figura 3.1 Sistema de control por computadora.

El sistema controlado por computadora contiene ambas señales de tiempo continuo y muestreadas o señales de tiempo discreto, tales sistemas han sido tradicionalmente llamados sistemas de datos muestreados.

La mezcla de diferentes tipos de señales algunas veces causa dificultades. Si las señales de interés solamente son en tiempo discreto, tales sistemas serán llamados sistemas de tiempo discreto. Sistemas de tiempo discreto tratan con secuencias de números, as un camino natural para representar estos sistemas es utilizar ecuaciones de diferencias.

3.2 Esbozo histórico del control por computadora

La idea de utilizar computadoras digitales como componentes en sistemas de control surgió alrededor de 1950. Aplicaciones en misiles y control de vuelos fueron investigados, pero al ser las computadoras demasiado grandes y consumidoras de potencia, estas no resultaban tener buen potencial. El desarrollo en la introducción de las computadoras se distingue seis periodos:

*__Periodo pionero, aproximadamente se da en 1955__. Los sistemas de cómputo eran lentos, costosos y no realizables.

*__Periodo de control digital directo, aproximadamente se da en 1962__. En las industrias químicas imperiales (ICI), una instrumentación analógica completa para control de procesos fue reemplazada por una computadora. Esta computadora monitoreaba 224 variables y controlaba 129 válvulas directamente. Esto fue el inicio de una nueva era en control de procesos: tecnología analógica era simplemente reemplazada por tecnología digital. El nombre de control digital directo (DDC) fue acuñado para enfatizar que la computadora controlaba el proceso directamente, lo cuál se muestra en la figura 3.2.

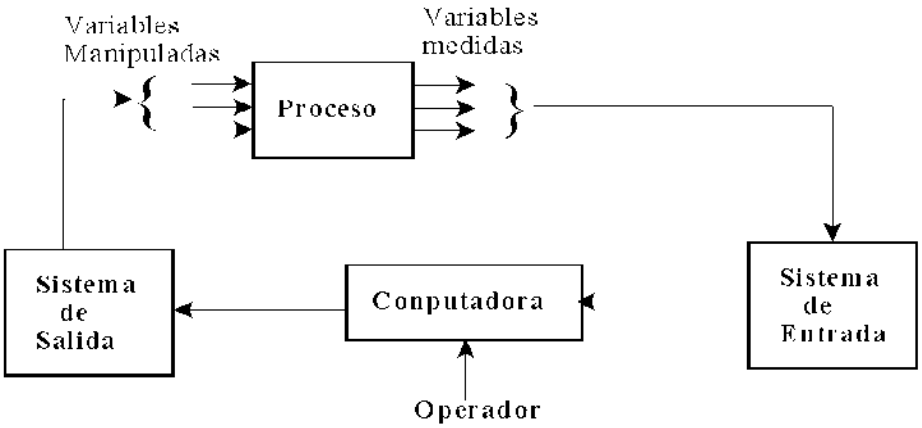


Figura 3.2 Control digital directo de un proceso.

*__Periodo de minicomputadora.__ Las computadoras llegaron a ser más pequeñas, más rápidas y económicas. El término minicomputadora fue acuñado para las nuevas computadoras que emergieron. Fue posible el diseño eficiente de sistemas de control de procesos utilizando minicomputadoras.

*__Periodo de microcomputadora__. El desarrollo de la microcomputadora en 1972 continúo con los avances de la tecnóloga de la microelectrónica y en 1990 los microprocesadores ya eran disponibles. Esto ha tenido un impacto profundo en el uso de control por computadora.

*__Lógica secuencial y control__. Sistemas automáticos industriales tradicionalmente han tenido dos componentes, controladores y lógica de relevadores. El llamado controlador lógico programable (PLC) emergió en el inicio de 1970s como reemplazo de los relevadores.

*__Control Distribuido__. El microprocesador también ha tenido un profundo impacto en el camino de las computadoras fueron aplicadas para controlar plantas de producción enteras. Llego a ser económicamente realizable desarrollar sistemas consistentes de algunas microcomputadoras interactuando a una estación de trabajo. Tales sistemas generalmente consisten de estaciones de procesos, controlando el proceso, estaciones del operador y varias estaciones auxiliares. Un sistema de control distribuido se ilustra en la figura 3.3.

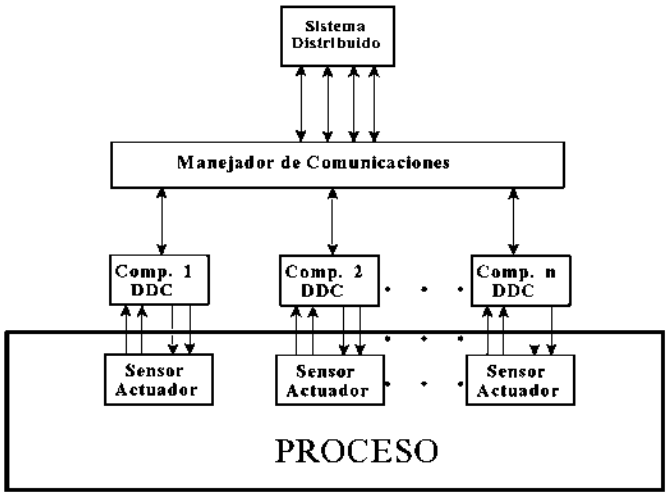


Figura 3.3. Sistema de control distribuido.

3.3 Por qué una Teoría de Control por Computadora

Utilizando computadoras para implementar controladores tiene ventajas sustanciales. Muchas de las dificultades con la implementación analógica pueden ser eliminadas. Por ejemplo, no existen problemas en la exactitud de los componentes, es más fácil tener cálculos sofisticados en la ley de control y es fácil incluir funciones lógicas y no lineales.

3.4. Características propias de los sistemas muestreados

Un diagrama esquemático de un sistema controlado por computadora es mostrado en la figura 3.4.

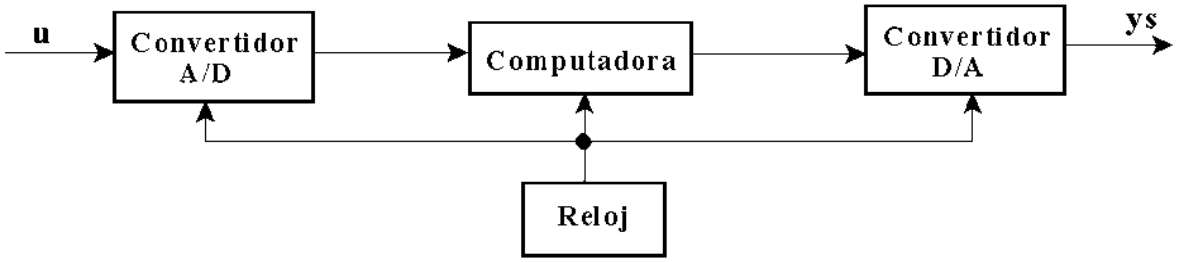


Figura 3.4 Sistema de control por computadora.

Este sistema contiene esencialmente de cinco partes: el proceso, los convertidores

A/D y D/A, el algoritmo de control y el reloj.

Su operación es controlada por el reloj. Los instantes cuando las señales medidas son convertidas a la forma digital son llamados instantes de muestreo. El tiempo entre muestreos sucesivos es llamado el periodo de muestreo.

3.4.1 Dependencia del Tiempo

La presencia del reloj genera que los sistemas controlados por computadora sean variantes en el tiempo. Tales sistemas pueden exhibir un comportamiento diferente a sistemas lineales invariantes en el tiempo, tal como se muestra en la figura 3.5.

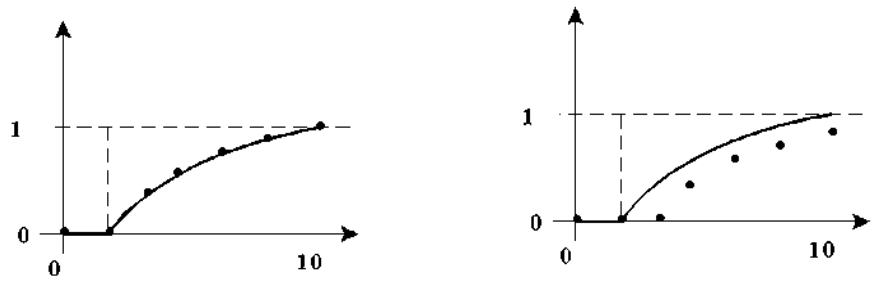


Figura 3.5. Diagramas de tiempo de un sistema.

El fenómeno ilustrado depende en el hecho que el sistema es controlado por un reloj. La respuesta del sistema a un estímulo externo entonces dependerá en como el evento externo es sincronizado con el reloj interno del sistema de cómputo.

Debido al muestreo es frecuentemente periódico, sistemas controlados por computadoras frecuentemente resultarán en sistemas de lazo cerrado que son sistemas periódicos lineales.

Una propiedad natural de sistemas lineales invariantes en el tiempo estable es que la respuesta de estado estacionario a excitaciones senoidales es senoidal con la frecuencia de la señal de excitación. Sera mostrado que sistemas controlados por computadora se comportan en una forma más complicada debido a que el muestreo creara señales con frecuencias nuevas.

3.4.2 Procesos inherentemente muestreados

Modelos muestreados son descripciones naturales para muchos fenómenos. La teoría de sistemas de datos muestreados, por lo tanto, tiene muchas aplicaciones diferentes al campo de control por computadora.

Muestreo debido al sistema de medición:

i) Radar, rota en un rango y dirección considerando las revoluciones.

ii) Instrumentos analíticos, sistemas que no son medibles en línea.

iii) Sistemas económicos.

Muestreo debido a la operación pulsada. Muchos sistemas son inherentemente muestreados debido a la información pulsada. Circuitos electrónicos son un ejemplo clásico:

- Control por tiristor
- Sistemas biológicos
- Encendido de un motor de combustión

3.4.3 Señales en Tiempo Continuo frente a Señales en Tiempo Discreto

Las señales se pueden clasificar en las siguientes categorías diferentes dependiendo de las características de la variable independiente y los valores que esta puede tomar.

* Señales en tiempo continuo o señales analógicas, están definidas para todos los valores del tiempo y pueden tomar cualquier valor en el intervalo continuo (*a;b*).

*Señales en tiempo discreto están definidas solo ciertos valores de tiempo. El valor de una señal en tiempo continuo o discreto, puede ser continuo o discreto. Si una señal toma todos los valores posibles en un intervalo tanto nito como in nito se dice que es continua. Por el contrario, si toma valores de un conjunto nito de valores se dice que es discreta.

Una señal en tiempo discreto, que toma valores en un conjunto discreto se denomina señal digital.

Para que una señal puede ser procesada digitalmente ha de ser de tiempo discreto y tomar valores discretos. Si la señal a procesar es analógica se convierte a digital muestreándola en el tiempo y obteniendo por tanto una señal en tiempo discreto y posteriormente cuantificando sus valores en un conjunto discreto.

3.4.3.1. Señales Senosoidales en tiempo continuo

Una señal en tiempo continuo se describe por

$$x_a(t) = A\cos(\Omega t + \theta), \infty \leq t \leq \infty$$

Donde A es la amplitud, Ω es la frecuencia en rad/seg y θ es la fase.

Se sabe que

$$\Omega = 2F$$

$$x_a(t) = A\cos(2\pi Ft + \theta)$$

La señal senosoidal está caracterizada por las siguientes propiedades:

1. Para todo valor fijo de la frecuencia F, $x_a(t)$ es periódica

$$x_a(t + T_p) = x_a(t)$$

Donde $T_p = 1/F$ es el periodo fundamental de la señal senosoidal.

2. Las señales en tiempo continuo con frecuencias diferentes, son diferentes.

3. El aumento en la frecuencia F resulta en un aumento en la tasa de oscilación de la señal.

3.4.3.2 Señales Senosoidales en tiempo discreto

Una señal senosoidal en tiempo discreto puede expresarse como

$$x(n) = A\cos(wn + \theta),\ \infty \leq n \leq \infty$$

donde n es una variable entera, denominada número de muestra, A es la amplitud de la senosoidal, w es la frecuencia en rad/muestra y es la fase en rad.

Utilizando $w = 2\pi f$ se tiene,

$$x(n) = A\cos(2\pi fn + \theta)$$

La frecuencia f tiene dimensiones de ciclos por muestra.

Las senoidales en tiempo discreto están caracterizadas por las propiedades siguientes:

1. Una senoidal en tiempo discreto es periódica, s lo si su frecuencia f es un número racional.

Una señal en tiempo discreto $x(n)$ es periódica con periodo N ($N \geq 0$) si y solo si

$x(n + N) = x(n)$ para todo n

El valor más pequeño de N para el que se cumple la anterior ecuación se denomina periodo fundamental.

$$f_0 = \frac{k}{N}$$

Así , una señal senoidal en tiempo discreto es periódica solo si su frecuencia f_0 se puede expresar como el cociente de dos enteros (si f_0 es racional).

2. Las senoidales en tiempo discreto cuyas frecuencias están separadas por un múltiplo entero de 2π, son idénticas.

$$\cos[(w_0 + 2\pi)n + \theta] = \cos[w_0 n + 2\pi n + \theta] = \cos[w_0 n + \theta]$$

3. La mayor tasa de oscilación en una senoidal en tiempo discreto se alcanza

Cuando $f = \frac{1}{2}$

3.4.4. Conversión Analógico-Digital y Digital-Analógico

La mayor a de las señales de interés práctico, señales de voz, biológicas, sísmicas, radar, etc. son analógicas. Para procesar señales analógicas por medios digitales es necesario convertirlas a formato digital, este procedimiento se denomina conversión analógico-digital (A/D). La conversión A/D se da en un proceso de tres pasos.

1. Muestreo.
2. Cuantificación.
3. Codificación.

El muestreo periódico uniforme. *T* es el periodo de muestreo.

$F_s = \frac{1}{T}$ se denomina velocidad de muestreo o frecuencia de muetreo.

$$t = nT = \frac{n}{F_s}$$

Existe una relación entre la variable de frecuencia F de las señales analógicas y la variable de frecuencia f de las señales en tiempo discreto.

Así ,

$$x_a(t) = A\cos(2\pi Ft + \theta)$$

muestreándola a $F_s = \frac{1}{T}$

$$x_a(nT) = x(n) = A\cos(2\pi FnT + \theta)$$

$$= A\cos\left(\frac{2\pi nF}{F_s} + \theta\right)$$

Se sabe que

$$x(n) = A\cos(2\pi fn + \theta)$$

comparando se tiene

$$f = \frac{F}{F_s}$$

Así, f también se le llama frecuencia normalizada o relativa. El rango de frecuencia para F o para senosoidales en tiempo continuo es

$$-\infty \leq F \leq \infty$$

3.5 Algunas Señales Elementales en Tiempo Discreto

En el estudio de sistemas y señales discretas en el tiempo existen varias señales básicas que aparecen con frecuencia y juegan un papel importante.

- Función Impulso Unitario (Figura 3.6)

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

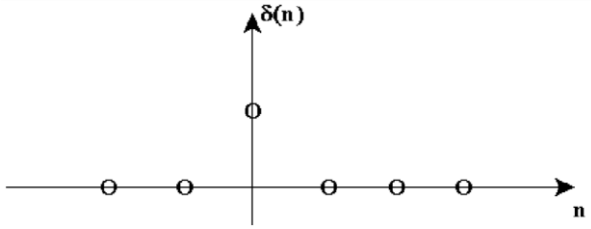


Figura 3.6. Impulso unitario.

- Función Paso Unitario (Escalón Unitario) (Función 3.7)

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

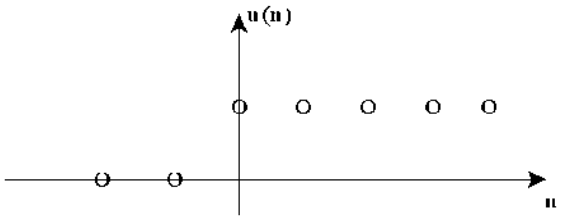


Figura 3.7. Escalón unitario.

- Función Rampa Unitaria (Figura 3.8)

$$r(n) = \begin{cases} n+1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

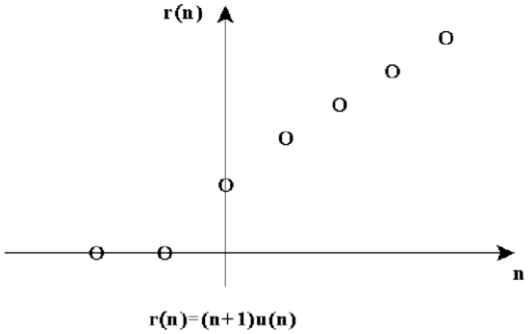


Figura 3.8. Rampa Unitaria.

3.6. Realización de Ecuaciones de Diferencias

Un sistema se puede ver como cualquier proceso que produce una transformación de señales. Todo sistema debe tener al menos una entrada *x* y una salida *y*. La señal de salida está relacionada con la entrada mediante una relación de transformación $y = f(x)$.

Los diagramas de bloques nos permiten representar las operaciones básicas entre sistemas esto es, su interconexión, la cual puede ser de tres tipos:

-Interconexión en serie o en cascada (Figura 3.9).

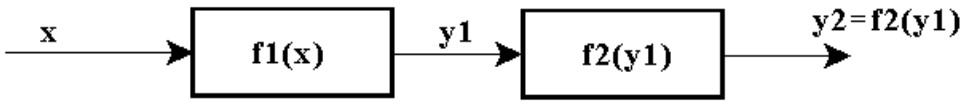


Figura 3.9. Conexión en cascada.

-Paralelo (Figura 3.10).

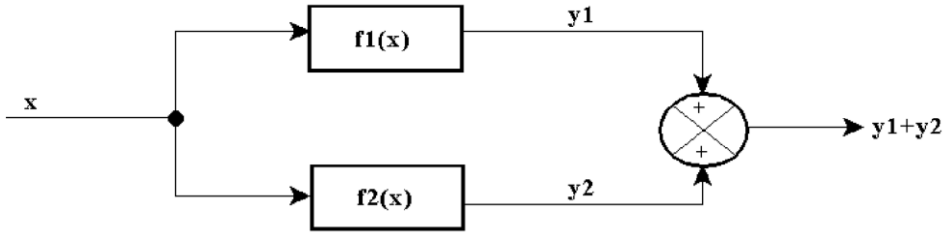


Figura 3.10. Conexión en paralelo.

-Retroalimentación (Figura 3.11).

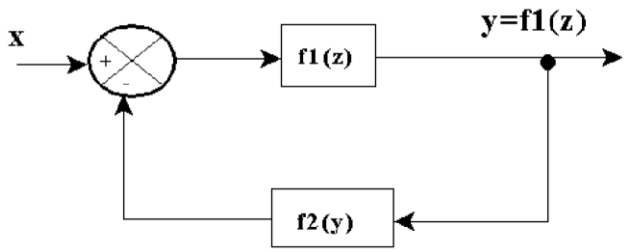


Figura 3.11. Sistema retroalimentado.

La forma general de las ecuaciones en diferencias pueden ser escritas por,

$$y(k) = \sum_{i=0}^{n} a_i x(k-i) - \sum_{i=1}^{n} b_i y(k-i)$$

en forma desarrollada

$y\ (k) = a_0x(k)+a_1x(k - 1)+,,,+a_nx(k - n) - b_1y\ (k - 1) - b_2y\ (k - 2) -,,,,- b_ny\ (k - n)$

La realización en diagrama de bloques denominada **Forma Directa I** como se muestra en la figura 3.12.

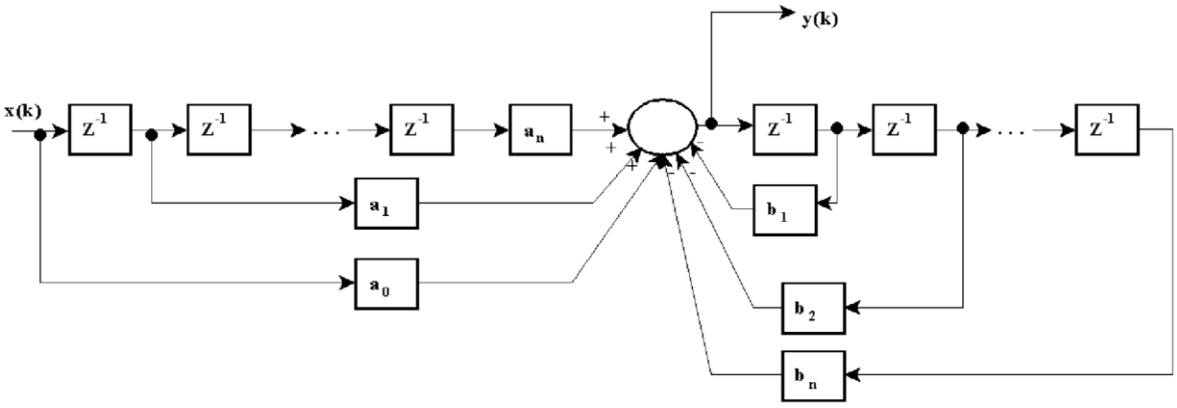


Figura 3.12. Forma directa I.

Ejemplo

Dibujar la forma directa I de la siguiente ecuación en diferencias

$$y\ (k) = a_0x(k) + a_1x(k - 1) - b_1y\ (k - 1) - b_2y\ (k - 2)$$

en la figura 3.13 se ilustra la forma directa I de este ejemplo.

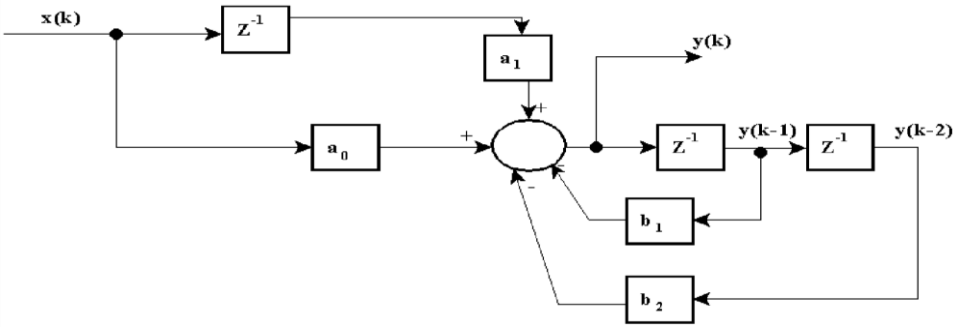


Figura 3.13. Forma directa I.

