UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

"SIMULACIÓN DE SISTEMAS CONTINUOS EN SIMNON"

TESIS

presenta

GERARDO ANGEL PALOMINO CISNEROS

Que, para obtener el grado de

INGENIERO ELÉCTRICISTA

DR. GILBERTO GONZÁLES ÁVALOS

Director de Tesis

Morelia Michoacán, Diciembre 2019

Agradecimientos

- A mis padres: José Dolores Palomino Pérez y Ana María Cisneros Pimentel por el cariño, amor, apoyo y enseñanzas que siempre me han brindado, porque gracias a ellos es que eh llegado hasta donde estoy ahora, porque gracias a ellos logre mis objetivos. Y porque este logro no es solo mío sino también de ellos, no me queda otra cosas más que decir que ¡GRACIAS!

- A Mis hermanas: Sonia Judith Palomino Cisneros y Ana Karina Palomino Cisneros por su apoyo y sus enseñanzas que a pesar de las dificultades que se llegaran a presentar siempre saben resolverlo de la manera más sencilla.

- A mis tíos: María Martha Cisneros Pimentel, José Jesús García Ruiz, Margarita Cisneros Pimentel y Jaime Moreno Por sus consejos y apoyo a través de mi vida y que además ellos siempre fueron unos segundos padres.

- A mis primos: Jorge Moreno, Salvador Moreno, Jaime Moreno, Brian Cisneros y Raúl Cisneros por tantas aventuras y locuras que eh vivido junto a ellos.

- A mi asesor de tesis: Gilberto Gonzáles Ávalos por su tiempo que invirtió en mí trabajo, puesto que el tiempo es algo único y especial y es lo más valioso que tiene el ser humano. Por el apoyo en la elaboración de la tesis. Por su amistad, por los puntos de vista y consejos que me brindo que sin su ayuda no se habría realizo de forma exitosa este trabajo.

- A mis amigos y compañeros: Víctor Manuel Ortiz Mercado, Enrique Rodríguez, Erick Vázquez, Danton Eduardo, Roberto López, Andrés Vargas, Alberto Rivera, Cristian, Israel Calderón, Fernando Yañes, Jesús Moreno, Mayra, Pablo, Manuel Ferreyra, Edson Alejandro, Oznar Hernández, Jorge Quevedo, Irineo Hazael, Sergio Alberto, Jesús Lucas y todos los compañeros con los que conviví en mi estancia por el apoyo, el tiempo que pase junto a ellos, por tantas aventuras y locuras que hemos hecho, convivios, buenos y malos momentos que pasamos juntos, porque a pesar de las dificultades que enfrentamos en la carrera de una forma u otra lo supimos sobrellevar por eso y muchas cosas más...

- A mis Profesores: A todos mis profesores que me apoyaron, que me motivaron, que me brindaron una parte de su conocimiento, que me orientaron, y que incluso algunos al final me brindaron su amistad. En especial al Ing. Víctor Quintero Rojas, a La Dr. Nandini Barbosa Cendejas, al Fis. José Juárez Palafox, Heriberto Luna Moreno, Gustavo Saucedo Zavala, al Ing. Rodrigo Guzmán, a la MI. Gabriela Barrera Díaz, al Dr. Rafael Cisneros Magaña.

- A los ingenieros: Daniel Cervantes Montero y Juan Carlos Renteria Ruiz, por el apoyo, enseñanzas y su amistad durante el tiempo de trabajo que convivimos.

- A la Universidad Michoacana de San Nicolás de Hidalgo y a la facultad de Ingeniería Eléctrica que me acogieron durante tanto tiempo.

Dedicatorias

Esta tesis va dedicada a mis padres. porque sin su apoyo sus consejos y cariño dudo que hubiera llegado tan lejos.

Citado especial

[Si un animal o un ser humano concentran
Toda su atención y su voluntad en una cosa determinada,
La consigue. Ese es todo el misterio.]- Demian. Herman Hesse

Resumen

El presente trabajo de tesis a nivel licenciatura trata sobre el software SIMNON para la simulación de sistemas continuos. Es importante que un ingeniero de cualquier tipo no se limite al resolver cualquier problema, y cuando se presenta un problema donde se incluyen sistemas se tiene que buscar alternativas para su solución con algún software en caso que los sistemas sean muy grandes, muy pequeños, muy complejos, muy tardados en el tiempo o de respuesta casi inmediata y/o además que sean sistemas muy costosos. Si se encuentra con el caso de alguno de estos problemas una de las alternativas más económicas y rápidas son las simulaciones. Y prácticamente cualquier sistema que se pueda representar de forma matemática se puede simular en SIMNON. Pero muchas ocasiones no conocemos que tipo de software implementar y no por el hecho que no se sepa utilizar o que sea muy complejo, si no por el hecho que no es muy conocido por la falta de información, o a fin porque simplemente no es tan comercial como otros, cabe resaltar que en la Facultad De Ingeniería Eléctrica de la Universidad Michoacana De San Nicolás de Hidalgo los profesores no limitan a los estudiantes en ese aspecto puesto que impulsan a utilizar software mas allá que los convencionales. Y partiendo de ahí los alumnos tienen la opción de utilizar una amplia variedad tanto de softwares convencionales o utilizar alguna otra herramienta, todo depende de que tan cómodos se sientan con ello.
Siempre es sabido que cuando alguien está motivado es más fácil que salgan las cosas bien, además mantener el interés en hacer lo que se está proponiendo, y una forma que sirve como motivación es saber qué es lo que pasa cuando se resuelve un problema, porque no siempre se llega a la solución que uno espera, pero se llega a una solución la cual hace que llegues a la satisfacción o que te reformules lo que estas planteando.

SIMNON es una herramienta tan útil para la simulación de procesos químicos como las centrales eléctricas. Puede usarse para algoritmos de control complejos, para modelos financieros, dinámica de robots. Todos los sistemas, que se pueden definir en términos matemáticos, también se pueden simular en SIMNON.
Miles de personas en todo el mundo están utilizando SIMNON como una herramienta eficiente para simular procesos y productos. Las universidades y centros de investigación en más de 40 países han encontrado que SIMNON es un gran programa para la simulación interactiva.

Palabras clave

Modelado

Gráfica

Software

Comandos

Parametros

Abstract

This thesis work at the bachelor level talks about the SIMNON tool for the simulation of continuous systems. It is important that an engineer of any kind is not limited to solving any problem, and when a problem arises where systems are included, you have to look for some support software in case the systems are very large, very small, very complex, very delayed in time or almost immediate response and / or also that they are very expensive systems. If you encounter any of these problems, one of the cheapest and fastest alternatives is the simulations. And Virtually any system that can be represented mathematically can be simulated in SIMNON. But many times we do not know what kind of software to implement and not because of the fact that it is not known to use or that is very complex, but because of the fact that it is not well known for the lack of information, or finally because it is simply not as commercial Like others, it should be noted that at the Michoacana University of San Nicolás de Hidalgo in the Faculty of Electrical Engineering, teachers do not limit students in that aspect since they encourage the use of software beyond conventional ones. And from there the students have the option of using the conventional ones or using some other tool, it all depends on how comfortable they feel with it.
It is always known that when someone is motivated it is easier for things to go well, in addition to maintaining interest in doing what is being proposed, and one way that serves as motivation is to know what happens when a problem is resolved, because It is not always possible to arrive at the solution that one expects, but a solution is reached which makes you reach satisfaction or reformulate what you are proposing.

SIMNON is as useful for simulating chemical processes as power plants. It can be used for complex control algorithms, for financial models, robot dynamics. All systems, which can be defined in mathematical terms, can also be simulated in SIMNON

Thousands of people worldwide are using SIMNON as an efficient tool to simulate processes and products. Universities and research centers in more than 40 countries have found that SIMNON is a great program for interactive simulation.

Contenido

Dedicatoria III
Resumen VII
Abstract IX
Contenido XI
Lista de Figuras XIII
Lista de Tablas XV
Lista de Símbolos XVII

1. INTRODUCCIÓN 1
1.1. La importancia de la simulación de sistemas 1
1.1.1. Definiciones de simulación 3
1.1.2. Ventajas y desventajas de la simulación 8
1.1.3. Elementos clave para garantizar el exito de un modelo de simulación 9
1.2. Objetivos 12
1.2.1. Generales 12
1.2.2. Particualares 12
1.3. Justificación 12
1.4. Metodologia para realizar un buen estudio de simulación 13
1.5. Estructura de la tesis 17

2. ANTECEDENTES DE SOFTWARE DE SIMULACIÓN DE SISTEMAS 19
2.1. Introducción 19
2.1.1. Programas computacionales 20
2.2. Antecedentes Históricos 21
2.2.1. Periodo de formación (1945-1970) 22
2.2.2. Periodo de expansión (1970-1981) 24
2.2.3. Softwares de simulación actuales 25
2.3. SIMNON 36
2.3.1. Introducción a SIMNON 36
2.3.2. ¿Para que sirve SIMNON? 37
2.3.3. ¿Que es la simulación? 38
2.3.4. Descripciones de sistemas dinámicos 38
2.3.5. Principios de Interacción 39
2.3.6. Ecuaciones Diferenciales 41

2.3.7. Ejemplo aplicado en la solucion del problema de VAN DER POL . . 41
2.3.8. Ingresando a la descripcion del sistema 42
2.3.9. Ejemplo de una simulación 43
2.3.10. Interrumpir una simulación 44
2.3.11. Cambiar parametros 45
2.3.12. Gráficos de plano de fase 47
2.3.13. Generalidades 48
2.4. Resumen 50

3. MODELADO DE SISTEMAS CONTINUOS 53
3.1. Introducción 53
3.2. Clasificacion de sistemas en tiempo continuo 54
3.2.1. Sistemas lineales y no lineales 54
3.2.2. Sistemas variantes e invariantes con el tiempo 57
3.2.3. Sistemas con memoria y sin memoria 59
3.3. Sistemas lineales e invariantes en el tiempo y sus propiedades 60
3.3.1. Propiedades de los sistemas LTI 60
3.4. Sistemas descritos por ecuaciones diferenciales 61
3.4.1. Metodo de espacios de estado parcial 61
3.5. Resumen 67

4. SIMULACIONES EN SIMNON 69
4.1. Caso de estudio 1 - Motor de CD 69
4.1.1. Simulación del motor de CD 71
4.2. Caso de estudio 2 - Rectificadores de onda aplicados a un motor de CD . . 77
4.2.1. Rectificador de media onda 77
4.2.2. Motor de CD con un rectificador de media onda 77
4.2.3. Rectificador de onda completa 83
4.2.4. Motor de CD con un puente rectificador de onda completa 86
4.2.5. Resumen 90

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS 93
5.1. Conclusiones 93
5.2. Trabajos futuros 94
5.3. Recomendaciones 95

A. Lista de comandos del editor de texto SIMON y terminal SIMNON 97

B. Descripciones del sistema SIMNON 101

C. Terminos importantes 103

D. Derechos de autor de las imagenes 105

Referencias 107

Lista de Figuras

1.1. Representación de conceptos de simulación 5
1.2. Gráfica de estabilización de una variable 8

2.1. La Aguja de Buffon - Derechos de autor de la imagen en(1.1b) 22
2.2. Proyecto manhattan - Derechos de autor de la imagenen en (1.2b) 23
2.3. Keith Douglas Tocher - Derechos de autor de la imagenen en (1.3b) 24
2.4. Ole-Johan Dahl (1931-2002) y Kristen Nygaard (1926-2002) La primera definición formal de Simula 67 apareció en mayo de 1967 - Derechos de autor de la imagenen en (1.4b) 25
2.5. Logo Matlab - Derechos de autor de la imagenen en (1.5b) 27
2.6. Logo Modelica - Derechos de autor de la imagenen en (1.5b) 27
2.7. Logo LabVIEW - Derechos de autor de la imagenen en (1.6b) 29
2.8. Logo de inria 31
2.9. Logo de Digiteo 32
2.10. Logo de Scilab 32
2.11. Logo scilab enterprises 33
2.12. Logo ESI group 33
2.13. Logo GNU Octave 35
2.14. Diagrama de sintaxis para el comando SIMU 40
2.15. Simulación de la ecuación de van der Pol para a = 1 y b = 1 con valores iniciales x (0) = 1 e y (0) = 0. 45
2.16. Solución de la ecuación de van der Pol para b = 1 y b = 2. 47
2.17. Gráfico del plano de fase de la ecuación de van der Pol para a = 1, b = 1, x (0) = 1, y (0) = 0. 48

3.1. Ejemplos de sistemas a) en tiempo continuo y b) tiempo discreto 54
3.2. El sistema del ejemplo 3.1 55
3.3. Circuito RL. 64
3.4. Circuito RLC. 66

4.1. Representación de un motor de DC 70
4.2. Proyecto nuevo 72
4.3. Terminal de comandos en el entorno SIMNON 73
4.4. Gráficador 74

4.5. Grafica de comportamiento de un motor de CD 75
4.6. Cambiar las escalas de una gráfica 75
4.7. Gráfica de comportamiento de un motor de CD en diagrama de fase 76
4.8. Representación de un motor de DC con un diodo rectificador 77
4.9. Polarizacion de un diodo; a) Semiciclo positivo b)Semiciclo negativo (Derechos de autor de la imagen en 4.1b) 78
4.10. Gráfica del comportamiento del motor de CD con el rectificador de media onda 80
4.11. Gráfica de planos de fase del comportamiento del motor de CD con el rectificador de media onda 81
4.12. Gráfica del voltaje de la fuente del circuito de la figura 4.1 82
4.13. Gráfica del voltaje del rectificador del circuito de la figura 4.1 82
4.14. Representación de un puente rectificador de diodos 83
4.15. Semiciclo positivo en un puente rectificador de diodos 84
4.16. Semiciclo negativo en un puente rectificador de diodos 85
4.17. a) Onda sinusoidal antes ser rectificada. b) Onda sinusoidal rectificada por un puente rectificador de diodos 85
4.18. Representación de un motor de CD con un puente rectificador de diodos . . 86
4.19. Gráfica del comportamiento del motor de CD con el rectificador de onda completa 87
4.20. Gráfica de planos de fase del comportamiento del motor de CD con el rectificador de onda completa 88
4.21. Gráfica del voltaje de la fuente del circuito de la figura 4.1 89
4.22. Gráfica del voltaje rectificado del circuito de la figura 4.1 90
4.23. Comparación de las 3 gráficas 91
4.24. Gráfica sinusoidal con un error de .001 92
4.25. Modificar el error en una simulación 92
4.26. Gráfica sinusoidal con un error de 1e-10 92

Lista de Tablas

2.1. Pasos de la sintaxis basica del comando SIMU 42
2.2. Listado 1. Un sistema de Simnon para la ecuación (2.4) 42
2.3. Listado 2. Forma genérica de descripción del sistema para la simulación de ecuaciones diferenciales. 49
2.4. Tabla de funciones en SIMNON 51

Lista de Símbolos

α	-	Constante alfa
β	-	Constante beta
π	-	pi
τ	-	Variable Tau
f	-	Función f
g	-	Función g
dx	-	Derivada en funcion de x
dy	-	Derivada en funcion de y
dt	-	Derivada en funcion de t
$y(t)$	-	Salida de un sistema
$x(t)$	-	Entrada de un sistema
exp	-	Exponencial
∞	-	Infinito
L	-	Inductancia
C	-	Capacitancia
R	-	Resistencia
v	-	Voltaje
i	-	Corriente
J	-	Momento de inercia en el rotor
B	-	Coeficiente de fricción
n	-	Constante de representación para K_m y K_a
E_a	-	Tensión generada
T_m	-	Torque del motor
ω	-	Velocidad angular
K_m	-	Relación entre el voltaje y velocidad en un motor
K_a	-	Relación entre el torque y la corriente eléctrica
V_s	-	Voltaje en corriente alterna
V_a	-	Voltaje en corriente directa
$\dot{i}$	-	Derivada de i
$\dot{\omega}$	-	Aceleración angular
x	-	Variable x
y	-	Variable y
$\dot{x}$	-	Otra Representacion de derivada en función de x
$\dot{y}$	-	Otra Representacion de derivada en función de y

Capítulo 1

INTRODUCCIÓN

1.1. La importancia de la simulación de sistemas

La creación de nuevos y mejores desarrollos en el área de la computación ha traído consigo innovaciones muy importantes tanto en la toma de decisiones como en el diseño de procesos y productos. Una de las técnicas para realizar estudios piloto, con resultados rápidos y a un relativo bajo costo, se basa en la modelación — la cual se conoce como simulación — que se ha visto beneficiada por estos avances.

Actualmente, el analista que desea hacer uso de esta herramienta dispone de una gran cantidad de software de simulación que le permite tomar decisiones en temas muy diversos. Por ejemplo, en aplicaciones donde se usan lenguajes de programación general como Fortran, para analizar esquemas de administración del inventario, en aplicaciones para analizar y mejorar la administración de la cadena de suministro, en análisis y validación de medidas de desempeño en sistemas de manufactura en aplicaciones de mejoras mediante despliegues seis sigma, o en procesos de capacitación a través de la simulación como herramienta base. Sin duda, la flexibilidad en cuanto a la modelación, el análisis y el mejoramiento de sistemas ha hecho de la simulación una herramienta cuyo uso y desarrollo se han visto alentados de manera significativa.

En la actualidad resulta fácil encontrar paquetes de software con gran capacidad de análisis, así como mejores animaciones (2D y 3D) y aplicaciones para generación de reportes con

información relevante para la toma de decisiones. En general, dichos paquetes — ya sea orientado a procesos, a servicios, o de índole general — nos proveen de una enorme diversidad de herramientas estadísticas que permiten tanto un manejo más eficiente de la información relevante bajo análisis como una mejor presentación e interpretación.

El concepto de simulación engloba soluciones para muchos propósitos diferentes. Por ejemplo, podríamos decir que el modelo de un avión a escala que se introduce a una cámara por donde se hace pasar un flujo de aire, puede simular los efectos que experimentará un avión real cuando se vea sometido a turbulencia. Por otro lado, algunos paquetes permiten hacer la representación de un proceso de fresado o torneado: una vez que el usuario establezca ciertas condiciones iniciales, podrá ver cómo se llevaría a cabo el proceso real, lo que le da la posibilidad de revisarlo sin necesidad de desperdiciar material ni poner en riesgo la maquinaria.

Por lo tanto, la simulación se refiere a un gran conjunto de métodos y aplicaciones que buscan imitar el comportamiento de sistemas reales, generalmente por medio de una computadora con un software apropiado.

Existen distintos modelos de simulación que permiten representar situaciones reales de diferentes tipos. Podemos tener **modelos físicos** — como el del avión que mencionamos en el párrafo anterior — o modelos matemáticos, a los cuales pertenecen los modelos de simulación de eventos discretos. Asimismo, los modelos pueden diferenciarse según el tipo de ecuaciones matemáticas que los componen. Por ejemplo, los **modelos continuos** son aquellos en los que las relaciones entre las variables relevantes de la situación real se definen por medio de ecuaciones diferenciales, ya que éstas permiten conocer el comportamiento de las variables en cierto tiempo. Problemas tales como saber de qué manera se transfiere el calor en un molde, o determinar cómo fluye cierto material dentro de una tubería, e incluso prever el comportamiento del nivel de un tan que de gasolina al paso del tiempo, mientras el vehículo está en marcha, pueden simularse con este tipo de modelo.

Además de modelos continuos tenemos **modelos discretos**. En ellos el comportamiento que nos interesa analizar puede representarse por medio de ecuaciones evaluadas en un punto determinado. Por ejemplo, si hacemos un muestreo del número de personas que llegaron a un banco en un lapso específico, podemos simular esta variable con ecuaciones ligadas a

distribuciones de probabilidad que reflejen dicho comportamiento.

Otro tipo de clasificación es el de los **modelos dinámicos o estáticos**. Los modelos dinámicos son aquellos en los que el estado del sistema que estamos analizando cambia respecto del tiempo. Por ejemplo, el número de personas que hacen fila para entrar a una sala de cine varía con el tiempo. Por otro lado, los modelos estáticos representan un resultado bajo un conjunto de situaciones o condiciones determinado; por ejemplo, al lanzar un dado los únicos valores que se puede obtener son 1, 2, 3, 4, 5 o 6, de manera que el resultado de la simulación será uno de tales valores posibles; a esto se le conoce generalmente como simulación de Monte Carlo.

Por último, podemos hablar de modelos **determinísticos y modelos probabilísticos**, llamados también estocásticos. Los primeros se refieren a relaciones constantes entre los cambios de las variables del modelo. Por ejemplo, si las cajas empleadas en un proceso contienen siempre 5 productos, cada vez que se añada una caja al inventario éste se incrementará en 5 unidades. Si, por el contrario, hay una distribución de probabilidad en el proceso de manera que, por ejemplo, algunas cajas contienen 3 productos y otras 4, el inventario se modificará según el número de piezas de cada caja y, en consecuencia, será necesario un modelo estocástico. En el caso de la simulación de eventos discretos hablaremos de modelos matemáticos, discretos, dinámicos, y que pueden incluir variables determmísticas y probabilísticas.[1]

1.1.1. Definiciones de simulación

Para poder realizar un buen estudio de simulación es necesario entender los conceptos básicos que componen nuestro modelo.

Comenzaremos por definir el concepto de **simulación de eventos discretos** como el conjunto de relaciones lógicas, matemáticas y probabilísticas que integran el comportamiento de un sistema bajo estudio cuando se presenta un evento determinado. El objetivo del modelo de simulación consiste, precisamente, en comprender, analizar y mejorar las condiciones de operación relevantes del sistema.

En la definición anterior encontramos elementos como sistema, modelo y evento, de los que

se desprenden otros conceptos importantes dentro de una simulación. A continuación abundaremos en cada uno.

La definición básica de **sistema** nos dice que se trata de *un conjunto de elementos que se interrelacionan para funcionar como un todo;* desde el punto de vista de la simulación, tales elementos deben tener una frontera clara. Por ejemplo, podemos hablar del sistema de atención a clientes en un banco, del sistema de inventarios de una empresa, o del sistema de atención en la sala de emergencia de un hospital. Cada uno puede dividirse en elementos que son relevantes para la construcción de lo que será su modelo de simulación; entre ellos tenemos **entidades, estado del sistema, eventos actuales y futuros, localizaciones, recursos, atributos, variables, y el reloj de la simulación**, los cuales a continuación se describen:

Una entidad por lo general es la representación dé los flujos de entrada y salida en un sistema; al entrar a un sistema una entidad es el elemento responsable de que el estado del sistema cambie. Ejemplos de entidades pueden ser; los clientes que llegan a la caja de un banco, las piezas que llegan a un proceso, o el embarque de piezas que llega a un inventario.

El **estado del sistema** es la condición que guarda el sistema bajo estudio en un momento de tiempo determinado; es como una fotografía de lo que está pasando en el sistema en cierto instante. El estado del sistema se compone de variables o características de operación puntuales (digamos el número de piezas que hay en el sistema en ese momento), y de variables o características de operación acumuladas, o promedio (como podría ser el tiempo promedio de permanencia de una entidad en el sistema, en una fila, almacén o equipo).

Un **evento** es un cambio en el estado actual del sistema; por ejemplo, la entrada o salida de una entidad, la finalización de un proceso en un equipo, la interrupción o reactivación de una operación (digamos por un descanso del operario), o la descompostura de una máquina. Podemos catalogar estos eventos en dos tipos: **eventos actuales**, aquellos que están sucediendo en el sistema en un momento dado, y **eventos futuros**, cambios que se presentarán en el sistema después del tiempo de simulación, de acuerdo con una programación específica. Por ejemplo, imagine que cierta pieza entra a una máquina para que ésta realice un proceso. El evento actual sería precisamente que la entidad llamada "pieza"se encuentra en la máquina. El evento futuro podría ser el momento en que la máquina concluirá su trabajo

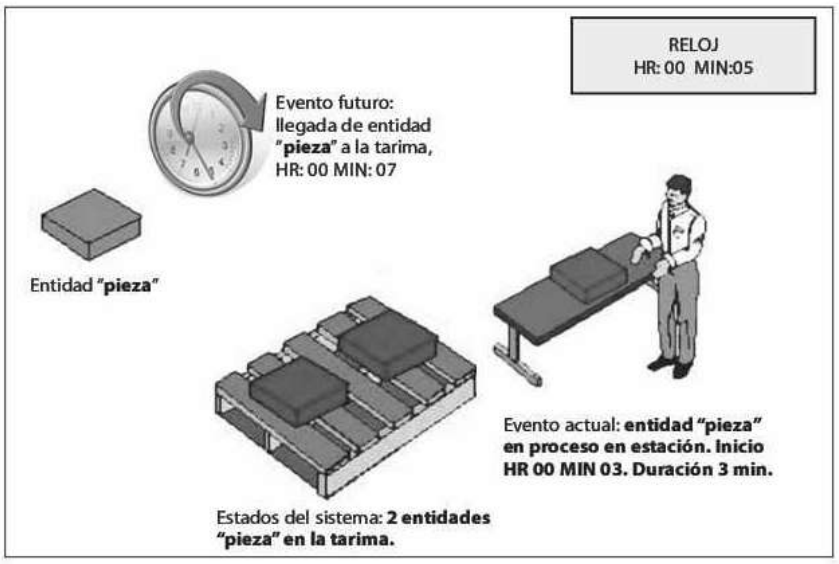

Figura 1.1: Representación de conceptos de simulación

con la pieza y ésta seguirá su camino hacia el siguiente proceso lógico, de acuerdo con la programación: almacenamiento, inspección o entrada a otra máquina.[1] Para ilustrar con mayor claridad estas definiciones veamos la figura 1.1.

