

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**SIMULACIÓN DE UN ROBOT CON 6 GRADOS DE LIBERTAD
UTILIZANDO TOOLBOX ROBOTICS DEL SOFTWARE
MATLAB.**

**PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE
TECNÓLOGO EN ELECTROMEÁNICA**

JAVIER SEBASTIÁN MARKLEY ERAZO

jmark98@hotmail.com
javier.markley@epn.edu.ec

DIRECTORA:

ING. CATALINA ELIZABETH ARMAS FREIRE

elizabeth.armas@epn.edu.ec

CODIRECTOR:

ING. CARLOS ORLANDO ROMO HERRERA

carlos.romo@epn.edu.ec

Quito, octubre 2020

DECLARACIÓN

Yo Javier Sebastián Markley Erazo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional, y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Firma:

A handwritten signature in blue ink that reads "Javier Markley". The signature is written over a horizontal line.

Javier Sebastián Markley Erazo

C.I.:1751284975

Teléfono: 0982313733

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por el señor Javier Sebastián Markley Erazo, bajo nuestra supervisión.



Ing. Catalina Elizabeth Armas Freire

DIRECTORA DE PROYECTO

Ing. Carlos Orlando Romo Herrera

CODIRECTOR DE PROYECTO

AGRADECIMIENTOS

Mis más sinceros agradecimientos a mis padres Fausto Javier Markley Galárraga y Norma Susana Erazo Insuasti, que como padres han sido un ejemplo a seguir, una imagen de individuos centrados, trabajadores, perseverantes, fieles, humildes, características que considero elementales para desarrollarse como una excelente persona y buen profesional. Gracias a su apoyo incondicional en el transcurso de mi vida y su compromiso de criarme como una persona de bien, han permitido que todo esto sea posible.

Agradezco a mi hermano menor Carlos Jorge Markley Erazo, que gracias a su apoyo en los momentos difíciles en mi vida personal y en el transcurso de mi formación profesional, he encontrado el soporte, compañía y estabilidad sentimental en las diferentes etapas de mi vida.

Mi gratitud hacia mi abuela María Inés Galárraga Rodríguez, que ha estado presente en todos los sucesos importantes de mi familia y ha brindado un apoyo incondicional en todos los aspectos necesarios para seguir adelante.

Finalmente debo mi gratitud a todos los ingenieros a cargo de las materias correspondientes a mi formación profesional, como a mi directora Catalina Elizabeth Armas Freire, que considero excelentes personas y profesionales a favor de impartir conocimientos y la dura labor de enseñanza dentro de los diferentes campos técnicos y profesionales.

DEDICATORIA

Todo el esfuerzo empleado en mi formación profesional, la instrucción obtenida en el campo técnico y el conocimiento empleado en el presente trabajo va dirigido a cada miembro de mi núcleo familiar. Mis padres, mi hermano y mi abuela, considero que son las personas merecedoras de todo mi trabajo, por el mismo hecho que son los actores principales e incentivo, para poder haber continuado mis estudios y cada día tratar de ser mejor persona y profesional. Cada uno de ellos me enseñó la importancia de seguir adelante a pesar de las dificultades, no dejar las cosas a medias, ser constante y realizar las cosas lo mejor que uno pueda. Gracias a todos y cada uno de ellos, he tenido la posibilidad de poder realizar el presente trabajo.

ÍNDICE

ÍNDICE DE TABLAS	VII
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE ANEXOS	X
RESUMEN	XI
ABSTRACT	XII
1. INTRODUCCIÓN	1
1.1. Planteamiento del problema.....	4
1.2. Justificación	5
1.3. Objetivos.....	6
2. METODOLOGÍA	7
2.1. Descripción de la metodología.....	7
3. RESULTADOS Y DISCUSIÓN	9
3.1. Software Matlab.....	9
3.2. Toolbox Robotics de Matlab.....	35
3.3. Algoritmo de Denavit-Hartenberg.....	38
3.4. Interfaz animada de usuario.....	45
3.5. Pruebas de funcionamiento.....	47
3.6. Manual de usuario	53
4. CONCLUSIONES Y RECOMENDACIONES	58
4.1. Conclusiones	58
4.2. Recomendaciones.....	59
5. BIBLIOGRAFÍA	60
6. ANEXOS	62

ÍNDICE DE TABLAS

Tabla 3-1: Funciones matemáticas elementales.....	24
Tabla 3-2: Funciones de redondeo.....	25
Tabla 3-3: Funciones matemáticas discretas.....	25
Tabla 3-4: Funciones trigonométricas (radianes).	26
Tabla 3-5: Funciones trigonométricas (grados).	26
Tabla 3-6: Funciones trigonométricas inversas (radianes).....	26
Tabla 3-7: Funciones trigonométricas inversas (grados).....	27
Tabla 3-8: Funciones trigonométricas hiperbólicas (radianes).....	27
Tabla 3-9: Estilo de línea para gráficas.....	31
Tabla 3-10: Colores para gráficas.	32
Tabla 3-11: Estilo de marca - Gráficas.....	32
Tabla 3-12: Comandos para fuentes, líneas y axes.	33
Tabla 3-13: Comandos para gráficas.	34
Tabla 3-14: Prueba de matrices Denavit - Hartenberg.	47
Tabla 3-15: Prueba de posicionamiento - Prueba 1.	48
Tabla 3-16: Prueba de posicionamiento - Prueba 2.	48
Tabla 3-17: Prueba de posicionamiento - Prueba 3.	49
Tabla 3-18: Prueba de posicionamiento - Prueba 4.	49
Tabla 3-19: Prueba de coordenadas efector final.	51
Tabla 3-20: Prueba menú slides.....	51
Tabla 3-21: Validación simulación.....	52

ÍNDICE DE FIGURAS

Figura 1-1: Cantidad de ejes de coordenadas.....	3
Figura 3-1: Espacio general de trabajo.....	9
Figura 3-2: Historial de comandos - Command Window.....	11
Figura 3-3: Opción New.....	14
Figura 3-4: Creación de un script.....	14
Figura 3-5: Editor de script.....	15
Figura 3-6: Pestañas Script.....	15
Figura 3-7: Ejemplo script.....	16
Figura 3-8: Opción Run.....	16
Figura 3-9: Guardar script.....	17
Figura 3-10: Change Folder - Script.....	17
Figura 3-11: Ejecución Script.....	18
Figura 3-12: Vector fila.....	19
Figura 3-13: Vector columna.....	19
Figura 3-14: Vector con intervalo regular (Incremento y Decremento en uno).....	20
Figura 3-15: Índice vectores.....	20
Figura 3-16: Índice vectores - Comando end.....	21
Figura 3-17: Construcción matriz.....	21
Figura 3-18: Extraer valor específico.....	22
Figura 3-19: Extraer varios datos.....	22
Figura 3-20: Notación científica.....	23
Figura 3-21: Formatos de despliegue.....	24
Figura 3-22: Datos para gráficas.....	28
Figura 3-23: Comando plot – Gráfica.....	28
Figura 3-24: Comando title.....	29
Figura 3-25: Comando xlabel.....	29
Figura 3-26: Comando ylabel.....	30
Figura 3-27: Comando grid on.....	30
Figura 3-28: Comando legend.....	31
Figura 3-29: Escalamiento gráficas.....	33
Figura 3-30: Anotaciones en gráficas.....	34
Figura 3-31: TOOLBOXES - ROBOTICS TOOLBOX.....	36
Figura 3-32: Documentación comandos Matlab - Toolbox Robotics.....	36
Figura 3-33: Ángulos azimuth y elevation.....	37

Figura 3-34: Ángulo ϕ - Eje z, Ángulo θ - Eje x. Ángulo Ψ - Eje z.....	38
Figura 3-35: Ángulos ROLL, PITCH & YAW - Ejes x, y, z.....	40
Figura 3-36: Ángulos ROLL, PITCH & YAW - Interfaz.	40
Figura 3-37: Parámetros Denavit-Hartenberg.	41
Figura 3-38: Denotación Denavit Hartenberg Estándar vs Modificado.	42
Figura 3-39: Distribución eslabones y articulaciones.	45
Figura 3-40: Prueba 1 y 2 de posicionamiento.	50
Figura 3-41: Prueba 3 y 4 de posicionamiento.	50
Figura 3-42: Valores resultantes - validación simulación.	53
Figura 3-43: Diagrama de Flujo.	54
Figura 3-44: Ventana nuevo script - Manual de usuario.	54
Figura 3-45 Pestaña nuevo script en curso - Manual de usuario.....	55
Figura 3-46 Fragmento del código - Etapa de programación.....	55
Figura 3-47: Inicio para correr el programa.	55
Figura 3-48: Guardar el algoritmo.	56
Figura 3-49: Ejemplificación - Datos de entrada.....	56
Figura 3-50: Figuras desplegadas - Ejecución de código.	57
Figura 3-51: Verificación parámetros Denavit Hartenberg.	57

ÍNDICE DE ANEXOS

ANEXO 1: COMPROBACIÓN ALGORITMO DENAVIT-HARTENBERG (STANDARD)	62
ANEXO 2: COMPROBACIÓN ALGORITMO DENAVIT-HARTENBERG (MODIFICADO)	65
ANEXO 3: CÓDIGO ROBOT 6 GRADOS DE LIBERTAD EN MATLAB (APLICACIÓN DE TRAYECTORIAS)	68
ANEXO 4: CÓDIGO PARA LA SOLICITUD DE DATOS EN COMMAND WINDOW...	75
ANEXO 5: CÓDIGO ROBOT 6 GRADOS DE LIBERTAD EN MATLAB (MATRICES DENAVIT-HARTENBERG - STANDARD).....	77
ANEXO 6: INSTRUCCIONES PARA INSTALCIÓN DE MATLAB Y LIBRERÍA TOOLBOX ROBOTICS	87

RESUMEN

El presente trabajo tiene la finalidad de introducir a los estudiantes de Tecnología en Electromecánica de la Escuela de Formación de Tecnólogos (ESFOT) en el campo de la robótica, mediante la simulación de un robot con 6 grados de libertad, utilizando el Toolbox Robotics del software Matlab.

El proyecto inicia mostrando una breve reseña de la historia de la robótica, palabras clave y sus conceptos generales para el entendimiento básico de este campo. A partir de esto, se inicia con el uso de la herramienta Matlab, la distribución de su espacio de trabajo y el detalle de todos los comandos necesarios para el entendimiento y ejecución de los procesos requeridos para la implementación del robot. Se determina así mismo, los comandos requeridos de la librería Toolbox Robotics y la explicación de los parámetros de Denavit-Hartenberg.

Con los criterios revisados a lo largo del trabajo, se elabora un manual, para que el estudiante sea capaz de programar su propio robot con todos los parámetros designados, como por ejemplo: la asignación de variables, designación de las matrices de transformación, gráficas estáticas y animadas del robot, dimensionamiento de los límites de las figuras y verificación de los indicadores finales del efector final.

Por ser un campo de estudio muy extenso, este proyecto puede ser complementado con un módulo práctico, que permita a los estudiantes aplicar de manera directa todos los cálculos que se realicen a través de las matrices de transformación y de su respectiva simulación.

De esta forma se concluye con la demostración del correcto funcionamiento de la simulación realizada con la validación y obtención de los mismos valores de las coordenadas del efector final, tanto en la interfaz propuesta como en la matriz resultante final.

Palabras clave: Robótica, robot, simulación, Toolbox Robotics, Matlab, Denavit-Hartenberg, matrices de transformación, efector final.

ABSTRACT

The present work has the purpose of introducing the students of Technology in Electromechanics of the School of Training of Technologists (ESFOT) in the field of robotics, by simulating a robot with 6 degrees of freedom, using the Toolbox Robotics of the software Matlab.

The project begins by showing a brief overview of the history of robotics, keywords and its general concepts for the basic understanding of this field. From this, it will start with the use of the Matlab tool, the distribution of its workspace and the detail of all the necessary commands for the understanding and execution of the processes required for the implementation of the robot. It will also determine the required commands from the Toolbox Robotics library and the explanation of the Denavit-Hartenberg parameters.

With the criteria reviewed throughout the work, a manual will be developed, so that the student is able to program their own robot with all the designated parameters, such as: the assignment of variables, designation of the transformation matrices, static graphs and animations of the robot, dimensioning of the limits of the figures and verification of the final indicators of the end effector.

As it is a very extensive field of study, this project can be complemented with a practical module that allows students to directly apply all the calculations carried out through the transformation matrices and their respective simulation.

In this way, we conclude with the demonstration of the correct operation of the simulation carried out with the validation and obtaining of the same values of the coordinates of the end effector, both in the proposed interface and in the final resulting matrix.

Keywords: Robotics, robot, simulation, Toolbox Robotics, Matlab, Denavit-Hartenberg, transformation matrices, end effector.

1. INTRODUCCIÓN

Bases y antecedentes de la robótica

En los últimos años la ciencia y la tecnología han avanzado de manera exponencial con el fin de facilitar la vida del ser humano y ahorrar costos en el proceso. La construcción de máquinas que pueden imitar y realizar actividades del hombre han estado presente muchos años atrás desde antiguas civilizaciones que realizaban homenaje a sus dioses decorando sus estatuas y templos con mecanismos robóticos arcaicos hasta la actualidad donde se ve industrias automatizadas con un eje principal en la robótica, fundamentalmente para la fabricación de muchas piezas mecánicas, eléctricas, electrónicas dentro de los diferentes campos de la ingeniería así también en ámbitos de la medicina con procesos mucho más sofisticados, en la intervención del cuerpo humano con procedimientos inseguros que pueden poner en riesgo la vida del paciente, en disciplinas como la biotecnología y química ya que en muchos ámbitos de estudio se requiere una atmósfera libre de contaminación donde los robots son la herramienta perfecta para llevar acabo un sinnúmero de actividades dentro del manejo de sustancias químicas y bioquímicas, en campos de estudio como la exploración espacial, que hoy en día está en su auge con los hallazgos y proyectos que se tiene a futuro con las misiones que pretenden llevar mecanismos robóticos a explorar mundos desconocidos en busca de vida o del hallazgo del enigma más grande dentro de la comunidad astronómica “el origen del universo”.

¿Qué es la robótica?

En función de lo presentado en numerales anteriores el término fue apareciendo poco a poco mientras los avances iban incrementando, hoy en día ya se tiene un concepto más claro ya que es un campo muy estudiado dentro de varias especialidades. La robótica se define como una ciencia o rama de la tecnología que estudia el diseño y construcción de máquinas con la capacidad de realizar diversidad de actividades de forma exacta y precisa programadas por el ser humano, con la aplicación diferentes lenguajes de programación informáticos y/o de control. (Anónimo, Inteligencia Robótica, 2015)

¿Qué es un robot?

Este término también ha ido evolucionando de tal manera que en la actualidad se lo ha podido dar un concepto más claro según las prestaciones que ofrece en la industria y en otros sistemas menos robustos. Un robot es aquella máquina automática, reprogramable, polivalente (que puede desempeñar varias funciones), capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias a través de variables reprogramables, para la ejecución de tareas variadas. Puede ser diseñado en base a los requerimientos propuestos, posee una unidad de control que incluye un dispositivo de memoria donde irán alojadas las instrucciones programadas. (Anónimo, Inteligencia Robótica, 2015)

Conceptos importantes

Articulación

Gracias a las articulaciones se obtienen movimientos de desplazamiento, giro o una combinación de las dos, ya sea o no al mismo tiempo. (Anónimo, Robótica y generalidades, 2016)

Eslabón

Es un elemento que al enlazarse con otros permite la formación de cadenas, los eslabones deben cumplir con la finalidad de sujeción y sostén en sus extremos. (Anónimo, Robótica y generalidades, 2016)

Grados de libertad

El robot está compuesto por piezas rígidas que permiten su funcionamiento (eslabones), éstas están unidas entre sí por las denominadas articulaciones. El conjunto que se obtiene con varias uniones entre piezas y articulaciones se las puede denominar cadenas, esta cadena generalmente empieza con un soporte fijo y termina en su extremo contrario que es móvil y libre, este extremo puede ser capaz de alojar otra unión o simplemente es la parte final donde va la herramienta necesaria para interactuar con el proceso. Los grados de libertad están designados por la cantidad de articulaciones que este posee o por los ejes de coordenadas que los representan. (Anónimo, Robótica y generalidades, 2016)

Para el cálculo de los grados de libertad de un robot se dispone de la siguiente ecuación.

$$GDL = (N - 1)$$

Ecuación 1-1

Donde:

N = Número de ejes de coordenadas tomando en cuenta la base.

Para el presente proyecto se define un robot de 6 grados de libertad. A continuación, se muestra en la siguiente figura:

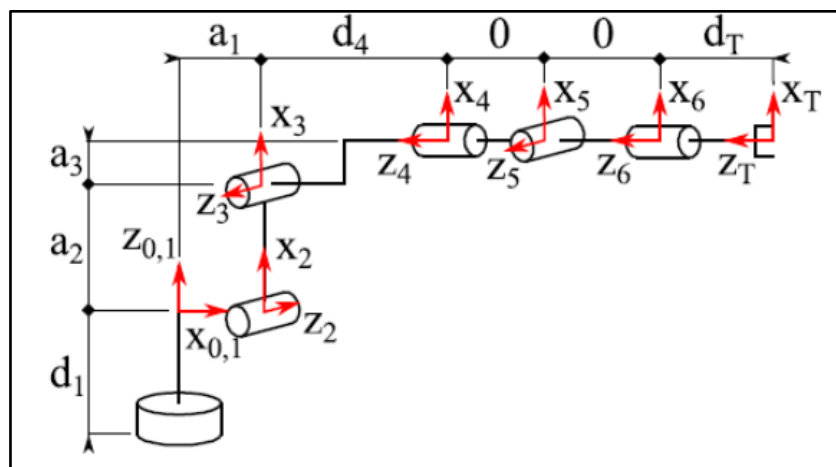


Figura 1-1: Cantidad de ejes de coordenadas.

(Muñoz, Rodríguez, SciELO, 2015)

Por tanto, el cálculo para los grados de libertad del robot de este trabajo se calcula de la siguiente manera:

$$GDL = (N - 1)$$

$$GDL = (7 - 1)$$

$$GDL = 6$$

1.1. Planteamiento del problema

Hoy en día la ciencia y la tecnología son los pilares fundamentales del desarrollo mundial, lo que ha permitido que el ser humano fabrique máquinas automatizadas, también conocidos como “robots”, para facilitar las actividades dentro del campo industrial, esto ha obligado a los profesionales técnicos a tener una visión diferente y multidisciplinaria de la forma en que se diseñan, construyen y gestionan los procesos en la actualidad, generando en ellos la necesidad y motivación a introducirse en el estudio de la robótica, debido al potencial e importancia que posee este campo dentro de los diferentes procesos industriales. (Huatuco, 2014)

El perfil profesional que requiere cumplir una persona formada en el área de electromecánica debe cubrir los conocimientos que le permitan vincularse en el campo de la industria, que actualmente está más desarrollada y robotizada. (Moreno Iveth, 2012)

En relación con el perfil laboral, expuesto por la Escuela de Formación de Tecnólogos (ESFOT) de la Escuela Politécnica Nacional (EPN) dice: “El profesional graduado en esta carrera tendrá un excelente nivel de desarrollo en competencias como: planificación, instalación, construcción, montaje, control, innovación y mantenimiento de equipo eléctrico y mecánico; controlar, supervisar y administrar recursos humanos y materiales de obras de este tipo”; por lo que un profesional de esta categoría, debe tener el conocimiento, la habilidad y haber desarrollado las capacidades técnicas, que le permitan desenvolverse en el campo industrial, esto incluye el conocimiento y manejo de las diferentes herramientas, que permiten el estudio y simulación de los elementos mecánicos, que forman parte de una industria robotizada, que actualmente está presente en casi cualquier proceso industrial; desde la industria agroalimentaria hasta la industria automovilística, petroquímica, etc. (Robotics, s.f.)

La malla de la carrera Tecnología en Electromecánica no posee materias en específico que aporten con el conocimiento en este campo, y en función de la importancia que este tema conlleva en el campo industrial, es indispensable escribir una guía que le permita al estudiante introducirse en el estudio de la robótica, pero no solo de los conceptos que abarca esta área de la mecánica, sino que esos conocimientos se vean consolidados con la ayuda de simulaciones. (Guzman, 2012)

1.2. Justificación

El conocimiento necesario para ser un buen profesional y desenvolverse correctamente dentro del ámbito industrial es más exigente en la actualidad, por esta razón los graduados que posean el perfil de tecnólogos de la Escuela Politécnica Nacional deberán ejercer cualquier actividad que se les solicite dentro de su campo, esto incluye un conocimiento básico de robótica, siendo este hoy en día la base fundamental de la fabricación en cadena de muchas empresas. (Carrera, 2015 - 2016)

Dentro del campo de la ingeniería existen varios softwares específicos, dependiendo de la aplicación que se desea, pero hay uno en particular que cumple con características que los expertos requieren para tener un proceso más eficiente y fiable, este software es Matlab. Las ventajas que posee Matlab sobre otros programas es que ocupa las matemáticas como lenguaje de programación, organizándolo como un libro académico, esto quiere decir, que al momento del análisis de datos y el procesamiento de gráficos e imágenes es ordenado y coherente. Matlab está diseñado para técnicos, ingenieros y científicos debido a que posee varias herramientas, funciones, librerías, toolboxes, aplicaciones que muestran el flujo de trabajo más metódico y estructurado en función del campo de estudio, como por ejemplo: Mantenimiento Predictivo y Preventivo, Máquinas Eléctricas, Sistemas de Abastecimiento de Agua, la industria Automovilística, etc. Por esta razón es considerado uno de los softwares más completos y utilizados a nivel industrial hoy en día. (Componentes, 2018)

El conocimiento de la robótica y todos los conceptos que abarca, actualmente son necesarios para el desarrollo y automatización de las industrias, por lo que si el estudiante tiene acceso a información, que le motive a complementar su conocimiento con temas actuales y útiles; como por ejemplo: un manual de robótica empleando el software Matlab, será una herramienta indispensable para la enseñanza, aprendizaje y aplicación de la robótica. (García, 2015)

1.3. Objetivos

1.3.1. Objetivo general

Simular un robot con 6 grados de libertad utilizando Toolbox Robotics del software Matlab.

1.3.2. Objetivos específicos

1. Describir el espacio de trabajo y herramientas básicas del software Matlab.
2. Investigar el manejo y aplicación del Toolbox Robotics de Matlab.
3. Aplicar el algoritmo Denavit Hartenberg en el estudio de mecanismos robóticos.
4. Crear una interfaz animada.
5. Realizar pruebas de funcionamiento.
6. Realizar un manual para introducir al estudiante en el estudio de la robótica.

2. METODOLOGÍA

2.1. Descripción de la metodología

Para llevar a cabo este proyecto se realizó una investigación aplicada y descriptiva (Universidad de, 2010 - 2011) (Corke, 2017) (Torriti, 2011).