3.7. Solución de Ecuaciones en Diferencias

Recordar que las ecuaciones en diferencias determinan sistemas en tiempo discreto y las ecuaciones diferenciales determinan sistemas en tiempo continuo, tal como se muestra en la figura 3.14.

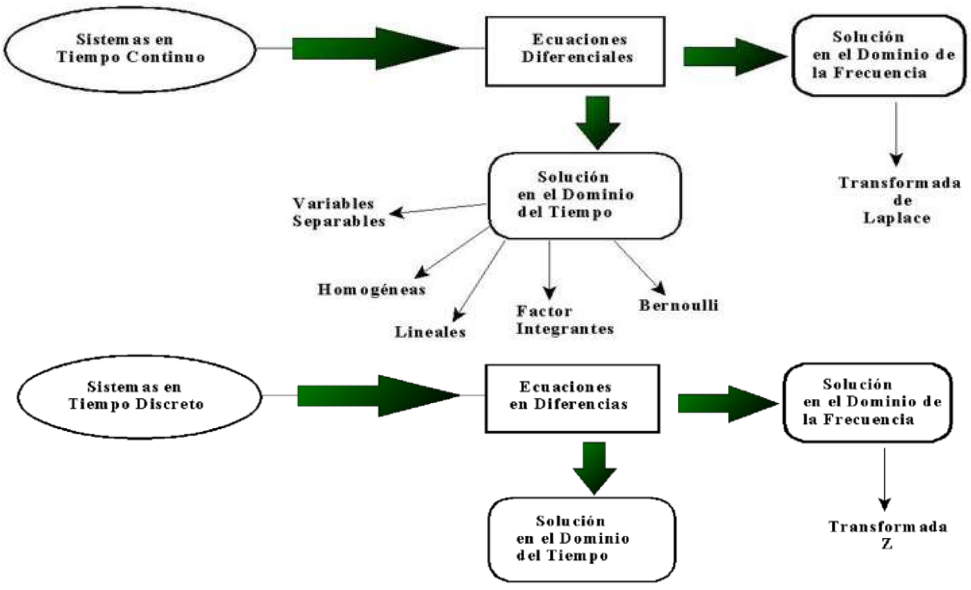


Figura 3.14. Forma directa I.

Una ecuación en diferencias de primer orden está definida por

$$y(k) + b_1 y(k-1) = a_0 x(k) + a_1 x(k-1) \tag{3.1}$$

cuya solución completa es de la forma

$$y(k) = y_h(k) + y_p(k) \tag{3.2}$$

donde $y_h(k)$ es la solución homogénea, la cuál es la solución sin excitación (sin entrada) y $y_p(k)$ es la solución particular denominada respuesta forzada.

Ejemplo.

Resolver la siguiente ecuación en diferencias

$$y(k) - 0{;}9y(k-1) = x(k)$$

donde

$$x(k) = (0{,}4)^k u(k)$$

$$y(-1) = 0$$

La solución homogénea tendrá la forma

$$y_h(k) = AZ^k$$

donde z es la raíz de la ecuación característica.

Determinando la ecuación homogénea, que es la ecuación correspondiente sin entrada

$$y(k) - 0.9y(k-1) = 0$$

sustituyendo $y(k)$ por AZ^k se tiene

$$AZ^k - 0.9AZ^{k-1} = 0$$

Factorizando

$$AZ^{k-1}(Z - 0.9) = 0$$

entonces la ecuación característica de este sistema es

$$Z - 0.9 = 0$$

cuya solución es

$$Z = 0.9$$

La solución homogénea es

$$y_h(k) = A(0.9)^k$$

Para la solución particular se debe utilizar la siguiente tabla.

Entrada	Solución Particular
ak	Ba^k
$sen\ bk$	$B_1 sen\ bk + B_2 \cos bk$
$cos\ bk$	$B_1 sen\ bk + B_2 \cos bk$
$a^k sen\ bk$	$a^k [B_1 senbk + B_2 \cos bk]$
$a^k \cos bk$	$a^k [B_1 senbk + B_2 \cos bk]$
a	A
$a\ k$	$B_1 k + B_2$
$nkan$	$an\ B1nk + B2nk\ 1 + ,,, + Bn+1$

Tabla 3.1 Solución particular de ecuaciones.

La solución particular que se propone, para este ejemplo, con entrada $x(k) = (0,4)^k\, u\,(k)$ es

$$y_p\,(k) = B\ (0.4)^k$$

Sustituyendo esta solución en la ecuación de diferencias $y\ (k) - 0.9y\ (k - 1) = x(k)$ se tiene

$$B\ (0.4)^k - 0.9B\ (0.4)^{k-1} = (0.4)^k$$

re-escribiendo

$$B\,(0{,}4)^k - 0{,}9B\frac{(0{,}4)^k}{(0{,}4)} = (0{,}4)^k$$

se puede eliminar el término $(0.4)^k$ reduciéndose a

$$B\left(1 - \frac{0{,}9}{0{,}4}\right) = 1$$

resolviendo

$$B = \frac{1}{1 - \frac{0{,}9}{0{,}4}} = -0{,}8$$

entonces la solución particular es

$$y_p\,(k) = -\,0.8(0.4)^k$$

La solución completa de cualquier ecuación de diferencias es $y\,(k) = y_h\,(k) + y_p\,(k)$ para el ejemplo se tiene

$$y\,(k) = A(0.9)^k - 0.8(0.4)^k \text{ para } k \geq 0$$

3.8 Solución de Ecuaciones en Diferencias Mediante Transformada Z

Para una ecuación de orden n,

$$y\,(k+n) + a_{n-1}y\,(k+n-1) + \ldots + a_1 y\,(k+1) + a_0 y\,(k) = b_m r\,(k+m) + \ldots + b_1 r\,(k+1) + b_0 r\,(k)$$

re-escribiendo

$y(k) + a_{n-1}y(k-1) + \ldots + a_1 y(k-n+1) + a_0 y(k-n) = b_m r(k+m-n) + \ldots$

$+ b_1 r(k-n+1) + b_0 r(k-n)$

Aplicando Transformada Z, si hay condiciones iniciales, utilizando

$Z[f(k-n)] = z^{-n}F(z) + z^{-n+1}f(-1) + \ldots + z^{-n}f(z-n) + z^{-1}f(1-n) + f(-n$

entonces $Y(z) + a_{n-1}[Z^{-1}Y(z) + y(-1) + \ldots + a_1[Z^{-n+1}Y(z) + z^{-n+2}y(-1) + \ldots + y(-n+1)] +$

$a_0[z^{-n}Y(z) + z^{-n+1}y(-1) + \ldots + y(-n) =$

$b_m[Z^{-n+m}R(z) + z^{-n+m+1}r(-1) + \ldots + r(-n+1) +$

$\ldots + b_0[Z^{-n}R(z) + z^{-n+1}r(-1) + \ldots + r(-n)$

Factorizando y despejando

$$Y(z) = \frac{b_m z^n + b_{n-1}z^{m-1} + \cdots + b_1 z + b_0}{z^n + a_{n-1}z^{n-1} + \cdots + a_1 z + a_0}$$

$$\frac{\left(\begin{array}{c}\text{Polinomio en } z \text{ de orden } n \text{ o menor con} \\ \text{coeficientes dependientes de las condiciones iniciales}\end{array}\right)}{z^n + a_{n-1}z^{n-1} + \cdots + a_1 z + a_0}$$

La componente de estado cero es

$$\frac{b_m z^n + b_{n\ \ 1} z^{m-1} + \cdots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0}$$

y la componente de entrada cero es

$$\frac{\left(\begin{array}{c}\text{Polinomio en } z \text{ de orden } n \text{ o menor con} \\ \\ \text{coeficientes dependientes de las condiciones iniciales}\end{array}\right)}{z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0}$$

Observase que la componente de entrada cero de la respuesta es cero, si todas las condiciones iniciales son cero, esto es $Y(z) = T(z)R(z)$, donde la función de transferencia $T(z)$ es

$$T(z) = \frac{b_m z^n + b_{n-1} z^{m\ \ 1} + \cdots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0}$$

3.9 Análisis de Sistemas de Control Digital

Los sistemas de control en tiempo discreto pueden operar en parte en tiempo discreto y en parte en tiempo continuo. De esta manera, en dicho sistema de control algunas señales aparecen como funciones en tiempo discreto y otras funciones en tiempo continuo.

Considerar un muestreado mediante impulsos: La salida muestreada mediante impulsos es una secuencia de impulsos, con la magnitud de cada impulso igual al valor de $x(t)$ en el instante de tiempo correspondiente. Un Muestreador se muestra en la figura 3.15.

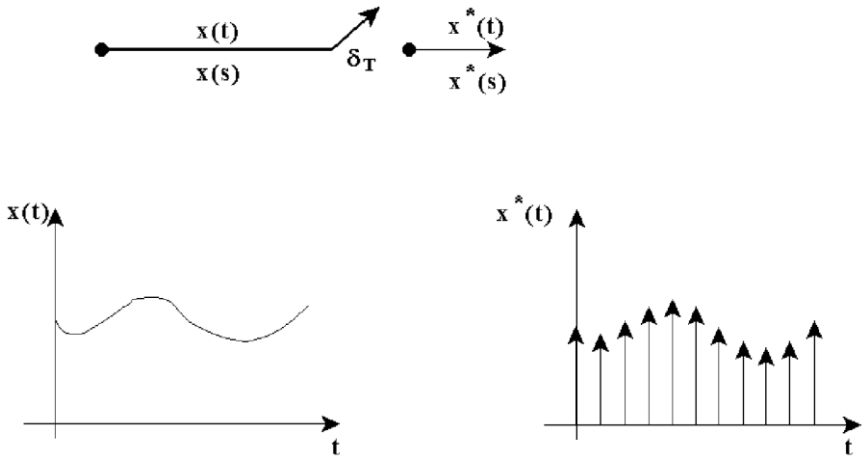


Figura 3.15. Muestreador.

Se utiliza la notación *x* (*t*) para representar la salida muestreada mediante impulsos

$$x^*(t) = \sum_{k=0}^{\infty} x(kT)\,\delta(t-kT)$$

$$x^*(t) = x(0)\,\delta(0) + x(T)\,\delta(t-T) + \cdots + x(kT)\,\delta(t-kT) +$$

La transformada de Laplace de la anterior expresión da

X *(s) = L[x* (t)] = x(0)L[(t)] + x(T)L[(t - T)] +… + x(2T)L[S(t – 2T)]+…

X* (s) = x(0) + x(T)e -Ts + x(2T)e -2Ts +…

$$X^*(s) = \sum_{k=0}^{\infty} x(kT)\,e^{\ kTs}$$

Definiendo

$$z = e^{Ts};\ s = \frac{1}{T}Ln(z)$$

entonces

$$X^*(s)|_{s=\frac{1}{T}Ln(z)} = \sum_{k=0}^{\infty} x(kT)\, z^{-k} = X(z)$$

Por lo tanto,

$$X^*(s)|_{s=\frac{1}{T}Ln(z)} = X^*\left(\frac{1}{T}Ln(z)\right) = X(z) = \sum_{k=0}^{\infty} x(kT)\, z^{\ k}$$

Ahora, un retenedor de datos es un proceso de generación de una señal en tiempo continuo $h(t)$ a partir de una secuencia en tiempo discreto $x(kT)$.

Un circuito retenedor convierte la señal muestreada en una señal en tiempo continuo, que reproduce aproximadamente la señal aplicada al muestreado. La señal $h(t)$ durante el intervalo $kT \leq t \leq (k + 1)T$ se puede aproximar mediante un polinomio en T como sigue

$h(k\ T + T\ ) = anT^n + an\text{-}\ 1T^{n-1}\ n + \ldots + a1\ \mathrm{T} + a0$

donde $0 \leq T \leq T$.

Observase que la señal $h(kT)$ debe ser igual a $x(kT)$, $h(kT) = x(kT)$, por lo

tanto,

$h(k\ T + T\ ) = anT^n + an\text{-}\ 1T^{n-1}\ n + \ldots + a1\ \mathrm{T} + x(kT)$

Si el circuito retenedor es un polinomio de n-esimo orden, se conoce como retenedor de n-esimo orden. Si $n = 1$ se denomina retenedor de primer orden.

El retenedor de datos más sencillo se obtiene cuando $n = 0$, asi , $h(kT +) = x(kT)$ donde 0 $\leq T \leq$y $k = 0, 1, 2, ...,$, al cual se denomina retenedor de orden cero, el cual se muestra 3.16.

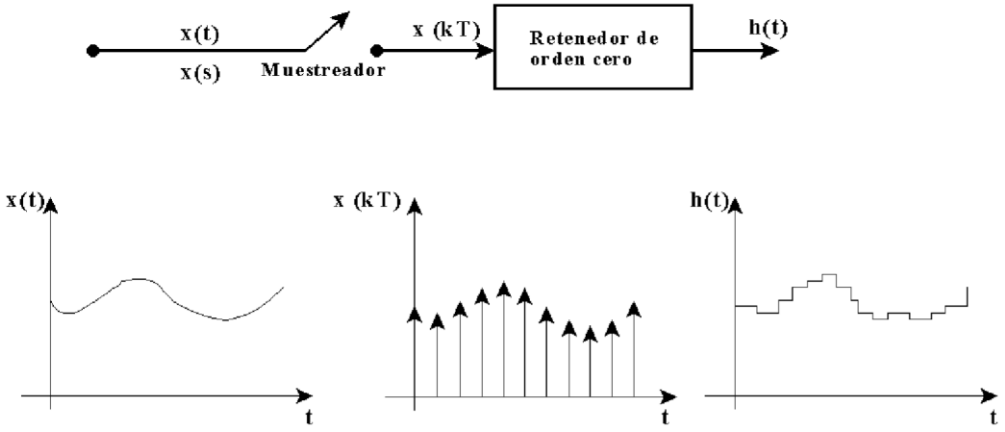


Figura 3.16. Forma directa I.

El modelo matemático de un muestreador retenedor de orden cero se determina a partir de la siguiente gura.

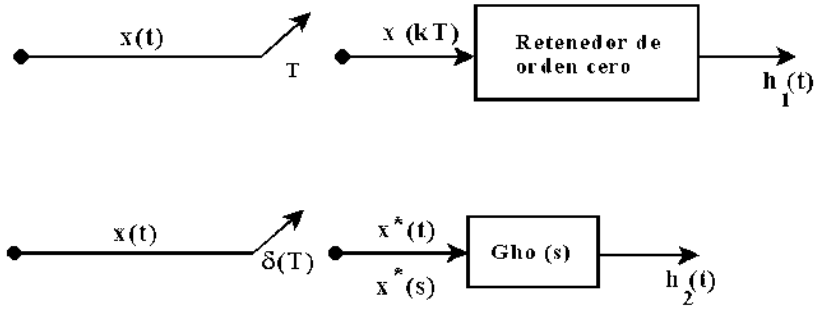


Figura 3.17. Modelo del muestreador-retenedor.

$h_1(t) = x(0)[1(t) - 1(t - T)] + x(T)[1(t - T) - 1(t-2T)] + x(2T)[1(t-2T) - 1(t-3T)] + \ldots$

$$h_1(t) = \sum_{k=0}^{\infty} x(kT)\left[1(t - kT) - 1(t - (k+1)T)\right]$$

Como la transformada de Laplace de

$$\mathcal{L}\left[1(t - kT)\right] = \frac{e^{-kTs}}{s}$$

entonces

$$\begin{aligned} \mathcal{L}\left[h_1(t)\right] &= H_1(s) = \sum_{k=0}^{\infty} x(kT)\,\frac{e^{\ kTs} - e^{\ (k+1)Ts}}{s} \\ &= \frac{1 - e^{\ Ts}}{s} \sum_{k=0}^{\infty} x(kT)\, e^{-kTs} \end{aligned}$$

La salida de este modelo debe ser la misma que la del retenedor de orden cero real, $L[h_2(t)] = H_2(s) = H_1(s)$ entonces

$$H_2(s) = \frac{1 - e^{\ Ts}}{s} \sum_{k=0}^{\infty} x(kT)\, e^{-kTs}$$

de la figura

$H_2(s) = Gho(s)X^*(s)$

entonces

$$H_2(s) = \frac{1 - e^{-Ts}}{s} X^*(s)$$

as la función de transferencia del retenedor de orden cero está dada por

$$Gho\,(s) = \frac{1 - e^{-Ts}}{s}$$

Relación entre *s* y *z*

$$Z = e^{TS}$$

Como normalmente los polos son complejos $s = {}^{\delta}+ Jw$

$$z = e^{(\sigma + J\omega)T} = e^{\sigma T} e^{J\omega T} = e^{\sigma T} \angle \omega T$$

$$\begin{aligned} \mathrm{Re}\,[z] &= e^{\sigma T} \cos \omega T \\ \mathrm{Im}\,[z] &= e^{\sigma T} \sin \omega T \end{aligned}$$

El sistema sera estable si todos sus polos son estables o se encuentran en el semiplano izquierdo de *s*, es decir, < 0 entonces

$$|z|\, e^{\sigma T} \text{ con } \sigma < 0$$

$$|z| < 1$$

El límite de estabilidad es [z] = 1, lo cual se muestra en la figura 3.18

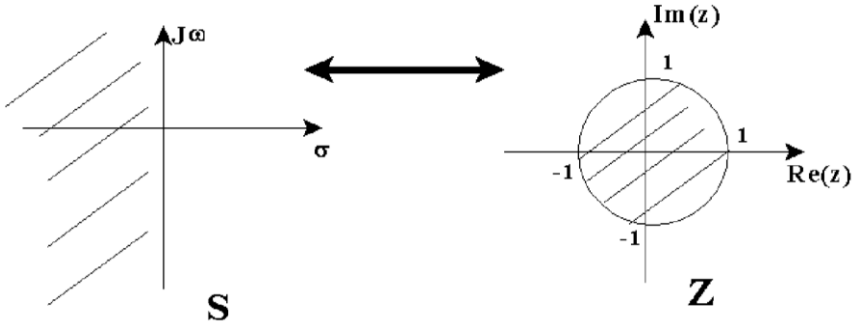


Figura 3.18. Relación entre S y Z.

Cuando una señal analógica $f(t)$ se muestrea para formar la sucesión $f(k)$, existe una relación directa entre la transformada de Laplace $F(s)$ de la señal analógica y la transformada Z $F(z)$ de la sucesión.

En consecuencia, la determinación de la transformada Z de la sucesión correspondiente comprende la separación de las operaciones de tiempo de retardo de la parte racional de la transformada de Laplace, después se sustituye z^{-1} por cada unidad de tiempo de retardo.

Suponer que el muestreador mediante impulsos es seguido por un elemento lineal en tiempo continuo y la función de transferencia es $G(s)$, como se ilustra en la figura 3.19.

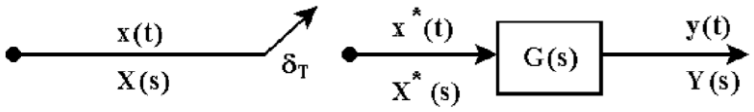


Figura 3.19. Muestreador con transformadas asterisco.

$$Y(s) = G(s)X^*(s)$$

$$Y(s) = [G(s)X^*(s)]^*$$

$$Y(s) = [G(s)]^* X^*(s) = G^*(s)X^*(s)$$

Debido a que la transformada Z puede entenderse como la transformada de Laplace asterisco con $z = e^{Ts}$, la transformada Z se puede considerar una notación corta para la transformada de Laplace asterisco $Y(z) = G(z)X(z)$

La función de transferencia $G(z)$ del sistema que se muestra a continuación es

$$\frac{Y(z)}{X(z)} = G(z) = \mathbb{Z}[G(s)]$$

La relación de la transformada Z con la transformada asterisco se muestra en la figura 3.20.

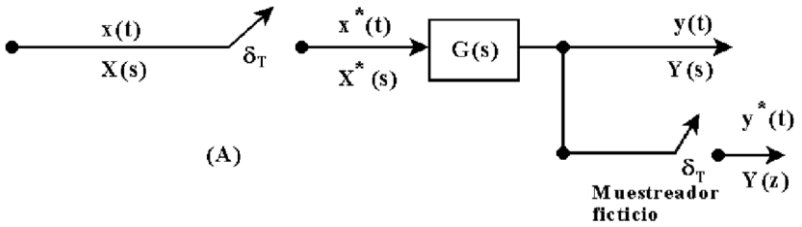


Figura 3.20. Transformada Z y asterisco.

Ahora, considerar el sistema de la figura 3.21.

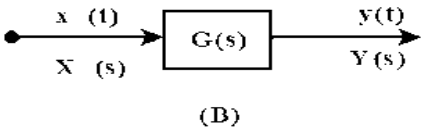


Figura 3.21. Diagrama de bloque.

La función de transferencia es

$$\frac{Y(s)}{X(s)} = G(s)$$

En este caso la función de transferencia $G(z)$ para este sistema no es $Z[G(s)]$ debido a la ausencia del muestre ador mediante impulsos.

Para el sistema de la figura (A), la transformada de Laplace de la salida, la transformada de Laplace de la salida $y(t)$ es

$$Y(s) = G(s)X^*(s)$$

Al aplicar la transformada de Laplace asterisco de $Y(s)$, se tiene

$$Y^*(s) = G^*(s)X^*(s)$$

en términos de la transformada Z

$$Y(z) = G(z)X(z)$$

Para la figura (B), la trasformada de Laplace de la salida $y(t)$ es

$$Y(s) = G(s)X(s)$$

entonces

$$Y^*(s) = [G(s)X(s)]^* = [GX(s)]^*$$

en términos de la transformada Z

$$Y(z) = Z[Y(s)] = Z[G(s)X(s)] = Z[GX(s)]$$

por lo que

$$Y(z) = GX(z) \neq G(z)X(z)$$

La presencia o ausencia del muestre ador a la salida del elemento (o el sistema) no afecta la función de transferencia, debido a que el si el muestre ador no está físicamente presente en el lado de la salida del sistema, es siempre posible suponer que al muestre ador ficticio esté presente a la salida.