Además del esquema transaccional (pieza en tarima $->$ pieza en estación) que se presenta en un modelo de simulación, es necesario considerar algunos otros elementos que también forman parte de este tipo de modelaciones. A continuación describiremos algunos de ellos.

Las localizaciones son todos aquellos lugares en los que la pieza puede detenerse para ser transformada o esperar a serlo. Dentro de estas localizaciones tenemos almacenes, bandas transportadoras, máquinas, estaciones de inspección, etcétera. En el caso de la figura 1.1 la tarima y la estación serían consideradas localizaciones del modelo. Observe que en el caso de la estación una sola localización puede ser representada gráficamente por varias figuras. En la estación observamos una mesa y a una persona que en conjunto forman una sola localización. En términos de simulación algunos paquetes permiten la animación de lo que se programó. En estos paquetes la representación iconográfica es sólo para aspectos visuales y no le resta o agrega potencia al modelo. Es decir, podríamos haber colocado una casa en

lugar de la estación con mesa y operador y el modelo nos hubiese dado el mismo resultado. Los recursos son aquellos dispositivos — diferentes a las localizaciones— necesarios para llevar a cabo una operación. Por ejemplo, un montacargas que transporta una pieza de un lugar a otro: una persona que realiza la inspección en una estación y toma turnos para descansar; una herramienta necesaria para realizar un proceso pero que no forma parte de una localización específica, sino que es trasladada de acuerdo con los requerimientos de aquel.
Un atributo es una característica de una entidad. Por ejemplo, si la entidad es un motor, los atributos serían su color, peso, tamaño o cilindraje. Los atributos son muy útiles para diferenciar entidades sin necesidad de generar una nueva, y pueden adjudicarse al momento de la creación de la entidad, o asignarse y/o cambiarse durante el proceso. Como indica su nombre, las variables son condiciones cuyos valores se crean y modifican por medio de ecuaciones matemáticas y relaciones lógicas. Pueden ser continuas (por ejemplo, el costo promedio de operación de un sistema) o discretas (como el número de unidades que deberá envasarse en un contenedor). Las variables son muy útiles para realizar conteos de piezas y ciclos de operación, así como para determinar características de operación del sistema.
El reloj de la simulación es el contador de tiempo de la simulación, y su función consiste en responder preguntas tales como cuánto tiempo se ha utilizado el modelo en la simulación, y cuánto tiempo en total se quiere que dure esta última. En general, el reloj de simulación se relaciona con la tabla de eventos futuros, ya que al cumplirse el tiempo programado para la realización de un evento futuro, éste se convierte en un evento actual. Regresando al ejemplo de la figura 1.1, cuando el tiempo de proceso se cumpla, la pieza seguirá su camino hasta su siguiente localización, si ésta es la última del sistema lo más probable es que su siguiente proceso sea salir del sistema; el reloj simula precisamente ese tiempo.
Podemos hablar de dos tipos de reloj de simulación: **el reloj de simulación absoluto**, que parte de cero y termina en un tiempo total de simulación definido, y **el reloj de simulación relativo**, que sólo considera el lapso que transcurre entre dos eventos. Por ejemplo, podemos decir que el tiempo de proceso de una pieza es relativo, mientras que el absoluto sería el tiempo global de la simulación: desde que la pieza entró a ser procesada hasta el momento en el que terminó su proceso.[1]

Otro concepto importante que vale la pena definir es el de réplica o corrida de la simulación. Cuando ejecutamos el modelo una vez, los valores que obtenemos de las variables y parámetros al final del tiempo de simulación generalmente serán distintos de los que se producirán si lo volvemos a correr con diferentes números pseudoaleatorios. Por lo tanto, es necesario efectuar más de una réplica del modelo que se esté analizando, con la finalidad de obtener estadísticas de intervalo que nos den una mejor ubicación del verdadero valor de la variable bajo los diferentes escenarios que se presentan al modificar los números pseudoaleatorios en cada oportunidad.
De esta manera, la pregunta clave es: ¿cuánto tiempo se debe simular un modelo para obtener resultados confiables? En general, podemos decir que todas las variables que se obtienen en términos de promedios presentan dos diferentes etapas: **un estado transitorio y un estado estable**. El primero se presenta al principio de la simulación; por ejemplo, en el arranque de una planta, cuando no tiene material en proceso: el último de los procesos estará inactivo hasta que el primer cliente llegue, y si el tiempo de simulación es bajo, su impacto sobre la utilización promedio de este proceso será muy alto, lo cual no ocurriría si el modelo se simulara lo suficiente para lograr una compensación. En el estado transitorio hay mucha variación entre los valores promedio de las variables de decisión del modelo, por lo que formular conclusiones con base en ellos sería muy arriesgado, toda vez que difícilmente nos darían una representación fiel de la realidad.[1]

Por otro lado, en el estado estable los valores de las variables de decisión permanecen muy estables, y presentan sólo variaciones poco significativas. En este momento las decisiones que se tomen serán mucho más confiables. Sin embargo, no todas las variables convergen al estado estable con la misma rapidez: algunas pasan con más lentitud que otras de un estado transitorio a uno estable. Es responsabilidad del analista verificar que las variables de decisión del modelo se encuentren en estado estable antes de detener el tiempo de la simulación (vea la figura 1.2). Otro factor importante para decidir el tiempo de simulación es el costo de la corrida. Mayor tiempo de simulación requiere más tiempo computacional, lo cual implica, necesariamente, un costo más alto. Por supuesto, la situación empeora si a esto le agregamos que en algunos casos es preciso efectuar más de tres réplicas.[1]

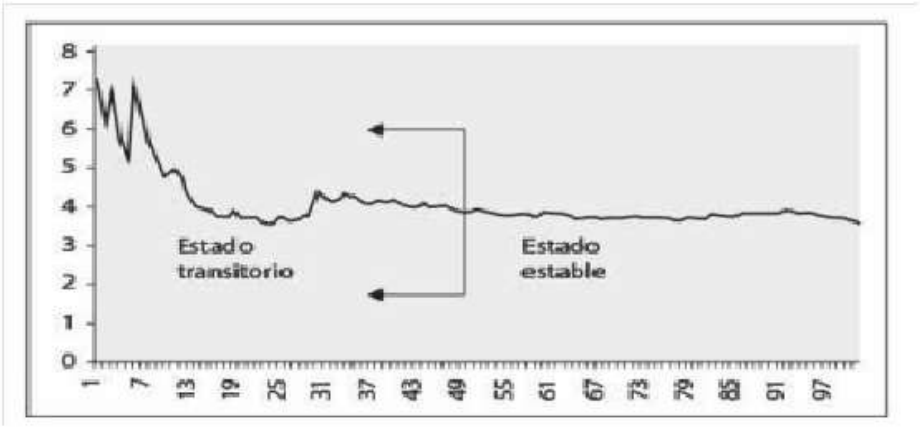

Figura 1.2: Gráfica de estabilización de una variable

1.1.2. Ventajas y desventajas de la simulación

Como hemos visto hasta ahora, la simulación es una de las diversas herramientas con las que cuenta el analista para tomar decisiones y mejorar sus procesos. Sin embargo, se debe destacar que, como todas las demás opciones de que disponemos, la simulación de eventos discretos presenta ventajas y desventajas que se precisa tomar en cuenta al decidir si es apta para resolver un problema determinado.

Dentro de las ventajas más comunes que ofrece la simulación podemos citar las siguientes:

a) Es muy buena herramienta para conocer el impacto de los cambios en los procesos, sin necesidad de llevarlos a cabo en la realidad.

b) Mejora el conocimiento del proceso actual ya que permite que el analista vea cómo se comporta el modelo generado bajo diferentes escenarios.

c) Puede utilizarse como medio de capacitación para la toma de decisiones.

d) Es más económico realizar un estudio de simulación que hacer muchos cambios en los procesos reales.

e) Permite probar varios escenarios en busca de las mejores condiciones de trabajo de los procesos que se simulan.

f) En problemas de gran complejidad, la simulación permite generar una buena solución.

g) En la actualidad los paquetes de software para simulación tienden a ser más sencillos, lo que facilita su aplicación.

h) Gracias a las herramientas de animación que forman parte de muchos de esos paquetes es posible ver cómo se comportará un proceso una vez que sea mejorado.

Éstas son algunas de las desventajas que la simulación puede presentar:

a) Aunque muchos paquetes de software permiten obtener el mejor escenario a partir de una combinación de variaciones posibles, la simulación no es una herramienta de optimización.

b) La simulación puede ser costosa cuando se quiere emplearla en problemas relativamente sencillos de resolver, en lugar de utilizar soluciones analíticas que se han desarrollado de manera específica para ese tipo de casos.

c) Se requiere bastante tiempo — por lo general meses — para realizar un buen estudio de simulación; por desgracia, no todos los analistas tienen la disposición (o la oportunidad) de esperar ese tiempo para obtener una respuesta.

d) Es preciso que el analista domine el uso del paquete de simulación y que tenga sólidos conocimientos de estadística para interpretar los resultados.

e) En algunas ocasiones el cliente puede tener falsas expectativas de la herramienta de simulación, a tal grado que le asocia condiciones similares a un videojuego o a una bola de cristal que le permite predecir con exactitud el futuro.[1]

1.1.3. Elementos clave para garantizar el exito de un modelo de simulación

Pese a los beneficios que conlleva la simulación, es imposible garantizar que un modelo tendrá éxito. Existen ciertas condiciones clave que pueden traer problemas si no se

les pone atención al momento de usar la simulación para la toma de decisiones. A continuación destacaremos algunas de las causas por las que un modelo de simulación podría no tener los resultados que se desean.

Tamaño insuficiente de la corrida. Como se mencionó antes, para poder llegar a conclusiones estadísticas válidas a partir de los modelos de simulación es necesario que las variables aleatorias de respuesta se encuentren en estado estable. El problema estriba en que, por lo general, cuando el modelo consta de más de una variable de decisión, es difícil que éstas alcancen un estado estable al mismo tiempo: es posible que una se encuentre estable y la otra no en un momento determinado, por lo que las conclusiones respecto de la segunda variable no serán confiables en cuanto a la estadística.

Variable(s) de respuesta mal definida(s). Aún cuando el modelo de simulación sea muy eficiente y represente en gran medida la realidad, si la variable de respuesta seleccionada no es la apropiada, será imposible tomar decisiones que tengan impacto en la operación del sistema bajo estudio. Por ejemplo, digamos que una variable de respuesta es el nivel de inventarios de cierto producto, sin embargo, la política de la empresa establece que no se debe parar ninguno de los procesos de fabricación. En consecuencia, el problema no será el inventario final, sino el ritmo de producción necesario para que aquel cumpla con los requerimientos de diseño que se desean.

Errores al establecer las relaciones entre las variables aleatorias. Un error común de programación es olvidar las relaciones lógicas que existen entre las variables aleatorias del modelo, o minimizar su impacto. Si una de estas variables no está definida de manera correcta, es posible tener un modelo que se apegue a la realidad actual; sin embargo, si el sistema no se lleva hasta su máxima capacidad para observar su comportamiento, podría resultar imposible visualizar el verdadero impacto de las deficiencias.

Errores al determinar el tipo de distribución asociado a las variables aleatorias del modelo. Este tipo de problema es muy similar al anterior, pero en este caso se utilizan distribuciones que no son las más adecuadas o que responden únicamente a un intento de simplificar los estudios estadísticos. Digamos, por ejemplo, que se nos dan los siguientes parámetros de producción aproximados: mínimo 10, máximo 40, y promedio 30. En esta circunstancia la tentación de simplificar el estudio de la variable asignándole una distribu-

ción triangular con parámetros (10, 30, 40) es muy grande; no obstante, hacerlo afectaría de manera importante los resultados de la simulación, pues el modelo podría alejarse de lo que sucede en la realidad.

Falta de un análisis estadístico de los resultados. Un problema común por el que la simulación suele ser objeto de crítica, radica en asumir que se trata de una herramienta de optimización. Esta apreciación es incorrecta, ya que involucra variables aleatorias y características propias de un modelo que incluye probabilidades. Por lo mismo — como se apuntó antes—, es necesario realizar varias corridas a fin de producir diferentes resultados finales para las variables de respuesta y, a partir de esos valores, obtener intervalos de confianza que puedan dar un rango en dónde encontrar los valores definitivos. Este tipo de problemas se presentan también al comparar dos escenarios: podríamos encontrar un mejor resultado para uno de ellos, pero si los intervalos de confianza de las variables de respuesta se traslapan resultaría imposible decir que el resultado de un escenario es mejor que el del otro. De hecho, en lo que a la estadística se refiere, ambos resultados pueden ser iguales. En ese caso incrementar el tamaño de corrida o el número de réplicas puede ayudar a obtener mejores conclusiones.

Uso incorrecto de la información obtenida. Un problema que se presenta en ocasiones es el uso incorrecto de la información recabada para la realización del estudio, ya sea a través de un cliente o de cualquier otra fuente. Muchas veces esta información se recolecta, analiza y administra de acuerdo con las necesidades propias de la empresa, lo que implica que no siempre está en el formato y la presentación que se requiere para la simulación. Si la información se utiliza para determinar los parámetros del modelo sin ser depurada y reorganizada, es muy probable que la precisión de los resultados del estudio se vea afectada.

Falta o exceso de detalle en el modelo. Otro punto importante a considerar es el nivel de detalle del modelo. En muchas ocasiones algún proceso se simplifica tanto que tiende a verse como una çaja negra"que nos impide ver qué ocurre en el interior, aunque sí haya entrada y salida de datos que interactúan con otras partes del modelo. Cuando esto sucede, el impacto que podrían tener los subprocesos que se llevan a cabo en la çaja negra"(es decir, del proceso sobre simplificado) no se incluye en la simulación. Por ejemplo, si se analiza un sistema de distribución y se da por sentado que el almacén siempre surte sus pedidos, no

incluiremos el impacto de los tiempos necesarios para surtir las órdenes, ni la posibilidad de que haya faltantes de producto; excluiremos también los horarios de comida, en los que no se surten pedidos, y las fallas en los montacargas que transportan los pedidos hasta los camiones para su distribución. Por otra parte, si el modelo se hace demasiado detallado, tanto el tiempo dedicado al estudio como el costo de llevarlo a cabo podrían incrementarse sustancialmente. Es labor del encargado de la simulación sugerir y clarificar los niveles de detalle que se requieren en el modelo, resaltando los alcances y limitaciones de cada uno.

1.2. Objetivos

1.2.1. Generales

- El presente trabajo tiene como objetivo principal emplear y dar a conocer una herramienta alternativa a los alumnos para la simulación de sistemas continuos.
- También mostrar la importancia que tiene hacer simulaciones de sistemas continuos.
- Hacer uso del software SIMNON para resolver diferentes casos de estudio.
- Hacer una pequeña guía introductoria de uso para el software SIMNON.

1.2.2. Particualares

- Aprender a utilizar el simulador SIMNON.
- Implementar los conocimientos adquiridos durante la carrera para desarrollar un documento de forma profesional.

1.3. Justificación

Este trabajo se justifica porque es una alternativa para hacer simulaciones de sistemas continuos, existe una gran variedad de programas dedicados a esta misma finalidad, pero este se diferencia por su gran simplicidad y la gran potencia con la que cuenta para resolver problemas específicos, ademas SIMNON es tan útil para la simulación de procesos

químicos como las centrales eléctricas. Puede usarse para algoritmos de control complejos, para modelos financieros, dinámica de robots. y en general todos los sistemas, que se pueden definir en términos matemáticos.

1.4. Metodologia para realizar un buen estudio de simulación

Debemos considerar que — igual a lo que ocurre con otras herramientas de investigación — la realización de un estudio de simulación requiere la ejecución de una serie de actividades y análisis que permitan sacarle el mejor provecho. A continuación se mencionan los pasos básicos para realizar un estudio de simulación, aunque en muchas ocasiones será necesario agregar otros o suprimir algunos de los aquí enumerados, de acuerdo con la problemática en cuestión.

1. **Definición del sistema bajo estudio.** En esta etapa es necesario conocer el sistema a modelar. Para ello se requiere saber qué origina el estudio de simulación y establecer los supuestos del modelo: es conveniente definir con claridad las variables de decisión del modelo, determinar las interacciones entre éstas, y establecer con precisión los alcances y limitaciones que aquel podría llegar a tener. Antes de concluir este paso es recomendable contar con la información suficiente para lograr establecer un modelo conceptual o un mapa mental del sistema bajo estudio, el cual debe incluir sus fronteras y todos los elementos que lo componen, además de las interacciones entre ellos, los flujos de productos, las personas y los recursos, así como las variables de mayor interés para el problema.

2. **Generación del modelo de simulación base.** Una vez que se ha definido el sistema en términos de un modelo conceptual, la siguiente etapa del estudio consiste en la generación de un modelo de simulación base. No es preciso que este modelo sea demasiado detallado, ya que se requiere mucha más información estadística sobre el comportamiento de las variables de decisión del sistema. La generación de este modelo es el primer reto para el programador de la simulación, ya que debe traducir a un lenguaje de simulación la información que se obtuvo en la etapa de definición del sistema, e incluir las interrelaciones de todos los posibles subsistemas que existan en el problema a modelar. En caso de

que se requiera una animación, éste también es un buen momento para definir qué gráfico puede representar mejor el sistema que se modela. Igual que ocurre en otras ramas de la investigación de operaciones, "la simulación exige ciencia y arte en la generación de sus modelos". El realizador de un estudio de simulación es, en este sentido, como un artista que debe usar toda su creatividad para realizar un buen modelo que refleje la realidad del problema que se está analizando. Conforme se avanza en el modelo base se pueden ir agregando las variables aleatorias del sistema, con sus respectivas distribuciones de probabilidad asociadas.

3. **Recolección y análisis de datos.** Es posible comenzar la recopilación de la información estadística de las variables aleatorias del modelo de manera paralela a la generación del modelo base. En esta etapa se debe establecer qué información es útil para la determinación de las distribuciones de probabilidad asociadas a cada una de las variables aleatorias necesarias para la simulación. Aunque en algunos casos se logra contar con datos estadísticos, suele suceder que el formato de almacenamiento o de generación de reportes no es el apropiado para facilitar el estudio. Por ello, es muy importante dedicar el tiempo suficiente a esta actividad. De no contar con la información requerida o en caso de desconfiar de la disponible, será necesario realizar un estudio estadístico del comportamiento de la variable que se desea identificar, para luego incluirla en el modelo. Más adelante se hará el análisis de los datos indispensables para asociar una distribución de probabilidad a una variable aleatoria, así como las pruebas que se le deben aplicar. Al finalizar la recolección y análisis de datos para todas las variables del modelo, se tendrán las condiciones para generar una versión preliminar del problema que se está simulando.

4. **Generación del modelo preliminar.** En esta etapa se integra la información obtenida a partir del análisis de los datos, los supuestos del modelo y todos los datos necesarios para crear un modelo lo más cercano posible a la realidad del problema bajo estudio. En algunos casos — sobre todo cuando se trata del diseño de un nuevo proceso o esquema de trabajo — no se cuenta con información estadística, por lo que debe estimarse un rango de variación o determinar (con ayuda del cliente) valores constantes que permitan realizar el modelado. Si éste es el caso, el encargado de la simulación puede, con base

en su experiencia, realizar algunas sugerencias de distribuciones de probabilidad que comúnmente se asocien al tipo de proceso que se desea incluir en el modelo. Al finalizar esta etapa el modelo está listo para su primera prueba: su verificación o, en otras palabras, la comparación con la realidad.

5. **Verificación del modelo.** Una vez que se han identificado las distribuciones de probabilidad de las variables del modelo y se han implantado los supuestos acordados, es necesario realizar un proceso de verificación de datos para comprobar la propiedad de la programación del modelo, y comprobar que todos los parámetros usados en la simulación funcionen correctamente. Ciertos problemas, en especial aquellos que requieren muchas operaciones de programación o que involucran distribuciones de probabilidad difíciles de programar, pueden ocasionar que el comportamiento del sistema sea muy diferente del que se esperaba. Por otro lado, no se debe descartar la posibilidad de que ocurran errores humanos al alimentar el modelo con la información. Incluso podría darse el caso de que los supuestos iniciales hayan cambiado una o varias veces durante el desarrollo del modelo. Por lo tanto, debemos asegurarnos de que el modelo que se va a ejecutar esté basado en los más actuales. Una vez que se ha completado la verificación, el modelo está listo para su comparación con la realidad del problema que se está modelando. A esta etapa se le conoce también como validación del modelo.

6. **Validación del modelo.** El proceso de validación del modelo consiste en realizar una serie de pruebas simultáneas con información de entrada real para observar su comportamiento y analizar sus resultados. Si el problema bajo simulación involucra un proceso que se desea mejorar, el modelo debe someterse a prueba con las condiciones actuales de operación, lo que nos dará como resultado un comportamiento similar al que se presenta realmente en nuestro proceso. Por otro lado, si se está diseñando un nuevo proceso la validación resulta más complicada. Una manera de validar el modelo en este caso, consiste en introducir algunos escenarios sugeridos por el cliente y validar que el comportamiento sea congruente con las expectativas que se tienen de acuerdo con la experiencia. Cualquiera que sea la situación, es importante que el analista conozca bien el modelo, de manera que pueda justificar aquellos comportamientos que sean contrarios a la experiencia de

los especialistas que participan en su validación.

7. **Generación del modelo final.** Una vez que el modelo se ha validado, el analista está listo para realizar la simulación y estudiar el comportamiento del proceso. En caso de que se desee comparar escenarios diferentes para un mismo problema, éste será el modelo raíz; en tal situación, el siguiente paso es la definición de los escenarios a analizar.

8. **Determinación de los escenarios para el análisis.** Tras validar el modelo es necesario acordar con el cliente los escenarios que se quieren analizar. Una manera muy sencilla de determinarlos consiste en utilizar un escenario pesimista, uno optimista y uno intermedio para la variable de respuesta más importante. Sin embargo, es preciso tomar en cuenta que no todas las variables se comportan igual ante los cambios en los distintos escenarios, por lo que tal vez sea necesario que más de una variable de respuesta se analice bajo las perspectivas pesimista, optimista e intermedia. El riesgo de esta situación radica en que el analista podría realizar un diseño de experimentos capaz de generar una gran cantidad de réplicas, lo que redundaría en un incremento considerable de costo, análisis y tiempo de simulación. Es por ello que muchos paquetes de simulación cuentan con herramientas para realizar este proceso, las cuales eliminan la animación y acortan los tiempos de simulación. Estas herramientas permiten realizar varias réplicas del mismo escenario para obtener resultados con estadísticas importantes respecto de la toma de decisiones (por ejemplo, los intervalos de confianza). Por su parte, el analista también puede contribuir a la selección de escenarios, sugiriendo aquellos que considere más importantes; al hacerlo dará pie a que se reduzca el número de combinaciones posibles.

9. **Análisis de sensibilidad.** Una vez que se obtienen los resultados de los escenarios es importante realizar pruebas estadísticas que permitan comparar los escenarios con los mejores resultados finales. Si dos de ellos tienen resultados similares será necesario comparar sus intervalos de confianza respecto de la variable de respuesta final. Si no hay intersección de intervalos podremos decir con certeza estadística que los resultados no son iguales; sin embargo, si los intervalos se sobreponen será imposible definir estadísticamente que una solución es mejor que otra. Si se desea obtener un escenario "ganador", será

necesario realizar más réplicas de cada modelo y/o incrementar el tiempo de simulación de cada corrida. Con ello se busca acortar los intervalos de confianza de las soluciones finales y, por consiguiente, incrementar la probabilidad de diferenciar las soluciones.

10. **Documentación del modelo, sugerencias y conclusiones.** Una vez realizado el análisis de los resultados, es necesario efectuar toda la documentación del modelo. Esta documentación es muy importante, ya que permitirá el uso del modelo generado en caso de que se requieran ajustes futuros. En ella se deben incluir los supuestos del modelo, las distribuciones asociadas a sus variables, todos sus alcances y limitaciones y, en general, la totalidad de las consideraciones de programación. También es importante incluir sugerencias tanto respecto del uso del modelo como sobre los resultados obtenidos, con el propósito de realizar un reporte más completo. Por último, deberán presentarse las conclusiones del proyecto de simulación, a partir de las cuales es posible obtener los reportes ejecutivos para la presentación final.[1]

1.5. Estructura de la tesis

Se pretende que esta tesis sea simple, didáctica y fácil de comprender para el lector; Para ello cuenta con imágenes, diagramas, tablas, y se desarrolla una introducción al inicio de cada capítulo y un pequeño resumen al final. Esta tesis tiene la siguiente estructura: Cuenta con una portada con los datos personales del pasante, dedicatorias y agradecimientos, un resumen y su justificación donde se presenta la importancia de este trabajo, cuenta con palabras clave o terminos importantes, un abstrac, un índice general, además de incluir índices imágenes.