Aplicada, porque en este trabajo se presentan las herramientas y mecanismos requeridos para que los estudiantes con escasos conocimientos en programación, uso del software Matlab y Robótica, puedan desempeñarse de manera correcta y adecuada dentro de este campo, según los parámetros básicos presentados en esta investigación.

Descriptiva, porque se trata de impartir los conocimientos básicos de forma minuciosa dentro del campo de la Robótica, por lo que en esta investigación se muestra de manera detallada cada paso a seguir para el uso adecuado del software Matlab, la librería Toolbox Robotics y los parámetros necesarios para la simulación de un robot de 6 grados de libertad.

Hoy en día existen varios softwares que permiten la simulación de robots para observar su comportamiento, entre estos se tiene: TagUI, que es un software libre que posibilita la programación rápida a través de comandos intuitivos, también se dispone del software Robocorp framework, su licencia es libre y dispone de un entorno para automatización de procesos robóticos basado en lenguaje Python y por último el software Takst, que es un programa de licencia abierta que está diseñado para la automatización de procesos robóticos, lo peculiar de esta aplicación es que no utiliza código.

A partir de la información brindada, en este trabajo de investigación se ha seleccionado el software Matlab, porque es considerado uno de los softwares matemáticos más completos existentes hoy en día en el mercado, los softwares vistos previamente utilizan comandos intuitivos, lenguajes de programación informáticos escritos como Python y no utilizan código, por esta razón se dificulta el análisis que comprende el comportamiento de un robot. Matlab es un laboratorio matemático, por lo que se adapta fácilmente al estudio de cualquier carrera de ciencias, ingeniería, o técnica. Para este proyecto Matlab se aplica para entender y emplear los parámetros Denavit-Hartenberg

en el estudio de los mecanismos robóticos, mediante una simulación del robot de 6 grados de libertad. El estudiante tendrá la capacidad de entender todos los temas elementales que comprenden el estudio de la Robótica con las explicaciones brindadas y el manual desarrollado.

3. RESULTADOS Y DISCUSIÓN

3.1. Software Matlab

Dentro de esta sección se explica brevemente la distribución y aplicación del software Matlab utilizado para el presente proyecto, si se desea un desarrollo más detallado de este tema, se recomienda seguir el “Manual Básico de Matlab” (Fernandez, s.f.).

3.1.1. Distribución espacio de trabajo Matlab

Dentro del espacio de Matlab, existen varias pestañas y herramientas, cada una de estas cumple con una función, la finalidad de este trabajo es mostrar las más importantes en el estudio de la robótica. A continuación, se detalla la distribución del espacio de trabajo en Matlab.

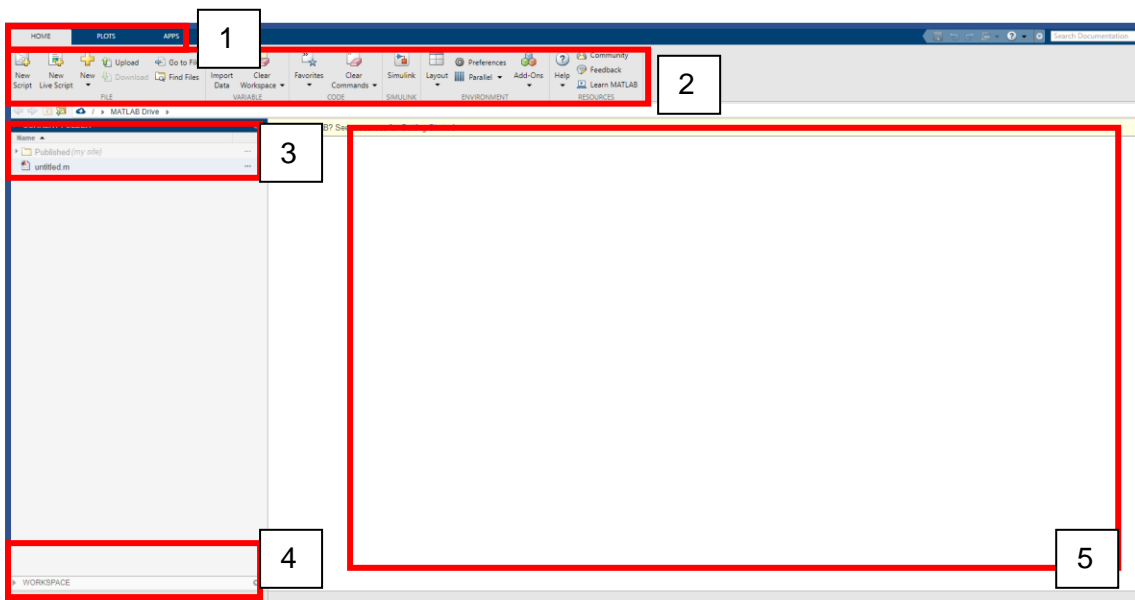


Figura 3-1: Espacio general de trabajo.

(Fuente: Propia)

Como se aprecia en la figura anterior, el espacio está dividido en:

1. Pestañas.
2. Barras de herramientas.
3. Current Folder.
4. Workspace.

5. Command Window.

Pestañas y barra de herramientas

En la barra de herramientas se tiene tres pestañas principales, home, plots y apps.

- **Home:** Se encuentran la mayoría de las herramientas que se utiliza dentro del programa.
- **Plots:** Se dispone de varias opciones para graficar las funciones que se coloquen en el software.
- **Apps:** En esta pestaña se encuentran las librerías que dispone Matlab y también opciones para visitar la página de apps y descargar alguna en específico.

Current folder

En esta ventana se encuentran los scripts que se han realizado y las funciones que se han propuesto, cada vez que se necesite abrir alguna en específico se tiene que hacerlo desde esta ventana, se debe buscar la carpeta en donde haya guardado dicho documento y ejecutarlo.

Workspace

Dentro de esta ventana se almacenan todas las variables que se vayan creando y el valor que éstas disponen. Las características de las funciones y operaciones matemáticas de Matlab están a disposición del usuario, haciendo click derecho en la barra de información dentro del "Workspace", se despliega todas las características que se dispone para observar las variables almacenadas.

Command window

El "Command Window" es la pestaña encargada de ejecutar todos los comandos, operaciones e instrucciones que el usuario programe.

3.1.2. Comandos generales – Command Window

Cuando se desea ingresar un comando o una instrucción desde el “Command Window”, se lo debe hacer siempre colocando el cursor a la derecha de la nomenclatura $fx \gg$, después de esto se empieza a escribir las instrucciones, comandos y operaciones que se desee. Todos los comandos que se ingresen desde el “Command Window”, tienen que ser escritos correctamente, caso contrario salta un error con un mensaje de aviso que indica que el comando está incorrecto.

Historial de comandos: Cuando se coloca el cursor como se indicó anteriormente y se presiona la flecha hacia arriba, aparecen todos los comandos que fueron utilizados, esto permite al usuario ahorrar tiempo al momento de la programación.

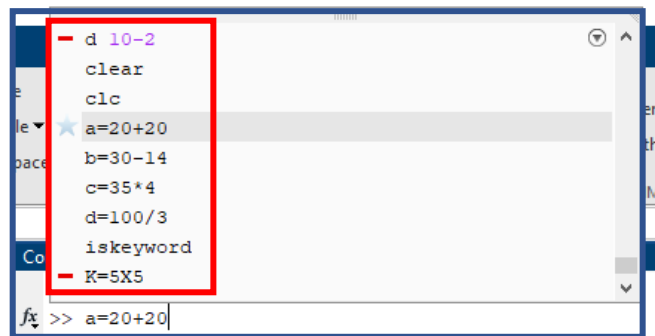


Figura 3-2: Historial de comandos - Command Window.

(Fuente: Propia)

A continuación, se detallan los comandos básicos y signos que pueden ser ejecutados en Command Window:

- **Comando *clear*:** Borra todas las variables del Workspace.
- **Comando *clc*:** Limpia toda la pantalla “Command Window” de MatLab.
- **Signo punto y coma (;):** Sirve para que no se muestre el valor que se le asigna a la variable después de ser ejecutada, pero si se guarda su valor en el Workspace.
- **Signo porcentaje (%):** Se lo utiliza para hacer comentarios, cualquier línea de código que esté precedida por este signo, Matlab no la va a ejecutar, este comando es muy útil para documentar e indicar que realiza el programa y cada línea de código.
- **Comando *iskeyword*:** Visualizar las palabras reservadas por el programa que

no se las puede utilizar para la creación de variables.

- **Comando *clear* variable:** Cumple la función del comando *clear*, pero este solo quita el valor de la variable indicada.
- **Comando *help* comando:** Despliega una ayuda, se debe colocar la palabra *help* y el comando que se desea conocer como utilizarlo o la estructura que tiene.

3.1.3. Nombrar variables en Matlab

Dentro del espacio de Matlab ya están definidas las reglas que se deben seguir para nombrar variables, no se permite utilizar cualquier asignación, si no se cumplen estos parámetros el programa va a detectar un error o mucho peor, va a haber interferencia dentro del programa que se cree, ya que ciertas palabras son designadas específicamente para comandos o funciones dentro del software y estas tampoco pueden ser utilizadas. A continuación, se detallan los parámetros que se deben tomar en cuenta para la creación de variables dentro del espacio de Matlab:

- **Todos los nombres deben comenzar con una letra:** Esta regla aplica tanto para variables y scripts (programas de Matlab), no importa si la letra es mayúscula o minúscula, no se permite números ni caracteres especiales para asignar un nombre.
- **Los nombres pueden tener cualquier longitud:** Esto aplica solo hasta Matlab 7, a partir de las siguientes versiones se permiten nombres de hasta 63 caracteres, los cuales son suficientes para la mayoría de las aplicaciones en el campo matemático.
- **Los caracteres permisibles son letras, números, y el guion bajo:** Esta regla sirve también para scripts y funciones que se vayan a crear, en caso de utilizar otros caracteres especiales, Matlab muestra un error.
- **Los nombres son sensibles a mayúsculas / minúsculas:** El software Matlab diferencia entre letras mayúsculas y minúsculas por lo que si se asigna una variable "a", esta va a ser diferente que la variable "A".
- **No se pueden utilizar las palabras reservadas de Matlab:** Existe palabras o comandos propias que utiliza el software y estas no pueden ser utilizadas para

crear variables.

- **Matlab permite reasignar nombres de función internos como nombres de variables:** Esto quiere decir que Matlab tiene funciones matemáticas como sin, cos, tan, etc.; estas son utilizadas para resolver ejercicios o así mismo, se las ocupa como variables asignándoles un valor, pero si se hace esto hay que tener cuidado, ya que se pueden perder datos si en un mismo programa se las utiliza de las dos maneras.

3.1.4. Script en Matlab

En el transcurso de este capítulo se ha mencionado mucho la palabra script, pero que es un script, dentro de Matlab un script es el programa más simple que hay, este se caracteriza por tener varias líneas de forma ordenada, secuencial y coherente de comandos con la finalidad de cumplir un determinado trabajo. Para el caso de este proyecto es suficiente utilizar un script para la programación de un robot dentro del espacio de Matlab.

Un script tiene la característica de que todo lo que se programe dentro permite ser guardado, todo lo contrario a cuando se colocaba los comandos básicos en “Command Window”, estos solo se ejecutaban, pero no forman parte de una estructura ordenada como un script.

Los archivos que se guardan como script, tienen una extensión “.m” y para nombrarlos deben cumplir los mismos 6 parámetros que se utilizan para la creación de variables.

Para poder crear un script dentro del espacio de Matlab se debe considerar los siguientes pasos:

1. Identificar la pestaña “Home”, la sección “File” la opción “New”.

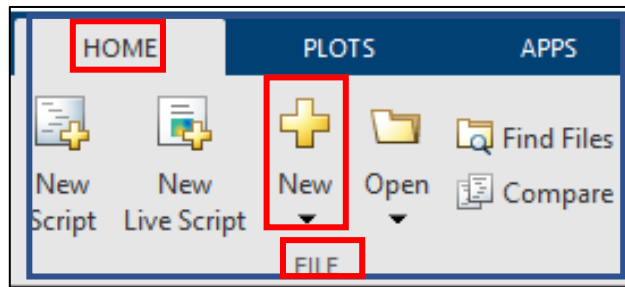


Figura 3-3: Opción New.

(Fuente: Propia)

2. Dentro de la opción de “New” se observa una gran cantidad de tipos de programa que se utilizan, en este caso se debe seleccionar un script.

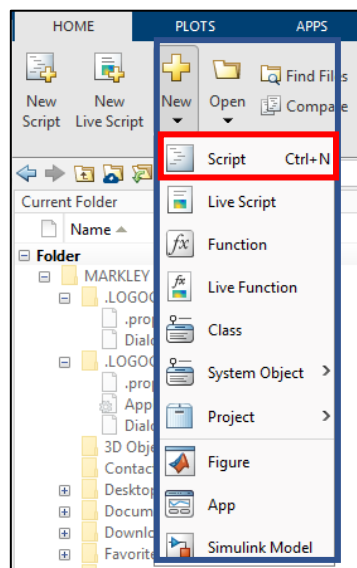


Figura 3-4: Creación de un script.

(Fuente: Propia)

3. Después de esto se tiene la ventana principal del editor de script donde se empieza a programar.

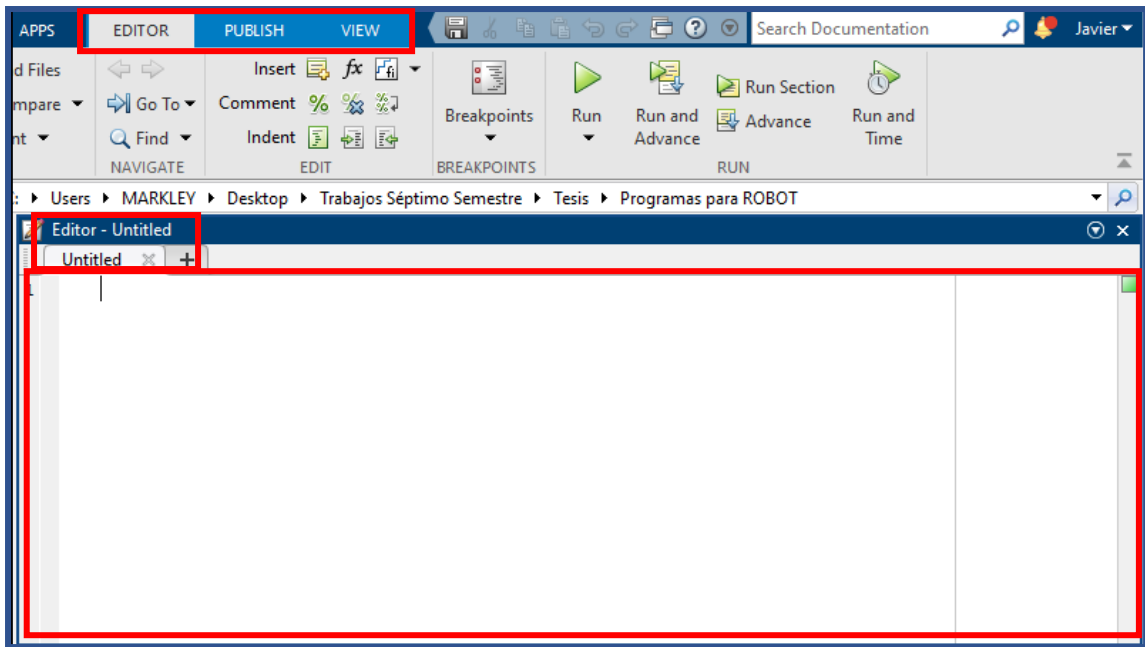


Figura 3-5: Editor de script.

(Fuente: Propia)

4. Gracias a que se creó un nuevo script, se agregan tres pestañas más donde se dispone más herramientas para trabajar.

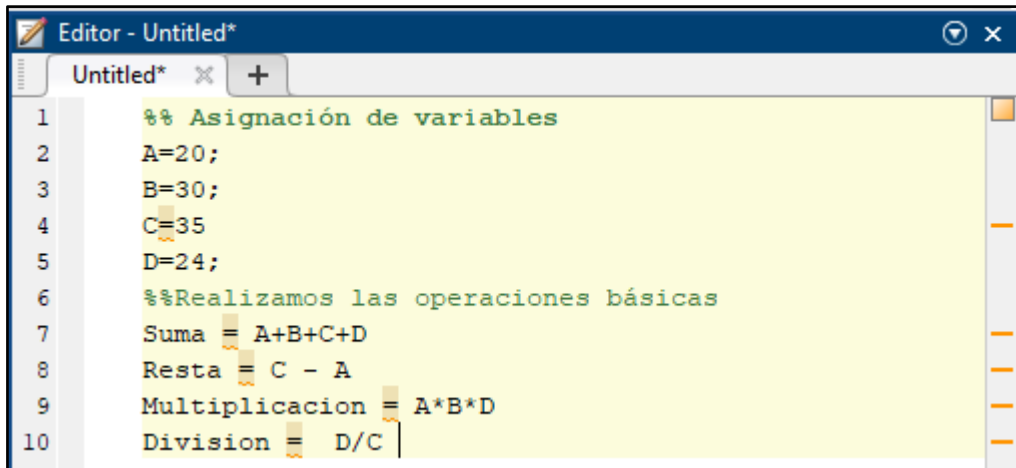


Figura 3-6: Pestañas Script.

(Fuente: Propia)

Antes de empezar a programar, tomar en cuenta que se debe guardar el archivo. A continuación, se muestra un ejemplo de script con funciones básicas y como guardarlo para ejecutarlo:

1. Primero se realiza un programa básico, en este se ha propuesto la asignación de algunas variables, para después aplicar las operaciones básicas, el programa todavía no está guardado.



```
1 %% Asignación de variables
2 A=20;
3 B=30;
4 C=35;
5 D=24;
6 %%Realizamos las operaciones básicas
7 Suma = A+B+C+D
8 Resta = C - A
9 Multiplicacion = A*B*D
10 Division = D/C
```

Figura 3-7: Ejemplo script.

(Fuente: Propia)

2. Cuando se tenga listo el programa que se desea, hay que dirigirse a la pestaña “Editor”, en la sección “Run”, la opción “Run”, esta permite correr el programa, pero antes el programa pide guardarlo.

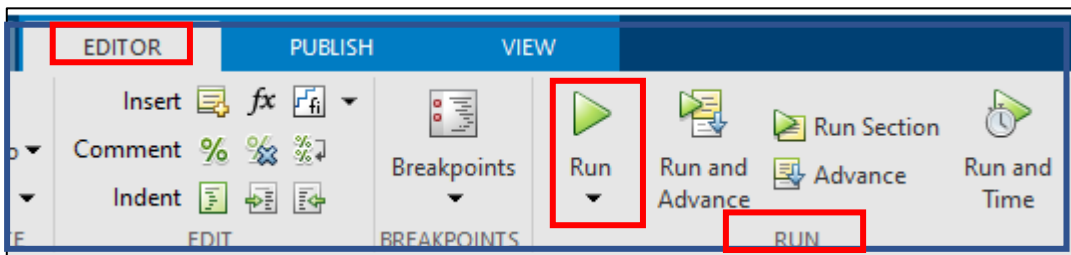


Figura 3-8: Opción Run.

(Fuente: Propia)

3. Se debe guardar el archivo en extensión .m, se debe colocar un nombre en función a los parámetros establecidos en el punto anterior, donde se detalló como nombrar las variables y scripts y se lo designa en una carpeta de su preferencia.

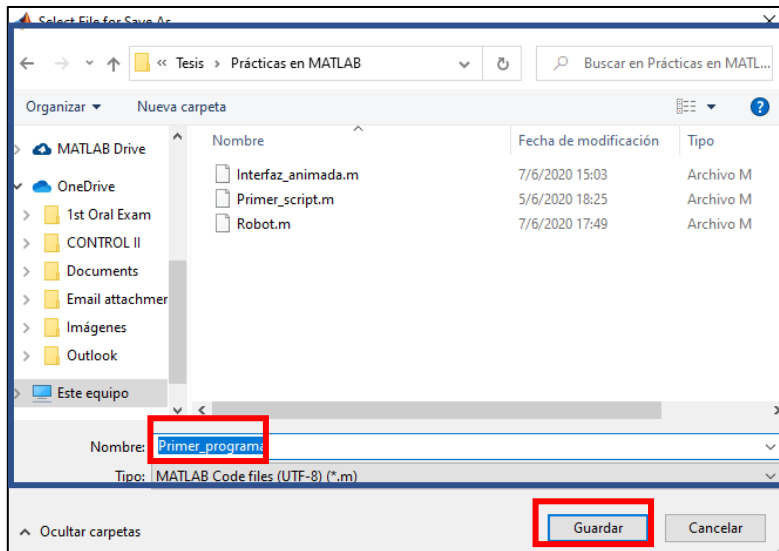


Figura 3-9: Guardar script.

(Fuente: Propia)

4. Después de guardarlo aparece una pestaña que muestra cuatro opciones:

- **Change Folder:** Esta opción coloca el script creado en la “Current Folder” de Matlab (Recomendada).
- **Add to Path:** Agregar a la ruta, se selecciona una ruta específica para el archivo.
- **Cancel:** Cancelar, esta opción permite que se cancele el proceso de guardado.
- **Help:** Abre una pestaña de Matlab explicando el destino de los archivos.

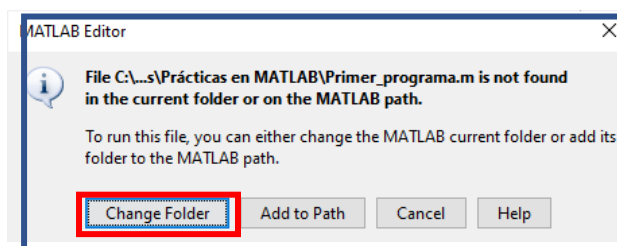
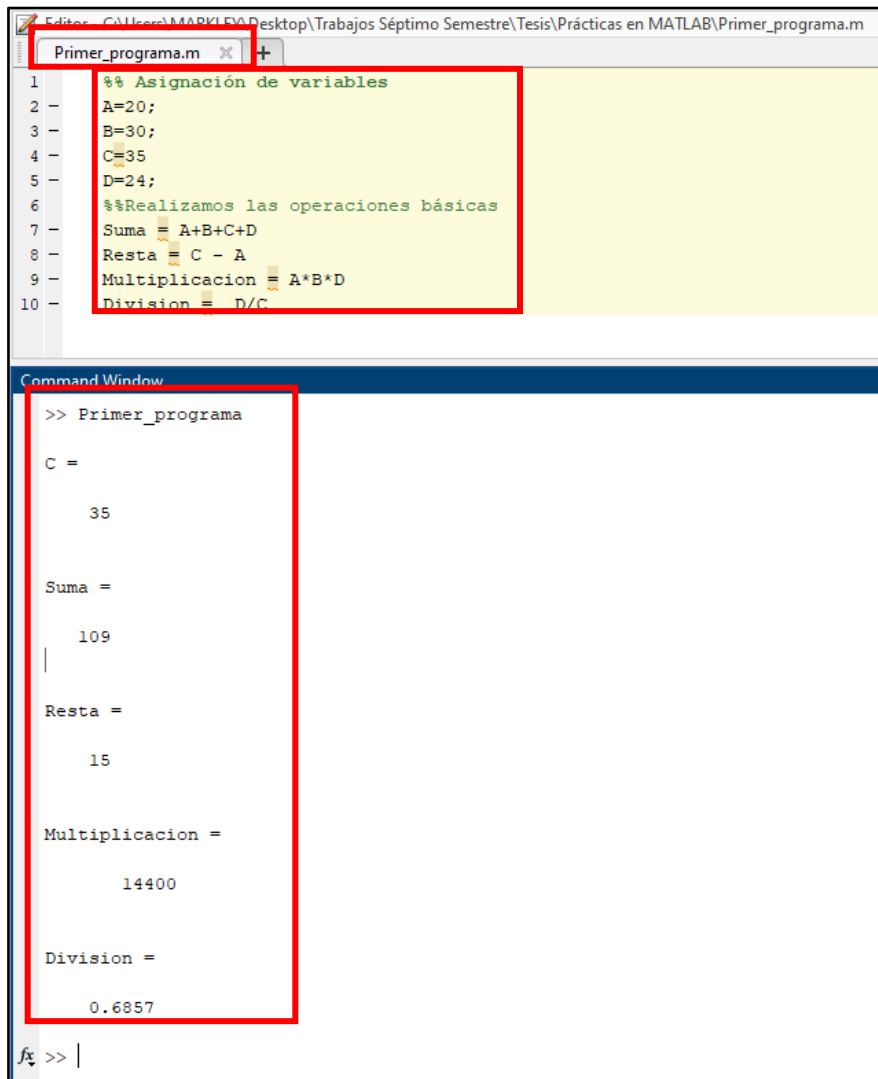


Figura 3-10: Change Folder - Script.