3.10 Función de Transferencia de Sistemas en Lazo Cerrado

En un sistema de lazo cerrado, la existencia o no con un muestre ador de salida en el lazo hace que el comportamiento del sistema sea diferente. Considere el sistema de control en lazo cerrado que se muestra en la figura 3.22.

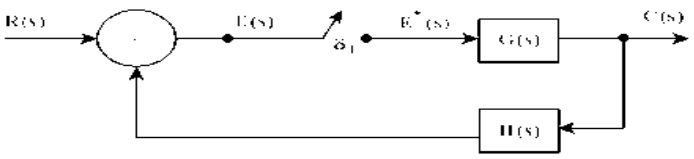


Figura 3.22. Sistema de lazo cerrado.

Las expresiones de este sistema en lazo cerrado son

$$E(s) = R(s) - H(s)C(s)$$

$$C(s) = G(s)E^*(s)$$

$$E(s) = R(s) - H(s)G(s)E^*(s)$$

Aplicando la transformada asterisco

$$E^*(s) = R^*(s) - GH^*(s)E^*(s)$$

$$\begin{aligned} E^*(s)\left(1 + GH^*(s)\right) &= R^*(s) \\ E^*(s) &= \frac{R^*(s)}{1 + GH^*(s)} \\ C^*(s) &= \frac{G^*(s)\,R^*(s)}{1 + GH^*(s)} \end{aligned}$$

en términos de la notación de transformada Z

$$C(z) = \frac{G(z)\,R(z)}{1 + GH(z)}$$

la función de transferencia de lazo cerrado es

$$\frac{C(z)}{R(z)} = \frac{G(z)}{1 + GH(z)}$$

Para los diagramas de bloques en lazo cerrado de las figuras 3.23, 3.24,3.25, 3.26 y 3.27 se tiene

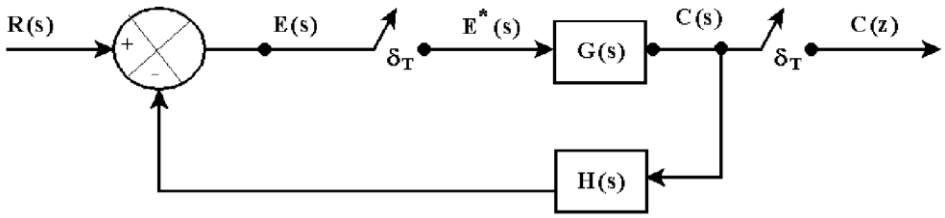


Figura 3.23. Sistema en lazo cerrado con un muestreador

$$C(z) = \frac{G(z)R(z)}{1 + GH(z)}$$

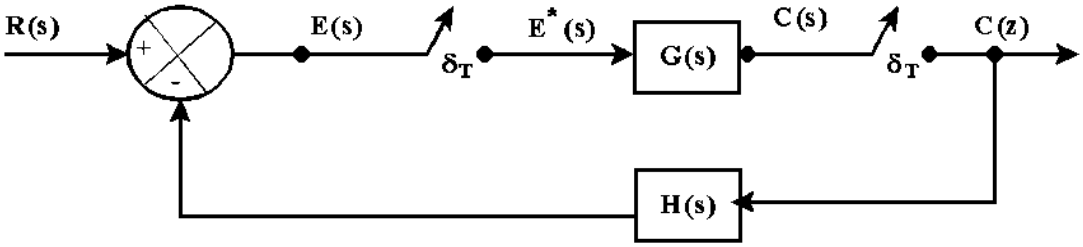


Figura 3.24. Sistema en lazo cerrado con dos muestreadores.

$$C(z) = \frac{G(z)R(z)}{1 + G(z)H(z)}$$

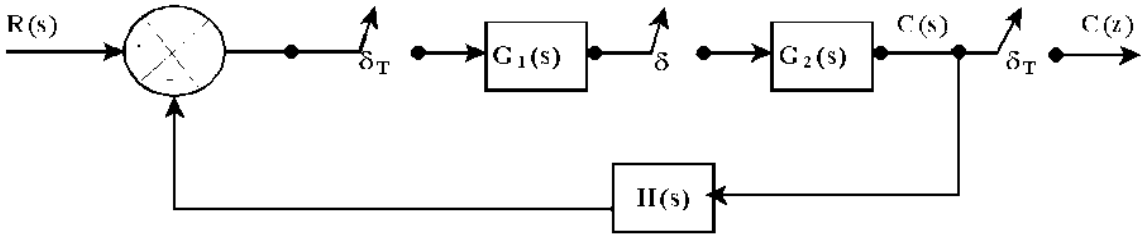


Figura 3.25. Sistema en lazo cerrado con tres muestreadores.

$$C(z) = \frac{G_1(z)G_2(z)R(z)}{1 + G_1(z)G_2(z)H(z)}$$

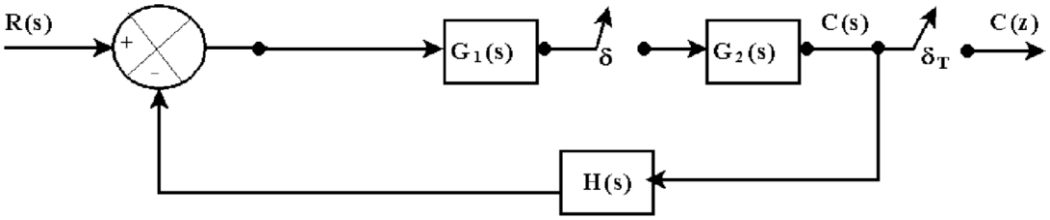
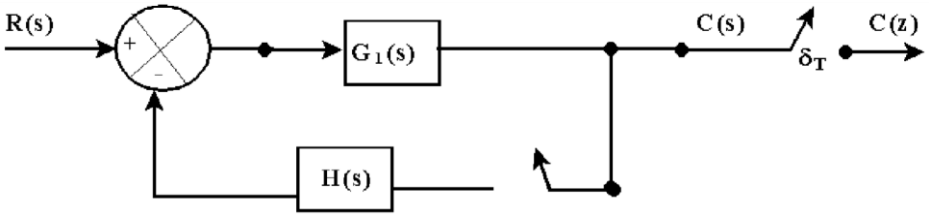


Figura 3.26. Sistema en lazo cerrado con dos muestreadores en diferente posición.

$$C(z) = \frac{G_1(z)\,G_2(z)\,R(z)}{1 + G_1G_2(z)\,H(z)}$$



$$C(z) = \frac{G_1(z)\,R(z)}{1 + G_1H(z)}$$

Figura 3.27. Sistema en lazo cerrado con un muestreador.

Capítulo 4

SIMULACION DE SISTEMAS DISCRETOS EN 20-SIM

4.1 Introducción a 20-sim

20-sim es un paquete de software de modelado y simulación para sistemas mecatrónicos. Se ejecuta en computadoras con sistema operativo Windows (Vista, 7, 8, 8.1 y 10) y requiere 450, MB de espacio en disco. En la figura 4.1, se ilustra una pantalla de edición de este software.

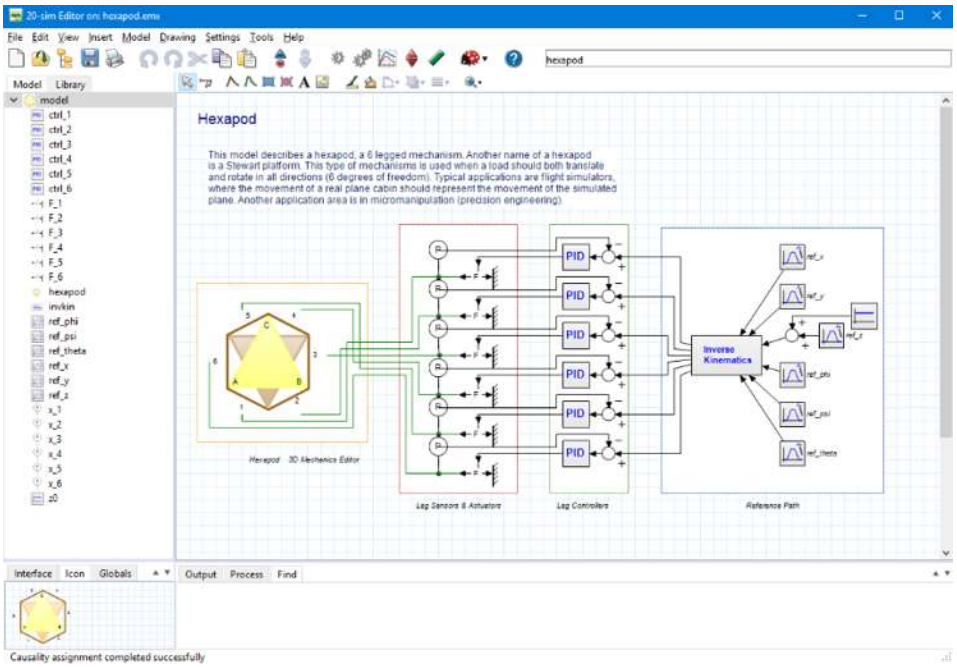


Figura 4.1. Ejemplo en 20 Sim.

Con 20-sim, los modelos se pueden crear gráficamente, de manera similar a dibujar un esquema de ingeniería. Con estos modelos, se puede analizar el comportamiento de los sistemas dinámicos y diseñar los sistemas de control. Los modelos de 20 sim se pueden exportar como código C para ejecutarse en hardware para la creación rápida de prototipos y la simulación HIL.

La imagen de arriba muestra 20 sim con un modelo de un hexápodo controlado. El mecanismo se genera con la función de mecánica 3D y se conecta con modelos de actuadores y sensores estándar de la biblioteca de mecánica. El ejemplo de un hexapódo es controlado por controladores PID que están sintonizados en el dominio de la frecuencia. Todo lo que se requiere para construir y simular este modelo está dentro del paquete. ¡No se necesita ningún software externo o compilador! Este artículo ofrece una pequeña introducción a 20 sim.

4.2 ¿Qué es 20-sim?

20-sim es un paquete de software de modelado y simulación para sistemas mecatrónicos. Con 20-sim puede ingresar modelos gráficamente, similar a dibujar un esquema de ingeniería. Con estos modelos, puede simular y analizar el comportamiento de sistemas dinámicos multidominio y crear sistemas de control. Incluso puede generar código C y ejecutar este código en hardware para la creación rápida de prototipos y la simulación HIL, como se muestra en la figura 4.2.

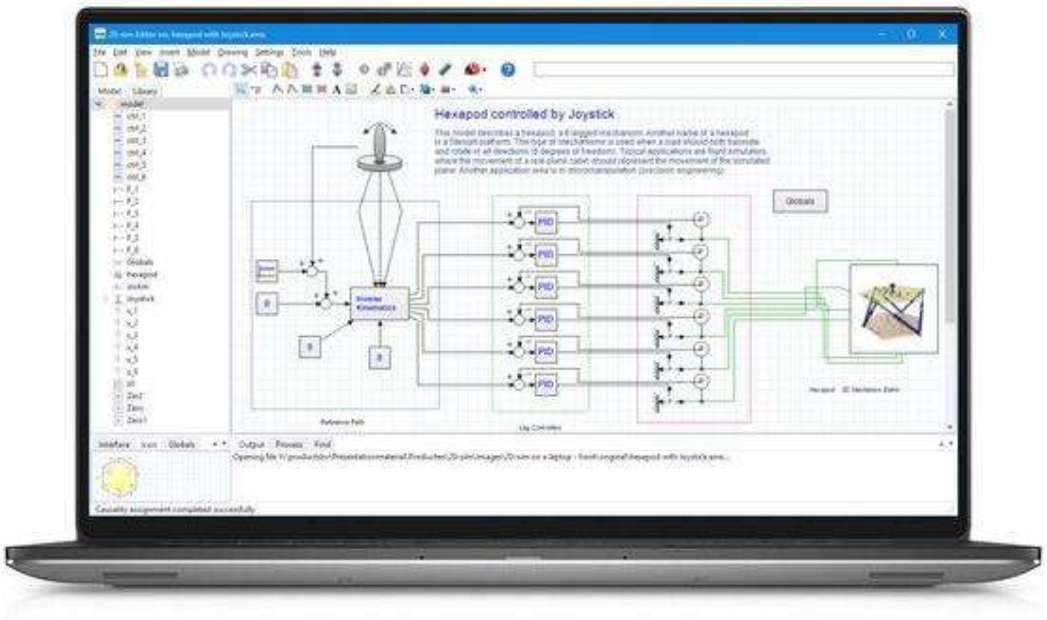


Figura 4.2. Simulación para un sistema mecatrónico en 20 Sim.

20-sim le proporciona características que le permiten crear modelos de manera muy rápida e intuitiva. Puede crear modelos utilizando ecuaciones, diagramas de bloques, bloques de física y gráficos de enlaces. Varias características lo ayudan a construir sus modelos, simularlos y analizar su rendimiento.

4.3 Editor

El software de 20 sim consta de 2 ventanas que están estrechamente integradas. Los modelos se crean en el editor y las ejecuciones de simulación y los resultados se muestran en el simulador. Cuando se inicia 20-sim, se abrirá el editor. El editor contiene un árbol de biblioteca de modelos desde el que puede arrastrar y soltar elementos en el lienzo de dibujo para construir sus modelos. El editor de 20 sim se mostrará como un editor gráfico o un editor de texto, según el modelo que se muestre. La biblioteca contiene elementos para construir modelos de gráficos de enlaces, componentes para construir sistemas físicos y bloques para construir modelos de diagrama de

bloques. Todos los elementos de la biblioteca están abiertos y el usuario puede cambiarlos. La biblioteca contiene los siguientes elementos:

- Gráfico de enlace: Elementos para construir modelos de gráfico de enlace.
- Diagramas icónicos: componentes para construir sistemas físicos.
- Eléctrico: Componentes para la construcción de redes eléctricas.
- Mecánico: Componentes para la construcción de estructuras mecánicas traslacionales y rotacionales.
- Hidráulica: componentes para la construcción de sistemas hidráulicos.
- Térmica: Componentes para modelar la transferencia de calor.
- Diagramas de bloques: bloques para construir modelos de diagrama de bloques: bloques lineales y no lineales, fuentes y sumideros, funciones de transferencia.
- Ejemplos: modelos de ejemplo que muestran el uso básico de los modelos de biblioteca.

4.3.1 Modelos gráficos

Los modelos en 20 sim están orientados jerárquicamente. El modelo en la parte superior se llama modelo principal. Está construido a partir de elementos gráficos que se denominan submodelos. Los submodelos se pueden conectar fácilmente. Dependiendo del submodelo, la conexión puede ser una variable compartida o una conexión física. Un submodelo en sí mismo puede construirse a partir de múltiples submodelos, y estos submodelos en sí pueden construirse con submodelos, con muchas capas de profundidad. En la parte inferior de la jerarquía, los modelos se describen mediante conjuntos de ecuaciones. Estos modelos se llaman modelos de ecuaciones.

4.3.2. Modelos de ecuaciones

Los modelos de ecuaciones se especifican en un lenguaje especial llamado SIDOPS ++. SIDOPS ++ tiene un gran parecido con Maple, Matlab y otros paquetes de software matemático, como se muestra en la figura 4.3.

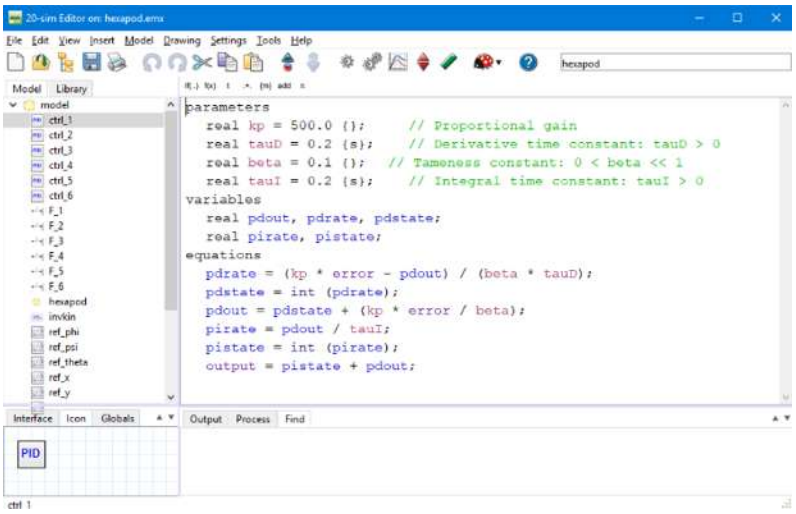


Figura 4.3. Programación de ecuaciones en 20 Sim.

La imagen de arriba muestra un ejemplo de un modelo de ecuación. Los modelos de ecuaciones de 20 sim tienen un diseño básico indicado por palabras clave

Parámetros: Definición de valores, que no cambian durante la simulación.

Variables: Definición de valores que cambian durante la simulación.

Ecuaciones: las ecuaciones reales.

Las ecuaciones son relaciones entre parámetros y variables e indicadas por un signo igual (=). Varias funciones y operadores matemáticos están disponibles para su uso en ecuaciones.

4.3.3. Herramientas de modelado

20-sim viene con una serie de características para construir modelos avanzados:

- Editor de diseño del controlador: esta característica ayuda a los usuarios a diseñar sistemas de retroalimentación con una planta lineal, controlador y prefiltro. Las respuestas de bucle abierto y cerrado pueden investigarse utilizando gráficos de Bode y Nyquist.

- Editor de mecánica 3D: los sistemas mecánicos 3D son notoriamente difíciles de modelar utilizando elementos 1D. Por lo tanto, el editor de mecánica 3D permite al usuario definir sistemas mecánicos arrastrando y soltando cuerpos, juntas y otros objetos en un espacio de trabajo 3D. El conjunto correspondiente de ecuaciones diferenciales optimizadas se genera automáticamente.

- Asistentes: varios asistentes lo ayudarán a crear perfiles de movimiento , definir levas , construir servomotores y mucho más.

4.3.4 Simulación

Cuando un modelo está listo, el simulador se puede abrir desde el editor. Bajo el capó, el modelo se compila automáticamente para crear un código de simulación. No se requieren herramientas externas. En la fase de compilación, 20-sim verificará si el modelo es correcto y optimizará las ecuaciones. El simulador se utiliza para ejecutar simulaciones y analizar modelos. Antes de que se pueda iniciar una ejecución de simulación, el usuario debe definir algunas configuraciones:

- Propiedades de ejecución: la hora de inicio y la hora de finalización de una ejecución.

- Método de integración: 20-sim admite varios métodos numéricos avanzados para ejecutar una simulación. El método numérico se puede elegir junto con la configuración adecuada. Estas configuraciones incluyen, por ejemplo, el error de integración máxima y el tamaño del paso.

- Valores de parámetros: antes de ejecutar una simulación, es posible que los valores de los parámetros predeterminados tengan que cambiarse.

- Propiedades del gráfico: se debe establecer el número y la apariencia de los gráficos y se deben elegir las variables que se van a representar. En la figura 4.4 se ilustra una gráfica obtenida de la simulación de un modelo en 20-sim.

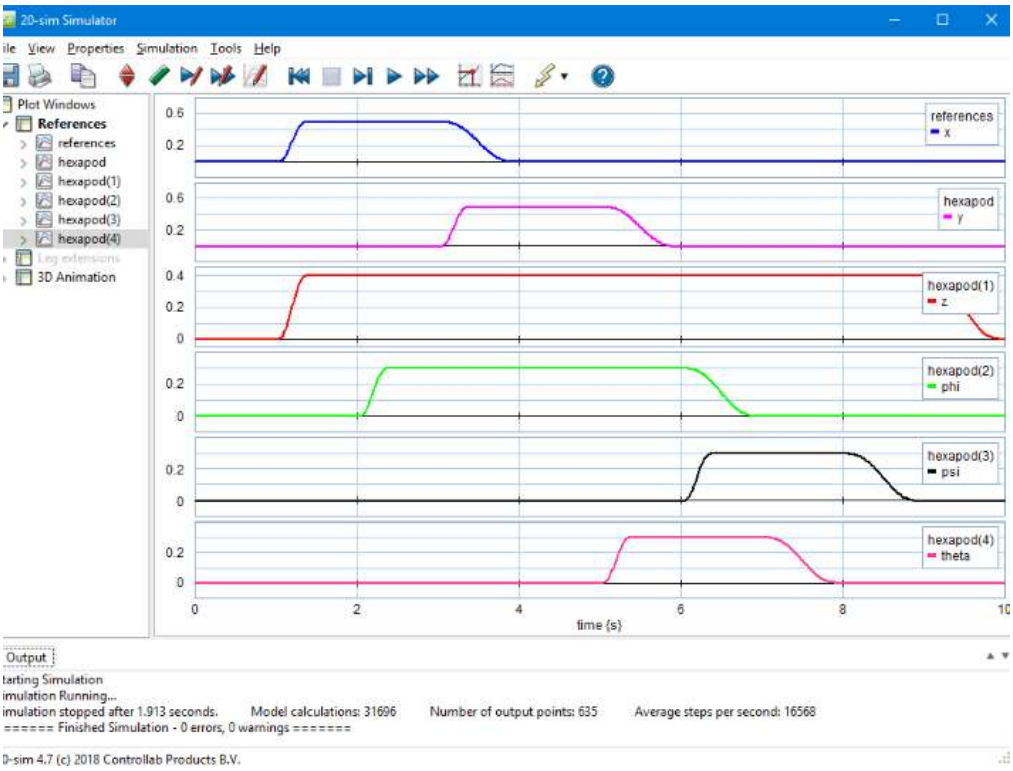


Figura 4.4. Simulación de un proceso en 20 Sim.

Junto a las opciones de simulación, los resultados también se pueden mostrar como animaciones 3D en 20 sim. Está disponible un editor especial en el que cualquier variable se puede conectar a la posición, orientación, tamaño y color de los objetos 3D. Los objetos 3D estándar están disponibles como cubos y esferas, pero los objetos también se pueden importar desde paquetes CAD.

a) Análisis

El paquete de 20-sim tiene dos características que se pueden utilizar para analizar modelos.