En el capítulo 1 se describe de forma general la importancia de las simulaciones, datos históricos relevante las ventajas y desventajas que implica hacer un estudio de simulaciones, los elementos clave para realizar un buen estudio, además de la metodología que se tiene que llevar para realizar dichos estudios. En el capítulo 2 se describen los antecedentes de software de simulación además se incluyen una pequeña guía y los comandos básicos para la utilización del software simnon.

En el capítulo 3 se desarrollo el modelado matemático de los sistemas continuos

con ayuda del metodo de espacios de estado de forma parcial.

En el capítulo 4 se detallan las simulaciones de los casos de estudio realizadas en el software de simulación simnon.

En el capítulo 5 se obtienen las conclusiones, el porque se recomienda éste trabajo y trabajos futuros que se pueden hacer. Además cuenta con Apéndices correspondientes como: bibliografía, una lista de comandos, descripciones generales del software simnon y lo ya mencionado los terminos mas importantes.

Capítulo 2

ANTECEDENTES DE SOFTWARE DE SIMULACIÓN DE SISTEMAS

2.1. Introducción

En la actualidad existen dos tipos de usuarios de software. Por un lado están aquellos que toman lo que se les da. Es decir, quienes se limitan a las capacidades que encuentran en el modo estándar de operación del software existente. Por ejemplo, resulta muy sencillo resolver un sistema de ecuaciones lineales o generar una gráfica con valores x-y con Excel o con MATLAB. Como este modo de operación por lo común requiere un mínimo esfuerzo, muchos de los usuarios adoptan este modo de operación. Además, como los diseñadores de estos paquetes se anticipan a la mayoría de las necesidades típicas de los usuarios, muchos de los problemas pueden resolverse de esta manera. Pero, ¿qué pasa cuando se presentan problemas que están más allá de las capacidades estándar de dichas herramientas? Por desgracia, decir "Lo siento jefe, pero no lo sé hacer" no es algo aceptado en la mayoría de los círculos de la ingeniería. En tales casos se tienen varias alternativas. Como buscar otro paquete y ver si sirve para resolver el problema o que es posible volverse un "potente usuario" si se aprende a escribir macros en Excel VBA1 o archivos M (M-

files) en MATLAB.[4] Cualquiera de los dos casos es muy bueno pero nos centraremos en el primer caso el de buscar un nuevo paquete o programa, lo cual nos lleva a SIMNON del cual hablaremos posteriormente.

2.1.1. Programas computacionales

Los programas computacionales son únicamente conjuntos de instrucciones que dirigen a la computadora para realizar una cierta tarea. Existen una gran divercidad de programas computacionales, pero en esta tesis sólo nos basaremos en los software de simulación que son los de nuestro interes.[4]

El software de simulación se basa en el proceso de modelar un fenómeno real con un conjunto de fórmulas matemáticas. Es esencialmente, un programa que permite al usuario observar una operación a través de la simulación sin realizar esa operación. El software de simulación se usa ampliamente para diseñar equipos para que el producto final esté lo más cerca posible de las especificaciones de diseño sin costosas modificaciones en el proceso. Software de simulación en tiempo real. La respuesta a menudo se usa en juegos, pero también tiene importantes aplicaciones industriales. Cuando la penalización por una operación inadecuada es costosa, como los pilotos de aviones, operadores de plantas de energía nuclear u operadores de plantas químicas, una maqueta del panel de control real está conectada a una simulación en tiempo real de la respuesta física, brindando una valiosa experiencia de entrenamiento sin miedo a un resultado desastroso.

Los programas informáticos avanzados pueden simular el comportamiento del sistema de energía, condiciones climáticas, circuitos electrónicos, reacciones químicas, mecatrónica, bombas de calor, sistemas de control de retroalimentación, reacciones atómicas, incluso procesos biológicos complejos. En teoría, cualquier fenómeno que pueda reducirse a datos matemáticos y ecuaciones puede simularse en una computadora. La simulación puede ser difícil porque la mayoría de los fenómenos naturales están sujetos a un número casi infinito de influencias. Uno de los trucos para desarrollar simulaciones útiles es determinar cuáles son los factores más importantes que afectan los objetivos de la simulación.

Además de imitar procesos para ver cómo se comportan bajo diferentes condiciones, las simulaciones también se utilizan para probar nuevas teorías. Después de crear una teoría de las relaciones causales, el teórico puede codificar las relaciones en forma de un programa de computadora. Si el programa se comporta de la misma manera que el proceso real, existe una buena posibilidad de que las relaciones propuestas sean correctas.

2.2. Antecedentes Históricos

Se podría considerar que la simulación nace en el año 1777 con el planteamiento del problema de "la aguja de Buffon". Un método matemático sencillo para aproximar el valor numérico de π a partir de sucesivos intentos.
Este modelo matematico se basa en una aguja de una longitud determinada lanzada sobre un plano segmentado por líneas paralelas separadas por unidades. ¿Cuál es la probabilidad que la aguja cruce por alguna línea?
En 1812 Laplace [1] mejoró y corrigió la solución de Buffon[2] y desde entonces se conoce como solución Buffon-Laplace. Posteriormente, el estadístico William Sealy Gosset[3] que trabajaba en la destilería de Arthur Guinnes, 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 lo 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 le permitiera que la producción no se viera afectada por las variaciones de suelo y/o clima. Para evitar filtraciones de información confidencial Guinnes prohibió que sus empleados no realizaran ningún tipo de publicación de cualquier tipo independientemente de su contenido, a partir de ahí el uso que hizo Gosset en sus publicaciones del seudónimo "student", para evitar que su empleador lo detectara. Es por esta razón su logro más famoso se conoce como "distribución τ student". Que de otra manera habría sido conocida como distribución

[1] Pirre-Simon Laplace (1749-1827) Astronomo, físico y matematico francés.
[2] Georges Louis Leclerc, Conde de Buffon (1707-1788). Naturalista botánico, matemático biólogo, cosmólogo y escritor francés
[3] William Sealy Gosset (1876-1937) Estadistico mejor conocido por su nombre literario student

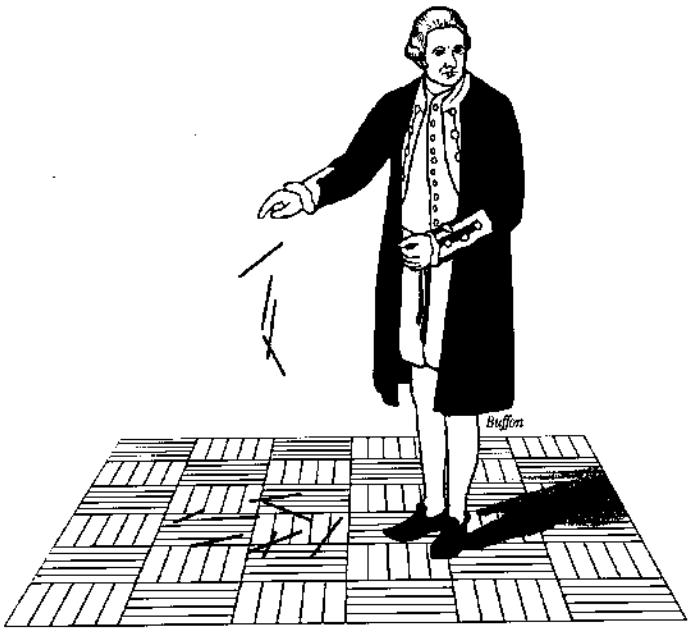

Figura 2.1: La Aguja de Buffon - Derechos de autor de la imagen en(1.1b)

τ de Gosset". Este acontecimiento abrió las puertas a 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. [7]

2.2.1. Periodo de formación (1945-1970)

a) **ENIAC y el método Montecarlo:**

A mediados de 1940 dos hechos sentaron las bases para la rápida evolución del campo de la simulación:

- La construcción de los primeros computadores de propósito general
- El trabajo de Stanislaw Ulam[4], John Von Neuman[5] y otros científicos para usar el

[4]Stanislaw Marcin Ulam (1909-1984). Matemático polaco que participo en el proyecto Manhattan y propuso el diseño Teller-Ulam de las armas termonucleares.

[5]John Von Neuman (1903-1957). Matemático húngaro-estadounidense que realizo contribuciones en física cuántica, análisis funcional y teoría de conjuntos, teoría de juegos, análisis de la computación, economía, análisis numérico, hidrodinámica, estadística etc. Considerado uno de los matemáticos mas importantes de la historia moderna

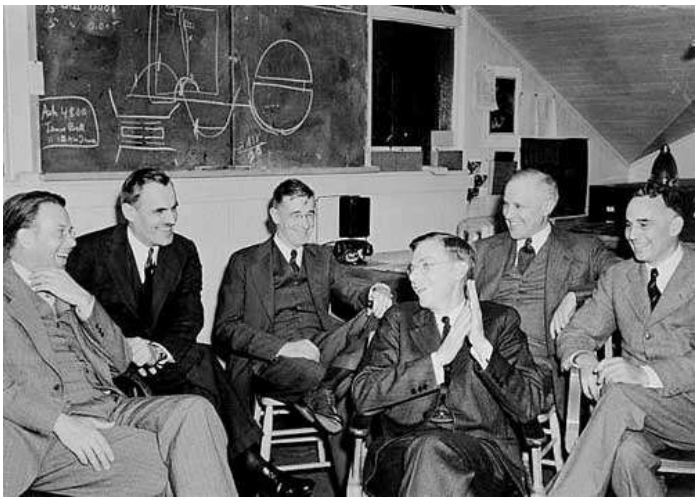

Figura 2.2: Proyecto manhattan - Derechos de autor de la imagenen en (1.2b)

método de Montecarlo en computadoras modernas de la época y solucionar problemas de difusión de neutrones en el diseño y desarrollo de la bomba de hidrogeno. Ulam y Von Neumann ya estuvieron presentes en el proyecto Manhattan en la figura 2.2. [7]

b) **El arte de la simulación:**

En 1960, Keith Douglas Tocher[6] desarrollo un programa de simulación general cuya principal tarea era la de simular el funcionamiento de una planta de producción donde las maquinas ciclaban por estados: ocupado, esperando, fallo y no disponible ; de manera que las sumulaciones en los cambios de estado de las maquinas marcan el estado definitivo de la producción de la planta. Este trabajo produjoademas el primer libro sobre simulación: "The art of simulation (1963). En la fig.(2.3) se muestra a Keith Douglas Tocher.

c) **Simulación de sistemas de propósito general y SIMSCRIPT:**

Entre 1960 y 1961 IBM desarrollo el sistema de simulación de propósito general o General Purpose Simulation System (GPSS). El GPSS se diseño para realizar simulaciones de

[6]Keith Douglas Tocher (1921-1981) Fue un informatico conocido por sus contribuciones a la simulación por computadora.

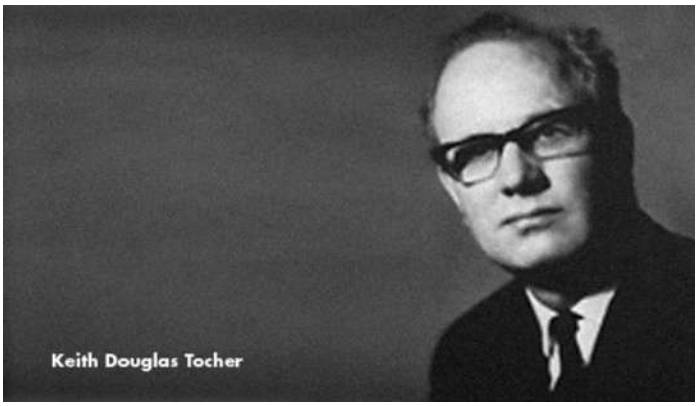

Figura 2.3: Keith Douglas Tocher - Derechos de autor de la imagenen en (1.3b)

teleprocesos involucrando por ejemplo: control de tráfico urbano, gestión de llamadas telefónicas, reservas de billetes de avión, etc. La sencillez uso de sistema lo popularizo como el lenguaje de simulación más usado de la época. Por otro lado en 1963 se desarrollo SIMSCRIPT, otra tecnología alternativa al GPSS basada en FORTRAN, mas enfocada a usuarios que no tenían porque ser expertos informáticos en RAND CORPORATION.

d) **SIMULA y WSC:**

Complementariamente a los desarrollos llevados a cabo por RAND e IBM, el Royal Norwegian Computing Center inicio en 1961 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 el WSC (Winter Simulation conference), Lugar donde desde entonces y hasta ahora se archivan los lenguajes de simulación y aplicaciones derivadas, siendo en la actualidad en lo que a avances en el campo de los sistemas de simulación se refiere. En la figura (2.4) se muestra el SIMULA en sus inicios.[7]

2.2.2. Periodo de expansión (1970-1981)

a) **Aplicaciones en múltiples campos:**

Durante este periodo se desarrollaron avanzadas herramientas de modelado y análisis de resultados. Gracias también a los desarrollos obtenidos en la generación de datos y a

Figura 2.4: Ole-Johan Dahl (1931-2002) y Kristen Nygaard (1926-2002) La primera definición formal de Simula 67 apareció en mayo de 1967 - Derechos de autor de la imagenen en (1.4b)

las técnicas de optimización y representación de datos, la simulación llega a su fase de expansión donde comienza a aplicarse en múltiples campos.

b) **Imágenes en Movimiento:**

Anteriormente, los datos de salida obtenidos de una simulación por computadora se representan en una tabla de matriz, de manera que se mostraba el efecto que los múltiples cambios en los parámetros tenían sobre los datos. El empleo del formato de matriz se debía al uso tradicional que se hacía de la matriz en los modelos matemáticos. Sin embargo en el desarrollo de las situaciones si miraban gráficos o incluso imágenes en movimiento o animaciones generadas a partir de dichos datos, como las que se ejecutan en las animaciones generadas a partir de dichos datos, como las que se ejecutan en animaciones de imágenes generadas por computadora.

2.2.3. Softwares de simulación actuales

1. **MATLAB**

MATLAB (abreviatura de MAtrix LAboratory, “laboratorio de matrices”) es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (Lenguaje M) Está— disponible para las plataformas Unix, Windows, Mac OS X y GNU/ Linux.

MATLAB combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arreglos directamente.[9] Entre sus prestaciones básicas se hallan la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets). MATLAB se puede utilizar en áreas como el Deep Learning, el procesamiento de señales e imágenes, los cálculos para biología computacional, los sistemas de comunicaciones, las finanzas computacionales, el diseño de controladores, el internet de las cosas, la analítica de datos y más. Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

Fue creado por el matematico y programador de computadoras Cleve Moler en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices LINPACK y EISPACK sin tener que usar Fortran.[9]

Figura 2.5: Logo Matlab - Derechos de autor de la imagenen en (1.5b)

2. **Modelica**

Modelica es un lenguaje de modelado orientado a objetos , declarativo y multidominio para el modelado orientado a componentes de sistemas complejos, por ejemplo, sistemas que contienen subcomponentes mecánicos, eléctricos, electrónicos, hidráulicos, térmicos, de control, de energía eléctrica o orientados a procesos. El esfuerzo de diseño de Modelica fue iniciado en septiembre de 1996 por Hilding Elmqvist. El objetivo era desarrollar un lenguaje orientado a objetos para el modelado de sistemas técnicos con el fin de reutilizar e intercambiar modelos de sistemas dinámicos en un formato estandarizado. Modelica 1.0 se basa en la tesis doctoral de Hilding Elmqvist y en la experiencia con los lenguajes de modelado Allan, Dymola , NMF ObjectMath, Omola, SIDOPS +, y Smile . Hilding Elmqvist es el arquitecto clave de Modelica, pero muchas otras personas también han contribuido. [10]

Figura 2.6: Logo Modelica - Derechos de autor de la imagenen en (1.5b)

3. **LabVIEW**

   LabVIEW es un software de ingeniería en sistemas que requieren pruebas, medidas y control acceso rápido a hardware e información de datos.

   LabVIEW ofrece un enfoque de programación gráfica que le ayuda a visualizar cada aspecto de su aplicación, incluyendo configuración de hardware, datos de medidas y depuración. Esta visualización hace que sea más fácil integrar hardware de medidas de cualquier proveedor, representar una lógica compleja en el diagrama, desarrollar algoritmos de análisis de datos y diseñar interfaces de usuario personalizadas.

   LabVIEW es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido.

   Este programa fue creado por National Instruments (1976) para funcionar en máquinas MAC, salió al mercado por primera vez en 1986, teniendo versiones disponibles para las plataformas Windows, UNIX, MAC y GNU/Linux actualmente. La penúltima versión es la 2013, con la increíble demostración de poderse usar simultáneamente para el diseño del firmware de un instrumento RF de última generación, a la programación de alto nivel del mismo instrumento, todo ello con código abierto. Y posteriormente la versión 2014 disponible en versión demo para estudiantes y profesional, la versión demo se puede descargar directamente de la página National Instruments. Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control de todo tipo de electrónica (Instrumentación electrónica) sino también a su programación embebida, comunicaciones, matemáticas, etc. Un lema tradicional de LabVIEW es: "La potencia está en el Software", que con la aparición de los sistemas multinúcleo se ha hecho aún más potente. Entre sus objetivos están el reducir el tiempo de desarrollo de aplicaciones de todo tipo (no sólo en ámbitos de Pruebas, Control y Diseño) y el permitir la entrada a la informática a profesionales de cualquier otro campo. LabVIEW consigue combinarse con todo tipo de software y hardware, tanto del propio fabricante -tarjetas de adquisición de datos, PAC, Visión, instrumentos y otro Hardware- como de otros fabricantes. Su principal carac-

Figura 2.7: Logo LabVIEW - Derechos de autor de la imagenen en (1.6b)

terística es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación pueden hacer programas relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. También es muy rápido hacer programas con LabVIEW y cualquier programador, por experimentado que sea, puede beneficiarse de él. Los programas en LabVIEW son llamados instrumentos virtuales (VIs) Para los amantes de lo complejo, con LabVIEW pueden crearse programas de miles de VIs (equivalente a millones de páginas de código texto) para aplicaciones complejas, programas de automatizaciones de decenas de miles de puntos de entradas/salidas, proyectos para combinar nuevos VIs con VIs ya creados, etc. Incluso existen buenas prácticas de programación para optimizar el rendimiento y la calidad de la programación. El labView 7.0 introduce un nuevo tipo de subVI llamado VIs Expreso (Express VIS). Estos son VIs interactivos que tienen una configuración de caja de diálogo que permite al usuario personalizar la funcionalidad del VI Expreso. El VIs estándar son VIs modulares y personalizables mediante cableado y funciones que son elementos fundamentales de operación de LabView.[14]

4. **Scilab**

Scilab es un software para análisis numérico, con un lenguaje de programación de alto nivel para cálculo científico. Es desarrollado por Scilab Enterprises, bajo la licencia CeCILL, compatible con la GNU General Public License. Las características de Scilab: Tiene un lenguaje de programación de alto nivel que permite el acceso a estructuras de datos, funciones avanzadas, análisis numérico, incluye cientos de fun-

ciones matemáticas, funciones gráficas y sus visualizaciones 2D y 3D, optimización, análisis estadístico, diseño y análisis de sistemas dinámicos, procesamiento de señales, e interfaces con Fortran, Java, C y C++. Mientras que la herramienta Xcos permite una interfaz gráfica para el diseño de modelos.

Scilab incluye una gran cantidad de funcionalidades: control, simulación, optimización, procesamiento de señal y Xcos, el modelador y simulador de sistemas dinámicos híbridos se proporciona con la plataforma.

Scilab es un software gratuito y de código abierto para ingenieros y científicos, con una larga historia (primer lanzamiento en 1994) y una comunidad en crecimiento (100 000 descargas cada mes en todo el mundo).

Los años 80, de Blaise a Basile

La historia del software Scilab comienza en los años 80, con Blaise, un software CACSD (Diseño de sistema de control asistido por computadora) creado en el IRIA (Instituto francés de investigación en ciencias y control informático) y desarrollado principalmente por François Delebecque y Serge Steer con el fin de proporcionar Una herramienta de control automático para investigadores. Fue inspirado por el software Matlab Fortran desarrollado por Cleve Moler, quien más tarde cofundó con la compañía John Little "The MathWorks".
En 1984, Blaise se convirtió en Basile y fue distribuido durante unos años por Simulog, la primera startup Inria (Instituto Nacional Francés de Investigación en Ciencias de la Computación y Control). [11]

Los años 90, nacimiento de Scilab

A principios de los 90, Simulog dejó de distribuir Basile. El nombre del software se convirtió en Scilab y luego fue desarrollado por Inria dentro del Grupo Scilab compuesto por los siguientes seis investigadores: Jean-Philippe Chancelier de la ENPC

(École Nationale des Ponts et Chaussées), François Delebecque, Claude Gomez, Maurice Goursat, Ramine Nikoukhah y Serge Steer de Inria.[11]

Figura 2.8: Logo de inria

Entonces Inria decidió distribuir Scilab como software gratuito de código abierto. Scilab 1.1, la primera versión lanzada de Scilab, se puso en sitio anónimo ftp el 2 de enero de 1994. El Grupo Scilab, con la colaboración activa de desarrolladores externos, desarrolló Scilab hasta finales de 2002 con la versión Scilab 2.7, distribuyendo código fuente y binario. Versiones en Internet. [11]

2003, El Consorcio Scilab

A principios de 2003, para tener en cuenta el mayor número de personas que descargaban y usaban Scilab, y para asegurar su futuro, desarrollo, mantenimiento, soporte y promoción, Inria decidió crear el Consorcio Scilab con el apoyo de empresas y organizaciones académicas.[11]

2008, El Consorcio Scilab (fase 2)

Naturalmente, el Consorcio Scilab integra la red de investigación Digiteo en 2008, para proporcionar un entorno apropiado para el crecimiento sostenido de la operación. El software Scilab había sido desarrollado, mantenido y promovido por el Consorcio Scilab dentro de Digiteo.[11]

Figura 2.9: Logo de Digiteo

También es desde 2008 que Scilab se distribuye bajo la licencia CeCILL , una licencia de código abierto compatible con GPL.

Figura 2.10: Logo de Scilab

2010, Scilab Enterprises

La empresa Scilab Enterprises se fundó en junio de 2010 con el apoyo de Inria, para garantizar el futuro de Scilab. Scilab Enterprises se ha encargado completamente de la edición y el desarrollo de Scilab desde julio de 2012.[11]

Figura 2.11: Logo scilab enterprises

Basado en el modelo de negocio clásico de código abierto, Scilab Enterprises también ofrece servicios profesionales y soporte en Scilab.[11]

2017, Grupo ESI

Después de 5 años de versiones Scilab hechas por Scilab Enterprises, el equipo operativo se une al Grupo ESI a través de la adquisición de la compañía.[11]

Figura 2.12: Logo ESI group

ESI Group es un proveedor pionero y líder mundial en creación virtual de prototipos, aprovechando la física de los materiales. El equipo sigue comprometido con un software Scilab gratuito y de código abierto para el cálculo numérico. Con la experiencia de

OpenFOAM , ESI Group ya ha demostrado su compromiso con el software de código abierto para ingenieros y científicos.[11]

5. **GNU octave**

GNU Octave es un lenguaje interpretado de alto nivel, destinado principalmente a cálculos numéricos. Proporciona capacidades para la solución numérica de problemas lineales y no lineales, y para realizar otros experimentos numéricos. También proporciona amplias capacidades gráficas para la visualización y manipulación de datos. GNU Octave se usa normalmente a través de su interfaz interactiva (CLI y GUI), pero también se puede usar para escribir programas no interactivos. El lenguaje GNU Octave es bastante similar a Matlab, por lo que la mayoría de los programas son fácilmente portátiles. GNU Octave es un lenguaje de alto nivel, destinado principalmente a cálculos numéricos. Proporciona una interfaz de línea de comandos convenientes para resolver problemas lineales y no lineales numéricamente, y para realizar otros experimentos numéricos utilizando un lenguaje que es principalmente compatible con Matlab. También se puede usar como un lenguaje orientado a lotes. Octave tiene amplias herramientas para resolver problemas comunes de álgebra lineal numérica, encontrar las raíces de ecuaciones no lineales, integrar funciones ordinarias, manipular polinomios e integrar ecuaciones diferenciales y algebraicas diferenciales ordinarias. Es fácilmente extensible y personalizable a través de funciones definidas por el usuario escritas en el propio lenguaje de Octave, o utilizando módulos cargados dinámicamente escritos en C ++, C, Fortran u otros idiomas. GNU Octave también es un software redistribuible libremente. Puede redistribuirlo y / o modificarlo según los términos de la Licencia Pública General de GNU (GPL) publicada por la Free Software Foundation .

