UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

SIMULACIÓN DE SISTEMAS FÍSICOS EN TRES DIMENSIONES UTILIZANDO 20-SIM

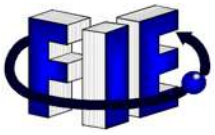
TESIS

QUE PARA OBTENER EL GRADO DE:

LICENCIADA EN INGENIERÍA EN COMPUTACIÓN

PRESENTA :

IRLANDA RENDÓN NEPOMUCENO



ASESOR DE TESIS:

DOCTOR EN INGENIERÍA ELÉCTRICA
GILBERTO GONZÁLEZ AVALOS

MORELIA, MICHOACÁN, MARZO 2018

DEDICATORIA

A mis padres, Melquiades Rendón y Manuela Nepomuceno, por creer en mí y secar mis malos ratos, por ser una motivación en mi vida, por formarme y guiarme en el camino, por inculcar en mi la importancia de estudiar, por ser ellos la motivación e inspiración para terminar este proyecto.

A mis hermanos, Ashuan y Grecia, por ser una motivación más en mi vida, a ambos por su apoyo y compresión.

En especial a Irving por todo su apoyo y motivación para mejorar mi persona, por todas las alegrías compartidas.

AGRADECIMIENTOS

A mi asesor el Dr. Gilberto González Avalos por guiarme en el desarrollo de este trabajo, por sus valiosas enseñanzas, por su apoyo y valiosos consejos a lo largo de este proceso de investigación y guía en mi formación.

A mi apreciada Universidad Michoacana de San Nicolás de Hidalgo y Facultad de Ingeniería Eléctrica.

Agradezco a todos mis profesores del área de Computación por su contribución en mi desarrollo tanto profesional como personal

Quiero agradecer al Ing. Ignacio Franco, por sus enseñanzas, por su aportación de conocimientos para que este trabajo se realizará de la mejor manera.

Agradezco a mis compañeros y amigos que me acompañaron a lo largo de la carrera.

A mis revisores de tesis, el Ing. Alfredo Rocha Villa, M.I Haydee Edith Lemus Castañeda, M.I José Rafael Rodríguez Ochoa, por brindarme su asesoría.

ÍNDICE

RESUMEN

La presente tesis abarca el tema de modelado y simulación de sistemas en tres dimensiones, dicho trabajo proporciona la capacidad de ser utilizada, como un método pedagógico eficaz, para transmitir a los estudiantes conocimientos teóricos de manera eficiente, con el fin de dar a conocer una nueva herramienta de software, como lo son los simuladores, los cuales poseen grandes capacidades, cualidades de representación y que superan diversas limitantes en el mundo real, como son los costos, la falta de equipo, tiempo, entre otros.

En el primer capitulo se describe el impacto de la simulación que genera en la actualidad, haciendo énfasis en la necesidad del uso de las herramientas de cómputo. Se estable el objetivo principal de la presente tesis, así como la justificación, la metodología y el contenido de la misma. El segundo capítulo refiere al marco histórico y la importancia que tiene el estudio de las herramientas de cómputo actuales. En el tercer capítulo se expone el funcionamiento de la herramienta de cómputo 20-sim y los toolboxes que esta contiene. En el cuarto capítulo se exponen el funcionamiento del toolbox mecánica 3D en el cual fueron desarrollados los casos de estudio. En el quinto y último capítulo se presentan las conclusiones a las que se llegaron con la investigación; y se enuncian las recomendaciones.

PALABRAS CLAVE:

Sistemas físicos

Simulación

Modelado

Herramientas de cómputo

20-sim

Bond Graphs

Tres dimensiones

ABSTRACT

The present thesis covers the subject of modeling and simulation systems in three dimensions, this work provides the ability to be used as an effective teaching method, to convey to the student's theoretical knowledge in an efficient manner, with a view to publishing a new software tool, as are the simulators, which have great capabilities, qualities of representation and that overcome various limitations in the real world, as are the costs, lack of equipment, time, among others.

The first chapter describes the impact of the simulation that it generates today, emphasizing the need for the use of computer tools. The main objective of this thesis is established, as well as the justification, the methodology and the content of it. The second chapter refers to the historical framework and the importance of the study of current computing tools. In the third chapter, the operation of the 20-sim computation tool and the toolboxes it contains is exposed. In the fourth chapter the functioning of the 3D mechanical toolbox in which the case studies were developed is exposed. In the fifth and last chapter, the conclusions reached with the research are presented; and the recommendations are enunciated.

KEYWORDS:

Physical systems

Simulation

Modeling

Computation tools

20-sim

Mathematical model

Bond graphs

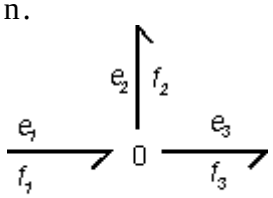
Three dimensions

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE SÍMBOLOS Y ABREVIATURAS

0 Supongamos que tenemos la siguiente unión 0 con un enlace (1) apuntando hacia la unión y dos enlaces (2 y 3) apuntando desde la unión.



$e_1 = e_2 = e_3$

$f_1 - f_2 - f_3 = 0$

1. ¡Para un cruce 0, los esfuerzos son siempre iguales! Esto significa:

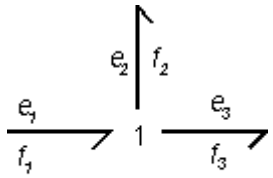
e1 = e2 = e3

2. La unión es poder continuo. Para la figura de arriba esto significa:

e1 × f1 = e2 × f2 + e2 × f2

3. Combinación de 1 y 2 rendimientos:
f1 - f2 - f3 = 0

1 Supongamos que tenemos la siguiente unión con un bond (1) apuntando hacia la unión y dos bond (2 y 3) apuntando desde la unión.



$f_1 = f_2 = f_3$

$e_1 - e_2 - e_3 = 0$

1. ¡Para un cruce, los flujos son siempre iguales! Esto significa:

$$f1 = f2 = f3$$

2. La unión es poder continuo. Para la figura de arriba esto significa:

$$e1 \times f1 = e2 \times f2 + e2 \times f2$$

3. Combinación de 1 y 2 rendimientos:

$$e1 - e2 - e3 = 0$$

2D	Dos dimensiones o bidimensional, tiene longitud y ancho, pero no profundidad.
3D	Tres dimensiones o tridimensional describe una imagen que proporciona la percepción de la profundidad.
DS	Sistemas Discretos.
Dymola	Es un entorno de modelado y simulación comercial basado en el lenguaje de modelado abierto de Modélica.
EISPACK	Es una librería para calculo numérico, escrito en FORTRAN.
GPL	Licencia Pública General GNU.
GPSS	Sistema de Simulación de propósito general.
GUI	Interfaz gráfica del usuario.
IBM	Corporación multinacional de computadoras, tecnología y consultoría de TI.
IDE	Entorno de desarrollo interactivo.
LINPACK	Librería para realizar operaciones de algebra lineal en computadoras digitales.

MATLAB	Software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio.
Mse	Este modelo representa una fuente ideal de esfuerzo modulado.
Msf	Este modelo representa una fuente de flujo modulada ideal.
NMF	Aproximación de matriz no negativa.
RAND CORPORATION	La Corporación RAND es un laboratorio de ideas estadounidense que forma a las Fuerzas Armadas de los Estados Unidos de América.
SIMSCRIPT	Es un lenguaje de programación de alto nivel diseñado específicamente para ser usado en entornos de simulación, para realizar simulaciones a gran escala.
SIMULA	Es un lenguaje de programación orientada a objetos.
SISO	Sistema que tiene una entrada y una salida.
TUTSIM	Fue el primer software de simulación comercial que se ejecutó en una PC de IBM. El paquete se usó para modelar y simular sistemas multidominio utilizando ecuaciones diferenciales y bond graphs.
UNIVAC	Computadora automática universal.
VHPL	Circuito integrado de alta velocidad.
WSC	Conferencia de simulación de invierno.

CAPITULO 1. INTRODUCCIÓN

1.1 Simulación de Sistemas

La modernidad en cuanto a tecnología se trata, se basa en los avances de la ciencia, en la capacidad humana para planear, diseñar, en construir innovadores dispositivos y sistemas; lo cual ha permitido tener una mejor interacción con el entorno, un ejemplo de ello es la simulación.

En diversos ámbitos se implementan innumerables sistemas y equipos basados en la simulación y con ello, se planean estrategias para la introducción de estas nuevas herramientas, todo esto depende de cómo sean modernizadas, así como cuales son las herramientas que demanda la sociedad en la actualidad, ya que ellos representan un factor muy importante en las diferentes áreas como la industria, los servicios de salud, la seguridad, el transporte, **la educación** y así, las diversas áreas de un país.

Al proceso que se pretende estudiar se le denomina sistema y para poderlo analizar se realiza una serie de hipótesis sobre su funcionamiento. Estas hipótesis, que normalmente se expresan mediante relaciones matemáticas o relaciones lógicas, constituyen un modelo del sistema. Este modelo se utiliza para comprender y prever el comportamiento real del sistema.

Si las relaciones matemáticas o lógicas que comprende el modelo son sencillas, entonces será posible utilizar un procedimiento analítico para obtener una solución o respuesta exacta sobre las características de interés del sistema analizado. No obstante, si las relaciones son complejas, puede ocurrir que no se pueda evaluar analíticamente el problema. En este caso, será necesario acudir a la simulación del sistema, evaluando numéricamente el modelo y analizando los datos obtenidos para estimar las características de dicho sistema.

Evidentemente, las características del sistema real que se desea estudiar van a condicionar el tipo de simulación que se va a desarrollar. Por lo tanto, conviene hacer una clasificación de los sistemas en base a los aspectos que van a condicionar su análisis posterior. Así, es útil realizar una clasificación de los sistemas atendiendo a tres aspectos fundamentales:

– **Sistemas estáticos y sistemas dinámicos** [1, pp. 10-11]

Un sistema se considera estático cuando sus variables de estado no cambian a lo largo del tiempo, es decir, cuando el tiempo no juega ningún papel en sus propiedades. Por el contrario, en un sistema dinámico los valores que toman todas o algunas de sus variables de acción evolucionan a lo largo del tiempo.

– **Sistemas deterministas y sistemas estocásticos**

Si un sistema no tiene ningún componente con características probabilistas (es decir, aleatorias) se considera determinista. En este caso, el comportamiento del sistema está determinado una vez que se hayan definido las condiciones iniciales y las relaciones que existen entre sus componentes. Por el contrario, un sistema no determinista o estocástico tiene algún elemento que se comporta de forma aleatoria, no estando predeterminado su comportamiento en función de las condiciones iniciales y de las relaciones entre sus componentes. En este caso, el sistema sólo se podrá estudiar en términos probabilistas, consiguiendo, en el mejor de los casos, conocer sus respuestas posibles con sus probabilidades asociadas.

– **Sistemas continuos y sistemas discretos**

En un sistema continuo las variables de estado cambian de forma continua a lo largo del tiempo, mientras que en uno discreto cambian instantáneamente de valor en ciertos instantes de tiempo. En un sistema de una cierta complejidad puede ocurrir que existan simultáneamente variables de estado continuas y discretas. En este caso, dependiendo de la predominancia de una y otras y del objetivo del estudio que se pretende realizar, se considerará el sistema como perteneciente a uno de los dos tipos.

El esquema de la figura 1.1 muestra las diferentes formas que se pueden utilizar para analizar un sistema.

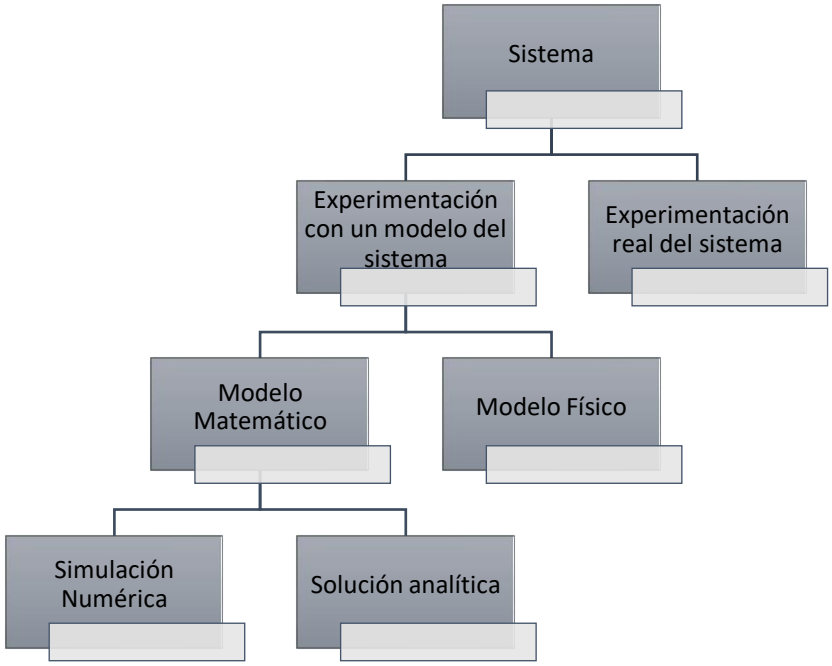


Figura 1. 1 Proceso de la simulación.

1.2 Objetivo

El desarrollo de nuevas tecnologías en el área de la computación ha traído innovaciones importantes en la toma de decisiones y el diseño de procesos. En este sentido una de las técnicas de mayor impacto es la simulación.

En la actualidad es más factible hacer un prototipo de un modelo que se llevará a cabo, ver cómo es que funciona, si es viable llevar a cabo dicho sistema. Hay muchos sistemas hoy en día son muy necesarios para la vida diaria, ejemplo de ello son, el modelado de un motor, la programación para un sistema automático, en el área de la salud, se puede realizar el prototipo de un corazón para ver su funcionamiento, son infinidades de ejemplos y áreas, en que la simulación es necesaria.

El objetivo es desarrollar una guía práctica para los alumnos de esta facultad, que con ella se pretende despertar el interés del alumnado para el uso nuevas herramientas, la cual les ayudará a fortalecer sus conocimientos en diferentes áreas.

Por lo tanto, en esta tesis se realiza el modelado, simulación y animación de sistemas basados en el software 20-sim.

1.3 Justificación

El uso de la simulación en la actualidad, permite a los usuarios poner en práctica sus conocimientos, antes de aplicarlos en un entorno real. Con este método se pretende no incurrir en altos costos o poner en riesgo el equipo con el que se pretenda trabajar. Además, este método puede ser utilizado por profesores, para transmitir a los estudiantes, conocimientos teóricos de una mejor manera. Hoy en día el manejo de herramientas de cómputo es una necesidad, ya que en algunas empresas requieren que tengas conocimientos previos de algún software. Este proyecto de investigación tiene como finalidad mostrar las bondades, facilidad de uso y los diferentes métodos de utilización de 20-sim.

Con el fin de dar una alternativa nueva, con el cual se beneficiarán al conocer los diferentes usos de la herramienta, para que puedan incursionar en el ámbito de la investigación, de igual manera podrán desarrollarse de mejor manera en el ámbito laboral.

Especialmente va enfocado a los alumnos y profesores de esta facultad para despertar el interés en conocer, aprender y realizar investigaciones muy interesantes sobre diversos temas relacionados con el mapa curricular.

1.4 Metodología

El trabajo de investigación es de tipo exploratorio, porque la herramienta a utilizar no es nueva, pero existe poca información sobre ella. A su vez es descriptiva, porque en ella se describirán y analizarán cada parte de este tema de estudio.

Todos, siempre, constante e intuitivamente en nuestra vida diaria realizamos proyectos y los llevamos a la práctica. Desde el momento que tenemos una idea, nos aqueja un problema o necesitamos enfrentar situaciones de cualquier índole; buscamos información al respecto, elaboramos alternativas, analizamos cuál de ellas nos ofrece las mejores posibilidades de éxito y finalmente, tomamos una decisión.

1.5 Contenido de la tesis

Esta investigación consta de cinco capítulos, a saber:

Capítulo I: Se establece la finalidad de esta investigación y los aspectos formales relacionados con el objeto de estudio, se describe la situación actual de las herramientas de cómputo y se resaltan algunos de los beneficios de 20-sim.

Capítulo II: Se presentan los antecedentes, un bosquejo histórico y los diferentes periodos por los que paso la simulación, así como la historia y función de algunos softwares actuales.

Capítulo III: La reseña histórica de 20-SIM, se muestra su amigable editor, las diferentes bibliotecas que contiene, sus funcionalidades.

Capítulo IV: Se presentan los casos de estudio utilizando el toolbox 3D Mecánica, donde por medio de modelado y animación se podrá observar los movimientos de los casos en los diferentes planos que este contiene.

Capítulo V: Se presentan las conclusiones y recomendaciones de esta investigación.

CAPÍTULO 2. SIMULACIÓN DE SISTEMAS FÍSICOS

2.1 Antecedentes [2]

Se podría considerar que la simulación nace en 1777 con el planteamiento del problema "la aguja de buffon", un método matemático sencillo para ir aproximando el valor del número π a partir de sucesivos intentos.

Este modelo matemático 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 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 Guinness, ya aplicaba sus conocimientos estadísticos en la destilería y en su propia explotación agrícola. El especial interés de Gosset en el cultivo de la cebada le llevó a especular que el diseño de experimentos debería dirigirse no sólo a mejorar la producción media, sino también a desarrollar variedades de cebada cuya mayor robustez permitiese que la producción no se viese afectada por las variaciones en el suelo y el clima.

Para evitar futuras filtraciones de información confidencial, Guinness prohibió a sus empleados la publicación de cualquier tipo de artículo independientemente de su contenido, de ahí el uso que hizo Gosset en

[1] Pierre-Simón Laplace (1749-1827). Astrónomo, físico y matemático francés. Descubrió y desarrolló la transformada de Laplace y la ecuación de Laplace; como estadístico sentó las bases de la teoría analítica de la probabilidad; y como astrónomo planteó la teoría nebular sobre la formación del sistema solar.

[2] Georges Louis Leclerc, Conde de Buffon (1707-1788). Naturalista, botánico, matemático, biólogo, cosmólogo y escritor francés.

[3] William Sealy Gosset (1876-1937). Estadístico, mejor conocido por su sobrenombre literario Student. Su logro más famoso se conoce ahora como la distribución t de Student, que de otra manera hubiera sido la distribución t de Gosset.

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

Este acontecimiento histórico abrió las puertas a la aplicación de la simulación en el campo del proceso de control industrial, así como a las sinergias que generaba esta simulación basada en la experimentación y técnicas de análisis para descubrir soluciones exactas a problemas clásicos de la industria y la ingeniería. En la figura 2.1 se ilustra los iniciadores de la simulación de procesos.

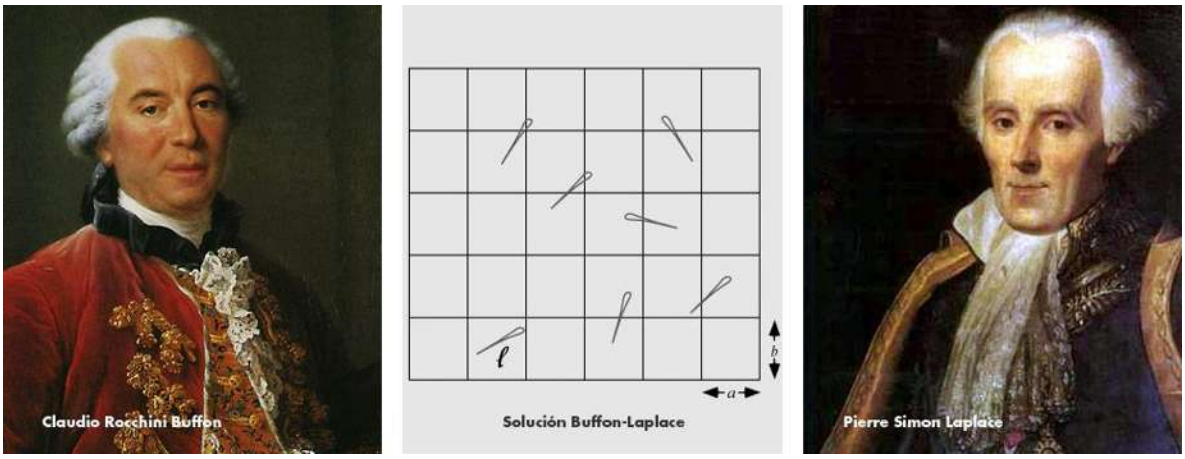


Figura 2. 1 Precursores de la simulación

2.2 Bosquejo Histórico

2.2.1 Periodo de Formación (1945-1970) [2]

A) ENIAC y el Método de 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 como el ENIAC.
- El trabajo de Stanislaw Ulam[4], John Von Neumann[5] y otros científicos para usar el método de Montecarlo en computadores

[4] Stanisław Marcin Ulam (1909–1984). Matemático polaco que participó en el proyecto Manhattan y propuso el diseño Teller–Ulam de las armas termonucleares. Sobre todo, es conocido por ser coautor (con Nicholas Metropolis) del método de Montecarlo.

[5] John Von Neumann (1903-1957). Matemático húngaro-estadounidense que realizó contribuciones fundamentales en física cuántica, análisis funcional, teoría

modernos y solucionar problemas de difusión de neutrones en el diseño y desarrollo de la bomba de hidrógeno. Ulam y Von Neumann ya estuvieron presentes en el proyecto Manhattan. En la figura 2.2 se ilustra la construcción de los primeros computadores.

Figura 2. 2 Construcción de los primeros computadores.

B) El Arte de la Simulación

En 1960, Keith Douglas Tocher desarrolló un programa de simulación general cuya principal tarea era la de simular el funcionamiento de una planta de producción donde las máquinas ciclaban por estados: Ocupado, Esperando, No disponible y Fallo; de manera que las simulaciones en los cambios de estado de las máquinas marcarán el estado definitivo de la producción de la planta. Este trabajo produjo además el primer libro sobre simulación: The Art of Simulation (1963). En la figura 2.3 se ilustra a Keith Douglas Tocher[6].

de conjuntos, teoría de juegos, ciencias de la computación, economía, análisis numérico, cibernética, hidrodinámica, estadística y muchos otros campos. Es considerado como uno de los más importantes matemáticos de la historia moderna.

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

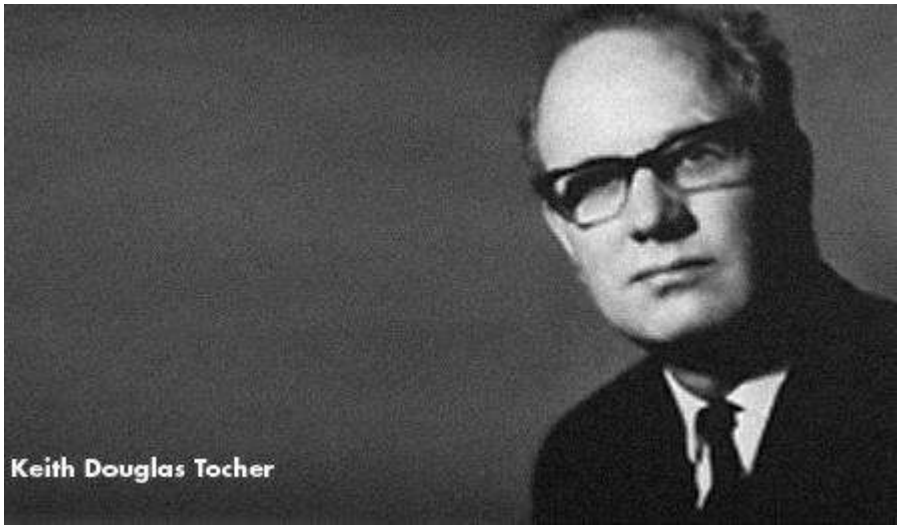


Figura 2. 3 Keith Douglas Tocher

C) Simulación de Sistemas de Propósito General y SIMSCRIPT

Para aquel entonces, IBM desarrolló entre 1960 y 1961 el Sistema de Simulación de propósito general o General Purpose Simulation System (GPSS). El GPSS se diseñó para realizar simulaciones de teleprocesos involucrando, por ejemplo: control de tráfico urbano, gestión de llamadas telefónicas, reservas de billetes de avión, etc. La sencillez de uso de este sistema lo popularizó como el lenguaje de simulación más usado de la época.

Por otro lado, en 1963 se desarrolló SIMSCRIPT, otra tecnología alternativa al GPSS basada en FORTRAN, más enfocada a usuarios que no tenían por qué ser obligatoriamente expertos informáticos en RAND CORPORATION. En la figura 2.4 se ilustra el lenguaje de programación FORTRAN.

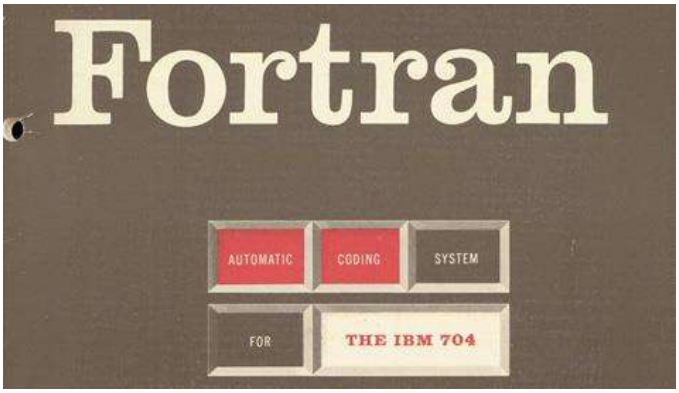


Figura 2. 4 Lenguaje de programación FORTRAN.

D) SIMULA I y WSC

Complementariamente a los desarrollos llevados a cabo por RAND e IBM, el Royal Norwegian Computing Center inició 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 el referente en lo que a avances en el campo de los sistemas de simulación se refiere. En la figura 2.5 se ilustran los desarrolladores de SIMULA.

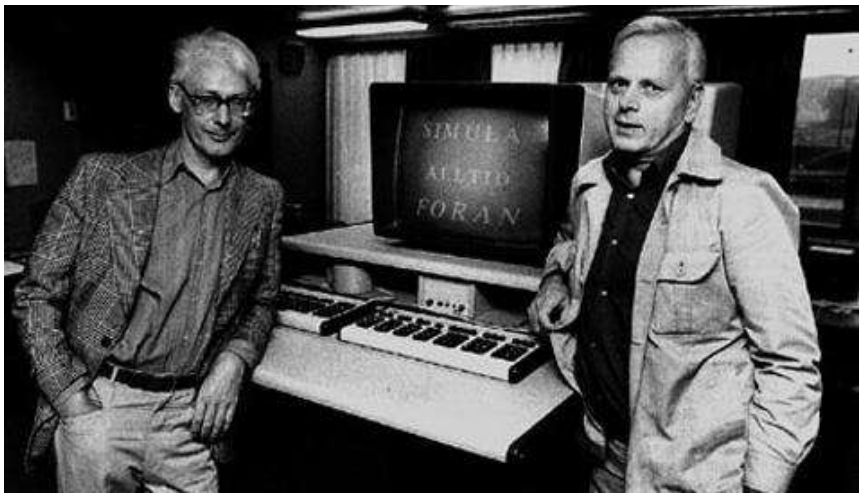


Figura 2. 5 Desarrollo de Simula.

2.2.2 Periodo de Expansión (1970-1981) [2]

A) Aplicaciones en múltiples campos

Durante este periodo se desarrollaron avanzadas herramientas de modelado y de análisis de resultados. Gracias también a los desarrollos obtenidos en la generación de datos y a las técnicas de optimización y representación de datos, la simulación llega a su fase de expansión donde comienza a aplicarse en múltiples campos.

B) Imagen en movimiento

Anteriormente, los datos de salida obtenidos de una simulación por computadora se presentaban en una tabla o 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, los psicólogos advirtieron que los seres humanos percibían mejor los cambios 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 de imágenes generadas por computadora.

2.3 Software de Simulación Actuales

A) 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/ [3]Linux.

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).

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 matemático 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. [3]

B) MODÉLICA

Modélica es un lenguaje de modelado orientado a objetos, declarativo, multi-dominio para el modelado orientado a componentes de sistemas complejos, por ejemplo, sistemas que contienen subcomponentes mecánicos, eléctricos, electrónicos, hidráulicos, térmicos, de control, eléctricos u orientados al proceso.

El esfuerzo de diseño de Modélica 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. Modélica 1.0 se basa en la tesis de PhD 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 Modélica, pero muchas otras personas también han contribuido. [4]

C) LabVIEW

LabVIEW es un software de ingeniería de sistemas que requiere pruebas, medidas y control con acceso rápido a hardware y análisis de datos.

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

puede interactuar o reutilizar bibliotecas de otros software y lenguajes de fuente abierta. [5]

D) Scilab

Scilab es un software libre y de código abierto para la computación numérica que proporciona un potente entorno de computación para aplicaciones científicas y de ingeniería.

Scilab es liberado como código abierto bajo la licencia CeCILL (compatible con GPL), y está disponible para descarga gratuita. Scilab está disponible bajo GNU / Linux, Mac OS X y Windows XP / Vista / 7/8

Las características de Scilab incluyen análisis numérico, visualización 2-D y 3-D, 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. [6]

E) GNU Octave

GNU Octave es un lenguaje de alto nivel, destinado principalmente a cálculos numéricos. Proporciona una interfaz de línea de comandos conveniente para resolver problemas lineales y no lineales numéricamente, y para realizar otros experimentos numéricos utilizando un lenguaje que es en su mayoría compatible con Matlab. También puede utilizarse como un lenguaje orientado por 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 ordinarias diferenciales y algébricas diferenciales. Es fácilmente extensible y personalizable a través de funciones definidas por el usuario escritas en el propio idioma de Octave, o utilizando módulos cargados dinámicamente escritos en C ++, C, Fortran u otros idiomas.

GNU Octave también es un software libremente redistribuible. Puede redistribuirlo y / o modificarlo bajo los términos de la Licencia Pública General GNU (GPL) publicada por la Free Software Foundation.

Octave fue escrito por John W. Eaton y muchos otros. Debido a que Octave es un software libre, se le anima a ayudar a que Octave sea más útil escribiendo y contribuyendo funciones adicionales para ella, y reportando cualquier problema que pueda tener. [7]

El avance en los sistemas computacionales facilitó el desarrollo de entornos software de modelado y simulación con DS. En sus inicios, estas herramientas facilitaban la labor de la simulación permitiéndole al modelador introducir las ecuaciones diferenciales o sistema de ecuaciones, para poder ser resueltos con sus algoritmos de métodos numéricos y luego entregar los resultados de la simulación.

Posteriormente, estas herramientas evolucionaron para brindar soporte, no solo a la simulación, sino además para el modelado y el análisis de sensibilidad, entre otras. Igualmente, se han adaptado a las necesidades específicas de los usuarios; por ejemplo, para el modelado y simulación de diversos fenómenos organizacionales, lo cual facilitó extenderse a sectores como el empresarial e industrial.

Dentro de las herramientas más utilizadas en el ámbito académico y empresarial, podemos nombrar, en orden alfabético, a:

- AnyLogic (AnyLogic, 2010);
- Evolución (Evolución, 2010);
- iThink/Stella (ISEE Systems, 2010);
- Powersim (PowerSim, 2010);
- Simile (Simile, 2010);
- Vensim (VenSim, 2010).

Actualmente estas herramientas ofrecen diferentes servicios, por medio de un entorno intuitivo para el usuario. Entre las principales prestaciones se encuentran: herramientas para el modelado, como los editores para la creación de diagramas causales y diagramas de flujo-nivel y el uso de funciones matemáticas. Herramientas para realizar y controlar la simulación del modelo. Al momento de realizar el análisis del modelo y su comportamiento, se observa que existen diferentes herramientas para este propósito. Para modelos complejos estos mecanismos de análisis son de gran de ayuda para el entendimiento del comportamiento, depuración y ajuste del modelo. [8, pp. 6-7]

Tabla 2- 1 Funcionalidades de los softwares actuales

Software/Versión	Organización	Simulación	Herramientas de análisis
		• Modelado basado en eventos • Modelado físico • Sistemas de control • Procesamiento de Señales y Comunicaciones • Código de GENERACIÓN • Simulación y pruebas en tiempo real • Verificación, Validación y Prueba • Gráficos de Simulación e Informes	• Optimización • Sistemas de control • Procesamiento de Señales y Comunicaciones • Procesamiento de imágenes y visión por ordenador • Prueba y medición • Finanzas Computacionales • Biología Computacional • Generación y verificación de código • Implementación de aplicaciones • Conectividad y generación de informes de bases de datos • Generador de informes MATLAB
		• Sistemas termo-fluidos y energéticos. • Los entornos de simulación de Modélica están disponibles comercialmente y de forma gratuita, como los sistemas CATIA, Dymola, LMS AMESim, JModelica.org, MapleSim, OpenModelica, SCICOS, SimulationX y Wolfram SystemModeler. Los modelos de Modélica se pueden importar convenientemente en Simulink	

		usando las funciones de exportación de Dymola, MapleSim y SimulationX.	
	National Instruments		Interfaces de comunicaciones: • Puerto serie • Puerto paralelo • GPIB • PXI • VXI • TCP/IP, UDP, DataSocket • Irda • Bluetooth • USB • OPC... Capacidad de interactuar con otros lenguajes y aplicaciones: • DLL: librerías de funciones • .NET • ActiveX • Multisim • Matlab/Simulink • AutoCAD, SolidWorks, etc • Herramientas gráficas y textuales para el procesado digital de señales. • Visualización y manejo de gráficas con datos dinámicos. • Adquisición y tratamiento de imágenes. • Control de movimiento (combinado incluso con todo lo anterior).