- Dominio del tiempo: Esto permite el análisis del modelo cambiando los valores de los parámetros y utilizando múltiples ejecuciones de simulación. Los barridos de parámetros, la optimización y el ajuste de curvas ayudarán a mejorar el rendimiento del sistema. El análisis de sensibilidad, el análisis de Monte Carlo y el análisis de variación ayudarán a verificar la solidez de un sistema.

- Dominio de frecuencia: los modelos en 20-sim se pueden linealizar para mostrar el sistema lineal correspondiente en el Editor de sistema lineal. El editor de sistemas lineales es una herramienta especializada para el diseño y visualización de sistemas lineales. El editor admite sistemas SISO de tiempo continuo y de tiempo discreto y puede mostrar la respuesta del sistema mediante gráficos de bode y Nyquist. Si los modelos no se pueden linealizar, las transformadas rápidas de Fourier se pueden usar para mostrar el comportamiento de frecuencia de un modelo.

b) Scripting

Con las secuencias de comandos, las tareas se pueden ejecutar en 20-sim automáticamente utilizando funciones de secuencias de comandos especializadas. Con estas funciones, los modelos

se pueden abrir y ejecutar automáticamente, los parámetros se pueden cambiar, los resultados se pueden exportar y mucho más. Una simulación con scripting es mostrada en la figura 4.5.

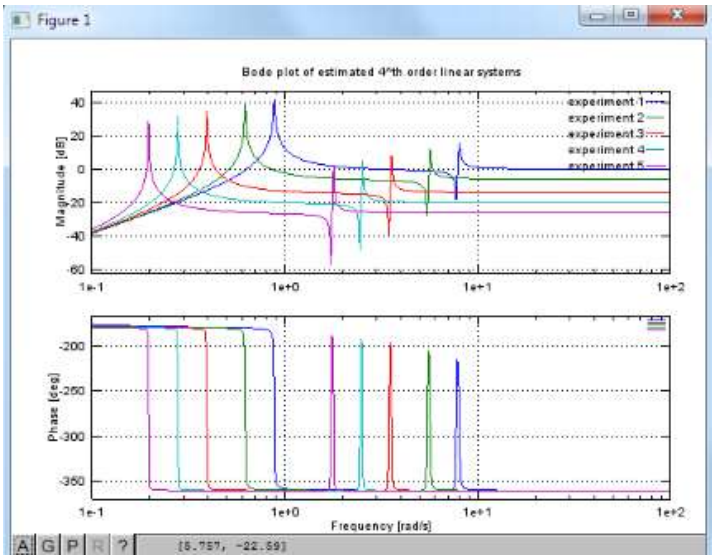


Figura 4.5. Simulación de un experimento de Scripting en 20-Sim.

Las funciones de scripting de 20 sim se pueden ejecutar como archivos m en Matlab o GNU Octave. GNU Octave es un entorno de código abierto que permite ejecutar archivos m similares a Matlab. Las funciones de secuencias de comandos están disponibles para abrir y simular modelos de 20-sim, exportar valores de parámetros a un modelo de 20-sim, ejecutar simulaciones, exportar gráficos de simulación y mucho más.

4.3.5 Código de GENERACION

De cualquier modelo de 20-sim, el código C se puede generar para su uso en sistemas externos, simuladores HIL, etc. Las plantillas permiten la personalización del código C con comandos previos y posteriores, enlaces de archivos, comentarios, etc. plantillas que le permiten generar código para varios objetivos:

- 20-sim 4C: el paquete 20-sim 4C ayuda a ejecutar el código C en hardware para controlar máquinas y sistemas. 20-sim 4C importa modelos (código c) de 20-sim y los ejecuta en hardware como placas de brazo integradas, sistemas PC 104 y mucho más.

- Matlab Simulink: la generación de código C para usar en MATLAB Simulink también incluye un bloque de submodelo con terminales de entrada y salida. 20-sim usa el compilador MEX, para compilar este código directamente en una función S. Estas funciones S también se pueden usar en el Taller en tiempo real para generar código para una plataforma específica, por ejemplo, xPC Targets.

- Código C: 20-sim puede generar código C independiente para su uso en programas C y C ++. El código C generado se suministra con varios algoritmos de simulación de pasos fijos para permitir que se ejecute en tiempo real. Los métodos Euler y Runge-Kutta son compatibles de forma predeterminada.

4.4 CARACTERÍSTICAS

4.4.1 Animación 3D

La animación 3D es un método poderoso para inspeccionar y verificar los resultados de la simulación, especialmente cuando se trabaja en tres dimensiones. Los resultados de la simulación en 20 sim se pueden mostrar como animaciones utilizando un editor de animación 3D. Las animaciones se componen de objetos predefinidos como cubos, esferas, líneas, cuadrados, cámaras y luces. Los objetos complejos se pueden importar desde paquetes CAD utilizando formatos de intercambio comunes. En la figura 4.6 se muestra una simulación con animación en este software.

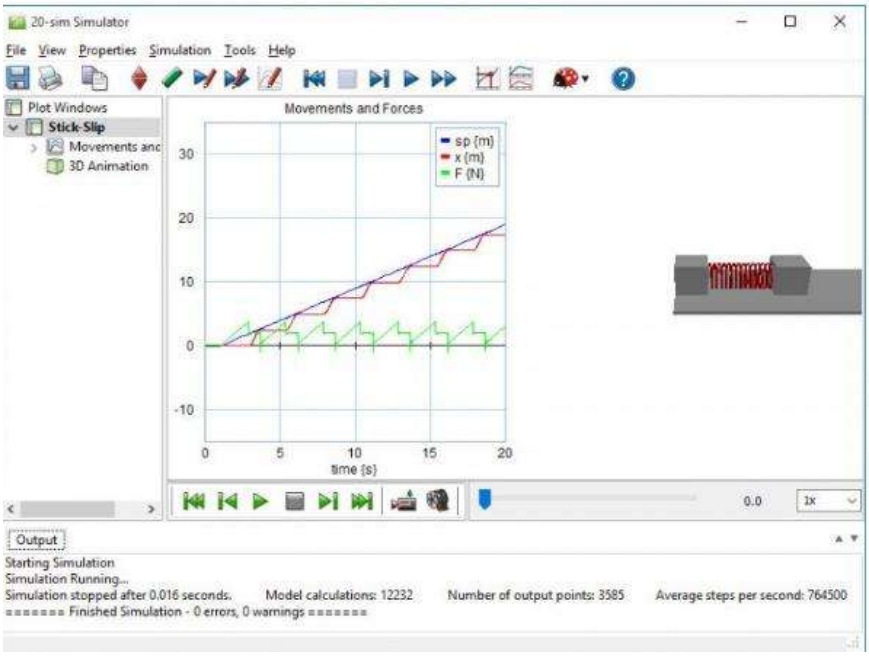


Figura 4.6. Simulación en Animación 3D en 20-Sim.

a) Objetos

Cualquier variable de un modelo de 20-sim se puede conectar a un objeto. Se pueden controlar varios atributos del objeto de esta manera: posición, orientación, tamaño, etc. Se pueden crear gráficos térmicos controlando el color de los objetos. Los objetos de marco de referencia se pueden usar para agrupar objetos de animación y heredar atributos de objeto. Los objetos se pueden duplicar, lo que resulta en una animación compleja con unos pocos clics del mouse.

b) Conectado a la simulación

Las animaciones están completamente vinculadas al Simulador en 20 sim. Mientras se dibuja una trama, simultáneamente se mostrará la animación. Este enlace se mantiene después de que una simulación ha finalizado. Al inspeccionar los valores numéricos, notará que la animación 3D cambia simultáneamente.

c) Películas

Cada animación se puede exportar como una película . 20-sim admite todos los formatos estándar, incluidos YouTube, AVI y WMV.

d) Mecánica 3D

3D Mechanics lo ayuda a construir sistemas mecánicos 3D rápidamente en 20-sim. Puede crear sistemas de varios cuerpos arrastrando cuerpos en un espacio de trabajo 3D. Las representaciones de cada cuerpo se pueden cambiar a una esfera, bloque, cilindro, etc. Además, los colores se pueden cambiar y se pueden agregar descripciones. El tamaño y la forma de un cuerpo son meramente representativos, un cuerpo se caracteriza completamente por sus coeficientes de inercia y masa. La figura 4.7 ilustra una simulación de mecánica 3D.

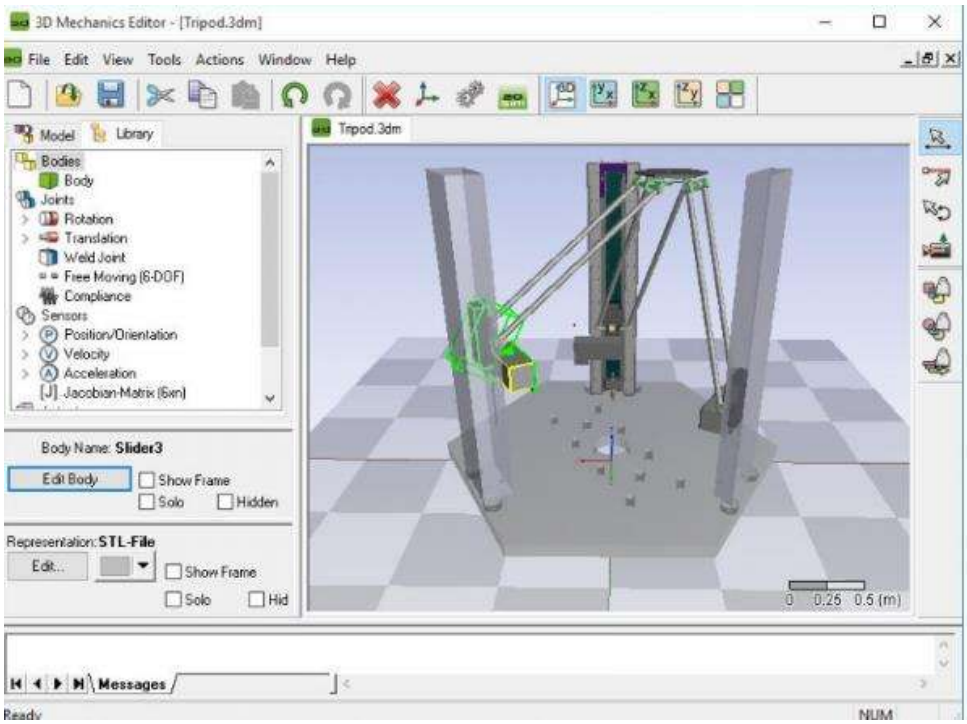


Figura 4.7. Modelado en Mecánica 3D en 20-Sim.

e) Articulaciones

Los cuerpos están interconectados mediante el uso de articulaciones. Varias juntas están presentes en la biblioteca, divididas en dos grupos, juntas rotacionales y juntas traslacionales. Estas uniones también se pueden arrastrar y soltar en el espacio de trabajo. Se pueden agregar restricciones para crear sistemas de circuito cerrado como mecanismos de cuatro barras o plataformas Stewart.

f) Interfaz

La interfaz de usuario tiene 4 modos diferentes en los que puede seleccionar, conectar, trasladar y rotar cuerpos y articulaciones. Se hace un gran esfuerzo para mantener la interfaz gráfica de usuario lo más natural posible. Se admiten múltiples vistas. Además del entorno 3D, puede ver intersecciones 2D en los planos xy xz e yz.

g) Modelos

El editor de mecánica 3D puede generar un modelo de 20 sim de su modelo 3D. Este modelo de 20 sim comprende todas las dinámicas y cinemáticas del modelo. Las fuerzas se pueden aplicar a las articulaciones o directamente a cada cuerpo. También puede acoplar resortes y amortiguadores desde la biblioteca de mecánica en 20-sim a las juntas, porque todo el modelo está basado en puertos. La gravedad se puede establecer como una fuerza externa. Eventualmente, la respuesta dinámica del modelo completo puede ser mostrada por la función de Animación 3D.

4.4.2 Código de Generación

Código de Generación le permite generar código C a partir de cualquier modelo de 20 sim. Usando plantillas integradas, se puede generar código C para varios objetivos.

a) 20-sim 4C

El paquete 20-sim 4C le ayuda a ejecutar código C en hardware para controlar máquinas y sistemas. 20-sim 4C permite la creación rápida de prototipos para ingenieros de control. Con 20-sim 4C puede ejecutar código c en hardware para controlar máquinas y sistemas. 20-sim 4C importa modelos (código c) de 20-sim, Simulink y Scilab y los ejecuta en hardware como placas de brazo integradas, sistemas PC 104 y mucho más.

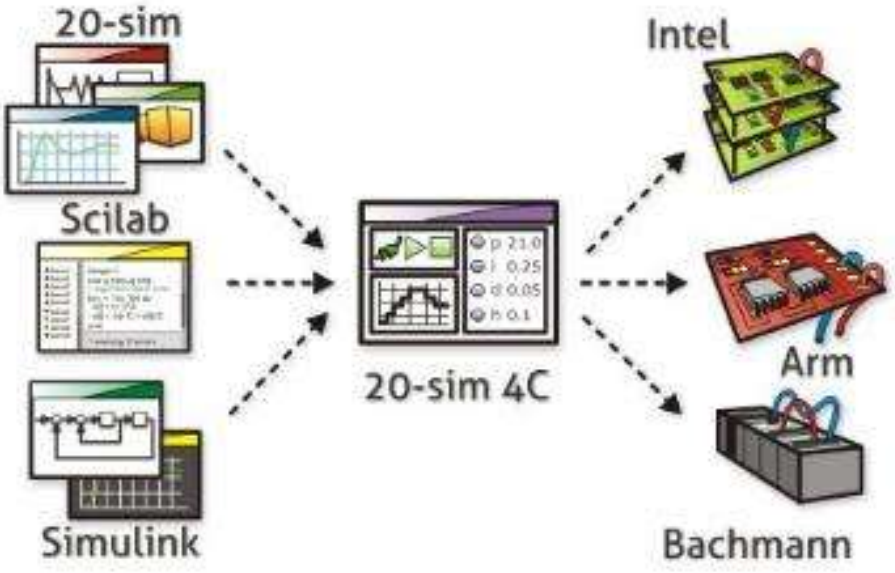


Figura 4.8. Herramientas de 20-Sim 4C.

20-sim 4C puede ser utilizado por estudiantes y profesionales. Con solo unos pocos clics del mouse, puede cargar el código C y ejecutarlo en el hardware. Desde 20-sim 4C puede iniciar y detener el código, inspeccionar variables y registrar datos. 20-sim 4C es versátil. Con 20-sim 4C

puede realizar mediciones, ejecutar actuadores y controlar máquinas. Se admite una amplia gama de hardware que le permite realizar todas las tareas.

b) Matlab Simulink

La generación de código C para usar en MATLAB Simulink también incluye un bloque de submodelo con terminales de entrada y salida. 20-sim usa el compilador MEX, para compilar este código directamente en una función S. Estas funciones S también se pueden usar en el Taller en tiempo real para generar código para una plataforma específica, por ejemplo, xPC Target.

c) C-Code

20-sim puede generar código C independiente para usar en programas C y C ++. El código C generado se suministra con varios algoritmos de simulación de pasos fijos para garantizar que se ejecutará en tiempo real. Los métodos Euler y RungeKutta son compatibles de forma predeterminada.

d) Plantillas

Todas las plantillas de código C están abiertas y el usuario puede adaptarlas para asignar compiladores, ejecutar sesiones ftp y automatizar casi todo entre la generación de código de 20 sim y la ejecución real del código en una máquina (remota). Durante la generación del código C, el código de soporte se genera para operadores de 20 sim como cálculos matriciales y funciones trigonométricas.

4.4.3 Diseño de controlador

Controller Design te ayuda a diseñar sistemas de control en 20 sim. Contiene varias herramientas que pueden ayudarlo a desarrollar controladores para máquinas como el Editor de diseño de controladores, el Editor de filtros y los Editores de redes neuronales. Este editor es una herramienta especializada para el diseño de sistemas de control de retroalimentación. Se presenta una estructura de retroalimentación de subsistemas con una planta lineal, controlador, medición y pre filtro. También están disponibles las ganancias de bucle abierto y de bucle cerrado y las sensibilidades / lag, o filtros de muesca. En la figura 4.9 se ilustra un ejemplo de un controlador en 20-sim.

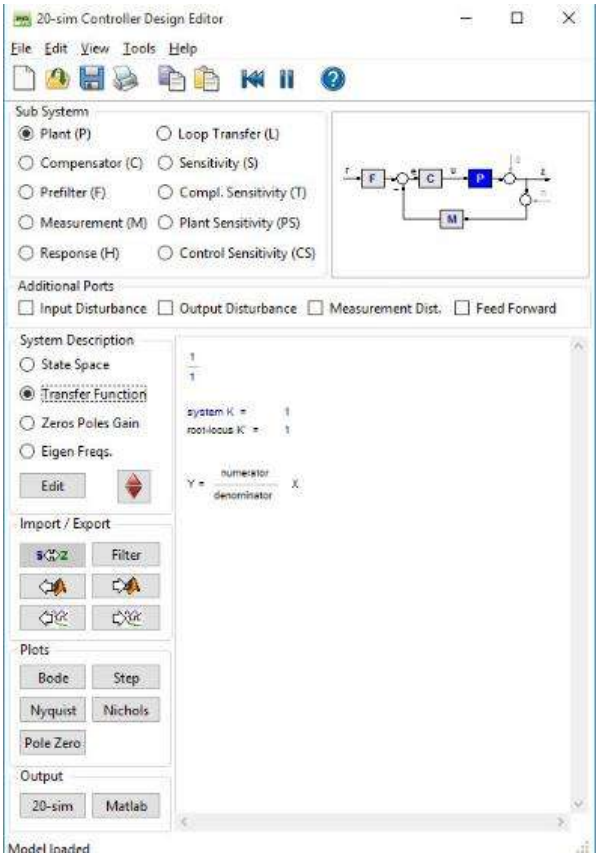


Figura 4.9. Diseño de un controlador en 20-Sim.

Puede editar su controlador como un sistema ABCD State Space, una función de transferencia o en un formulario de ganancia de polo cero. Los cambios en uno de los subsistemas actualizan directamente todos los diagramas y diagramas abiertos. Por ejemplo, la adaptación de la ganancia del controlador cambia inmediatamente los polos y ceros del sistema de circuito cerrado y la respuesta general del paso. ¡La integración en el intercambio de sistemas lineales y de 20-sim con MATLAB hace de este editor una herramienta poderosa para diseñar sistemas de control de retroalimentación!. En la figura 4.10 se muestra la respuesta de un sistema en lazo cerrado.

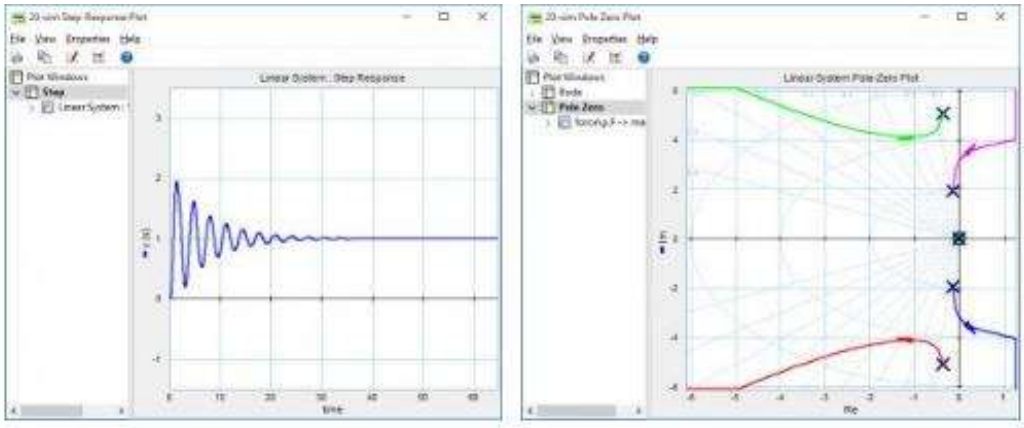


Figura 4.10. Graficas de los polos y ceros del sistema de circuito cerrado en 20 Sim.

4.4.4 Editor de filtros

Con el editor de filtros puede crear sus propios filtros lineales de acuerdo con sus especificaciones. Los filtros disponibles son los filtros Bessel, Butterworth y ChebyChev donde puede especificar el orden y las frecuencias características. También se puede elegir entre filtros PID, adelanto / retraso o de muesca.

4.4.5 Editores de redes neuronales

El diseño del controlador es compatible con dos redes conocidas: las redes adaptativas B-Spline y las redes de múltiples capas Perceptron. Estas redes neuronales se deben entrenar presentando repetidamente ejemplos a la red. Cada ejemplo incluye entradas y salidas deseadas. Según el error entre las salidas deseadas y la salida de la red real, la red neuronal adapta el peso de cada neurona de acuerdo con una tasa de aprendizaje definida por el usuario. Si la respuesta es lo suficientemente precisa, puede guardar los pesos de las neuronas para utilizar la red neuronal en su controlador. Se puede usar un número ilimitado de neuronas.

4.4.6 Presupuesto de error dinámico

El rendimiento de las máquinas de precisión está limitado principalmente por las perturbaciones que se inyectan en estas máquinas. Estas perturbaciones suelen ser de naturaleza estocástica. El presupuesto dinámico de errores es un método mediante el cual se puede calcular el efecto de estas perturbaciones en el rendimiento final. La ventaja de este método es que permite al diseñador ingresar las contribuciones de las perturbaciones individuales y ver y optimizar el rendimiento general de la máquina. Puede usar el presupuesto dinámico de errores para diseñar de manera efectiva máquinas de ultra alta precisión, tal como se muestra en la figura 4.11.

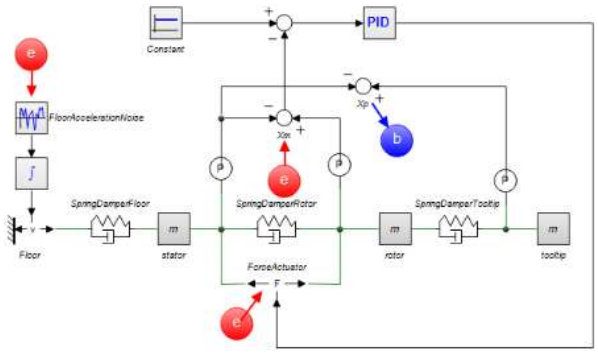


Figura 4.11. Presupuesto de error dinámico en 20-Sim.