Figura 2.13: Logo GNU Octave

Historia

Octave fue concebido originalmente (aproximadamente en 1988) como un software complementario para un libro de texto de nivel universitario sobre diseño de reactores químicos escrito por James B. Rawlings de la Universidad de Wisconsin-Madison y John G. Ekerdt de la Universidad de Texas. Originalmente imaginamos algunas herramientas muy especializadas para la solución de problemas de diseño de reactores químicos. Más tarde, después de ver las limitaciones de ese enfoque, optamos por intentar construir una herramienta mucho más flexible. Todavía había algunas personas que decían que deberíamos usar Fortran en su lugar, porque es el lenguaje informático de la ingeniería, pero cada vez que lo habíamos intentado, los estudiantes pasaron demasiado tiempo tratando de averiguar por qué falló su código Fortran y No hay suficiente tiempo para aprender sobre ingeniería química. Creemos que con un entorno interactivo como Octave, la mayoría de los estudiantes podrían aprender lo básico rápidamente y comenzar a usarlo con confianza en solo unas pocas horas. El desarrollo a tiempo completo comenzó en la primavera de 1992. La primera ver-

sión alfa fue el 4 de enero de 1993, y la versión 1.0 se lanzó el 17 de febrero, Desde entonces, Octave ha pasado por varias revisiones importantes, se incluye con Debian GNU / Linux , openSUSE y muchas otras distribuciones de GNU / Linux. Octave fue revisado en la edición de julio de 1997 del Linux Journal .[12]

Es de gran importancia mencionar algunos softwares más conocidos, que pueden llevar a cabo simulaciones de sistemas en general así como un poco de su historia. Regresando al contexto principal de este trabajo que es dar a conocer el software SIMNON, el cual se describe en la siguiente sección de este capitulo.

2.3. SIMNON

2.3.1. Introducción a SIMNON

El objetivo de esta sección y en general del capítulo es proporcionar una introducción al programa de simulación Simnon, además de brindar las herramientas base para la solución de sistemas continuos a través de ecuaciones diferenciales, es importante mencionar que Simnon es capaz de hacer mucho más, pero solo nos centraremos en esto.
Simnon es un lenguaje de programación especial para simular sistemas dinámicos. Los sistemas pueden describirse como ecuaciones diferenciales ordinarias, como ecuaciones de diferencia o como combinaciones de tales ecuaciones. Los modelos de este tipo son comunes en matemáticas, biología, economía y en muchas ramas de la ingeniería. Simnon requiere una computadora con una terminal gráfica. Los resultados se muestran como curvas en el terminal. El lenguaje tiene una implementación interactiva que facilita que un usuario trabaje con el sistema. Simnon puede usarse de una manera muy simple para encontrar soluciones a ecuaciones diferenciales o diferenciales. Esto requiere sólo seis comandos. Hay 38 comandos adicionales en el sistema. También permiten la optimización, la introducción de datos experimentales y el ajuste de parámetros". El propósito de este informe es proporcionar una introducción al lenguaje de simulación.[2]
Se recomienda ir experimentando en la pantalla a medida que avanza en la lectura. Se hace el Recordatorio que siempre puede usar el comando HELP y que se proporciona una des-

cripción detallada de la construcción del lenguaje. También recuerde que una buena manera de aprender Simnon es comenzar aprendiendo algunos comandos y expandir el vocabulario gradualmente. Todas las palabras específicas de Simnon están escritas en mayúsculas en este escrito, pero Simnon no distingue entre mayúsculas y minúsculas.[2]

2.3.2. ¿Para que sirve SIMNON?

Simnon es un lenguaje interactivo para simular sistemas dinámicos. El sistema puede describirse mediante ecuaciones diferenciales ordinarias o ecuaciones diferenciales. También es posible simular sistemas que consisten en subsistemas interconectados. Esto es útil para estructurar un sistema grande. Simnon también puede usarse para otros fines por ejemplo para graficar funciones para ajustar modelos a datos, etc. El lenguaje de simulación tiene facilidades para editar las descripciones del sistema, integrar ecuaciones diferenciales, almacenar y recuperar datos, mostrar las soluciones como gráficos y cambiar los parámetros y las condiciones iniciales.

Simnon está diseñado para resolver ecuaciones de diferencia y ecuaciones diferenciales ordinarias. Para simular sistemas dinámicos, los sistemas pueden describirse como una interconexión de subsistemas cuyo comportamiento se caracteriza por ecuaciones diferenciales o por ecuaciones de diferencia. Los modelos de este tipo son comunes en matemáticas, biología, economía y en muchas ramas de la ingeniería. Simnon tiene una implementación interactiva, lo que facilita que un usuario trabaje con el sistema. El usuario interactúa con el sistema escribiendo comandos, los parámetros, las condiciones iniciales y las descripciones del sistema se pueden modificar de forma interactiva.

Los resultados se muestran de forma gráfica o numérica en la pantalla. El diseño puede modificarse fácilmente y los resultados pueden documentarse utilizando una función de copia impresa. Simnon puede usarse de una manera muy simple para encontrar soluciones a ecuaciones diferenciales o de diferencia. Esto requiere solo seis comandos. Hay más de 40 comandos en Simnon. Además La función de macro incorporada le permite crear sus propios comandos (Solo se hace mención de esta función).[2]

2.3.3. ¿Que es la simulación?

Ahora bien recordemos un concepto fundamental antes de abordar el software. Como ya se ha descrito anteriormente la simulación que significa imitar el comportamiento de un sistema. A través de la simulación es posible realizar investigaciones sin tener acceso al sistema real. Se podría decir que es un experimento dentro de la computadora. En las simulaciones, se pueden estudiar los efectos de diferentes condiciones iniciales y valores de parámetros. Por ejemplo, en un sistema de control se pueden investigar diferentes estructuras reguladoras y configuraciones de parámetros. Cuando el rendimiento es satisfactorio en la simulación, el regulador puede ser probado en el proceso real. La simulación también es una buena ayuda para capacitar a los operadores. Por ejemplo, los pilotos de aviones se entrenan regularmente en simuladores.

Las simulaciones se pueden clasificar en: Un ejemplo típico de una simulación de evento discreto es la simulación de colas y sistemas operativos. Simnon está destinado a la simulación de tiempo continuo, así como a sistemas dinámicos de tiempo discreto. Las rutinas de integración numérica se utilizan para simular ecuaciones diferenciales. Las ecuaciones de diferencia que describen sistemas de tiempo discreto se resuelven al redactar las ecuaciones.[2]

2.3.4. Descripciones de sistemas dinámicos

Para usar Simnon es necesario tener una comprensión básica de los sistemas dinámicos. En particular para familiarizarse con las nociones de entrada, salida y estado. La forma genérica de un sistema de tiempo continuo es:

$$\begin{cases} \frac{dx}{dt} = f(x,u) \\ y = g(x,u) \end{cases} \tag{2.1}$$

donde u es un vector de entradas, y es un vector de salidas y x es el vector de estado. Un sistema de ecuaciones como (2.1) se especifica como un SISTEMA CONTINUO en

Simnon. La forma análoga para un sistema de tiempo discreto es:

$$\begin{cases} x(t_{k+1}) = f(x(t_k), u(t_k)) \\ y(t_k) = g(x(t_k), u(t_k)) \qquad k = 1, 2... \end{cases} \tag{2.2}$$

Un sistema de ecuaciones como (2.2) se especifica como UN SISTEMA DISCRETO. Simnon permite que un sistema o un subsistema sean descritos por cualquiera de las formas (2.1) o (2.2). También es posible tener sistemas interconectados donde cada subsistema tiene la forma de (2.1) o (2.2). Las conexiones se describen como un SISTEMA DE CONEXIÓN.

2.3.5. Principios de Interacción

Simnon proporciona información al usuario a través de una pantalla gráfica que puede mostrar curvas, texto y números. También es posible obtener una copia impresa de una imagen y enumerar las descripciones y datos del sistema. simnon recibe información del usuario mediante comandos del teclado. Los comandos tienen la forma:

```
CMND arg1 & arg2...
```

donde CMND es el nombre del comando y argl, arg2, etc. son los argumentos. El nombre es una combinación de hasta ocho caracteres. Los argumentos pueden ser identificadores o números. Los espacios se usan como separadores. Un comando es terminado por el retorno de carro (CR).

Valores Predeterminados

Es deseable que los comandos seann cortos y flexibles. Una posibilidad para lograr estos objetivos conflictivos es permitir variaciones de un comando que son seleccionados por los argumentos. Una descripción del comando de simulación ilustra la idea.

El comando de Simulación

Las diferentes formas del comando SIMU que ejecuta una simulación de un sistema se ilustran en el diagrama de sintaxis fugura 2.14. El diagrama implica que se permite

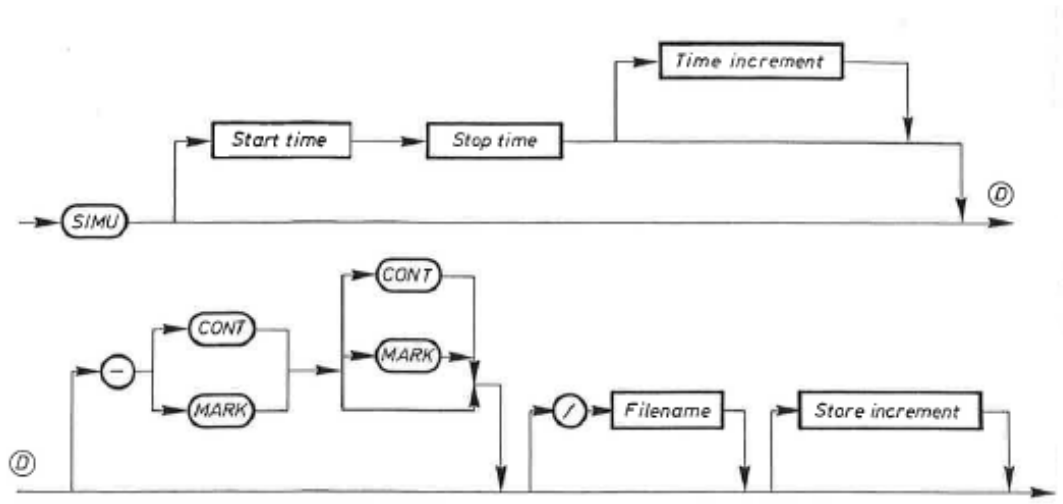

Figura 2.14: Diagrama de sintaxis para el comando SIMU

cualquier forma del comando que se obtiene atravesando el gráfico en las direcciones de las flechas. Por ejemplo el comando

```
SIMU  0 100
```

Simula un sistema de tiempo 0 a tiempo 100. Si queremos repetir la simulación por segunda vez con diferentes parámetros, es suficiente escribir

```
SIMU
```

Luego se utilizan los valores anteriores de los argumentos, es decir, 0 y 100. La mayoría de los comandos se proporcionan con argumentos predeterminados. Estos argumentos se utilizan a menos que se indique lo contrario.

De la figura 2.14 se deduce que se pueden especificar los tiempos de inicio y parada y el incremento de tiempo inicial. También es posible marcar curvas mediante el argumento opcional MARK.

Una simulación también puede continuar utilizando las condiciones finales de una simulación previa como valores iniciales. Esto se realiza mediante la opción de comando CONT. Los resultados de una simulación también pueden almacenarse en un archivo. El espacio de tiempo entre los valores almacenados se especifica por incremento.

La sintaxis para el comando de simulación también se puede describir de la siguiente manera

```
SIMU[<start time> <stop time>[<increment>]]
    [-{CONT | MARK}][/<filename>[increment>]]
```

Esta descripción se llama la forma Backus-normal (BNF), o la forma de Bachus-Naur. < ... > denota un argumento, es decir, un número o un identificador. [a1 a2] denota argumentos opcionales que pueden omitirse y $\{a1|a2|...\}$ denota que se debe elegir uno de los argumentos alternativos. Un asterisco (*) después de un argumento denota que puede repetirse. la sintaxis de cualquier comando se obtiene escribiendo el comando HELP seguido del nombre del comando. La sintaxis se muestra en la notación Backus-Naur. La sintaxis de los comandos se proporciona en el Apéndice A. [2]

2.3.6. Ecuaciones Diferenciales

La solución de ecuaciones diferenciales no lineales ordinarias es una aplicación simple de Simnon. En tales aplicaciones, Simnon puede ser visto como un calculador de ecuaciones diferenciales. Hay dos diferencias menores en comparación con una calculadora. Simnon puede mostrar curvas y en una calculadora una función se activa presionando una tecla dedicada. En Simnon, las funciones se activan escribiendo un comando en el teclado. Un ejemplo ilustra cómo se puede usar Simnon para generar soluciones a una ecuación diferencial ordinaria.[2]

2.3.7. Ejemplo aplicado en la solucion del problema de VAN DER POL

Supongamos que nos gustaría saber el carácter de la solución a la ecuación de Van der pol.

$$\frac{d^2y}{dt^2} + a(y^2 - b)\frac{dy}{dt} + y = 0 \tag{2.3}$$

para diferentes condiciones iniciales y diferentes valores de los parámetros a y b. La ecuación de van der Pol es un modelo para un oscilador electrónico. El camino hacia la solución del problema se puede dividir en los siguientes pasos. para diferentes condiciones iniciales y diferentes valores de los parámetros a y b. La ecuación de van der Pol es un modelo para

un oscilador electrónico. El camino hacia la solución del problema se puede dividir en los siguientes pasos.

Tabla 2.1: Pasos de la sintaxis basica del comando SIMU

1. ingrese las descripciones del sistema
2. simular
3. analizar los resultados
4. cambiar parámetros y condiciones iniciales

Donde los pasos 2,3 y 4 se repiten hasta obtener un resultado satisfactorio. Los diferentes pasos se describirán ahora con cierto detalle.

Tabla 2.2: Listado 1. Un sistema de Simnon para la ecuación (2.4)

```
continuous system VDPOL

"The van der Pol equation"

state x y

der dx dy

dy=x

dx=a*x* (b-y*y)-y

a:1

b:1

END
```

2.3.8. Ingresando a la descripcion del sistema

La ecuación (2.3) se reescribe primero en la forma de espacio de estado estándar (2.1). Como la ecuación (2.3) es de segundo orden, se introduce una variable adicional. La ecuación (2.3) se puede escribir como

$$\frac{dy}{dt} = x$$
$$\frac{dx}{dt} = ax(b - y^2) - y \tag{2.4}$$

Las ecuaciones ahora están en forma de espacio de estado estándar, que es el formato necesario para usar Simnon. Ahora se debe preparar un archivo que describa el sistema. Este archivo con la etiqueta VDPOL aparece en el Listado 1 de la tabla 2.2. La primera línea indica que es un sistema de tiempo continuo con el nombre VDPOL. Se declaran las variables de estado x e y sus derivadas dx y dy. Las ecuaciones diferenciales se definen entonces. Observe la gran similitud con (2.4). Finalmente, los valores se asignan a los parámetros a y b. Simnon se separa entre parámetros y variables. A los parámetros se les pueden asignar valores en una descripción del sistema utilizando la notación $'{:}'$ para la asignación. Los valores de los parámetros se pueden reasignar interactivamente utilizando el comando PAR. Las variables se definen usando la notación $'{=}'$.

El archivo se puede editar con cualquier editor con el que esté familiarizado. Suponiendo que el editor estándar escriba EDIT VDPOL.T si está en VMS y $ EDIT VDPOL.T si está en Simnon. En el sistema VAX / VMS, todos los archivos del sistema Simnon tienen la extensión ".T". Esto también se aplica a los archivos Macro. También hay un sencillo editor orientado a líneas integrado en simnon que se puede usar para ingresar el archivo. Este editor es invocado por el comando

```
EDIT <filename>
```

donde el argumento <filename> es un identificador, es decir, una letra posiblemente seguida de letras o dígitos. La lista de comandos aparece en el Apéndice A. La sintaxis de las descripciones del sistema se proporciona en el Apéndice B. El comando

```
LIST <filename>
```

enumera una descripción del sistema en la terminal.

2.3.9. Ejemplo de una simulación

Para ejecutar un sistema, primero debe estar activado. Esto lo hace el comando

```
SYST VDPOL
```

Si hay algún error durante la compilación, se muestra un mensaje de error y el sistema ingresa al editor de Simnon para que se pueda corregir el error. Si desea utilizar otro editor, simplemente escriba LEAVE para salir del editor de Simnon.

Si nos gustaría ver las curvas de solución como están integradas, primero debemos dibujar ejes en la pantalla. Esto se hace con el comando

```
AXES H 0 20 V -6 6
```

EL comando

```
PLOT  x  y
```

instruye al programa para trazar las variables x y y como funciones del tiempo.

Para realizar una simulación es necesario dar condiciones iniciales apropiadas a las variables de estado. El comando

```
INIT x:1
```

asigna el valor inicial 1 a la variable de estado x. los valores iniciales se establecen automáticamente en cero si no se realizan asignaciones. Ahora estamos listos para realizar una simulación. El comando

```
SIMU 0 20 - MARK
```

activa una simulación del tiempo 0 al tiempo 20, y se obtiene el resultado que se muestra en la figura 2.15. El argumento - MARK implica que las curvas están etiquetadas con enteros 1,2... en el orden en que aparecen en el comando PLOT.

2.3.10. Interrumpir una simulación

Algunas veces, cuando haces una simulación, encontrarás que los resultados son incorrectos al principio. Entonces es útil poder romper la simulación inmediatamente. Hay instalaciones para hacer esto en Simnon. Los detalles dependen de la implementación. En el sistema Vax estándar, la simulación se interrumpe escribiendo CTRL-C.

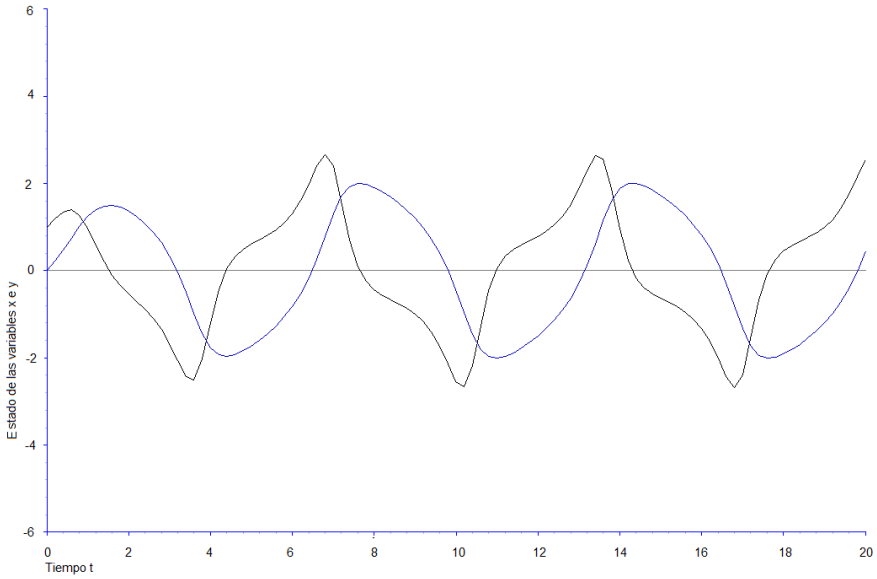

Figura 2.15: Simulación de la ecuación de van der Pol para a = 1 y b = 1 con valores iniciales x (0) = 1 e y (0) = 0.

2.3.11. Cambiar parametros

Suponga que es interesante explorar cómo el carácter de la solución está influenciado por los parámetros a y b. El comando

```
PAR b:2
```

Asigna el valor 2 al parámetro b. El comando

```
SIMU
```

Ahora genera una nueva simulación. Tenga en cuenta que no es necesario especificar ningún argumento en el comando SIMU. Los valores 0 y 20 dados anteriormente se usan automáticamente como valores predeterminados. El uso de valores predeterminados simplifica considerablemente la interacción del usuario.
Se pueden mostrar los valores actuales de todos los estados, derivados, variables y parámetros. El comando

```
DISP
```

Muestra todos los datos actuales. El comando realiza visualizaciones selectivas del parámetro a, el estado x y la variable y

```
DISP a x y
```

La pantalla también puede dirigirse a la impresora de líneas. Una forma sencilla de descubrir cómo funciona el comando DISP es escribir

```
HELP DISP
```

La función de ayuda se puede aplicar a todos los comandos

Almacenamiento y edición de resultados de una simulación

Puede ser útil almacenar algunos resultados, comparar resultados de diferentes simulaciones y trazar diferentes variables de estado en diferentes diagramas. Supongamos, por ejemplo, que nos gustaría comparar los resultados para los conjuntos de parámetros $a = 1$, $b = 1$ con $a = 1$, $b = 2$. Primero se generan dos archivos de datos. El comando

```
PAR a:1
PAR b:1
```

establece los parámetros. El comando

```
STORE x y
```

indica que las variables de estado x y y deben almacenarse. El comando

```
SIMU / B1
```

luego realiza una simulación y almacena x y y en un archivo llamado B1. El valor de b se establece en 2 por

```
PAR b: 2
```

y el comando

```
SIMU / B2
```

simula y almacena los resultados en el archivo B2. El comando

```
SPLIT 2 1
```

divide la pantalla en dos ventanas. El comando

```
ASHOW x / B2
```

traza la variable x del archivo B2 en la primera ventana usando la escala automática. El comando

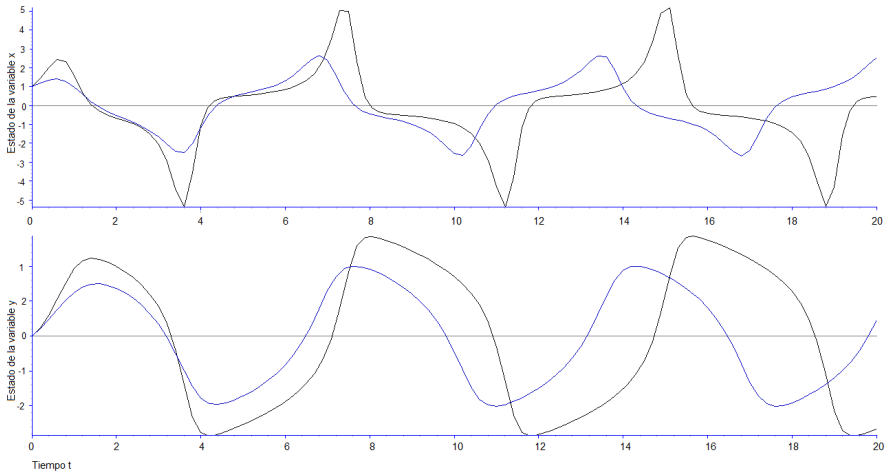

Figura 2.16: Solución de la ecuación de van der Pol para b = 1 y b = 2.

```
SHOW x / B1
```

traza la variable x del archivo B1 en la misma ventana. Los comandos

```
ASHOW y / B2
SHOW y / B1
```

traza la variable y de los archivos B2 y B1 en la segunda ventana. Para documentar los resultados es útil generar una copia impresa de las curvas obtenidas. Los detalles dependen del hardware. En una insallación normal. El comando

```
HCOPY
```

envía una copia de la imagen en la pantalla a la cola del trazador. Luego se obtienen los resultados mostrados en la figura 2.16. El comando

```
SPLIT 1 1
```

Borra la pantalla y restablece el trazado a una ventana que cubre la pantalla completa.

2.3.12. Gráficos de plano de fase

Para ecuaciones diferenciales bidimensionales es útil visualizar los resultados como gráficos de planos de fase. Esto puede hacerse simplemente por

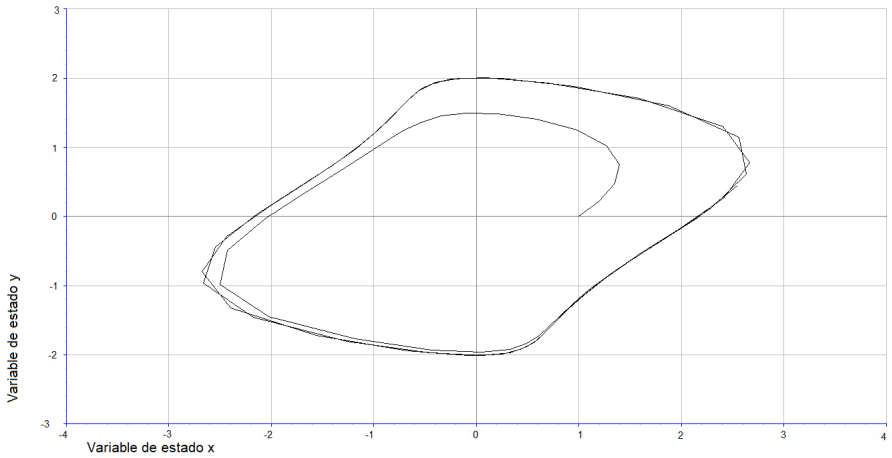

Figura 2.17: Gráfico del plano de fase de la ecuación de van der Pol para a = 1, b = 1, x (0) = 1, y (0) = 0.

```
AXES H -4 4 V -3 3
SHOW y(x) / B1
```

que genera la figura 2.17.

2.3.13. Generalidades

Ahora se dará la forma general de usar Simnon como una çalculadora para ecuaciones diferenciales". El genérico de una descripción del sistema se proporciona en el Listado 2 de la tabla 2.3. La descripción del sistema comienza con CONTINUOUS SYSTEM ¡Identifier¿. Está terminado por una línea que contiene END. Un identificador es una secuencia de letras y dígitos.

donde el primer carácter debe ser una letra. Se pueden usar letras mayúsculas y minúsculas, aunque el compilador no distingue entre ellas. La descripción del sistema tiene dos partes, una declaración y un cuerpo.