			• Tiempo Real estrictamente hablando. • Programación de FPGAs para control o validación. • Sincronización entre dispositivos.
	Scilab Enterprises	Xcos - Sistemas dinámicos híbridos Modelado de sistemas mecánicos, circuitos hidráulicos, sistemas de control	• Matemáticas y Simulación • Visualización 2-D y 3-D • Mejoramiento • Estadística • Diseño y Análisis de Sistemas de Control • Procesamiento de la señal • Desarrollo de aplicaciones
			• Octave está escrito en C++ usando la librería STL. • Tiene un intérprete de su propio lenguaje (de sintaxis similar a Matlab), y permite una ejecución interactiva o por lotes. • Puede extenderse el lenguaje con funciones y procedimientos por medios de módulos dinámicos. • Utiliza otros programas GNU para ofrecer al

			usuario crear gráficos para luego imprimirlos o guardarlos (Grace). • Dentro del lenguaje también se comporta como una consola de órdenes (shell). Esto permite listar contenidos de directorios, por ejemplo. • Además de correr en plataformas Unix también lo hace en Windows. • Puede cargar archivos con funciones de Matlab de extensión .m.

CAPITULO 3. MODELADO DE SISTEMAS EN 20-SIM

3.1 ¿Qué es 20-sim? [9]

Es un programa de modelado y simulación que se ejecuta bajo Microsoft Windows, desarrollado por Controllab. Con 20-sim puede simular el comportamiento de sistemas dinámicos, tales como sistemas eléctricos, mecánicos e hidráulicos o cualquier combinación de estos.

20-sim soporta totalmente el modelado gráfico, permitiendo diseñar y analizar sistemas dinámicos de una manera intuitiva y fácil de usar, sin comprometer la potencia. 20-sim soporta el uso de componentes. Esto le permite introducir modelos como en un esquema de ingeniería: al elegir componentes de la biblioteca y conectarlos, su esquema de ingeniería es en realidad reconstruido, sin entrar en una sola línea de matemáticas.

20-sim es la mejora del paquete de software de simulación TUTSIM, que fue desarrollado en el Laboratorio de Control de la Universidad de Twente. Mientras que TUTSIM se vendió a finales de los años 70, la investigación en el modelado y la simulación continuó en el laboratorio. Un nuevo programa fue desarrollado como parte del Ph. D, Proyecto de Jan Broenink. El programa fue equipado con una interfaz gráfica de usuario y permitió la creación de modelos por bond graphs. El nombre del prototipo para este paquete de modelado y simulación fue CAMAS. Mientras que CAMAS fue construido alrededor de los bond graphs, un nuevo prototipo llamado MAX se desarrolló para investigar técnicas de modelado orientadas a objetos y modelado por diagramas icónicos. Después de una extensa prueba, en agosto de 1995 la versión 1.0 del software fue lanzada comercialmente bajo el nombre comercial 20-sim (Twente Sim). El nombre comercial se refiere al origen (Universidad de Twente) del paquete y la región (Twente) donde se hizo. La compañía Controllab Products fue establecida para desarrollar y distribuir el paquete.

20-sim viene con un número de toolbox integrados que ayudan a diseñar y analizar modelos.

- Gráficos en 3D
- Control
- Generación de código en C
- Dominio de la frecuencia
- Animación

3.1.1 EDITOR [10]

Al iniciar 20-sim consta de dos ventanas principales (Editor y Simulador) y un buen número de herramientas. El Editor se abre al iniciar 20-sim. En el editor puede crear sus modelos.

Los modelos se introducen y compilan en el Editor de 20-sim. El Editor es una herramienta versátil que le ayuda a introducir modelos que soportan una amplia variedad de sistemas, incluyendo sistemas lineales, no lineales, de tiempo discreto, de tiempo continuo e híbridos, sin restringir al usuario a una representación determinada del modelo. Después de entrar y depurar, el modelo puede ser comprobado y compilado. Esto se realiza automáticamente en segundo plano, al abrir el Simulador. En la figura 3.1 se ilustra el editor de 20-SIM.

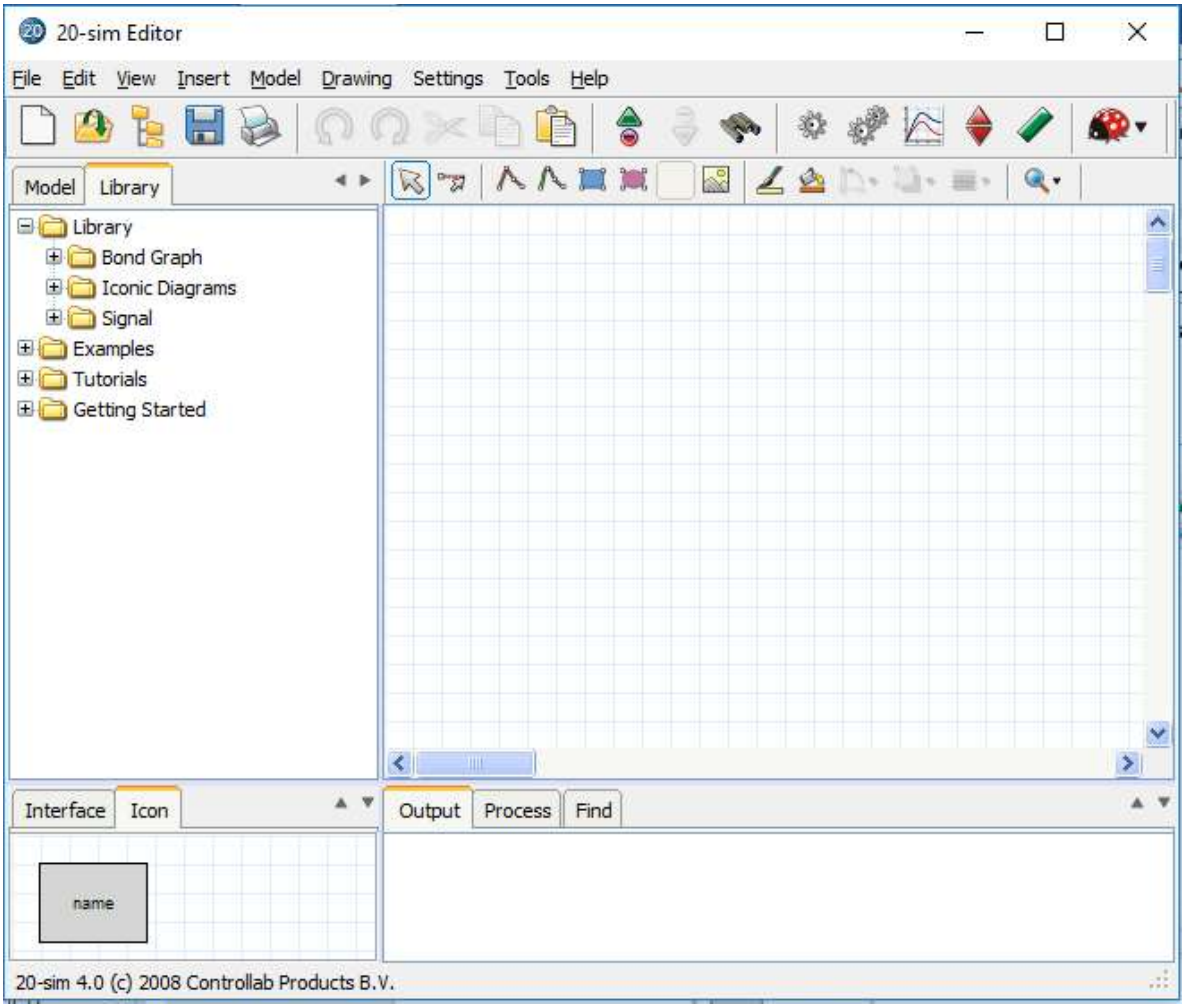


Figura 3. 1 Editor de 20-SIM.

El Editor consta de cuatro partes:

a) **Pestaña Modelo / Pestaña Biblioteca:** Esta es la parte que se encuentra en la mitad izquierda. La pestaña Modelo muestra la jerarquía del modelo, es decir, la composición jerárquica (todos los elementos) del modelo que se crea en el Editor. La pestaña Biblioteca muestra la biblioteca 20-sim.

b) **Editor Gráfico / Editor de Ecuación:** Este es el gran espacio en blanco en la parte central derecha. En este editor puede crear modelos gráficos e introducir ecuaciones.

c) **Pestaña de Salida / Pestaña de Proceso / Pestaña de Buscar:** Esta en la parte inferior derecha. La pestaña Salida muestra los archivos que se abren y almacenan. La pestaña Proceso muestra los mensajes del compilador. La pestaña Buscar muestra los resultados de la búsqueda.

d) **Pestaña Interfaz / Pestaña Icono:** Esta es la parte en la parte inferior izquierda. La pestaña Interface muestra la interfaz de un modelo seleccionado. Si hace doble clic, se abrirá el editor de interfaces. La pestaña Icono muestra el icono de un modelo seleccionado. Haciendo doble clic, se abrirá el icono.

Los sistemas pueden ser modelados en 20-sim, usando ecuaciones, descripciones de espacios de estados, bond graphs y componentes o diagramas icónicos. Estas descripciones pueden acoplarse completamente para crear modelos mixtos.

Probaremos algunos de los ejemplos que 20-SIM tiene en una de sus diferentes bibliotecas, arrastre y suelte el modelo del controlador discreto al editor gráfico. Enseguida se abre el modelo. Su Editor debería tener el siguiente aspecto.

En la figura 3.2 se ilustra el editor con un sistema de control discreto.

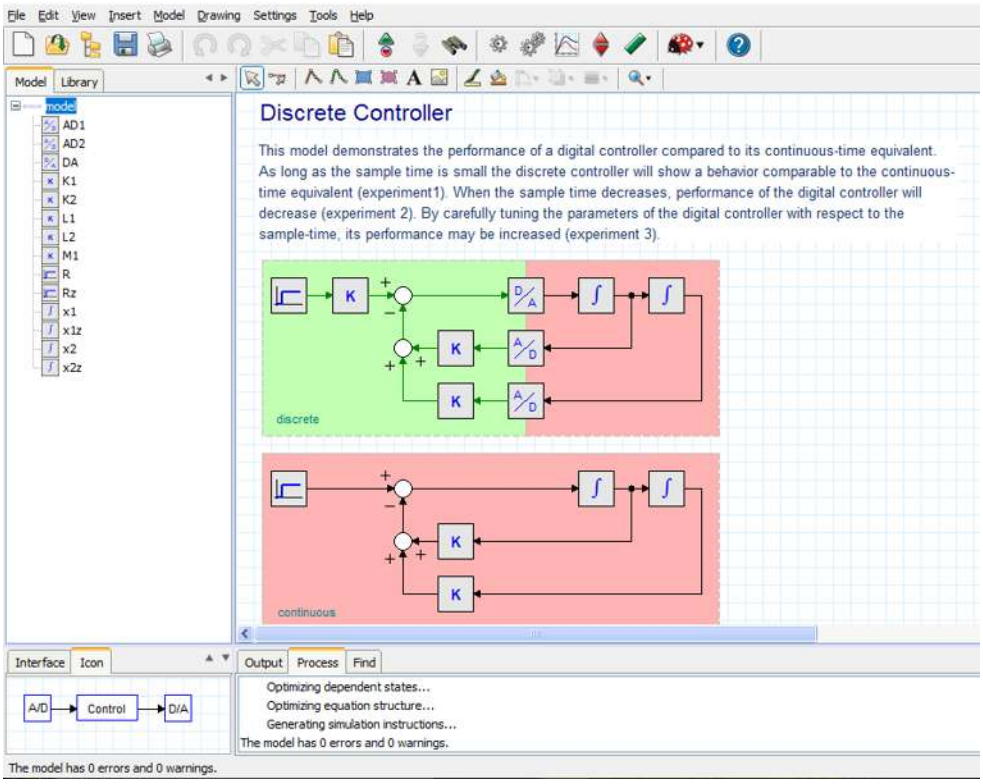


Figura 3. 2 Editor con el modelo de control discreto.

En el menú de nuestro modelo seleccionaremos la opción **Start Simulator**, enseguida el simulador se abrirá. En la figura 3.3 se ilustra el simulador con el modelo de control discreto.

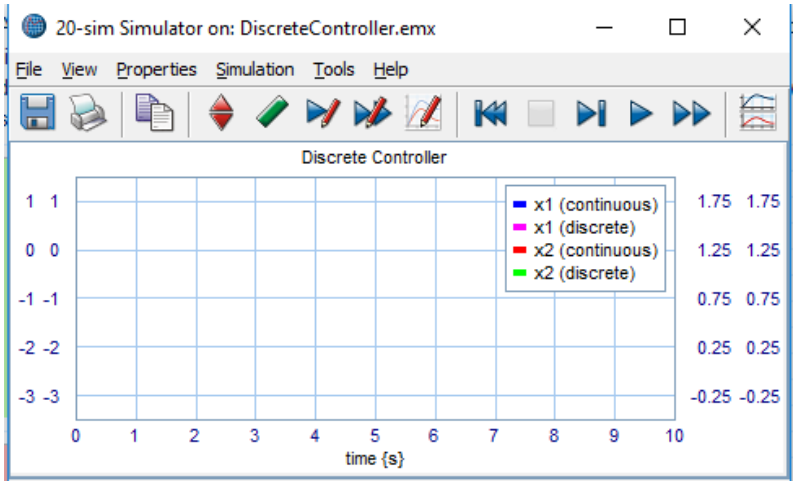


Figura 3. 3 Simulador con el modelo de control discreto.

En el Simulador se puede ejecutar una simulación y mostrar los resultados en tramas y animaciones.

El Simulador contiene varias herramientas para analizar los resultados de la simulación.

En el menú del modelo seleccione Ejecutar/Run. Ahora se ejecutará una simulación. Su Simulador debe tener el siguiente aspecto. En la figura 3.4 se ilustra el resultado de la simulación.

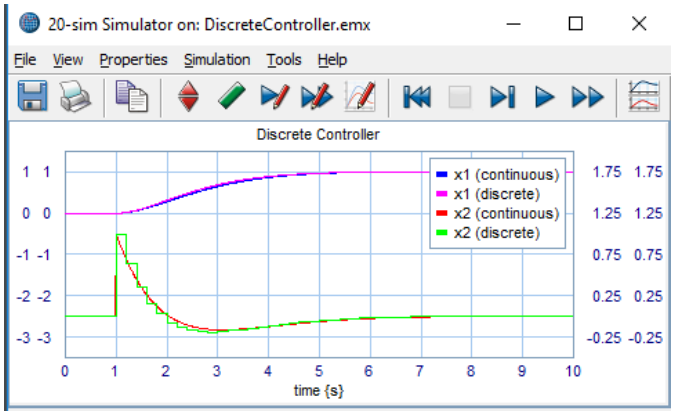


Figura 3. 4 Resultados de la simulación.

3.1.2 BIBLIOTECA

En 20-sim, crear modelos solo requiere de unos pocos clics del mouse. Al arrastrar un elemento de la biblioteca y soltarlo en el editor gráfico, su modelo se construye de la misma manera que dibujaría un esquema de ingeniería. 20-sim soporta diversas representaciones de modelos, tales como diagramas de bloques y diagramas icónicos. Estas representaciones pueden combinarse en un modelo. [11]

Buscador de la biblioteca

Todos los modelos en 20-sim se almacenan en archivos con la extensión .emx. Las librerías estándar se pueden encontrar en la carpeta 20-sim:

C:\Program Files\20-sim 4.0\Models

O en sistemas de 64 bits:

C:\Program Files (x86) \20-sim 4.0\Models

Esta carpeta contiene todos los modelos que están visibles en el Explorador de bibliotecas en la parte izquierda del Editor. En la figura 3.5 se ilustra el contenido de la librería de 20-SIM.

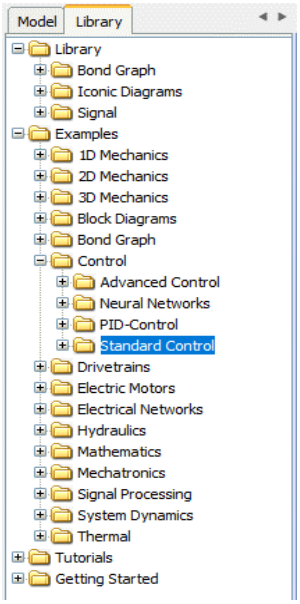


Figura 3. 5 Contenido de la librería de 20-sim.

La biblioteca contiene 4 secciones:

→ **Bond Graph**: elementos de bond graph.
→ **Diagramas Icónicos:** Componentes Físicos.
→ **Señal**: elementos del diagrama de bloques.
→ **Tutorial:** ejemplos de modelos que muestran cómo realizar varias tareas en 20-sim.
→ **Introducción:** todos los modelos que necesita en las lecciones del manual de introducción.

A) MODELOS DE CÓDIGO ABIERTO

Todos los modelos de biblioteca son de código abierto. Puede inspeccionar el contenido de cualquier modelo en el Editor. Si el modelo contiene una jerarquía, puede utilizar el comando **Ir** hacia abajo del menú Modelo para descender en la jerarquía. Si un modelo abre un editor específico, todavía puede inspeccionar el código subyacente

manteniendo la tecla **Mayús** presionada mientras hace clic en el comando **Ir** hacia abajo.

B) BIBLIOTECAS PERSONALIZADAS

Se pueden crear sus propias librerías de modelos en 20-sim:

1. En el menú Herramientas, haga clic en Opciones - Carpetas - Carpetas de biblioteca.
2. Agregue su carpeta.
3. Dele un nombre útil haciendo clic en Editar etiqueta.
4. Haga clic en Aceptar para cerrar el cuadro de diálogo.

A continuación, puede agregar sus propios modelos de biblioteca a la biblioteca:

5. Seleccione el submodelo que desea almacenar en su biblioteca.
6. En el menú Archivo, seleccione Guardar submodelo.
7. Guarde el submodelo en la carpeta de su biblioteca.

La próxima vez que inicie 20-sim, la biblioteca mostrará el nuevo submodelo. [11]

3.2 DIAGRAMAS DE BLOQUES [12]

Los diagramas de bloques le permiten representar gráficamente las relaciones matemáticas entre señales en un sistema. Son especialmente adecuados para sistemas de control de modelos. En 20-sim una gran biblioteca de elementos de diagrama de bloques está disponible. Los elementos se muestran en el Editor mediante iconos. Puede crear modelos de diagrama de bloques arrastrando los elementos al Editor gráfico y haciendo las conexiones adecuadas entre los elementos.

3.2.1 BIBLIOTECA DE DIAGRAMA DE BLOQUES

20-sim tiene una gran biblioteca de elementos de diagrama de bloques tales como elementos lineales, no lineales, discretos y de origen. En 20-sim, puede crear elementos de diagrama de bloques personalizados y agregarlos a las bibliotecas existentes o combinarlos en bibliotecas nuevas.

En la figura 3.6 se ilustra la biblioteca de los diagramas de bloques.

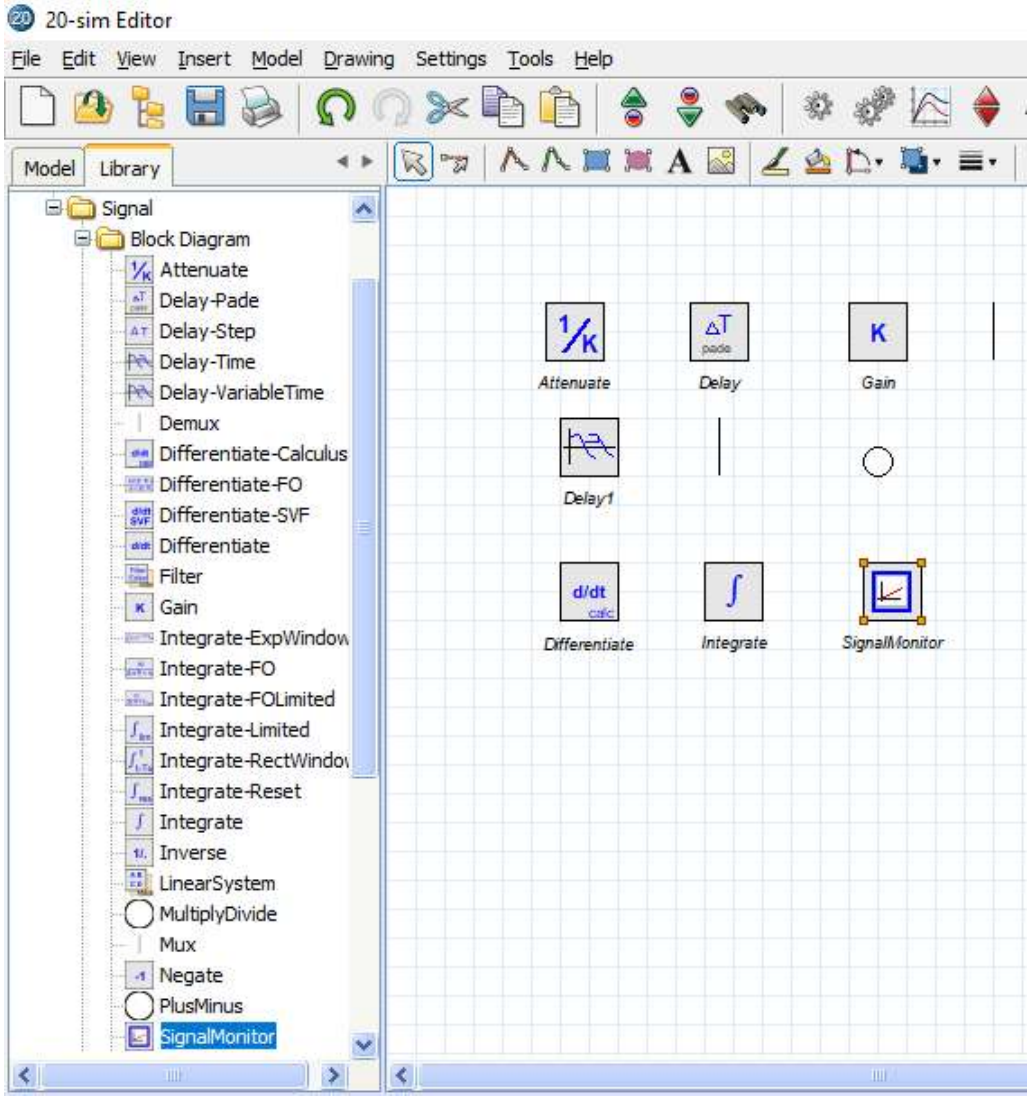


Figura 3. 6 Desde el navegador de la Biblioteca (izquierda) puede arrastrar y soltar los elementos en el Editor Gráfico (derecha).

Señales

La base de los elementos del diagrama de bloques es el uso de señales de entrada y salida. 20-sim le permite crear elementos de diagrama de bloques definidos por el usuario con un número arbitrario de señales de entrada y salida. El tamaño de la señal puede ser 1 (predeterminado) o mayores.

En la figura 3.7 se ilustra como de manera predeterminada el tamaño de la señal es 1, pero nosotros como usuarios podemos cambiar dicho tamaño por el que se requiera.

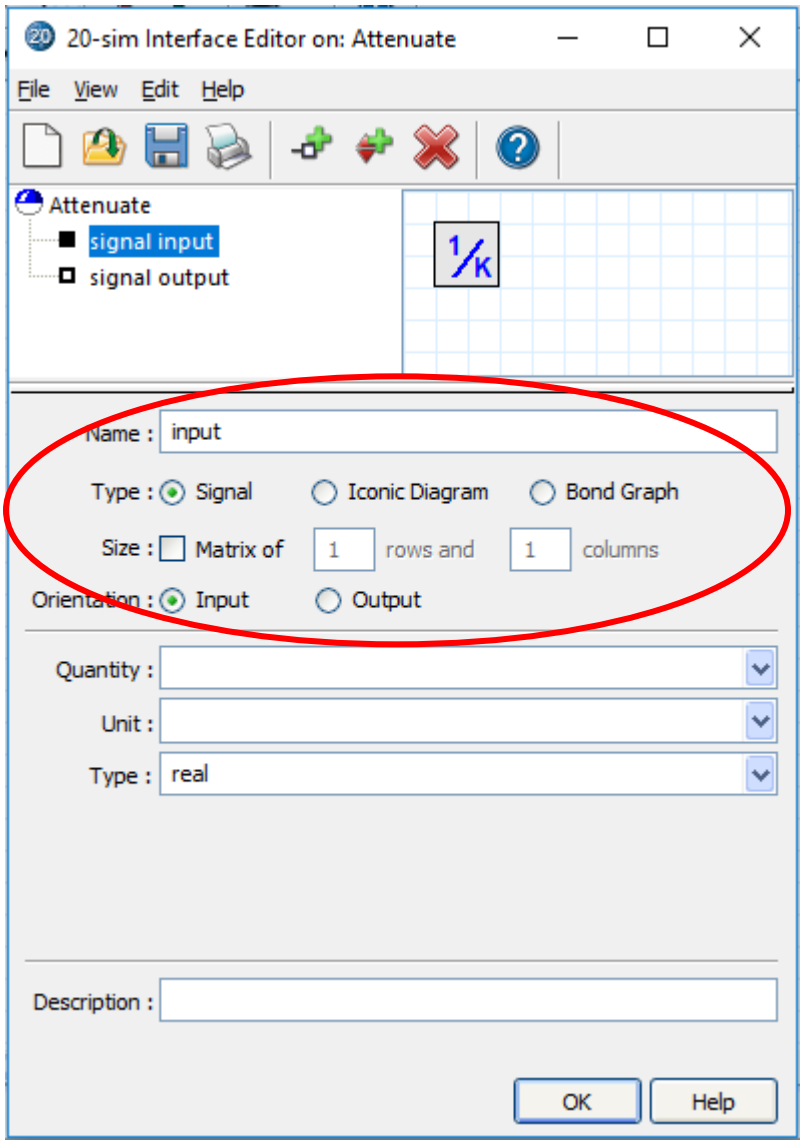


Figura 3. 7 Ajuste del tamaño de las señales.

Modelos a medida

En 20-sim puede crear sus propios elementos de diagrama de bloques y guardarlos en su propia biblioteca de modelos. Los modelos pueden tener un número arbitrario de puertos, señales de entrada y salida. Se puede utilizar un editor de dibujo especializado para dar a los modelos cualquier tipo de representación.

3.3 DIAGRAMAS ICÓNICOS [13]

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

3.3.1 BIBLIOTECA DE DIAGRAMAS ICÓNICOS

20-sim tiene una gran biblioteca de elementos del diagrama icónico tales como modelos eléctricos, hidráulicos, mecánicos y térmicos. En 20-sim, puede crear elementos de diagrama icónico personalizados y agregarlos a las bibliotecas existentes o combinarlos en bibliotecas nuevas. En la figura 3.8 se ilustra el contenido de la librería de diagramas icónicos.

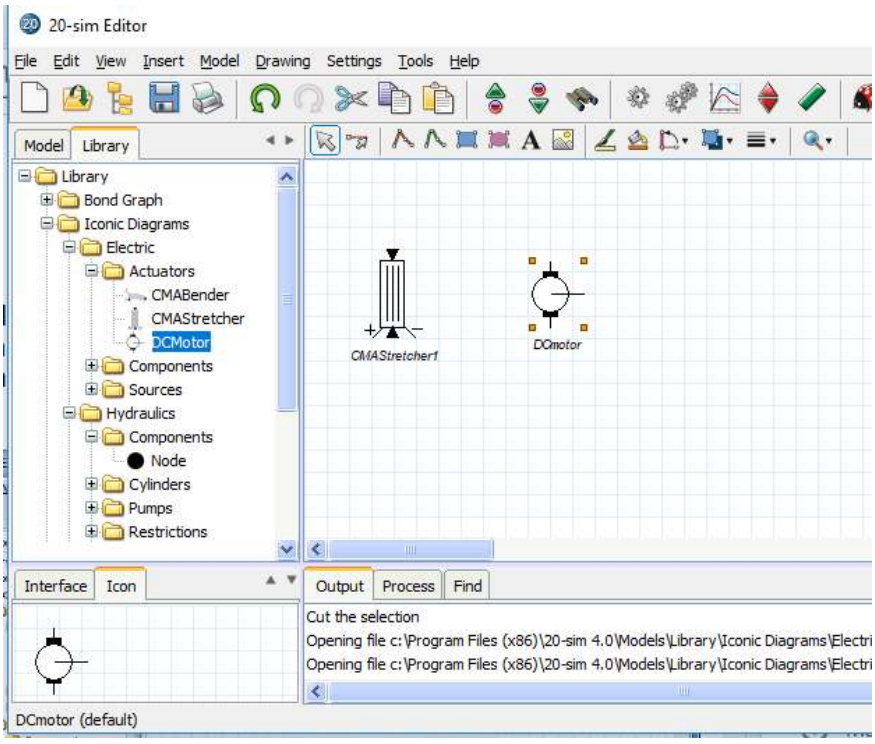


Figura 3. 8Desde el Navegador de la Biblioteca (izquierda) puede arrastrar y soltar elementos en el Editor Gráfico (derecha).

Puertos y Multipuertos

La base de los elementos del diagrama icónico es el uso de puertos de alimentación. Los puertos de alimentación permiten la conexión entre elementos describiendo el flujo de potencia entre los elementos. Un puerto de alimentación consta de dos señales que se llaman de principio a fin. 20-sim le permite crear elementos de diagrama icónico definidos por el usuario con un número arbitrario de puertos de alimentación. Los tamaños de los puertos pueden ser 1 (predeterminado) o mayor (multipuertos).

3.3.2 LAZOS ALGEBRAICOS Y CAUSALIDAD DIFERENCIAL

Los circuitos algebraicos y las causalidades diferenciales se trazan automáticamente. Si es posible, 20-sim volverá a escribir las ecuaciones simbólicamente para eliminar los lazos algebraicos y causalidades diferenciales.

Modelos a medida

En 20-sim puedes crear tus propios elementos de diagrama icónico y guardarlos en tu propia biblioteca de modelos. Los modelos pueden tener un número arbitrario de puertos, señales de entrada y salida.

3.4 BOND GRAPHS [14]

20-sim fue el primer paquete de software lanzado comercialmente para soportar el modelado de bond graphs. La primera versión de 20-sim con una biblioteca de bond graphs fue lanzada en 1995.

Desde entonces, un esfuerzo continuo para mejorar el modelado de los bond graphs ha hecho de 20-sim el estándar en el modelado de los bond graphs.

Los bond graphs son una descripción similar a una red de sistemas físicos en términos de procesos físicos ideales. Con el método del bond graph, las características del sistema se dividen en un conjunto (imaginario) de elementos separados. Cada elemento describe un proceso físico idealizado. Para facilitar el dibujo de las bond graphs, los elementos comunes se indican mediante símbolos especiales.

3.4.1 BIBLIOTECA DE BOND GRAPH

20-sim tiene una gran biblioteca que contiene todos los elementos de bond graph estándar. Junto a los elementos estándar, 20-sim soporta modelos de bond graphs hechos por el usuario. En la figura 3.9 se ilustra el contenido de la librería de Bond Graph.