(Fuente: Propia)

5. Con esto, el programa completa el proceso de guardado y se observa que se ejecuta en el “Comman Window” de Matlab todos los comandos colocados en el script.



The screenshot shows the MATLAB environment. The top window is the Editor, displaying a script named 'Primer_programa.m'. The script contains the following code:

```
1 %% Asignación de variables
2 A=20;
3 B=30;
4 C=35;
5 D=24;
6 %%Realizamos las operaciones básicas
7 Suma = A+B+C+D
8 Resta = C - A
9 Multiplicacion = A*B*D
10 Division = D/C
```

The bottom window is the Command Window, showing the execution of the script. The output is as follows:

```
>> Primer_programa

C =

    35

Suma =

    109

Resta =

    15

Multiplicacion =

    14400

Division =

    0.6857

fx >> |
```

Figura 3-11: Ejecución Script.

(Fuente: Propia)

3.1.5. Vectores en Matlab

Matlab es un programa que está diseñado para el manejo de vectores y matrices, posee varios comandos para poder manejarlos de muchas formas y permitir al programador moverse de una manera más rápida y eficiente en el espacio de Matlab, los vectores

siempre se deben ingresar entre corchetes y sus dígitos tienen que ir separados por un espacio o por una coma para el caso de filas y para las columnas se lo hace con punto y coma. Existen diferentes tipos de vectores. A continuación, se detallan los más utilizados y su programación:

Vector fila: Es aquel que contiene una cantidad de datos determinada dispuestos en una sola fila, pero puede tener varias columnas. A continuación, se observa su programación en el "Command Window" en sus dos formatos, tanto separados por coma o por solo un espacio.

```
>> x = [5, 10, 3.5, 24.7, 27.5, 32.8]
x =
    5.0000    10.0000    3.5000    24.7000    27.5000    32.8000
>> x = [5 10 3.5 24.7 27.5 32.8]
x =
    5.0000    10.0000    3.5000    24.7000    27.5000    32.8000
```

Figura 3-12: Vector fila.

(Fuente: Propia)

Vector columna: Es parecido al anterior, pero en cambio los datos están dispuestos en una sola columna, y puede tener la cantidad de filas que se desea. A continuación, se observa su programación en el "Command Window". Para que los datos generen una columna, es obligatorio colocar punto y coma entre éstos.

```
>> [5; 6.7; 4.5; 2; 3]
ans =
    5.0000
    6.7000
    4.5000
    2.0000
    3.0000
```

Figura 3-13: Vector columna.

(Fuente: Propia)

Vectores con intervalo regular: Los datos de un vector con intervalo regular están dispuestos de tal forma que siguen incrementando o decrementando su valor desde el valor inicial hasta el final en la misma magnitud. Existen varias maneras de programar esto en Matlab. A continuación, se presenta un ejemplo.

```
>> %Incrementos de uno
>> a = [1:10]

a =

     1     2     3     4     5     6     7     8     9    10

>> %Decremento de uno
>> b = [10:-1:1]

b =

    10     9     8     7     6     5     4     3     2     1
```

Figura 3-14: Vector con intervalo regular (Incremento y Decremento en uno).

Índices en vectores: Los índices dentro de los vectores son muy útiles, ya que, si se desea extraer algún dato en específico del vector, se debe conocer su posición y esto se logra con los índices. Los índices determinan en que posición están los datos, por ejemplo, si de un vector se quiere extraer el número que está en la posición 5, el índice que se debe escribir dentro de la programación es el 5, de esta manera se extrae este valor. A continuación, se muestra la forma de realizar esto en Matlab.

```
>> c

c =

     0     4     8    12    16    20

>> c(2)

ans =

     4
```

Figura 3-15: Índice vectores.

(Fuente: Propia)

Existe un caso en particular, en el que no es necesario utilizar los índices, esto es gracias al comando *end*, que permite extraer el último dígito del vector, éste es muy útil en vectores que son muy extensos y se necesita conocer el valor del último dígito. A

continuación, se muestra un ejemplo.

```
>> g
g =
    1.0000    5.7500   10.5000   15.2500   20.0000
>> g(end)
ans =
    20
```

Figura 3-16: Índice vectores - Comando end.

(Fuente: Propia)

3.1.6. Matrices en Matlab

Una matriz matemática es un arreglo bidimensional de números, estas matrices están dispuestas por n filas y m columnas, éstas son muy útiles para el cálculo de varios procesos dentro del campo de la ingeniería, en el presente trabajo se lo aplica para explicar el algoritmo de “Denavit Hartenberg”, su utilidad y aplicación en el estudio de la robótica.

Construcción de una matriz: Dentro del espacio de Matlab es muy sencillo crear matrices, con la base aprendida en el manejo de vectores, la programación de matrices es muy similar. Para crear una matriz se debe hacer uso de los corchetes, los espacios (o la coma) para separar los datos de las filas; y el punto y coma para poder crear las respectivas columnas. A continuación, se presenta un ejemplo.

```
>> A= [24,21,14.5,16 ; 32.6,67.8,91.2,2; 55.2,32,1,43.8 ; 3,5,9,1]
A =
    24.0000    21.0000    14.5000    16.0000
    32.6000    67.8000    91.2000     2.0000
    55.2000    32.0000     1.0000    43.8000
     3.0000     5.0000     9.0000     1.0000
```

Figura 3-17: Construcción matriz.

(Fuente: Propia)

Valor específico: Extraer valores de una matriz es muy similar a los vectores, pero en lugar de índices, se utiliza la posición de cada número indicando la fila y la columna en

donde se encuentra. Esto se lo tiene que hacer colocando la variable que contiene la matriz y colocar entre paréntesis el número de fila y columna, su formato es “ $x(\text{fila}, \text{columna})$ ”. A continuación, se presenta un ejemplo:

- De la *matriz A* de tamaño 4x4, se desea extraer el número 67.8 que se encuentra en la fila 2, columna 2.

```
>> A
A =
    24.0000    21.0000    14.5000    16.0000
    32.6000    67.8000    91.2000     2.0000
    55.2000    32.0000     1.0000    43.8000
     3.0000     5.0000     9.0000     1.0000

>> A(2,2)
ans =
    67.8000
```

Figura 3-18: Extraer valor específico

(Fuente: Propia)

Varios valores: En las matrices también es posible extraer varios datos que se deseen, para esto se deben identificar las filas y las columnas que tiene la matriz, se debe cumplir el formato, dependiendo si se desea extraer varios valores solo de una fila o varios valores solo de una columna. A continuación, se presenta un ejemplo.

- De la *matriz A* de tamaño 4x4 y se desea extraer el tercer y cuarto dato de la fila tres.

```
>> A
A =
    24.0000    21.0000    14.5000    16.0000
    32.6000    67.8000    91.2000     2.0000
    55.2000    32.0000     1.0000    43.8000
     3.0000     5.0000     9.0000     1.0000

>> A(3,3:4)
ans =
    1.0000    43.8000
```

Figura 3-19: Extraer varios datos

(Fuente: Propia)

3.1.7. Formatos de despliegue

Dentro del espacio de trabajo de Matlab, existen varias formas de representar los números que se ingrese, esto depende del usuario y de las aplicaciones en el campo científico e industrial. A continuación, se presentan los formatos más utilizados que ofrece Matlab para desplegar los datos dentro su programación.

Notación científica: La notación científica o también conocida notación de forma exponencial es una forma de escribir valores que sean demasiado grandes o pequeños, este formato utiliza las potencias de 10 para mostrar cuantas veces hay que recorrer la coma, ya sea a la derecha o a la izquierda y obtener el número en su forma original. Para programar se debe tomar en cuenta siempre los dígitos significativos y cumplir con el formato “*xy*”. Donde *x* es el número con las cifras representativas, *e* representa la forma escrita de la notación científica y *y* representa la cantidad de posiciones que debe recorrer la coma decimal para obtener el número original. A continuación, se muestra un ejemplo.

<pre>>> 74.56e35 ans = 7.4560e+36</pre>	<pre>>> 0.557e-27 ans = 5.5700e-28</pre>
<pre>>> 6.022e23 ans = 6.0220e+23</pre>	<pre>>> 0.2254e-14 ans = 2.2540e-15</pre>

Figura 3-20: Notación científica.

Format long: Este formato, permite modificar los decimales que se muestran a la salida de cualquier operación, muestra catorce cifras decimales, para este formato se utiliza el comando *format long*.

Format bank: El formato banco es muy parecido al anterior, pero este en cambio muestra dos cifras decimales en los resultados de cualquier operación, para este formato se utiliza el comando *format bank*.

Format short: Dentro del formato short, se muestra únicamente cuatro cifras decimales en los números de salida, para este formato se utiliza el comando *format short*.

A continuación, se muestran los formatos expuestos programados en Matlab.

<pre>>> format long >> 12.345 ans = 12.3450000000000001 >> 32.345+6323.234 ans = 6.3555790000000001e+03 >> 234.76-123.12 ans = 1.1164000000000000e+02</pre>	<pre>>> format bank >> 34.34567 ans = 34.35 >> 1245.5464+123.568 ans = 1369.11 >> 6424.237-353.32 ans = 6070.92</pre>	<pre>>> format short >> 423.46345 ans = 423.4635 >> 523423.5345+3423.342 ans = 5.2685e+05 >> 55342.4234-332.523 ans = 5.5010e+04</pre>
---	--	--

Figura 3-21: Formatos de despliegue.

(Fuente: Propia)

3.1.8. Funciones matemáticas en Matlab

En función de lo explicado en puntos anteriores la herramienta de Matlab es muy importante en el campo de la ingeniería ya que posee varias librerías, estas librerías contienen una infinidad de funciones que permiten al usuario trabajar con problemas mucho más complejos de manera rápida y sencilla. Para el presente trabajo se los ha tabulado en las siguientes:

Tabla 3-1: Funciones matemáticas elementales.

(Fuente: Propia)

Funciones matemáticas elementales		
Función	Formato	Descripción
Valor absoluto	<i>abs(x)</i>	Valor absoluto de x
Raíz cuadrada	<i>sqrt(x)</i>	Raíz cuadrada de un número
Enésima raíz real	<i>nthroot(x,n)</i>	n-ésima raíz real de x
Signo número	<i>sign(x)</i>	Muestra el signo x

Funciones matemáticas elementales		
Residuo	$rem(x, y)$	Residuo de x/y
Exponencial	$exp(x)$	Calcula el exponencial de x
Logaritmo natural	$log(x)$	Calcula el logaritmo natural de x
Logaritmo base 10	$log10(x)$	Calcula el logaritmo natural base 10 de x

Tabla 3-2: Funciones de redondeo.

(Fuente: Propia)

Funciones de redondeo		
Función	Formato	Descripción
Redondeo	$round(b)$	Redondea b al entero más cercano
Redondeo	$fix(x)$	Redondea x al entero más cercano a x
Redondeo	$floor(x)$	Redondea x al entero más cercano hacia el infinito negativo
Redondeo	$ceil(x)$	Redondea x al entero más cercano hacia el infinito positivo

Tabla 3-3: Funciones matemáticas discretas.

(Fuente: Propia)

Funciones matemáticas discretas		
Función	Formato	Descripción
Factores primos	$factor(x)$	Encuentra factores primos de x
MCD	$gcd(x, y)$	Encuentra el máximo común denominador de x y y
MCM	$lcm(x, y)$	Encuentra el mínimo común múltiplo de x y y
Fracción	$rats(x)$	Representa a x como fracción
Factorial	$factorial(x)$	Encuentra x factorial
Números primos	$primes(x)$	Encuentra los números primos menores que x
Verificación número primo	$isprime$	Verifica si el número es primo o no (1=primo, 0=no primo)

Tabla 3-4: Funciones trigonométricas (radianes).

(Fuente: Propia)

Funciones trigonométricas (radianes)		
Función	Formato	Descripción
Seno	$\sin(x)$	Seno en radianes
Coseno	$\cos(x)$	Coseno en radianes
Tangente	$\tan(x)$	Tangente en radianes
Cosecante	$\csc(x)$	Cosecante en radianes
Secante	$\sec(x)$	secante en radianes
Cotangente	$\cot(x)$	Cotangente en radianes

Tabla 3-5: Funciones trigonométricas (grados).

(Fuente: Propia)

Funciones trigonométricas (grados)		
Función	Formato	Descripción
Seno	$\text{sind}(x)$	Seno en grados
Coseno	$\text{cosd}(x)$	Coseno en grados
Tangente	$\text{tand}(x)$	Tangente en grados
Cosecante	$\text{cscd}(x)$	Cosecante en grados
Secante	$\text{secd}(x)$	Secante en grados
Cotangente	$\text{cotd}(x)$	Cotangente en grados

Tabla 3-6: Funciones trigonométricas inversas (radianes).

(Fuente: Propia)

Funciones trigonométricas inversas (radianes)		
Función	Formato	Descripción
Seno inverso	$\text{asin}(x)$	Seno inverso en radianes
Coseno inverso	$\text{acos}(x)$	Coseno inverso en radianes
Tangente inverso	$\text{atan}(x)$	Tangente inverso en radianes
Cosecante inversa	$\text{acsc}(x)$	Cosecante inversa en radianes
Secante inversa	$\text{asec}(x)$	Secante inversa en radianes
Cotangente inversa	$\text{acot}(x)$	Cotangente inversa en radianes

Tabla 3-7: Funciones trigonométricas inversas (grados).

(Fuente: Propia)

Funciones trigonométricas inversas (grados)		
Función	Formato	Descripción
Seno inverso	$asind(x)$	Seno inverso en grados
Coseno inverso	$acosd(x)$	Coseno inverso en grados
Tangente inverso	$atand(x)$	Tangente inverso en grados
Cosecante inversa	$acscd(x)$	Cosecante inversa en grados
Secante inversa	$asecd(x)$	Secante inversa en grados
Cotangente inversa	$acotd(x)$	Cotangente inversa en grados

Tabla 3-8: Funciones trigonométricas hiperbólicas (radianes).

(Fuente: Propia)

Funciones trigonométricas hiperbólicas (radianes)		
Función	Formato	Descripción
Seno hiperbólico	$\sinh(x)$	Seno hiperbólico en grados
Coseno hiperbólico	$\cosh(x)$	Coseno hiperbólico en grados
Tangente hiperbólico	$\tanh(x)$	Tangente hiperbólico en grados
Cosecante hiperbólica	$\operatorname{csch}(x)$	Cosecante hiperbólica en grados
Secante hiperbólica	$\operatorname{sech}(x)$	Secante hiperbólica en grados
Cotangente hiperbólica	$\operatorname{coth}(x)$	Cotangente hiperbólica en grados

3.1.9. Gráficas en Matlab

Las gráficas son muy utilizadas por los ingenieros y científicos para muchas aplicaciones y se las utiliza por sus ventajas al momento de la presentación de datos, sus principales características son:

- Cuando se tiene una tabla de datos muy grande y son difíciles de interpretar, por medio de las gráficas son más sencillos de hacerlo.
- Cuando los datos están graficados es mucho más sencillo identificar características, parámetros, tendencias, altos, bajos, aislar puntos, etc.

- Las gráficas son un medio de verificación rápida para conocer si un proceso funciona según lo que se ha previsto en términos teóricos.

Dentro de esta sección se explica las herramientas y comandos básicos necesarios para realizar gráficas en 2D con Matlab y su programación.

Una vez se defina cualquier tipo de vector x y y y todos sus valores correspondientes, es posible crear una gráfica.

```
>> %Ejemplo funciones para graficar
>> x=[0:0.1:10]; % Vector que representa el tiempo
>> y=2*sin(x); %Función seno que irá cambiando conforme el tiempo
```

Figura 3-22: Datos para gráficas

(Fuente: Propia)

Plotear gráficas: Una vez se tengan todos los datos correspondientes a las gráficas se puede plotear, se debe utilizar el comando *plot* y cumple el formato “*plot(Eje de las x, Eje de las y)*”.

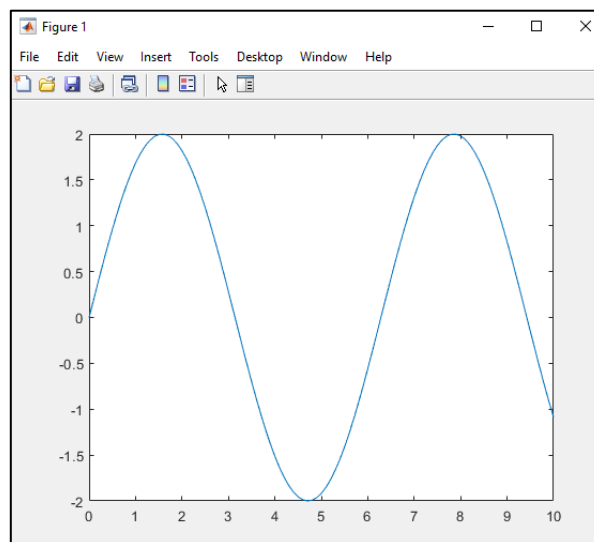


Figura 3-23: Comando plot – Gráfica

(Fuente: Propia)

Títulos en gráficas: La gráfica se presenta muy simple por lo que es posible agregar un título para identificar la función que se está representando, esto se lo debe hacer con

el comando *title* y cumple con el formato "*title('título')*".

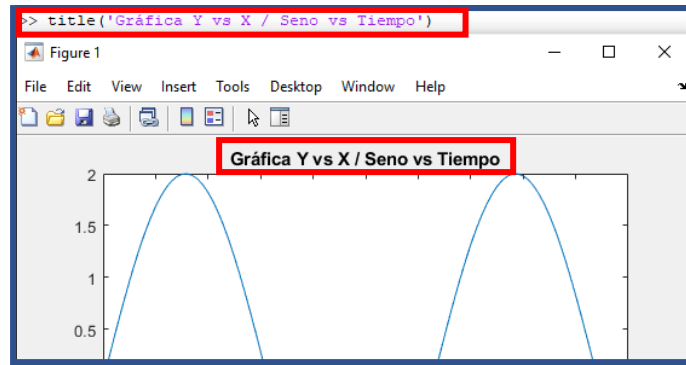


Figura 3-24: Comando *title*.

(Fuente: Propia)

Etiqueta eje x: Para poder diferenciar las variables que se tienen en los ejes, en este caso en el eje *x*, se puede colocar un nombre representativo, para esto se utiliza el comando *xlabel* que cumple el formato "*xlabel('nombre deseado')*".

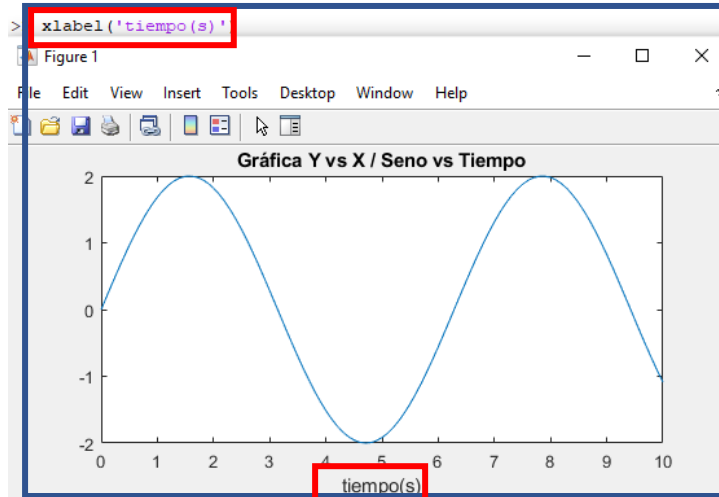


Figura 3-25: Comando *xlabel*.

(Fuente: Propia)

Etiqueta eje y: Para poder diferenciar las variables que se tienen en los ejes, en este caso en el eje *y*, se puede colocar un nombre representativo, para esto se utiliza el comando *ylabel* que cumple el formato "*ylabel('nombre deseado')*".

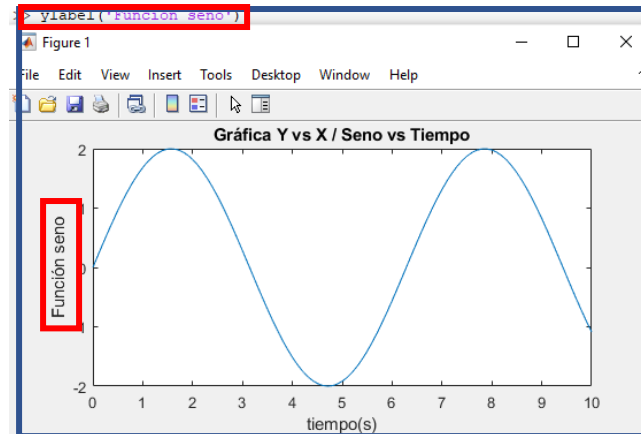


Figura 3-26: Comando ylabel

(Fuente: Propia)

Cuadrículas gráficas: Las cuadrículas en las gráficas también dependen de la aplicación, ya que, en muchos casos, son útiles para identificar los puntos exactos por donde pasa la función, para activar la cuadrícula se lo debe hacer con el comando *grid* y en frente se coloca la palabra *on* que en inglés significa encender.