Puede abrir la **herramienta de presupuesto dinámico de errores** en el simulador de 20-sim. Esta herramienta le permite ingresar perturbaciones (como densidades espectrales de potencia) en cualquier punto de su modelo. Para cada perturbación simplemente tiene que seleccionar una variable de modelo correspondiente y definir las propiedades de ruido. Al seleccionar la salida, puede inspeccionar los resultados de estas perturbaciones en esa salida específica, como se muestra en la figura 4.12.

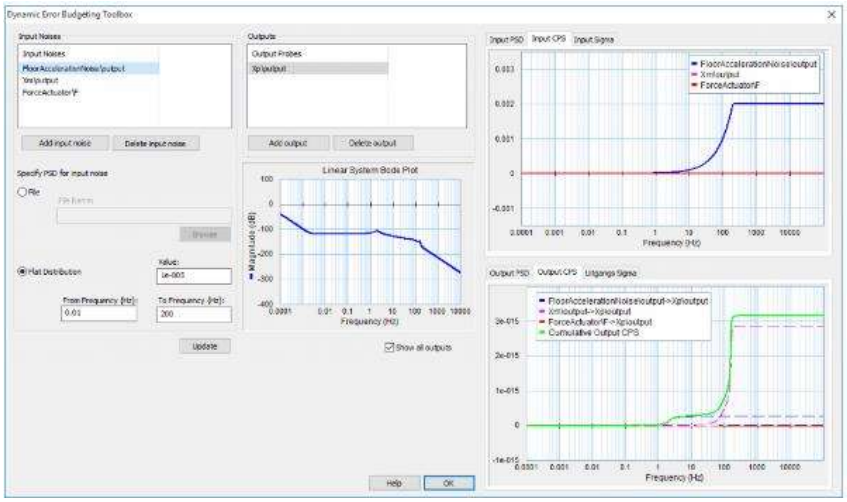


Figura 4.12. Gráficas de presupuesto de error dinámico en 20-Sim.

En la gráfica en la parte inferior derecha puede ver el error resultante en la salida seleccionada como resultado de las perturbaciones. El error se da en forma de densidad espectral de potencia (PSD) y densidad espectral de potencia acumulada (CPS). La raíz cuadrada del valor final del CPS es igual a la desviación estándar del error de salida. Las desviaciones estándar se muestran en la pestaña Salida Sigma.

La herramienta de presupuesto dinámico de errores es parte de la función de dominio de frecuencia de 20-sim.

4.5 Editor

Los modelos se ingresan y compilan en el editor de 20-sim. El editor es una herramienta versátil que lo ayuda a ingresar modelos que admiten una amplia variedad de sistemas, incluidos sistemas lineales, no lineales, de tiempo discreto, de tiempo continuo e híbridos, sin restringir al usuario a una determinada representación del modelo. Después de ingresar y depurar, el modelo se puede verificar y compilar. Esto se realiza automáticamente en segundo plano, al abrir el simulador como se muestra en la figura 4.13.

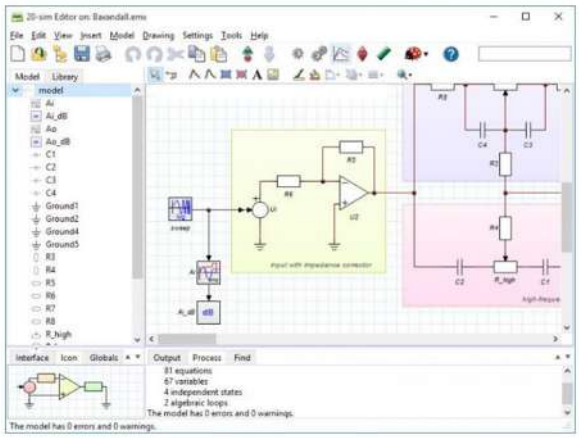


Figura 4.13. Modelo hibrido en 20-Sim.

a) Representaciones modelo

Los sistemas se pueden modelar en 20-sim, utilizando ecuaciones, descripciones de espacios de estado, diagramas de bloques de gráficos de enlaces y componentes o diagramas icónicos. Estas descripciones se pueden combinar completamente para crear modelos mixtos.

b) Fuente abierta

Los modelos de 20-sim se almacenan en archivos. El Explorador de Windows se usa para la administración de bibliotecas. ¡Todos los modelos de 20-sim están abiertos! Puede arrastrarlos y

soltarlos desde el Explorador de Windows al editor de 20-sim y cambiarlos. Puede almacenar los modelos originales y modificados en carpetas separadas para crear su propia biblioteca de modelos.

c) Dominio de la frecuencia

El dominio de frecuencia consta del editor de sistema lineal, las herramientas de análisis FFT, la funcionalidad de linealización del modelo y el presupuesto dinámico de errores.

d) Editor de sistema lineal

El editor de sistemas lineales es una herramienta especializada para el diseño y análisis de sistemas lineales. El editor admite sistemas SISO de tiempo continuo y tiempo discreto que utilizan varias representaciones. La interfaz gráfica le permite editar un sistema lineal en cualquier forma deseada: espacio de estado ABCD, función de transferencia o ganancia de polo cero. El análisis de la respuesta de Paso, el diagrama de Bode, el diagrama de Nyquist, el diagrama de Nichols y los diagramas de Pole-Zero le permiten evaluar rápidamente el comportamiento del sistema, lo cuál se ilustra en la figura 4.14.

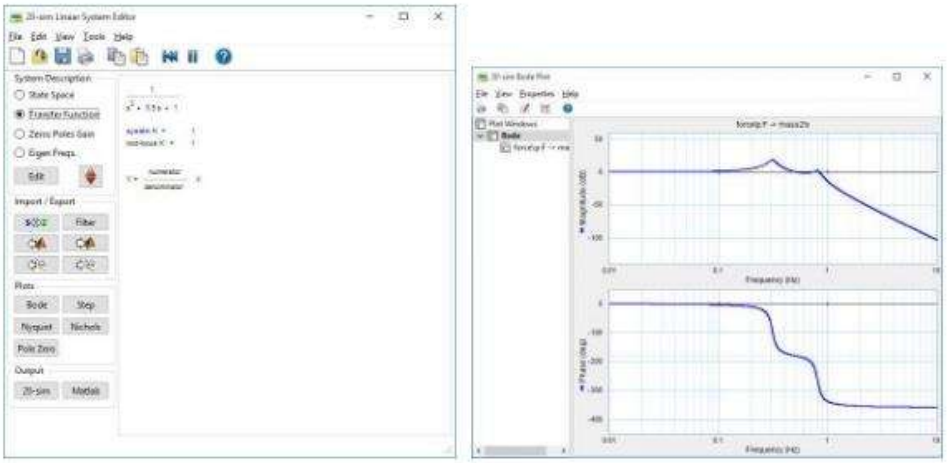


Figura 4.14. Modelo de sistema lineal en 20-Sim.

Se calculan los márgenes de fase, ganancia y módulo, así como el tiempo de aumento, el sobreimpulso y el valor de estado estable. La entrada puede originarse en un modelo de sistema lineal de 20 sim, filtro de 20 sim o editor de control, espacio de trabajo de MATLAB o entrada del usuario. Puede generar resultados para todos los editores de 20 sim, el portapapeles y el espacio de trabajo de MATLAB.

e) Transformada rápida de Fourier

Las transformaciones rápidas de Fourier (FFT) se pueden aplicar a cualquier diagrama de dominio de tiempo en 20 sim. Los resultados de la simulación o los datos de medición funcionarán. Cuando los datos no están igualmente espaciados, primero se aplica la interpolación lineal, después de lo cual se usa la Transformada rápida de Fourier para calcular el contenido de frecuencia. Se admiten tres representaciones: gráfico de amplitud y fase, gráfico de frecuencia y gráfico de densidad espectral de potencia.

f) Linealización

Cualquier modelo de 20 sim se puede linealizar en forma de espacio de estados. Si es posible, la linealización se realizará simbólicamente. De lo contrario, la linealización se realizará numéricamente. El modelo de espacio de estado resultante se muestra en el Editor de sistema lineal de 20 sim. El editor de sistemas lineales es una herramienta especializada para el diseño y análisis de sistemas lineales. El editor admite sistemas SISO de tiempo continuo y tiempo discreto que utilizan varias representaciones. Las parcelas estandarizadas le permiten evaluar rápidamente el comportamiento del sistema. Si es posible, cualquier modelo de 20 sim se puede linealizar.

g) Características

- Edición como espacio de estado ABCD, Función de transferencia o Ganancia de polo cero con transformación automática entre estas formas.
- Transferencia entre representación en tiempo continuo y en tiempo discreto.
- Ver propiedades características como frecuencias propias y amortiguación.
- Maneja modelos numéricos y simbólicos.
- Varias opciones de trazado: Respuesta de paso, Gráfico de Bode, Diagrama de Nyquist, Gráfico de Nichols y Gráfico de polo cero.

h) Presupuesto dinámico de errores

El presupuesto dinámico de errores es un método mediante el cual se puede calcular el efecto de las perturbaciones en el rendimiento final de un sistema. 20-sim tiene una herramienta especial que hace que el presupuesto dinámico de errores sea simple y directo.

4.6 Mecatrónica

La mecatrónica le ayuda a modelar sistemas mecatrónicos. Esta característica incluye el Asistente de perfil de movimiento, el Asistente de CAM y el Editor de servomotores.

a) Editor de servomotores

El Editor de servomotor es un programa que ayuda a los ingenieros a elegir el servomotor adecuado para cualquier sistema electromecánico:

- Cepillo DC (motor de armadura de hierro, motor de rotor hueco, motor de armadura de disco)

- DC sin escobillas
- AC síncrono
- AC síncrono lineal

En cooperación con los fabricantes de motores, se han creado tablas de datos de motores para Servo Motor Editor. El rendimiento de cada motor se puede mostrar mediante la curva de velocidad de par. El Servo Motor Editor puede generar modelos dinámicos para el programa de simulación 20-sim. En este programa, puede simular el comportamiento térmico y dinámico del servomotor en combinación con bucles de control y cargas dinámicas.

Cualquier ingeniero involucrado en el diseño de máquinas electromecánicas puede beneficiarse del Servo Motor Editor, el cuál se ilustra en la figura 4.15. Se puede ahorrar un tiempo y dinero al encontrar el servomotor óptimo en pocos minutos, sin correr el riesgo de sobrecalentamiento o falta de potencia.

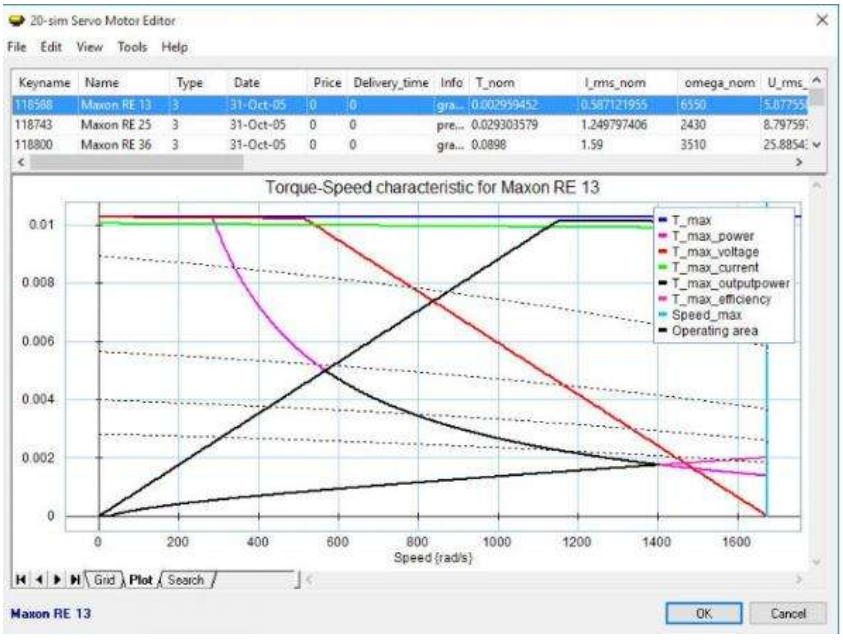


Figura 4.15. Gráfica de un servomotor en 20-Sim.

Las características son las siguientes:

- Soporte de tablas de datos del motor predefinidas.
- Agregue sus propios motores a la tabla de datos.
- Búsqueda rápida por múltiples parámetros.
- Curvas de par-velocidad con: línea de corriente máxima, par máximo, voltaje máximo, velocidad máxima, potencia máxima, eficiencia máxima y potencia máxima de salida.
- Área de operación segura, área de operación deseada.
- Genere modelos dinámicos para el programa de simulación 20-sim. Estos modelos incluyen efectos térmicos de las bobinas y la carcasa, pérdidas eléctricas por disipación, pérdidas magnéticas por histéresis, corrientes parásitas y engranajes.
- Mostrar curva de velocidad de par con curva de carga dinámica.
- Modelos de plantillas, que cubren la mayoría de los esquemas de control comercial y muchas cargas mecánicas.

Áreas de aplicación

- Diseño de control
- Equipos Industriales y Maquinaria
- Ingeniería de precisión
- Simulación
- Servicios públicos y energía

- Análisis y control de vibraciones

El Asistente de perfil de movimiento le ayuda a crear perfiles de movimiento de una manera fácil de usar. Puede construir el perfil combinando secciones que pueden contener movimientos predefinidos. Se admiten las siguientes mociones:

- Plano
- Rampa
- Trapezoidal
- Trapezoidal parcial
- Mecanismo de Ginebra
- Seno
- Cúbico
- Cúbico parcial
- Cicloidal
- Seno modificado estándar
- Trapezoidal modificado
- Seno modificado con velocidad constante (MSC50)
- Seno modificado general con velocidad constante (MSC%)
- 3-4-5 polinomio
- 1-3-5-7-9 Polinomio

El asistente de cámara de 20 sim le ayuda a generar modelos de cámara y mecanismo de leva. Al igual que el Asistente de perfil de movimiento, puede usar varios tipos de perfiles de movimiento que son continuos en velocidad, aceleración o tirón.

4.7 Bibliotecas modelo

20-sim viene con una gran colección de componentes y submodelos que se almacenan en bibliotecas. Las bibliotecas están disponibles de forma estándar en todas las versiones de 20-sim. Con estos componentes de la biblioteca, puede crear sistemas hidráulicos, estructuras mecánicas, accionamientos eléctricos, sistemas de control y mucho más. Puede inspeccionar las bibliotecas haciendo clic en la pestaña Biblioteca en el lado izquierdo del Editor.

La biblioteca de Diagramas icónicos contiene bloques de construcción físicos para sistemas eléctricos, mecánicos, hidráulicos y térmicos.

a) Eléctrico

- Varias fuentes de tensión y corriente.
- Inductores, resistencias, condensadores, etc.
- Diodos, amplificadores operacionales y rectificadores

b) Hidráulico

- Orificios y resistencias laminares
- Cilindros, acumuladores, tanques.
- Bombas y motores.
- Valvulas

- Sensores de flujo, potencia y presión.

c) Térmico

- Capacidad de calor y flujo de calor
- Elementos de convección y radiación.
- Sensores
- Generadores de calor

d) Mecánico

- Inercia, muelles, amortiguadores, etc.
- Modelos de rodamientos, holgura y embrague
- Modelos avanzados de fricción (LuGre, Dahl, KFM)
- Correas, husillos, engranajes y diferenciales.
- Varios tipos de sensores y codificadores.

Todos los elementos en la biblioteca mecánica están definidos para el dominio rotacional y traslacional. Además, hay una biblioteca especial para cuerpos 2D y 3D con pequeñas rotaciones.

e) Señal

La biblioteca de señales contiene una gran colección de bloques de construcción para Diagramas de bloques.

- Ganar, integrar, diferenciar, retrasar, etc.
- Varios modelos de filtros (Butterworth, Bessel)

- Elementos discretos (AD, DA, muestra, retención)
- Generadores de ondas y señales.
- Operadores logicos
- Varios controladores

4.8 Gráfico de bonds

20-sim contiene una biblioteca de gráficos de enlaces con todos los elementos de gráficos de enlaces estándar.

- Fuentes: Se, Sf, MSe, MSf
- Elementos de almacenamiento: I, C
- Resistencia: R
- Transformadores: TF, GY, MTF, MGY
- Uniones 1, 0.

Puede crear elementos de gráficos de enlace personalizados y agregarlos a la biblioteca.

4.9 Personalización

Todos los modelos de 20-sim se almacenan en archivos. Una carpeta con una colección de archivos de modelo se denomina biblioteca de modelos. En la configuración del Editor, puede definir qué bibliotecas de modelos deben mostrarse en el Editor. Todos los modelos están abiertos. Puede inspeccionar sus contenidos y cambiarlos para crear nuevos modelos. Al recopilar modelos en una nueva carpeta, puede crear bibliotecas personalizadas.

a) Diagramas de bloques

Los diagramas de bloques le permiten representar gráficamente las relaciones matemáticas entre señales en un sistema, como se muestra en la figura 4.16. Son especialmente adecuados para modelar sistemas de control. En 20-sim está disponible una gran biblioteca de elementos de diagrama de bloques. Los elementos se muestran en el Editor mediante iconos. Puede crear modelos de diagrama de bloques arrastrando los elementos al Editor y haciendo las conexiones adecuadas entre los elementos. 20-sim le permite crear elementos de diagrama de bloques definidos por el usuario con un número arbitrario de señales de entrada y salida. El tamaño de la señal puede ser 1 (predeterminado) o mayor.

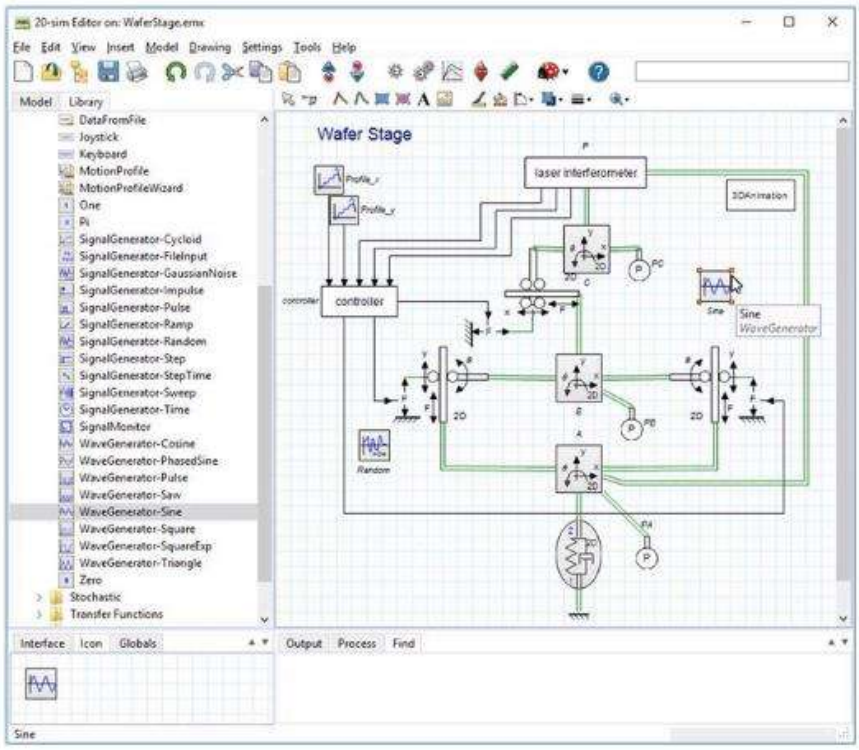


Figura 4.16. Diseño de un diagrama de bloque en 20-Sim.

20-sim tiene una gran biblioteca de elementos de diagrama de bloques como elementos lineales, no lineales, discretos y fuente. En 20-sim puede crear elementos de diagrama de bloques personalizados y agregarlos a las bibliotecas existentes o combinarlos en bibliotecas recientemente definidas. Desde el Navegador de la biblioteca (izquierda) puede arrastrar y soltar elementos en el Editor (derecha).

b) Gráficos de bonds

20-sim fue el primer paquete de software lanzado comercialmente para soportar el modelado de gráficos de bonos. La primera versión de 20-sim se lanzó en 1995. Desde entonces, un esfuerzo continuo para mejorar el modelado de gráficos de bonds ha convertido a 20-sim en el paquete de software número uno para el modelado de gráficos de bonds. Los gráficos de enlaces son una descripción similar a una red de sistemas físicos en términos de procesos físicos ideales. Con el método de gráfico de enlace, las características del sistema se dividen en un conjunto (imaginario) de elementos separados. Cada elemento describe un proceso físico idealizado. Para facilitar el dibujo de gráficos de enlaces, los elementos comunes se denotan con símbolos especiales como se ilustra en la figura 4.17.

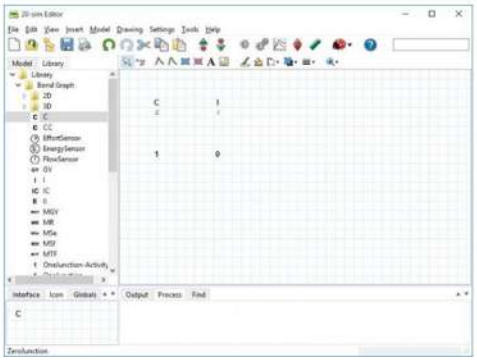


Figura 4.17. Diseño de gráfico de bonds en 20-Sim.

1. **Puertos y multipuertos**

La base del modelado de gráficos de bonos es el uso de puertos. 20-sim le permite crear modelos definidos por el usuario con un número arbitrario de puertos de alimentación y señales. El tamaño de los puertos puede ser 1 (predeterminado) o mayor (multipuertos). Para cada puerto, puede especificar la causalidad como preferencial fija, indiferente o según la causalidad de otros puertos.