Hay tres tipos de declaraciones. Se puede declarar una variable de tiempo para la simulación de ecuaciones diferenciales que varían en el tiempo. Esto se realiza mediante la palabra clave TIME seguida de un identificador. Las variables de estado y sus derivados se declaran mediante las palabras clave STATE y DER, seguidas de una lista de las variables de estado y las derivadas asociados por su orden secuencial en las listas.

Tabla 2.3: Listado 2. Forma genérica de descripción del sistema para la simulación de ecuaciones diferenciales.

CONTINUOUS SYSTEM <Identifier>	
General diferential equation } comentario	
$state < Identifier > ... < Identifier >$	declaraciones
$der < Identifier > ... < Identifier >$	declaraciones
$time < Identifier >$	declaraciones
computation of auxiliary variables	Cuerpo
computation of assignment	Cuerpo
parameter assignment	Cuerpo
initial value assignment	Cuerpo
END	

El cuerpo de la descripción del sistema especifica las derivadas de las variables de estado en términos de variables de estado y parámetros. También se pueden usar variables auxiliares. El cuerpo también contiene la asignación de parámetros y valores iniciales. El orden de las declaraciones en el cuerpo no es importante. Una variable solo se puede definir una vez.
Hay tres tipos de declaraciones. Se puede declarar una variable de tiempo para la simulación de ecuaciones diferenciales que varían en el tiempo. Esto se realiza mediante la palabra clave TIME seguida de un identificador. Las variables de estado y sus derivados se declaran mediante las palabras clave STATE y DER, seguidas de una lista de las variables de estado y las derivadas. asociados por su orden secuencial en las listas.
El cuerpo de la descripción del sistema especifica las derivadas de las variables de estado en términos de variables de estado y parámetros. También se pueden usar variables auxiliares. El cuerpo también contiene la asignación de parámetros y valores iniciales. El orden de las declaraciones en el cuerpo no es importante. Una variable solo se puede definir una vez. [2]

Expresiones y operadores

Las expresiones disponibles en Simnon son similares a las de un lenguaje de procedimiento como Algol o Pascal. Una expresión puede ser una cadena, una constante numérica o una variable. También puede ser combinaciones de variables, operadores y funciones. Las expresiones condicionales de la forma IF... THEN... ELSE también están permitidas. Simnon tiene operadores aritméticos, relacionales y lógicos. Todas las variables son números

de coma flotante. Los números se escriben de forma convencional como 4, 1.1 o 6E7. El resultado de las expresiones booleanas es 1.0 si es verdadero y 0.0 si es falso.

Operadores aritméticos

Los operadores aritméticos son suma, resta, multiplicación, división y exponenciación. Se denotan como
+ - * / ^

Respectivamente

Operadores de relación

=><

Operadores lógicos

Los operadores lógicos son AND, OR y NOT.

Funciones

Las siguientes funciones están disponibles en la tabla 2.4.

2.4. Resumen

Se ha demostrado que Simnon puede usarse para generar soluciones a ecuaciones diferenciales ordinarias de una manera muy simple. Para hacer esto, las ecuaciones diferenciales se escriben primero como un sistema de ecuaciones de primer orden como

$$\frac{dx}{dt} = f(x, t).$$

Luego se genera un sistema de tiempo continuo en Simnon declarando las variables de estado y las derivadas. La función f, que define el lado derecho de la ecuación diferencial, se introduce utilizando expresiones matemáticas ordinarias y asignaciones de parámetros. No hay notación vectorial, por lo que todas las ecuaciones deben escribirse en notación escalar.

Tabla 2.4: Tabla de funciones en SIMNON

abs(x)	valor absoluto
sing(x)	$\begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$
int(x)	entero largo menor que x
mod(x.y)	x mod y
max(x,y)	El valor mayor de x e y
min(x,y)	El valor mayor de x e y
sqrt(x)	raiz cuadrada de x, x $\geqslant 0$
exp(x)	funcion exponencial
ln(x)	logaritmo natural de x
log(x)	logaritmo (base 10) de x
sin(x)	función seno de x (x es en radianes)
cos(x)	función coseno de x (x es en radianes)
tan(x)	función tangente de x (x es en radianes)
arcsin(x)	función arcoseno de x
arccos(x)	función arcocoseno de x
atan(x)	función arcotangente de x el resultado en$(-\pi/2, \pi/2)$
atan2(x,y)	función arcotangente de x/y $(-\pi, \pi)$
sinh(x)	función seno hiperbolico de x
cosh(x)	función coseno hiperbolico de x
tanh(x)	función tangente hiperbolico de x

Se puede usar cualquier editor. También hay un editor especial incorporado en Simnon, que es invocado por el comando EDIT. El comando LIST puede usarse para enumerar archivos. Una simulación se ejecuta utilizando seis comandos básicos: SYST, AXES, PLOT, INIT, PAR y SIMU. Sin embargo, para editar, manipular y documentar resultados de varias simulaciones diferentes, también es útil usar seis comandos adicionales, a saber, STORE, SHOW, DISP, SPLIT, AREA y HCOPY.

El comando HELP es útil para ver qué hacen los comandos.

Capítulo 3

MODELADO DE SISTEMAS CONTINUOS

3.1. Introducción

Una característica de casi cualquier sistema físico es su capacidad de aceptar entradas como tensión, corrientes, fuerza, presión, desplazamiento, etc., y producir una salida en respuesta a esa entrada. Por ejemplo, un receptor radar es un sistema electrónico cuya entrada es la reflexión de una señal electromagnética en un blanco, y cuya salida es una señal de vídeo que se muestra en la pantalla del radar. Otro ejemplo puede ser un robot, que es un sistema cuya entrada es una señal eléctrica de control y cuya salida es un determinado movimiento o una determinada acción realizada por el robot. Un tercer ejemplo es un filtro, cuya entrada es una señal distorsionada por ruido e interferencias, y cuya salida es la señal deseada. En pocas palabras, un sistema se puede ver como un proceso que transforma señales de entrada en otras señales de salida. Nuestro interés se centrará por lo pronto solo en sistemas en tiempo continuo y se hará mención de los sistemas en tiempo discreto. Un sistema en tiempo continuo es un sistema en el que señales de entrada en tiempo continuo se transforman en señales de salida en tiempo continuo. Un sistema de este tipo se representa gráficamente como muestra la figura 3.1(a), donde $x(t)$ es la entrada e $y(t)$ es la salida. Un sistema en tiempo discreto es un sistema que transforma entradas en tiempo discreto en

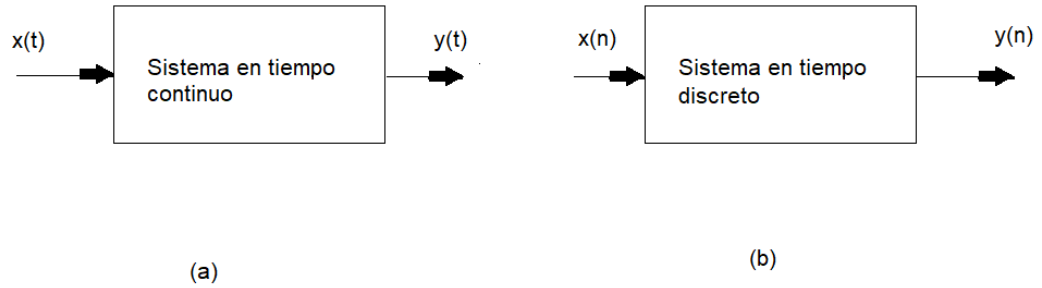

Figura 3.1: Ejemplos de sistemas a) en tiempo continuo y b) tiempo discreto

salidas en tiempo discreto [véase figura 3.l (b)]. El resultado se puede expresar matemáticamente en el dominio del tiempo, ademas Se demuestra que el análisis de sistemas lineales se puede reducir al estudio de la respuesta del sistema a ciertas señales básicas de entrada.[3]

3.2. Clasificacion de sistemas en tiempo continuo

En esta sección se pretende profundizar en el concepto de sistema presentando una clasificación atendiendo a la interacción del sistema con la señal de entrada. Esta interacción, que define el modelo de sistema, puede ser lineal o no lineal, variante con el tiempo o invariante con el tiempo, con memoria o sin memoria. La mayor parte de las veces, nuestro interés estará centrado en sistemas lineales, invariantes. En esta sección examinaremos brevemente las propiedades de cada una de esas clases.[3]

3.2.1. Sistemas lineales y no lineales

Cuando un sistema es lineal se puede aplicar el principio de superposición. Este hecho tan importante ha sido la causa de que se hayan desarrollado tanto las técnicas de análisis de sistemas lineales. El principio de superposición establece simplemente que la

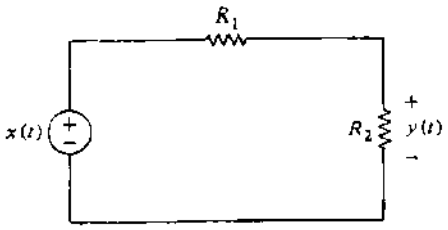

Figura 3.2: El sistema del ejemplo 3.1

respuesta de un sistema a una suma de señales de entrada es la suma de las respuestas del sistema a cada señal de entrada por separado. Matemáticamente, el principio de superposición se puede enunciar como sigue: sea $y_l(t)$ la respuesta de un sistema a la entrada $x_1(t)$, e $y_2(t)$ la respuesta correspondiente a la entrada $x_2(t)$. El sistema es lineal (es decir, sigue el principio de superposición) si

1. La respuesta a es $x_1(t) + x_2(t)$ es $y_l(t) + y_2(t)$, y

2. La respuesta a es $\alpha x_1(t)$ es $\alpha yl(t)$, siendo α una constante arbitraria.

La primera propiedad se denomina propiedad de aditividad, y la segunda, propiedad de homogeneidad. Estas dos propiedades que definen un sistema lineal se pueden combinar en una sola

$$\alpha x_1(t) + \beta x_2(t) \rightarrow \alpha y_l(t) + \beta y_2(t) \tag{3.1}$$

en donde la notación $x(t) \rightarrow y(t)$ representa la relación entrada/salida del sistema en tiempo continuo. Se dice que un sistema es no lineal si la relación (3.1) no es válida para al menos un conjunto de $x_1(t), x_2(t), \alpha$ y β.

Ejemplo 3.1 Consideremos el divisor de tensión que se muestra en la Figura 3.2 con Rl = R2, Para cualquier entrada $x(t)$ y salida $y(t)$ se trata de un sistema lineal. La relación entrada/salida se puede escribir explícitamente

$$y(t) = \frac{R_2}{R_1 + R_2} x(t) = \frac{1}{2} x(t) \tag{3.2}$$

es decir, la transformación consiste únicamente en multiplicar por una constante. Para demostrar que el sistema es realmente lineal, debemos demostrar que satisface la Ecuación (3.1).Consideremos la entrada $x(t) = \alpha x1(t)+\beta x2(t)$. La correspondiente salida es

$$\begin{aligned} y(t) &= \frac{1}{2}x(t) \\ &= \frac{1}{2}[ax_1(t) + bx_2(t)] \\ &= a\frac{1}{2}x_1(t) + b\frac{1}{2}x_2(t) \\ &= ay_1(t) + by_2(t) \end{aligned} \tag{3.3}$$

en donde

$$y_1(t) = \frac{1}{2}x_1(t) \tag{3.4}$$

$$y_2(t) = \frac{1}{2}x_2(t) \tag{3.5}$$

Por otra parte, si R_1 fuera una resistencia dependiente de la tensión como $R1 = R_2x(t)$ el sistema sería no lineal. En este caso, la relación entrada/salida se puede escribir así

$$\begin{aligned} y(t) &= \frac{R_2}{R_2x(t) + R_2}x(t) \\ &= \frac{x(t)}{x(t) + 1} \end{aligned} \tag{3.6}$$

para una entrada de la forma de la ecuación 3.1

$$ax_1(t) + bx_2(t)$$

la salida seria

$$y(t) = \frac{ax_1(t) + bx_2(t)}{ax_1(t) + bx_2(t) + 1} \tag{3.7}$$

el sistema en este caso es no lineal ya que

$$\frac{ax_1(t) + bx_2(t)}{ax_1(t) + bx_2(t) + 1} \neq a\frac{x_1(t)}{x_1(t) + 1} + b\frac{x_2(t)}{x_2(t) + 1} \tag{3.8}$$

para determinados valores de$x_1(t)$, $x_2(t)$, a, y b [por ejemplo, $x_1(t) = x_2(t)$, y $a = 1, b = 2$].

El concepto de linealidad es muy importante en teoría de sistemas. El principio de superposición se puede utilizar para determinar la respuesta de un sistema lineal a cualquier entrada arbitraria que se pueda descomponer en una suma (que puede tener infinitos términos) de señales básicas. La respuesta a cada señal básica se puede calcular por separado, y después sumar todas las respuestas para obtener la respuesta global del sistema. Y en muchos casos produce resultados en forma matemáticamente cerrada, lo que no es posible para sistemas no lineales. Muchos sistemas físicos, cuando se analizan con detalle, muestran un comportamiento no lineal. En estas situaciones, se puede encontrar una solución para una excitación y condiciones iniciales dadas, bien analíticamente, o con la ayuda del ordenador. Frecuentemente, se necesita determinar el comportamiento del sistema en las cercanías de esa solución. Una técnica común para resolver este problema es aproximar el sistema por un modelo lineal que sea válido en las cercanías del punto de trabajo. Esta técnica se conoce como linealización. Algunos ejemplos importantes son la técnica de análisis de pequeña señal que se aplica a circuitos de transistores, y el modelo de pequeña señal, que se aplica al análisis del péndulo simple.[3]

3.2.2. Sistemas variantes e invariantes con el tiempo

Sé dice que un sistema es invariante con el tiempo si un desplazamiento temporal de la señal de entrada causa un desplazamiento temporal idéntico en la señal de salida. Concretamente, si $y(t)$ es la salida correspondiente a la entrada $x(t)$, un sistema invariante con el tiempo producirá como salida $y(t - t_0)$ cuando la entrada sea $x(t - t_0)$. Es decir, la regla que se utiliza para obtener la salida del sistema no depende del instante en el que se aplica la entrada. El procedimiento para comprobar si un sistema es invariante con el tiempo se resume en los siguientes pasos: [3]

1. Sea $y_1(t)$ la salida correspondiente a $x_1(t)$.

2. Consideremos una segunda entrada, $x_2(t)$, obtenida desplazando $x_1(t)$
   $x2(t) = x_1(t - t_0)$

y encontremos la salida $y_2(t)$ correspondiente a la entrada $x_2(t)$.

3. Obtengamos la señal $y_1(t-t_0)$ a partir de la señal $y_1(t)$ (del paso 1), y comparémosla con $y_2(t)$.

4. Si $y_2(t) = y_1(t-to)$, el sistema es invariante con el tiempo. Caso contrario es variante.

Ejemplo 3.2 Deseamos determinar si los sistemas descritos por las ecuaciones que siguen son invariantes con el tiempo:

(a) $y(t) = cosx(t)$

(b) $\frac{dy(t)}{dt} = -ty(t) + x(t), \qquad t \geqslant 0, y(0) = 0$

Consideremos el sistema del apartado (a), $y(t) = cosx(t)$. Seguimos los pasos indicados anteriormente:

1. para la entrada $x_1(t)$, la salida es

$$y_1(t) = cosx_1(t) \tag{3.9}$$

2. Consideremos la segunda entrada $x_2(t) = x_1(t - t_0)$. La salida correspondiente es

$$\begin{aligned} y_2(t) &= cosx_2(t) \\ &= cosx_1(t - t_0) \end{aligned} \tag{3.10}$$

3. de la ecuación 3.2

$$y_1(t - t_0) = cosx_1(t - t_0) \tag{3.11}$$

4. La comparación de las Ecuaciones (3.10) y (3.——) demuestra que el sistema $y(t) = cosx(t)$ es invariante con el tiempo.

Consideramos ahora el sistema del apartado (b).

1. Si la entrada es $x_1(t)$, se puede verificar muy fácilmente por sustitución directa en la ecuación diferencial que la salida $y_1(t)$ viene dada por

$$y_1(t) = \int_0^1 exp\left[-\frac{t^2}{2} + \frac{\tau^2}{2}\right] x_1(\tau)d\tau \tag{3.12}$$

2. Consideremos la entrada $x_2(t) = x_1(t - t_0)$. La salida correspondiente es

$$\begin{aligned} y_2(t) &= \int_0^1 exp\left[-\frac{t^2}{2} + \frac{\tau^2}{2}\right] x_2(\tau)d\tau) \\ &= \int_0^1 exp\left[-\frac{t^2}{2} + \frac{\tau^2}{2}\right] x_1(\tau - t_0)d\tau \\ &= \int_{t_0}^{t-t_0} exp\left[-\frac{t^2}{2} + \frac{(\tau + t_0)^2}{2}\right] x_1(\tau)d\tau \end{aligned} \tag{3.13}$$

3. de la ecuación 3.5

$$y_1(t - t_0) = \int_{t_0}^{t-t_0} exp\left[-\frac{(t - t_0)^2}{2} + \frac{\tau^2}{2}\right] x_1(\tau)d\tau \tag{3.14}$$

4. Comparando las Ecuaciones (3.13) y (3.14) llegamos a la conclusión de que el sistema no es invariante con el tiempo.

3.2.3. Sistemas con memoria y sin memoria

En la mayoría de los sistemas, las entradas y las salidas son funciones de la variable independiente. Se dice que un sistema es sin memoria o instantáneo, si el valor actual de la salida depende solamente del valor actual de la entrada. Por ejemplo, una resistencia es un sistema sin memoria, ya que si la entrada $x(t)$ es la corriente que circula por la resistencia y la salida $y(t)$ es la tensión entre los extremos de la resistencia, la relación entrada/salida es

$$y(t) = Rx(t) \tag{3.15}$$

Siendo el valor de la resistencia. Por tanto, el valor de $y(t)$ en cualquier instante depende sólo del valor de $x(t)$ en ese instante. Por el contrario, un condensador es un ejemplo de sistema con memoria. Si la entrada es la corriente que circula por el condensador y la salida es la tensión entre sus extremos, la relación entrada/salida es en este caso

$$y(t) = \frac{1}{C}\int_{\infty}^{t} x(\tau)d\tau \tag{3.16}$$

siendo C el valor de la capacitancia. Es obvio que la salida en cualquier instante t depende de la historia pasada completa de la entrada. Si un sistema es sin memoria o instantáneo, la relación entrada/salida se puede poner de la forma

$$y(t) = F(x(t)) \tag{3.17}$$

Para sistemas lineales, la relación se reduce a

$$y(t) = k(t)x(t) \tag{3.18}$$

y si el sistema es invariante con el tiempo, tenemos

$$y(t) = k(t)x(t) \tag{3.19}$$

siendo k una constante. Un ejemplo de sistema lineal, invariante con el tiempo y sin memoria es un amortiguador mecánico. La dependencia lineal entre la fuerza $f(t)$ y la velocidad $v(t)$ es

$$v(t) = \frac{1}{D} f(t) \tag{3.20}$$

siendo D la constante de amortiguamiento.

Un sistema cuya respuesta en el instante t está determinada completamente por las señales de entrada durante los últimos T segundos (es decir, en el intervalo desde $t - T$ hasta t), se denomina sistema con memoria finita. La memoria de un sistema de este tipo es de longitud T unidades de tiempo.

3.3. Sistemas lineales e invariantes en el tiempo y sus propiedades

En la sección anterior hemos estudiado diversas propiedades de sistemas. Dos de ellas, la linealidad y la invarianza con el tiempo juegan un papel fundamental en el análisis de señales y sistemas, debido a que muchos fenómenos físicos se pueden modelar mediante sistemas lineales invariantes con el tiempo, y debido también a que el análisis matemático del comportamiento de esos sistemas se puede realizar por procedimientos muy directos. [3]

3.3.1. Propiedades de los sistemas LTI

En esta seccion se hara mención y se dara una breve descripción de las propiedades de los sistemas lineales e invariantes en el tiempo.[3]

Propiedad de memoria de los sistemas LTI

Definíamos los sistemas sin memoria como aquellos en los que la salida en cualquier instante depende sólo del valor de la entrada en ese mismo instante. [3]

Sistemas LTI causales

La salida de un sistema causal depende solamente de los valores presentes y pasados de la entrada. Utilizando la integral de convolución, podemos relacionar esta propiedad con una propiedad equivalente de la respuesta al impulso de un sistema LTI.[3]

Sistemas LTI invertibles

Mencionamos que un sistema es invertible sólo si se puede diseñar un sistema inverso que cuando se conecta en cascada con el sistema original produce una salida igual a la entrada del sistema inicial.[3]

Sistemas LTI estables

Un sistema en tiempo continuo es estable si y sólo si cualquier entrada acotada produce una salida acotada.[3]

3.4. Sistemas descritos por ecuaciones diferenciales

Quizá la parte más importante de este capítulo es hacer la representación de los sistemas físicos en modelos matemáticos a través de ecuaciones diferenciales. Para ello se utiliza el método de espacios de estado pero de una forma parcial sin llegar a la representación matricial que implica este método. Y se dice que es muy importante puesto que a partir de aquí obtenemos las ecuaciones que introduciremos en el software SIMNON en el siguiente capítulo.

3.4.1. Metodo de espacios de estado parcial

Ahora, procedemos a establecer el método en el espacio de estados un como método para representar los sistemas físicos. En esta sección se prepara el escenario para la definición

formal de la representación en el espacio de estados al hacer algunas observaciones acerca de los sistemas y sus variantes.

- **Estado.** El estado de un sistema dinámico es el conjunto de variables más pequeño (llamadas variables de estado), de forma que el conocimiento de estas variables en $t = t_0$, junto con el conocimiento de la entrada para $t \geqslant t_0$, determinan completamente el comportamiento del sistema en cualquier $t \geqslant t_0$. Obsérvese que el concepto de estado no está limitado a sistemas físicos. Es aplicable a sistemas biológicos, sistemas económicos, sistemas sociales y otros.[5]

- **Variables de estado.** Las variables de un sistema dinámico son las variables que constituyen el menor conjunto de variables que determinan el estado del sistema dinámico. Si al menos se necesitan n variables $x1, x2, ..., x_n$ para describir completamente el comportamiento de un sistema dinámico (de forma que una vez que la entrada para $t \geqslant t_0$ está dada y el estado inicial en $t = t_0$ está especificado, el estado futuro del sistema está determinado completamente), entonces tales n variables son un conjunto de variables de estado.

  Obsérvese que las variables de estado no necesitan ser físicamente medibles o cantidades observables. Se pueden seleccionar como variables de estado variables que no representan cantidades físicas y aquellas que no son medibles ni observables. Tal libertad en la elección de las variables de estado es una ventaja de los métodos en el espacio de estados. Sin embargo, prácticamente es conveniente seleccionar para las variables de estado cantidades físicamente medibles, si esto es posible, porque las leyes de control óptimo requerirán realimentar todas las variables de estado con una ponderación adecuada.[5]

- **Vector de estado.** Si se necesitan n variables de estado para describir completamente el comportamiento de un sistema dado, entonces esas n variables de estado se pueden considerar como las n componentes de un vector $\mathbf{x}$. Este vector se denomina vector de estado. Un vector de estado es, por lo tanto, un vector que determina unívocamente el estado del sistema $x(t)$ en cualquier instante del tiempo $t \geqslant t_0$, una vez que se conoce el estado en $t = t_0$ y se especifica la entrada $u(t)$ para $t \geqslant t_0$.[5]

- **Ecuaciones en el espacio de estados.** En el análisis en el espacio de estados se centra la atención en los tres tipos de variables que aparecen en el modelado de los sistemas dinámicos; las variables de entrada, las variables de salida y las variables de estado, la representación en el espacio de estados de un sistema dado no es única, salvo que el número de variables de estado es el mismo para cualquiera que sea la representación en variables de estado de un mismo sistema.[5]

Aún cuando se emplean redes eléctricas para ilustrar los conceptos, se pudo haber usado un sistema mecanico o cualquier otro sistema físico.
A continuación demostraremos que para un sistema con muchas variantes, como voltaje de un inductor voltaje de un resistor y carga de un capacitor necesitamos usar ecuaciones diferenciales sólo para despejar un subconjunto seleccionado de variables del sistema, porque se pueden evaluar algebraicamente todas las otras variantes restantes del sistema a partir de las variantes del subconjunto. Nuestros ejemplos toman el siguiente método:[6]

1. Seleccionamos un subconjunto particular de todas las posibles variantes del sistema, y a las variantes de este subconjunto las llamamos variantes de estado.

2. Para un sistema de orden n, ecuaciones diferenciales simultaneas de primer orden en términos de las variantes de estado. Este sistema de ecuaciones diferenciales simultaneas se denomina ecuaciones de estado.

3. Si conocemos las condiciones iniciales de todas las variantes de estado como en i_0, asi como la entrada del sistema para $t \geqslant t_0$ de las ecuaciones diferenciales simultáneas es posible despejar las variantes de estado para$t \geqslant t_0$

4. Algebraicamente combinamos las variantes de estado con la entrada del sistema y encontramos todas las otras variantes del sistema para $t \geqslant t_0$ Esta ecuación algebraica se llama ecuación de salida.