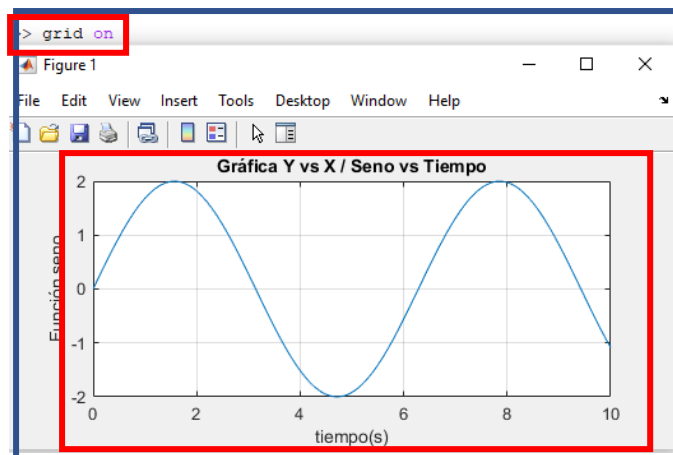


Figura 3-27: Comando grid on.

(Fuente: Propia)

Leyendas: Este comando es muy útil cuando se tienen varias gráficas sobrepuestas, con éstas leyendas se puede identificar cual curva corresponde a una función en específico, esto se lo tiene que hacer con el comando *legend* y va manejado por el formato "*legend('nombre deseado')*".

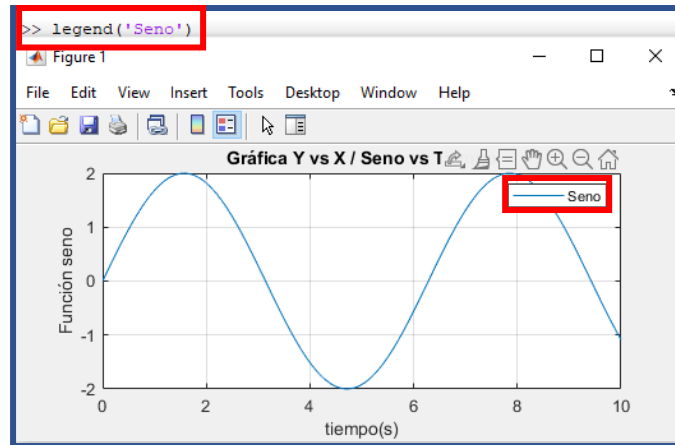


Figura 3-28: Comando legend.

(Fuente: Propia)

Cuando se desea cerrar todas las ventanas de Matlab correspondientes a las figuras creadas, con el comando *close all*, se cierran todas.

Línea, color y estilo de marca: Dentro de las figuras que se mostraron en los puntos anteriores, las líneas de las gráficas y la representación de datos siempre se presentaban igual, con los mismos colores, con el mismo grosor y estilo, Matlab ofrece también comandos para poder modificar estos parámetros y realizar gráficas en función de la aplicación o necesidad del usuario, para esto se necesitan las siguientes tres tablas que indican la cantidad de herramientas programables que se dispone para modificar los criterios mencionados.

Tabla 3-9: Estilo de línea para gráficas.

(Fuente: Propia)

Línea	
Tipo de línea	Indicador
Sólida	—
Punteada	:
Raya- punto	—.
Rayada	--

Tabla 3-10: Colores para gráficas.

(Fuente: Propia)

Color	
Tipo de línea	Indicador
Azul	<i>b</i>
Verde	<i>g</i>
Rojo	<i>r</i>
Cian	<i>c</i>
Magenta	<i>m</i>
Amarillo	<i>y</i>
Negro	<i>k</i>

Tabla 3-11: Estilo de marca - Gráficas.

(Fuente: Propia)

Estilo de marca	
Tipo de línea	Indicador
Punto	.
Círculo	<i>o</i>
Marca x	<i>X</i>
Más	+
Estrella	*
Cuadrado	<i>s</i>
Diamante	<i>d</i>
Triángulo bajo	<i>v</i>
Triángulo arriba	^
Triángulo izquierdo	<
Triángulo derecho	>
Pentagrama	<i>p</i>
Hexagrama	<i>H</i>

Fuentes, líneas y axes: Para poder cambiar el tamaño de las fuentes, el grosor de las líneas, y el tamaño de los axes, Matlab ofrece tres comandos:

Tabla 3-12: Comandos para fuentes, líneas y axes.

(Fuente: Propia)

Fuentes, Líneas y Axes		
Propiedad	Comando	Uso
Fuentes	<i>FontSize</i>	Modifica el título, nombre de los x y y .
Líneas	<i>linewidth</i>	Modifica las líneas que son graficadas con los datos.
Axes	<i>set (gca)</i>	Modifica los números del axe, el título y las etiquetas de los ejes x y y .

Escalamiento de ejes: Cuando se coloca los datos de una función dentro del espacio de programación de Matlab y se los dibuja, Matlab va a tomar los puntos de los ejes x y y automáticamente para que la gráfica quede centrada, pero en caso de que se requiera cambiar el escalamiento de los ejes, Matlab ofrece esta opción con el comando *axis*, este tiene el formato “*axis([Xmin Xmax Ymin Ymax])*”. Donde cada parámetro indica el escalamiento desde el valor mínimo hasta el valor máximo en los respectivos ejes.

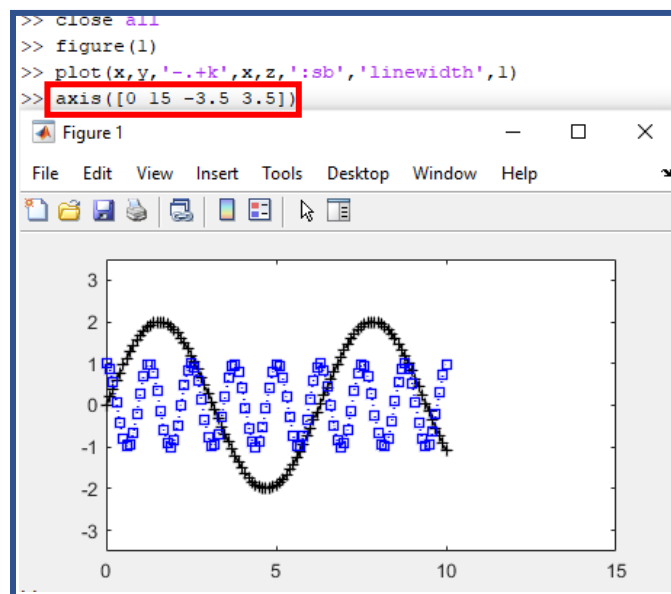


Figura 3-29: Escalamiento gráficas.

(Fuente: Propia)

Comandos para gráficas		
		y poder seguir plotando en la misma figura.
Gráficar varias funciones	<code>plot(x, y, x, z)</code>	Varias curvas en la misma gráfica en una sola línea de programación.
Varias leyendas	<code>legend('leyenda1', 'leyenda2')</code>	Colocar varias leyendas
Varias figuras	<code>figure(#)</code>	Para ir abriendo pestañas con las figuras que se requiera
Texto en gráfica	<code>text(x, y, 'titulo', 'fontsize', #)</code>	Colocar texto en las coordenadas x y y con un tamaño de letra requerido
Cerrar todo	<code>close all</code>	Cerrar todas las figuras de Matlab
Varias gráficas	<code>subplot(filas, columnas, axe)</code>	Colocar varias gráficas separadas en la misma figura
Escalamiento	<code>axis([Xmin Xmax Ymin Ymax])</code>	Para poder escalar los ejes x y y a conveniencia

3.2. Toolbox Robotics de Matlab

3.2.1. Manual comandos

Dentro de la librería de Toolbox Robotics, se encuentran varios comandos que pueden ser utilizados para varias aplicaciones, en la página de Peter Corke, existe una guía completa, donde se puede ver absolutamente todas a detalle. A continuación, se muestra donde conseguir el manual de Peter Corke.

1. Entrar en el siguiente link: <https://petercorke.com/> , se redirige a la página oficial de Peter Corke.
2. Ir a las pestañas de la página y seleccionar "TOOLBOXES" y escoger la opción "ROBOTICS TOOLBOX".

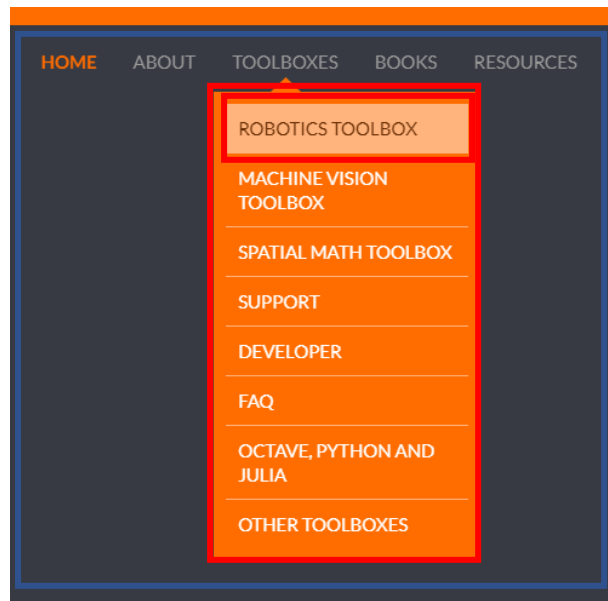


Figura 3-31: TOOLBOXES - ROBOTICS TOOLBOX.

(Fuente: Propia)

- Una vez dentro de “Robotics Toolbox”, se debe buscar la documentación correspondiente, aquí se encuentra un manual “RTB manual”, dar click y se descarga automáticamente.

Documentation

- The book *Robotics, Vision & Control, second edition* (Corke, 2017) is a detailed introduction to mobile robotics, navigation, localization; and arm robot kinematics, Jacobians and dynamics illustrated using the Robotics Toolbox for MATLAB.
- The manual (below) is a PDF file is a printable document (over 400 pages). It is auto-generated from the comments in the MATLAB code and is fully: to external web sites, the table of content to functions, and the “See also” functions to each other.
- The Toolbox documentation also appears in the MATLAB help browser under Supplemental Software.

RTB manual
Size: 9,00 B Format : PDF
[Preview](#)

Figura 3-32: Documentación comandos Matlab - Toolbox Robotics.

(Fuente: Propia)

Comando *Link*: El comando *LINK* es un objeto que contiene toda la información relacionada con la articulación y enlace del robot, entre estas características se tiene parámetros cinemáticos, parámetros de inercia, parámetros de transmisión, etc. Dentro de este se colocan parámetros del algoritmo de Denavit-Hartenberg y cumple con el formato “*Link*([*theta* = *q*, *d* = 0, *a* = 0, *alpha* = 0])”.

Comando *Seriallink*: Sirve para unir todos los eslabones que se realizó con el comando *Link*, cumple con el formato "*SerialLink (n1 n2 n3 n4 n5 n6)*".

Comando *jttraj*: Permite calcular, la trayectoria espacial que debe tomar el robot, según los puntos que fueron asignados por el usuario, este comando cumple con el formato "*jttraj (Q₀, Q_f, M)*". Donde Q_0 es la posición inicial de donde va a arrancar el robot, Q_f es la posición final a donde se desea que el robot se mueva y M es el vector que determina los pasos (vector tiempo).

Comandos *xlim, ylim, zlim*: Este comando permite establecer los límites de visualización final del robot en los ejes x , y y z , estos límites aplican cuando el robot haya cumplido las trayectorias programadas por el usuario, su formato es "*x/y/z (límite inferior, límite superior)*".

Comando *view*: El comando *view*, permite fijar los puntos del control de cámara dentro de las figuras realizadas en Matlab, pudiendo observar desde el punto deseado por el usuario y tener una mejor visión del trabajo ploteado. El comando *view* considera dos ángulos en grados, el ángulo *azimuth* que va de 0° a 360° , gira en sentido a las manecillas del reloj en el propio eje del objeto y el ángulo *elevation*, va de 0° a 90° considerando el ángulo vertical. A continuación, en la siguiente figura se presenta los ángulos *azimuth* y *elevation*.

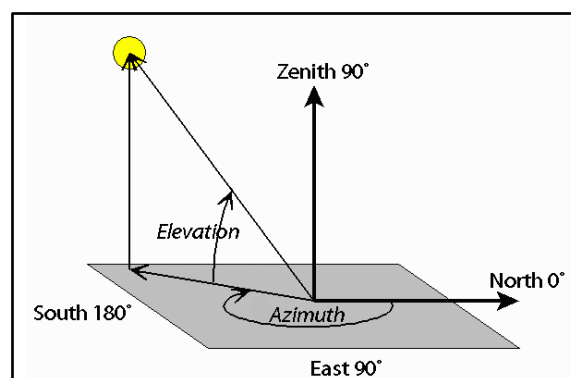


Figura 3-33: Ángulos azimuth y elevation.

(Fuente: Propia)

Para el caso de este proyecto se consideró los puntos 150 y 30, ya que son valores por debajo de los máximos manteniendo una vista completa y clara del robot simulado.

Comando *teach*: El comando *teach* es muy utilizado, ya que permite una interacción con el robot que se plotea en el espacio de Matlab, la interacción es a través de slides, el usuario tiene la capacidad de moverlas y ver como el robot tiene que desplazarse para alcanzar dichas coordenadas en el efector final del robot, el formato que se aplica es "*nombrerobot.teach()*"

3.3. Algoritmo de Denavit-Hartenberg

3.3.1. Ángulos de Euler ZXZ

Los ángulos de Euler, es un conjunto de tres ángulos en el espacio, que se los representa con los símbolos ϕ , θ y Ψ , estos ángulos determinan la orientación del sistema de referencia final respecto al inicial.

A continuación, se presentan los tres ángulos independientes, según el eje en el que se encuentran, cabe recalcar que estos ángulos pueden tomar muchas más configuraciones dentro del espacio.

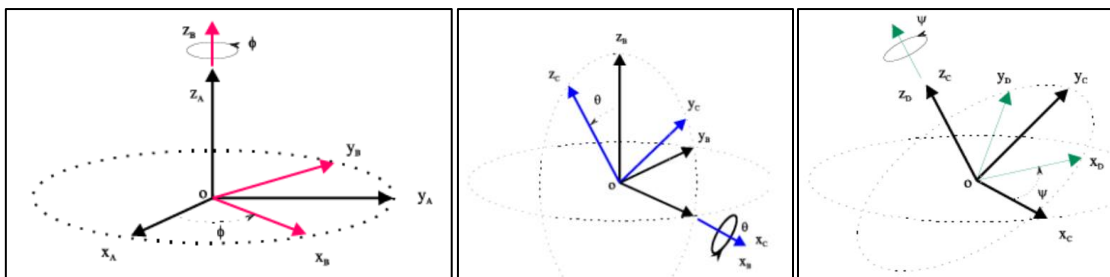


Figura 3-34: Ángulo ϕ - Eje z, Ángulo θ - Eje x. Ángulo Ψ - Eje z.

(Marcos Simó, Ángulos, 2018)

Primer giro en el eje z:

$$R_z(\varphi) = \begin{bmatrix} \cos(\varphi) & -\text{sen}(\varphi) & 0 \\ \text{sen}(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-1

(Marcos Simó, Ángulos, 2018)

Primer giro en el eje x:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\text{sen}(\theta) \\ 0 & \text{sen}(\theta) & \cos(\theta) \end{bmatrix}$$

Ecuación 3-2

(Marcos Simó, Ángulos, 2018)

Segundo giro en el eje z:

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\text{sen}(\psi) & 0 \\ \text{sen}(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-3

(Marcos Simó, Ángulos, 2018)

Matriz resultante de las matrices de giro z, x, z

$$R(\varphi, \theta, \Psi) = R_z(\varphi) \cdot R_x(\theta) \cdot R_z(\Psi)$$

$$\begin{bmatrix} c(\varphi) \cdot c(\Psi) - s(\varphi) \cdot c(\theta) \cdot s(\Psi) & -c(\varphi) \cdot s(\Psi) - s(\varphi) \cdot c(\theta) \cdot c(\Psi) & s(\varphi) \cdot c(\theta) \\ s(\varphi) \cdot c(\Psi) + c(\varphi) \cdot c(\theta) \cdot s(\Psi) & -s(\varphi) \cdot s(\Psi) + c(\varphi) \cdot c(\theta) \cdot c(\Psi) & -c(\varphi) \cdot s(\theta) \\ s(\theta) \cdot s(\Psi) & s(\theta) \cdot c(\Psi) & c(\theta) \end{bmatrix}$$

Ecuación 3-4

(Marcos Simó, Ángulos, 2018)

3.3.2. Roll, Pitch & Yaw (balanceo, cabeceo y oscilación)

Dentro del código realizado para este proyecto en Matlab, con la ayuda de la interfaz animada y con el comando `teach()` visto anteriormente, se podrá visualizar estos ángulos de manera referencial. Los ángulos ROLL, PITCH & YAW es otro conjunto de ángulos de Euler, estos sirven para determinar la posición y orientación de un sólido rígido en el espacio. En estos ángulos ya se consideran los tres ejes x, y, z , donde el ángulo de balanceo o ROLL corresponde a una rotación en el eje z , el ángulo de cabeceo o PITCH corresponde a una rotación en el eje y y el ángulo de oscilación o YAW corresponde a un giro alrededor del eje x .

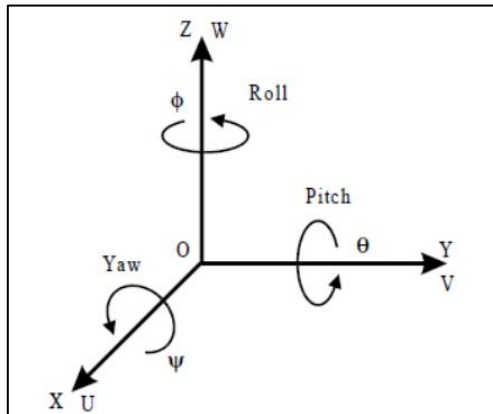


Figura 3-35: Ángulos ROLL, PITCH & YAW - Ejes x, y, z.

(Marcos Simó, Ángulos, 2018)

Estos ángulos se muestran en la interfaz animada realizada en Matlab, los ángulos son referenciales, ya que en el presente trabajo se enfoca en los parámetros del algoritmo de Denavit Hartenberg, donde se toman otras variables para observar las coordenadas del efector final. A continuación, se muestra estos ángulos dentro de la interfaz, donde la R es el ángulo ROLL, la P es el ángulo PITCH y la Y es el ángulo YAW.

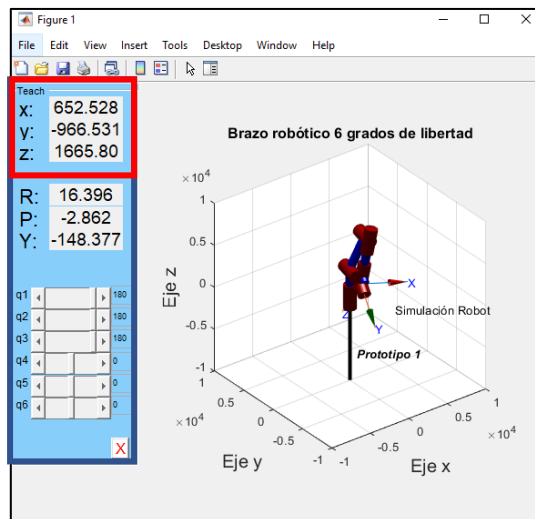


Figura 3-36: Ángulos ROLL, PITCH & YAW - Interfaz.

(Fuente: Propia)

3.3.3. Parámetros Denavit-Hartenberg

Dentro de los parámetros de Denavit-Hartenberg se toma en cuenta dos ángulos y dos

distancias, estos van en orden según el modelo que se aplique. A continuación, se explica a breves rasgos el modelo estándar y el modelo modificado, pero hay que entender que los 4 parámetros mencionados aplican para los dos métodos, estos son:

- **Ángulo θ_i :** Es el ángulo formado por el eje X_{i-1} , con el sistema de coordenadas X_i medido en un plano perpendicular al eje Z_i , utilizando la regla de la mano derecha. Es un parámetro variable en articulaciones de revolución.
- **Distancia d_i :** Es la distancia a lo largo del eje Z_{i-1} , desde el origen del sistema de coordenadas $i-1$ hasta la intersección con el eje Z_{i-1} con el eje X_i .
- **Distancia a_i :** Es la distancia a lo largo del eje X_i que va desde la intersección del eje Z_{i-1} , con el X_i , hasta el origen del sistema de coordenadas i –ésimo, para los pares rotatorios.
- **Ángulo α_i :** Es aquel ángulo determinado por la separación del eje Z_{i-1} y Z_i , medido en un plano perpendicular al eje X_i , utilizando la regla de la mano derecha.

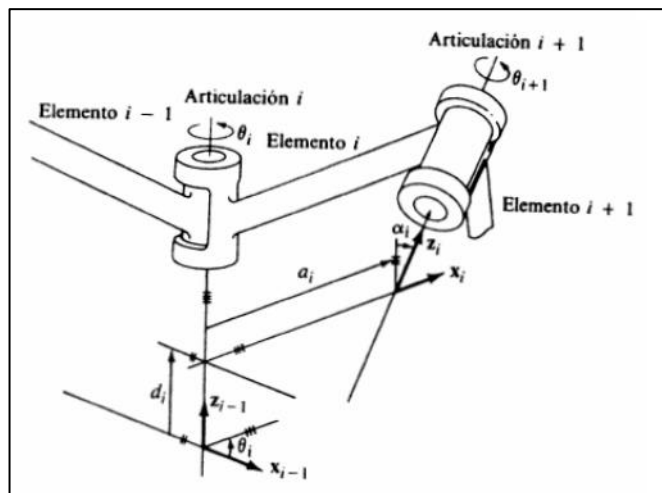


Figura 3-37: Parámetros Denavit-Hartenberg.

(Ánimo, Cinemática, sf)

3.3.4. Modelos del algoritmo de Denavit-Hartenberg

Modelo estándar Denavit-Hartenberg: El modelo estándar de la matriz de Denavit Hartenberg fue el primer modelo que se obtuvo al tomar en cuenta varios parámetros estos son: los ángulos de rotación, la distancia de los eslabones, los grados de libertad y los ejes de coordenadas colocados en cada uno de ellos, es importante conocer que

el eje de coordenadas principal es el que se encuentra en la base, por lo que se tiene las denotaciones de x, y y z .

Modelo modificado Denavit-Hartenberg: El modelo modificado de la matriz de Denavit Hartenberg también toma en cuenta varios parámetros como los ángulos de rotación, la distancia de los eslabones, los grados de libertad y los ejes de coordenadas colocados en cada uno de ellos, similar al modelo estándar, pero la diferencia principal radica en los sistemas coordenadas que se asigna, ya que a la base se la considera un eslabón más pero fijo, por lo que la connotación de la base empezará con x_1, y_1 y z_1 , totalmente diferente a la nomenclatura tomada en el modelo estándar. A continuación, se presenta las diferencias entre los ejes de coordenadas de los dos modelos descritos.