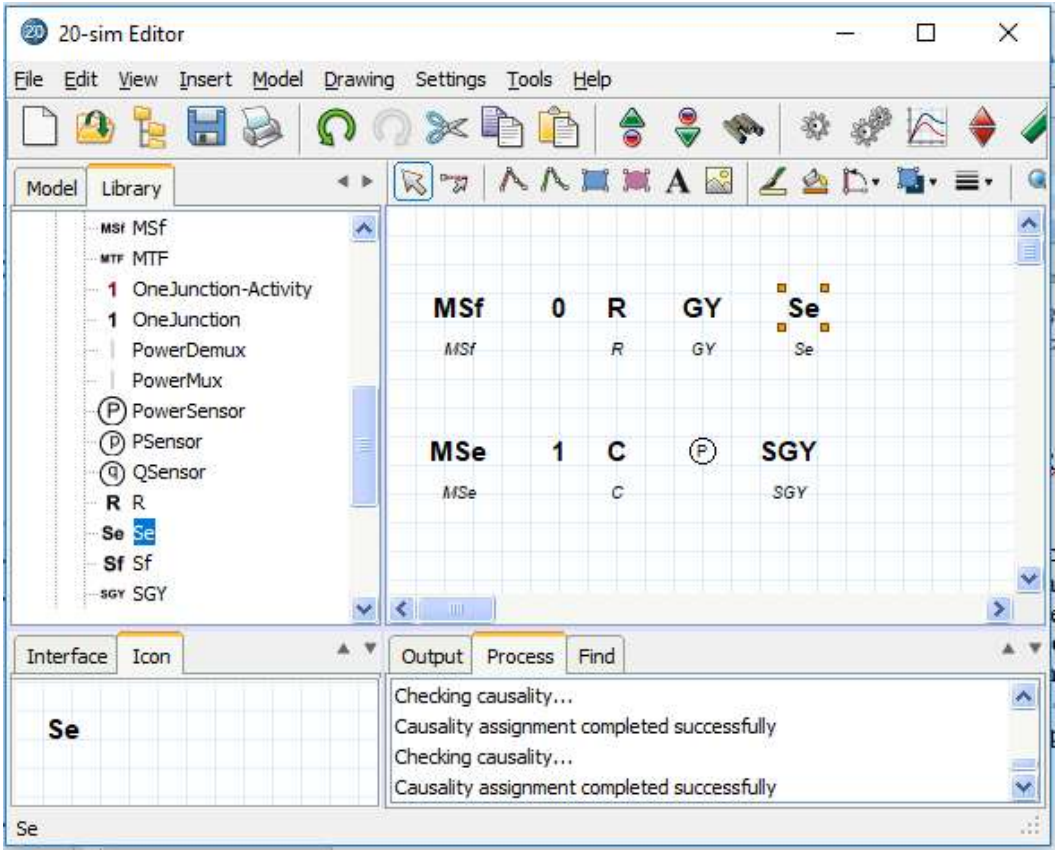


Figura 3. 9 Desde el Navegador de la Biblioteca (izquierda) puede arrastrar y soltar elementos en el Editor Gráfico (derecha), Librería de Bond Graph.

3.4.2 PUERTOS Y MULTIPUERTOS

El fundamento del modelado de bond graph es el uso de los puertos de la energía. Los puertos de alimentación forman las conexiones con otros elementos de bond graph y consisten en dos señales que se llaman esfuerzo y flujo y se multiplican a la potencia. 20-sim le permite crear modelos definidos por el usuario con un número arbitrario de puertos de alimentación y señales. Los tamaños de los puertos pueden ser 1 (por defecto) o más grandes (multipuertos). Para cada puerto se puede especificar la causalidad como prefijo fijo, indiferente o dependiendo de la causalidad de otros puertos.

3.4.3 Causalidad

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

En la figura 3.10 se ilustra que el software asigna automáticamente la causalidad a cada uno de los elementos del modelo.

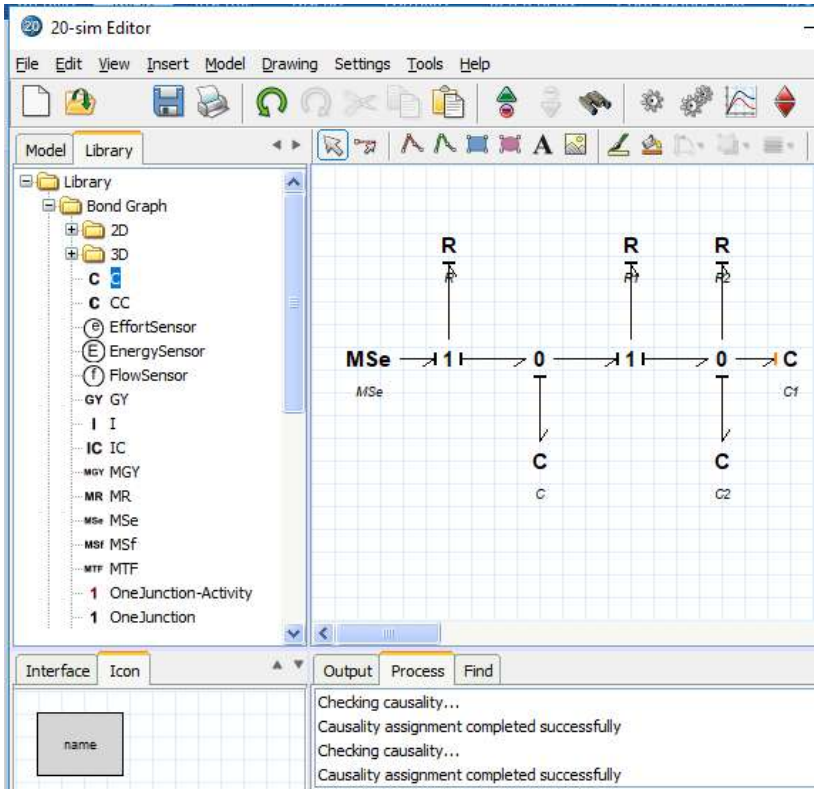


Figura 3. 10 20-sim asignará la causalidad automáticamente a su modelo de bond graph

3.4.4 Lazos Algebraicos y Causalidad Diferencial

Los circuitos algebraicos y las causalidades diferenciales se trazan automáticamente. Si es posible, 20-sim volverá a escribir las ecuaciones simbólicamente para eliminar los lazos algebraicos y causalidades diferenciales.

Modelos a medida

En 20-sim puede crear sus propios modelos de bond graph y guardarlos en una biblioteca de modelos personalizados. Los modelos pueden tener un número arbitrario de puertos, señales de entrada y salida. Se puede utilizar un editor de dibujo especializado para dar a los modelos cualquier tipo de representación.

En la figura 3.11 se ilustra el ejemplo de la bomba de un acuario.

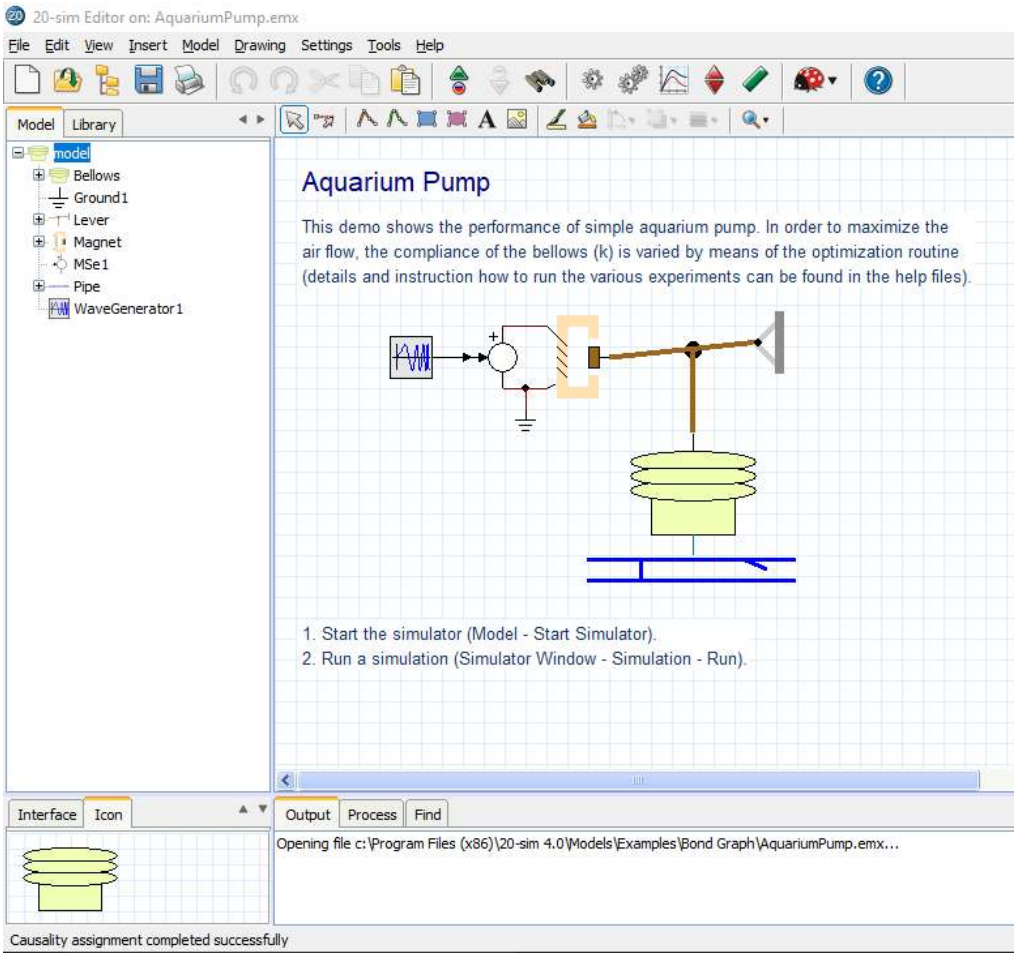


Figura 3. 11 20-sim permite crear elementos de bond graph personalizados

3.5 TOOLBOXES [15]

20-sim contiene una serie de Toolbox:

1. **Toolbox de Mecánica 3D:** Esta caja de herramientas consiste en el Editor de Mecánica 3D.
2. **Toolbox de Animación:** Esta caja de herramientas consiste en las herramientas Animación 3D y Animación gráfica.
3. **Toolbox de control:** Esta caja de herramientas está formada por el editor de diseño del controlador, el editor de red MLP, el editor B-Spline y el editor de filtros.
4. **Toolbox de dominio de frecuencia**: Esta caja de herramientas consiste en análisis FFT y linealización.

5. **Toolbox Mecatrónica:** Esta caja de herramientas consiste en el Asistente de Cam, el Asistente de Perfil de Movimiento y el Asistente de Servo Motor.
6. **Toolbox en tiempo real:** Esta caja de herramientas le permite crear C-código fuera de cualquier modelo de 20-sim para el uso en aplicaciones de tiempo real.
7. **Toolbox del dominio del tiempo**: Esta caja de herramientas contiene herramientas poderosas para inspeccionar el comportamiento de su modelo usando la simulación de dominio de tiempo: barridos de parámetros, optimización, ajuste de curva, análisis de tolerancia, análisis de sensibilidad, análisis de Monte Carlo y análisis de variación. También puede utilizar DLL externas para ejecutar sus simulaciones de dominio de tiempo.
8. **Toolbox de secuencias de comandos:** Esta caja de herramientas contiene herramientas poderosas para inspeccionar el comportamiento de su modelo utilizando la simulación de dominio de tiempo: barridos de parámetros, optimización, ajuste de curva, análisis de tolerancia, análisis de sensibilidad, análisis de Monte Carlo y análisis de variación. También puede utilizar DLL externas para ejecutar sus simulaciones de dominio de tiempo. [15]

3.5.1 Modo de empleo de los toolbox

20-SIM cuenta con diferentes toolbox, uno de ellos es la **caja de herramientas de dominio de frecuencia**, el cual consiste en un Editor de Sistema Lineal, Análisis FFT y funcionalidad de Linealización de Modelos.

Editor de sistema lineal

El editor de sistema lineal es una herramienta especializada para el diseño y análisis de sistemas lineales. El editor admite sistemas SISO de tiempo continuo y discreto usando varias representaciones.

La interfaz gráfica le permite editar un sistema lineal en cualquier forma deseada: espacio de estado ABCD, función de transferencia o ganancia de polo cero. Al analizar la respuesta al Paso, el diagrama de Bode, el diagrama de Nyquist, el gráfico de Nichols y los gráficos de Pole-Zero le permiten evaluar rápidamente el comportamiento del sistema. Se calculan los márgenes de fase, ganancia y módulo, así como el tiempo de subida, el sobreimpulso y el valor de estado estacionario.

En la figura 3.12 se ilustra el editor de sistema lineal.

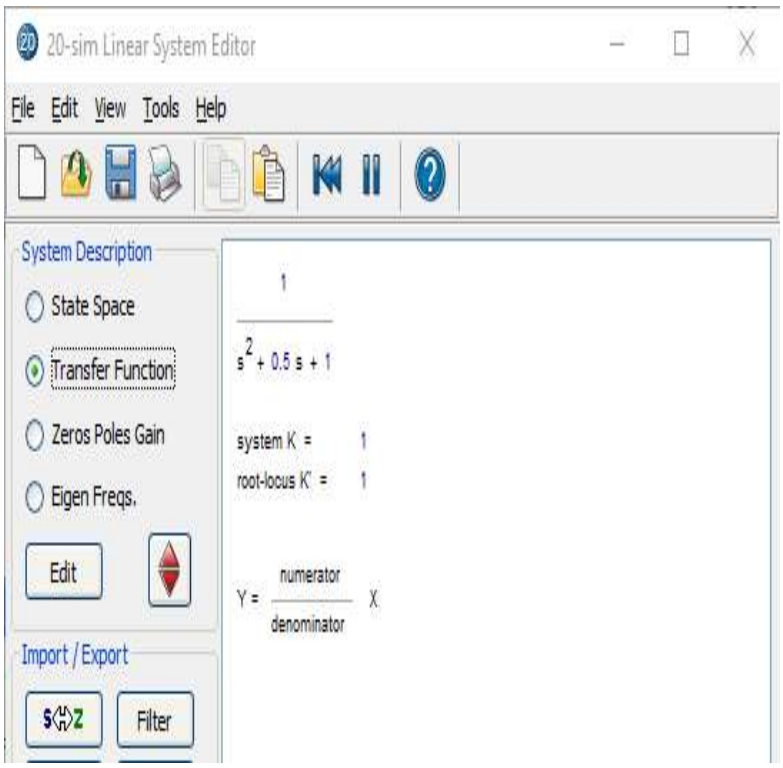


Figura 3. 12 Editor de Sistema Lineal

En la figura 3.13 se ilustra el resultado de la función con la opción de diagrama de Bode.

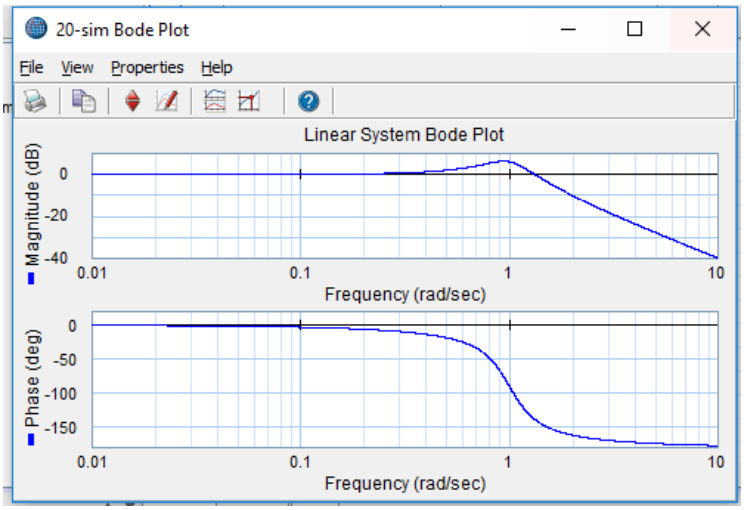


Figura 3. 13 Diagrama de Bode

En la figura 3.14 se ilustra el resultado de la función con la opción de diagrama de Nyquist.

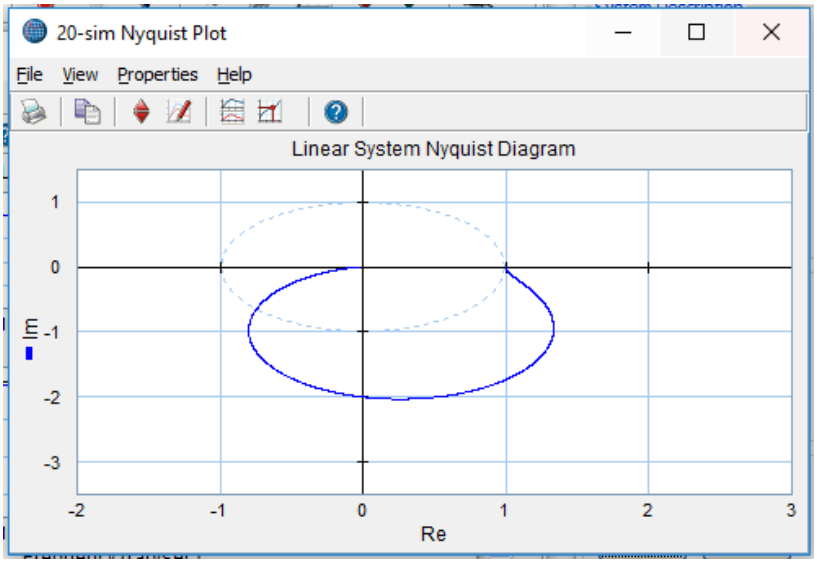


Figura 3. 14 Diagrama de Nyquist

En la figura 3.15 se ilustra el resultado de la función con la opción de diagrama de Polos y Zeros.

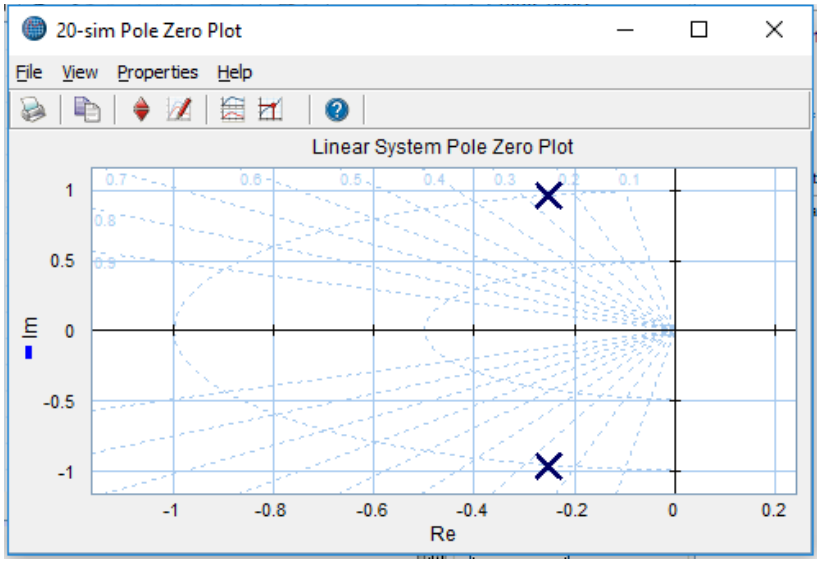


Figura 3. 15 Diagrama de Polos y Zeros de la función

En la figura 3.16 se ilustra el resultado de la función con la opción de diagrama de Nichols.

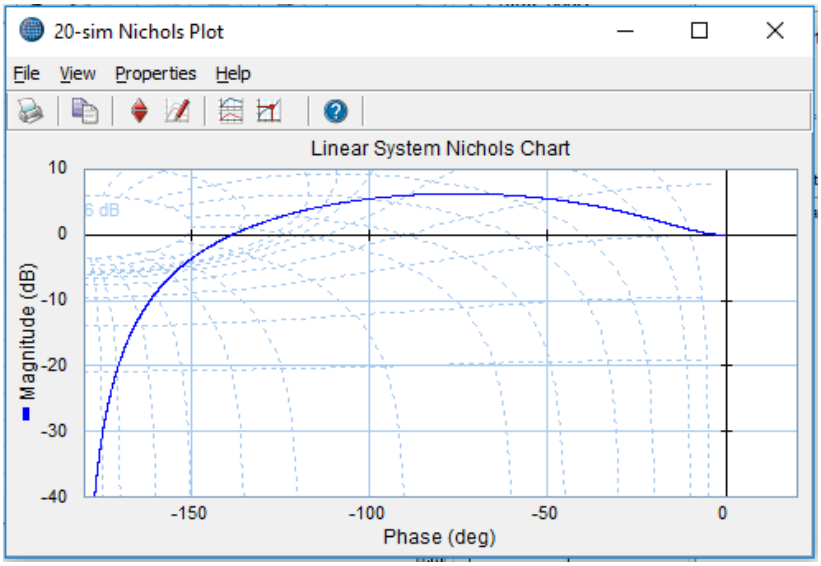


Figura 3. 16 Diagrama de Nichols

En la figura 3.17 se ilustra el resultado de la función con la opción de diagrama de respuesta al paso.

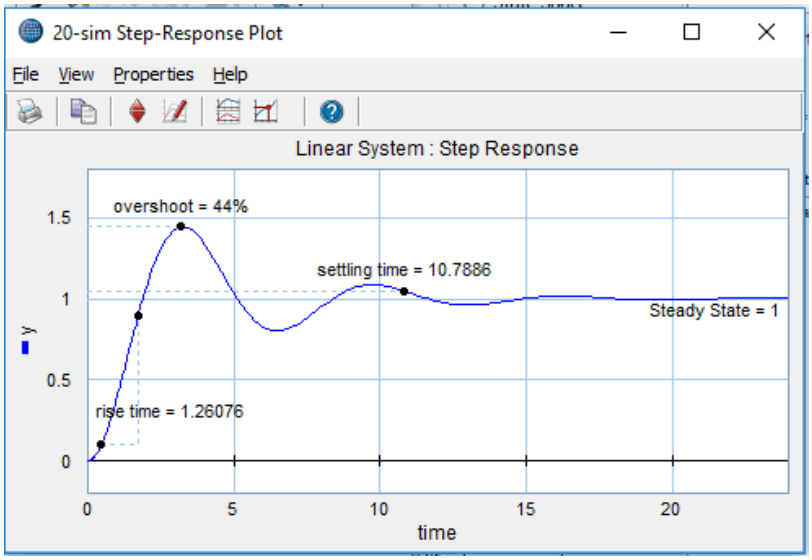


Figura 3. 17 Respuesta al paso.

3.6 MODELOS DE ECUACIONES

3.6.1 Introducción

Las ecuaciones son la base para todos los modelos en 20-sim. En el nivel más bajo de un modelo siempre encontrarás ecuaciones. Las ecuaciones se pueden introducir en el Editor de 20-sim.

1. Abra 20-sim y seleccione Archivo, Nuevo y Modelo Gráfico. La parte derecha del Editor ahora les permitirá a los modelos gráficos. Es por eso que hemos nombrado a esta parte del Editor el Editor Gráfico. El Editor Gráfico se convertirá en un Editor de 3Ecuaciones si vamos al nivel más profundo de cualquier modelo.

2. Vaya a la izquierda del Editor y haga clic en la pestaña Biblioteca Ahora aparecerá el Explorador de la Biblioteca.

2. Haga clic en Ejemplos y Sistemas Dinámicos.

3. Arrastre el modelo Lorenz Attractor al espacio en blanco a la derecha (Editor gráfico). En la figura 3.18 se ilustra el modelo de ecuaciones Attractor Lorenz.

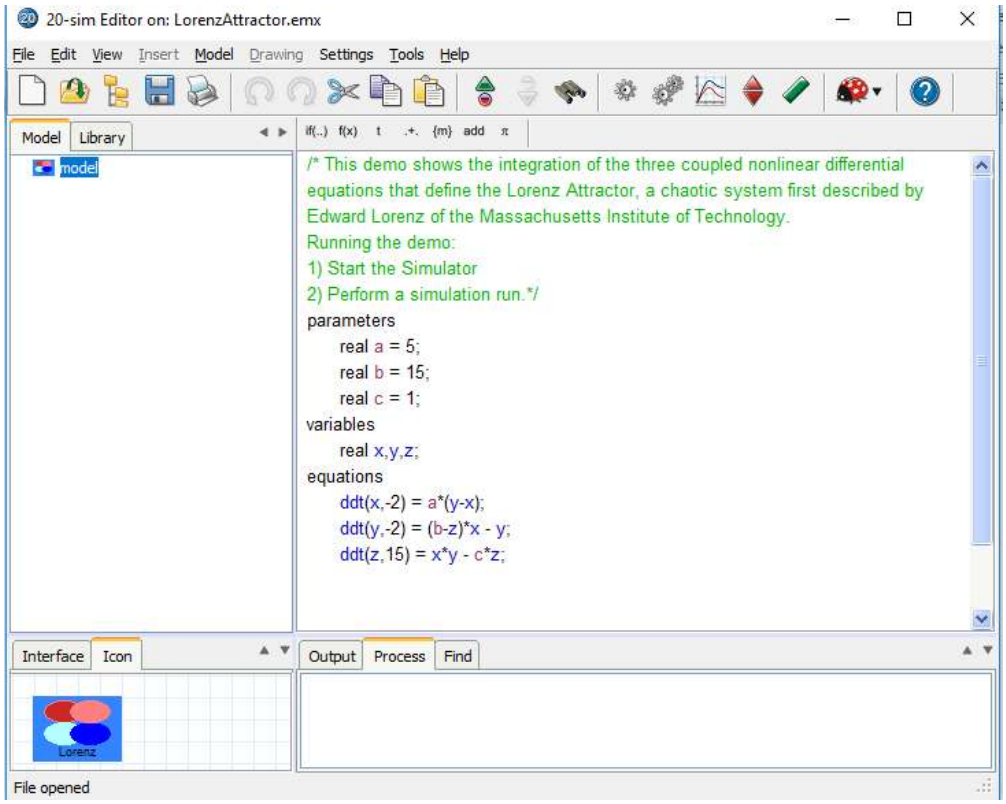


Figura 3. 18 Modelo de ecuaciones de Attractor Lorenz.

Como verá el editor gráfico cambia a un editor de ecuaciones y a un modelo de ecuaciones. Este modelo se le llama modelo principal de una ecuación, porque no tiene señales de entrada, señales de salida o puertos. Esto se puede verificar en la parte inferior izquierda del Editor que muestra la interfaz del modelo (vacía). Un modelo principal es un modelo que no se puede conectar con otros modelos.

4. Abra 20-sim y seleccione Archivo, Nuevo y Modelo Gráfico.
5. Vaya a la izquierda del Editor y haga clic en la pestaña Biblioteca. Ahora aparecerá el Explorador de bibliotecas.
6. Haga clic en Ejemplos y Diagramas de bloque.
7. Arrastre el modelo Oscillator al espacio en blanco a la derecha (Editor gráfico).

Notar qué, se abre un modelo de diagrama de bloques. Inspeccionaremos el elemento de integración de este modelo. En la figura 3.19 se ilustra el modelo de diagrama de bloques de un oscilador.

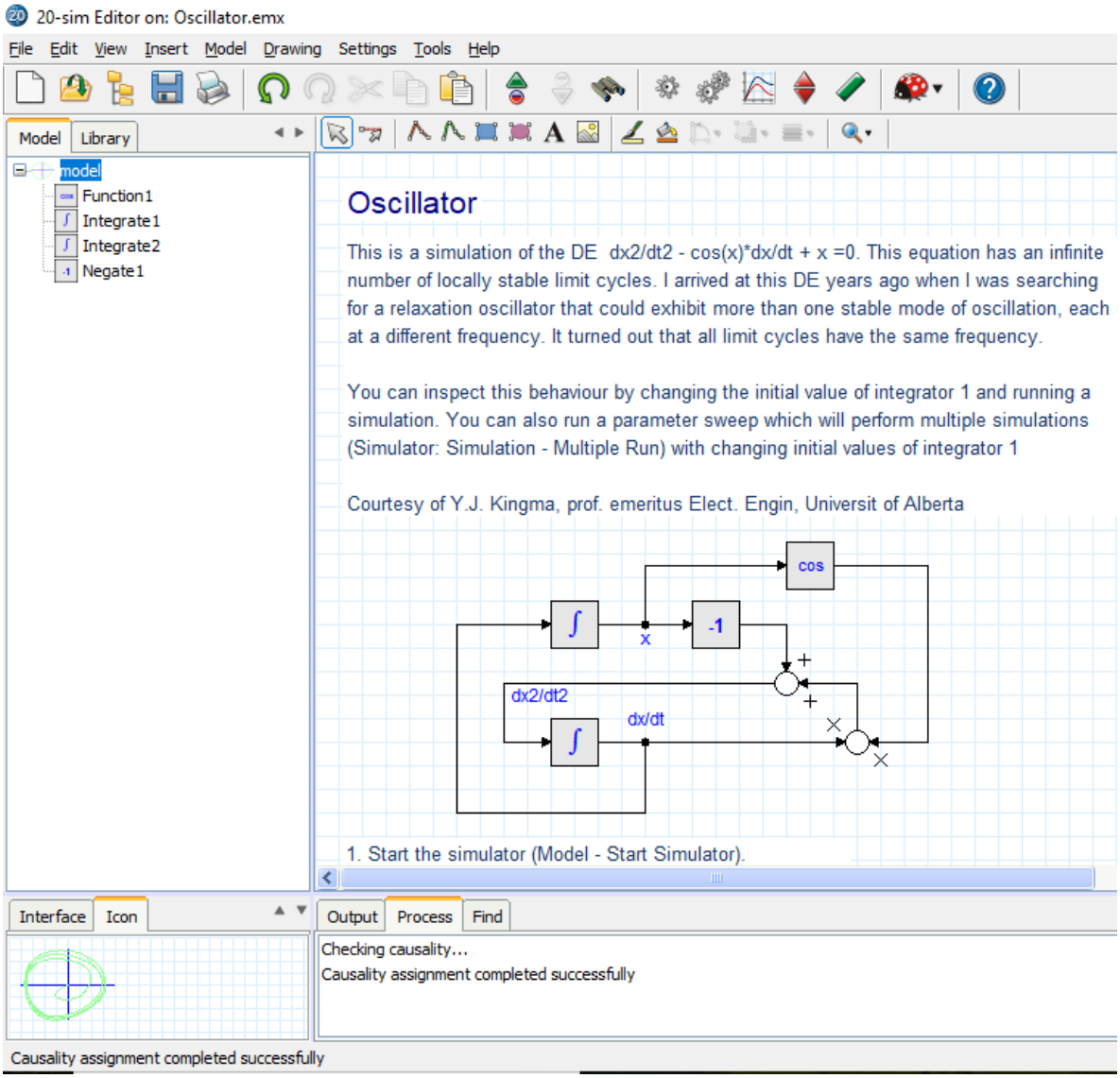


Figura 3. 19 Modelo de diagrama de bloque de un oscilador.

8. Vaya a la izquierda del Editor y haga clic en la pestaña Modelo.

Ahora aparecerá el navegador de modelos. El Navegador de Modelos muestra los elementos relevantes de este modelo.

9. Seleccione el elemento Integrate1.

En la figura 3.20 se ilustra la implementación de una ecuación de un elemento de integración.

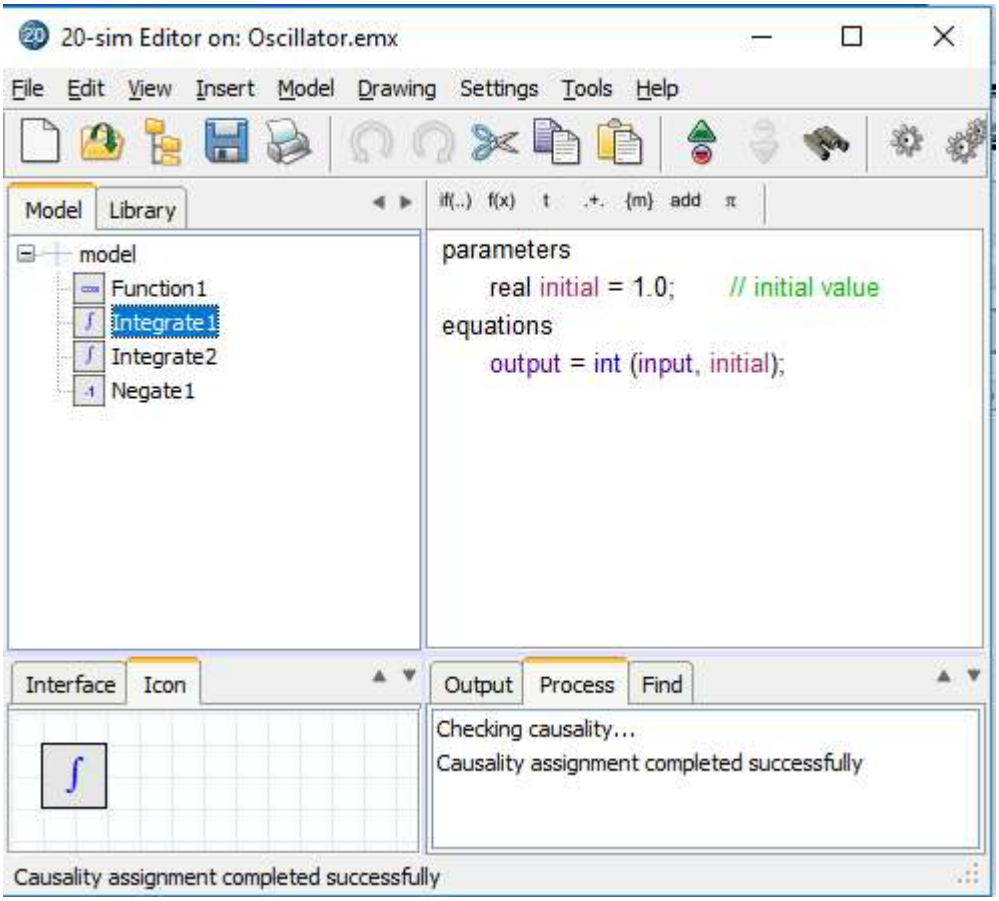


Figura 3. 20 Implementación de una ecuación de un elemento de integración.