2. **Causalidad**

Los trazos causales indican la dirección de los esfuerzos y los flujos en un modelo de gráfico de enlace. En 20-sim, sólo tiene que ingresar las ecuaciones en una de las posibles formas causales. Si se cambia la causalidad, las ecuaciones se reescriben automáticamente. 20-sim muestra trazos causales en color negro para la causalidad preferida y en trazos causales en color naranja para la causalidad no preferida. La causalidad de un modelo completo se deriva automáticamente, pero se puede cambiar manualmente.

3. **Lazos algebraicos y causalidad diferencial**

Los bucles algebraicos y las causalidades diferenciales se rastrean automáticamente. Si es posible, 20-sim reescribirá las ecuaciones simbólicamente para eliminar bucles algebraicos y causalidades diferenciales.

4. **Modelos a medida**

En 20-sim puedes crear tus propios modelos de gráficos de bonds y guardarlos en tu propia biblioteca de modelos. Los modelos pueden tener un número arbitrario de puertos, señales de entrada y salida. Se puede utilizar un editor de dibujos especializado para dar a los modelos cualquier tipo de representación, como se ilustra en la figura 4.18.

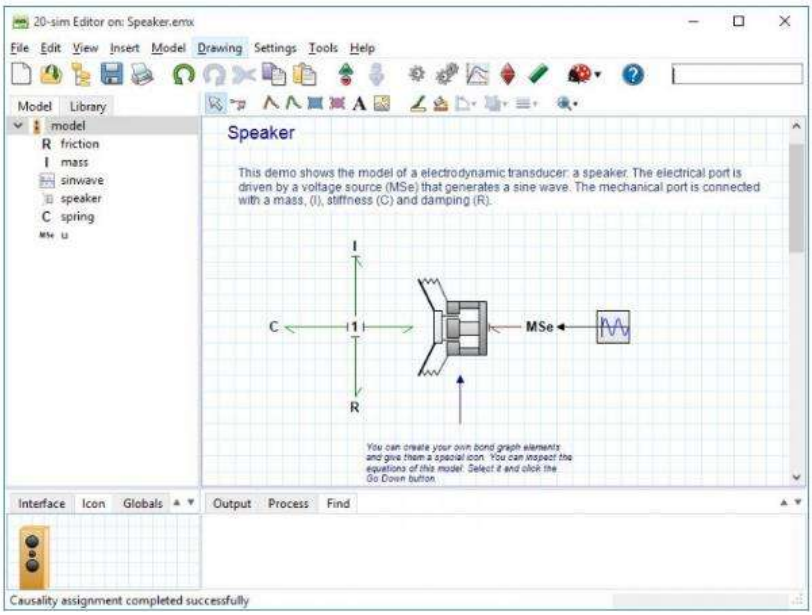


Figura 4.18. Diseño de modelos a medida en 20-Sim.

c) Espacio de Estado

Cualquier gráfico de enlace se puede transformar automáticamente en un modelo de espacio de estado lineal. El resultado se muestra en el Editor de sistemas lineales, donde puede mostrar el diagrama de Bode resultante o exportar el modelo de espacio de estado como un archivo M de Matlab o una función S. Si el modelo de gráfico de enlace contiene elementos no lineales, la transformación se realiza utilizando la linealización en un punto de trabajo definido por el usuario.

d) Diagramas icónicos

Los diagramas o componentes icónicos son los componentes básicos de los sistemas físicos. Le permiten ingresar modelos de sistemas físicos gráficamente, de manera similar a dibujar un esquema de ingeniería. En 20-sim está disponible una gran biblioteca de elementos de diagrama icónicos. Los elementos se muestran en el Editor mediante iconos que se parecen a las partes correspondientes del modelo físico ideal. Puede crear modelos arrastrando los elementos al Editor y haciendo las conexiones adecuadas entre los elementos.

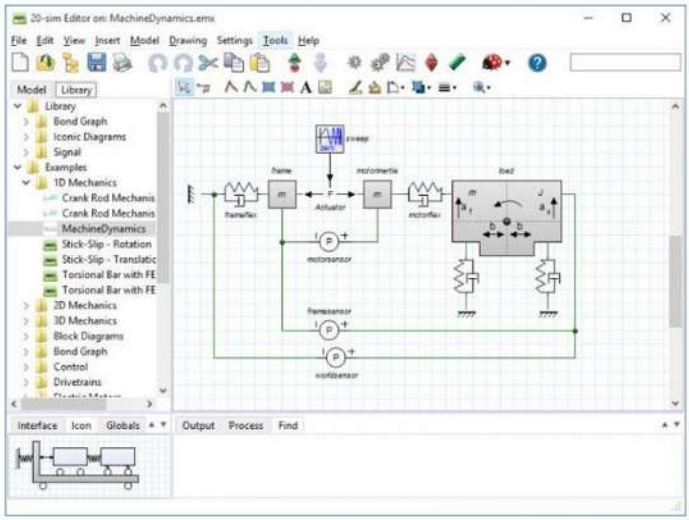


Figura 4.19. Diseño de diagramas icónico en 20 Sim.

Las secuencias de comandos le permiten ejecutar tareas en 20 sim automáticamente utilizando funciones de secuencias de comandos especializadas. Con estas funciones puede abrir modelos, ejecutar simulaciones, cambiar parámetros, procesar los resultados y mucho más.

Python es un lenguaje de programación de alto nivel de propósito general con énfasis en la legibilidad del código y los algoritmos de escritura en menos líneas de código que otros lenguajes de programación. Las funciones de scripting de 20 sim se pueden ejecutar desde un programa de

Python. Las secuencias de comandos de 20 sim se han probado con las siguientes versiones de Python: Python 2.7.x, Python 3.4.xy Python 3.5.x (32 bits y 64 bits).

Con la instalación de 20-sim, puede instalar opcionalmente el paquete Python 3.4, como se muestra en la figura 4.20. Esta instalación proporciona suficiente soporte para comenzar con las secuencias de comandos de 20 sim. Sin embargo, no proporciona un IDE de desarrollo o un amplio conjunto de bibliotecas científicas y matemáticas.

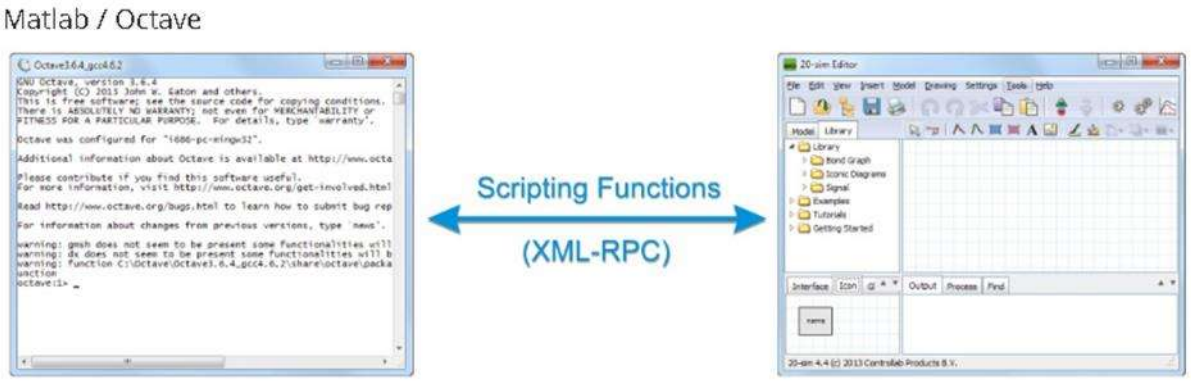


Figura 4.20. Pasar de Matlab/Octave a 20 Sim.

Las funciones de scripting de 20 sim se pueden ejecutar como archivos m en Matlab o GNU Octave. GNU Octave es un entorno de código abierto que le permite ejecutar archivos m similares a Matlab. Si no tiene una licencia válida de Matlab, GNU Octave será una alternativa útil. Todas las funciones de Octave se pueden encontrar aquí.

4.10 Instalación de 20-Sim

1- ingresamos al buscador (google), ingresamos 20-Sim.

2- Nos aparecerá como se ilustra en la figura 4.21.

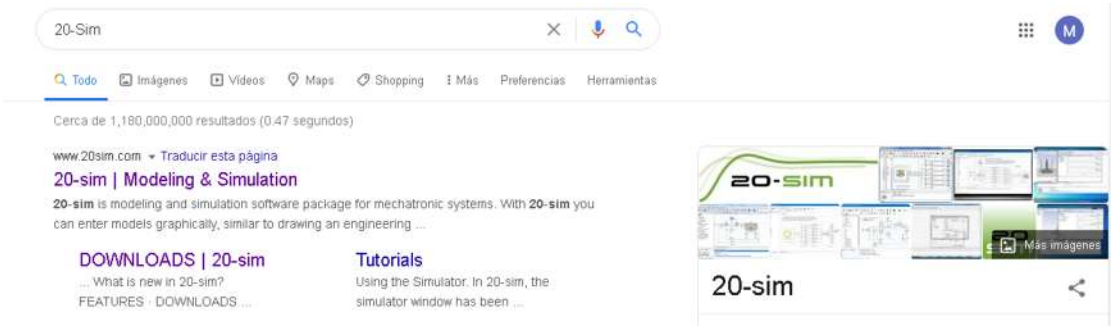


Figura 4.21. Búsqueda en google de 20 sim.

3- Observemos que sea de la página oficial de 20-Sim. Nos aparece un resultado le damos clik.

4- Nos aparecerá como se muestra en la figura 4.22.

Figura 4.22. Página oficial de 20-Sim.

5- Nos vamos donde se tienen descargas de acuerdo a la figura 4.23.

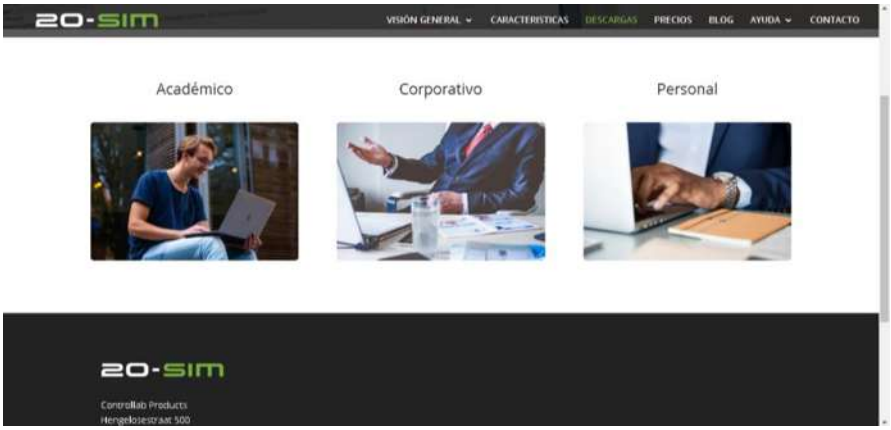


Figura 4.23. Selección del uso de 20 Sim.

6- Seleccionamos el Academico, como la figura 4.24.

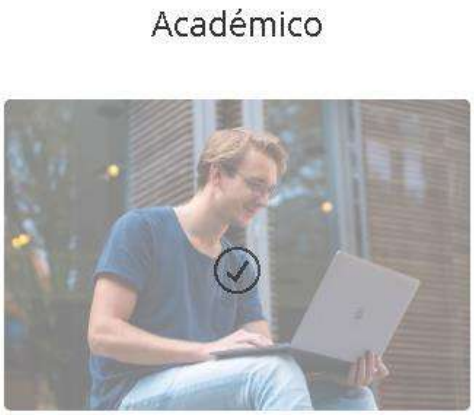


Figura 4.24. Selección del uso académico de 20-Sim.

7- Rellenamos los recuadros con la información que nos solicita de acuerdo a la figura 4.25.

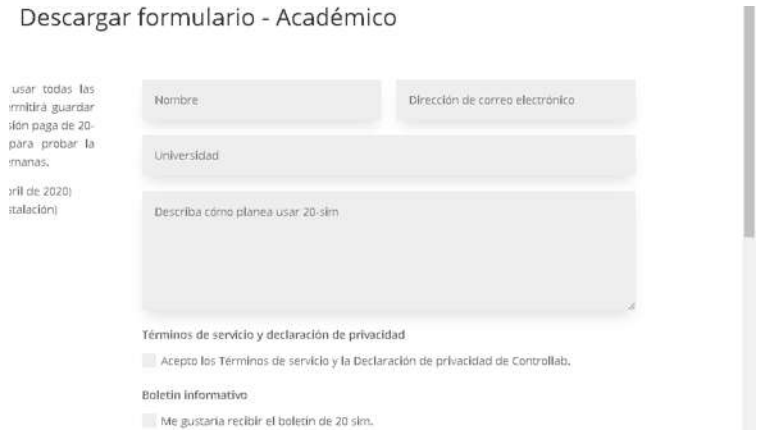


Figura 4.25. Formulario de 20-Sim, para la descarga.

8- Marcamos, Acepto los Términos de servicio y la Declaración de privacidad de Controllab. Es opcional si queremos recibir el boletín de 20 sim y le damos en descargar como la figura 4.26.

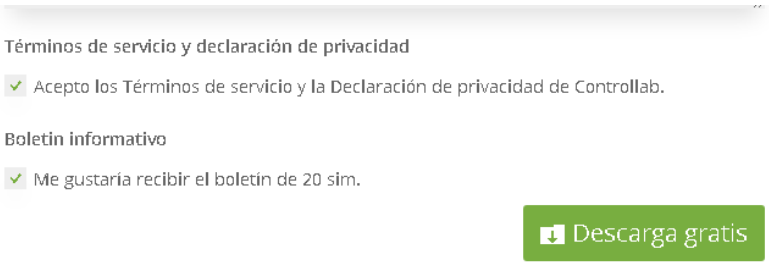


Figura 4.26. Descarga de 20- Sim.

9- Lo ejecutamos como administrador y le damos permisos como la figura 4.27.

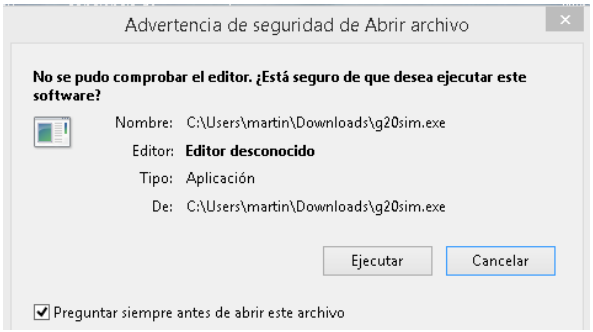


Figura 4.27. Advertencia de seguridad.

10- Nos aparecerá la figura 4.28 y le damos en next (siguiente).

Figura 4.28. Página inicial.

11- Nos parecerá otra ventana como la figura 4.29 y 4.30. Oprimimos I Agree.

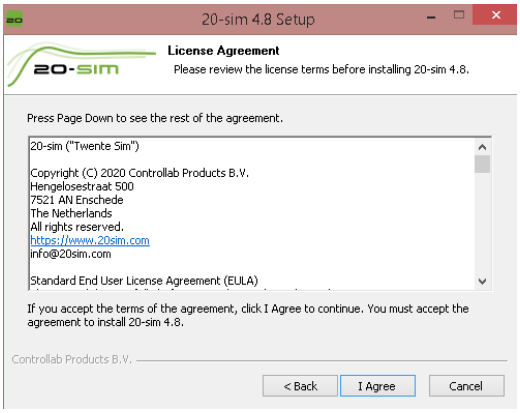


Figura 4.29. Consentimiento para la instalación.

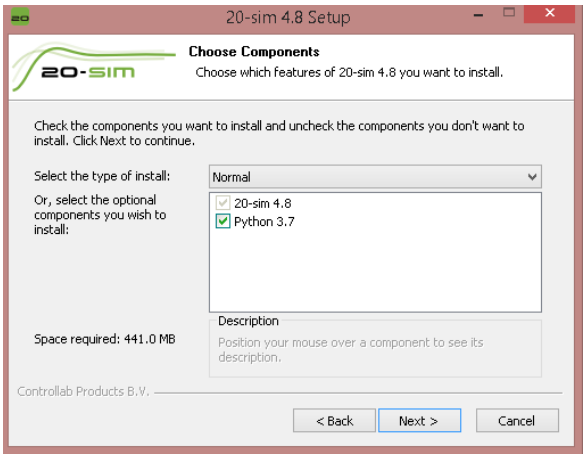


Figura 4.30. Archivos descargados.

12- Seleccionamos la ruta de la carpeta donde se instalará el software, o lo podemos dejar como lo selecciono el software, como la figura 4.31. Y oprimimos Next.

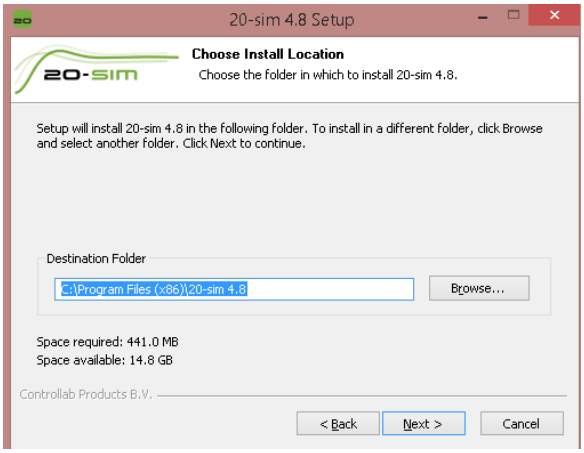


Figura 4.31. Ubicación de archivos.

13- Aparecerá esta nueva ventana de acuerdo a la figura 4.32. Marcamos el recuadro donde dice Don not créate shortcuts o puede ser opcional, oprimimos Install.

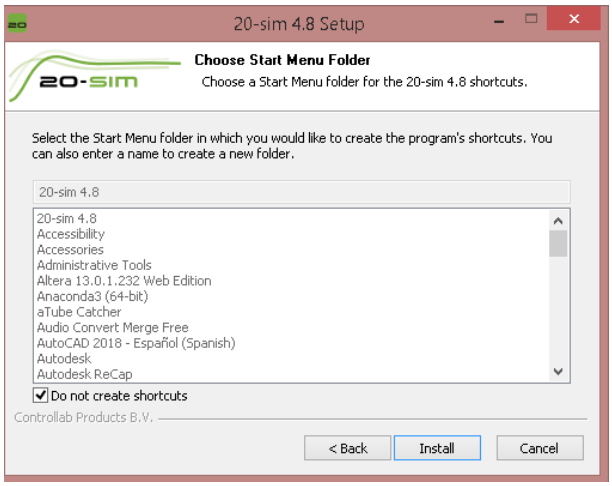


Figura 4.32. Selección de archivos.

14- Se comenzará a instalar de acuerdo a la figura 4.33.

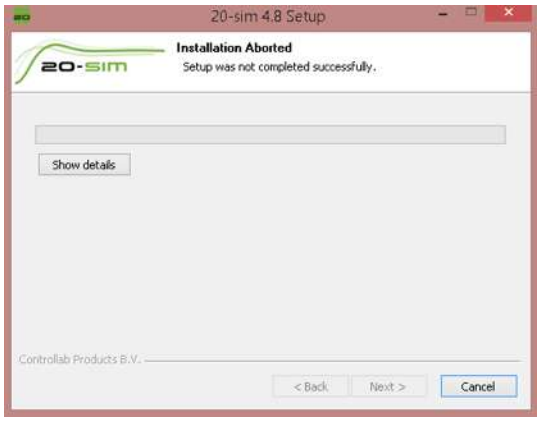


Figura 4.33. Finalización de descarga.

15- Por último lo abrimos el software 20-Sim instalado.

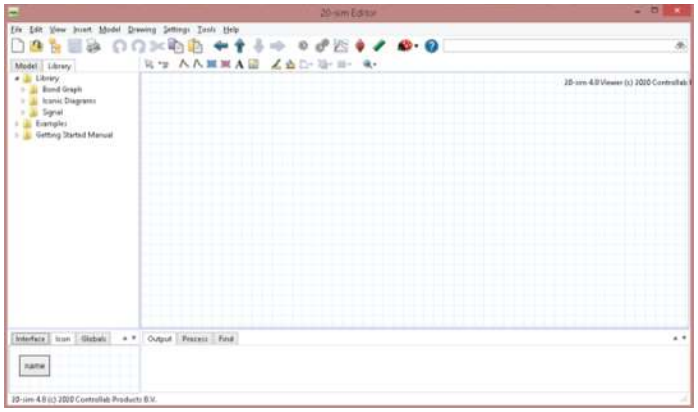


Figura 4.34. 20-Sim funcionando.

Capítulo 5. Simulaciones de Sistemas Discretos

5.1 Primer caso de Estudio

Dado la ecuación en diferencias, se realiza la simulación en 20-Sim, la cual es:

$$y(k) - 0.9y(k-1) = x(k)$$

Aplicando transformada Z, se obtiene

$$Y(Z) - 0.9Z^{-1}Y(Z) = X(Z)$$

factorizando

$$\frac{Y(Z)}{X(Z)} = \frac{1}{Z - 0.9}$$

ante una entrada escalón unitario

$$X(Z) = \frac{Z}{Z-1}$$

sustituyendo

$$Y(Z) = \frac{1}{Z - 0.9}\frac{Z}{Z-1}$$

Aplicando transformada inversa de Z

$$\frac{Y(Z)}{Z} = \frac{1}{(Z-0.9)(Z-1)}$$

la expansión en fracciones parciales

$$Y(Z) = \frac{-10Z}{Z - 0.9} + \frac{10Z}{Z - 1}$$

Por lo tanto, la función en tiempo discreto es

$$y(k) = -10(0.9)^k + 10$$

es decir,

$$y(k) = 10(1 - 0.9^k)u(k)$$

La edición de la función de transferencia en el dominio Z, se introduce en 20-Sim, como se muestra en la figura 5.1.