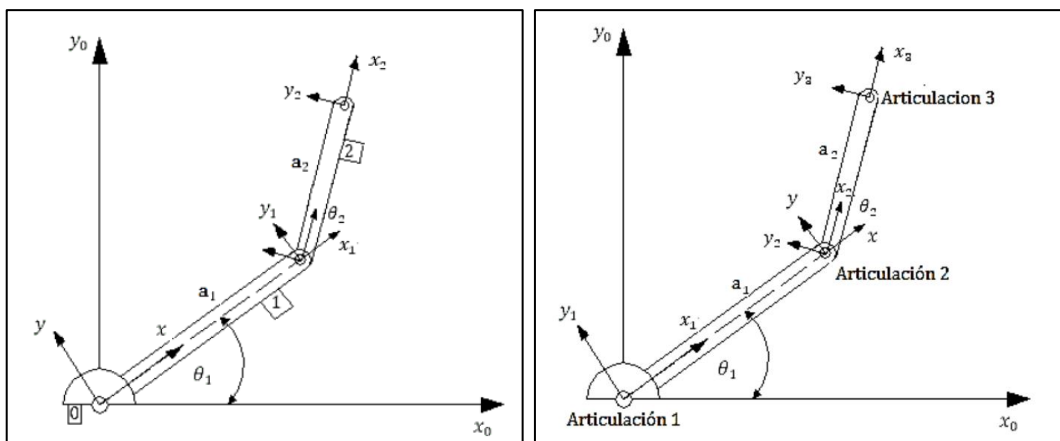


Figura 3-38: Denotación Denavit Hartenberg Estándar vs Modificado.

(Ánimo, Cinemática, sf) - (Mario Granja, Modelación Robótica, 2014)

Matrices Denavit-Hartenberg

En esta sección se muestran las diferencias de las diferentes matrices y como estas disponen de los cuatro parámetros necesarios para Denavit-Hartenberg.

Matriz resultante modelo estándar

$$T_n = Rot(Z_{n-1}, \theta_n). Trans(Z_{n-1}, d_n). Trans(X_n, a_n). Rot(X_n, \alpha_n)$$

$$T_n = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) \cdot \cos(\alpha_n) & \sin(\theta_n) \cdot \sin(\alpha_n) & a_n \cdot \cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n) \cdot \cos(\alpha_n) & -\cos(\theta_n) \cdot \sin(\alpha_n) & a_n \cdot \sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-5

(Torriti, M. T., 2011)

Donde:

- **Primera matriz:** Rotación en el eje “z” con un ángulo “teta”.
- **Segunda matriz:** Traslación de una distancia “d” en el eje “z”.
- **Tercera matriz:** Traslación de una distancia “a” en el eje “x”.
- **Cuarta matriz:** Rotación en el eje “x” de un ángulo “alfa”.

Matriz resultante modelo modificado

$$T_n = Rot(X_{n-1}, \alpha_n) \cdot Trans(X_{n-1}, a_n) \cdot Rot(Z_n, \theta_n) \cdot Trans(Z_n, d_n)$$

$$T_n = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) & 0 & a_n \\ \sin(\theta_n) \cdot \cos(\alpha_n) & \cos(\theta_n) \cdot \cos(\alpha_n) & -\sin(\alpha_n) & -\sin(\alpha_n) \cdot d_n \\ \sin(\theta_n) \cdot \sin(\alpha_n) & \cos(\theta_n) \cdot \sin(\alpha_n) & \cos(\alpha_n) & \cos(\alpha_n) \cdot d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-6

(Torriti, M. T., 2011)

Donde:

- **Primera matriz:** Rotación en el eje “x” de un ángulo “alfa”.
- **Segunda matriz:** Traslación de una distancia “a” en el eje “x”.
- **Tercera matriz:** Rotación en el eje “z” de un ángulo “teta”.
- **Cuarta matriz:** Traslación de una distancia “d” en el eje “z”.

Matrices homogéneas

Primera Matriz (Estándar vs Modificado)

$$Rot(Z_{n-1}, \theta_n) = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) & 0 & 0 \\ \sin(\theta_n) & \cos(\theta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Rot(X_{n-1}, \alpha_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_n) & -\sin(\alpha_n) & 0 \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-7

(Torriti, M. T., 2011)

Segunda Matriz (Estándar vs Modificado)

$$Trans(Z_{n-1}, d_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Trans(X_{n-1}, a_n) = \begin{bmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-8

(Torriti, M. T., 2011)

Tercera Matriz (Estándar vs Modificado)

$$Trans(X_n, a_n) = \begin{bmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Rot(Z_n, \theta_n) = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) & 0 & 0 \\ \sin(\theta_n) & \cos(\theta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-9

(Torriti, M. T., 2011)

Cuarta Matriz

$$Rot(X_n, \alpha_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_n) & -\sin(\alpha_n) & 0 \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Trans(Z_n, d_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-10

(Torriti, M. T., 2011)

3.3.5. Modelo cinemático

Para el presente proyecto, se pretende realizar un robot de 6 grados de libertad, por lo que es necesario que se muestre la distribución tanto de los eslabones y de las articulaciones ya que, en función de estas se tiene los resultados de las matrices finales.

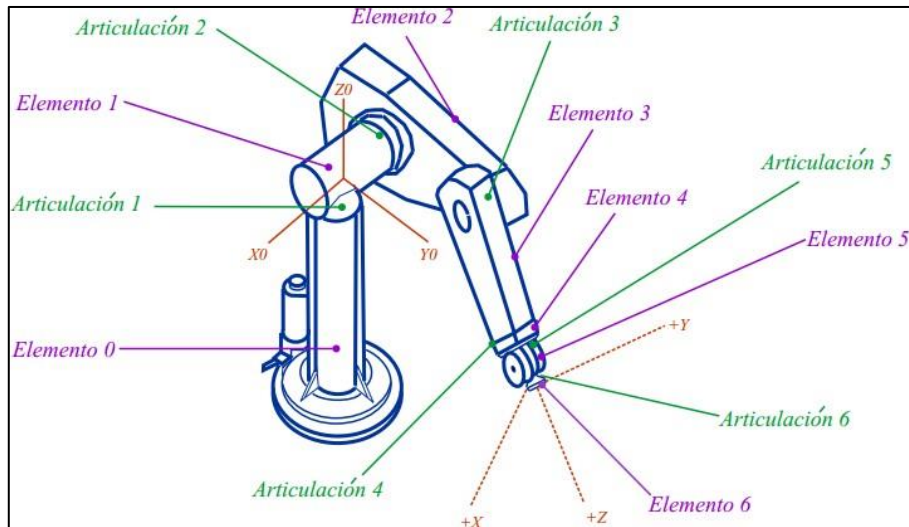


Figura 3-39: Distribución eslabones y articulaciones.

(Andrés Chang, Robot Articular, 2014)

Dentro de Matlab y gracias a la librería Toolbox Robotics, se tiene un comando para determinar la matriz final de la posición del robot, por lo que para verificar de mejor manera, se utiliza el comando `brazo_robot.fkine(posicion)`, con esto se observa en la interfaz animada y en los tres primeros dígitos de la columna cuatro de la matriz de tamaño 4x4 la posición final en coordenadas x, y y z del efector final. A continuación, se muestra la matriz de la posición final en relación con su base, mostrando T_x, T_y y T_z como los puntos en coordenadas dentro del espacio del efector final.

$$M_{(n-1, n)} = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & T_x \\ R_{yx} & R_{yy} & R_{yz} & T_y \\ R_{zx} & R_{zy} & R_{zz} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3-11

(Torriti, M. T., 2011)

3.4. Interfaz animada de usuario

Una interfaz animada, es aquella que permite al usuario interactuar indirectamente con la programación de cualquier algoritmo, no importa que dicho programa tenga una programación en alto nivel, bajo nivel, código escrito, código en bloques, etc.

Para el caso del presente proyecto, el usuario tiene dos herramientas para la interacción

con el robot, estas son el "Command Window", para introducir los ángulos solicitados y requeridos para los parámetros de Denavit Hartenberg y el menú por medio de slides que ofrece la librería Toolbox Robotics, para la observación de las coordenadas del efector final, en función de los parámetros designados anteriormente.

3.4.1. Implementación de comandos

Tanto para la entrada de datos que se lo solicita al usuario, los parámetros necesarios para el robot y el menú con slides que ofrece la librería Toolbox Robotics, se utilizaron algunos comandos. A continuación, se detallan los principales:

Comando *disp*: El comando *disp* viene de la palabra display en inglés, que significa mostrar, con este comando principalmente se le detalla al usuario el proceso que va a ejecutarse, esto quiere decir que, con un título o una nota se le indica al usuario que es lo que debe de hacer, para el caso del robot simulado, muestra que se deben ingresar los ángulos correspondientes. Este comando cumple con un formato muy sencillo "*disp. ('x')*". Donde *x* es la palabra o frase que se desee introducir y tiene que ir entre comillas simples.

Comando *input*: Ayuda a que el usuario pueda ingresar los ángulos que se desea, a través del "Command Window", el programa va solicitando los ángulos necesarios para los parámetros de Denavit Hartenberg, este comando cumple con el formato "*x = input('título')*". Donde *x* es la variable que guarda el valor que asigne el usuario, *input* indica que es un valor de entrada y *título* es una nota acerca del valor que se solicite, este generalmente es el nombre de la variable y las unidades en que dicha variable debe ser ingresada.

Datos de entrada en Command Window: Los datos de entrada son aquellos que el programa solicita para completar y resolver las operaciones que fueron programadas y que son necesarios para la ejecución completa de este. Para el caso del robot, se solicita los ángulos de rotación teta y alfa que deben ser ingresados en radianes. Para esta aplicación se utiliza el comando "*input*".

Menú con slides: Gracias a la librería Toolbox Robotics de Peter Corke con el comando *teach*, se despliega un menú con slides donde, permite guiar el robot en la dirección que se desee en función del movimiento de sus grados de libertad. Adicional a esto, se

observan tres coordenadas, correspondientes a los ejes x, y y z , estas coordenadas indican la posición final del efector final en el espacio, también se muestran los ángulos, donde la R es el ángulo ROLL, la P es el ángulo PITCH y la Y es el ángulo YAW, correspondientes a los ángulos de balanceo, cabeceo y oscilación, cabe recalcar que estos ángulos están en grados.

3.5. Pruebas de funcionamiento

Para verificar que todo dentro del proyecto esté funcionando, se realizaron algunas pruebas y se las ha dividido de la siguiente manera:

3.5.1. Prueba de matrices

Tabla 3-14 Prueba de matrices Denavit - Hartenberg.

(Fuente: Propia)

Prueba	Descripción	CUMPLE	NO CUMPLE
Matrices de transformación Denavit-Hartenberg Standard	Dentro del espacio de Matlab se colocaron las matrices de transformación del método standard y los parámetros necesarios en cada una de ellas, con esto se realizó el producto necesario entre matrices y se obtuvo la matriz resultante.	√	
Matrices de transformación Denavit-Hartenberg Modificado	Dentro del espacio de Matlab se colocaron las matrices de transformación del método modificado y los parámetros necesarios en cada una de ellas, con esto se realizó el producto necesario entre matrices y se obtuvo la matriz resultante.	√	
Matriz resultante Denavit-Hartenberg Standard	Dentro del espacio de Matlab se colocó solo la matriz resultante del método standard con los parámetros necesarios, para comprobar que se puede realizar el producto matriz por	√	

Prueba	Descripción	CUMPLE	NO CUMPLE
	matriz o directamente la aplicación de la matriz general.		
Matriz resultante Denavit-Hartenberg Modificado	Dentro del espacio de Matlab se colocó solo la matriz resultante del método modificado con los parámetros necesarios, para comprobar que se puede realizar el producto matriz por matriz o directamente la aplicación de la matriz general.	√	

3.5.2. Pruebas de posicionamiento del efector final

Dentro de esta prueba se verifican los valores de las coordenadas en x, y y z en el espacio, tanto en la matriz final resultante de Denavit-Hartenberg como en la interfaz animada, con la finalidad comprobar el correcto posicionamiento del efector final.

Prueba 1

Tabla 3-15: Prueba de posicionamiento - Prueba 1.

(Fuente: Propia)

GDL	Ángulo Teta	Distancia d	Distancia a	Ángulo Alfa
1	$\pi/3$	1000	3500	$\pi/4$
2	$\pi/2$	3000	2000	$\pi/4$
3	$\pi/6$	2500	1500	π
4	π	500	560	π
5	$\pi/7$	600	670	$\pi/2$
6	π	100	120	$\pi/6$

Prueba 2

Tabla 3-16: Prueba de posicionamiento - Prueba 2.

(Fuente: Propia)

GDL	Ángulo Teta	Distancia d	Distancia a	Ángulo Alfa
1	π	1677	2524	$\pi/2$
2	π	3456	2110	$\pi/2$

GDL	Ángulo Teta	Distancia d	Distancia a	Ángulo Alfa
3	$\pi/4$	2550	2534	π
4	π	1500	567	$\pi/7$
5	$\pi/2$	623	270	$\pi/3$
6	$\pi/2$	115	134	$\pi/8$

Prueba 3

Tabla 3-17: Prueba de posicionamiento - Prueba 3.

(Fuente: Propia)

GDL	Ángulo Teta	Distancia d	Distancia a	Ángulo Alfa
1	$\pi/4$	1060	1788	$\pi/2$
2	$\pi/3$	4056	2476	$\pi/2$
3	$\pi/3$	1340	1670	π
4	π	670	2830	$\pi/5$
5	π	780	893	$\pi/3$
6	$\pi/8$	241	1346	$\pi/5$

Prueba 4

Tabla 3-18: Prueba de posicionamiento - Prueba 4.

(Fuente: Propia)

GDL	Ángulo Teta	Distancia d	Distancia a	Ángulo Alfa
1	$\pi/2$	3261	3799	π
2	$\pi/2$	5645	4116	π
3	π	3467	2110	$\pi/2$
4	π	776	3519	$\pi/2$
5	π	805	935	$\pi/2$
6	$\pi/7$	411	341	$\pi/7$

Resultados pruebas

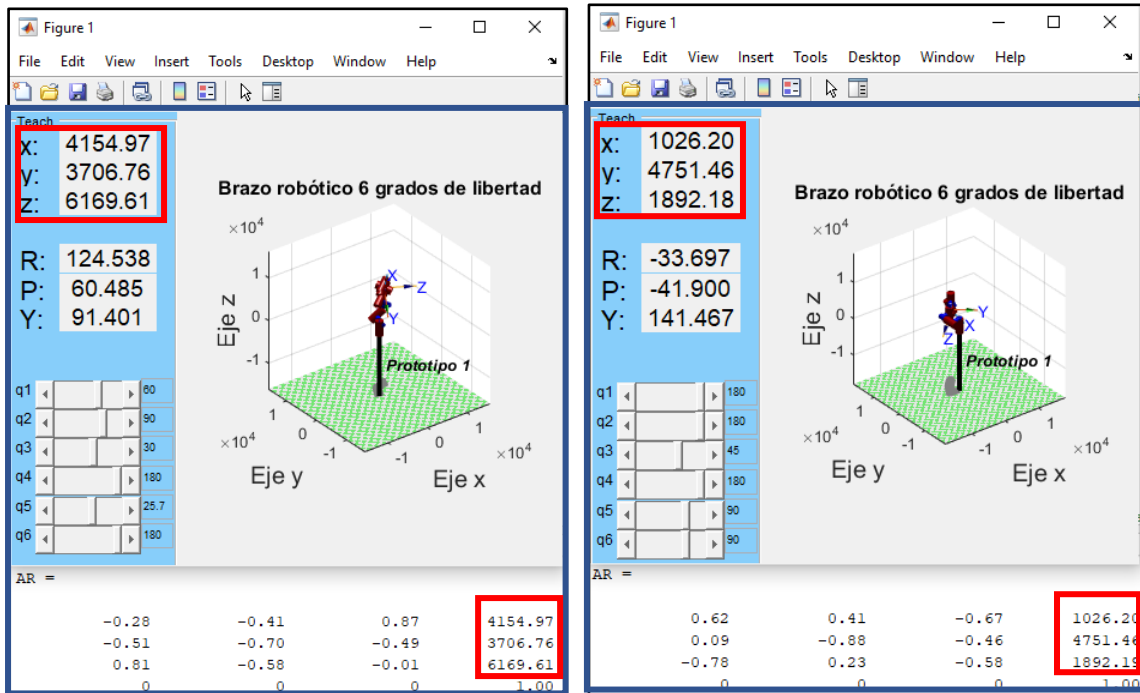


Figura 3-40: Prueba 1 y 2 de posicionamiento.

(Fuente: Propia)

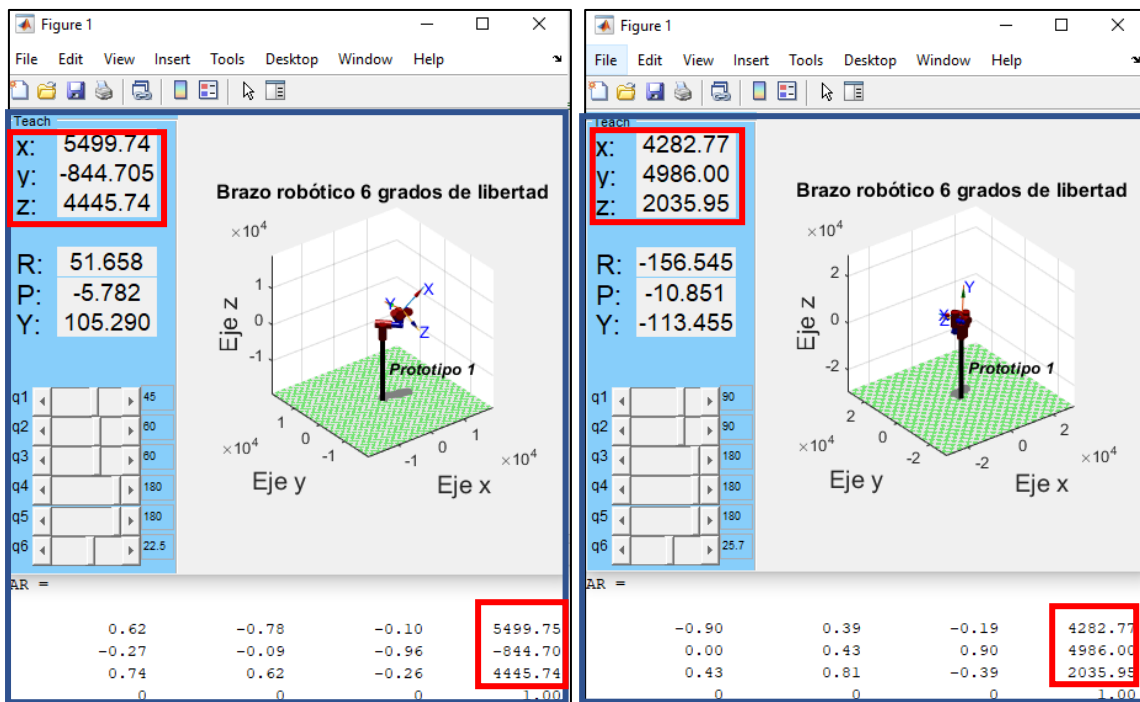


Figura 3-41: Prueba 3 y 4 de posicionamiento.

(Fuente: Propia)

Tabla 3-19 Prueba de coordenadas efector final.

(Fuente: Propia)

Prueba	Descripción	CUMPLE	NO CUMPLE
Coordenada x	Coordenada x del efector final mostrado en el menú de slides y en la componente de la matriz Denavit-Hartenberg.	√	
Coordenada y	Coordenada y del efector final mostrado en el menú de slides y en la componente de la matriz Denavit-Hartenberg.	√	
Coordenada z	Coordenada z del efector final mostrado en el menú de slides y en la componente de la matriz Denavit-Hartenberg.	√	
Coordenadas en la matriz Denavit-Hartenberg.	Verificación de las coordenadas del efector final según los componentes T_x , T_y y T_z de la matriz Denavit-Hartenberg.	√	

3.5.3. Prueba menú slides

Tabla 3-20: Prueba menú slides.

(Fuente: Propia)

Prueba	Descripción	CUMPLE	NO CUMPLE
Slides	Los slides permiten el movimiento deseado en cada uno de los grados de libertad dentro del espacio x, y y z .	√	
Coordenadas	Se muestran las coordenadas x, y y z en la interfaz.	√	
Ángulos	Se presentan los ángulos explicados	√	

Prueba	Descripción	CUMPLE	NO CUMPLE
	Roll, Pitch y Yaw.		
Límites	Los límites deben estar bien dimensionados para la correcta visualización del robot.	√	

3.5.4. Validación de la simulación

Para demostrar que el proyecto realizado en simulación funciona correctamente, se valida con los datos del proyecto de titulación (Nikolas, 2014). Para esto se ha simulado en el programa realizado para este proyecto, el robot de 6 grados de libertad con todos los parámetros de Denavit-Hartenberg plasmados en el mencionado trabajo de titulación, con la finalidad de verificar los mismos resultados finales del efector final en la interfaz realizada y la matriz resultante.

Por tanto, en la página 98 del trabajo de titulación mencionado se tienen los siguientes parámetros de Denavit-Hartenberg.

Tabla 3-21: Validación simulación.

(Andrés Chang, Robot Articular, 2014)

GDL	Ángulo Teta	Distancia d	Distancia a	Ángulo Alfa
1	-90°	300	0	90°
2	180°	200	400	180°
3	90°	100	0	90°
4	0°	400	0	0°
5	0°	100	0	-90°
6	0°	0	0	90°

Obteniendo en la página 102 del mismo trabajo, las coordenadas en x, y y z en el espacio del efector final tal y como se observa en la simulación realizada para este trabajo.

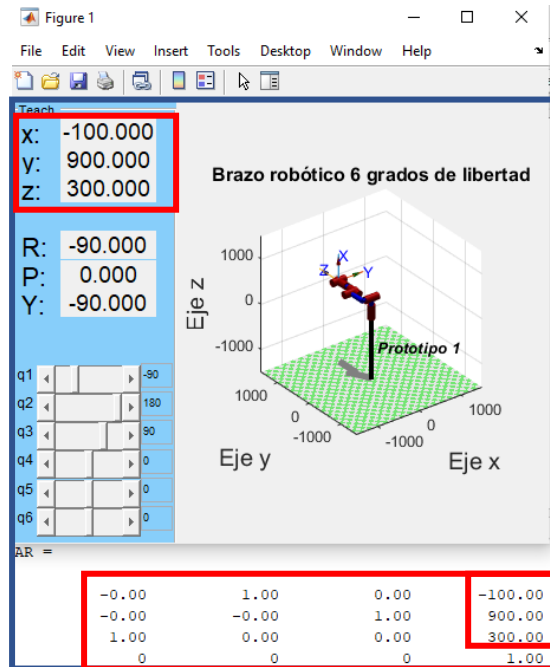


Figura 3-42: Valores resultantes - validación simulación.

(Fuente: Propia)

3.6. Manual de usuario

En el Anexo 6 se encuentran las instrucciones para la descarga y correcta instalación del software Matlab y de la librería Toolbox Robotics. En esta sección se muestra paso a paso la programación del robot para el presente proyecto según la información impartida durante el trabajo.