CAPITULO 4. SIMULACIÓN DE SISTEMAS EN TRES DIMENSIONES

4.1 Toolbox de Mecánica 3D

El toolbox de Mecánica 3D que contiene 20-sim ofrece las herramientas que facilitan el modelado dinámico en 3D.

a) Cuerpos

Puede crear modelos 3D arrastrando cuerpos en un espacio de trabajo 3D. Las representaciones de cada cuerpo se pueden cambiar a una esfera, un bloque, un cilindro etc. Además, los colores se pueden cambiar y las descripciones se pueden agregar. El tamaño y la forma de un cuerpo son meramente representación, un cuerpo se caracteriza por sus coeficientes de inercia y masa. En la figura 4.1 se ilustra el editor del toolbox de Mecánica 3D en 20-sim.

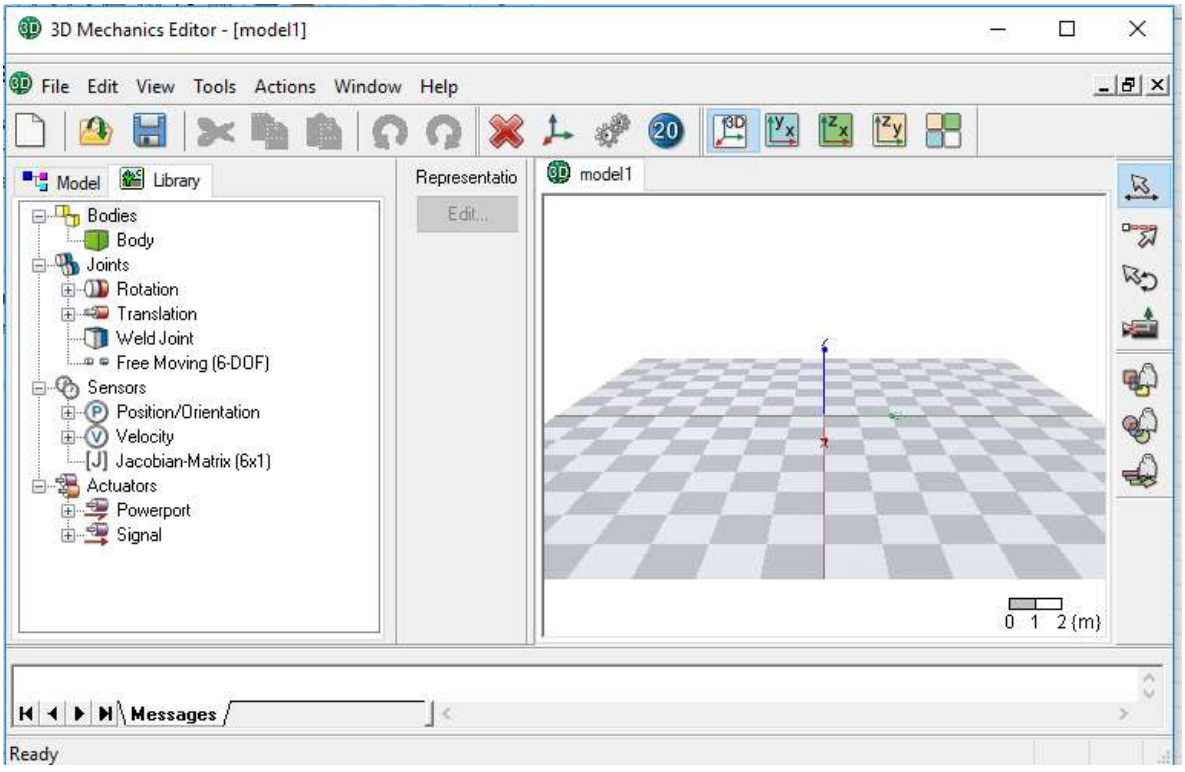


Figura 4. 1. Editor de Mecánica 3D de 20-SIM

b) Articulaciones

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

c) Interfaz

La interfaz de usuario tiene 4 modos diferentes en los que puede seleccionar, conectar, traducir y girar los cuerpos y las articulaciones. Se hace mucho esfuerzo para mantener la interfaz gráfica de usuario tan natural como sea posible. Se admiten varias vistas. Además del entorno 3D, se pueden ver intersecciones 2D en xy, xz y plano yz.

d) Modelos

El Editor de Mecánica 3D puede generar un modelo de 20-sim desde su modelo 3D. Este modelo de 20-sim comprende toda dinámica y cinemática del modelo. Las fuerzas se pueden aplicar directamente a las articulaciones de cada cuerpo. También puede acoplar resortes y amortiguadores de la biblioteca de mecánica en 20-sim, a las articulaciones, porque todo el modelo está basado en puertos.

La gravedad se puede establecer como una fuerza externa. Finalmente, el Editor de Animación 3D puede mostrar la respuesta dinámica del modelo completo. [16]

En esta sección se utilizará el toolbox de mecánica 3D para crear un modelo. En este editor se visualizará los posibles movimientos, desde el editor se generará un modelo de 20-sim en el cual se incorporarán las ecuaciones de movimiento y una escena de 20-sim para la animación del modelo.

Este modelo que simulara tendrá articulaciones accionadas, esto quiere decir que se aplicara un torque a las uniones.

e) Inserción de componentes

Abra 20-sim y seleccione Archivo, Nuevo y Modelo Gráfico.

En el Editor del menú Herramientas, seleccione 3D Mecánica Toolbox y Editor de Mecánica 3D

Editor de Mecánica.

Se insertará un modelo de mecánica 3D en el Editor y se abrirá el Editor de Mecánica 3D:

Al igual que el Editor de 20-sim, el Editor de Mecánica 3D consta de varias partes: [16]

a) Sección del modelo / de la biblioteca: Ésta es la parte en la mitad izquierda. La pestaña Modelo abre el Navegador de modelos que muestra una composición jerárquica (todos los elementos) del modelo que se crea en el Editor de mecánica 3D. La pestaña Biblioteca abre el explorador de biblioteca que muestra todos los objetos que se pueden utilizar.

b) Propiedades del objeto: Esta es la parte de la derecha que en la imagen de arriba muestra "Sin selección". Tan pronto como se selecciona un objeto, mostrará el nombre de ese objeto y le permitirá cambiar las propiedades del objeto.

c) Representación del objeto: Esta es la parte de la derecha que en la imagen de arriba muestra "Ninguno". Tan pronto como se selecciona un objeto, mostrará el nombre de ese objeto y le permitirá cambiar su representación.

d) Editor gráfico: Este es el espacio con la tablilla a la derecha. En este editor puede arrastrar objetos y crear modelos.

e) Ficha Mensajes: Esta es la parte en la parte inferior derecha, donde se muestran todos los mensajes.

El editor tiene varios modos de operación que se indican con los botones a la derecha del editor:

Tabla 4- 1 Descripción de los botones de Mecánica 3D

Botón	*Modo*	*Descripción*
	Modo de traslación	Este botón se utiliza para mover de lugar a los objetos.
	Modo de rotación	Este botón se utiliza para rotar los objetos.
	Modo de conexión	Este botón conecta los diferentes objetos
	Movimiento de la cámara	Este botón se utiliza para ver las diferentes vistas de la cámara

Puede elegir que los objetos sean transparentes:

Tabla 4- 2 Descripción de los botones en modo transparente.

Botón	*Modo*	*Descripción*
	Modo fantasma para los cuerpos.	Este botón se utiliza para hacer transparentes los cuerpos.
	Modo fantasma para las articulaciones.	Este botón se utiliza para hacer transparentes las articulaciones.
	Modo fantasma para los sensores y actuadores.	Este botón se utiliza para hacer transparente a los sensores y actuadores.

El Editor siempre está en modo de traslación, que es adecuado para arrastrar y soltar componentes. [16]

4.2 Caso de Estudio 1 “Robot Scara”

En esta sección se documentará la creación y simulación del modelo dinámico del robot Scara.

Comenzaremos por revisar el robot, identificaremos los enlaces, los parámetros de las masas e inercia. Luego, presentaremos el modelo de 20-sim que se usará para simular el robot Scara. Este modelo contiene generadores de punto de ajuste, controladores y unidades que se conectarán al robot. Continuaremos ingresando al robot Scara en el Editor de Mecánica 3D. Desde el Editor 3D Mecánica generaremos un submodelo de 20-sim que incorpora las ecuaciones de movimiento del robot Scara. Desde el Editor 3D Mecánica también puedes generar un escenario que describe la animación del robot Scara. Finalmente, el modelo de robot Scara se conectará a las unidades y se simulará.

a) Revisión

Comenzamos la parte de modelado revisando el robot Scara. El robot cuenta con una base con brazos que pueden moverse en el plano horizontal y una carga que puede moverse verticalmente. Por lo tanto,

el robot se puede modelar utilizando cuatro cuerpos que están conectados por articulaciones. Entre la base y el primer brazo, se conecta una articulación giratoria, que se puede activar. Entre el primer y el segundo brazo se conecta una articulación similar.

Entre la carga y el segundo brazo se conecta una articulación de traslación, que también se activa. En la figura 4.2 se ilustra las conexiones del robot scara.

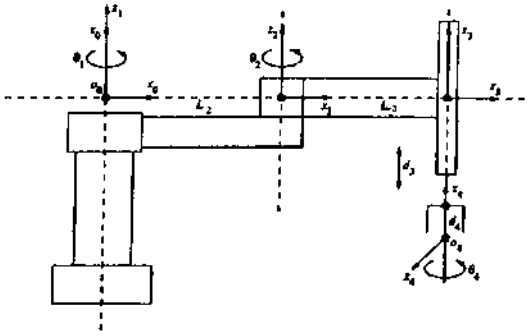


Figura 4. 2 Conexiones del robot scara.

El modelo de robot que vamos a crear en el Editor 3D Mecánica tendrá, por lo tanto, tres puertos de potencia. Dos puertos de alimentación para las articulaciones de rotación y uno para la articulación de traslación. Para simular el modelo de robot, tenemos que conectar actuadores a estas uniones. En la figura 4.3 se ilustran los parámetros geométricos utilizados para modelar el robot Scara.

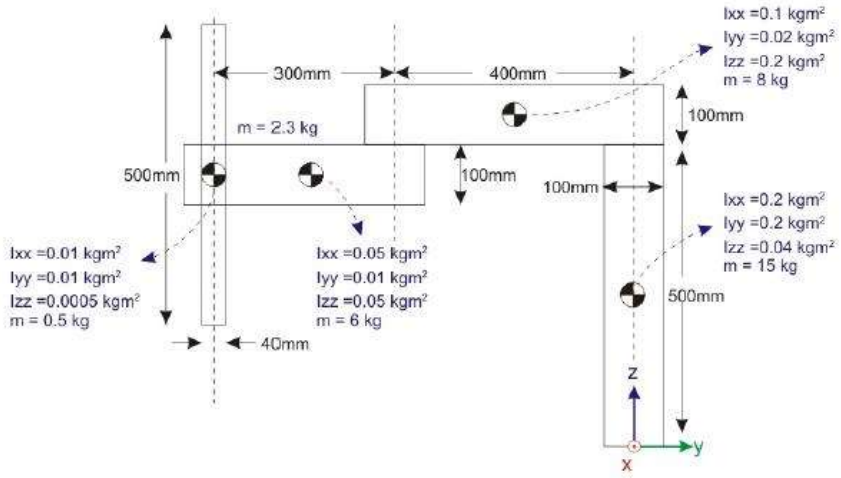


Figura 4. 3 Parámetros geométricos del robot scara

Los parámetros geométricos, las masas y la inercia se pueden observar en el de diseño del robot Scara, como se observa en la figura anterior.

El modelo completo que se utilizará para simular el robot Scara se muestra en la figura 4.4. El recuadro gris con el nombre robot se genera con el editor de 3D Mecánica este representa la estructura mecánica del robot. Las articulaciones de rotación son por motores eléctricos accionados con cajas de engranajes. La articulación de traslación es impulsada por un motor eléctrico con un husillo. Los motores están controlados por controladores PD, que comparan el movimiento deseado con la posición real del robot.

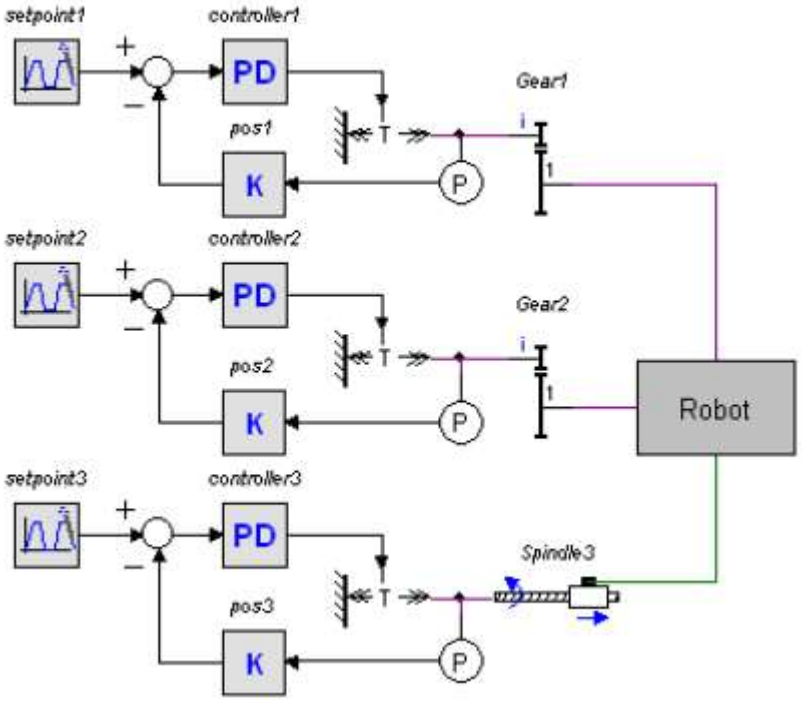


Figura 4. 4 Modelo de 20-sim para accionar los movimientos del robot, el recuadro gris(Robot) es el que contiene el grafico realizado en el editor de Mecánica 3D.

Las dimensiones del robot Scara son bastante pequeñas. Por lo tanto, se cambiará la escala del editor.

Primero, se cambia el tamaño de los cuadros de referencia a 0.1 y el tamaño de cuadrícula a 0.1 como se muestra en la figura 4.5:

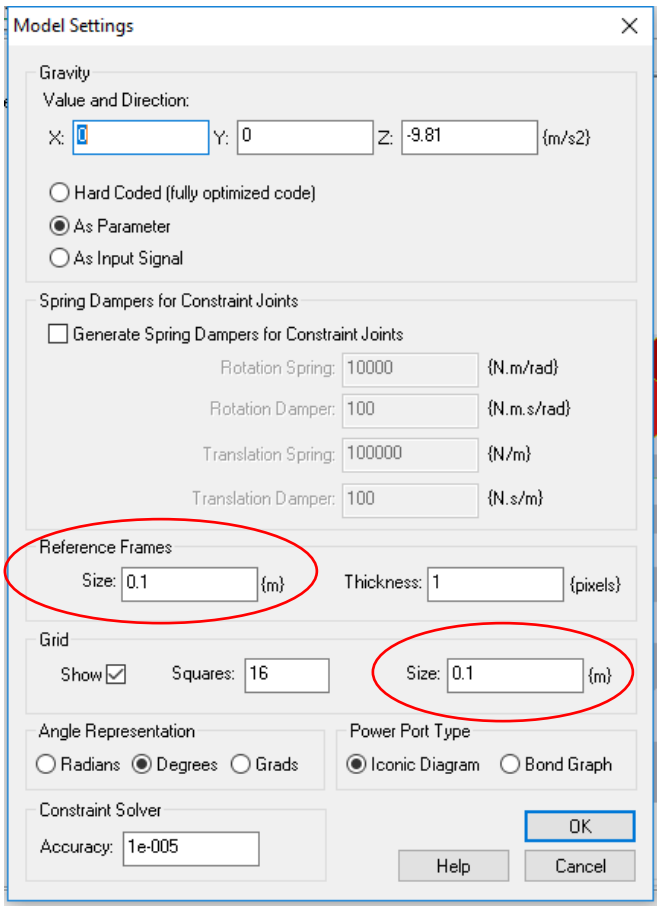


Figura 4. 5 Cambio de tamaño de las cuadrículas y los cuadros de referencia

Damos clic en OK para cerrar el cuadro de diálogo configuración del modelo. Ya que al cambiarse la dimensión de la cuadricula el marco de referencia está demasiado lejos, podemos mover la cámara más cerca del origen, cambiando al modo cámara (indicado por ), presione la tecla Ctrl y arrastramos el mouse hasta obtener una mejor vista.

En la figura 4.6 se ilustra el resultado de nuestro plano, después de dimensionarlo.

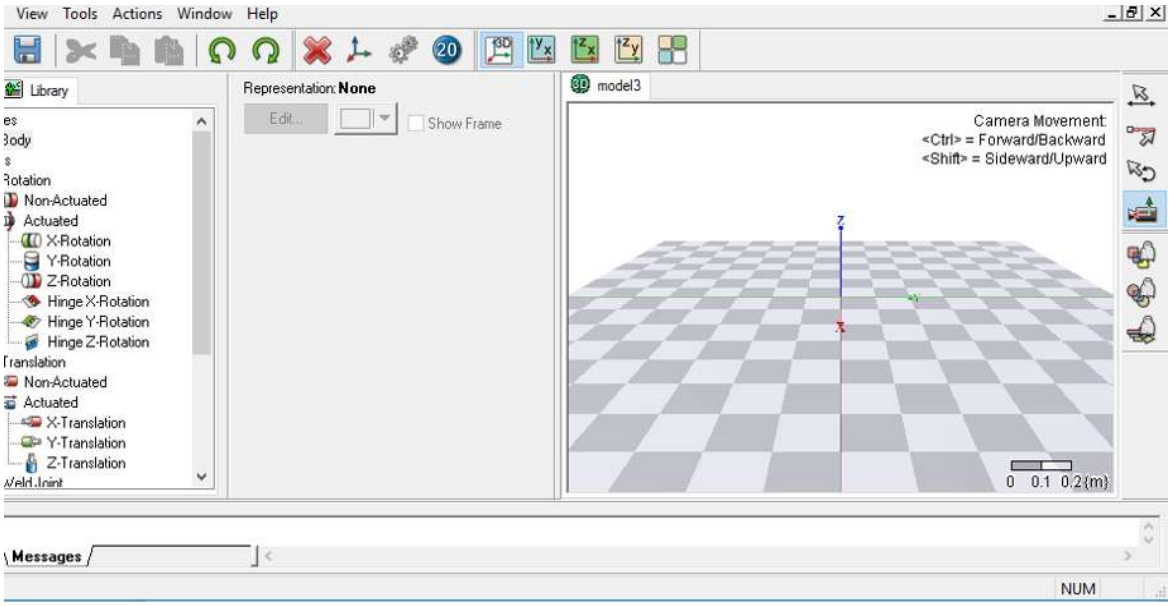


Figura 4. 6 Uso de la cámara para acercar más al origen.

Volvemos al modo de traslación (indicado por ).

Se insertarán los componentes necesarios para construir el robo.

Insertamos los siguientes componentes en el Editor gráfico y cambiaremos el nombre de acuerdo a la tabla 4- 3.

Tabla 4- 3 Se muestra los elementos que vamos a utilizar y el nombre con el cual los renombraremos.

Acción	*Componente*	*Nombre*
1	Bodies\Body	Base
2	Bodies\Body	Brazo1
3	Bodies\Body	Brazo2
4	Bodies\Body	Carga
5	Joints\Rotation\Actuated\Z-rotation	Base_Brazo1
6	Joints\Rotation\Actuated\Z-rotation	Brazo1_Brazo2
7	Joints\Translation\Actuated\Z-translation	Brazo2_Carga

En la figura 4.7 se ilustra el resultado obtenido al insertar los componentes el Editor de Mecánica 3D el cual se verá así:

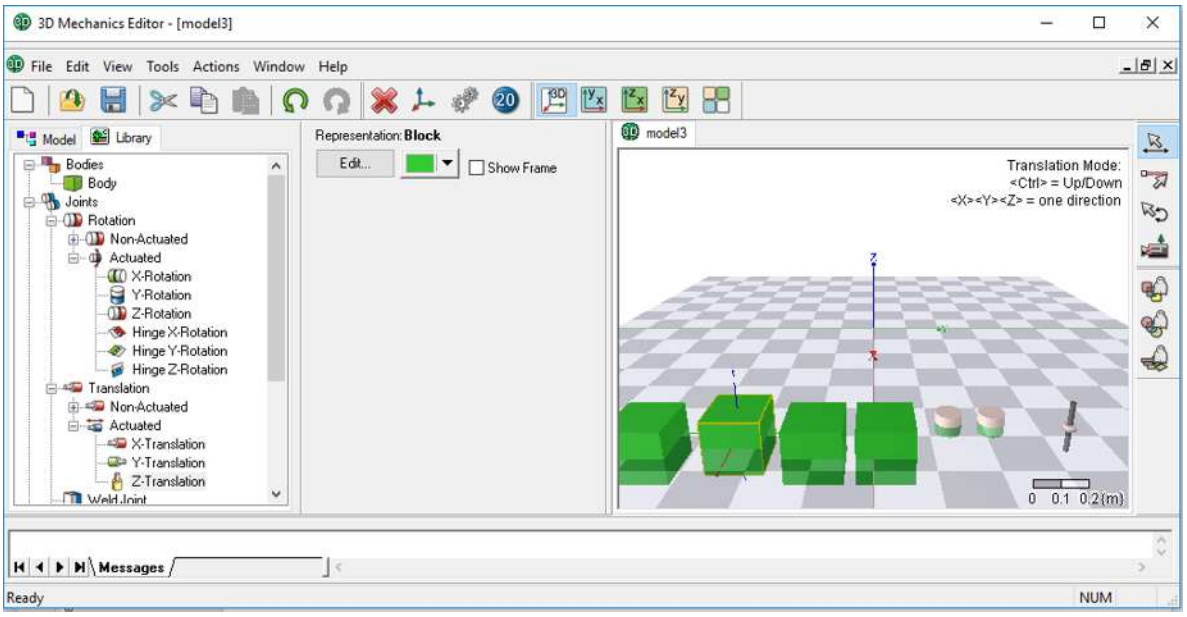


Figura 4. 7 El robot Scara necesita de 4 cuerpos y 3 articulaciones

Para hacer que el modelo se observe como un robot real, se tiene que cambiar la representación de los cuerpos. Se utilizarán los parámetros geométricos del robot que anteriormente se mencionaron.

Seleccione el cuerpo Base. El botón Editar (a la izquierda del Editor) ahora debe estar activo.

Hacer clic en el botón Editar para que se abra el cuadro de diálogo Representación 3D.

En la figura 4.8 se ilustra el cuadro de dialogo donde se harán las modificaciones a la Base.

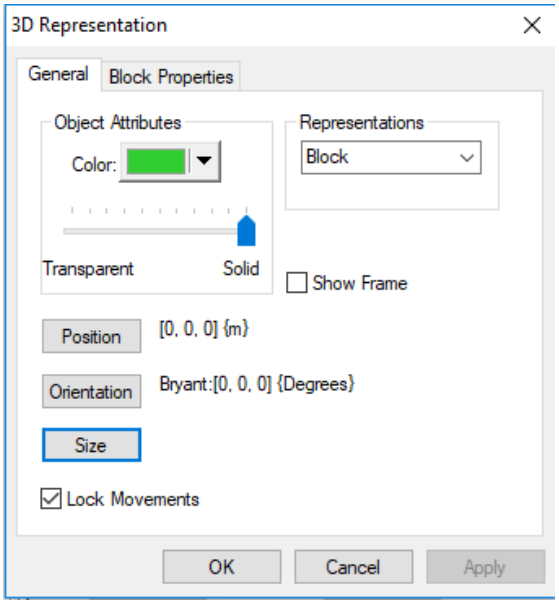


Figura 4. 8 En el cuadro de diálogo de Representación 3D se puede cambiar el aspecto de un objeto.

Utilizando el diálogo de representación en 3D, se cambiará el aspecto del objeto.

Cambiamos la representación del cuerpo base de acuerdo con las especificaciones de la tabla.

Tabla 4- 4 Ajuste de la representación de los cuerpos.

Acción	*Body*	*Tamaño*	*Color*	*Representación*
1	Base	x = 0.1, y = 0.1, z = 0.5	Verde	Bloque
2	Brazo1	x = 0.1, y = 0.5, z = 0.1	Azul	Bloque
3	Brazo2	x = 0.1, y = 0.4, z = 0.1	Rojo	Bloque
4	Carga	x = 0.04, y = 0.04, z = 0.5	Naranja	Cilindro

Realizamos la misma acción con los otros cuerpos de acuerdo con la tabla (acciones 2, 3 y 4).

Hacemos doble clic en el cuerpo base que abre el cuadro de diálogo Propiedades del cuerpo.

Aseguramos de que este cuerpo tenga la opción “Is fixed world” activada.

Se establece la posición en x = 0, y = 0 y, z = 0.25 y se cierra el cuadro de diálogo Propiedades del cuerpo.

En las figuras 4.9 y 4.10 se ilustra cómo se realizan los ajustes necesarios para establecer los parámetros.

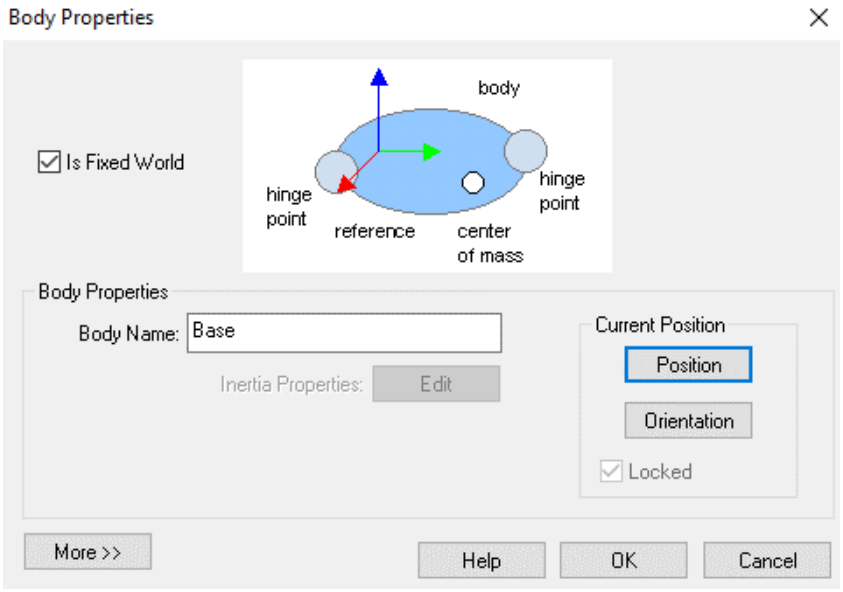


Figura 4. 9 Se seleccionó la opción "Is Fixed World", y se cambió el nombre del elemento a Base.

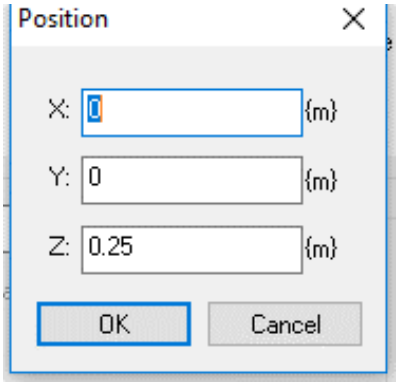


Figura 4. 10 Se cambiaron los parámetros de la posición para el elemento Base.

En la figura 4.11 se ilustra el resultado obtenidos después de cambiar los parámetros para poder llevar a cabo nuestro robot.

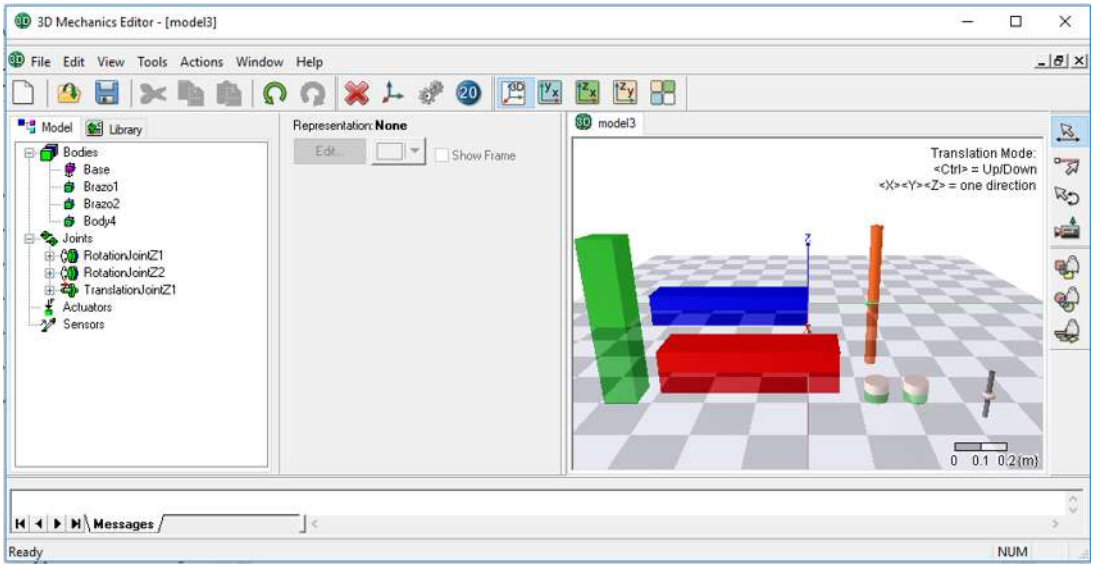


Figura 4. 11 Resultado de los ajustes de parámetros.

Los elementos aún tienen parámetros de masa e inercia predeterminados, se cambiarán de acuerdo a las especificaciones dadas previamente.

Hacemos doble clic en el cuerpo del brazo1 que despliega el cuadro de diálogo Propiedades del cuerpo.

Hacemos clic en el botón Editar las Propiedades de inercia y cambie los parámetros del cuerpo del brazo1 de acuerdo con las siguientes especificaciones de la tabla.

Acción	*Body*	*Masa [kg]*	*Ixx [kgm2]*	*Iyy [kgm2]*	*Izz [kgm2]*
1	Brazo1	8	0.1	0.02	0.1
2	Brazo2	6	0.05	0.01	0.05
3	Carga	0.5	0.01	0.01	0.0005

El cuerpo base está fijo al suelo, lo que hace que los parámetros de inercia sean irrelevantes, si se deseara insertar de todas maneras, desactivar la opción “Is fixed world”.

En la figura 4.12 se ilustra el cambio de parámetros de la masa e inercia del elemento Brazo1.

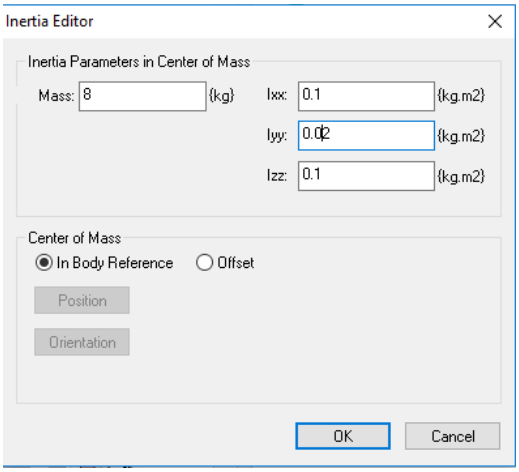


Figura 4. 12 Cambio de los parámetros masa e inercia para el Brazo1.