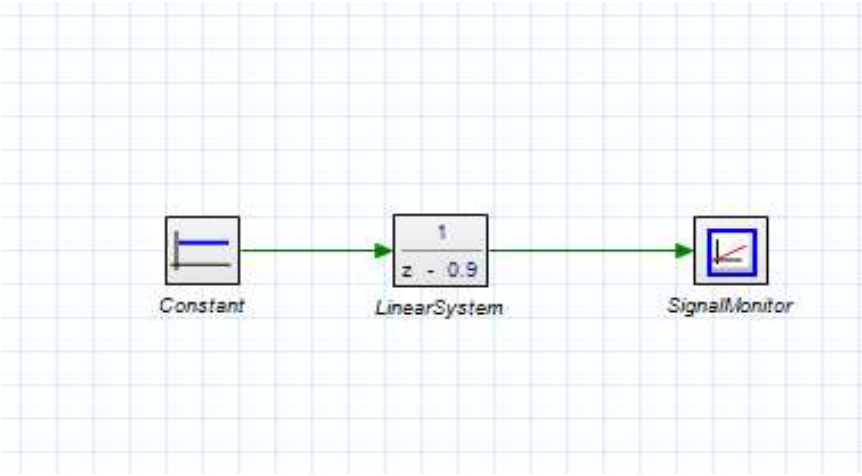


Figura 5.1. Caso de estudio 1 en 20-Sim.

La simulación del sistema descrito en la figura 5.1 se describe en la figura 5.2, con una frecuencia de muestreo de Fs=1Hz.

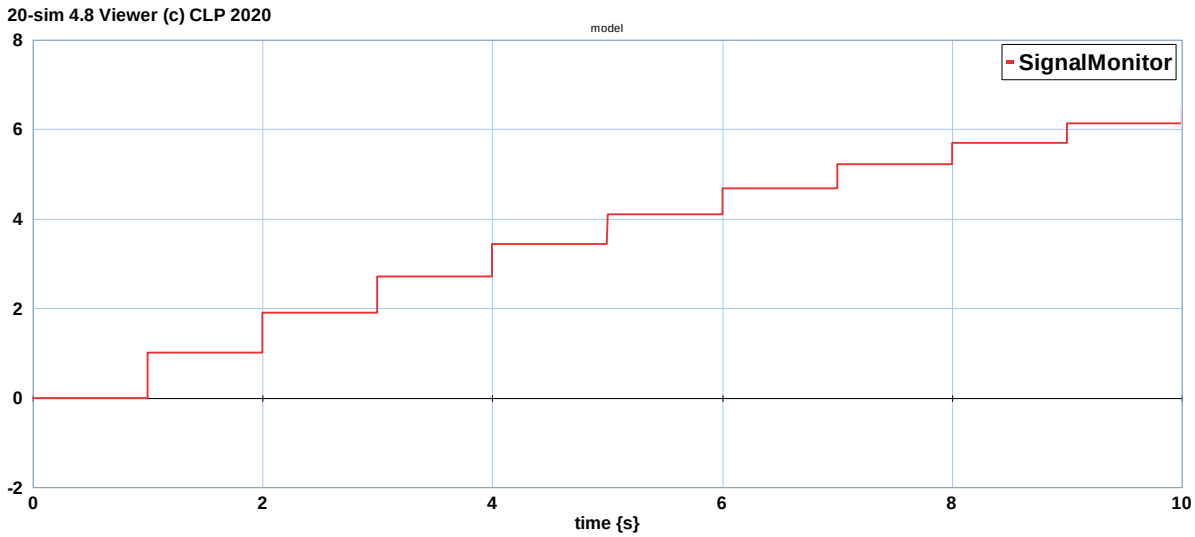


Figura 5.2. Simulación del sistema de la figura 5.1.a Fs=1Hz

Aumentando la frecuencia de muestreo a Fs=10Hz se obtiene la figura 5.3.

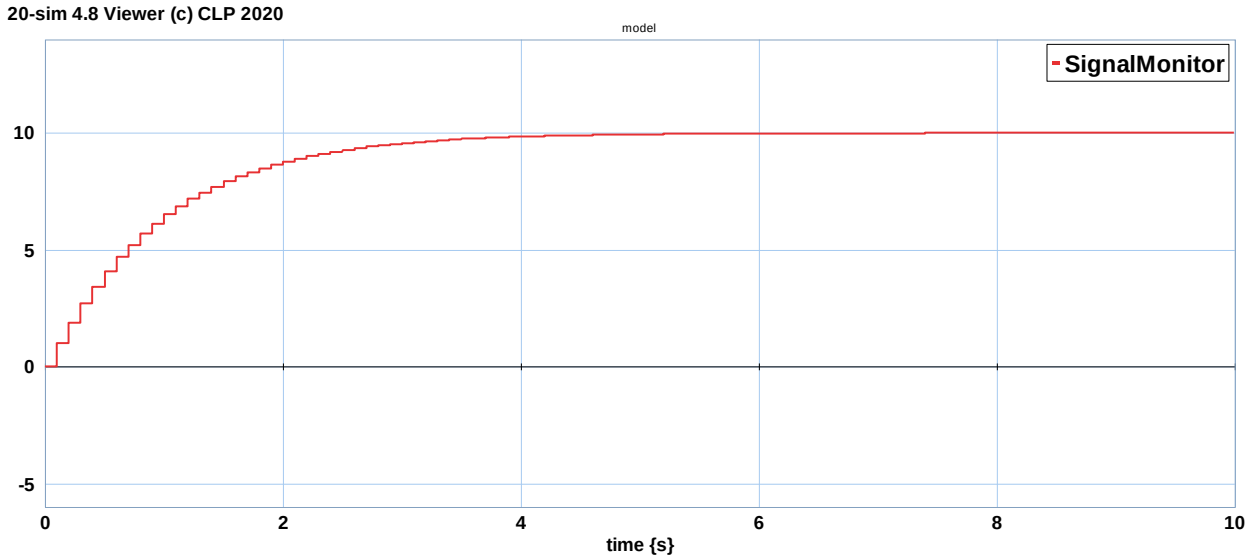


Figura 5.3. Simulación del sistema de la figura 5.1.a Fs=10Hz

A una frecuencia de muestreo de Fs=50Hz el resultado se muestra en la figura 5.4.

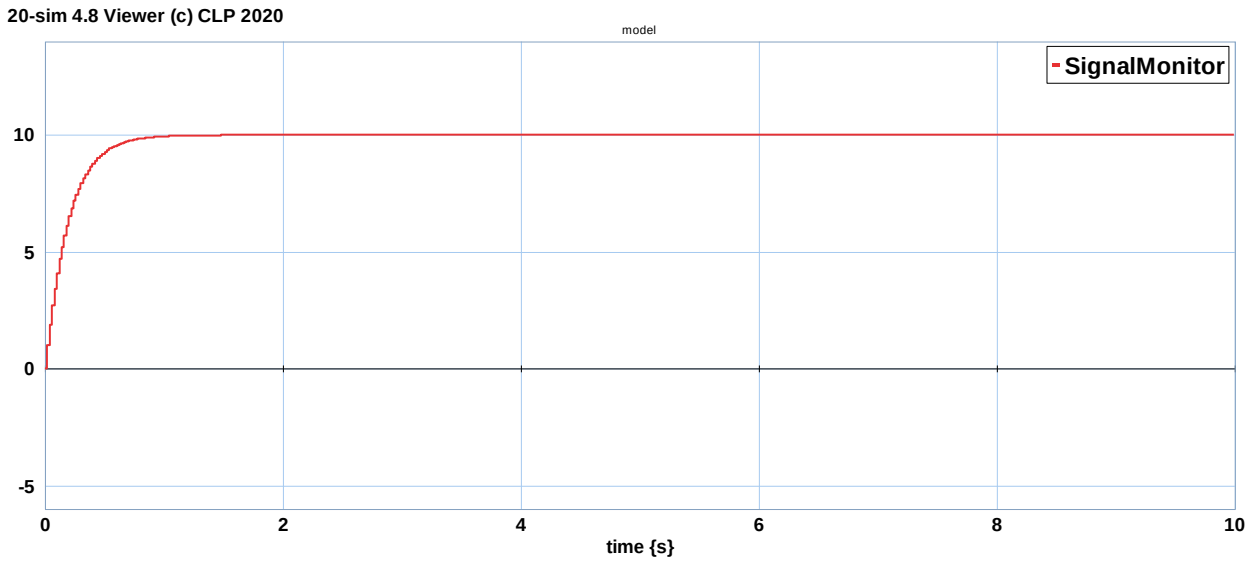


Figura 5.4. Simulación del sistema de la figura 5.1.a Fs=50Hz

Notar que la respuesta en estado estacionario debe alcanzar la magnitud de 10, de acuerdo al análisis realizado anteriormente.

Si se desea tener este sistema en lazo cerrado, la figura 5.5 muestra este sistema.

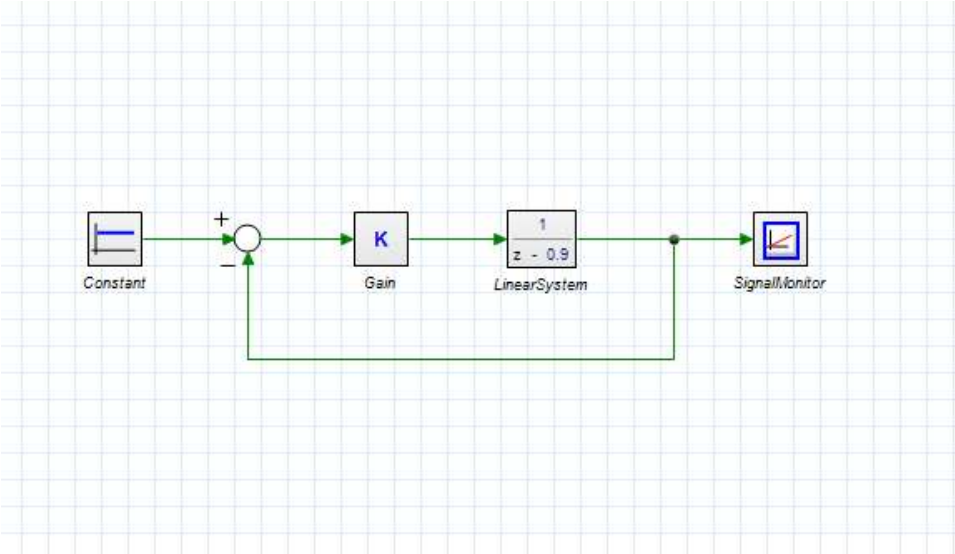


Figura 5.5. Sistema en lazo cerrado.

La función de transferencia de la figura 5.5 está dada por

$$\frac{Y(Z)}{x(Z)} = \frac{\frac{K}{Z-0.9}}{1+\frac{K}{Z-0.9}} = \frac{K}{Z-0.9+K}$$

Nuevamente si la entrada es una señal escalón unitario, la salida del sistema en el dominio Z es

$$Y(Z) = \frac{K}{Z-0.9+K}\frac{Z}{Z-1}$$

Aplicando fracciones parciales

$$\frac{Y(Z)}{Z} = \frac{K}{K+0.1}\frac{1}{Z-1} - \frac{K}{K+0.1}\frac{1}{Z-0.9+K}$$

Reduciendo

$$Y(Z) = \frac{K}{K+0.1}\left[\frac{Z}{Z-1} - \frac{Z}{Z-0.9+K}\right]$$

Mediante la transformada inversa de Z, se obtiene la salida del sistema en lazo cerrado en tiempo discreto descrita por

$$y(k) = \left(\frac{K}{K+0.1}\right)[1-(-0.9+K)^k]u(k)$$

Considerando una ganancia de K=0.45, la respuesta en tiempo discreto para este sistema se muestra en la figura 5.6.

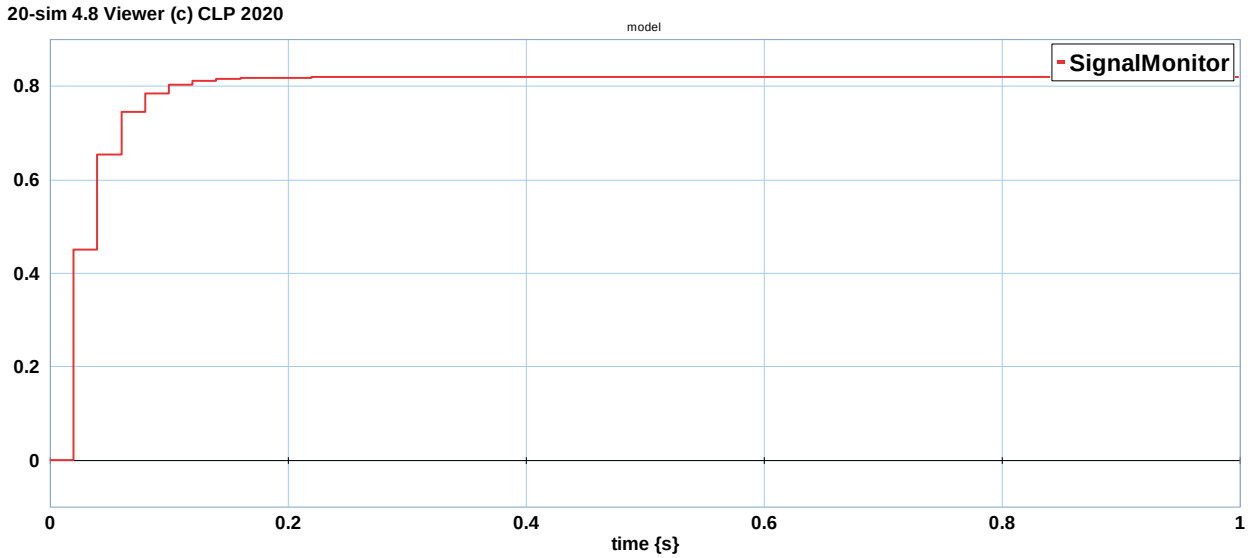


Figura 5.6. Respuesta del sistema en lazo cerrado con K=0.45.

Sustituyendo valores

$$y(k) = 0.8181[1-(-0.45)^k]u(k)$$

Siendo su valor de estado estacionario

$$y_{ss} = 0.8181$$

Lo cual se comprueba en la gráfica de la figura 5.6.

Finalmente, si la ganancia se ajusta a K=0.8, la salida está dada por

$$y(k) = 0.888[1 - (-0.1)^k]u(k)$$

Y el valor de estado estacionario es

$$y_{ss} = 0.88$$

La gráfica correspondiente a esta ganancia se muestra en la figura 5.7.

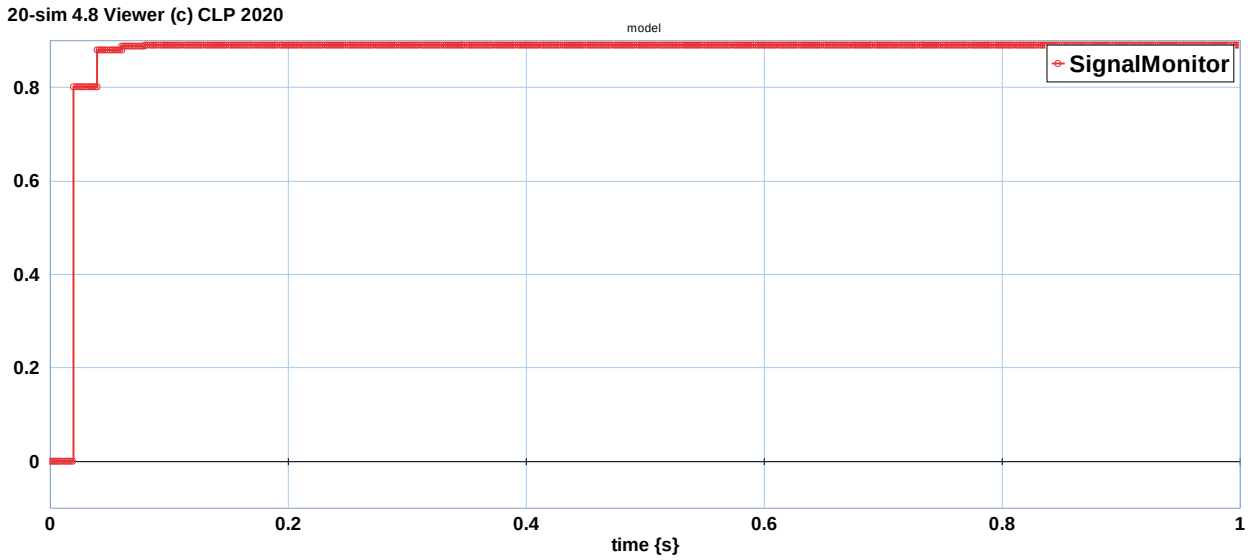


Figura 5.7. Respuesta del sistema en lazo cerrado con K=0.80.

5.2 Segundo Caso de Estudio

A continuación se describe una ecuación de diferencias que representa un sistema de segundo orden dada por

$$y(k) - 0.25y(k-1) - 0.10y(k-2) = x(k)$$

Aplicando transformada Z

$$Y(Z) - 0.25Z^{-1}Y(Z) - 0.10Y(Z) = X(Z)$$

$re-$escribiendo

$$Y(Z) = \frac{Z^2}{Z^2 - 0.25Z - 0.10}X(Z)$$

Ante una entrada escalón unitario

$$X(Z) = \frac{Z}{Z-1}$$

Entonces el sistema se puede escribir de la forma

$$\frac{Y(Z)}{Z} = \frac{Z^2}{(Z-1)(Z-0.465)(Z+0.215)}$$

Utilizando fracciones parciales resulta

$$Y(Z) = 2.8571\frac{Z}{Z-1} - 0.3257\frac{Z}{Z-0.465} + 0.0878\frac{Z}{Z+0.215}$$

Aplicando transformada inversa de Z, la solución en el tiempo descrito es

$$y(k) = [2.8571 - 0.3257(0.465)^k + 0.0.0878(-0.215)^k]u(k)$$

El modelo en 20-sim de este caso de estudio se ilustra en la figura 5.8.

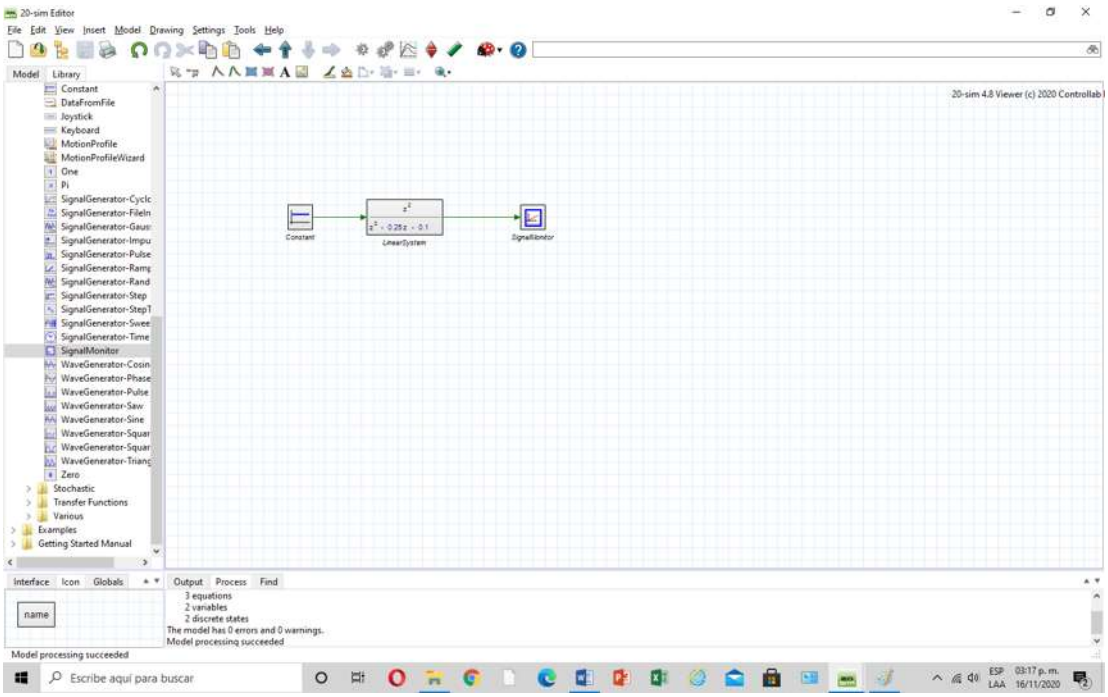


Figura 5.8. Editor en 20-sim del segundo caso de estudio.

El comportamiento de este sistema a una frecuencia de muestreo de Fs=1Hz se ilustra en la figura 5.9.

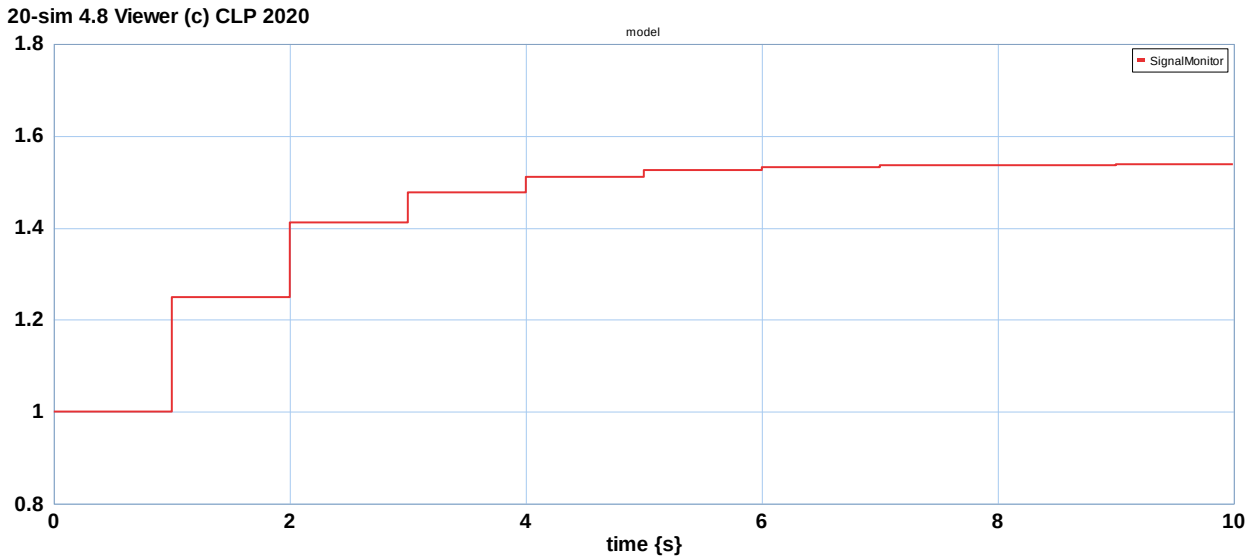


Figura 5.9. Respuesta del sistema del segundo caso de estudio a Fs=1Hz.