Programación

Dentro de esta sección se indica como realizar un robot de 6 grados de libertad utilizando el software Matlab en base a lo que se ha explicado y en función del código mostrado en el Anexo 5. A continuación, se muestran los pasos correspondientes mediante un diagrama de flujo:

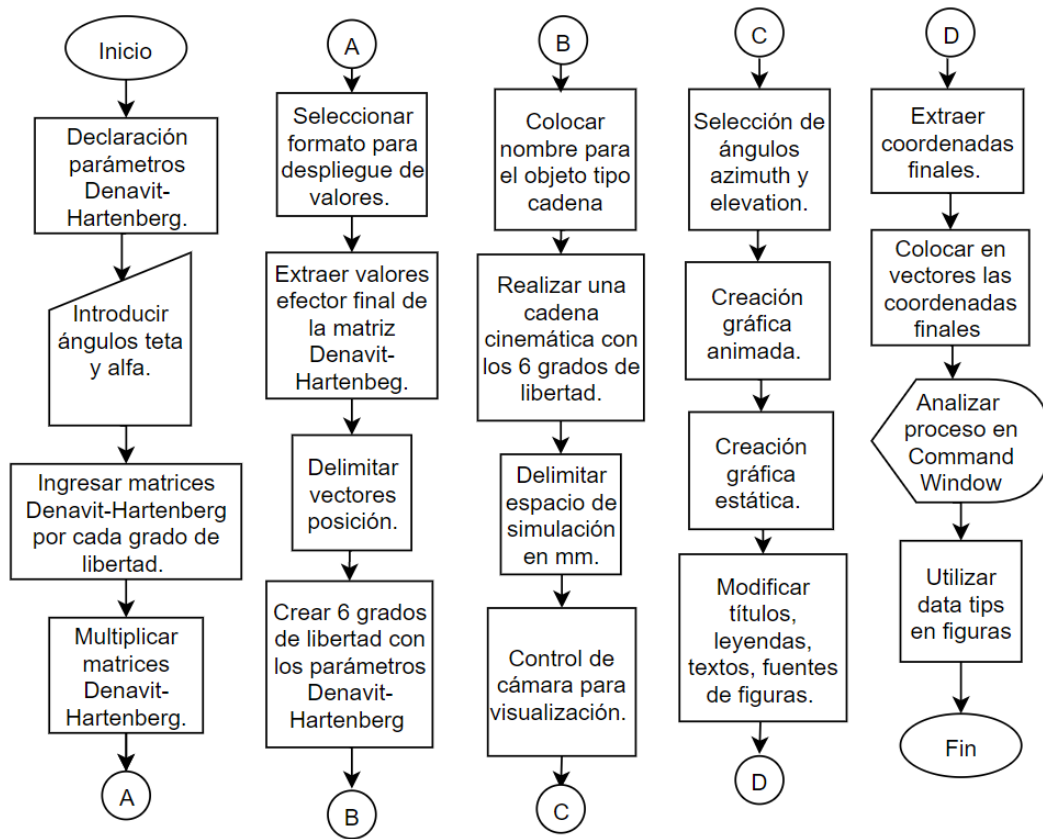


Figura 3-43: Diagrama de Flujo.

(Fuente: Propia)

A continuación, se detalla como iniciar la programación según lo visto en el flujograma:

1. Iniciar Matlab y crear un script para empezar a programar.

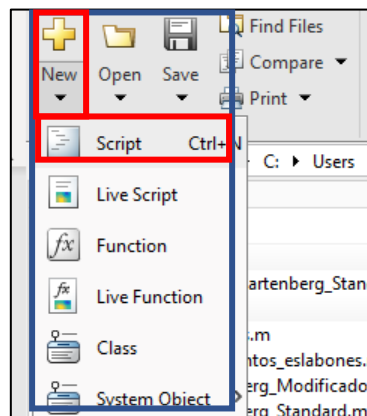


Figura 3-44: Ventana nuevo script - Manual de usuario.

(Fuente: Propia)

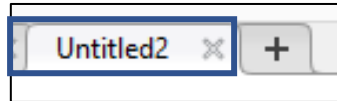


Figura 3-45 Pestaña nuevo script en curso - Manual de usuario

(Fuente: Propia)

2. Según las directrices propuestas a lo largo del trabajo, se debe empezar a colocar los comandos necesarios para inicializar el programa, los parámetros para la creación de los grados de libertad, las líneas necesarias para los valores de entrada para Denavit-Hartenberg y los comandos para el ploteo respectivo del robot.

Si todavía no está claro como realizar el código, revisar el Anexo 5 donde se encuentra detallado el código y los comentarios respectivos para el correcto análisis.

```
123  
124 - A_01 [cos(teta_1), -sin(teta_1), 0, a_1;  
125     sin(teta_1)*cos(alfa_1), cos(alfa_1)*cos(teta_1), -sin(alfa_1), d_1*sin(alfa_1);  
126     sin(alfa_1)*sin(teta_1), cos(teta_1)*sin(alfa_1), cos(alfa_1), d_1;  
127     0, 0, 0, 1]  
128
```

Figura 3-46 Fragmento del código - Etapa de programación.

(Fuente: Propia)

3. Después de haber codificado, en la pestaña de “Editor” presionar el botón “Run”, guardar el archivo y verificar si no existen errores de programación.

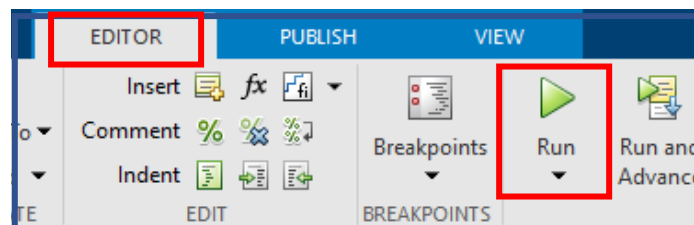


Figura 3-47: Inicio para correr el programa.

(Fuente: Propia)

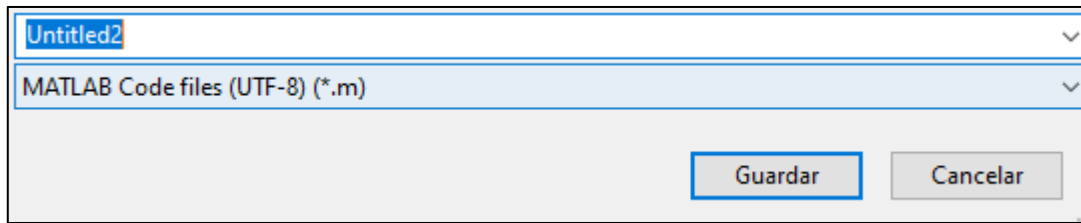


Figura 3-48: Guardar el algoritmo.

(Fuente: Propia)

- Una vez ejecutado el programa, Matlab solicita los ángulos correspondientes teta y alfa, los cuales deben ser ingresados en radianes.

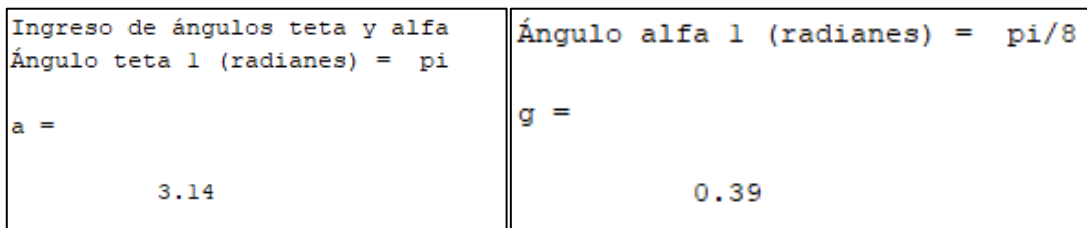


Figura 3-49: Ejemplificación - Datos de entrada.

(Fuente: Propia)

- Cuando los seis ángulos teta y alfa hayan sido ingresados el programa debe seguir corriendo, las matrices y los parámetros de Denavit-Hartenberg se ejecutan correctamente desplegando las dos figuras correspondientes, la estática y la animada con sus respectivos slides.

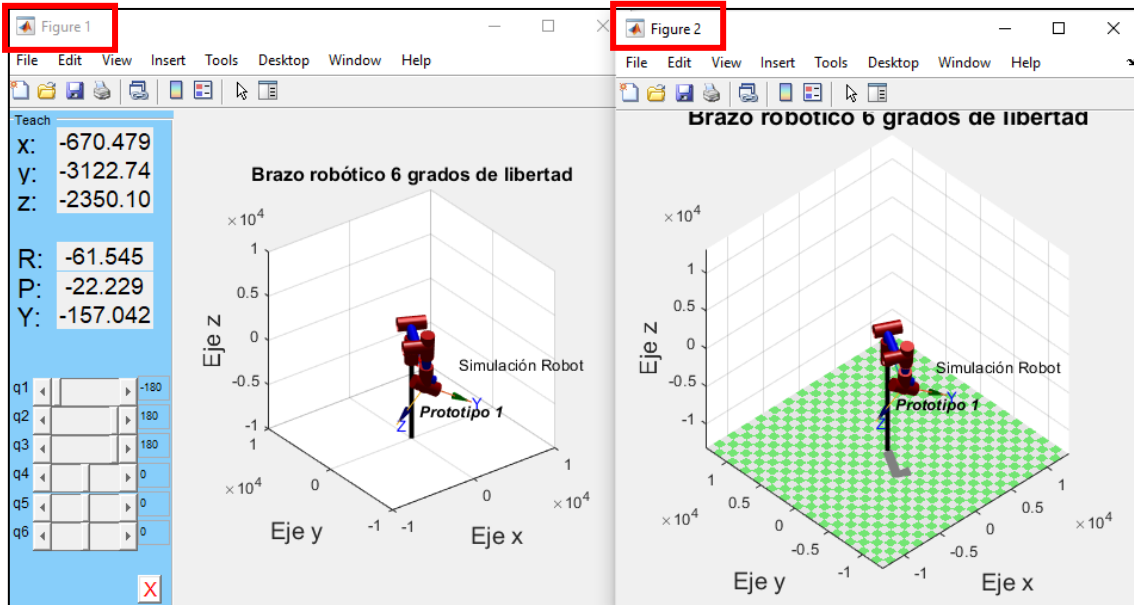


Figura 3-50: Figuras desplegadas - Ejecución de código.

(Fuente: Propia)

- Finalmente se comprueba los parámetros T_x, T_y y T_z dado por la matriz resultante en el "Command Window" y las coordenadas en el espacio en x, y y z mostradas en el menú de slides correspondientes a la posición final del efector final.

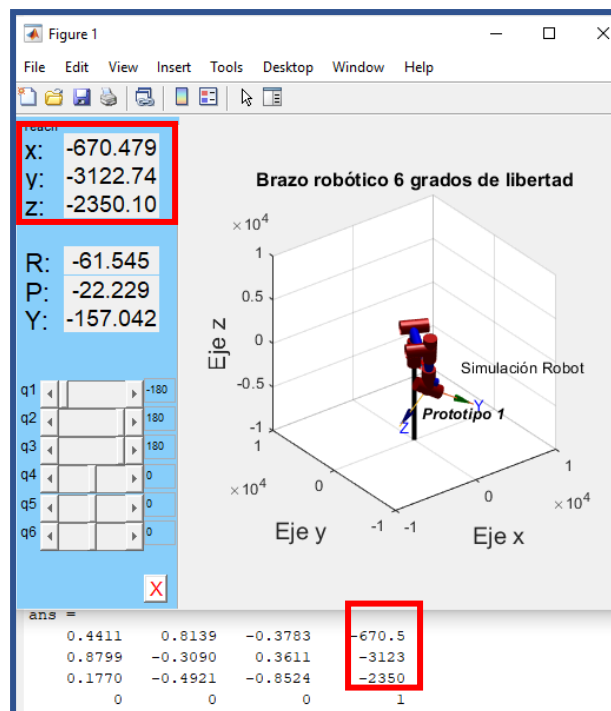


Figura 3-51: Verificación parámetros Denavit Hartenberg.

(Fuente: Propia)

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- El espacio de trabajo y las herramientas básicas dentro del software Matlab, le permiten al programador realizar operaciones básicas con vectores y matrices, como también el correcto dimensionamiento, configuración y arreglo de figuras para las correspondientes gráficas de funciones dentro del programa.
- La simulación de un robot es muy compleja, pero gracias a la librería de Peter Corke, Toolbox Robotics presenta un listado de comandos simplificados y sencillos para que los programadores tengan la posibilidad de recrear un robot en función de los parámetros de Denavit-Hartenberg con la menor cantidad de líneas posibles.
- El procedimiento para obtención de los parámetros de Denavit-Hartenberg puede ser realizado, con el producto separado de cada matriz de transformación o únicamente con el manejo de su matriz resultante.
- La interfaz animada, es el medio principal de interacción entre el usuario y el código fuente del robot, que dispone de los recursos para visualizar coordenadas finales, ángulos finales del efector final y los slides correspondientes a cada grado de libertad.
- Mediante las pruebas de funcionamiento de matrices, código, gráficas e interfaz, se determinó el correcto funcionamiento del robot.
- El manual de usuario se llevó a cabo con la finalidad de que cualquier persona con escasos conocimientos en programación, uso del software Matlab y de robótica, sea capaz de entender y simular un robot aplicando de los parámetros de Denavit Hartenberg.
- La diferencia entre el método Denavit-Hartenberg estándar y modificado es la asignación de los sistemas de coordenadas. En el método Denavit-Hartenberg estándar se asignan los sistemas de coordenadas a los eslabones, en cambio,

al método Denavit-Hartenberg modificado se aplican los sistemas de coordenadas a las articulaciones. En consecuencia, el producto matricial resultante para los dos métodos debe ser en la misma secuencia de movimientos respecto al último sistema de coordenadas ya sea eslabón o articulación.

- Con el fin de otorgar un mayor conocimiento en el campo de la robótica, el estudiante que desee continuar esta investigación podrá implementar un módulo práctico para que los estudiantes puedan a través de los cálculos realizados en el software y la simulación, replicar de manera física los movimientos del robot.

4.2. Recomendaciones

- La Escuela de Formación de Tecnólogos (ESFOT) tiene como misión: “Preparar profesionales con una sólida formación académica, humanística, conceptual y práctica”, por lo que se estima que la enseñanza básica de una herramienta como Matlab, es de gran utilidad para las actividades académicas dentro de las carreras ofertadas por la facultad.
- La Escuela Politécnica Nacional, facilita a su comunidad, el acceso a las herramientas básicas para sus actividades académicas y científicas, otorgando la licencia del software Matlab, en virtud de esto se motiva a la Escuela de Formación de Tecnólogos (ESFOT), la enseñanza de este programa, que debe ser aplicado en cualquier área.
- La Escuela de Formación de Tecnólogos (ESFOT) dispone de laboratorios para fines de estudio en campos eléctricos, electrónicos, sistemas micro procesados y control automático industrial, por lo cual, se motiva a los alumnos próximos a graduarse a implementar un módulo básico de robótica, para que los docentes tengan la capacidad impartir los conocimientos elementales de este campo, entregando a los estudiantes una visión más amplia del campo industrial.
- Se recomienda analizar en Matlab un robot comercial, que permita el análisis de la cinemática de este mecanismo.

5. BIBLIOGRAFÍA

- Anónimo. (2015). *Inteligencia Robótica*. Recuperado el 30 de 07 de 2020, de <https://bit.ly/2PgWMwJ>
- Anónimo. (2016). *Robótica y generalidades*. Recuperado el 02 de 08 de 2020, de <https://bit.ly/2PgJrQi>
- Carrera, E. M. (2015 - 2016). *La creciente automatización de los puestos de trabajo*. (Universidad Miguel Hernandez) Recuperado el 15 de 08 de 2020, de <https://bit.ly/2CuGUPQ>
- Componentes, e. (26 de 01 de 2018). *Ventajas Matlab*. Recuperado el 17 de 08 de 2020, de <https://bit.ly/31ZYG6b>
- Corke, P. (2017). *Robotics Toolbox*. Recuperado el 15 de 08 de 2020, de <https://bit.ly/31SKwUE>
- Fernandez, C. C. (s.f.). *Manual Básico para Matlab*. Obtenido de Manual Básico para Matlab: <https://bit.ly/2IJHtrE>
- García, J. M. (15 de 09 de 2015). *Robótica Educativa. La programación como parte de un proceso*. (FLACSO) Recuperado el 15 de 08 de 2020, de <https://bit.ly/33ZFaJP>
- Guzman, F. A. (2012). *LA ROBÓTICA COMO UN RECURSO PARA FACILITAR EL APRENDIZAJE Y DESARROLLO DE COMPETENCIAS GENERALES*. (Universidad de Salamanca) Recuperado el 15 de 08 de 2020, de <https://bit.ly/3iOnTXZ>
- Huatuco, V. P. (03 de 2014). *Actualidad y perspectiva de la robótica*. Recuperado el 15 de 08 de 2020, de <https://bit.ly/30ZURyQ>
- Moreno Iveth, M. L. (2012). *LA ROBÓTICA EDUCATIVA, UNA HERRAMIENTA PARA LA ENSEÑANZA*. (Universidad de Salamanca) Recuperado el 15 de 08 de 2020, de <https://bit.ly/2Y242ww>
- Nikolas, C. H. (08 de 2014). *DISEÑO Y SIMULACIÓN DE UN ROBOT ARTICULAR CON SEIS GRADOS DE LIBERTAD UTILIZANDO EL TOOLBOX ROBOTICS DE MATLAB PARA FORALECER LAS CLASES TEÓRICAS REALIZANDO PRÁCTICAS DE LABORATORIO CON EL SOFTWARE PRESENTADO EN ESTE TRABAJO*. Obtenido de DISEÑO Y SIMULACIÓN DE UN ROBOT ARTICULAR CON SEIS GRADOS DE LIBERTAD UTILIZANDO EL TOOLBOX ROBOTICS DE MATLAB PARA FORALECER LAS CLASES TEÓRICAS REALIZANDO PRÁCTICAS DE LABORATORIO CON EL SOFTWARE PRESENTADO EN ESTE TRABAJO: <https://bit.ly/3nl8IHl>
- Robotics, D. (s.f.). *Robótica y Automatización*. (2018) Recuperado el 15 de 08 de 2020,

de <https://bit.ly/2Y24yuM>

Torriti, M. T. (11 de 12 de 2011). *Cinemática directa y procedimiento de Denavit Hartenberg*. (Universidad Católica de Chile / Departamento de Ingeniería Eléctrica) Recuperado el 15 de 08 de 2020, de <https://bit.ly/2E6gKDs>

Universidad de, O. (2010 - 2011). *Manual uso Matlab*. Recuperado el 15 de 08 de 2020, de <https://bit.ly/31ZRgzQ>

Villanueva, M. S. (07 de 2018). *Modelado y simulación dinámica del robot industrial Staübli Unimation PUMA 560*. (Universidad Politécnica de Valencia) Recuperado el 09 de 09 de 2020, de <https://bit.ly/3marPFk>

6. ANEXOS

ANEXO 1:

COMPROBACIÓN ALGORITMO DENAVIT-HARTENBERG

```

%Modelo Standard

%Para la representacion Denavit-hartenberg en Cinematica Directa se
requieren 4 parámetros: a(i), alfa(i), teta(i), d(i)
%• 2 relativos a la forma y tamaño del eslabón: a(i), alfa(i)
%• 2 describen posición relativa del eslabón respecto a su
predecesor: teta(i), d(i)

%Los parámetros de forma y tamaño quedan determinados en tiempo de
diseño. Los parámetros de posición relativa varían
%• teta(i) es variable si la rotación es articular (d(i) Constante)
%• d(i) variable si la rotación es prismática (teta(i) Constante)

%El archivo en MATLAB DENAVIT_MATRIZ demuestra la funcionalidad de la
Matriz de transformación homogénea del metodo DENAVIT-HARTENBERG.

%A partir de los parámetros de Denavit-Hartenberg (teta, d, a, alfa)
se cuenta con cuatro matrices principales

%1. De rotacion angular teta (R_teta);
%2. De desplazamiento d (D_d);
%3. De desplazamiento a (D_a) y
%4. De rotacion angular alfa (R_alfa)

%Donde al final la matriz A será el resultado del producto de estas
cuatro matrices demostrando el metodo DENAVIT-HARTENBERG

%Igualmente se tiene la matriz B que es la representacion directa del
%metodo DENAVIT-HARTENBERG (Matriz directa)

%teta : ángulo que existe entre las líneas normales de la articulación
i si se cortasen en el mismo punto del eje i
%d : distancia entre las intersecciones de las normales comunes al eje
i, medida a lo largo de i
%a : (longitud eslabón) distancia entre ejes i, i+1 de las
articulaciones a lo largo de la perpendicular común
%alfa : (ángulo torsión) ángulo que existe entre ejes i,i+1 si se
cortasen en punto de corte de la perpendicular común

%-----
----
% Comandos antes de la inicialización del programa

clear; % Comando para borrar todas las variables creadas en el
workspace
close all; % Comando para cerrar todas las pestañas de Matlab y
empezar desde cero
clc; % Borrar todos los procesos de Command Window

%-----
----

% Datos para las matrices

teta = -pi/2; % Ejemplo de ángulo teta
alfa = pi/2; % Ejemplo de ángulo alfa
a=0; % Ejemplo de longitud a
d=3000; % Ejemplo de longitud d

```

```

%-----
%-----

% Construcción de las matrices de transformación

% Primera matriz de rotación en el eje z con un ángulo teta R(z,?)
R1 =[cos(teta), -sin(teta), 0, 0;
     sin(teta), cos(teta), 0, 0;
     0, 0, 1, 0;
     0, 0, 0, 1];

% Segunda matriz de desplazamiento en el eje z con una distancia d
D(z,d)

D1 =[1, 0, 0, 0;
     0, 1, 0, 0;
     0, 0, 1, d;
     0, 0, 0, 1];

% Tercera matriz de desplazamiento en el eje x con una distancia a
D(x,a)

D2 =[1, 0, 0, a;
     0, 1, 0, 0;
     0, 0, 1, 0;
     0, 0, 0, 1];

% Cuarta matriz de rotación en el eje x con un ángulo alfa R(x,?)

R2 =[1, 0, 0, 0;
     0, cos(alfa), -sin(alfa), 0;
     0, sin(alfa), cos(alfa), 0;
     0, 0, 0, 1];

% Matriz resultante de las cuatro matrices de transformación
(Multiplicación de todas)
% A = R1*D1*D2*R2

A = R1*D1*D2*R2

% Matriz directa de Denavit-Hartenberg
B= [cos(teta), -cos(alfa)*sin(teta), sin(alfa)*sin(teta), a*cos(teta);
    sin(teta), cos(alfa)*cos(teta), -sin(alfa)*cos(teta), a*sin(teta);
    0, sin(alfa), cos(alfa), d;
    0, 0, 0, 1]

%end

```

ANEXO 2:

COMPROBACIÓN ALGORITMO DENAVIT-HARTENBERG
(MODIFICADO)

```

% Modelo modificado

%Para la representacion Denavit-hartenberg en Cinematica Directa se
requieren 4 parámetros: a(i), alfa(i), teta(i), d(i)
%• 2 relativos a la forma y tamaño del eslabón: a(i), alfa(i)
%• 2 describen posición relativa del eslabón respecto a su
predecesor: teta(i), d(i)

%Los parámetros de forma y tamaño quedan determinados en tiempo de
diseño. Los parámetros de posición relativa varían
%• teta(i) es variable si la rotación es articular (d(i) Constante)
%• d(i) variable si la rotación es prismática (teta(i) Constante)

%El archivo en MATLAB DENAVIT_MATRIZ demuestra la funcionalidad de la
Matriz de transformación homogénea del metodo DENAVIT-HARTENBERG.

%A partir de los parámetros de Denavit-Hartenberg (teta, d, a, alfa)
se cuenta con cuatro matrices principales

%1. De rotacion angular teta (R_teta);
%2. De desplazamiento d (D_d);
%3. De desplazamiento a (D_a) y
%4. De rotacion angular alfa (R_alfa)

%Donde al final la matriz A será el resultado del producto de estas
cuatro matrices demostrando el metodo DENAVIT-HARTENBERG

%Igualmente se tiene la matriz B que es la representacion directa del
%metodo DENAVIT-HARTENBERG (Matriz directa)

%teta : ángulo existe entre las líneas normales de la articulación i
si se cortasen en el mismo punto del eje i
%d : distancia entre las intersecciones de las normales comunes al eje
i, medida a lo largo de i
%a : (longitud eslabón) distancia entre ejes i, i+1 de las
articulaciones a lo largo de la perpendicular común
%alfa : (ángulo torsión) ángulo que existe entre ejes i,i+1 si se
cortasen en punto de corte de la perpendicular común

%-----
----
% Comandos antes de la inicialización del programa

clear; % Comando para borrar todas las variables creadas en el
workspace
close all; % Comando para cerrar todas las pestañas de Matlab y
empezar desde cero
clc; % Borrar todos los procesos de Command Window

%-----
----

% Datos para las matrices

teta = -pi/2; % Ejemplo de ángulo teta
alfa = pi/2; % Ejemplo de ángulo alfa
a=0; % Ejemplo de longitud a
d=3000; % Ejemplo de longitud d

```

```

%-----
%-----

% Construcción de las matrices de transformación

% Primera matriz de rotación en el eje x con un ángulo alfa R(x,?)

R1 =[1, 0, 0, 0;
     0, cos(alfa), -sin(alfa), 0;
     0, sin(alfa), cos(alfa), 0;
     0, 0, 0, 1];

% Segunda matriz de desplazamiento en el eje x con una distancia a
D(x,a)

D1 =[1, 0, 0, a;
     0, 1, 0, 0;
     0, 0, 1, 0;
     0, 0, 0, 1];

% Tercera matriz de rotación en el eje z con un ángulo teta R(z,?)
R2 =[cos(teta), -sin(teta), 0, 0;
     sin(teta), cos(teta), 0, 0;
     0, 0, 1, 0;
     0, 0, 0, 1];

% Cuarta matriz de desplazamiento en el eje z con una distancia d
D(z,d)

D2 =[1, 0, 0, 0;
     0, 1, 0, 0;
     0, 0, 1, d;
     0, 0, 0, 1];

% Matriz resultante de las cuatro matrices de transformación
(Multiplicación de todas)
% A = R1*D1*D2*R2

A = R1*D1*R2*D2

% Matriz directa de Denavit-Hartenberg

B = [cos(teta), -sin(teta), 0, a;
     sin(teta)*cos(alfa), cos(teta)*cos(alfa), -sin(alfa), -
sin(alfa)*d;
     sin(alfa)*sin(teta), cos(teta)*sin(alfa), cos(alfa), cos(alfa)*d;
     0, 0, 0, 1]

%end

```

ANEXO 3:

CÓDIGO ROBOT 6 GRADOS DE LIBERTAD EN MATLAB
(APLICACIÓN DE TRAYECTORIAS)


```

%Iniciar el Programa
startup_rvc; % Reconocimiento de los comandos necesarios dentro de la
librería de Peter Corke
% y poder realizar el simulado y planteado de un robot de n grados de
% libertad
%-----
-----
% Determino el tiempo deseado para el cambio entre cada trayectoria
t=0:.06:3; %tiempo de movimiento
% Formato [Tiempo inicial: saltos de n en n entre cada valor : Tiempo
final]
% Nota: Si se reduce los pasos entre valores la trayectoria del robot
será
% más lenta, gracias a esto se ve más detallado su movimiento en los
% 3 planos.
%-----
-----
%Se define los puntos que se requiere que el robot siga, para esto la
%cantidad de valores dentro del vector irá de acuerdo a los grados de
%libertad del robot.

% A continuación se podrá colocar las posiciones que se desee, como
% recomendación es mejor colocar en la posición cero, para de esta
manera
% observar de mejor manera la trayectoria del robot

% Formato
% posicionx=[ x Brazo1 x Brazo2 x Brazo3 x Brazo4 x Brazo5 x Brazo6]

posicion0=[0 0 0 0 0 0]
posicion1=[-2.6762 3.1416 1.5708 0 0 0];
posicion2=[-2.6762 2 0.853 0 0 0];
posicion3=[-0.953 3.1416 -0.43 0 0 0];

%-----
-----

%Implementación de los n eslabones respectivos para el robot

%Para poder determinar cada uno de estos existen tres formatos
diferentes
%pero con todos ellos se cumple la misma función; Realizar los
eslabones
%necesarios según Denavit Hartenberg

    %Lx = Link([0 1.2 0.3 pi/2]);
    %Lx = Link('revolute', 'd', 1.2, 'a', 0.3, 'alpha', pi/2);
    %Lx = Revolute('d', 1.2, 'a', 0.3, 'alpha', pi/2);

% Un objeto realizado con el comando Link contiene toda la información
% relacionada con la articulación y enlace del robot

% Formato Link([theta=q, d=0, a=0, alpha=0])
% Para este caso se tiene un robot de 6 grados de libertad, por lo que
A
% continuación se determinan sus 6 articulaciones

```

```

articulacion1 = Link([-pi/2 3000 0 pi/2]); %Valores correspondientes a
la articulación 1

articulacion2 = Link([pi 200 4000 pi]); %Valores correspondientes a la
articulación 2

articulacion3 = Link([pi/2 1000 0 pi/2]); %Valores correspondientes a
la articulación 3

articulacion4 = Link([0 4000 0 0]); %Valores correspondientes a la
articulación 4

articulacion5 = Link([0 1000 0 -pi/2]); %Valores correspondientes a la
articulación 5

articulacion6 = Link([0 0 0 pi/2]); %Valores correspondientes a la
articulación 6

%-----
----

% Unión de todas las articulaciones respectivas

% SerialLink = Representa un objeto tipo brazo robótico que está
conformado
% por enlace en serie de varias articulaciones

%SerialLink (n1 n2 n3 n4 n5 n6)

brazo_robot=SerialLink([articulacion1 articulacion2 articulacion3
articulacion4 articulacion5 articulacion6]);

% Nota: brazo_robot= Nombre del objeto creado para unir todas las
% articulaciones

%-----
----

% Colocación del nombre al robot
brazo_robot= SerialLink (brazo_robot, 'name', 'Prototipo 1' )

% Formato
% Objeto= SerialLink (Objeto, 'name', 'Nombre deseado')

%-----
----

% Determinación de trayectorias en función de las posiciones

% Formato nombre_trayectoria = jtraj (xinicial, xfinal, vector de
tiempo)

% Gracias al vector asignado de tiempo se pueda pasar en pasos más
pequeños
% e ir planteando el movimiento del robot desde las posiciones
iniciales
% hasta las posiciones finales configuradas

trayectoria1=jtraj(posicion0,posicion1,t); % Trayectoria 1 configurada

```

```

trayectoria2=jtraj(posicion1,posicion2,t); % Trayectoria 2 configurada
trayectoria3=jtraj(posicion2,posicion3,t); % Trayectoria 3 configurada
trayectoria4=jtraj(posicion3,posicion0,t); % Trayectoria 4 configurada

%-----
-----

% Ploteo respectivo del robot
% Con los siguientes comandos se ploteará cada trayectoria
correspondiente
% configurada para el robot

figure % Comando para agregar una nueva figura y poder empezar a
plotear

title('Brazo robótico 6 grados de libertad', 'FontSize',15)% Comando
para
%colocar título al gráfico 1 en movimiento automático
%y el respectivo tamaño de la letra
% Formato title('Nombre deseado', 'FontSize', Tamaño de letra)

grid on % Comando para agregar cuadrícula dentro de la figura
respectiva

text(2000,-5000,'Simulación Robot') % Comando para colocar una
leyenda dentro
% del plano correspondiente en las coordenadas deseadas
%Formato
% text(x,y,'Nombre deseado')

% Formato objeto.plot(posición o trayectoria deseada)

brazo_robot.plot(posicion0) % Posición cero del robot

%label--> Comando para colocar nombre a los respectivos ejes con un
tamaño
%de letra deseado
%Formato xlabel('Nombre deseado','FontSize', Tamaño de letra)

xlabel('Eje x','FontSize',15) % Nombrar al eje x y tamaño de letra
para figura 1 en movimiento

ylabel('Eje y','FontSize',15) % Nombrar al eje y y tamaño de letra
para figura 1 en movimiento

zlabel('Eje z','FontSize',15) % Nombrar al eje z y tamaño de letra
para figura 1 en movimiento

brazo_robot.plot(trayectoria1); % Trayectoria 1 ploteado en los planos
brazo_robot.plot(trayectoria2); % Trayectoria 2 ploteado en los planos
brazo_robot.plot(trayectoria3); % Trayectoria 3 ploteado en los planos
brazo_robot.plot(trayectoria4); % Trayectoria 4 ploteado en los planos

```

```

%-----
%-----

% Límites para visualización del robot en el plano

%Establecer los límites de visualización final después que haya
completado todos los movimientos
%Estos límites servirán para determinar el tamaño del robot en el
plano en
%la viusalización final después que el robot haya completado las
%trayectorias deseadas

xlim([-10000,10000]) % Límite de visualización final del robot en el
plano x

ylim([-10000,10000]) % Límite de visualización final del robot en el
plano y

zlim([-10000,10000]) % Límite de visualización final del robot en el
plano z

%Nota: Las coordenadas descritas en:

    %xlim([-10000,10000])

    %ylim([-10000,10000])

    %zlim([-10000,10000]);

% Como se mencionó son las medidas finales en las que se va a ver todo
el
% plano después del movimiento del robot, por lo que es necesario que
resulte
% igual al plano mientras el robot se está moviendo, como en el plano
% inicial aparece  $10^4=10000$ , las coordenadas que tienen que ser
introducidas
% tanto en X,Y ,Z, son esas, si no se realiza esto el robot no se va a
centrar
% correctamente en el plano siendo este o muy grande o muy pequeño en
todas
% sus dimensiones

%-----
%-----

% Control de la cámara dentro del plano

view(150,30); % Comando para control de la visión de cámara dentro de
los planos de los gráficos

% En este caso el gráfico presentado al final el robot con los botones
% interactivos correspondientes

%-----
%-----

% Función .teach (Figura 1- Gráfica cinemática)

```

```

brazo_robot.teach() % Comando para plotear el robot de forma
cinemática

% Comando para desplegar un panel de control gráfico y poder controlar
al robot manualmente
% dentro de los tres planos con botones interactivos

% Presionar el botón de la x para salir de la simulación}

title('Brazo robótico 6 grados de libertad', 'FontSize',12)% Comando
para
%colocar título al gráfico 1 y el respectivo tamaño de la letra
% Formato title('Nombre deseado', 'FontSize', Tamaño de letra)

%label--> Comando para colocar nombre a los respectivos ejes con un
tamaño
%de letra deseado
%Formato xlabel('Nombre deseado','FontSize', Tamaño de letra)

xlabel('Eje x','FontSize',15) % Nombrar al eje x y tamaño de letra
para figura 1

ylabel('Eje y','FontSize',15) % Nombrar al eje y y tamaño de letra
para figura 1

zlabel('Eje z','FontSize',15) % Nombrar al eje z y tamaño de letra
para figura 1

%-----
----

% Función figure (Figura 2 - Gráfica estática)

figure % Comando para agregar una nueva figura y poder empezar a
plotear

brazo_robot.plot(posicion0)% Comando para plotear el brazo robótico de
forma
% estática

title('Brazo robótico 6 grados de libertad', 'FontSize',15)% Comando
para
%colocar título al gráfico 2 y el respectivo tamaño de la letra
% Formato title('Nombre deseado', 'FontSize', Tamaño de letra)

%label--> Comando para colocar nombre a los respectivos ejes con un
tamaño
%de letra deseado
%Formato xlabel('Nombre deseado','FontSize', Tamaño de letra)

xlabel('Eje x','FontSize',15) % Nombrar al eje x y tamaño de letra
para figura 2

ylabel('Eje y','FontSize',15) % Nombrar al eje y y tamaño de letra
para figura 2

zlabel('Eje z','FontSize',15) % Nombrar al eje z y tamaño de letra
para figura 2

```

```
grid on % Comando para agregar cuadrícula dentro de la figura
respectiva

text(2000,-5000,'Simulación Robot') % Comando para colocar una
leyenda dentro
% del plano correspondiente en las coordenadas deseadas
%Formato
% text(x,y,'Nombre deseado')

%Fin
```

ANEXO 4:

CÓDIGO PARA LA SOLICITUD DE DATOS EN COMMAND WINDOW

```

%-----

% Comandos antes de la inicialización del programa

clear; % Comando para borrar todas las variables creadas en el
workspace
close all; % Comando para cerrar todas las pestañas de Matlab y
empezar desde cero
clc; % Borrar todos los procesos de Command Window

%-----

% Entrada de datos

disp('Ingreso de ángulos teta y alfa') % Comando para mostrar un
título de
%los datos que se solicitan.

%Valor solicitado para el ángulo teta 1 en radianes
a=input('Ángulo teta 1 (radianes) = ')

%Valor solicitado para el ángulo teta 2 en radianes
b=input('Ángulo teta 2 (radianes) = ')

%Valor solicitado para el ángulo teta 3 en radianes
c=input('Ángulo teta 3 (radianes) = ')

%Valor solicitado para el ángulo teta 4 en radianes
d=input('Ángulo teta 4 (radianes) = ')

%Valor solicitado para el ángulo teta 5 en radianes
e=input('Ángulo teta 5 (radianes) = ')

%Valor solicitado para el ángulo teta 6 en radianes
f=input('Ángulo teta 6 (radianes) = ')

%Valor solicitado para el ángulo alfa 1 en radianes
g=input('Ángulo alfa 1 (radianes) = ')

%Valor solicitado para el ángulo alfa 2 en radianes
h=input('Ángulo alfa 2 (radianes) = ')

%Valor solicitado para el ángulo alfa 3 en radianes
i=input('Ángulo alfa 3 (radianes) = ')

%Valor solicitado para el ángulo alfa 4 en radianes
j=input('Ángulo alfa 4 (radianes) = ')

%Valor solicitado para el ángulo alfa 5 en radianes
k=input('Ángulo alfa 5 (radianes) = ')

%Valor solicitado para el ángulo alfa 6 en radianes
l=input('Ángulo alfa 6 (radianes) = ')

% end

```


ANEXO 5:

CÓDIGO ROBOT 6 GRADOS DE LIBERTAD EN MATLAB
(MATICES DENAVIT-HARTENBERG - STANDARD)

```

%Para la representacion Denavit-hartenberg en Cinematica Directa se
requieren 4 parámetros: a(i), alfa(i), teta(i), d(i)
%• 2 relativos a la forma y tamaño del eslabón: a(i), alfa(i)
%• 2 describen posición relativa del eslabón respecto a su
predecesor: teta(i), d(i)

%Los parámetros de forma y tamaño quedan determinados en tiempo de
diseño. Los parámetros de posición relativa varían
%• teta(i) es variable si la rotación es articular (d(i) Constante)
%• d(i) variable si la rotación es prismática (teta(i) Constante)

% Se demuestra la funcionalidad de la Matriz de transformación
homogénea del método DENAVIT-HARTENBERG.

%A partir de los parámetros de Denavit-Hartenberg (teta, d, a, alfa)
se cuenta con cuatro matrices principales

%teta : ángulo existiría entre las líneas normales de la articulación
i si se cortasen en el mismo punto del eje i
%d : distancia entre las intersecciones de las normales comunes al eje
i, medida a lo largo de i
%a :(longitud eslabón) distancia entre ejes i, i+1 de las
articulaciones a lo largo de la perpendicular común
%alfa :(ángulo torsión) ángulo que existiría entre ejes i,i+1 si se
cortasen en punto de corte de la perpendicular común

%-----
----

% Comandos antes de la inicialización del programa

clear; % Comando para borrar todas las variables creadas en el
workspace
close all; % Comando para cerrar todas las pestañas de Matlab y
empezar desde cero
clc; % Borrar todos los procesos de Command Window

%-----
----

% Ingreso de los ángulos teta y alfa para los parámetros de DENAVIT
% HARTENBERG

disp('Ingreso de ángulos teta y alfa') % Comando para mostrar un
título de
%los datos que se solicitan.

%Valor solicitado para el ángulo teta 1 en radianes
a=input('Ángulo teta 1 (radianes) = ')

%Valor solicitado para el ángulo teta 2 en radianes
b=input('Ángulo teta 2 (radianes) = ')

%Valor solicitado para el ángulo teta 3 en radianes
c=input('Ángulo teta 3 (radianes) = ')

%Valor solicitado para el ángulo teta 4 en radianes

```

```

d=input('Ángulo teta 4 (radianes) = ')

%Valor solicitado para el ángulo teta 5 en radianes
e=input('Ángulo teta 5 (radianes) = ')

%Valor solicitado para el ángulo teta 6 en radianes
f=input('Ángulo teta 6 (radianes) = ')

%Valor solicitado para el ángulo alfa 1 en radianes
g=input('Ángulo alfa 1 (radianes) = ')

%Valor solicitado para el ángulo alfa 2 en radianes
h=input('Ángulo alfa 2 (radianes) = ')

%Valor solicitado para el ángulo alfa 3 en radianes
i=input('Ángulo alfa 3 (radianes) = ')

%Valor solicitado para el ángulo alfa 4 en radianes
j=input('Ángulo alfa 4 (radianes) = ')

%Valor solicitado para el ángulo alfa 5 en radianes
k=input('Ángulo alfa 5 (radianes) = ')

%Valor solicitado para el ángulo alfa 6 en radianes
l=input('Ángulo alfa 6 (radianes) = ')

%-----
-----
%Parámetros de Denavit Hartenberg
%Nota: Pueden ser modificados a consideración del usuario, hay que
tomar en
%cuenta que los datos de ángulos hay que ingresarlos en radianes

%Valores ángulo teta
teta_1= a; % Valor de ángulo teta del eslabón 1
teta_2= b; % Valor de ángulo teta del eslabón 2
teta_3= c; % Valor de ángulo teta del eslabón 3
teta_4= d; % Valor de ángulo teta del eslabón 4
teta_5= e; % Valor de ángulo teta del eslabón 5
teta_6= f; % Valor de ángulo teta del eslabón 6

%Valores ángulo alfa

alfa_1= g; % Valor de ángulo alfa del eslabón 1
alfa_2= h; % Valor de ángulo alfa del eslabón 2
alfa_3= i; % Valor de ángulo alfa del eslabón 3
alfa_4= j; % Valor de ángulo alfa del eslabón 4
alfa_5= k; % Valor de ángulo alfa del eslabón 5
alfa_6= l; % Valor de ángulo alfa del eslabón 6

%Valores distancias d (Distancia a lo largo del eje Z)

d_1=3000; % Valor d para eslabón 1
d_2=200; % Valor d para eslabón 2
d_3=1000; % Valor d para eslabón 3
d_4=4000; % Valor d para eslabón 4

```

```

d_5=1000; % Valor d para eslabón 5
d_6=0; % Valor d para eslabón 6

%Valores distancias a

a_1=0; % Valor a para eslabón 1
a_2=4000; % Valor a para eslabón 2
a_3=0; % Valor a para eslabón 3
a_4=0; % Valor a para eslabón 4
a_5=0; % Valor a para eslabón 5
a_6=0; % Valor a para eslabón 6

% Nota: Las distancias "a" y "d" de los parámetros de Denavit
Hartenber se
% los puede cambiar, pero deben estar en función de los límites que se
% aplique a la figura final, caso contrario no se podrá visulizar
% correctamente el robot implementado

%-----
----
%Matrices algoritmo Denavit-Hartenberg

% Matriz del punto 0 al 1 (A0-1)

A_01 = [cos(teta_1), -cos(alfa_1)*sin(teta_1),
sin(alfa_1)*sin(teta_1), a_1*cos(teta_1);
sin(teta_1), cos(alfa_1)*cos(teta_1), -sin(alfa_1)*cos(teta_1),
a_1*sin(teta_1);
0, sin(alfa_1), cos(alfa_1), d_1;
0, 0, 0, 1];
% Matriz del punto 1 al 2 (A1-2)

A_12 = [cos(teta_2), -cos(alfa_2)*sin(teta_2),
sin(alfa_2)*sin(teta_2), a_2*cos(teta_2);
sin(teta_2), cos(alfa_2)*cos(teta_2), -sin(alfa_2)*cos(teta_2),
a_2*sin(teta_2);
0, sin(alfa_2), cos(alfa_2), d_2;
0, 0, 0, 1];
% Matriz del punto 2 al 3 (A2-3)

A_23 = [cos(teta_3), -cos(alfa_3)*sin(teta_3),
sin(alfa_3)*sin(teta_3), a_3*cos(teta_3);
sin(teta_3), cos(alfa_3)*cos(teta_3), -sin(alfa_3)*cos(teta_3),
a_3*sin(teta_3);
0, sin(alfa_3), cos(alfa_3), d_3;
0, 0, 0, 1];

% Matriz del punto 3 al 4 (A3-4)

A_34 = [cos(teta_4), -cos(alfa_4)*sin(teta_4),
sin(alfa_4)*sin(teta_4), a_4*cos(teta_4);
sin(teta_4), cos(alfa_4)*cos(teta_4), -sin(alfa_4)*cos(teta_4),
a_4*sin(teta_4);
0, sin(alfa_4), cos(alfa_4), d_4;
0, 0, 0, 1];

% Matriz del punto 4 al 5 (A4-5)

```

```

A_45 = [cos(teta_5), -cos(alfa_5)*sin(teta_5),
sin(alfa_5)*sin(teta_5), a_5*cos(teta_5);
sin(teta_5), cos(alfa_5)*cos(teta_5), -sin(alfa_5)*cos(teta_5),
a_5*sin(teta_5);
0, sin(alfa_5), cos(alfa_5), d_5;
0, 0, 0, 1];

% Matriz del punto 5 al 6 (A5-6)

A_56 = [cos(teta_6), -cos(alfa_6)*sin(teta_6),
sin(alfa_6)*sin(teta_6), a_6*cos(teta_6);
sin(teta_6), cos(alfa_6)*cos(teta_6), -sin(alfa_6)*cos(teta_6),
a_6*sin(teta_6);
0, sin(alfa_6), cos(alfa_6), d_6;
0, 0, 0, 1];

% Matriz resultante AR = A0-1*A1-2*A2-3*A3-4*A4-5*A5-6

AR=A_01*A_12*A_23*A_34*A_45*A_56
%-----
----

% Formato de número

format bank % Comando para observar los números con menos decimales ya
que las
%matrices son muy extensas.

%-----
----

% Componentes x, y, z del efector final

% Nota: En esta sección se extrae los términos x, y, z de la matriz de
% Denavit-Hartenberg

% Coordenada en x del efector final
x=AR(1,4)
% Coordenada en y del efector final
y=AR(2,4)
% Coordenada en z del efector final
z=AR(3,4)

%-----
----

%Inicializar los comandos para el robot
startup_rvc; % Reconocimiento de los comandos necesarios dentro de la
librería de Peter Corke
% y poder realizar el simulado y ploteado de un robot de n grados de
% libertad
%-----
----

%Posiciones del robot

% Nota: La posición donde se encontrará el efector final, gracias al
% algortimo de Denavit-Hartenberg.