Realizar lo mismo con los otros cuerpos de acuerdo con las especificaciones.

Para ensamblar el robot se crearán conexiones entre los diferentes cuerpos y articulaciones.

Hacemos clic en  para configurar el Editor en modo de conexión.

Realizando esa acción se pueden definir las conexiones haciendo clic desde las articulaciones a los cuerpos. Para hacer las articulaciones visibles, usaremos el Modo fantasma para los cuerpos.

Hacemos clic en  para configurar el editor en modo fantasma para cuerpos.

Realizamos las conexiones de acuerdo con las especificaciones de la tabla 4-5:

Tabla 4- 5 Pasos para realizar las conexiones de manera correcta.

Acción	*Primer click*	*Segundo click*	*Conexión*	*Posición*
1	Base_Brazo1	Base	ConnectionPoint1	x = 0, y = 0, z = 0.25
2	Base_Brazo1	Brazo1	ConnectionPoint2	x = 0, y = 0.2, z = -0.05
3	Brazo1_Brazo2	Brazo1	ConnectionPoint1	x = 0, y = -0.2, z = -0.05
4	Brazo1_Brazo2	Brazo2	ConnectionPoint2	x = 0, y = 0.15, z = 0.05
5	Brazo2_Carga	Brazo2	ConnectionPoint1	x = 0, y = -0.15, z = 0
6	Brazo2_Carga	Carga	ConnectionPoint2	x = 0, y = 0, z = 0

En la figura 4.13 se ilustra con que puerto vamos a conectar la base a la articulación.

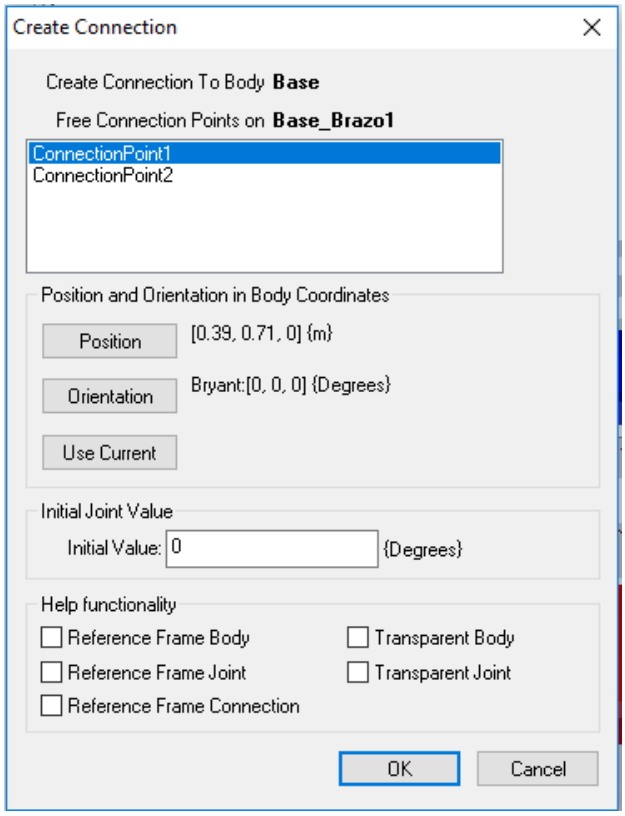


Figura 4. 13 Conexión entre la base y la articulación.

En la figura 4.14 se ilustra el ajuste de la posición en el cual queremos que se muestre la base con dicha articulación.

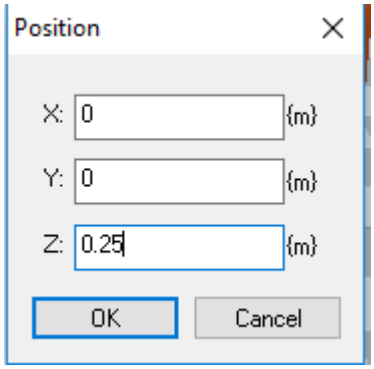


Figura 4. 14 Ajuste de la posición de la base con la articulación.

Cuando todos los cuerpos y articulaciones están conectados, 20-sim comenzará a armar el robot. En la figura 4.15 se ilustra el resultado de conectar los cuerpos con las articulaciones.

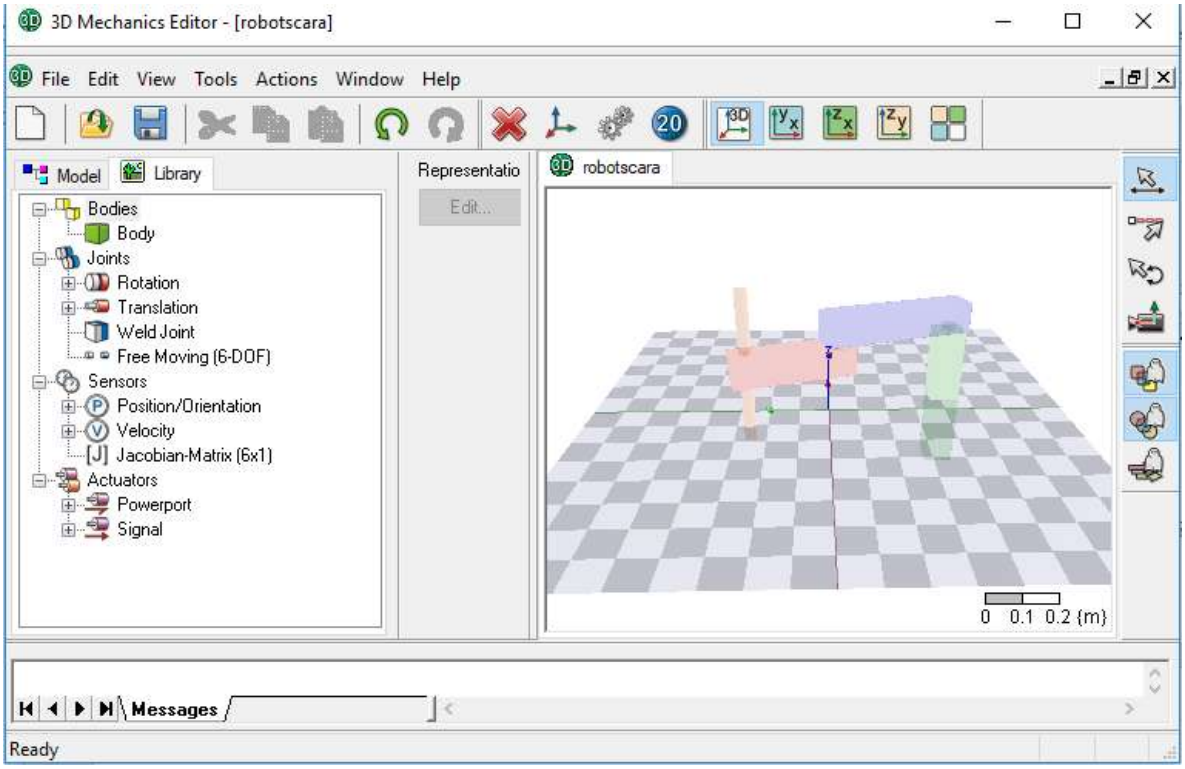


Figura 4. 15 Armado del robot, después de conectar los cuerpos con las articulaciones.

Comprobación de movimiento

Para ver si el modelo se definió correctamente, podemos verificarlo y ver cómo se puede mover.

Hacemos clic en  para configurar el Editor en modo de traslación.

Hacemos clic en  para anular la selección del modo fantasma.

Desde el menú **Acciones**, seleccionamos **Verificar modelo**.

En la figura 4.16 se ilustra la verificación del modelo la cual debe mostrar el mensaje "Análisis completado con éxito". Comprobación de los posibles movimientos del mecanismo.

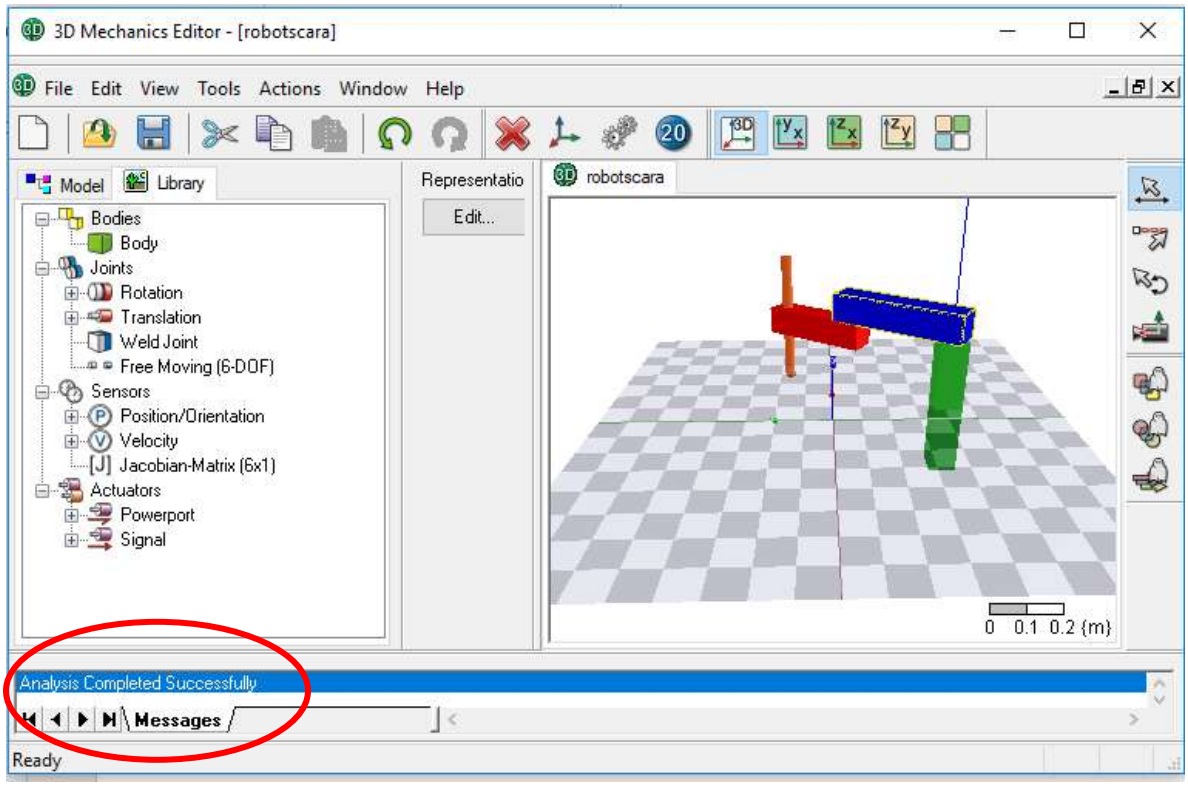


Figura 4. 16 Verificamos el modelo y corroboramos que el mensaje saliente diga "Analysis Completed Successfully", como se ilustra en la figura de arriba (marcada con un recuadro rojo).

Coloque el puntero del mouse en la parte superior del cuerpo del brazo1 y haga clic con el botón izquierdo del mouse.

La unión base_arm1 mostrará una flecha que indica rotación positiva.

Mantenemos presionado el botón izquierdo del mouse y muévalo hacia arriba y hacia abajo.

En la figura 4.17 se ilustra como el cuerpo ahora gira alrededor de la articulación.

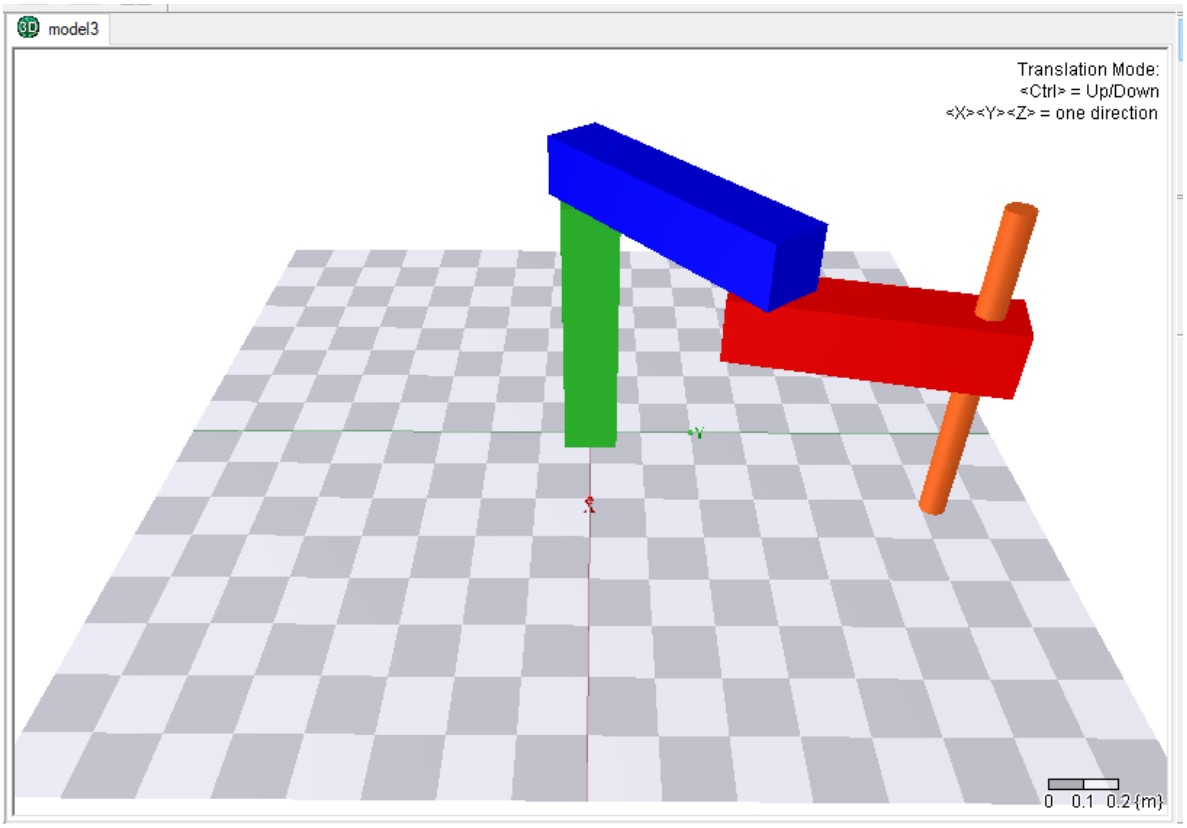


Figura 4. 17 Si el modelo está ensamblado correctamente, se observa el movimiento arrastrando el puntero del mouse.

Repetimos el movimiento con el brazo2 y los cuerpos de carga para ver sus movimientos.

Generación del modelo a 20-sim

En el menú Acciones, seleccione Generar modelo 20-sim. Aparecerá una ventana que le pedirá que guarde el modelo. Guarde el modelo en el mismo directorio temporal que el modelo 20-sim (por ejemplo, C:\Users\Irlanda\Documents\IRLANDA_DOCS\TESIS\RobotScara1.emx) usando el nombre RobotScara.emx.

En la figura 4.18 se ilustra el cuadro de dialogo para generar el código a 20-sim.

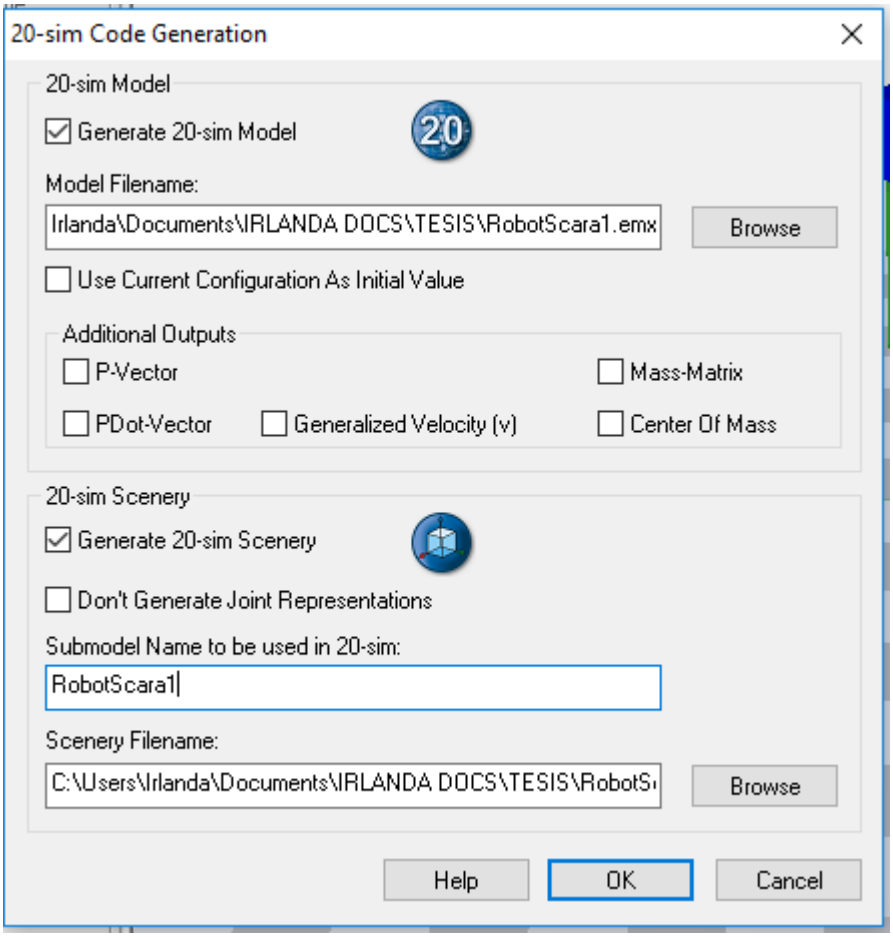


Figura 4. 18 Especificamos donde queremos guardar el código y la escena de este modelo.

Nombre de archivo de salida: esta es la ubicación del modelo 20-sim exportado. Guarde el modelo en un directorio (por ejemplo, C:\Users\Irlanda\Documents\IRLANDA_DOCS\TESIS\) utilizando el nombre RobotScara1.emx.

Nombre del submodelo que se utilizará en 20-sim: Para crear una animación correcta, el nombre que se usará en 20-sim para él, se debe conocer de antemano. Ingresamos el nombre RobotScara1.

Nombre de la escena: este es el archivo que se puede cargar en el editor de animación 3D. Guarde el escenario en un directorio (por ejemplo, C:\Users\Irlanda\Documents\IRLANDA_DOCS\TESIS\) usando el nombre RobotScara1.scn.

Haga clic en OK para exportar el modelo a 20-sim.

Simulación

Regresamos a 20-sim donde insertaremos el modelo de robot en el Editor.

Abrir 20-sim y cargar el modelo ScaraRobot desde el directorio donde se almaceno.

Ahora insertaremos el modelo de Robot Scara que se ha creado en el toolbox de Mecánica 3D.

Abrir el Explorador de archivos (por ejemplo, el Explorador de Windows) y buscar el archivo RobotScara1.emx que tenemos almacenado previamente.

Arrastramos y soltamos en el Editor 20-sim.

Seleccionamos el modelo insertado y cambie el nombre local (Modelo - Atributos) a Robot.

En la figura 4.19 se ilustra cómo se deberá ver su editor, después de agregar nuestro modelo del robot.

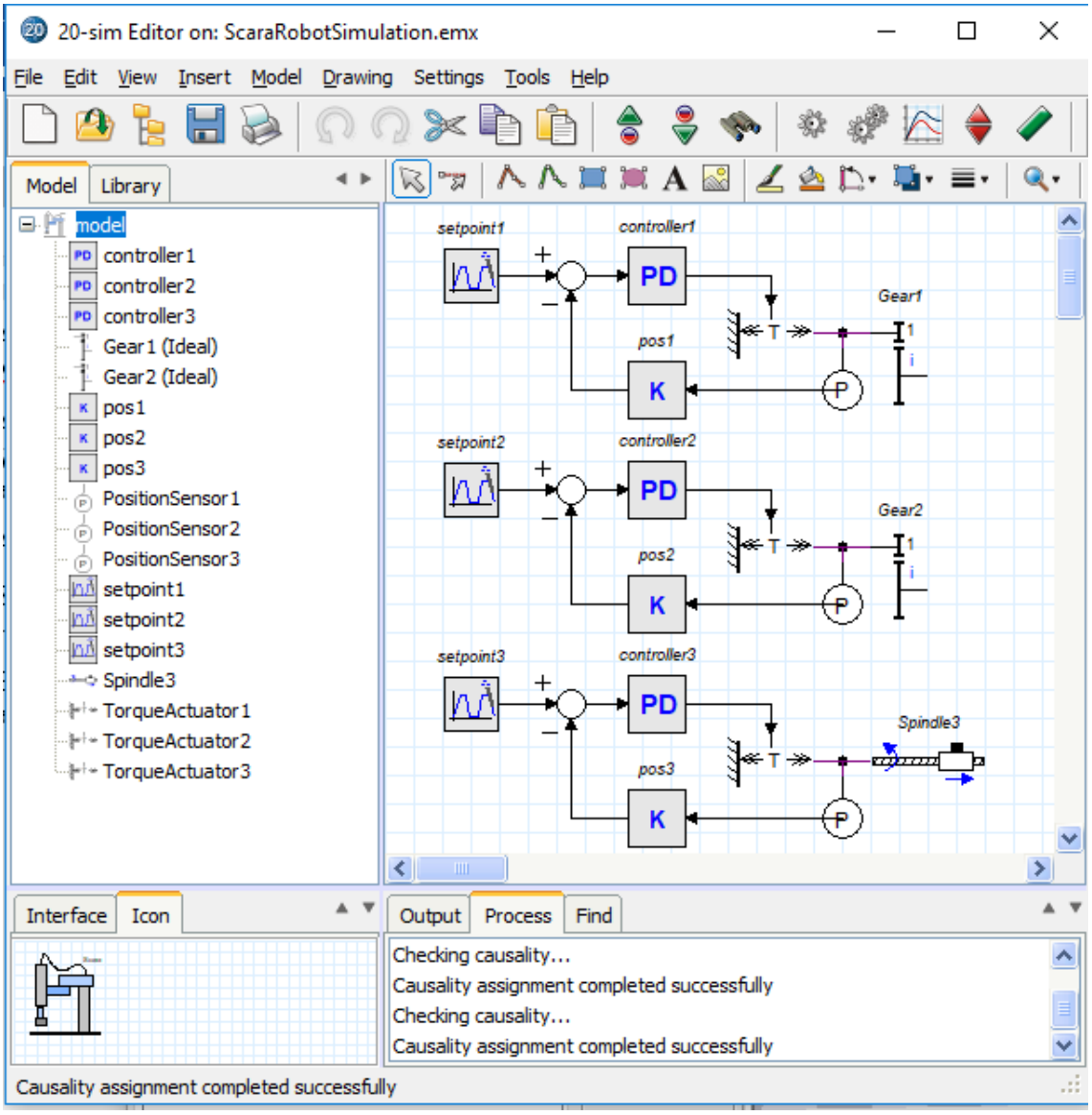


Figura 4. 19 El modelo 20-sim del sistema de accionamiento y el modelo de robot 3D (gris).

Si se observa el modelo Robot, se puede ver que tiene tres puertos. Uno para la articulación Base_Brazo1, uno para la articulación Brazo1_Brazo2 y uno para la articulación Brazo2_Carga. Conectaremos estas uniones a los actuadores.

En la figura 4.20 se ilustra la conexión de las uniones a los actuadores, para accionar los movimientos de nuestro robot.

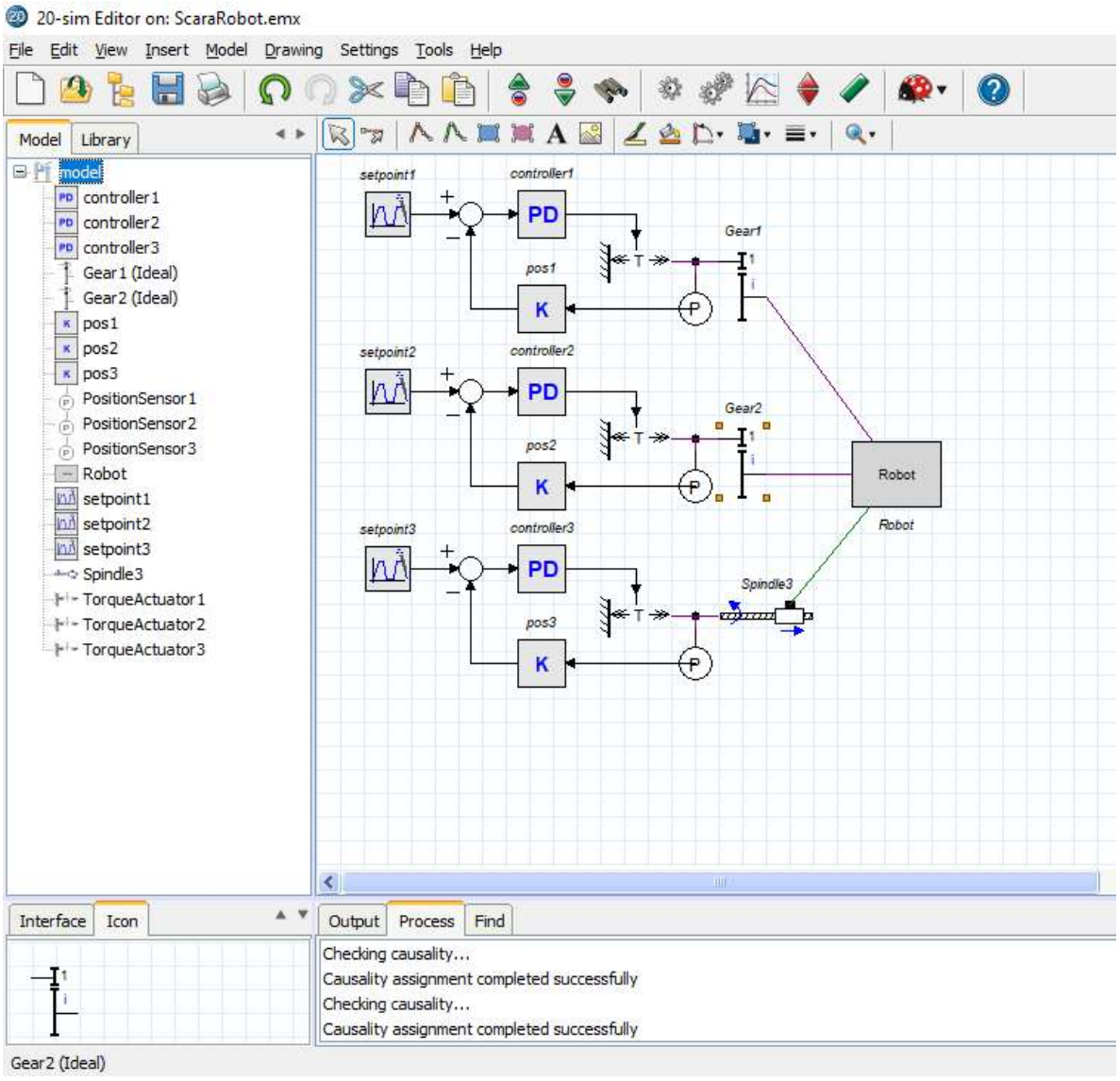


Figura 4. 20 Conexión de las uniones a los actuadores.

En el menú Archivo, hacemos clic en Guardar como, guardamos el modelo en un directorio,

por ejemplo, C:\Users\Irlanda\Documents\IRLANDA_DOCS\TESIS\ utilizando el nombre RobotScara1.emx.

En la opción **Modelo**, hacer clic en Iniciar simulador, esto abrirá el simulador, como se puede observar en la figura 4.21, se carga un experimento predefinido que muestra una gráfica con variables de punto de ajuste y posición. Añadiremos una animación 3D a este experimento.

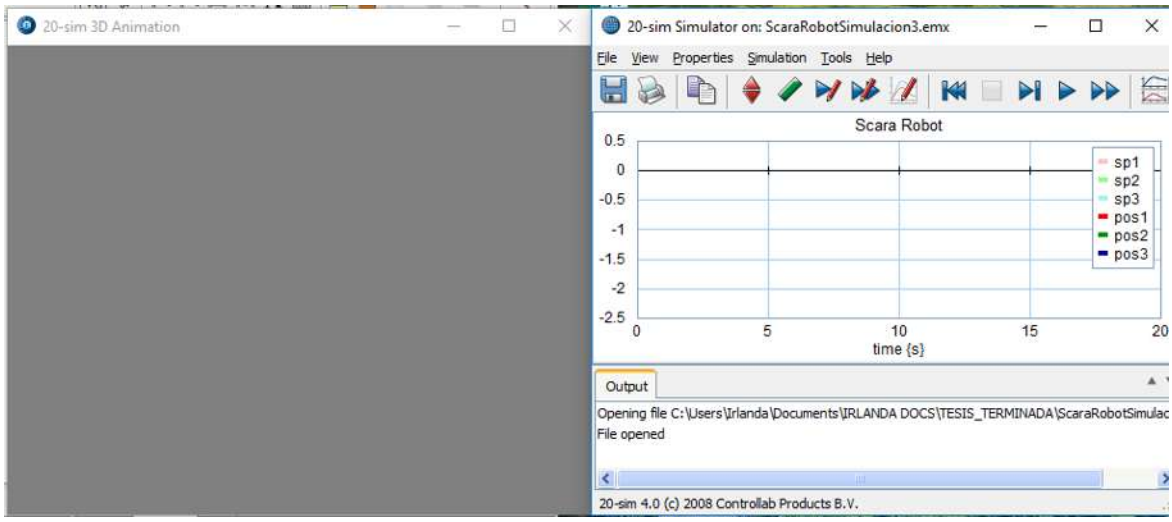


Figura 4. 21 Visualización del Simulador con la carga predefinida.

En el simulador, buscamos la opción **"Tools"**, y seleccionamos el toolbox de animación y seleccionamos la opción animación 3D.

En la ventana de Animación 3D, hacemos doble clic para abrir la ventana Propiedades de Animación 3D como se muestra en la figura 4.22.

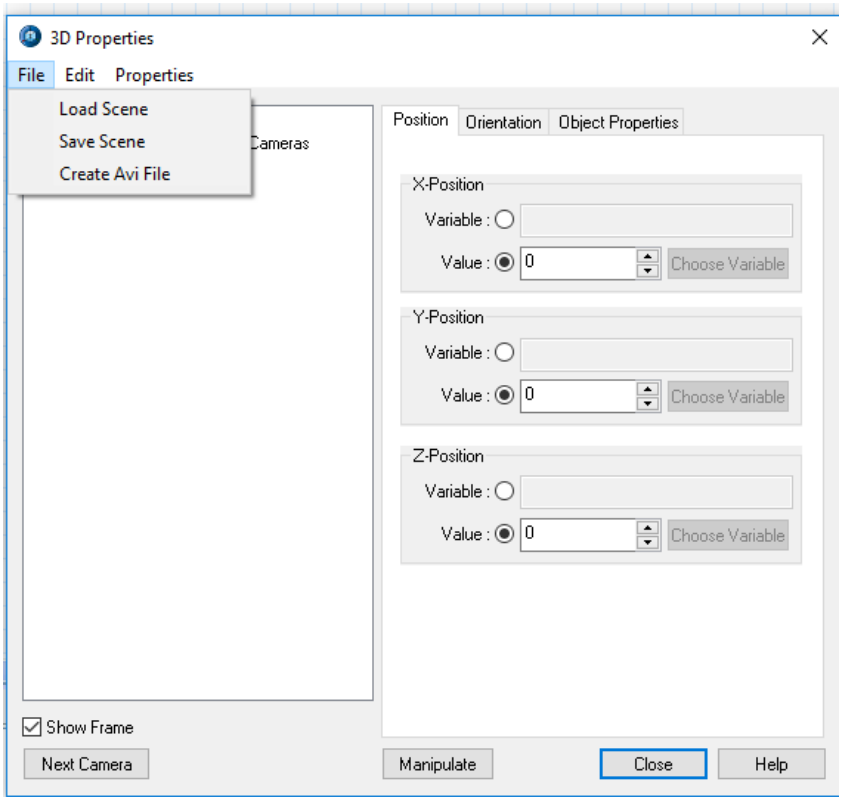


Figura 4. 22 Cargar escena previamente realizada en el toolbox mecánica 3D.

En el menú Archivo, seleccione la opción cargar escena y elija el archivo RobotScara1.scn.