5. Consideramos que las ecuaciones de estado las ecuaciones de salida son una representación viable del sistema. Esta representación del sistema se llama representación en el espacio de estado.

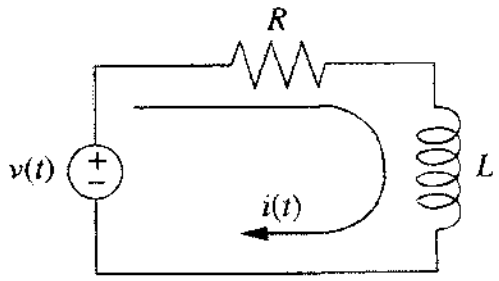

Figura 3.3: Circuito RL.

Aplicando estos pasos al **ejemplo 3.3** Considere el circuito RL que se muestra en la figura 3.3, con una corriente inicial de i(0)

1. Seleccionamos la corriente, i(t), para la cual escribiremos y despejaremos una ecuación diferencial usando transformadas de Laplace,

2. Escibimos la ecuación de malla

$$L\frac{di}{dt} + R_i = v(t) \tag{3.21}$$

3. al usar la transformada de Laplace e incluyendo las condiciones iniciales tenemos

$$L[sI(s) - i(0)] + RI(s) = V(s) \tag{3.22}$$

Si se supone que la entrada, $v(t)$, debe ser un escalón unitario, $u(t)$ cuya transformada de $V(s) = 1/s$ despejamos $I(s)$ y obtenemos

$$I(s) = \frac{1}{r}\left(\frac{1}{s} - \frac{1}{s + \frac{R}{L}}\right) + \frac{i_0}{s + \frac{R}{L}} \tag{3.23}$$

de la cual

$$i(t) = \frac{1}{R}(1 - e^{-(R/L)t}) + i(0)e^{(R/L)}t \tag{3.24}$$

La función $i(t)$ es un subconjunto de todas las posibles variantes del circuito que estamos en posibilidad de hallar de la ecuación (3.24) si conocemos condición inicial,

$i(0)$, y la entrada, $v(t)$. Así, $i(t)$ es una variante de estado y la ecuación diferencial (3.21), una ecuación de estado.

4. Ahora podemos despejar algebraicamente todas las otras variantes del circuito en términos de $i(t)$ y el voltaje aplicado, $v(t)$. Por ejemplo, el voltaje entre los terminales del resistor es

$$v_R(t) = R_i(t) \tag{3.25}$$

El voltaje en los terminales del inductor es

$$v_L(t) = v(t) - R_i(t) \tag{3.26}$$

La derivada de la corriente es

$$\frac{di}{dt} = \frac{1}{L}[v(t) - R_i(t)] \tag{3.27}$$

Entonces al conocer la variante de estado $i(t)$, y la entrada, $v(t)$, podemos hallar el valor, o estado, de cualquier variante de la red en cualquier tiempo, $t \geqslant t_0$. En consecuencia, las ecuaciones algebraicas, ecuaciones de la (3.25) a la (3.27), son ecuaciones de salida de salida.

5. Como las variantes de interés están descritas por completo por la ecuación (3.21) y las ecuaciones de la (3.25) a la (3.27), decimos que la ecuación combinada de estado (3.21) y las ecuaciones (3.25 a la 3.27) forman una representación viable de la red, que llamamos presentación en el espacio de estados.

La ecuación (3.21), que describe la dinámica de la red, no es única. Podría escribirse esta ecuación en términos de cualquier otra variante del curcuito. Por ejemplo, al sustituir $i = v_R/R$ en la ecuación 3.21 produce

$$\frac{L}{R}\frac{dv_R}{dt} + v_R = v(t) \tag{3.28}$$

que se puede resolver si la condición inicial $v_R(0) = R_i(0)$ y se conoce $v(t)$. Del mismo modo, se pueden escribir ahora todas las otras variantes de red en términos de variante de estado $v_R(t)$, y la entrada,$v_(t)$ ahora para hacer un poco mas complejo el estudio se desarrolla un sistema de segundo orden como el que se ilustra en la figura 3.4.

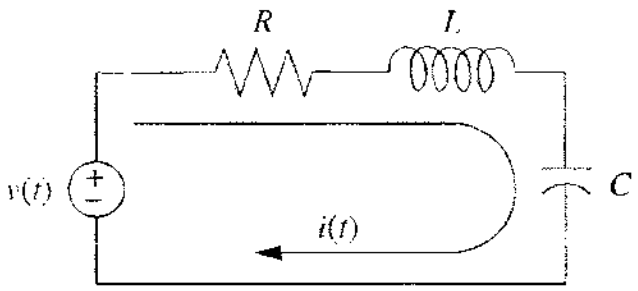

Figura 3.4: Circuito RLC.

1. Debido a que el circuito es de segundo orden, se necesitan dos ecuaciones diferenciales simultáneas de primer orden para despejar las dos variantes de estado. Seleccionarnos $i(t)$ y $q(t)$, la carga en el capacitor, como las dos variables de estado.

2. escribir ecuación de malla resulta

$$L\frac{di}{dt} + R_i + \frac{1}{c}\int idt = v(t) \tag{3.29}$$

Si convertimos a carga, con $i(t) = dq/dt$, obtenemos

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{c}q = v(t) \tag{3.30}$$

Pero una ecuación diferencial de orden n se puede convertir a n ecuaciones diferenciales simultáneas de primer orden con cada ecuación de la forma

$$\frac{dx_1}{dt} = a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n + b_i f(t) \tag{3.31}$$

donde cada x_i es una variante de estado, y las a_{i1} y b_i son constantes para los sistemas lineales e invariantes con el tiempo. Decimos que el segundo miembro de la ecuación (3.31) es una combinación lineal de las variantes de estado entrada, $f(t)$. Es posible convertir la ecuación (3.30) en dos ecuaciones dlferenciales Simultáneas de primer orden en términos de $i(t)$ y $q(t)$. La primera ecuación puede ser $dq/dt = i$. se puede formar la segunda ecuación al sustituir $\int idt = q$ en la ecuación (3.29) y despejar di/dt. Si se resumen las dos ecuaciones resultantes, obtenemos

$$\frac{dq}{dt} = i \tag{3.32}$$

$$\frac{di}{dt} = -\frac{1}{LC}q - \frac{R}{L}i + \frac{1}{L}v(t) \tag{3.33}$$

3. Estas ecuaciones son las ecuaciones de estado y de ellas se pueden resolver simultáneamente las variantes de estado, $q(t)$ e $i(t)$, usando transformada Laplace y otros metodos, si conocemos las condiciones iniciales de $q(t)$ e $i(t)$ y si conocemos $v(t)$, la entrada

4. De estas dos variables de estado, podemos despejar todas las otras variables del circuito. Por ejemplo, se puede escribir el voltaje entre terminales del inductor en términos de las variables de estado despejadas y la entrada como

$$v_l(t) = -\frac{1}{c}q(t) - Ri(t) + v(t) \tag{3.34}$$

   La ecuación (3.34) es una ecuación de salida; decimos que $v_L(t)$ es una combinación lineal de las variables de estado, $q(t)$ e $i(t)$, y la entrada $v(t)$.

5. Las ecuaciones de estado combinadas 3.32 y 3.33 ademas la ecuación de salida (3.34) forman una representación viable del circuito que llamamos representación en el espacio de estados.

3.5. Resumen

En este capitulo se abordaron temas importantes, como el metodo de espacios de estado parcial donde solo llegamos a la representación de ecuaciones diferenciales y a partir de ellas se pueden introducir a un software de simulación y asi ver su comportamiento. ademas se introdujeron varios terminos importantes y se desarrollaron a grandes rasgos otros.

Capítulo 4

SIMULACIONES EN SIMNON

Bien ahora sin perder la idea principal y el objetivo de este trabajo que es la simulación de sistemas continuos en Simnon. En este capítulo procedemos a realizar dos casos de estudio aplicando el método parcial de espacios de estado y posteriormente introducir esas ecuaciones diferenciales en el software Simnon para analizar su comportamiento.

4.1. Caso de estudio 1 - Motor de CD

Para nuestro primer caso de estudio se escogió el motor de corriente directa (se pudo haber implementado cualquier otro sistema pero por la aplicación al área de ingeniería eléctrica se escogió este).

El motor de corriente continua, denominado también motor de corriente directa, motor CC o motor DC (por las iniciales en inglés direct current), es una máquina eléctrica que convierte energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

Los elementos más importantes de un motor DC se representados en la figura 4.1

La armadura del motor DC se modela como si tuviera una resistencia constante R en serie con una inductancia constante L que representa la inductancia de la bobina de la armadura, y una fuente de alimentación v que representa la tensión generada en la armadura.

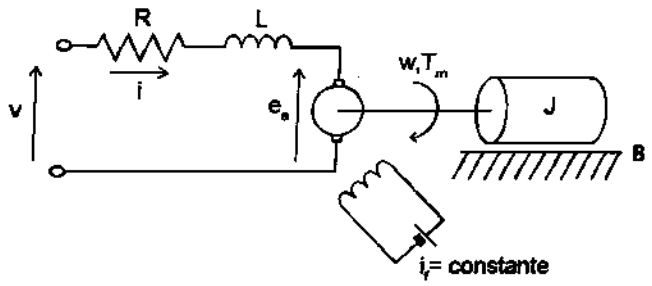

Figura 4.1: Representación de un motor de DC

La primera ecuación se realiza haciendo un análisis de la malla del circuito:

$$v(t) = R_i + L\frac{di(t)}{dt} + E_a(t)$$

o

$$L\frac{di(t)}{dt} = v(t) - R_i - E_a(t) \tag{4.1}$$

Donde $E_a(t)$ es una tensión generada que resulta cuando los conductores de la armadura se mueven a través del flujo de campo establecido por la corriente del campo i_f.

Naturalmente, en toda potencia mecánica desarrollada en el rotor se entrega a la carga mecánica conectado al eje del motor de DC. Parte de la potencia desarrollada se pierde a través de la resistencia de la bobina del rotor y la fricción y por histéresis y pérdidas por corrientes de Focault en el hierro del rotor. Desde aquí las pérdidas por fricción y parte de la energía desarrollada es almacenada como energía cinética en la masa girante del rotor. La ecuación de la sección mecánica viene dada por el modelo

$$T_m(t) = J\frac{d\omega(t)}{dt} + B\omega(t)$$

o

$$T_m(t) = J\frac{d\omega(t)}{dt} = T_m(t) - B\omega(t) \tag{4.2}$$

Donde $T_m(t)$ es el torque del motor de corriente continua, B es el coeficiente de fricción equivalente al motor de CD (corriente continua) y la carga montados sobre el eje del motor,J es el momento de inercia total del rotor y de la carga con relación al eje del motor,$\omega(t)$ es la velocidad angular del motor y $\dfrac{d\omega(t)}{dt}$ es la aceleración angular.

Para poder lograr la interacción entre las ecuaciones anteriores se proponen las siguientes relaciones que asumen que existe una relación proporcional,K_a, entre el voltaje inducido en la armadura y la velocidad angular del eje del motor.

$$E_a(t) = K_a(t) \tag{4.3}$$

Y se supone la siguiente relación electromecánica que establece que el torque mecánico es proporcional,K_m, a la corriente eléctrica que circula por el motor de DC.

$$T_m(t) = K_m i(t) \tag{4.4}$$

Sustituyendo la ecuación (4.3) en 4.1 obtenemos

$$L\frac{di(t)}{dt} = v(t) - Ri(t) - K_a\omega$$

y en su representación de espacios de estado esta dada por la ecuación (4.5)

$$\dot{i} = \frac{1}{L}(V) - \frac{R}{L}i - \frac{K_a}{L}\omega \tag{4.5}$$

tambien sustituimos la ecuación (4.2) en la ecuación (4.4) y obtenemos

$$J\frac{d\omega(t)}{dt} = K_m i(t) - B\omega(t)$$

y su representación en espacios de estado esta dada por la ecuacion (4.6)

$$\dot{\omega} = \frac{K_m}{J}i - \frac{B}{J}\omega \tag{4.6}$$

4.1.1. Simulación del motor de CD

Para simplificar un poco más el modelo sustituiremos las constantes K_m y K_a por n y las variables $\omega = x$, $\dot{\omega} = \dot{x}$ y $i = y$, $\dot{i} = \dot{y}$. como se muestra en la lista de comandos. Para empezar a utilizar simnon ejecutamos la opción de nuevo proyecto como se muestra en la imagen 4.2 y escogemos la opción de CONTINIUS SYSTEM.

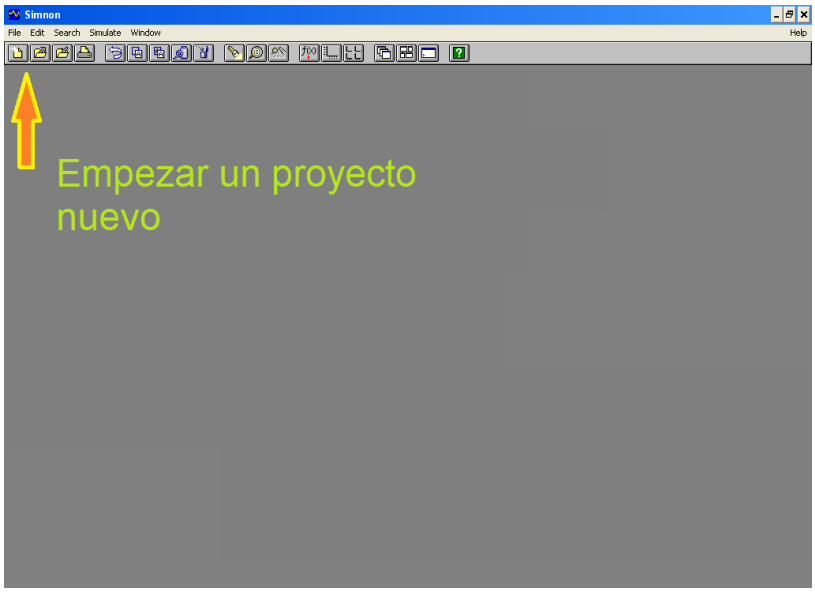

Figura 4.2: Proyecto nuevo

Ahora introducimos en el editor de texto de Simnon el siguiente codigo:

```
CONTINUOUS SYSTEM MOTORCD
" Version:      1.0
" Abstract:
" Description:
" Revision:     1.0
" Author:       Gerardo Angel Palomino Cisneros  Morelia Mich. 1021156
" Created:      06/11/2019

" Inputs and outputs:


" States, derivates and time:
STATE x y
DER dx dy
TIME t

" Initializations:
x:1
y:1

" Equations:
dx=-(B/J)*x+(n/J)*y
dy=-(R/L)*y-(n/L)*x+(1/L)*v
" Parameter values:
B=1.5
J=3.2
L=0.001
n=0.1
R=0.1
v=100
END
```

ahora pasamos a la terminal de comandos la cual se muestra en la imagen 4.3.

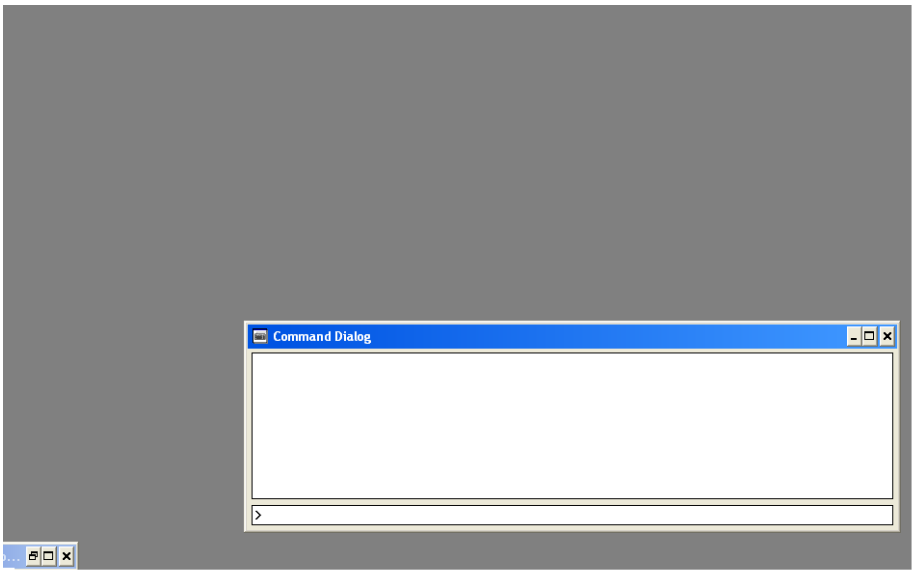

Figura 4.3: Terminal de comandos en el entorno SIMNON

Para inicializar el sistema escribimos en la terminal el nombre del comando seguido el nombre asignado al proyecto en este caso "MOTORCD"

```
SYST MOTORCD
```

si no envia algun error el programa ha compilado con éxito. Ahora para ver los gráficos del sistema, primero se ejecuta el comando

```
AXES H 0 20 V 0 100
```

para dibujar los ejes en la pantalla o simplemente se presiona el botón graficador y envía ejes por defecto como se muestra en la figura 4.4.

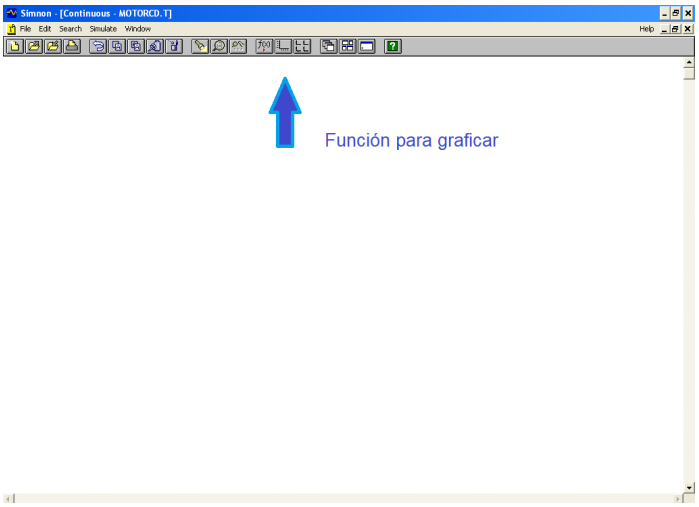

Figura 4.4: Gráficador

ahora asignamos los ejes con el comando

```
plot x y
```

y por último ejecutamos el comando

```
SIMU 0 20
```

con esto hace la gráfica en el tiempo de 0 a 20 como se muestra en la figura 4.5. Nota: en caso de que muestre el error de undefined variable se inicializa de nuevo la variable x con el comando

```
init x:1
```

y se vuelve a escribir el comando ”PLOT x yz el comando ”SIMU 0 20”.

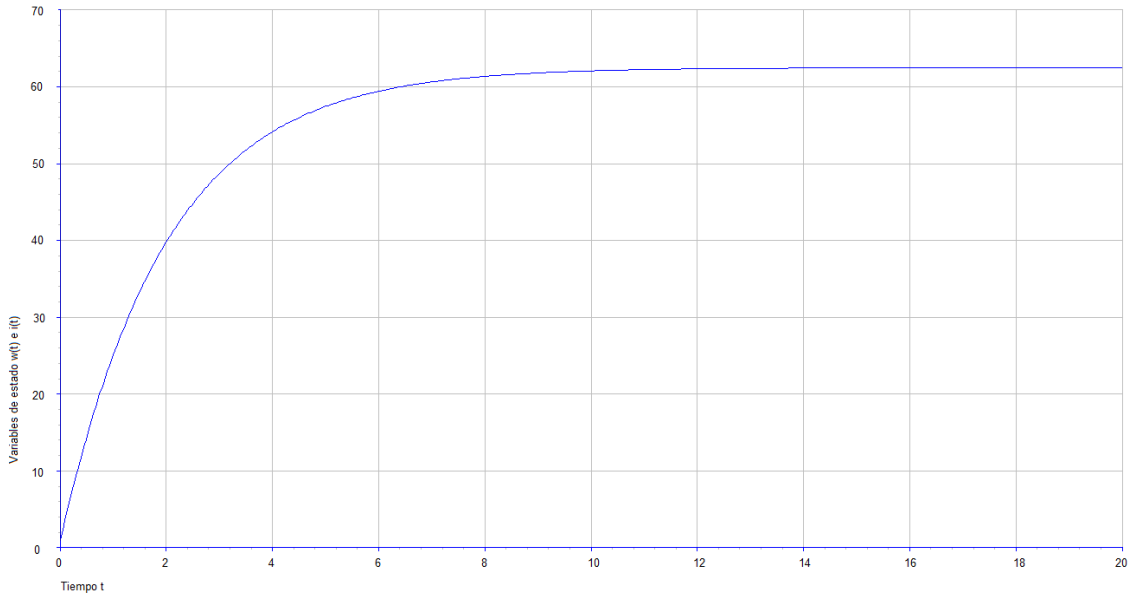

Figura 4.5: Grafica de comportamiento de un motor de CD

Nota: para cambiar las escalas de la grafica se puede hacer dando clic izquierdo y se editan los valores como se observa en la figura 4.6

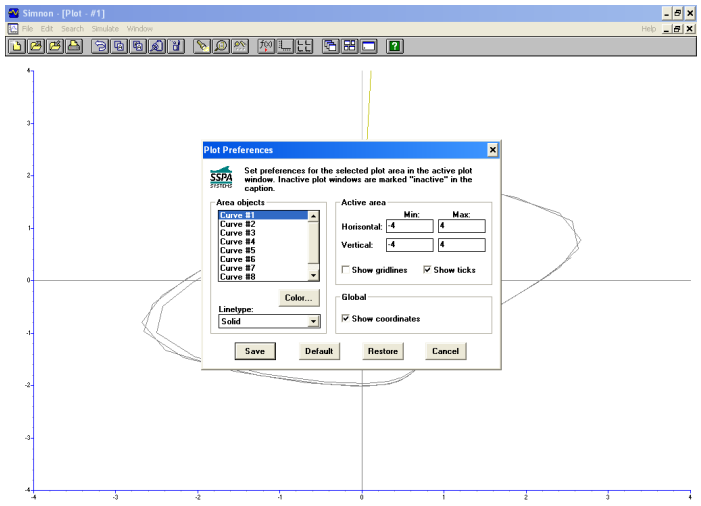

Figura 4.6: Cambiar las escalas de una gráfica

Ahora para ver el resultado en forma de diagrama de fase como se muestra en la

figura 4.7 utilizamos los siguientes comandos

```
SIMU / B1
AXES H -4 4 V -3 3
SHOW y(x) \ B1
```

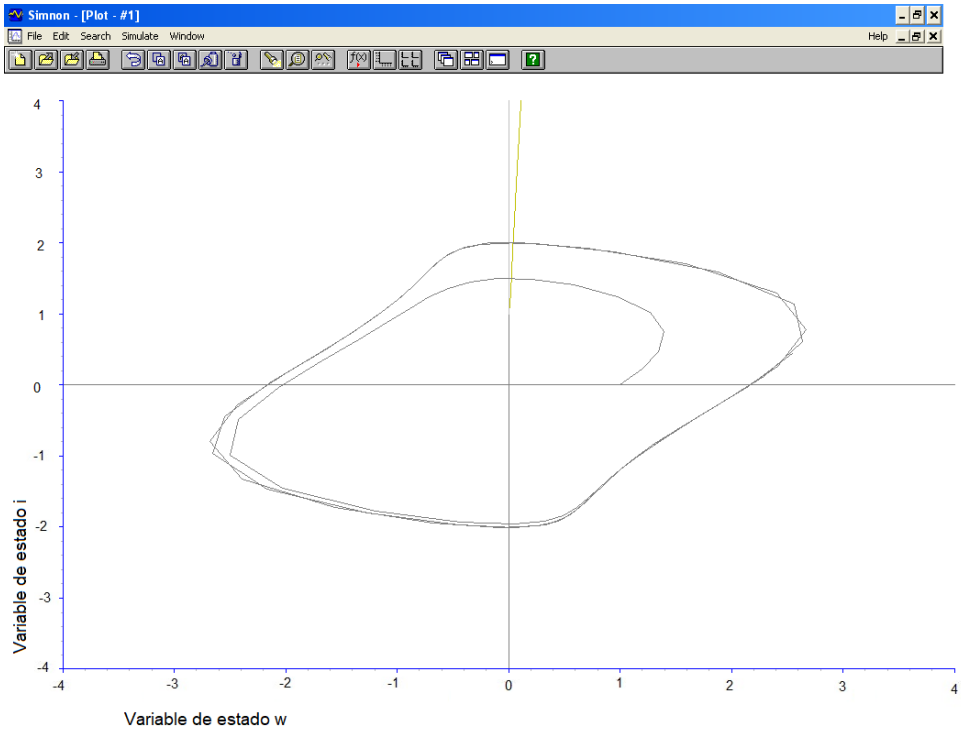

Figura 4.7: Gráfica de comportamiento de un motor de CD en diagrama de fase

En ambas simulaciones podemos observar como el sistema se vuelve estable. En la grafica de la figura 4.5 observamos como incrementa la velocidad angular $\omega(t)$ con respecto al tiempo t hasta que llega el momento donde deja de cambiar, aproximadamente cuando $t = 7s$ y en ese momento se dice que es estable. En el grafico de plano de fase se puede concluir algo similar porque se observa que ambas variables tanto $i(t)$ como $\omega(t)$ se vuelven estables.