Aumentando la frecuencia de muestreo a Fs=10Hz se obtiene la gráfica de su comportamiento de la figura 5.10.

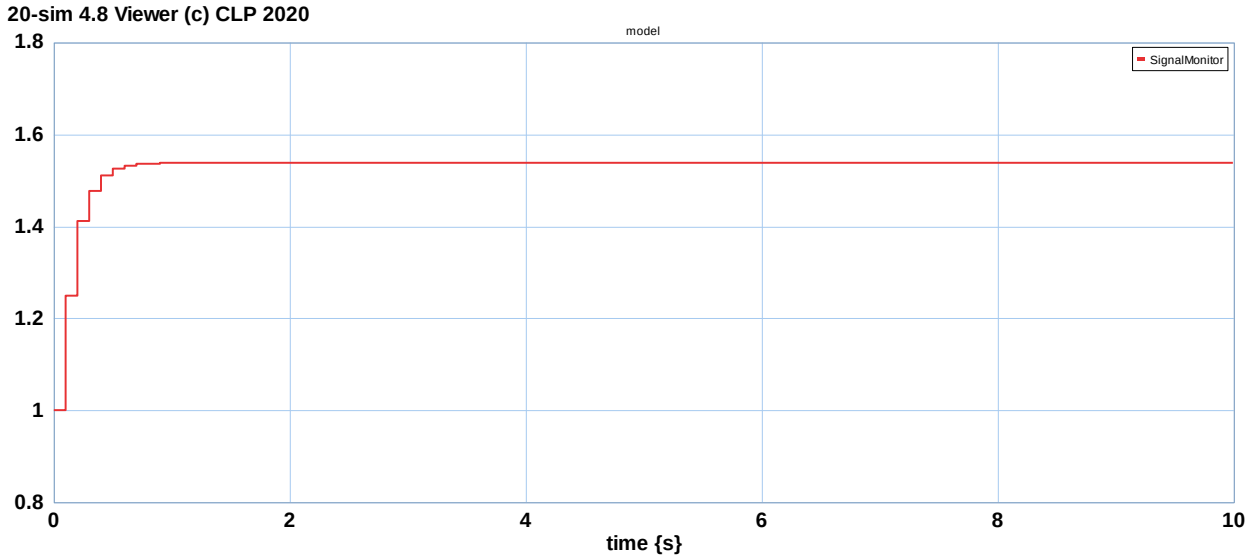


Figura 5.10 Respuesta del sistema del segundo caso de estudio a Fs=10Hz.

Finalmente, a una frecuencia de muestro de Fs=100Hz, resulta el comportamiento muestreado de la figura 5.11.

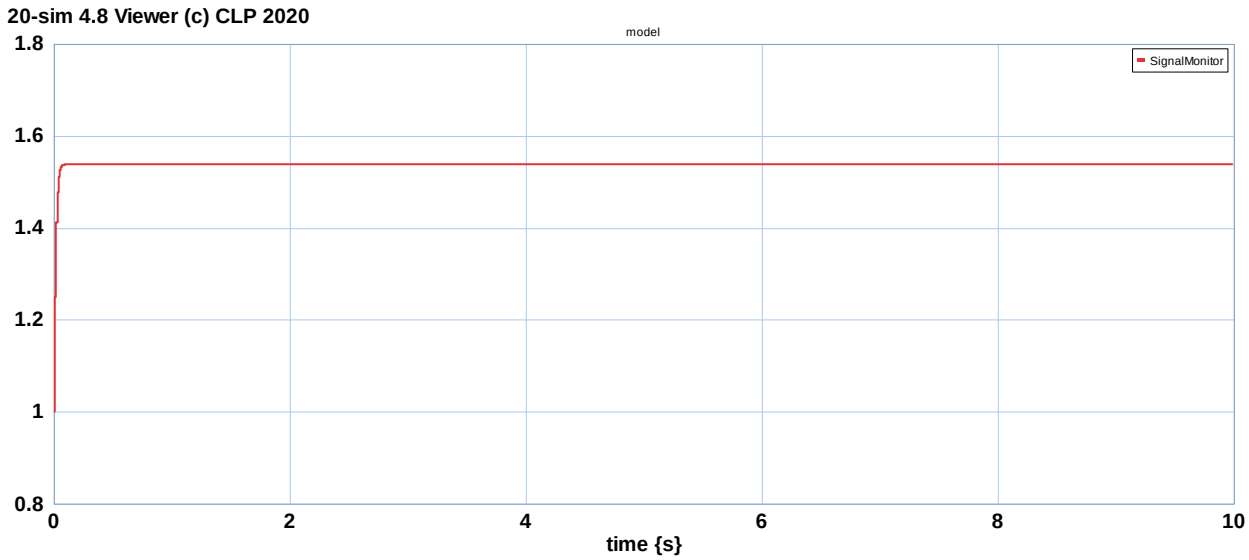


Figura 5.11. Respuesta del sistema del segundo caso de estudio a Fs=100Hz.

Si a este caso de estudio, se introduce una retroalimentación unitaria, se tiene el sistema en lazo cerrado que se ilustra en la figura 5.12.

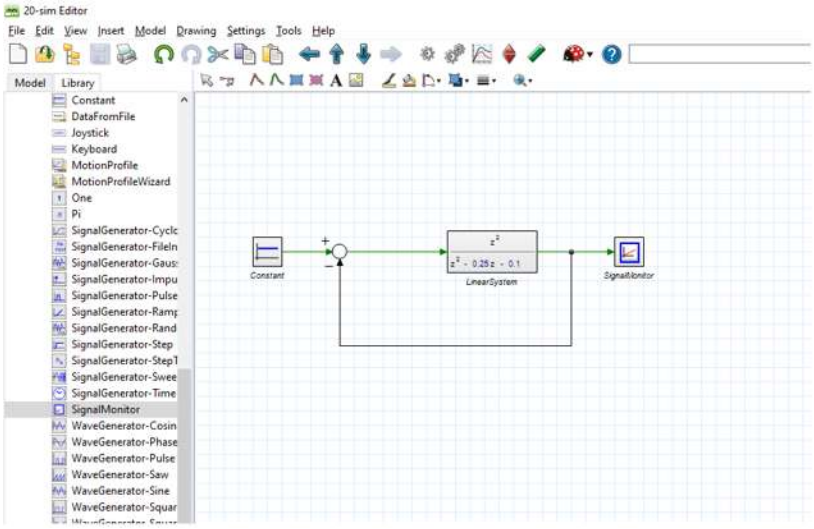


Figura 5.12. Edición del segundo caso de estudio con retroalimentación unitaria.

El comportamiento de este sistema a una frecuencia de muestreo de Fs=10Hz, se describe en la figura 5.13.

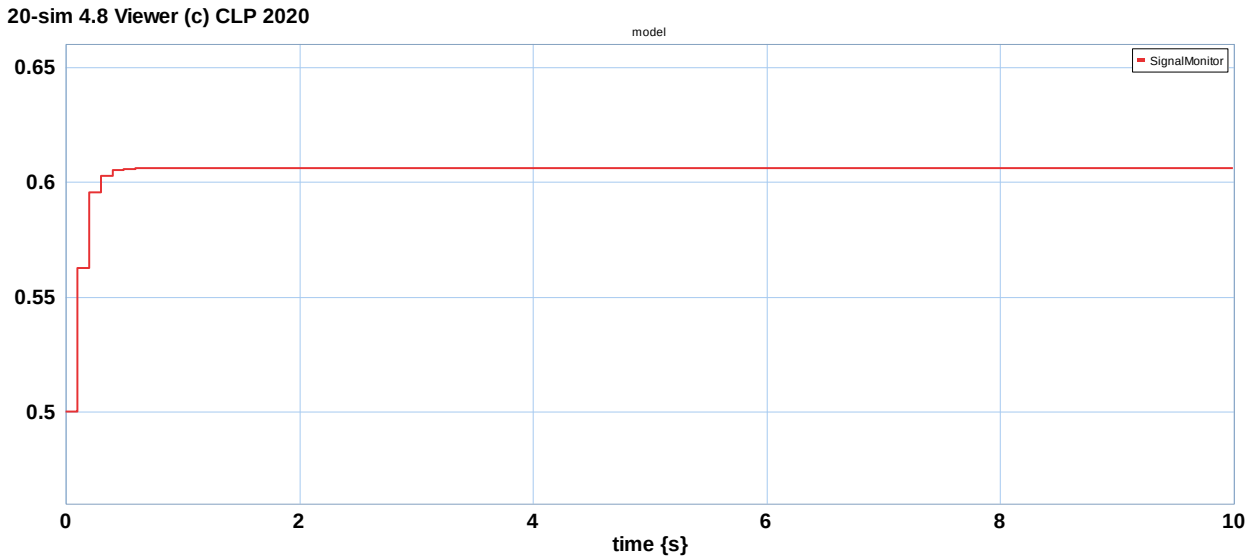


Figura 5.13. Respuesta del sistema del segundo caso de estudio con retroalimentación unitaria a Fs=10Hz.

Ahora, introduciendo un bloque de ganancia dentro del lazo de retroalimentación, se obtiene el sistema que se muestra en la figura 5.14.

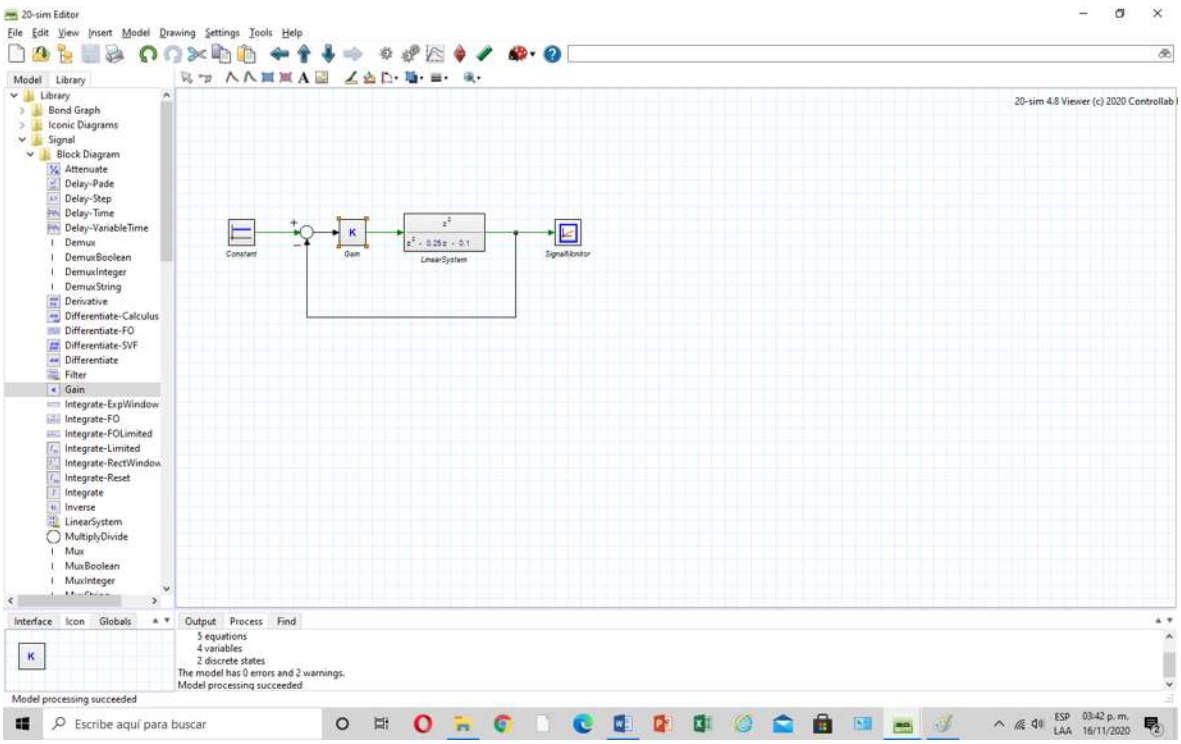


Figura 5.14. Edición del segundo caso de estudio con ganancia.

Si la ganancia es de K=10 a una frecuencia de muestreo de Fs=10Hz, se obtiene la respuesta de la figura 5.15.

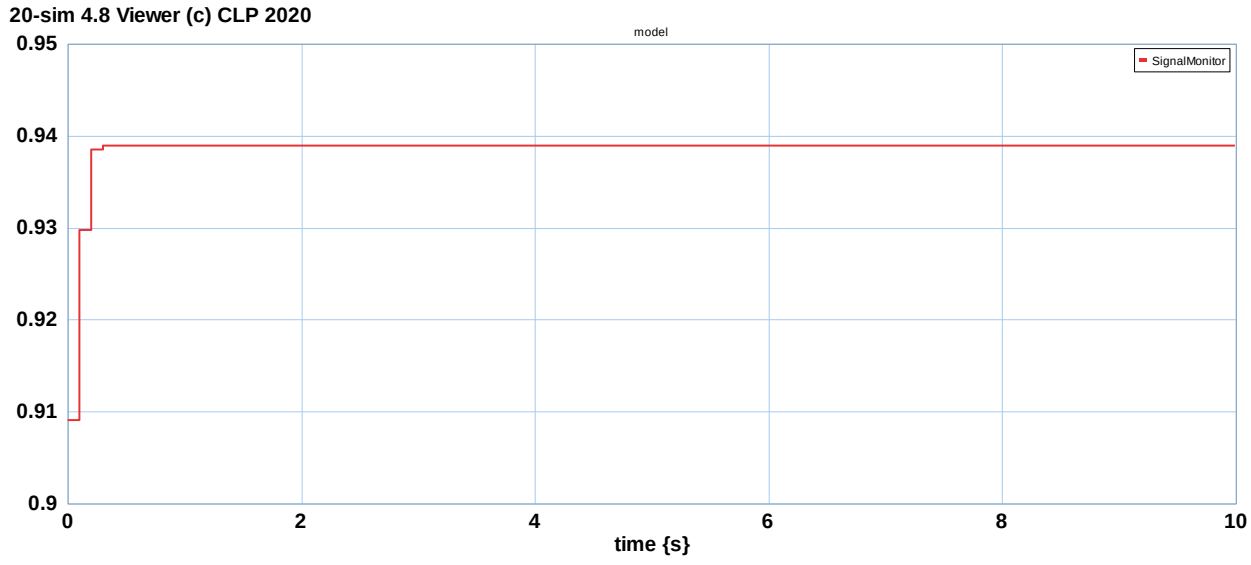


Figura 5.15. Respuesta del sistema en lazo cerrado, K=10 y Fs=10Hz

Aumentando la ganancia a K=100 y Fs=10hz la respuesta se observa en la figura 5.16.

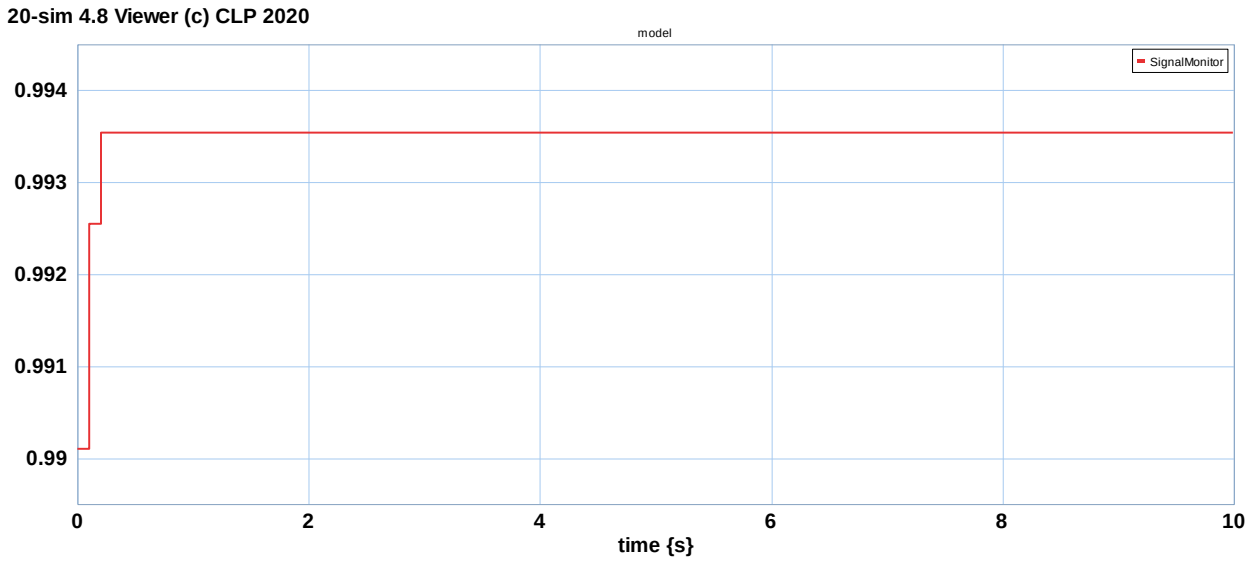


Figura 5.16. Respuesta del sistema a K=100 y Fs=10Hz.

Finalmente, ante una ganancia de K=500 y Fs=100Hz se tiene el comportamiento que se ilustra en la figura 5.17.

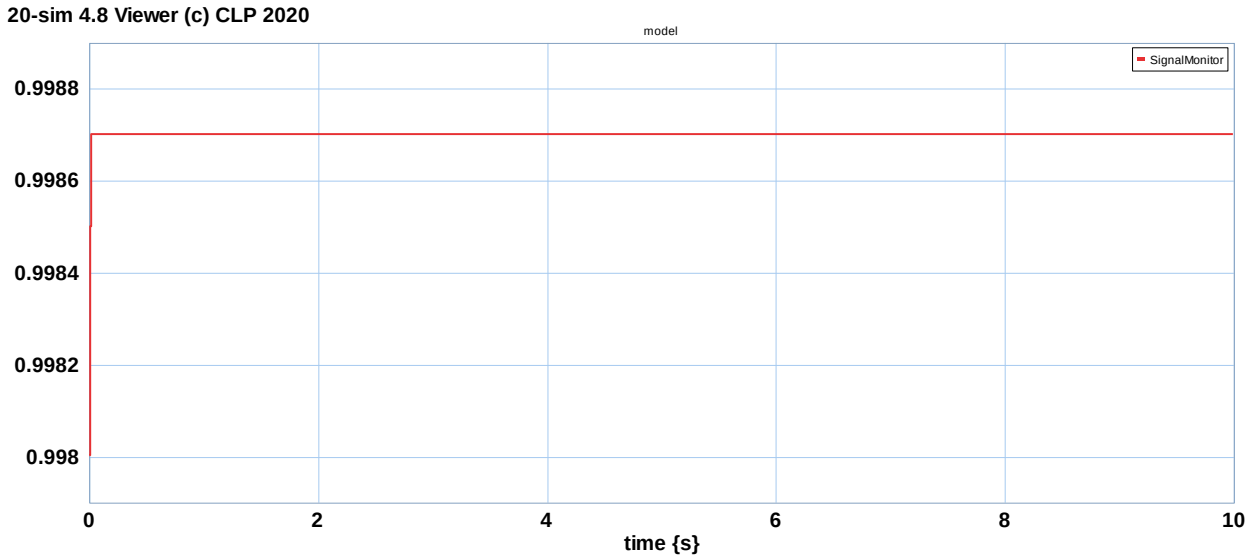


Figura 5.17. Respuesta del sistema a K=500 y Fs=100Hz.

Capítulo 6. Conclusiones y Recomendaciones

6.1 Conclusiones

Los sistemas discretos han sido un desafío en el modelado y simulación, ya que se consideran sistemas que no son fácilmente manejables. También, las herramientas matemáticas en su modelado, análisis y control genera dificultades a los estudiantes de ingeniería.

Por lo tanto, en este trabajo de tesis, se presenta algunas de las herramientas del modelado de sistemas en tiempo discreto y su aplicación en el software 20-sim, demostrando que es un software de fácil uso e instalación, que puede ser utilizado en lazo abierto y cerrado.

Las ventajas de la simulación de sistemas son evidentes, ya que se puede realizar múltiples simulaciones, para obtener los mejores resultados esperados. El cambio de funciones de transferencia en el dominio Z en 20-sim, representa una herramienta bastante versátil.

La simulación puede ser en lazo abierto y cerrado, con una o varias ganancias.

Así mismo, se puede cambiar fácilmente la frecuencia de muestreo mostrando cual podría ser la mejor opción en un caso de implementación real del sistema.

Por lo tanto, se obtuvieron satisfactoriamente los objetivos planteados en este trabajo de tesis, el cuál puede ser útil para los estudiantes de licenciatura, como un trabajo básico de modelado y simulación y que puede ser aplicado a casos de estudio más complejos.

6.2 Recomendaciones

Este trabajo de tesis es básico, por lo que puede ser fácilmente extendido a sistemas con varias funciones de transferencia y varias retroalimentaciones.

Se puede aplicar controladores en el dominio discreto P, PI o PID que se encuentran en la librería o bien diseñar estos controladores con las teorías bien desarrolladas en control digital.

Bibliografía

1. https://www.monografias.com/trabajos82/lenguaje-uml-importancia-modelar/lenguaje-uml-importancia-modelar.shtml
2. Notas de los modelos de simulación:una herramienta multidisciplinar de investigación
3. Los modelos de simulación: una herramienta multidisciplinar de investigación
4. Notas Software para el aprendizaje de las técnicas de modelado y simulación
5. Notas 5163-3. Las herramientas de modelado y simulación para la identificación y solución de problemas aeroportuarios.
6. Notas Herramientas Software para el Diseño y Simulación en Aplicaciones de Control e Instrumentación Electrónica.
7. https://es.wikipedia.org/wiki/Simulaci%C3%B3n_por_computadora
8. https://www.fib.upc.edu/retro-informatica/avui/simulacio.html
9. Notas_Control_Digital1_Casi_Completas
10. www.20sim4c.com.