Desactivamos la opción **"Show Frame"** (mostrar marco) para cada Marco de referencia.

Cerramos la ventana Propiedades.

Ahora se visualizará al robot que hemos hecho en el editor de mecánica 3D.

Regresamos al Simulador y desde el menú de Simulación elegir Ejecutar para comenzar una simulación.

En la figura 4.23 se ilustra la animación del robot Scara.

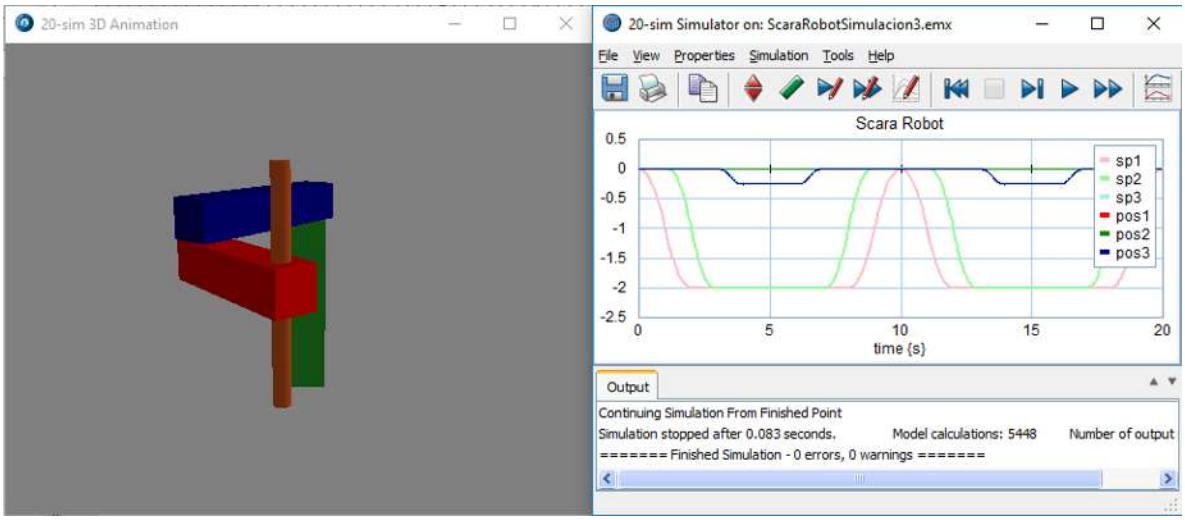


Figura 4. 23 Simulación del Robot Scara

4.3 Caso de Estudio 2 "Doble Péndulo"

En esta sección utilizaremos el Editor de mecánica 3D para crear el modelo de un doble péndulo. En este editor se puede definir la geometría del péndulo y revisar los movimientos posibles. Desde el editor de mecánica 3D generaremos un modelo de 20-sim que contendrá las ecuaciones de movimiento y un escenario para la animación del péndulo.

El modelo del péndulo que se va a crear tiene articulaciones accionadas. Esto quiere decir que podemos aplicar un torque a las articulaciones. En 20-sim, vamos a insertar el modelo de péndulo y activaremos el accionamiento a través de amortiguadores pasivos. Después de eso vamos a simular el modelo y mostrar una animación en 3D.

a) Inserción de componentes

Abra 20-sim y seleccione Archivo, Nuevo y Modelo Gráfico.

En el Editor, en el menú Herramientas, seleccione el toolbox de mecánica 3D, esto abrirá el editor de mecánica 3D.

En la figura4.24 se ilustra el editor de Mecánica 3D.

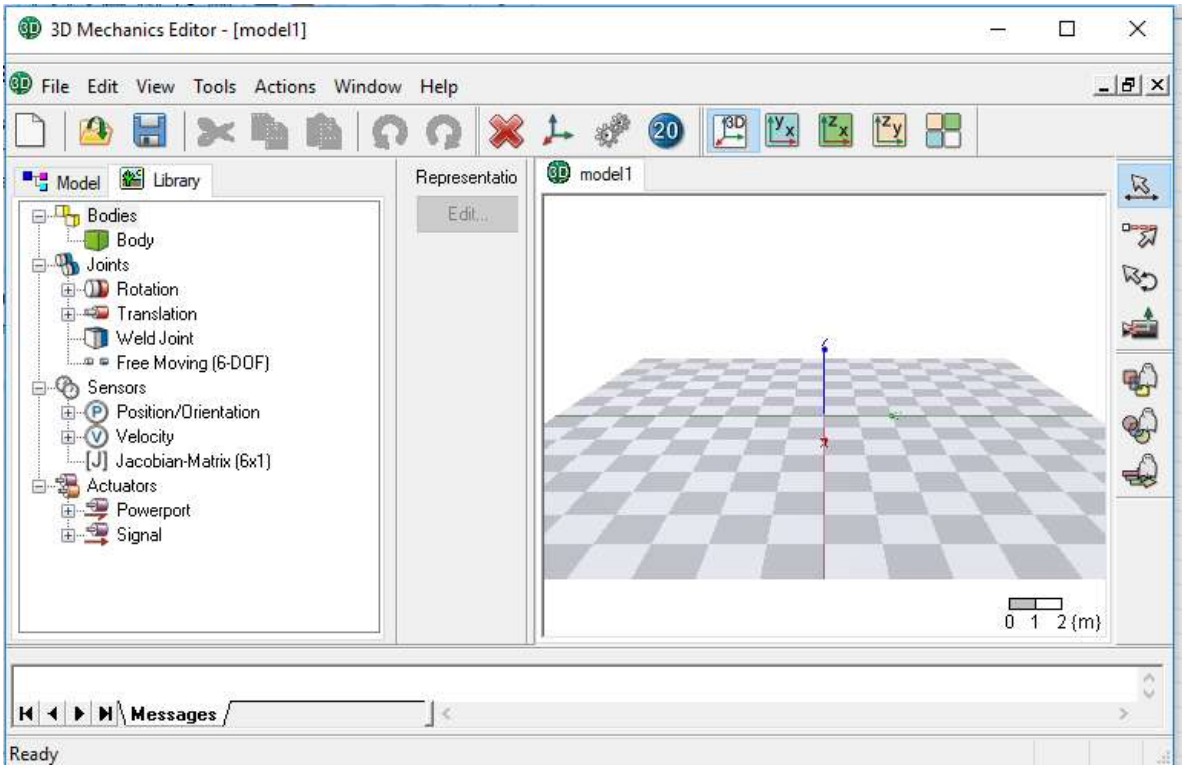


Figura 4. 24 Editor de Mecánica 3D.

Haga clic en la pestaña Biblioteca.

En la Biblioteca, haga clic en Cuerpos – Cuerpo(elemento) y arrástrelo al Editor.

El primer cuerpo que se inserta en el Editor gráfico se fijará al piso y no se moverá durante la simulación. El péndulo estará unido a este cuerpo.

Hacemos doble clic en el cuerpo para abrir las Propiedades del cuerpo.

Cambiamos el nombre del cuerpo a Base.

Seleccionamos la opción “Is Fixed World”.

Hacemos clic en el botón Posición y cambiamos la posición a (x = 4, y = -4, z = 0).

En la figura 4.25 se ilustra el cambio de nombre del cuerpo y la activación de la opción “Is Fixed World”.

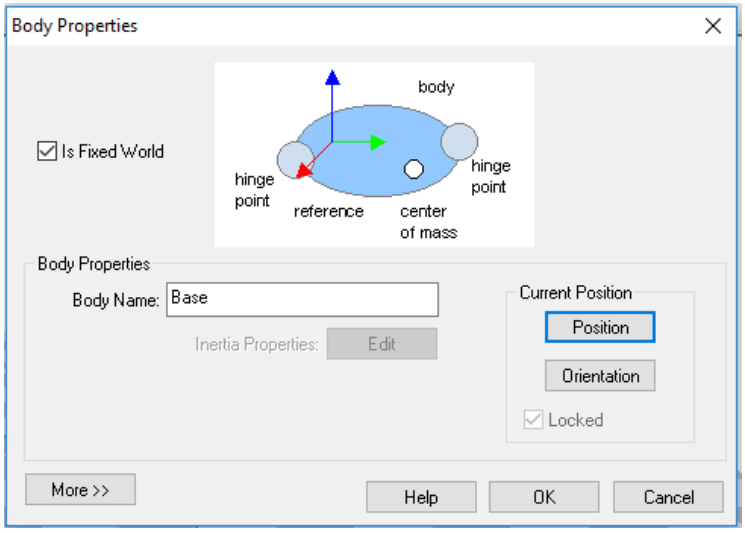


Figura 4. 25 Propiedades del cuerpo.

En la figura 4.26 se ilustra el cambio de posición de nuestro elemento base.

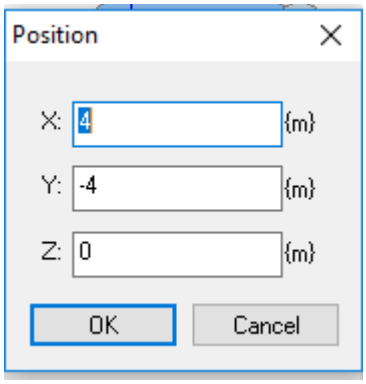


Figura 4. 26 Ajustes del cambio de posición.

Cerramos el cuadro de dialogo de Propiedades del cuerpo.

En la figura 4.27 se ilustra cómo queda el elemento después de aplicar los cambios anteriores.

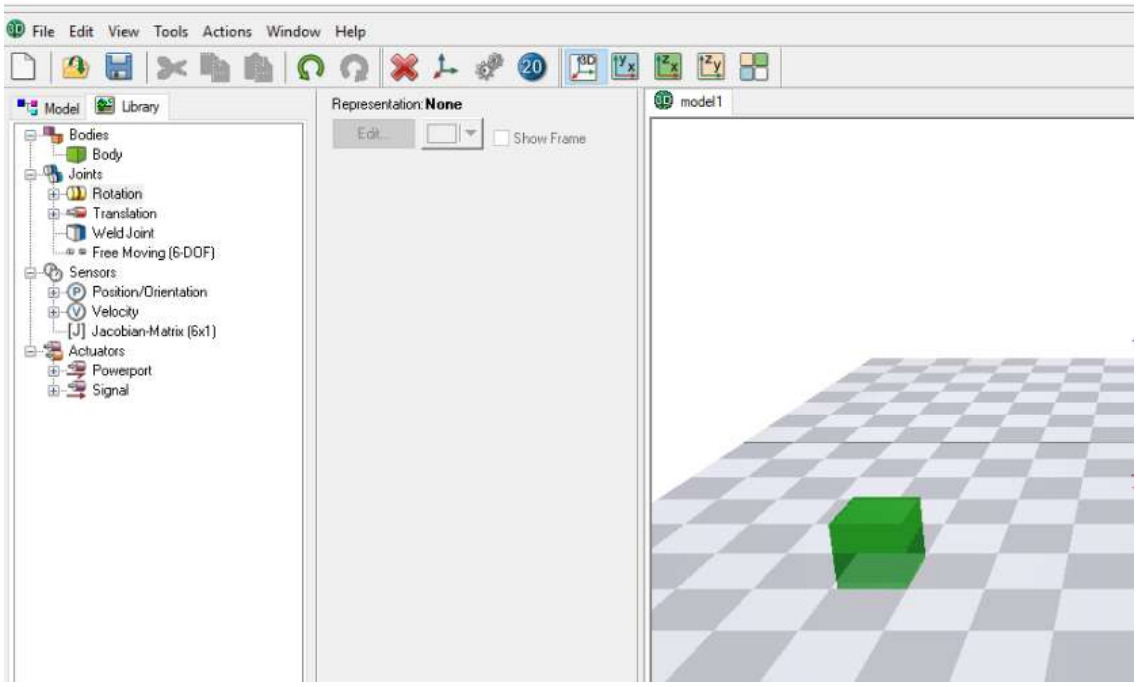


Figura 4. 27 Resultado del elemento después de los cambios realizados.

En la pestaña Biblioteca, hacemos clic en Joints - Rotation - Actuated – X-Rotation y arrastramos al Editor Gráfico.

Si se selecciona una unión, se alinea a través del centro de la articulación, esto indica los posibles movimientos que puede hacer con el cuerpo. Si hace clic con el botón izquierdo del mouse, puede mover la unión sobre la superficie.

Si presiona el botón Ctrl mientras mantiene presionado el botón izquierdo del mouse, el cuerpo sube y baja.

Seleccionamos la articulación y se coloca junto al cuerpo de la siguiente manera, como se ilustra en la figura 4.28.

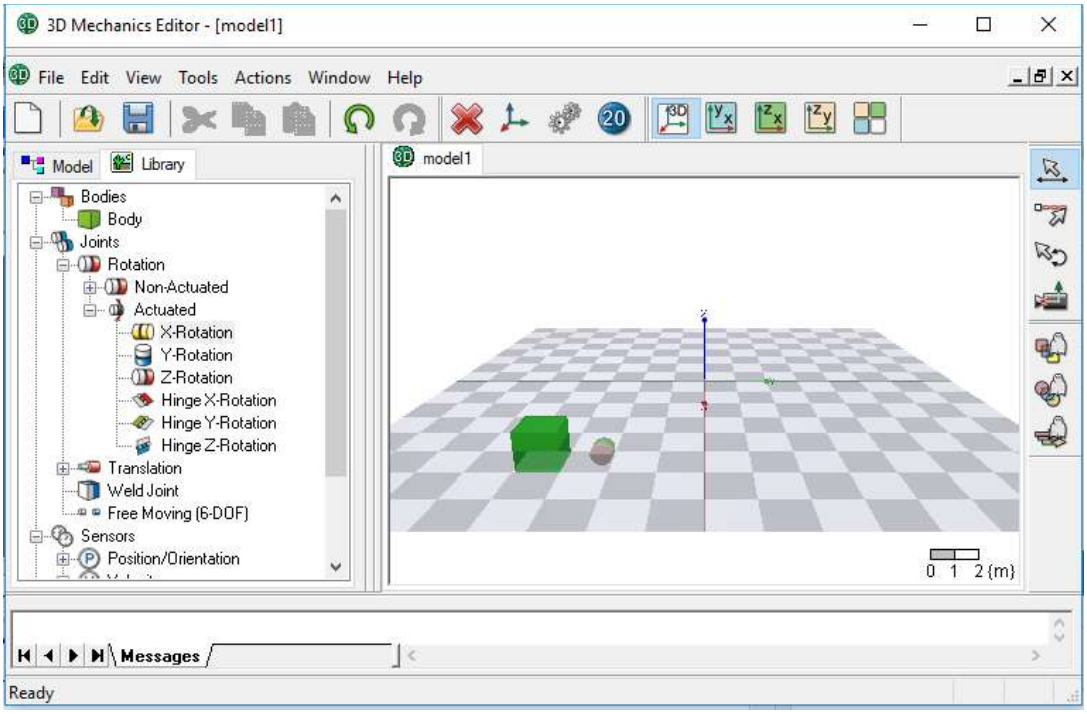


Figura 4. 28 En el modo de traslación, se pueden mover objetos fácilmente con el puntero del mouse.

Hacemos doble clic en la unión para abrir las Propiedades. En la figura 4.29 se ilustra cómo cambiamos el nombre de la unión a Base_Brazo1.

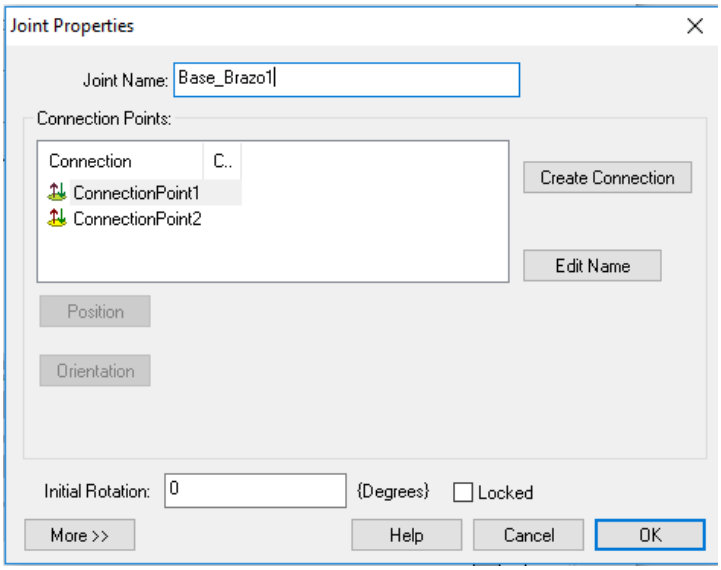


Figura 4. 29 Cambio el nombre del elemento unión a Base_Brazo1.

Cerramos el cuadro de dialogo “Propiedades”.

Insertamos dos cuerpos más y otra articulación actuada (x-rotation), a las cuales vamos a cambiar el nombre como, Brazo1, Brazo2, Brazo1_Brazo2. El resultado se verá cómo se ilustra en la figura 4.30.

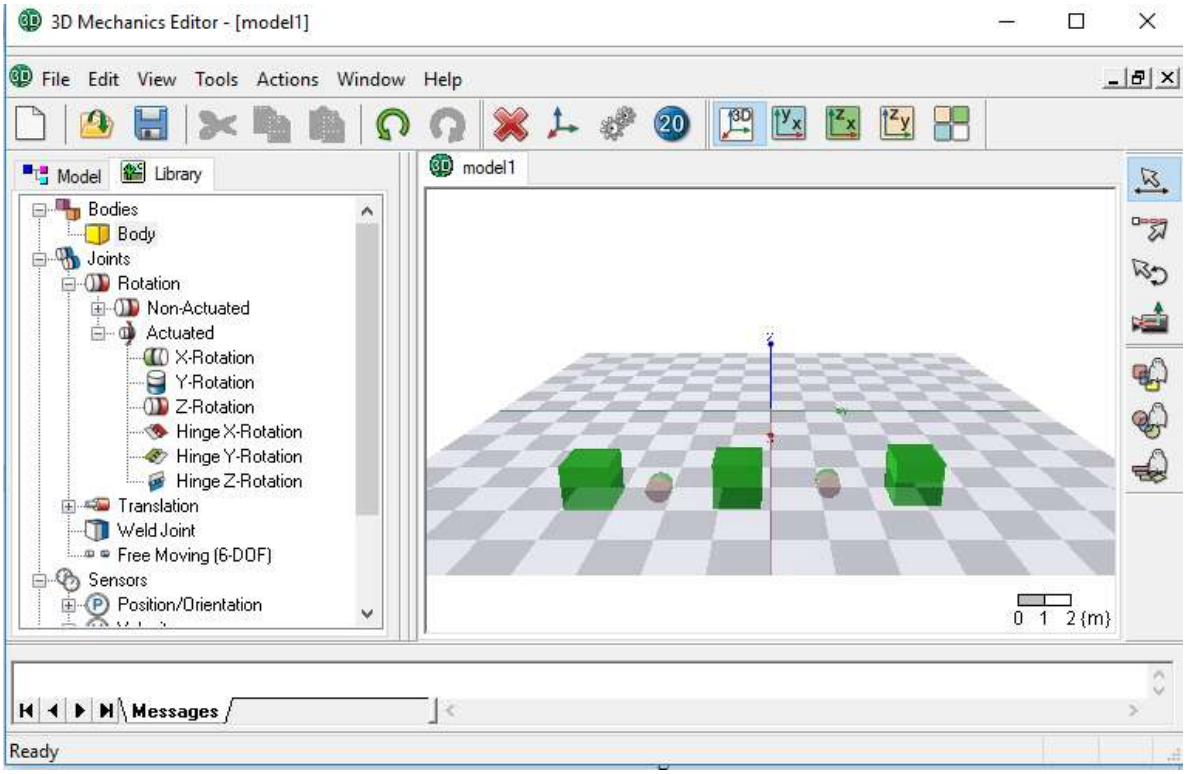


Figura 4. 30 Inserción de tres cuerpos y dos uniones.

Hacemos clic en  para configurar el Editor en modo de conexión.

Ahora podemos definir las conexiones haciendo clic en los componentes.

Colocamos el puntero del mouse en la parte superior del cuerpo de la base y hacemos clic con el botón izquierdo del mouse.

En la figura 4.31 se ilustra cómo es que ahora aparece una línea de puntos entre el cuerpo y el puntero del mouse para indicar que se está definiendo una conexión entre el cuerpo y un segundo componente.

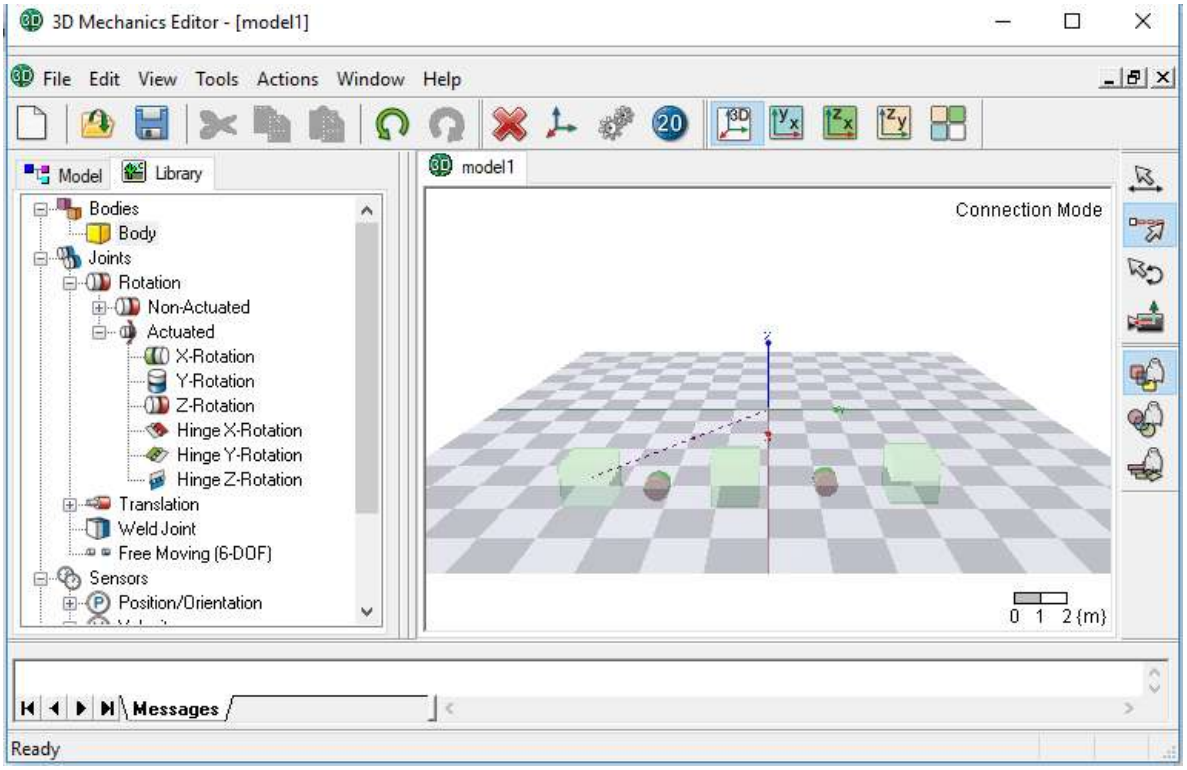


Figura 4. 31 Trazando conexiones con otros elementos.

Colocamos el mouse encima de la unión Base_Brazo1 y hacemos clic nuevamente.

Ahora se establece la conexión y se abre el cuadro de diálogo Crear conexión. En este cuadro de diálogo, puede definir el desplazamiento de posición de la unión Base_Brazo1 con respecto al cuerpo Base. Como se muestra en el cuadro de diálogo Crear conexión, cada unión tiene dos conexiones. Elegiremos “ConnectionPoint1” para comenzar.

Hacemos clic en el botón de posición y establecemos una nueva posición en x = 0, y = 0, z = 0.

En la figura 4.32 se ilustra cómo se crea la conexión entre la Base y la unión.

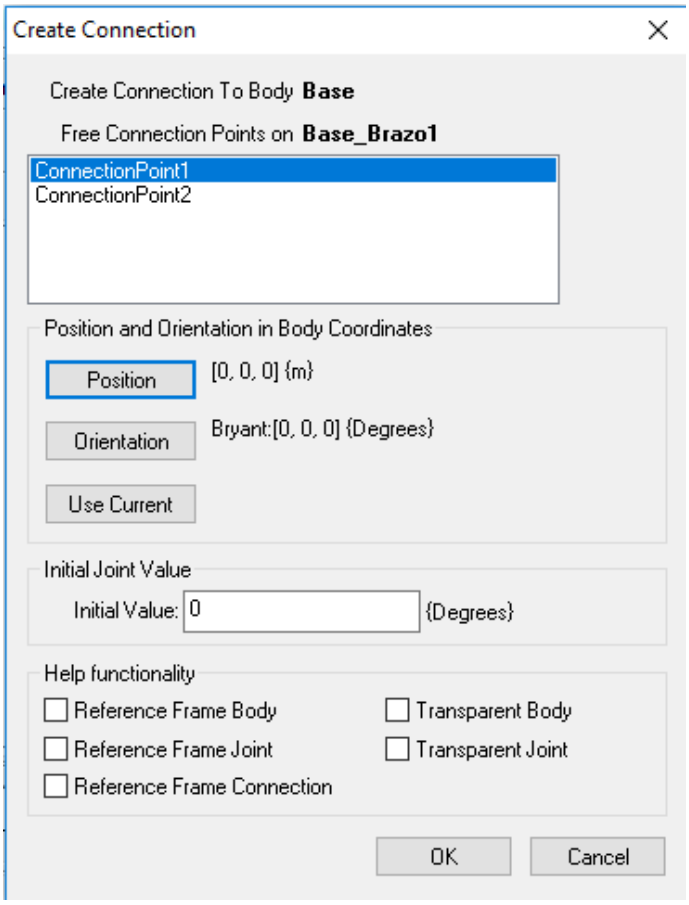


Figura 4. 32 Cuadro de dialogo "Crear conexión".

En la figura 4.33 se ilustra el cambio de posición de la unión.

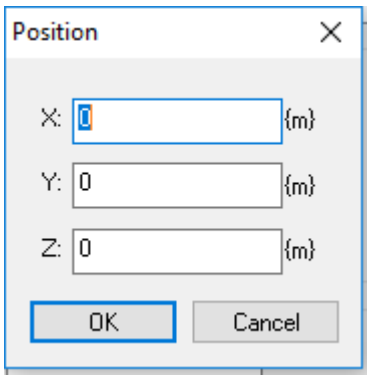


Figura 4. 33 Cambio de posición de la unión.

Cerramos el cuadro de diálogo Crear conexión.

Debido a que hemos establecido la posición de desplazamiento en cero, la unión Base_Brazo1 tiene la misma posición que el cuerpo base. Para ver mejor la articulación activaremos el modo fantasma para cuerpos, para hacer que los cuerpos sean transparentes.

Hacemos clic en  para configurar el editor de mecánica 3D en modo fantasma para cuerpos.

En la figura 4.34 se ilustra como la Base y la unión están en la misma posición y utilizando el modo fantasma podemos observar el resultado.

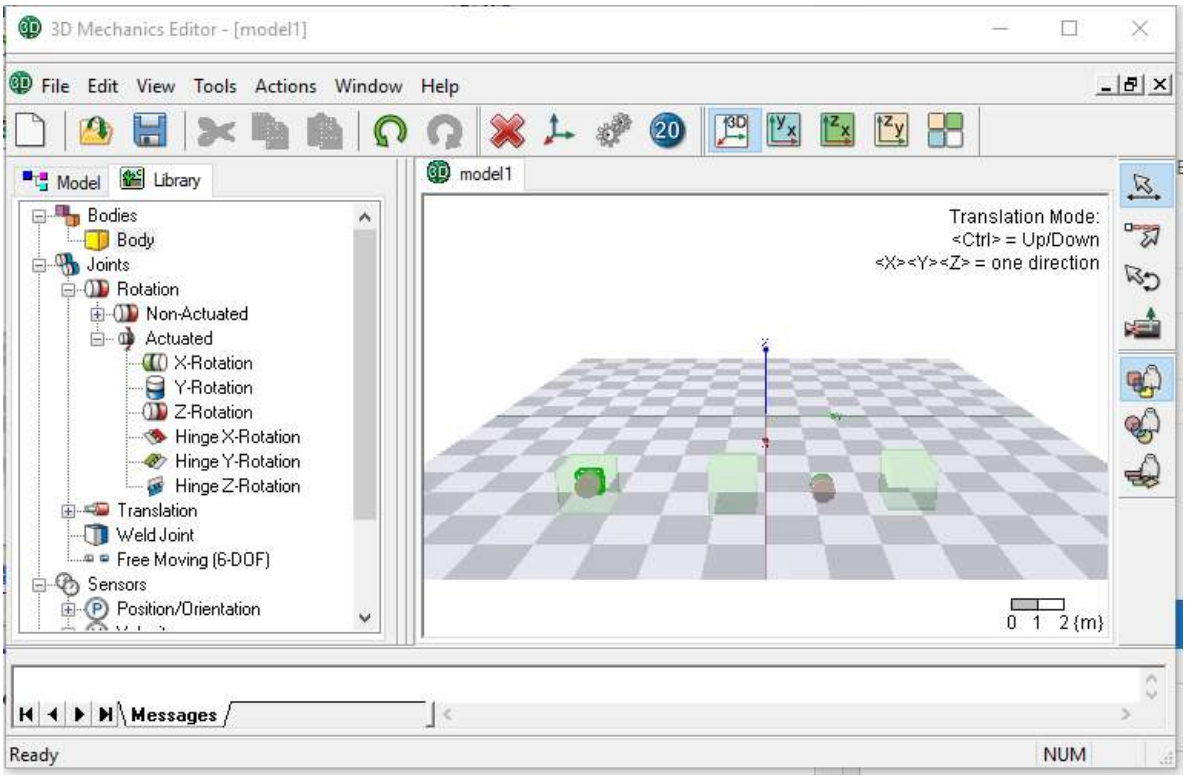


Figura 4. 34 Usamos el modo fantasma para hacer que los objetos sean transparentes.

Desactivamos el modo fantasma para cuerpos.

Crearemos las conexiones 2, 3 y 4 de acuerdo a las especificaciones de la tabla 4- 6.

Tabla 4- 6 Creando conexiones.

Acción	*Primer click*	*Segundo click*	*Posición*
1	Base_Brazo1	Base	x = 0, y = 0, z = 0
2	Base_Brazo1	Brazo1	x = 0, y = -2, z = 0
3	Base_Brazo2	Brazo1	x = 0, y = 2, z = 0
4	Base_Brazo2	Brazo2	x = 0, y = -2, z = 0

En la figura 4.35 se ilustra el resultado de todas las conexiones del péndulo.

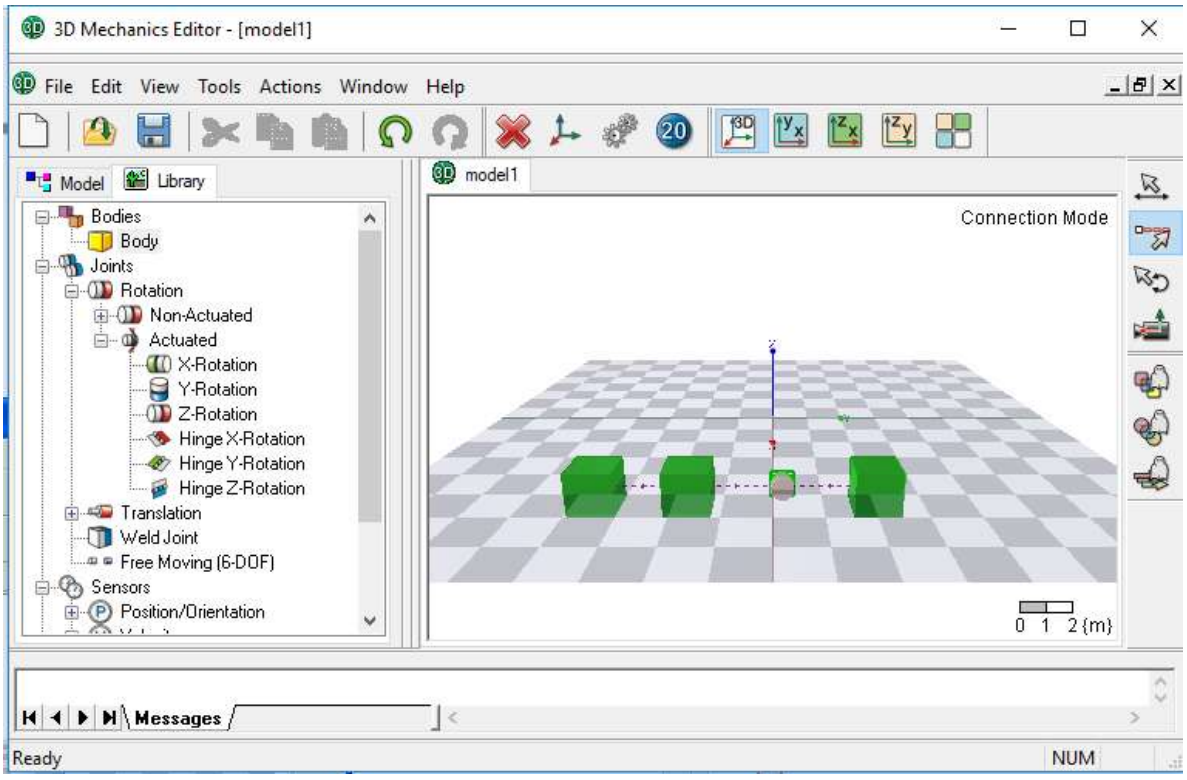


Figura 4. 35 Resultado de crear las conexiones del péndulo.