4.2. Caso de estudio 2 - Rectificadores de onda aplicados a un motor de CD

Ahora para ver la gran potencia de programación del software, complicaremos el caso de estudio anterior haciendolo un poco más realista al incorporar los rectificadores de onda más utilizados.

4.2.1. Rectificador de media onda

El rectificador de media onda es un circuito empleado para eliminar la parte negativa o positiva de una señal de corriente alterna.

4.2.2. Motor de CD con un rectificador de media onda

En este caso se modifica el circuito del motor de CD de la figura 4.1 agregando una fuente de corriente alterna y un diodo rectificador observe los cambios en la figura 4.8.

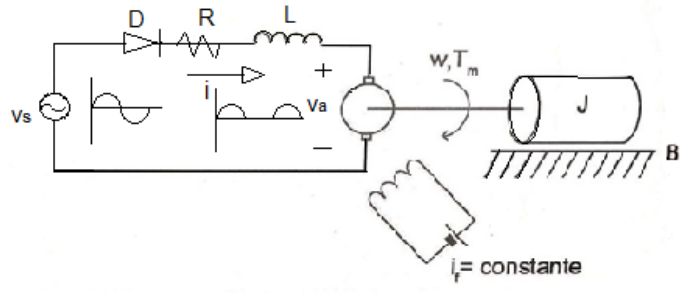

Figura 4.8: Representación de un motor de DC con un diodo rectificador

Recordando que el volteje en corriente alterna obedece la ecuación (4.7)

$$V_s = V_m sin\omega t \tag{4.7}$$

donde $\omega = 2\pi F$ y V_m es un valor maximo o valor pico.

Retomando las ecuaciones (4.5) y (4.6) que rigen el comportamiento del motor y la ecuación (4.7) del comportamiento de la corriente alterna, lo que falta es la condición que rige al comportamiento del diodo. Esta condición es fácil de representar puesto que un diodo solo trabaja cuando se encuentra polarizado de forma directa y se comporta como un corto circuito, en caso contrario se comporta como un circuito abierto, lo cual genera una onda recortada del semiciclo negativo observe la figura 4.9.

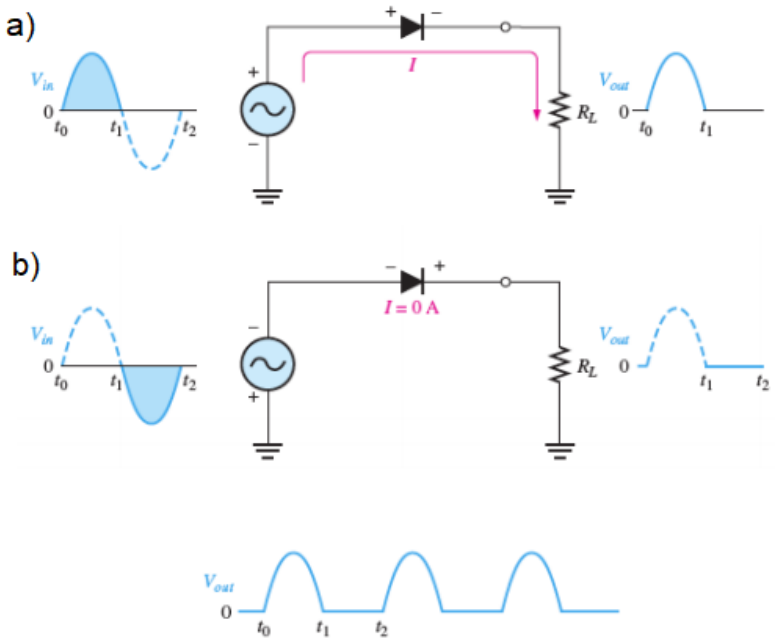

Figura 4.9: Polarizacion de un diodo; a) Semiciclo positivo b)Semiciclo negativo (Derechos de autor de la imagen en 4.1b)

Ya que se tiene todo lo necesario, procedemos a hacer la simulación en simnon, para ello modificamos el código que aparece en el caso de estudio 1, agregamos la condición del diodo la cual modifica el voltaje de entrada v y la fórmula del comportamiento de la corriente alterna tal como se muestra en el código:

```
CONTINUOUS SYSTEM MCDDIO2
" Version:      1.0
" Abstract:
" Description:
" Revision:     1.0
" Author:       Gerardo Angel Palomino Cisneros  Morelia Mich. 1021156
" Created:      06/11/2019

" Inputs and outputs:
```

```
" States, derivates and time:
STATE x y
DER dx dy
TIME t

" Initializations:
x:1
y:1

" Equations:
dx=-(B/J)*x+(n/J)*y
dy=-(R/L)*y-(n/L)*x+(1/L)*v


" Parameter values:
vm=100
pi=3.141516
F=60
B=1.5
J=3.2
L=0.001
n=0.1
R=0.1
w=2*pi*F
vs=vm*sin(w*t)
"condiciones
v=if vs>0 then vs else 0
END
```

procedemos con la serie de comandos para obtener la grafica de la simulación

```
AXES H 0 20 V 0 100
PLOT x y
INIT x:1
SIMU 0 20
```

y obtenemos la gráfica de la figura 4.10.

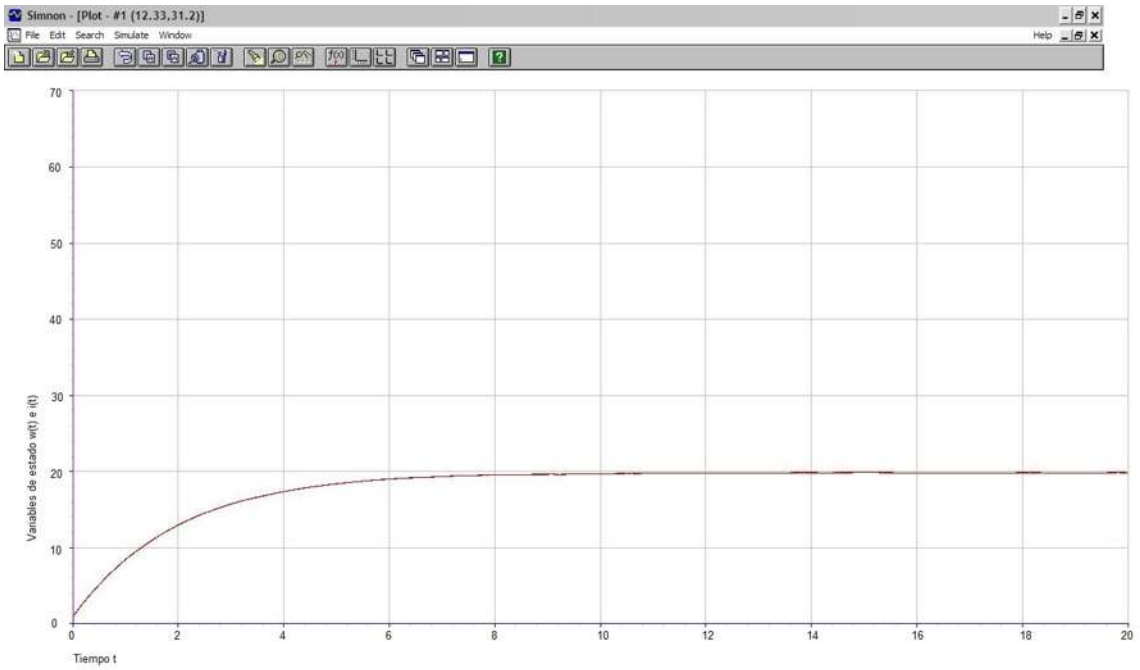

Figura 4.10: Gráfica del comportamiento del motor de CD con el rectificador de media onda

Para la gráfica de fase de plano introducimos los comandos

```
SIMU / B1
AXES H -4 4 V -4 4
SHOW y(x) / B1
```

y obtenemos la gráfica de la figura 4.11.

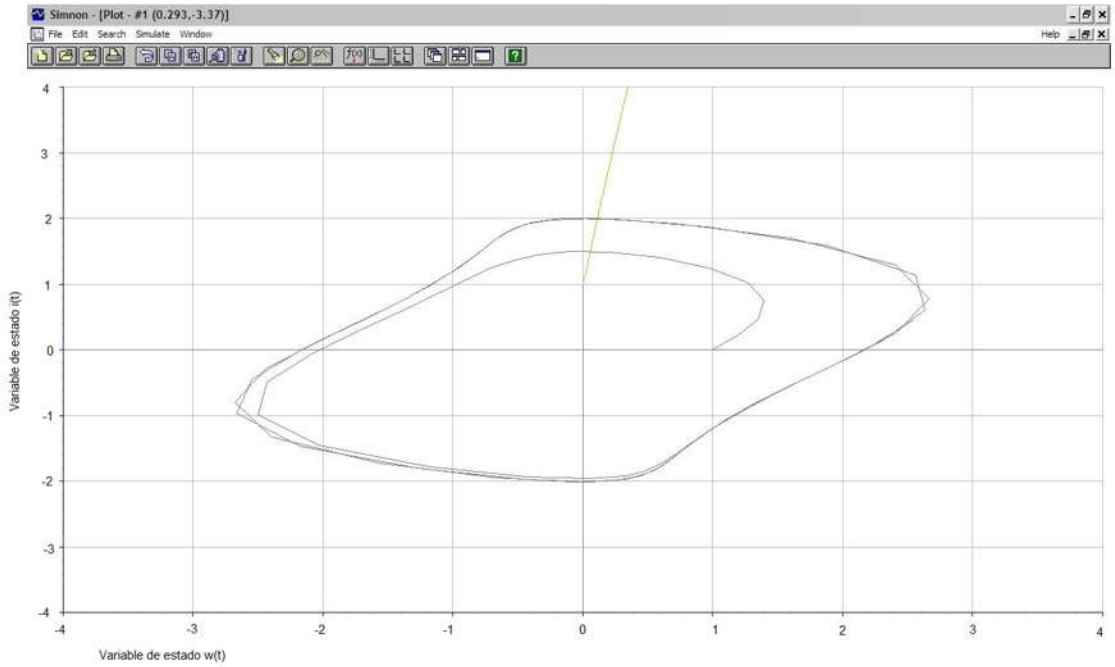

Figura 4.11: Gráfica de planos de fase del comportamiento del motor de CD con el rectificador de media onda

Existen más variables en las cueles también se puede ver su comportamiento. Por ejemplo para ver el voltaje de la fuente V_s para ello basta con escribir los siguientes comandos en la terminal de comandos de simnon.

```
SPLIT 1 1
AXES H 0 .1 V -100 100
PLOT vs
SIMU 0 1
```

y da como resultado la gráfica de la figura 4.12. **Nota:** Es importante reasignar nuevas escalas en esta simulación ya que el voltaje en CA se repite 60 veces en un lapso de 1 segundo y su voltaje de pico es $Vm = 100$ y sino se hace esto no se logra apreciar bien.

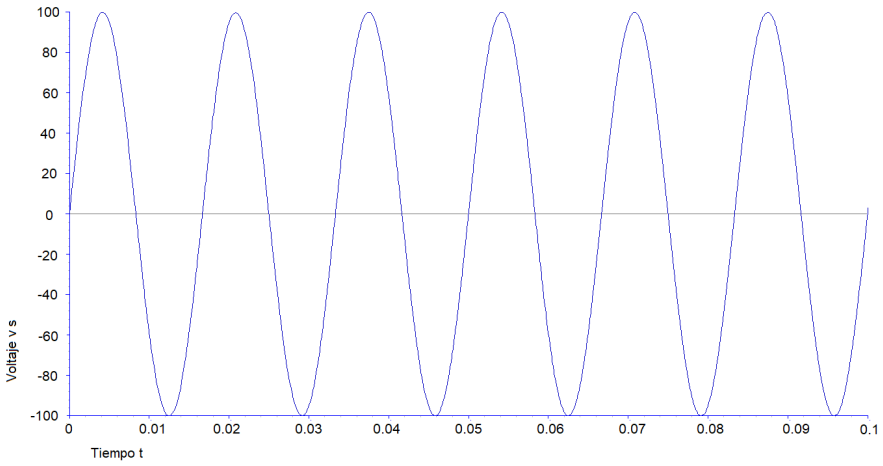

Figura 4.12: Gráfica del voltaje de la fuente del circuito de la figura 4.1

ahora para ver la simulación del voltaje despues de que pasa por el rectificador V que es el que alimenta al motor, aplicamos practicamente la misma lista de comandos hechos anteriormente. Solo modificamos la orden "PLOT vs"por "PLOT vz muestra la grafica de la figura 4.13.

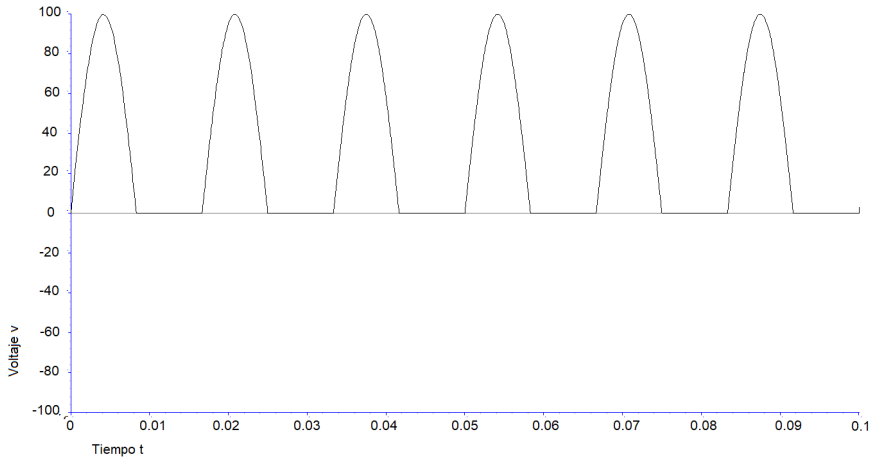

Figura 4.13: Gráfica del voltaje del rectificador del circuito de la figura 4.1

en la simulación se observa que tiene un comportamiento similar a lo que idealmente es la rectificación mostrada en la figura 4.9, por lo tanto concluimos que es una buena

simulación.

4.2.3. Rectificador de onda completa

El puente de diodos o también llamado puente rectificador o puente de Graetz es un circuito rectificador de onda completa, el puente de diodos requiere de cuatro diodos rectificadores o diodos de potencia conectados en serie en forma de puente, la principal ventaja de este circuito de puente rectificador permite la rectificación de onda completa de un transformador que no tenga una toma central (tap central) lo que reduce su tamaño y costo.[17]

Aunque se puede utilizar cuatro diodos rectificadores o de potencia para hacer un puente de diodos es posible encontrar componentes electrónicos que ya disponen internamente de 4 diodos y listo para utilizar, estos "puentes de diodos" tienen rango de diferentes voltajes, corriente, tamaños y demás datos que podremos verificar en la hoja de datos de cada componente. Como se muestra en la imagen se dispone de 4 diodos (D1, D2, D3 y D4) los cuales están conectados en serie con un respectivo par opuesto, de los cuales únicamente dos diodos conducen la corriente durante cada medio ciclo.[17] Como se muestra en la figura 4.14. se dispone de 4 diodos (D1, D2, D3 y D4) los cuales están conectados en serie con un respectivo par opuesto, de los cuales únicamente dos diodos conducen la corriente durante cada medio ciclo.

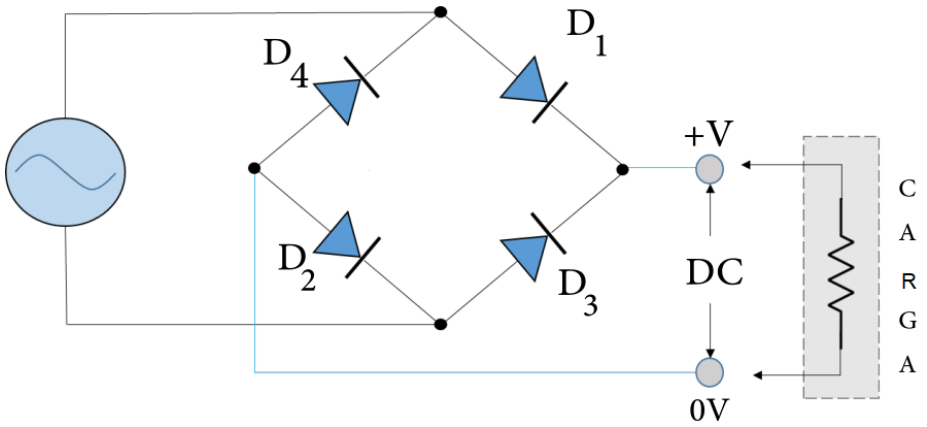

Figura 4.14: Representación de un puente rectificador de diodos

Semiciclo positivo del puente rectificador de diodos

Durante el semiciclo positivo los diodos D1 y D2 son los que conducen en serie, mientras que los diodos D3 y D4 tienen polarización inversa por lo tanto se comportarían como un circuito abierto. Como se observa en la figura 4.15.

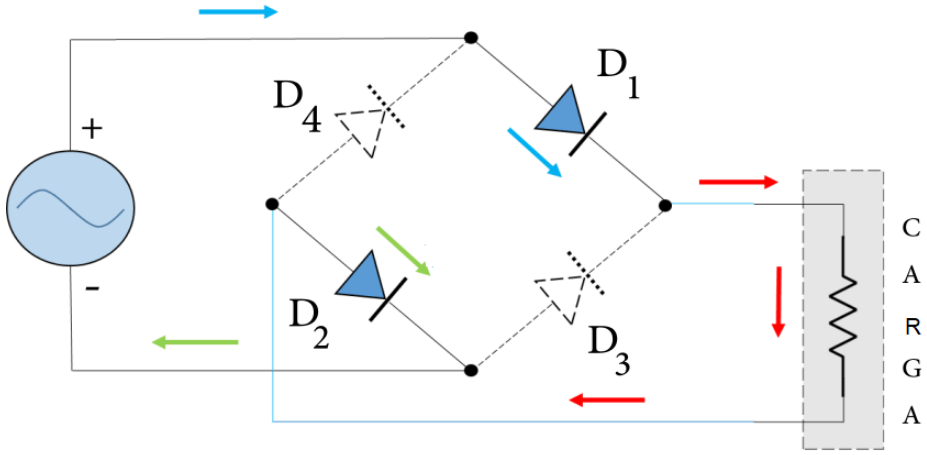

Figura 4.15: Semiciclo positivo en un puente rectificador de diodos

Semiciclo negativo del puente rectificador de diodos

Durante el semiciclo negativo los diodos D3 y D4 son los que conducen en serie, mientras que los diodos D1 y D2 tienen polarización inversa por lo tanto se comportarían como un circuito abierto. Como se observa en la figura 4.16.

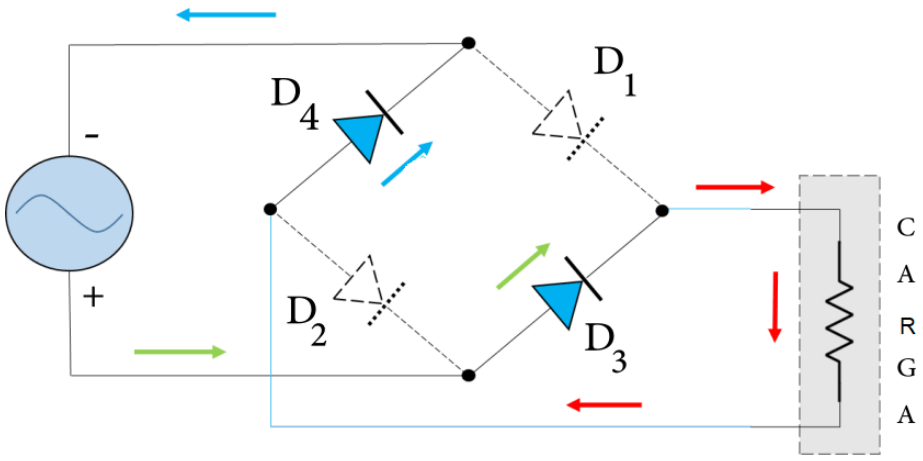

Figura 4.16: Semiciclo negativo en un puente rectificador de diodos

NOTA: Se puede observar en la imagen del semiciclo positivo y semiciclo negativo que la dirección de la corriente en la carga es la misma. y produce un resultado parecido al mostrado en la figura 4.17b.

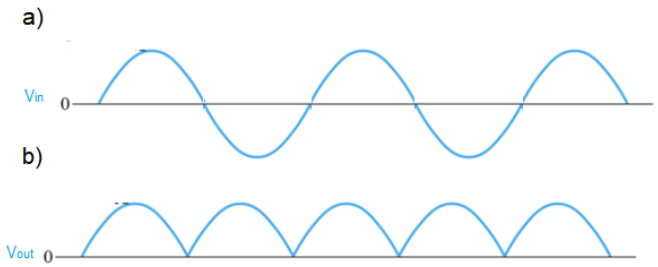

Figura 4.17: a) Onda sinusoidal antes ser rectificada. b) Onda sinusoidal rectificada por un puente rectificador de diodos

Como la corriente que fluye a través de la carga es unidireccional, el voltaje desarrollado a través de la carga también es unidireccional, se debe considerar que los diodos tendrán una caída de voltaje de aproximadamente 0.7 V por diodo, esto quiere decir que en cada semiciclo se tiene una caída de 1.4V (2 diodos * 0.7 V = 1.4V).

4.2.4. Motor de CD con un puente rectificador de onda completa

Para este caso se modifica el circuito del motor de CD de la figura 4.1 agregando una fuente de corriente alterna y el puente rectificador el cual se considera de forma ideal despreciando las perdidas de voltaje que generan los diodos. Observe los cambios en la figura 4.18.

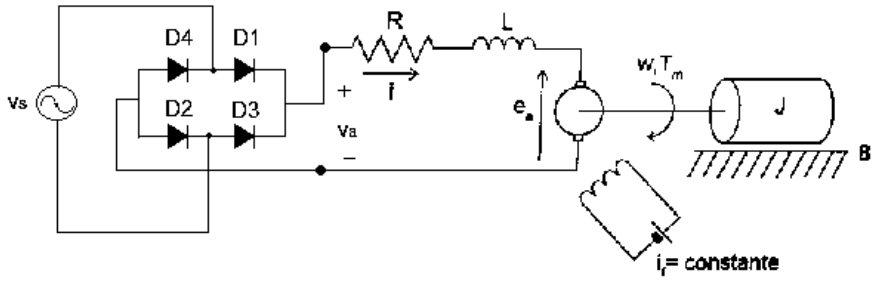

Figura 4.18: Representación de un motor de CD con un puente rectificador de diodos

Ahora que sabemos la forma como se comporta el puente rectificador de diodos, procedemos a la simulación y para ello simplemente se modifica la condición del codigo del caso del rectificador de media onda tal como se observa en el siguiente codigo:

```
CONTINUOUS SYSTEM MCDDIO2
" Version:      1.0
" Abstract:
" Description:
" Revision:     1.0
" Author:       Gerardo Angel Palomino Cisneros  Morelia Mich. 1021156
" Created:      06/11/2019

" Inputs and outputs:


" States, derivates and time:
STATE x y
DER dx dy
TIME t

" Initializations:
x:1
y:1

" Equations:
dx=-(B/J)*x+(n/J)*y
dy=-(R/L)*y-(n/L)*x+(1/L)*v
```

```
" Parameter values:
vm=100
pi=3.141516
F=60
B=1.5
J=3.2
L=0.001
n=0.1
R=0.1
w=2*pi*F
vs=vm*sin(w*t)
"condiciones
v=if vs>0 then vs else -vs
END
```

se procede con la serie de comandos para obtener la gráfica de la simulación

```
AXES H 0 20 V 0 100
PLOT x y
INIT x:1
SIMU 0 20
```

y se obtiene la gráfica de la figura 4.19.