Guardamos el modelo, así que siempre podemos volver a cargarlo cuando algo salga incorrecto.

En el menú Archivo, haga clic en “Guardar como” y seleccionamos el directorio donde lo queremos guardar.

Comprobando el movimiento

Para ver si definimos correctamente el modelo, verificamos cómo se puede mover.

Hacemos clic en  para configurar el modo de traslación.

Desde el menú Acciones, seleccione Verificar modelo.

El cuadro de diálogo Mensaje debe mostrar el “Análisis completado con éxito". Vamos a revisar los posibles movimientos del modelo.

Colocamos el puntero del mouse en la parte superior del cuerpo de Arm1 y haga clic con el botón izquierdo del mouse.

La unión Base_Brazo1 mostrará una flecha que indica rotación positiva.

Mantenga presionado el botón izquierdo del mouse y mover hacia arriba y hacia abajo.

En la figura 4.36 se ilustra el resultado y se revisa que las uniones y el brazo se puedan mover libremente.

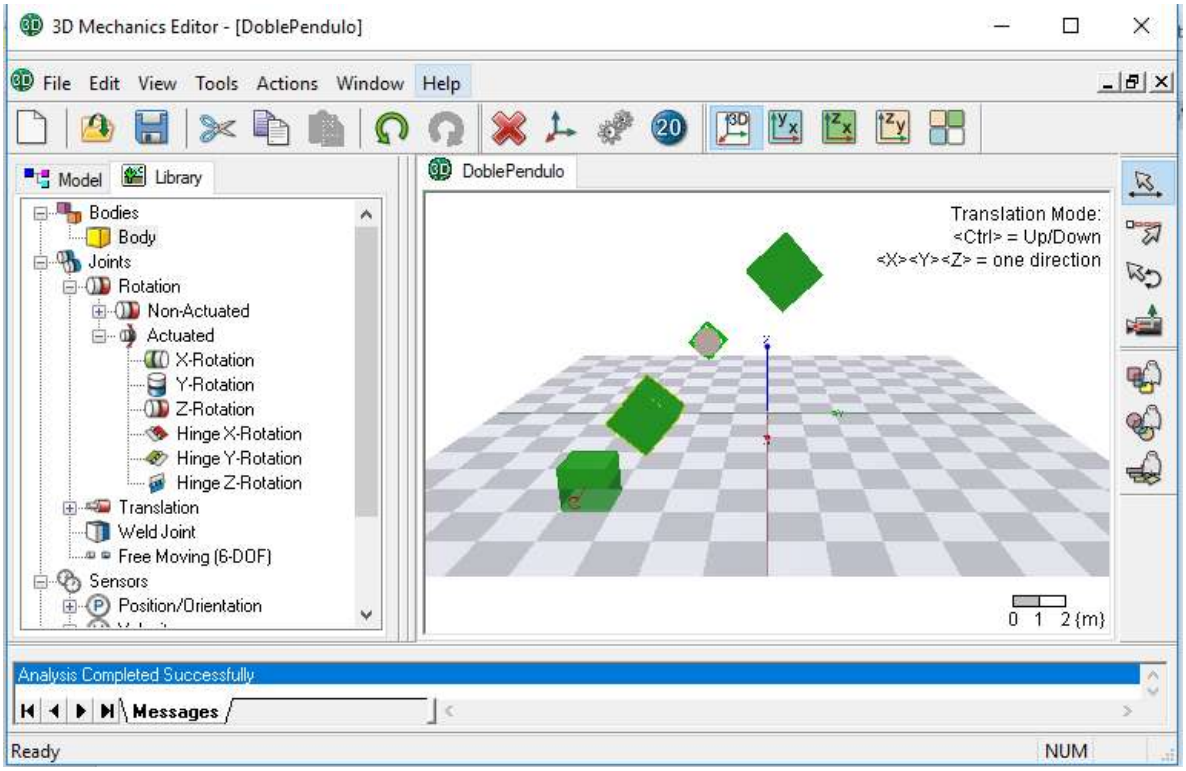


Figura 4. 36 Se verifica que el péndulo se puede mover.

Finalmente centraremos el mecanismo en el marco de referencia base. Esto facilitará la animación en 3D durante la simulación.

Hacemos doble clic en el cuerpo Base que abre el cuadro de diálogo Propiedades del cuerpo.

Se restablece la posición en x = 0, y = 0 y, z = 0 se cierra el cuadro de diálogo Propiedades del cuerpo.

Desde el menú Acciones, seleccionamos “Verificar modelo”.

Durante la comprobación, todos los ángulos de articulación se restablecerán a su valor inicial. Como no hemos cambiado los valores iniciales (por defecto = 0) el mecanismo volverá a su configuración original, pero ahora con el cuerpo base en el centro del marco de referencia global.

En la figura 4.37 se ilustra como devolvemos el péndulo a su configuración inicial y centramos el cuerpo base al marco de referencia.

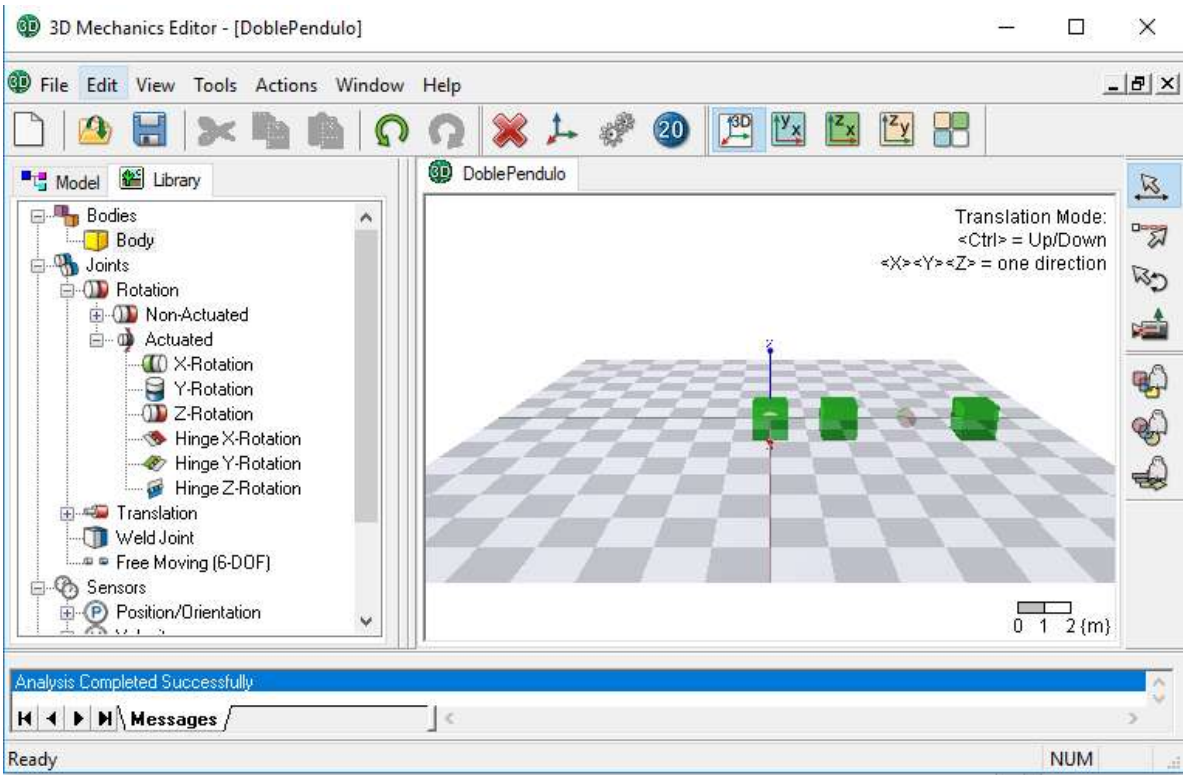


Figura 4. 37 Se traslada el péndulo al marco de referencia.

Generación del modelo a 20-sim

Desde el menú Acciones seleccionamos Generar modelo de 20-sim. Con lo cual se abrirá un cuadro de dialogo.

En la figura 4.38 se ilustra el cuadro de dialogo para generar nuestro modelo de 20-sim y la escena de este mismo.

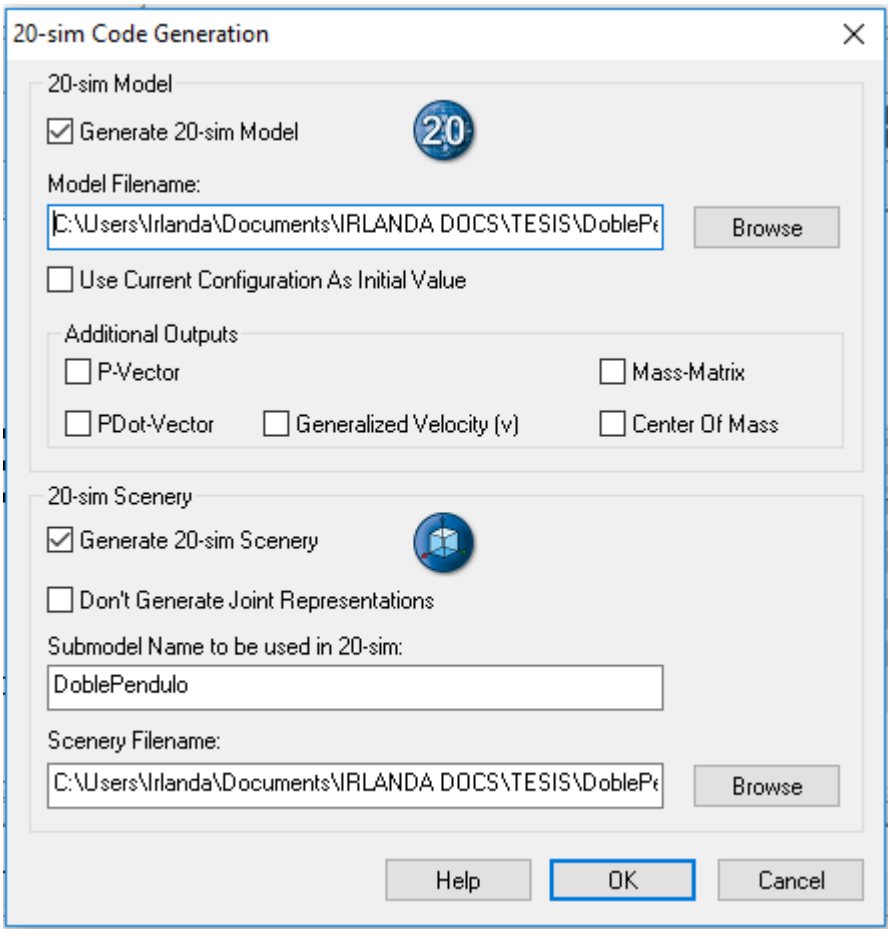


Figura 4. 38 Generando modelo de 20-sim y la escena de este mismo.

Hacemos clic en OK para generar el código de nuestro modelo.

Simulación

Volveremos a 20-sim donde usaremos el modelo para realizar una simulación. El editor deberá mostrar el modelo vacío, sino no es así abrimos un nuevo modelo gráfico.

Abrimos el Explorador de archivos y buscaremos en el directorio que almacenamos el péndulo.

Lo arrastramos y lo soltamos en el Editor.

En la figura 4.39 se ilustra como seleccionamos el modelo insertado y cambiamos el nombre local (Modelo - Atributos) a DoublePendulum.

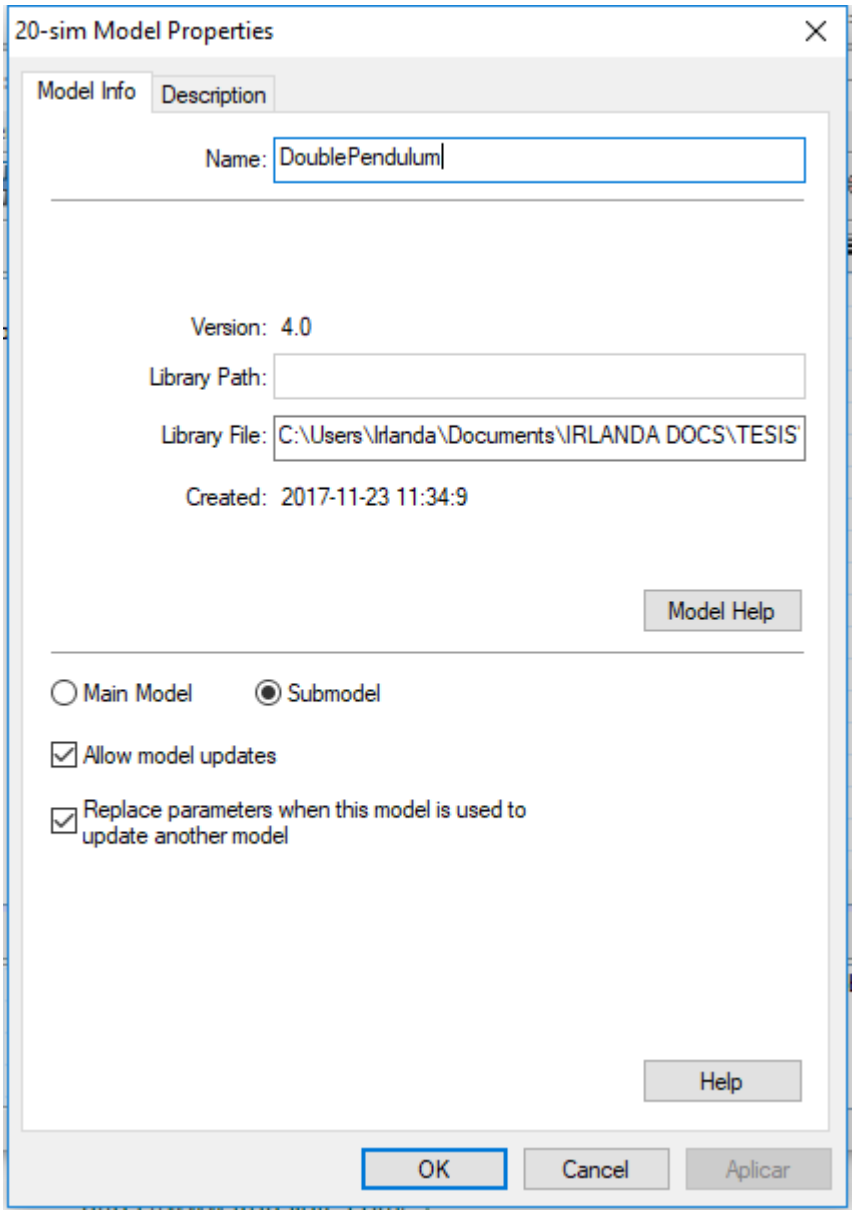


Figura 4. 39 Cambiamos el nombre de nuestro modelo.

En la figura 4.40 se ilustra cómo es que queda el modelo después de cambiar el nombre.

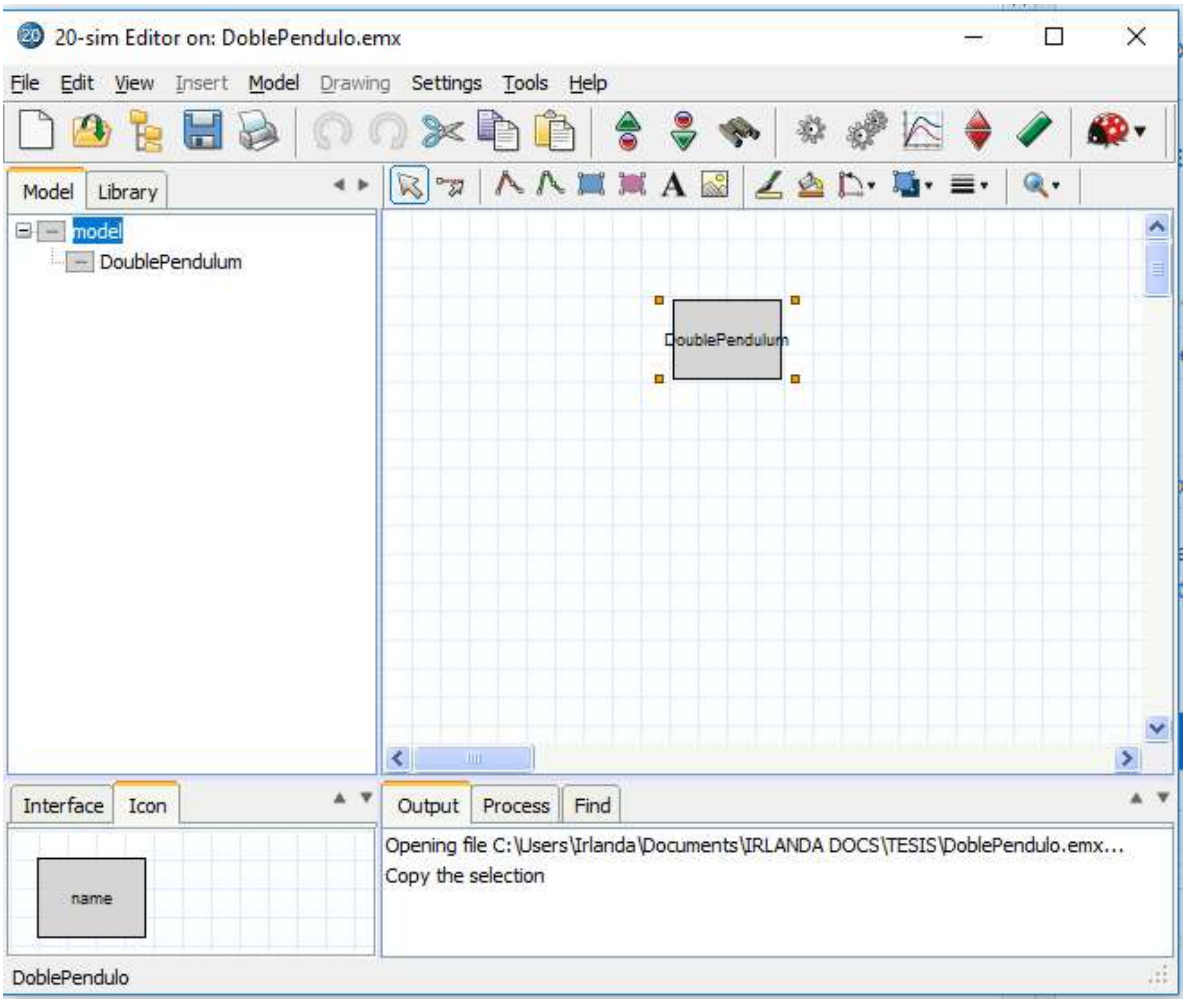


Figura 4. 40 Resultado después de cambiar el nombre de nuestro modelo.

Como se mencionó al principio, hemos utilizado articulaciones actuadas. Esto significa que podemos aplicar un torque a las articulaciones y ver cómo se moverá el péndulo. El par se aplica a través de los puertos de potencia. Si observa en la pestaña Interfaz, se puede ver que tiene dos puertos. Uno para el brazo Base_Arm1 conjunta y uno para la unión Brazo1_Brazo2. Vamos a aplicar la activación pasiva uniendo los amortiguadores al modelo.

Vamos a la pestaña Biblioteca y seleccione Biblioteca \ Diagramas icónicos \ Mecánica \ Rotación \ Componentes.

Insertamos el modelo “Bearing Twice”.

Conectamos los modelos de rodamiento con el DoublePendulum.

En la figura 4.41 se ilustra la conexión de los “Bearings”.

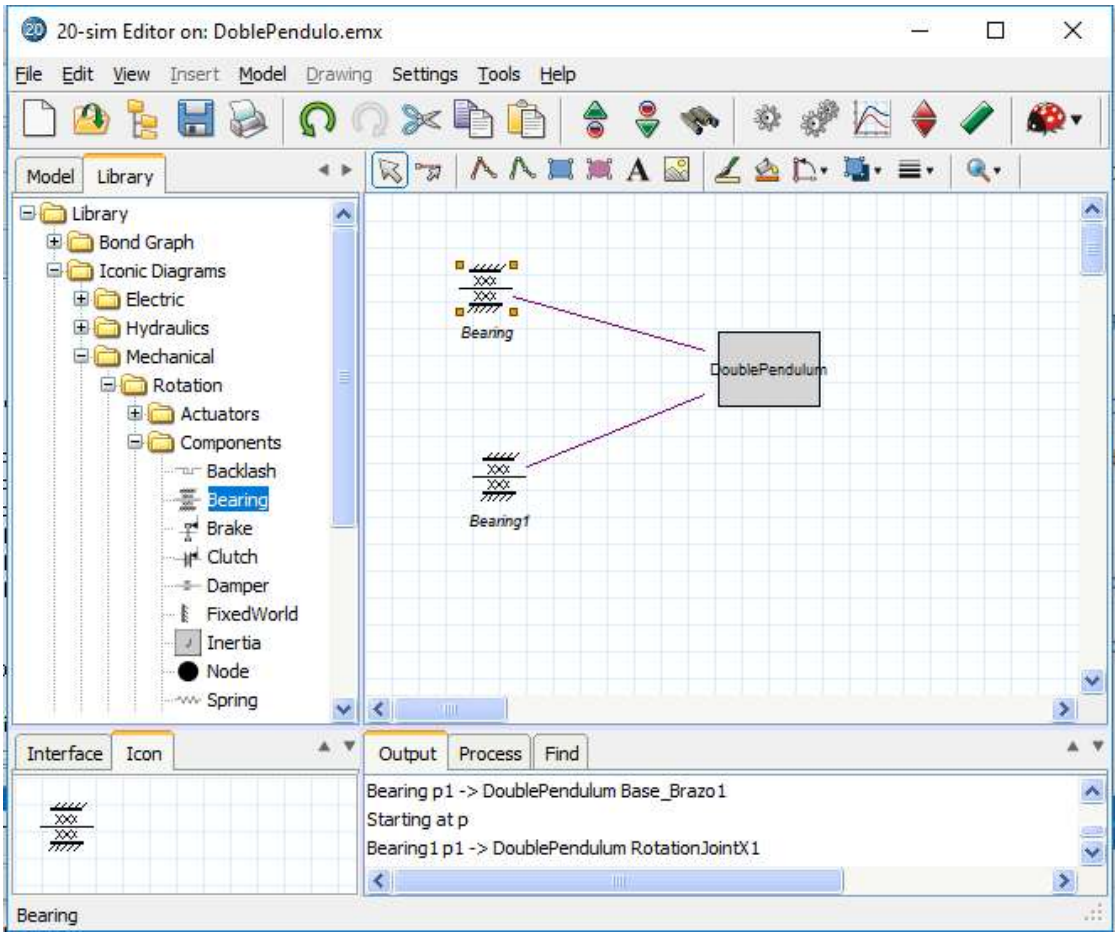


Figura 4. 41 Conexión de los bearings.

En el menú Archivo, hacemos clic en Guardar. Almacene el modelo en un directorio utilizando el nombre DoublePendulumSimulation.emx.

En el menú Modelo, haga clic en Iniciar simulador. Esto abrirá el menú del simulador. Ahora se puede establecer los valores de los parámetros, elegir un método de integración, establecer las variables de trazado, etc. Nosotros utilizaremos los valores de parámetros predeterminados y realizaremos una animación.

En el simulador, en el menú Herramientas, seleccionamos el toolbox de animación y animación 3D.

En la ventana de Animación 3D, haga doble clic para abrir la ventana Propiedades de Animación 3D.

En el menú Archivo, seleccione la opción “Cargar escena” y elija el archivo DoblePendulo.scn. En la figura 4.42 se ilustra la escena del péndulo cargada.

Figura 4. 42 Escena del péndulo cargada.

Regresamos al simulador e iniciamos la simulación.

En la figura 4.43 se ilustra el resultado de la animación del péndulo.

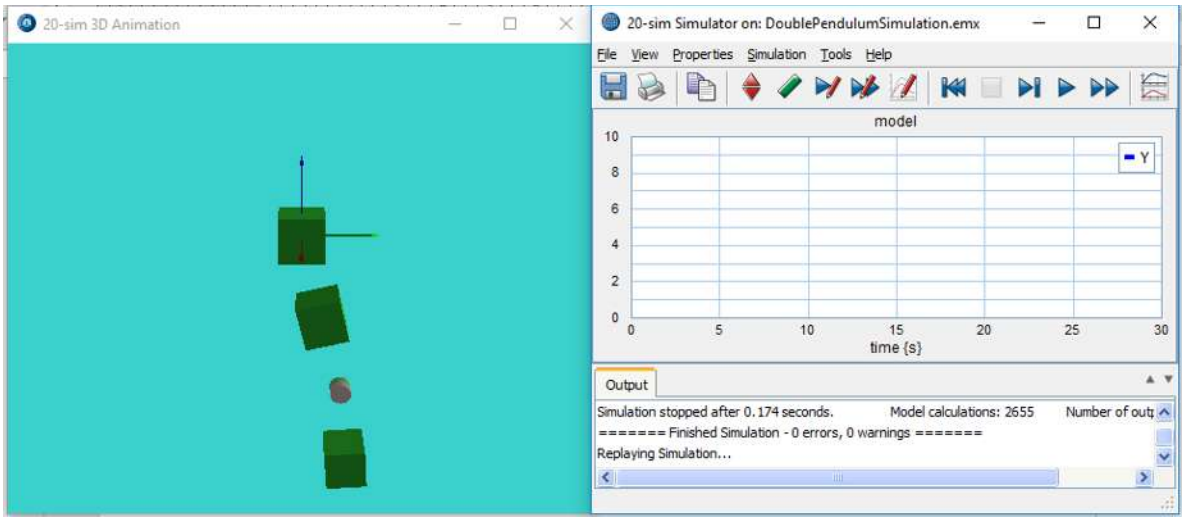


Figura 4. 43 Simulación del doble péndulo.

CAPITULO 5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

La técnica de simulación fue un concepto que tomó fuerza en la década de 1950, cuando se dio gran importancia al proceso de dividir un problema en partes para analizar la interacción entre cada una de ellas. De esta manera surgió la simulación como una potente herramienta que permite realizar pruebas sobre un proyecto sin dañar algún equipo.

El actual trabajo de investigación presenta un apoyo importante para los estudiantes que han adquirido conocimientos teóricos y quieren llevar acabo ese conocimiento a la práctica, que quizás se han limitado por diversos factores como lo son, el tiempo, el costo del equipo, material, pues la simulación les permite reforzar algunos de los conocimientos de manera visual e interactiva.

Los casos de estudios fueron realizados utilizando el toolbox de Mecánica 3D que contiene 20-sim, es un toolbox muy interesante ya que puedes experimentar con diversos elementos que este nos proporciona, y así llevar en práctica los conocimientos teóricos adquiridos, ver cómo se comporta en un entorno virtual, antes de llevarlo a un entorno real.

No solamente se puede realizar la animación desde este toolbox, sino también estando en el editor de 20-sim con nuestro modelo podemos llegar a generar dicha animación, es un software muy versátil, ya que cuenta con diferentes funciones para probar con diferentes e infinidad de ejemplos.

Me dio mucha satisfacción poder aprender a manejar un software realmente interesante, termino este trabajo sabiendo manejarlo, quizás no al 100% ya que me enfoque en conocer el toolbox de Mecánica 3D y 20-SIM cuenta con varios toolboxes y bibliotecas, las cuales desglosan infinidades de elementos, los cuales me faltó el tiempo de explorar y saber su utilización, en un futuro espero saber su utilización en un 100%, porque es un software realmente bueno, que vale la pena conocer, probar y realizar diversos proyectos con este.

5.2 RECOMENDACIONES

Es necesario que todos los estudiantes de ingeniería, conozcan y manejan al menos un software de simulación. En esta tesis se utilizó el software 20-sim, por la razón de que se tenía muy poco conocimiento de este y se quiso explorar la funcionalidad del toolbox de mecánica 3D. Sin embargo, en la actualidad existen varios softwares de simulación, algunos de ellos son: Matlab, Modelica, LabView, Scilab, Octave, los cuales se pueden aplicar en la industria para el diseño de diferentes tipos de sistemas y ver cómo es que se comportan, antes de llevarlos a un entorno real.

También recomiendo que cuando se desee llevar acabo el modelado y simulación de sistemas, se formule claramente el problema, se definan los objetivos que se deseen alcanzar, así como tener en cuenta de las limitaciones con las que se cuentan, siempre que se pueda hacer simplificaciones, obviamente cuidando no eliminar los detalles que sean importantes.

Por último, recomiendo al lector seguir aprendiendo de este software, sacar el máximo de las características de este y así mismo facilitar la mejor comprensión de los conocimientos adquiridos teóricamente en las aulas.

BIBLIOGRAFÍA

[1] E. E. TARIFA, «http://www.econ.unicen.edu.ar,» [En línea]. Available: http://bit.ly/2EIhiOJ. [Último acceso: 22 AGOSTO 2017].

[2] L. S. &. T. Solutions, «http://www.landersimulation.com,» 2016. [En línea]. Available: http://bit.ly/2EyjkkW. [Último acceso: 21 AGOSTO 2017].

[3] T. MathWorks, «www.mathworks.com,» The MathWorks, 1994-2017. [En línea]. Available: http://bit.ly/2Gl1M8A. [Último acceso: 09 SEPTIEMBRE 2017].

[4] M. Association, «www.modelica.org,» 2000-2017. [En línea]. Available: http://bit.ly/2Ev81WI. [Último acceso: 09 SEPTIEMBRE 2017].

[5] N. I. Corporation, «www.ni.com/es-mx/,» 2017. [En línea]. Available: http://bit.ly/2EJOoNZ. [Último acceso: 11 SEPTIEMBRE 2017].

[6] E. GROUP, «www.scilab.org,» 2017. [En línea]. Available: http://bit.ly/2BAzTdE. [Último acceso: 18 SEPTIEMBRE 2017].

[7] J. W. Eaton, «www.gnu.org,» 1998-2017. [En línea]. Available: http://bit.ly/1uhpETz. [Último acceso: 15 AGOSTO 2017].

[8] F. d. C. Empresariales, «dinamicasistemas.utalca.cl,» 2012. [En línea]. Available: http://bit.ly/2Crq7a9. [Último acceso: 02 SEPTIEMBRE 2017].

[9] Controllab, «www.controllab.nl,» 2017. [En línea]. Available: http://bit.ly/2Gh5GiH. [Último acceso: 15 AGOSTO 2017].

[10] C. Products, «www.20sim.com,» [En línea]. Available: http://bit.ly/2sALtCn. [Último acceso: 15 AGOSTO 2017].

[11] C. Products, «www.20sim.com,» [En línea]. Available: http://bit.ly/2F7I9kS. [Último acceso: 15 AGOSTO 2017].

[12] «Controllab Products,» [En línea]. Available: http://bit.ly/2Bx3Wmq. [Último acceso: 15 AGOSTO 2017].

[13] C. Products, «www.20sim.com,» [En línea]. Available: http://bit.ly/2F9Crir. [Último acceso: 17 AGOSTO 2017].

[14] C. Products, «www.20sim.com,» [En línea]. Available: http://bit.ly/2EIfsxl. [Último acceso: 19 AGOSTO 2017].

[15] C. Products, «www.20sim.com,» [En línea]. Available: http://bit.ly/2o4KFBg. [Último acceso: 2017 AGOSTO 2017].

[16] C. Products, «www.20sim.com,» [En línea]. Available: http://bit.ly/2o9WhC2. [Último acceso: 29 AGOSTO 2017].