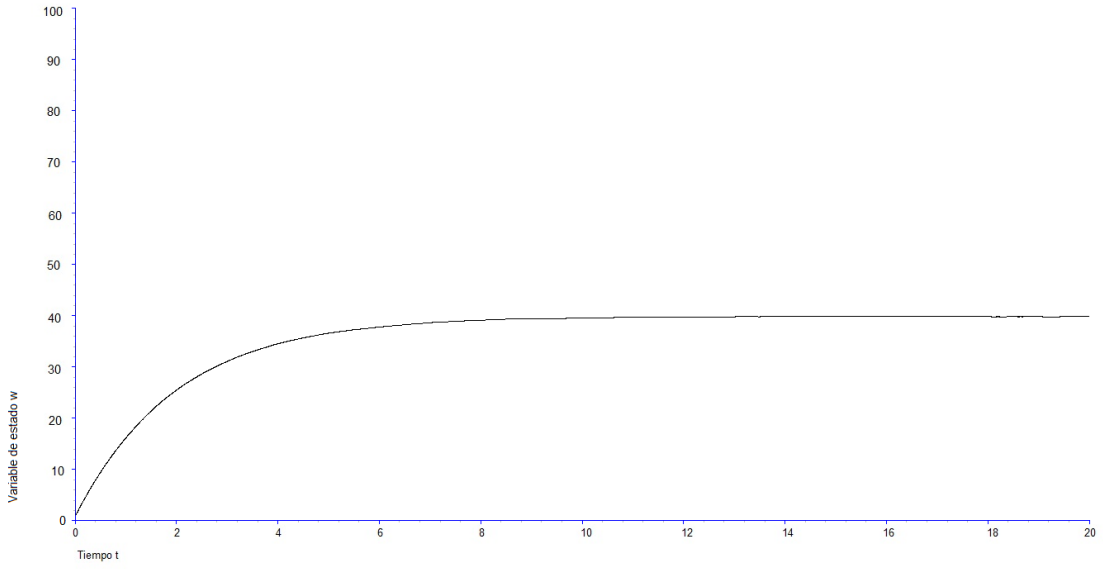

Figura 4.19: Gráfica del comportamiento del motor de CD con el rectificador de onda completa

para la gráfica de fase de plano se introducen los comandos

```
SIMU / B1
AXES H -4 4 V -4 4
SHOW y(x) / B1
```

y obtenemos la gráfica de la figura 4.20.

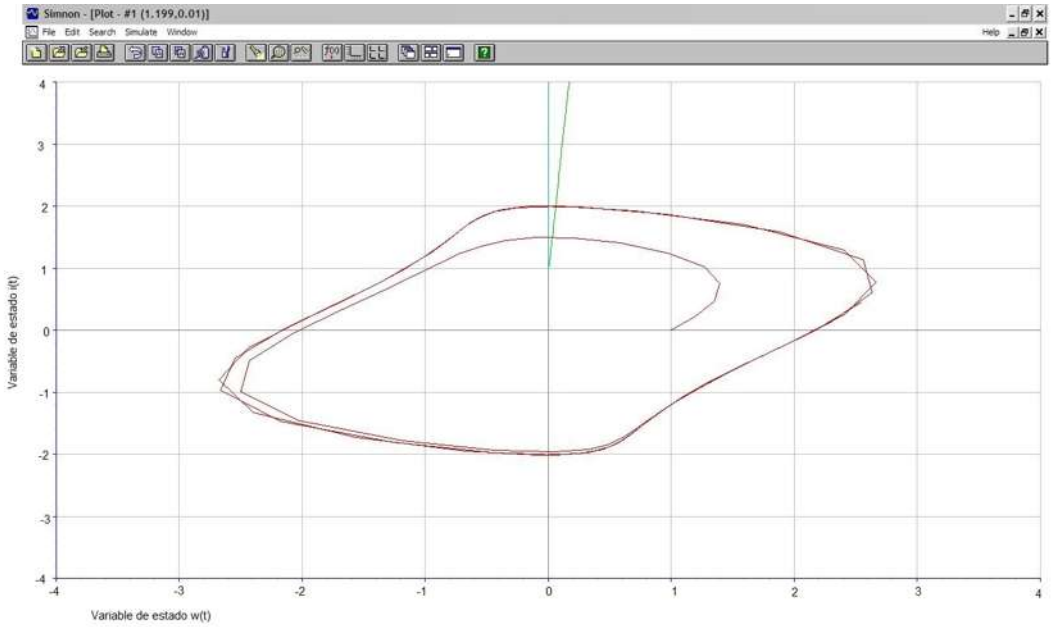

Figura 4.20: Gráfica de planos de fase del comportamiento del motor de CD con el rectificador de onda completa

Observemos que en la grafica 4.19 la velocidad $\omega(t)$ aumenta al doble que en el caso anterior, pero de igual forma llega a un punto donde se vuelve estable en (t=10). pero ¿porque pasa esto? para ver que esta pasando con el voltaje en la fuente y el voltaje a la salida del puente se hacen las simulaciones correspondientes esto con la finalidad de entender el comportamiento del motor. Primeramente para visualizar la gráfica de vs o el voltaje de la fuente se introduce en la terminal de comandos de simnon lo siguiente

```
SPLIT 1 1
AXES H 0 .1 V -100 100
PLOT vs
SIMU 0 1
```

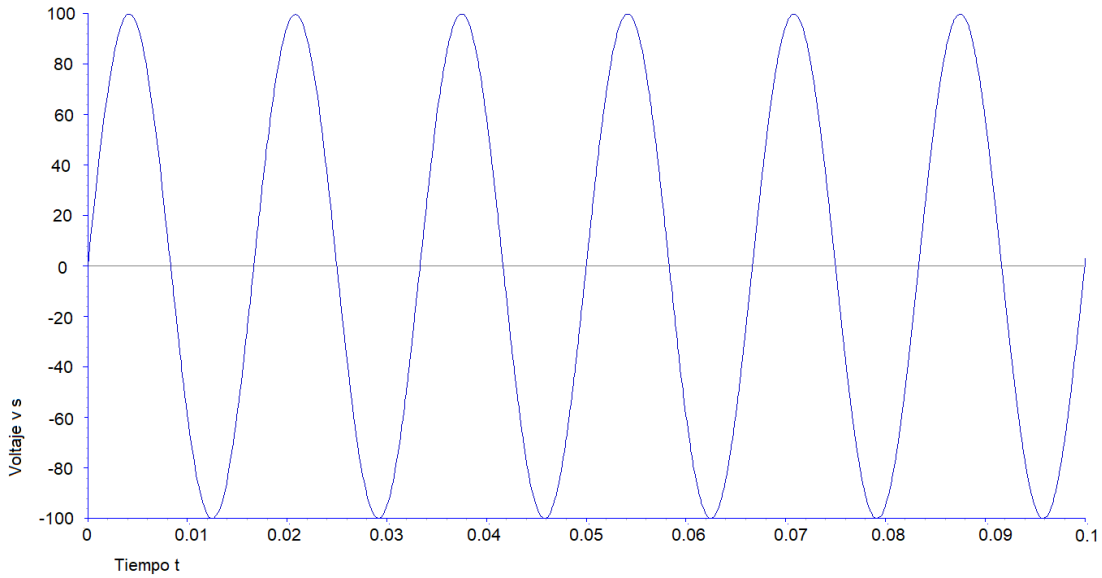

Figura 4.21: Gráfica del voltaje de la fuente del circuito de la figura 4.1

ahora para visualizar el voltaje a la salida del puente rectificador escribimos en la terminal de comandos de simnon las siguientes ordenes

```
SPLIT 1 1
AXES H 0 .1 V -100 100
PLOT v
SIMU 0 1
```

y obtenemos la gráfica de la figura 4.22. Donde nos muestra que efectivamente se está rectificando el voltaje y el semiciclo negativo lo está invirtiendo por lo tanto simula lo que idealmente sucede, tal y como se muestra en la figura 4.17.

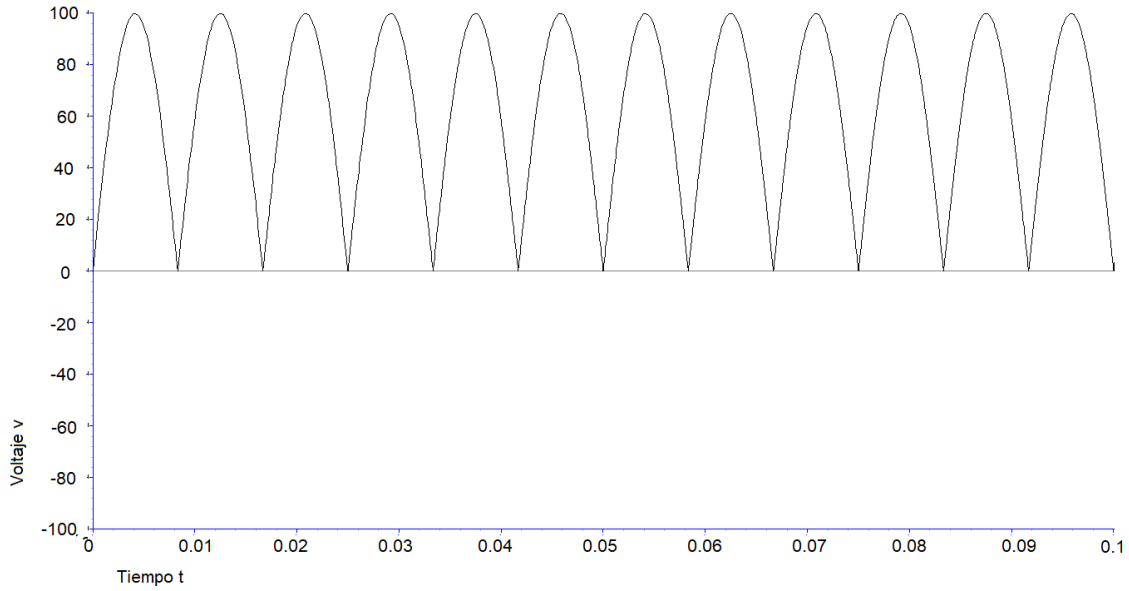

Figura 4.22: Gráfica del voltaje rectificado del circuito de la figura 4.1

4.2.5. Resumen

Para ver el comportamiento de los casos de estudio se puede hacer la comparación de las 3 gráficas como se muestra en la figura 4.23. Se hace la simulación del sistema MOTORCD, despues la simulación del sistema MCDDIO1 y por último la simulación del sistema MCDDIO2.

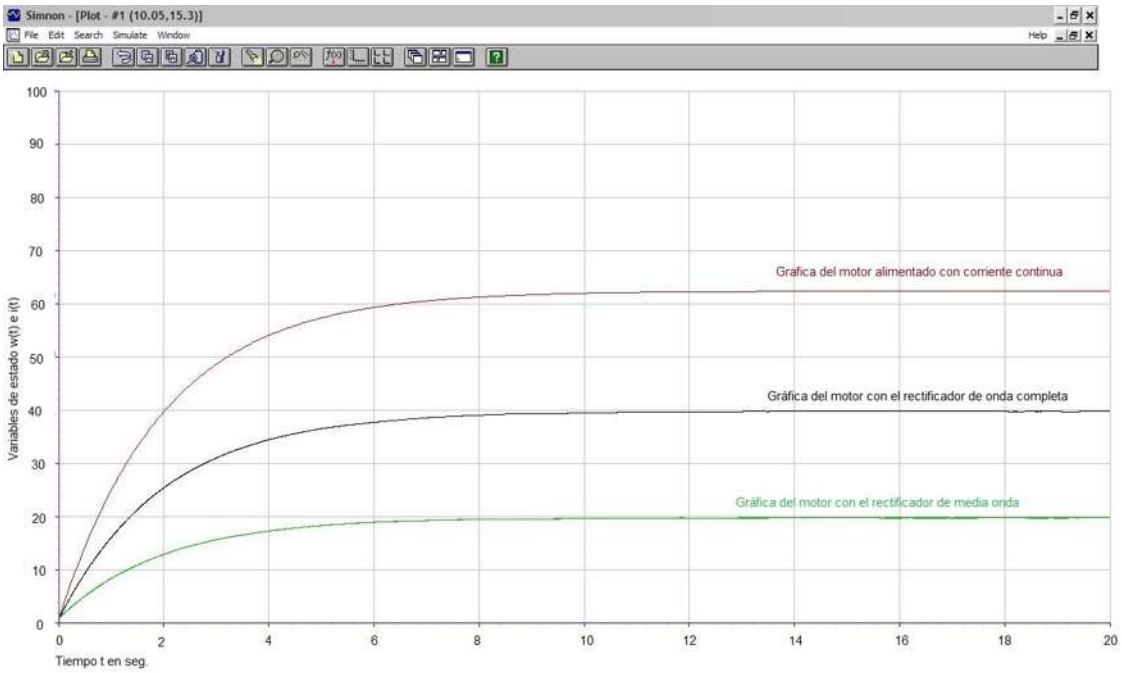

Figura 4.23: Comparación de las 3 gráficas

En esta gráfica comprobamos que es muy acertado el comportamiento de cada simulación por ejemplo en la del rectificador de media onda nos muestra una gráfica en un rango de 20 y es lógico pensar en ello puesto que solo se está ocupando la mitad del voltaje, también en la gráfica del rectificador de onda completa nos muestra un valor cercano a 40 lo cual nos dice que ya está aprovechando la parte negativa y positiva del voltaje en corriente alterna.

Otra observación importante es que se puede cambiar el margen de error de la simulación para suavizar las gráficas. Por ejemplo al simular el voltaje de la fuente tenia por defecto un error del 1e-3 lo cual nos genero una grafica parecida a la figura 4.24

pero se ve un tanto distorsionada, por fortuna simplemente se cambia el margen dando clic derecho sobre la función de simulación y se abre una ventada donde se pueden cambiar los parámetros observe la figura 4.25. Para asignar un error más preciso en este caso 1e-10 y nos muestra una gráfica mucho más suave como se observa en la figura 4.26.

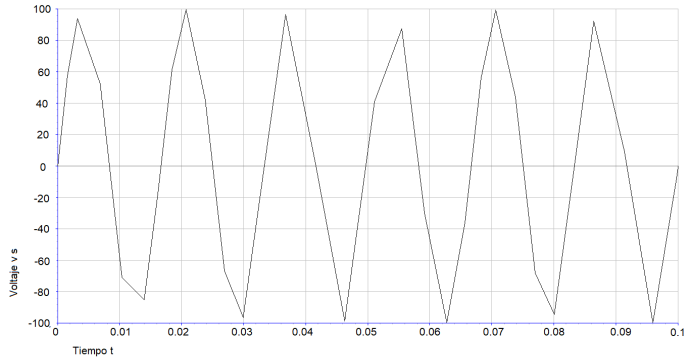

Figura 4.24: Gráfica sinusoidal con un error de .001

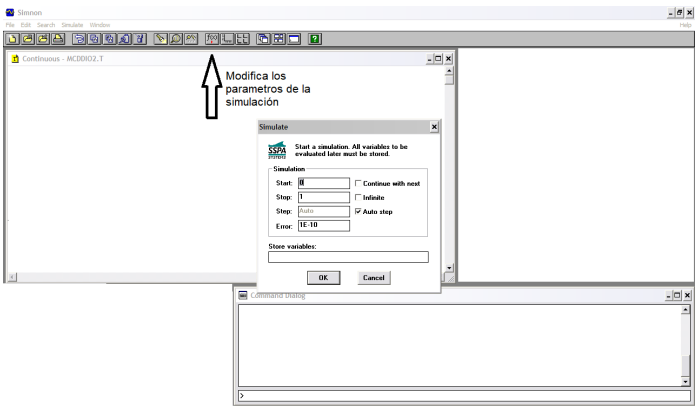

Figura 4.25: Modificar el error en una simulación

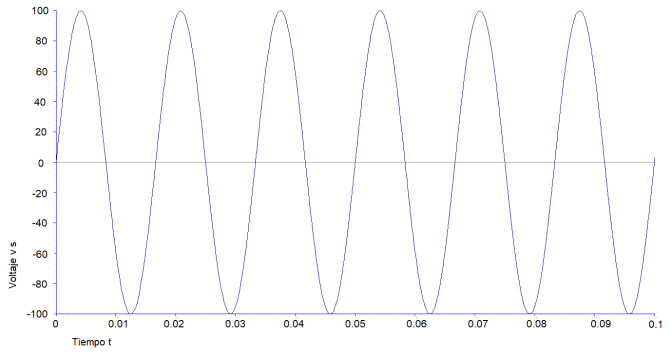

Figura 4.26: Gráfica sinusoidal con un error de 1e-10

Capítulo 5

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

- Una de las principales ventajas que tiene el software es su entorno de trabajo el cual es muy parecido al de MATLAB y uno como usuario se puede acoplar muy fácilmente a su entorno.
- Una de las partes que más llama la atención es que en simnon se puede programar condiciones como se mostró en el segundo caso de estudio con los puentes rectificadores.
- Otro aspecto importante es la forma de entregar resultados y la agilidad en como los procesa.
- Realmente la cantidad de comandos que se utilizan para la potencia de cálculo que se obtiene es mínima.
- En cuestión de las simulaciones es muy fácil cambiar los parámetros para obtener los datos de la forma más optima posible.

- Una de las mayores ventajas que se pudo observar al compilar en SIMNON es; cuando se comete algún error en el código, automáticamente el cursor del mouse se posiciona en el error e indica que existe determinado error en esa parte.

- Quizá una de las partes que no es tan convincente es al generar las graficas ya que no se puede cambiar el tamaño de la fuente para visualizar mejor la gráfica.

- Otra desventaja es que varia al generar las etiquetas en las gráficas dependiendo del tipo de ordenador y sistema operativo en donde se ejecute.

- Otro limitante que tiene es que no en todos los sistemas operativos se puede ejecutar y de la vez una ventaja porque puede llegar a ser el caso que en alguna industria tengan equipos que sean viejos y si se requiere la simulación con algún simulador de alto nivel no podría ser ejecutado y en cambio SIMNON requiere muy poca capacidad de computo para ser ejecutado.

5.2. Trabajos futuros

Como se menciono a lo largo del trabajo el software SIMNON es capaz de mucho más, por ejemplo la simulación de sistemas discretos y la conexión entre sistemas, además de que se pueden definir MACROS.

5.3. Recomendaciones

Se recomienda este trabajo a los alumnos de grados intermedios y avanzados a nivel licenciatura. Puesto que es una gran recopilación de información de sistemas continuos, simulaciones, una introducción a los espacios de estado, también un pequeño tutorial para hacer simulaciones en el software Simnon. Además de incluir las ecuaciones que simulan el motor de corriente continúa y aplicando ciertas variantes, también explica de forma sencilla el funcionamiento de un puente rectificador de diodos y en su defecto el comportamiento de un diodo.

Apéndice A

Lista de comandos del editor de texto SIMON y terminal SIMNON

Editor de texto Simnon

1. Comandos de entrada

   READ - Lee variables del teclado

   WRITE - Escribe variables

   SWITCH - Comando de utilidad

   STOP - Dentiene la ejecucion y regresa OS

2. Asignación

   FREE - Libera variables globales asignadas

   LET - Asigna variables globales

3. Control del flujo del programa

   LABEL L - declaración de etiqueta

   GOTO L - Control de transferencia

   IF..GOTO - Control de transferencia

   FOR..TO - ciclo

NEXT V -

4. Macro

   DEFAULT - asigna valores por defecto

   MACRO - define una macro.

   FORMAL - declara argumentos formales

   END - termina la definicion de macro.

   SUSPEND - suspende la ejecucion de la macro.

   RESUME - Reactiva la ejecución de la macro.

COMANDOS SIMNON

1. entrada y salida

   EDIT - Editar descripción del sistema

   DISP - Muestra los parametros

   GET - Obtener parámetros y valores iniciales

   LIST - Lista de archivos

   PRINT - Imprimir archivos

   SAVE - Guardar valores de parámetros y valores iniciales en un archivo

   STOP - detener

2. Graficas de salida

   AREA - Seleccionar ventana en pantalla

   ASHOW - Trazar variables almacenadas con escalado automático

   AXES - Dibujar ejes

   HCOPY - Copia impresa de la pantalla.

   MARK - Escribir texto en ejes de gráficos

   SHOW - Trazar variables almacenadas

   SPLIT - División de pantalla en ventanas

TEXT - Transferir cadena de texto al gráfico

3. Simulación.

   ALGOR - Seleccionar algoritmo de integración

   ERROR - Elija error vinculado para la rutina de integración

   INIT - Cambiar valores iniciales de variables de estado

   PAR - Cambiar parámetros

   PLOT - Elija las variables que se trazarán

   SIMU - Simular

   STATE - Comando especial para la rutina de integración DAS

   STORE - Elija variables para almacenar

   SYST - Activar sistemas

4. Auxiliares

   HELP - Da orientación

   NEWS - Da noticias sobre Simnon

   TURN - Establecer interruptores

Apéndice B

Descripciones del sistema SIMNON

Simon permite tres tipos de sistemas: SISTEMA CONTINUO, SISTEMA DISCRETO y SISTEMA DE CONEXIÓN. En este caso solo se describe lo necesario para ejecutar el sistema continuo. Las siguientes notaciones de Backus-Naur se utilizan para describir la sintaxis.

<> unidad sintáctica

= denota

| exclusivo o

[] elemento opcional

{} elemento obligatorio

$*$ repetición

Los siguientes elementos sintácticos son necesarios para describir los sistemas.

LETRAS

$< letras > = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z$

$< letras > = a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$

$< digitos > = 0|1|2|3|4|5|6|7|8|9|$

IDENTIFICADORES

$< identifier >::=< letter > | < identifier >< letter > |$
$< identifier >< digit >$

VARIABLES

<system identifier>: : =<identifier>

< simple variable >:: = <identifier>

<variable>: : = <simple variable> | <simple variable> [system identifier>]
SISTEMAS CONTINUOS <identificador del sistema>

INPUT < simple variable >*
OUTPUT < simple variable >*
STATE < simple variable >*
DER < simple variable >*
TIME < simple variable >

INITIAL
calcula los valores iniciales de las variables de estado
SHORT

Calcula las variables auxiliares
calcula las variables de salida
calcula las derivadas
asigna los parametros
asigna valores iniciales
END

Apéndice C

Terminos importantes

Simulación

Sistema

Modelado

Programa

Espacio

Estado

Sistema lineal

Sistema invariante

LTI

Ecuación

Software

Memoria

Comandos

Almacenamiento

Parametros

Variables

Constantes

Gráfica

Motor

Rectificador

Plano de fase

Estabilidad

Apéndice D

Derechos de autor de las imagenes

[1.1b] https://www.taringa.net/+ciencia_educacion/la-aguja-de-buffon-y-el-calculo-de-pi_hlrqr
[1.2b] https://historiageneral.com/2014/06/19/que-fue-el-proyecto-manhattan/
[1.3b] https://www.timetoast.com/timelines/simulacion-6c5796e4-3c4d-443c-8c8b-2a640d718a63
[1.4b] https://history-computer.com/ModernComputer/Software/Simula.html
[1.5b] https://ingenieria.bogota.unal.edu.co/es/dependencias/vicedecanatura-academica/item/331-matlab-para-todos-y-para-todo-ya-esta-disponible-la-licencia-del-software-matlab-para-toda-la-comunidad-unal.html
[1.6b] https://www.modelica.org/
[1.7b] https://www.scientec.com.mx/labview/
[4.1b] https://www.mecatronicalatam.com/es/diodo/puente-de-diodos
[4.2b] http://www.labc.usb.ve/paginas/mgimenez/EC1167/Clases/Clase2.pdf

Referencias

[1] Eduardo García Dunna Heriberto García Reyes y Leopoldo E. Cárdenas Barrón, *Simulación y análisis de sistemas con ProModel*, segunda edición, PEARSON, México, 2013.

[2] Karl Johan Aström ,Department of Automatic Control, Lund Institute of Technology (LTH). ,*A Simnon Tutorial*, 1985.

[3] Samir S. SOLIMAN y Mandyam D. SRINATH, SEÑALES y SISTEMAS continuos y discretos, *Segunda edici6n*,PRENTICE HALL IBERIA, S.R.L..Madrid, 1999

[4] Steven C. Chapra y Raymond P. Canale, Métodos numéricos para ingenieros, *Quinta edición*, McGRAW-HILL/INTERAMERICANA EDITORES, S.A. DE C.V. ,México, D. F. 2007.

[5] Katsuhiko Ogata, Ingeniería de control moderna, *Quinta edición*, PEARSON EDUCACIÓN, S.A., Madrid, 2010

[6] Norman S. Nise, SISTEMAS DE CONTROL PARA INGENIERÍA, *Tercera edición en inglés(Primera edición en español)*, Compañia Editorial continental ,México, D. F. 2006.

[7] Irlanda Redón, Simulación de sistemas fisicos en tres dimenciones utilizando 20-SIM, *Primer edicion*, Universidad Michoacana De San Nicolás de Hidalgo, Morelia, Michoacan , México, 2018.

[8] http://www.mpassociates.gr/software/catalog/sci/simnon/simnon.html

[9] https://www.software.unam.mx/producto/matlab/

[10] https://www.modelica.org/modelicalanguage

[11] https://www.scilab.org/about/company/history

[12] https://www.gnu.org/software/octave/about.html

[13] https://portal.research.lu.se/portal/en/publications/a-simnon-tutorial(d961f43e-a711-4851-a356-747329200147).html

[14] https://www.ni.com/es-mx/shop/labview.html

[15] https://controlautomaticoeducacion.com/analisis-de-sistemas/modelo-de-motor-dc/

[16] https://www.lasalle.edu.co/wcm/connect/080875de-3890-462a-a0d1-5bc76675ca26/Combita-Casas-3.pdf?MOD=AJPERES&CVID=lVN8YDi&CVID=lVN8YDi$CVID=lVN8YDi

[17] https://www.mecatronicalatam.com/es/diodo/puente-de-diodos

[18] http://www.labc.usb.ve/paginas/mgimenez/EC1167/Clases/Clase2.pdf