APÉNDICES

Código generado del Robot Scara de 3D Mecánica a 20-sim

```
variables
real hidden World\AbsH[4,4];
real hidden World\mass_x_I[6,6];
real hidden BaseWeld\ConnectionPoint1\AbsH[4,4];
real hidden BaseWeld\AbsH[4,4];
real hidden BaseWeld\ConnectionPoint2\AbsH[4,4];
real hidden Base\AbsH[4,4];
real hidden Base\mass_x_I[6,6];
real hidden Base_Brazo1\ConnectionPoint1\AbsH[4,4];
real hidden Base_Brazo1\AbsH[4,4];
real hidden Base_Brazo1\RelH[4,4];
real Base_Brazo1\theta;
real Base_Brazo1\thetaDot;
real Base_Brazo1\sinTh;
real Base_Brazo1\CosTh;
real Base_Brazo1\Rot[3,3];
real Base_Brazo1\Pos[3];
real hidden Base_Brazo1\ConnectionPoint2\AbsH[4,4];
real hidden Brazo1\AbsH[4,4];
real hidden Brazo1\mass_x_I[6,6];
real hidden Brazo1_Brazo2\ConnectionPoint1\AbsH[4,4];
real hidden Brazo1_Brazo2\AbsH[4,4];
real hidden Brazo1_Brazo2\RelH[4,4];
real Brazo1_Brazo2\theta;
real Brazo1_Brazo2\thetaDot;
real Brazo1_Brazo2\sinTh;
real Brazo1_Brazo2\CosTh;
real Brazo1_Brazo2\Rot[3,3];
real Brazo1_Brazo2\Pos[3];
real hidden Brazo1_Brazo2\ConnectionPoint2\AbsH[4,4];
real hidden Brazo2\AbsH[4,4];
```

```
real hidden Brazo2\mass_x_I[6,6];
real hidden Brazo2_Carga\ConnectionPoint1\AbsH[4,4];
real hidden Brazo2_Carga\AbsH[4,4];
real hidden Brazo2_Carga\RelH[4,4];
real Brazo2_Carga\pos;
real Brazo2_Carga\posDot;
real Brazo2_Carga\Rot[3,3];
real Brazo2_Carga\Pos[3];
real hidden Brazo2_Carga\ConnectionPoint2\AbsH[4,4];
real hidden Carga\AbsH[4,4];
real hidden Carga\mass_x_I[6,6];
real Brazo2_Carga\Wrench[1,6];
real Brazo2_Carga\force;
real Brazo2_Carga\ConnectionPoint2\AbsHinverse[4,4];
real Brazo1_Brazo2\Wrench[1,6];
real Brazo1_Brazo2\torque;
real Brazo1_Brazo2\ConnectionPoint2\AbsHinverse[4,4];
real Base_Brazo1\Wrench[1,6];
real Base_Brazo1\torque;
real Base_Brazo1\ConnectionPoint2\AbsHinverse[4,4];
real v[3];
real P[3];
real PDot[3];
real PDotBodies[3];
real hidden AdjX2[6];
real hidden AdjX2_T[1,6];
real hidden adjAdjX2[6,6];
real hidden AdjX3[6];
real hidden AdjX3_T[1,6];
real hidden adjAdjX3[6,6];
real hidden AdjX4[6];
real hidden AdjX4_T[1,6];
real hidden adjAdjX4[6,6];
real hidden J1[6,3];
real hidden J2[6,3];
real hidden J3[6,3];
real hidden J4[6,3];
real hidden T1_00[6];
```

```
real hidden T1_00T[1,6];
real hidden T2_00[6];
real hidden T2_00T[1,6];
real hidden T3_00[6];
real hidden T3_00T[1,6];
real hidden T4_00[6];
real hidden T4_00T[1,6];
real hidden M1[6,6];
real hidden AdjInv1[6,6];
real hidden TM1[1,6];
real hidden M2[6,6];
real hidden AdjInv2[6,6];
real hidden TM2[1,6];
real hidden M3[6,6];
real hidden AdjInv3[6,6];
real hidden TM3[1,6];
real hidden M4[6,6];
real hidden AdjInv4[6,6];
real hidden TM4[1,6];
real MassMatrix[3,3];
real hidden M2_2;
real hidden M3_2;
real hidden M3_3;
real hidden M4_2;
real hidden M4_3;
real hidden M4_4;
real hidden wrenchGrav2[1,6];
real hidden wrenchGrav3[1,6];
real hidden wrenchGrav4[1,6];
parameters
real hidden World\H[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden World\I[6,6] = [
0.1, 0, 0, 0, 0, 0;
0, 0.1, 0, 0, 0, 0;
```

```
0, 0, 0.1, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real World\mass = 1;
real hidden BaseWeld\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0.25;
0, 0, 0, 1];
real hidden Base\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden Base\I[6,6] = [
0.1, 0, 0, 0, 0, 0;
0, 0.1, 0, 0, 0, 0;
0, 0, 0.1, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real Base\mass = 1;
real hidden Base_Brazo1\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0.25;
0, 0, 0, 1];
real Base_Brazo1\thetaInitial = 0;
real hidden Brazo1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, -0.2;
0, 0, 1, 0.05;
0, 0, 0, 1];
real hidden Brazo1\I[6,6] = [
0.0125, 0, 0, 0, 0, 0;
0, 0.0025, 0, 0, 0, 0;
0, 0, 0.0125, 0, 0, 0;
```

```
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real Brazo1\mass = 8;
real hidden Brazo1_Brazo2\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, -0.2;
0, 0, 1, -0.05;
0, 0, 0, 1];
real Brazo1_Brazo2\thetaInitial = 0;
real hidden Brazo2\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, -0.15;
0, 0, 1, -0.05;
0, 0, 0, 1];
real hidden Brazo2\I[6,6] = [
0.00833333333333333, 0, 0, 0, 0, 0;
0, 0.00166666666666667, 0, 0, 0, 0;
0, 0, 0.00833333333333333, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real Brazo2\mass = 6;
real hidden Brazo2_Carga\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, -0.15;
0, 0, 1, 0;
0, 0, 0, 1];
real Brazo2_Carga\posInitial = 0;
real hidden Carga\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden Carga\I[6,6] = [
0.02, 0, 0, 0, 0, 0;
0, 0.02, 0, 0, 0, 0;
0, 0, 0.001, 0, 0, 0;
```

```
0, 0, 0, 1, 0, 0;

0, 0, 0, 0, 1, 0;

0, 0, 0, 0, 0, 1];

real Carga\mass = 0.5;

real gravity[3] = [0; 0; -9.81];

real hidden PDotInitial[3] = [0; 0; 0];

initialequations

J1 = 0;

J2 = 0;

J3 = 0;

J4 = 0;

MassMatrix = 0;

equations

World\AbsH = World\H;

World\mass_x_I = World\mass * World\I;

BaseWeld\ConnectionPoint1\AbsH = multiplyH (World\AbsH, BaseWeld\ConnectionPoint1\RelH);

BaseWeld\AbsH = BaseWeld\ConnectionPoint1\AbsH;

BaseWeld\ConnectionPoint2\AbsH = BaseWeld\AbsH;

Base\AbsH = multiplyH (BaseWeld\ConnectionPoint2\AbsH, Base\RelH);

Base\mass_x_I = Base\mass * Base\I;

Base_Brazo1\ConnectionPoint1\AbsH = multiplyH (Base\AbsH, Base_Brazo1\ConnectionPoint1\RelH);

Base_Brazo1\theta = int (Base_Brazo1\thetaDot) + Base_Brazo1\thetaInitial;

Base_Brazo1\sinTh = sin (Base_Brazo1\theta);

Base_Brazo1\CosTh = cos (Base_Brazo1\theta);

Base_Brazo1\Rot = [Base_Brazo1\CosTh, -Base_Brazo1\sinTh, 0;

Base_Brazo1\sinTh, Base_Brazo1\CosTh, 0;

0, 0, 1];

Base_Brazo1\Pos = 0;

Base_Brazo1\RelH = homogeneous (Base_Brazo1\Rot, Base_Brazo1\Pos);

Base_Brazo1\AbsH = multiplyH (Base_Brazo1\ConnectionPoint1\AbsH, Base_Brazo1\RelH);

Base_Brazo1\ConnectionPoint2\AbsH = Base_Brazo1\AbsH;

Brazo1\AbsH = multiplyH (Base_Brazo1\ConnectionPoint2\AbsH, Brazo1\RelH);

Brazo1\mass_x_I = Brazo1\mass * Brazo1\I;

Brazo1_Brazo2\ConnectionPoint1\AbsH = multiplyH (Brazo1\AbsH, Brazo1_Brazo2\ConnectionPoint1\RelH);

Brazo1_Brazo2\theta = int (Brazo1_Brazo2\thetaDot) + Brazo1_Brazo2\thetaInitial;

Brazo1_Brazo2\sinTh = sin (Brazo1_Brazo2\theta);
```

```
Brazo1_Brazo2\CosTh = cos (Brazo1_Brazo2\theta);

Brazo1_Brazo2\Rot = [Brazo1_Brazo2\CosTh, -Brazo1_Brazo2\sinTh, 0;

Brazo1_Brazo2\sinTh, Brazo1_Brazo2\CosTh, 0;

0, 0, 1];

Brazo1_Brazo2\Pos = 0;

Brazo1_Brazo2\RelH = homogeneous (Brazo1_Brazo2\Rot, Brazo1_Brazo2\Pos);

Brazo1_Brazo2\AbsH = multiplyH (Brazo1_Brazo2\ConnectionPoint1\AbsH,
Brazo1_Brazo2\RelH);

Brazo1_Brazo2\ConnectionPoint2\AbsH = Brazo1_Brazo2\AbsH;

Brazo2\AbsH = multiplyH (Brazo1_Brazo2\ConnectionPoint2\AbsH, Brazo2\RelH);

Brazo2\mass_x_I = Brazo2\mass * Brazo2\I;

Brazo2_Carga\ConnectionPoint1\AbsH = multiplyH (Brazo2\AbsH,
Brazo2_Carga\ConnectionPoint1\RelH);

Brazo2_Carga\pos = int (Brazo2_Carga\posDot) + Brazo2_Carga\posInitial;

Brazo2_Carga\Rot = eye (3);

Brazo2_Carga\Pos = [0;

0;

Brazo2_Carga\pos];

Brazo2_Carga\RelH = homogeneous (Brazo2_Carga\Rot, Brazo2_Carga\Pos);

Brazo2_Carga\AbsH = multiplyH (Brazo2_Carga\ConnectionPoint1\AbsH, Brazo2_Carga\RelH);

Brazo2_Carga\ConnectionPoint2\AbsH = Brazo2_Carga\AbsH;

Carga\AbsH = multiplyH (Brazo2_Carga\ConnectionPoint2\AbsH, Carga\RelH);

Carga\mass_x_I = Carga\mass * Carga\I;

Brazo2_Carga\ConnectionPoint2\AbsHinverse = inverseH
(Brazo2_Carga\ConnectionPoint2\AbsH);

Brazo2_Carga\Wrench = [0, 0, 0, 0, 0, Brazo2_Carga\force] * Adjoint
(Brazo2_Carga\ConnectionPoint2\AbsHinverse);

Brazo1_Brazo2\ConnectionPoint2\AbsHinverse = inverseH
(Brazo1_Brazo2\ConnectionPoint2\AbsH);

Brazo1_Brazo2\Wrench = [0, 0, Brazo1_Brazo2\torque, 0, 0, 0] * Adjoint
(Brazo1_Brazo2\ConnectionPoint2\AbsHinverse);

Base_Brazo1\ConnectionPoint2\AbsHinverse = inverseH
(Base_Brazo1\ConnectionPoint2\AbsH);

Base_Brazo1\Wrench = [0, 0, Base_Brazo1\torque, 0, 0, 0] * Adjoint
(Base_Brazo1\ConnectionPoint2\AbsHinverse);

AdjX2[1:3] = Base_Brazo1\ConnectionPoint1\AbsH[1:3, 3];

AdjX2[4] = Base_Brazo1\ConnectionPoint1\AbsH[2, 4] *
Base_Brazo1\ConnectionPoint1\AbsH[3, 3] - Base_Brazo1\ConnectionPoint1\AbsH[3, 4] *
Base_Brazo1\ConnectionPoint1\AbsH[2, 3];

AdjX2[5] = Base_Brazo1\ConnectionPoint1\AbsH[3, 4] *
Base_Brazo1\ConnectionPoint1\AbsH[1, 3] - Base_Brazo1\ConnectionPoint1\AbsH[1, 4] *
Base_Brazo1\ConnectionPoint1\AbsH[3, 3];
```

```
AdjX2[6]        =        Base_Brazo1\ConnectionPoint1\AbsH[1,        4]        *
Base_Brazo1\ConnectionPoint1\AbsH[2, 3] - Base_Brazo1\ConnectionPoint1\AbsH[2, 4] *
Base_Brazo1\ConnectionPoint1\AbsH[1, 3];

AdjX2_T = transpose (AdjX2);

adjAdjX2 = adjoint (AdjX2);

AdjX3[1:3] = Brazo1_Brazo2\ConnectionPoint1\AbsH[1:3, 3];

AdjX3[4]        =        Brazo1_Brazo2\ConnectionPoint1\AbsH[2,        4]        *
Brazo1_Brazo2\ConnectionPoint1\AbsH[3, 3] - Brazo1_Brazo2\ConnectionPoint1\AbsH[3, 4] *
Brazo1_Brazo2\ConnectionPoint1\AbsH[2, 3];

AdjX3[5]        =        Brazo1_Brazo2\ConnectionPoint1\AbsH[3,        4]        *
Brazo1_Brazo2\ConnectionPoint1\AbsH[1, 3] - Brazo1_Brazo2\ConnectionPoint1\AbsH[1, 4] *
Brazo1_Brazo2\ConnectionPoint1\AbsH[3, 3];

AdjX3[6]        =        Brazo1_Brazo2\ConnectionPoint1\AbsH[1,        4]        *
Brazo1_Brazo2\ConnectionPoint1\AbsH[2, 3] - Brazo1_Brazo2\ConnectionPoint1\AbsH[2, 4] *
Brazo1_Brazo2\ConnectionPoint1\AbsH[1, 3];

AdjX3_T = transpose (AdjX3);

adjAdjX3 = adjoint (AdjX3);

AdjX4[1:3] = 0;

AdjX4[4:6] = Brazo2_Carga\ConnectionPoint1\AbsH[1:3, 3];

AdjX4_T = transpose (AdjX4);

adjAdjX4 = adjoint (AdjX4);

J2[1:6, 1] = AdjX2;

J3[1:6, 1] = AdjX2;

J3[1:6, 2] = AdjX3;

J4[1:6, 1] = AdjX2;

J4[1:6, 2] = AdjX3;

J4[1:6, 3] = AdjX4;

T1_00 = J1 * v;

T1_00T = transpose (T1_00);

T2_00 = J2 * v;

T2_00T = transpose (T2_00);

T3_00 = J3 * v;

T3_00T = transpose (T3_00);

T4_00 = J4 * v;

T4_00T = transpose (T4_00);

AdjInv1 = Adjoint (inverseH (Base\AbsH));

M1 = transpose (AdjInv1) * (Base\mass_x_I * AdjInv1);

TM1 = T1_00T * M1;

AdjInv2 = Adjoint (inverseH (Brazo1\AbsH));

M2 = transpose (AdjInv2) * (Brazo1\mass_x_I * AdjInv2);

TM2 = T2_00T * M2;
```

```
AdjInv3 = Adjoint (inverseH (Brazo2\AbsH));

M3 = transpose (AdjInv3) * (Brazo2\mass_x_I * AdjInv3);

TM3 = T3_00T * M3;

AdjInv4 = Adjoint (inverseH (Carga\AbsH));

M4 = transpose (AdjInv4) * (Carga\mass_x_I * AdjInv4);

TM4 = T4_00T * M4;

M2_2 = AdjX2_T * (((M2 + M3) + M4) * AdjX2);

M3_2 = AdjX3_T * ((M3 + M4) * AdjX2);

M3_3 = AdjX3_T * ((M3 + M4) * AdjX3);

M4_2 = AdjX4_T * (M4 * AdjX2);

M4_3 = AdjX4_T * (M4 * AdjX3);

M4_4 = AdjX4_T * (M4 * AdjX4);

MassMatrix[1, 1] = M2_2;

MassMatrix[1, 2] = M3_2;

MassMatrix[1, 3] = M4_2;

MassMatrix[2, 1] = M3_2;

MassMatrix[2, 2] = M3_3;

MassMatrix[2, 3] = M4_3;

MassMatrix[3, 1] = M4_2;

MassMatrix[3, 2] = M4_3;

MassMatrix[3, 3] = M4_4;

wrenchGrav2 = Brazo1\mass * [gravity[3] * Brazo1\AbsH[2, 4] - gravity[2] * Brazo1\AbsH[3,
4], gravity[1] * Brazo1\AbsH[3, 4] - gravity[3] * Brazo1\AbsH[1, 4], gravity[2] *
Brazo1\AbsH[1, 4] - gravity[1] * Brazo1\AbsH[2, 4], gravity[1], gravity[2], gravity[3]];

wrenchGrav3 = Brazo2\mass * [gravity[3] * Brazo2\AbsH[2, 4] - gravity[2] * Brazo2\AbsH[3,
4], gravity[1] * Brazo2\AbsH[3, 4] - gravity[3] * Brazo2\AbsH[1, 4], gravity[2] *
Brazo2\AbsH[1, 4] - gravity[1] * Brazo2\AbsH[2, 4], gravity[1], gravity[2], gravity[3]];

wrenchGrav4 = Carga\mass * [gravity[3] * Carga\AbsH[2, 4] - gravity[2] * Carga\AbsH[3, 4],
gravity[1] * Carga\AbsH[3, 4] - gravity[3] * Carga\AbsH[1, 4], gravity[2] * Carga\AbsH[1, 4]
- gravity[1] * Carga\AbsH[2, 4], gravity[1], gravity[2], gravity[3]];

PDotBodies[1] = -(((TM2 + TM3) + TM4) * (adjAdjX2 * T2_00));

PDotBodies[2] = -((TM3 + TM4) * (adjAdjX3 * T3_00));

PDotBodies[3] = -(TM4 * (adjAdjX4 * T4_00));

PDot = PDotBodies + transpose (((((wrenchGrav2 * J2 + wrenchGrav3 * J3) + wrenchGrav4 *
J4) + Base_Brazo1\Wrench * (J2 - J1)) + Brazo1_Brazo2\Wrench * (J3 - J2)) +
Brazo2_Carga\Wrench * (J4 - J3));

P = int (PDot) + PDotInitial;

v = linsolve (MassMatrix, P, 'cholesky');

Base_Brazo1\thetaDot = v[1];

Brazo1_Brazo2\thetaDot = v[2];

Brazo2_Carga\posDot = v[3];
```

```
Base_Brazo1\torque = Base_Brazo1.e;
Base_Brazo1.f = Base_Brazo1\thetaDot;
Brazo1_Brazo2\torque = Brazo1_Brazo2.e;
Brazo1_Brazo2.f = Brazo1_Brazo2\thetaDot;
Brazo2_Carga\force = Brazo2_Carga.e;
Brazo2_Carga.f = Brazo2_Carga\posDot;
```

Código generado de mecánica 3D a 20-sim del doble péndulo.

```
variables
real hidden World\AbsH[4,4];
real hidden World\mass_x_I[6,6];
real hidden BaseWeld\ConnectionPoint1\AbsH[4,4];
real hidden BaseWeld\AbsH[4,4];
real hidden BaseWeld\ConnectionPoint2\AbsH[4,4];
real hidden Base\AbsH[4,4];
real hidden Base\mass_x_I[6,6];
real hidden Base_Brazo1\ConnectionPoint1\AbsH[4,4];
real hidden Base_Brazo1\AbsH[4,4];
real hidden Base_Brazo1\RelH[4,4];
real Base_Brazo1\theta;
real Base_Brazo1\thetaDot;
real Base_Brazo1\sinTh;
real Base_Brazo1\CosTh;
real Base_Brazo1\Rot[3,3];
real Base_Brazo1\Pos[3];
real hidden Base_Brazo1\ConnectionPoint2\AbsH[4,4];
real hidden Brazo1\AbsH[4,4];
real hidden Brazo1\mass_x_I[6,6];
real hidden RotationJointX1\ConnectionPoint1\AbsH[4,4];
real hidden RotationJointX1\AbsH[4,4];
real hidden RotationJointX1\RelH[4,4];
real RotationJointX1\theta;
real RotationJointX1\thetaDot;
real RotationJointX1\sinTh;
real RotationJointX1\CosTh;
real RotationJointX1\Rot[3,3];
real RotationJointX1\Pos[3];
real hidden RotationJointX1\ConnectionPoint2\AbsH[4,4];
real hidden Body2\AbsH[4,4];
real hidden Body2\mass_x_I[6,6];
real RotationJointX1\Wrench[1,6];
real RotationJointX1\torque;
real RotationJointX1\ConnectionPoint2\AbsHinverse[4,4];
```

```
real Base_Brazo1\Wrench[1,6];
real Base_Brazo1\torque;
real Base_Brazo1\ConnectionPoint2\AbsHinverse[4,4];
real v[2];
real P[2];
real PDot[2];
real PDotBodies[2];
real hidden AdjX2[6];
real hidden AdjX2_T[1,6];
real hidden adjAdjX2[6,6];
real hidden AdjX3[6];
real hidden AdjX3_T[1,6];
real hidden adjAdjX3[6,6];
real hidden J1[6,2];
real hidden J2[6,2];
real hidden J3[6,2];
real hidden T1_00[6];
real hidden T1_00T[1,6];
real hidden T2_00[6];
real hidden T2_00T[1,6];
real hidden T3_00[6];
real hidden T3_00T[1,6];
real hidden M1[6,6];
real hidden AdjInv1[6,6];
real hidden TM1[1,6];
real hidden M2[6,6];
real hidden AdjInv2[6,6];
real hidden TM2[1,6];
real hidden M3[6,6];
real hidden AdjInv3[6,6];
real hidden TM3[1,6];
real MassMatrix[2,2];
real hidden M2_2;
real hidden M3_2;
real hidden M3_3;
real hidden wrenchGrav2[1,6];
real hidden wrenchGrav3[1,6];
parameters
```

```
real hidden World\H[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden World\I[6,6] = [
0.1, 0, 0, 0, 0, 0;
0, 0.1, 0, 0, 0, 0;
0, 0, 0.1, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real World\mass = 1;
real hidden BaseWeld\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden Base\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden Base\I[6,6] = [
0.1, 0, 0, 0, 0, 0;
0, 0.1, 0, 0, 0, 0;
0, 0, 0.1, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real Base\mass = 1;
real hidden Base_Brazo1\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 0;
0, 0, 1, 0;
0, 0, 0, 1];
real Base_Brazo1\thetaInitial = 0;
real hidden Brazo1\RelH[4,4] = [
```

```
1, 0, 0, 0;
0, 1, 0, 2;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden Brazo1\I[6,6] = [
0.1, 0, 0, 0, 0, 0;
0, 0.1, 0, 0, 0, 0;
0, 0, 0.1, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real Brazo1\mass = 1;
real hidden RotationJointX1\ConnectionPoint1\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 2;
0, 0, 1, 0;
0, 0, 0, 1];
real RotationJointX1\thetaInitial = 0;
real hidden Body2\RelH[4,4] = [
1, 0, 0, 0;
0, 1, 0, 2;
0, 0, 1, 0;
0, 0, 0, 1];
real hidden Body2\I[6,6] = [
0.1, 0, 0, 0, 0, 0;
0, 0.1, 0, 0, 0, 0;
0, 0, 0.1, 0, 0, 0;
0, 0, 0, 1, 0, 0;
0, 0, 0, 0, 1, 0;
0, 0, 0, 0, 0, 1];
real Body2\mass = 1;
real gravity[3] = [0; 0; -9.81];
real hidden PDotInitial[2] = [0; 0];
initialequations
J1 = 0;
J2 = 0;
J3 = 0;
MassMatrix = 0;
```

```
equations

World\AbsH = World\H;

World\mass_x_I = World\mass * World\I;

BaseWeld\ConnectionPoint1\AbsH = multiplyH (World\AbsH,
BaseWeld\ConnectionPoint1\RelH);

BaseWeld\AbsH = BaseWeld\ConnectionPoint1\AbsH;

BaseWeld\ConnectionPoint2\AbsH = BaseWeld\AbsH;

Base\AbsH = multiplyH (BaseWeld\ConnectionPoint2\AbsH, Base\RelH);

Base\mass_x_I = Base\mass * Base\I;

Base_Brazo1\ConnectionPoint1\AbsH = multiplyH (Base\AbsH,
Base_Brazo1\ConnectionPoint1\RelH);

Base_Brazo1\theta = int (Base_Brazo1\thetaDot) + Base_Brazo1\thetaInitial;

Base_Brazo1\sinTh = sin (Base_Brazo1\theta);

Base_Brazo1\CosTh = cos (Base_Brazo1\theta);

Base_Brazo1\Rot = [1, 0, 0;

0, Base_Brazo1\CosTh, -Base_Brazo1\sinTh;

0, Base_Brazo1\sinTh, Base_Brazo1\CosTh];

Base_Brazo1\Pos = 0;

Base_Brazo1\RelH = homogeneous (Base_Brazo1\Rot, Base_Brazo1\Pos);

Base_Brazo1\AbsH = multiplyH (Base_Brazo1\ConnectionPoint1\AbsH, Base_Brazo1\RelH);

Base_Brazo1\ConnectionPoint2\AbsH = Base_Brazo1\AbsH;

Brazo1\AbsH = multiplyH (Base_Brazo1\ConnectionPoint2\AbsH, Brazo1\RelH);

Brazo1\mass_x_I = Brazo1\mass * Brazo1\I;

RotationJointX1\ConnectionPoint1\AbsH = multiplyH (Brazo1\AbsH,
RotationJointX1\ConnectionPoint1\RelH);

RotationJointX1\theta = int (RotationJointX1\thetaDot) + RotationJointX1\thetaInitial;

RotationJointX1\sinTh = sin (RotationJointX1\theta);

RotationJointX1\CosTh = cos (RotationJointX1\theta);

RotationJointX1\Rot = [1, 0, 0;

0, RotationJointX1\CosTh, -RotationJointX1\sinTh;

0, RotationJointX1\sinTh, RotationJointX1\CosTh];

RotationJointX1\Pos = 0;

RotationJointX1\RelH = homogeneous (RotationJointX1\Rot, RotationJointX1\Pos);

RotationJointX1\AbsH = multiplyH (RotationJointX1\ConnectionPoint1\AbsH,
RotationJointX1\RelH);

RotationJointX1\ConnectionPoint2\AbsH = RotationJointX1\AbsH;

Body2\AbsH = multiplyH (RotationJointX1\ConnectionPoint2\AbsH, Body2\RelH);

Body2\mass_x_I = Body2\mass * Body2\I;

RotationJointX1\ConnectionPoint2\AbsHinverse = inverseH
(RotationJointX1\ConnectionPoint2\AbsH);
```

```
RotationJointX1\Wrench = [RotationJointX1\torque, 0, 0, 0, 0, 0] * Adjoint (RotationJointX1\ConnectionPoint2\AbsHinverse);

Base_Brazo1\ConnectionPoint2\AbsHinverse = inverseH (Base_Brazo1\ConnectionPoint2\AbsH);

Base_Brazo1\Wrench = [Base_Brazo1\torque, 0, 0, 0, 0, 0] * Adjoint (Base_Brazo1\ConnectionPoint2\AbsHinverse);

AdjX2[1:3] = Base_Brazo1\ConnectionPoint1\AbsH[1:3, 1];

AdjX2[4] = Base_Brazo1\ConnectionPoint1\AbsH[2, 4] * Base_Brazo1\ConnectionPoint1\AbsH[3, 1] - Base_Brazo1\ConnectionPoint1\AbsH[3, 4] * Base_Brazo1\ConnectionPoint1\AbsH[2, 1];

AdjX2[5] = Base_Brazo1\ConnectionPoint1\AbsH[3, 4] * Base_Brazo1\ConnectionPoint1\AbsH[1, 1] - Base_Brazo1\ConnectionPoint1\AbsH[1, 4] * Base_Brazo1\ConnectionPoint1\AbsH[3, 1];

AdjX2[6] = Base_Brazo1\ConnectionPoint1\AbsH[1, 4] * Base_Brazo1\ConnectionPoint1\AbsH[2, 1] - Base_Brazo1\ConnectionPoint1\AbsH[2, 4] * Base_Brazo1\ConnectionPoint1\AbsH[1, 1];

AdjX2_T = transpose (AdjX2);

adjAdjX2 = adjoint (AdjX2);

AdjX3[1:3] = RotationJointX1\ConnectionPoint1\AbsH[1:3, 1];

AdjX3[4] = RotationJointX1\ConnectionPoint1\AbsH[2, 4] * RotationJointX1\ConnectionPoint1\AbsH[3, 1] - RotationJointX1\ConnectionPoint1\AbsH[3, 4] * RotationJointX1\ConnectionPoint1\AbsH[2, 1];

AdjX3[5] = RotationJointX1\ConnectionPoint1\AbsH[3, 4] * RotationJointX1\ConnectionPoint1\AbsH[1, 1] - RotationJointX1\ConnectionPoint1\AbsH[1, 4] * RotationJointX1\ConnectionPoint1\AbsH[3, 1];

AdjX3[6] = RotationJointX1\ConnectionPoint1\AbsH[1, 4] * RotationJointX1\ConnectionPoint1\AbsH[2, 1] - RotationJointX1\ConnectionPoint1\AbsH[2, 4] * RotationJointX1\ConnectionPoint1\AbsH[1, 1];

AdjX3_T = transpose (AdjX3);

adjAdjX3 = adjoint (AdjX3);

J2[1:6, 1] = AdjX2;

J3[1:6, 1] = AdjX2;

J3[1:6, 2] = AdjX3;

T1_00 = J1 * v;

T1_00T = transpose (T1_00);

T2_00 = J2 * v;

T2_00T = transpose (T2_00);

T3_00 = J3 * v;

T3_00T = transpose (T3_00);

AdjInv1 = Adjoint (inverseH (Base\AbsH));

M1 = transpose (AdjInv1) * (Base\mass_x_I * AdjInv1);

TM1 = T1_00T * M1;

AdjInv2 = Adjoint (inverseH (Brazo1\AbsH));

M2 = transpose (AdjInv2) * (Brazo1\mass_x_I * AdjInv2);

TM2 = T2_00T * M2;
```

```
AdjInv3 = Adjoint (inverseH (Body2\AbsH));

M3 = transpose (AdjInv3) * (Body2\mass_x_I * AdjInv3);

TM3 = T3_00T * M3;

M2_2 = AdjX2_T * ((M2 + M3) * AdjX2);

M3_2 = AdjX3_T * (M3 * AdjX2);

M3_3 = AdjX3_T * (M3 * AdjX3);

MassMatrix[1, 1] = M2_2;

MassMatrix[1, 2] = M3_2;

MassMatrix[2, 1] = M3_2;

MassMatrix[2, 2] = M3_3;

wrenchGrav2 = Brazo1\mass * [gravity[3] * Brazo1\AbsH[2, 4] - gravity[2] * Brazo1\AbsH[3,
4], gravity[1] * Brazo1\AbsH[3, 4] - gravity[3] * Brazo1\AbsH[1, 4], gravity[2] *
Brazo1\AbsH[1, 4] - gravity[1] * Brazo1\AbsH[2, 4], gravity[1], gravity[2], gravity[3]];

wrenchGrav3 = Body2\mass * [gravity[3] * Body2\AbsH[2, 4] - gravity[2] * Body2\AbsH[3, 4],
gravity[1] * Body2\AbsH[3, 4] - gravity[3] * Body2\AbsH[1, 4], gravity[2] * Body2\AbsH[1,
4] - gravity[1] * Body2\AbsH[2, 4], gravity[1], gravity[2], gravity[3]];

PDotBodies[1] = -((TM2 + TM3) * (adjAdjX2 * T2_00));

PDotBodies[2] = -(TM3 * (adjAdjX3 * T3_00));

PDot = PDotBodies + transpose (((wrenchGrav2 * J2 + wrenchGrav3 * J3) +
Base_Brazo1\Wrench * (J2 - J1)) + RotationJointX1\Wrench * (J3 - J2));

P = int (PDot) + PDotInitial;

v = linsolve (MassMatrix, P, 'cholesky');

Base_Brazo1\thetaDot = v[1];

RotationJointX1\thetaDot = v[2];

Base_Brazo1\torque = Base_Brazo1.e;

Base_Brazo1.f = Base_Brazo1\thetaDot;

RotationJointX1\torque = RotationJointX1.e;

RotationJointX1.f = RotationJointX1\thetaDot;
```