```

```

posicion = [0, 0, 0, 0, 0, 0]; % Posición 0 para inicializar el vector
posicion_1 = [a, b, c, d, e, f]; %Extraer las variables de la matriz
Denavit-Hartenberg
%para que el robot tome la posición correspondiente.
%-----
-----

%Implementación de los "n" eslabones respectivos para el robot

%Para poder determinar cada uno de estos existen tres formatos
diferentes
%pero con todos ellos se cumple la misma función; Realizar los
eslabones
%necesarios según Denavit Hartenberg

    %Lx = Link([teta d a alfa]);

% Un objeto realizado con el comando Link contiene toda la información
% relacionada con la articulación y enlace del robot

% Formato Link([theta=q, d=0, a=0, alpha=0])
% Para este caso se tiene un robot de 6 grados de libertad, por lo que
a
% continuación se determinan sus 6 articulaciones

articulacion1 = Link([a d_1 a_1 g]) %Valores correspondientes a la
articulación 1

articulacion2 = Link([b d_2 a_2 h]) %Valores correspondientes a la
articulación 2

articulacion3 = Link([c d_3 a_3 i]) %Valores correspondientes a la
articulación 3

articulacion4 = Link([d d_4 a_4 j]) %Valores correspondientes a la
articulación 4

articulacion5 = Link([e d_5 a_5 k]) %Valores correspondientes a la
articulación 5

articulacion6 = Link([f d_6 a_6 l]) %Valores correspondientes a la
articulación 6

%-----
-----

% Unión de todas las articulaciones respectivas

% SerialLink = Representa un objeto tipo brazo robótico que está
conformado
% por enlace en serie de varias articulaciones

%SerialLink (n1 n2 n3 n4 n5 n6)

```

```

brazo_robot=SerialLink([articulacion1 articulacion2 articulacion3
articulacion4 articulacion5 articulacion6])

% Nota: brazo_robot= Nombre del objeto creado para unir todas las
% articulaciones

%-----
----

% Colocación del nombre al robot
brazo_robot= SerialLink (brazo_robot, 'name', 'Prototipo 1' )

% Formato
% Objeto= SerialLink (Objeto, 'name', 'Nombre deseado')

%-----
----

% Ploteado de robot estático

figure % Comando para agregar una nueva figura y poder empezar a
plotear

title('Brazo robótico 6 grados de libertad', 'FontSize',15)% Comando
para
%colocar título al gráfico 1 en movimiento automático
%y el respectivo tamaño de la letra
% Formato title('Nombre deseado', 'FontSize', Tamaño de letra)

grid on % Comando para agregar cuadrícula dentro de la figura
respectiva

text(2000,-5000,'Simulación Robot') % Comando para colocar una
leyenda dentro
% del plano correspondiente en las coordenadas deseadas
%Formato
% text(x,y,'Nombre deseado')

% Formato objeto.plot(posición o trayectoria deseada)

brazo_robot.plot(posicion_1) % Posición cero del robot

%label--> Comando para colocar nombre a los respectivos ejes con un
tamaño
%de letra deseado
%Formato xlabel('Nombre deseado','FontSize', Tamaño de letra)

xlabel('Eje x','FontSize',15) % Nombrar al eje x y tamaño de letra
para figura 1 en movimiento

ylabel('Eje y','FontSize',15) % Nombrar al eje y y tamaño de letra
para figura 1 en movimiento

zlabel('Eje z','FontSize',15) % Nombrar al eje z y tamaño de letra
para figura 1 en movimiento

%-----
----

```

```

% Límites para visualización del robot en el plano

%Establecer los límites de visualización final después que haya
completado todos los movimientos
%Estos límites servirán para determinar el tamaño del robot en el
plano en
%la viusalización final después que el robot haya completado las
%trayectorias deseadas

xlim([-10000,10000]) % Límite de visualización final del robot en el
plano x

ylim([-10000,10000]) % Límite de visualización final del robot en el
plano y

zlim([-10000,10000]) % Límite de visualización final del robot en el
plano z

%Nota: Las coordenadas descritas en:

    %xlim([-10000,10000])

    %ylim([-10000,10000])

    %zlim([-10000,10000]);

% Como se mencionó son las medidas finales en las que se va a ver todo
el
% plano después del movimiento del robot, por lo que es necesario que
resulte
% igual al plano mientras el robot se está moviendo, como en el plano
% inicial aparece  $10^4=10000$ , las coordenadas que tienen que ser
introducidas
% tanto en X,Y ,Z, son esas, si no se realiza esto el robot no se va a
centrar
% correctamente en el plano siendo este o muy grande o muy pequeño en
todas
% sus dimensiones

%-----
%-----

% Control de la cámara dentro del plano

    view(150,30)% Comando para control de la visión de cámara dentro de
los planos de los gráficos

% En este caso el gráfico presentado al final el robot con los botones
% interactivos correspondientes

%-----
%-----

% Función .teach (Figura 1- Gráfica cinemática)

brazo_robot.teach(posicion_1) % Comando para plotear el robot de forma
cinemática

```



```

% Comando para desplegar un panel de control gráfico y poder controlar
al robot manualmente
% dentro de los tres planos con botones interactivos

% Presionar el botón de la x para salir de la simulación}

title('Brazo robótico 6 grados de libertad', 'FontSize',12)% Comando
para
%colocar título al gráfico 1 y el respectivo tamaño de la letra
% Formato title('Nombre deseado', 'FontSize', Tamaño de letra)

%label--> Comando para colocar nombre a los respectivos ejes con un
tamaño
%de letra deseado
%Formato xlabel('Nombre deseado','FontSize', Tamaño de letra)

xlabel('Eje x','FontSize',15) % Nombrar al eje x y tamaño de letra
para figura 1

ylabel('Eje y','FontSize',15) % Nombrar al eje y y tamaño de letra
para figura 1

zlabel('Eje z','FontSize',15) % Nombrar al eje z y tamaño de letra
para figura 1

%-----
----

% Función figure (Figura 2 - Gráfica estática)

figure % Comando para agregar una nueva figura y poder empezar a
plotear

brazo_robot.plot(posicion_1)% Comando para plotear el brazo robótico
de forma
% estática

title('Brazo robótico 6 grados de libertad', 'FontSize',15)% Comando
para
%colocar título al gráfico 2 y el respectivo tamaño de la letra
% Formato title('Nombre deseado', 'FontSize', Tamaño de letra)

%label--> Comando para colocar nombre a los respectivos ejes con un
tamaño
%de letra deseado
%Formato xlabel('Nombre deseado','FontSize', Tamaño de letra)

xlabel('Eje x','FontSize',15) % Nombrar al eje x y tamaño de letra
para figura 2

ylabel('Eje y','FontSize',15) % Nombrar al eje y y tamaño de letra
para figura 2

zlabel('Eje z','FontSize',15) % Nombrar al eje z y tamaño de letra
para figura 2

```

```

grid on % Comando para agregar cuadrícula dentro de la figura
respectiva

text(2000,-5000,'Simulación Robot') % Comando para colocar una
leyenda dentro
% del plano correspondiente en las coodernadas deseadas
%Formato
% text(x,y,'Nombre deseado')

%-----
-----
%Verificación

% Nota: Con el siguiente comando se podrá verificar según los
parámetros de
% Denavit Hartenberg asignados al inicio, si el efector final del
robot se
% ubica en las posiciones adecuadas. El comando
% "brazo_robot.fkine(posicion)", mostrará con sus tres componentes de
la
% columna final, las coordenadas que se indica en la interfaz animada
% generada por el comando "teach".

AR_1=brazo_robot.fkine(posicion_1) %Comando de la librería Toolbox
Robotics
%que arroja un valor de clase SE3.

%-----
-----

% Data tips

% Con la ayuda de los data tips se puede colocar con el cursor del
mouse en
% un punto del robot en el espacio para conocer su posición, en este
caso
% la posición final del efector final.

AR_2=double(AR_1) % Transformación del valor clase SE3 a una matriz
4x4 y
%extraer los valores de correspondientes a las coordenadas x, y, z.

Xf= [AR_2(1,4) 0 0 0 0 0 ] % Vector para data tip coordenada en X
Yf= [AR_2(2,4) 0 0 0 0 0 ] % Vector para data tip coordenada en Y
Zf= [AR_2(3,4) 0 0 0 0 0 ] % Vector para data tip coordenada en Z

% Nota: Para el proceso de data tips es necesario realizar esto ya que
como
% el efector final tiene su propio eje de referencia, si se coloca el
mouse
% sobre el eje de coordenadas del efector final indicará 0 en x, y ,
z.,
% por lo que se extraen los valores correspondientes en forma de
vector y asignarlos al data tips de la simulación.
%Fin

```

ANEXO 6:

INSTRUCCIONES PARA INSTALCIÓN DE MATLAB Y LIBRERÍA TOOLBOX
ROBOTICS

Dentro de los beneficios de ser estudiante de la ESCUELA POLITÉCNICA NACIONAL es que se dispone de varias herramientas de software con sus respectivas licencias para las diferentes carreras y aplicaciones dentro del conocimiento impartido, por lo que MATLAB no es la excepción por ser una herramienta completa que se utiliza en varias carreras ya que ofrece un sinnúmero de librerías para ciencias mecánicas, electrónicas, eléctricas, físicas, matemáticas, etc. A continuación, se presenta el proceso para su descarga e instalación con licencia por medios institucionales, así también la librería necesaria para este trabajo de robótica.

VERIFICACIÓN DEL SISTEMA OPERATIVO

El programa que se descargó para este trabajo es MATLAB y para el caso de Windows necesita un sistema operativo de 64 bits, caso contrario ocurre un error al momento de descomprimir el archivo, este error se lo presenta más adelante. Para verificar el sistema operativo se debe tomar las siguientes consideraciones:

1. Dentro del buscador del computador se colocará "Este equipo", y se da click en la aplicación.

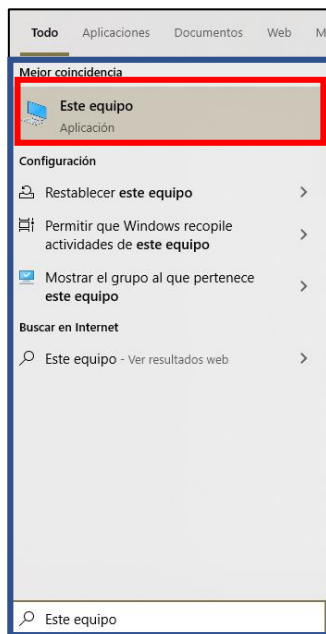


Figura 6-1: Buscador "Este equipo".

(Fuente: Propia)

2. En la parte izquierda aparecen muchas carpetas propias del computador y

correspondiente a datos personales, se debe buscar de nuevo “Este equipo” y se selecciona propiedades.

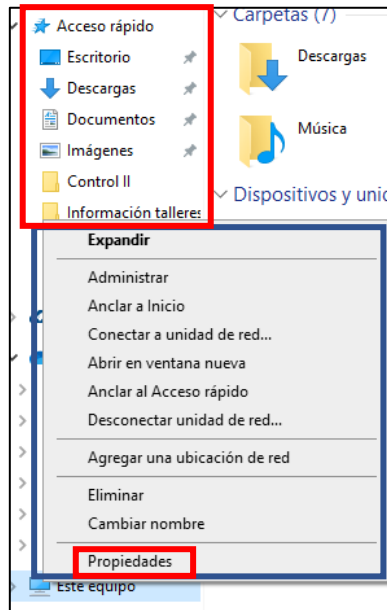


Figura 6-2: Propiedades del equipo.

(Fuente: Propia)

3. Dentro de esta sección aparece toda la información técnica del computador y dentro del apartado del sistema se observa la característica necesaria para la descarga de MATLAB, debe decir sistema operativo de 64 bits, procesador X64.

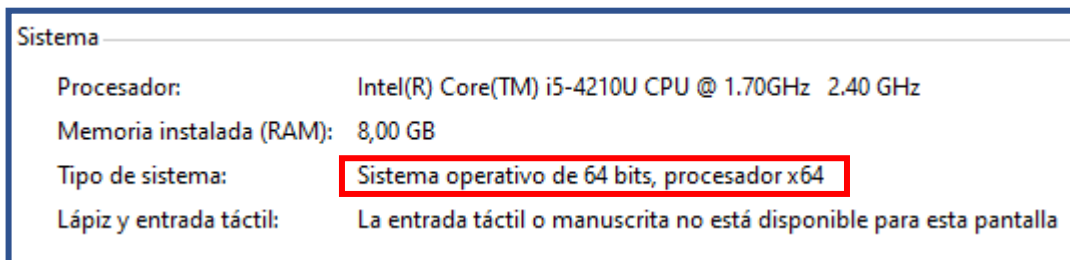


Figura 6-3 Características computador

(Fuente: Propia)

DESCARGA DEL SOFTWARE MATLAB

1. En el buscador de su preferencia colocar el siguiente link donde redirige a la

página oficial de la ESCUELA POLITÉCNICA NACIONAL donde se observa los programas ofrecidos por la institución.

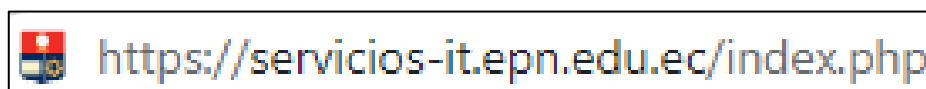


Figura 6-4 Enlace página DGIP (Descarga del software).

(Fuente: Propia)

2. Cuando se encuentre en la página oficial de la DGIP se debe presionar el acceso de “Descargas”, como se aprecia en la siguiente figura.

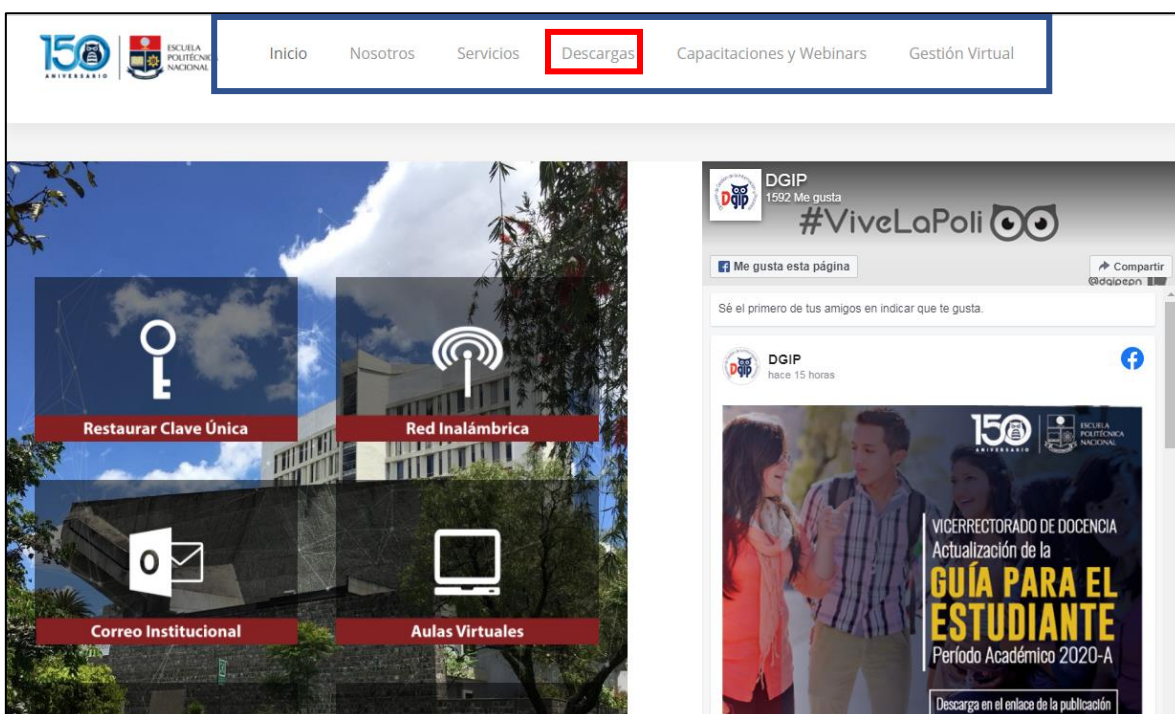


Figura 6-5: Sección descarga DGIP.

(Fuente: Propia)

3. Una vez posicionado en la sección de descargas se visualiza la página respectiva y las ofertas que ofrece la institución de softwares y servicios para

almacenamiento.







 <p>IBM Cloud Académico</p> <p>Aproveche el poder de IBM. Obtenga acceso fácil y gratuito a las herramientas que necesita para desarrollar grandes cosas. Disfrute de potentes recursos técnicos y estratégicos de IBM. Acceda a la nube de IBM y utilice servicios potentes, así como las tecnologías informáticas de código abierto más destacadas de la industria, o aproveche los recursos prácticos que le enseñarán sobre datos y análisis, Internet de las cosas y seguridad, etc. como son IBM Cloud, Ciencia de datos, IBM Watson, Blockchain, Power Systems entre otras.</p> <p>La Iniciativa Académica de IBM permite y alienta a profesores, estudiantes e investigadores de la Escuela Politécnica Nacional aprovechar las herramientas, cursos y otros recursos de IBM en el aula. Nuestra misión es permitir que los estudiantes se gradúen con experiencia práctica directa en herramientas de "fuerza industrial" que los ayudarán en la carrera elegida.</p> <p>INGRESO</p> <p>MANUAL DE INGRESO</p>	 <p>Microsoft Azure Dev Tools for Teaching*</p> <p>Las herramientas de informática y desarrollo Microsoft y los recursos de aprendizaje que anteriormente formaban parte de la cuenta de Microsoft Imagine ahora están disponibles como Microsoft Azure Dev Tools for Teaching, y el ingreso lo realiza a través de las credenciales de su correo institucional (clave única).</p> <p>Esta plataforma ofrece a los estudiantes las herramientas, los recursos y las experiencias que necesita para mejorar sus conocimientos con vistas a su desarrollo profesional. Ya se trate de crear un juego, diseñar una aplicación o lanzar un proyecto, Microsoft Dev Tools for Teaching ayuda a la comunidad politécnica a desarrollar sus ideas y hacerlas realidad.</p> <p>Mediante su cuenta Microsoft Dev Tools for Teaching usted tendrá acceso a herramientas de Inteligencia Artificial y Machine Learning, Analisis de datos, Ofimática y productividad, Sistemas operativos, Bases de datos, Herramientas de desarrollo, IoT, Sistemas de Middleware, Redes, Seguridad, Servicios de aprendizaje, entre otros.</p> <p>INGRESO</p> <p>MANUAL DE INGRESO</p>	 <p>Office365 Pro Plus*</p> <p>Microsoft Office, donde quiera que vaya. Obtenga las últimas herramientas de productividad, colaboración, cumplimiento normativo entre otras, actualizadas con regularidad.</p> <p>Office 365 ProPlus incluye las aplicaciones de Office para PC y Mac en sus últimas versiones, para ser instaladas gratuitamente en hasta 5 dispositivos a través de su correo institucional (clave única).</p> <p>Office Pro Plus incluye:</p> <ul style="list-style-type: none">• Conjunto de programas de Office para PC y Mac• Office en PC, tabletas y teléfonos• Almacenamiento y uso compartido de archivos mediante One Drive DE 1TB• Office Online entre otras. <p>* Office 365 Pro Plus y los beneficios de Office 365 se encuentran ya activados por defecto a través de su cuenta de correo institucional.</p> <p>DESCARGA</p> <p>MANUAL DE DESCARGA</p>
 <p>Matlab*</p> <p>Herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la</p>	 <p>Minitab*</p> <p>Software profesional que incluye todos los recursos estadísticos que usted necesita para todos los cursos, desde básicos hasta avanzados, Minitab es el software que más se utiliza para enseñar</p>	 <p>Statgraphics</p> <p>Potente herramienta de análisis de datos que combina una amplia gama de procedimientos analíticos con extraordinarios gráficos interactivos para proporcionar un entorno integrado de análisis</p>

Figura 6-6: Ofertas DGIP.

(Fuente: Propia)

4. Cuando se haya encontrado el software MATLAB, se ofrece una pequeña

descripción acerca de su contenido y su aplicación.



MATLAB

Matlab*

Herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. 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 y la comunicación con programas en otros lenguajes y con otros dispositivos de hardware.

Mediante su cuenta de Mathworks, usted tiene acceso a:

- Descargas y activación de Matlab + 40 Toolboxes.
- Acceso a recursos de entrenamiento gratuitos y de pago mediante la academia Matlab.
- Matlab Drive (Permite almacenar de manera segura sus archivos MATLAB desde cualquier lugar).

[PORTAL MATHWORKS](#)

[MANUAL DE INSTALACIÓN](#)

Figura 6-7: Acceso directo para descarga MATLAB.

(Fuente: Propia)

5. Dentro del espacio descriptivo para la descarga de MATLAB se encuentra con dos botones, el primero lleva a la página de los creadores de MATLAB y el otro es un acceso directo para el manual de instalación del software, se debe dar click en "PORTAL MATHWORKS".

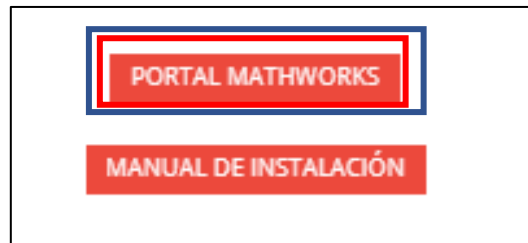


Figura 6-8: Acceso PORTAL MATHWORKS.

(Fuente: Propia)

6. A continuación, la página de MathWorks permite seleccionar el idioma de su preferencia según su ubicación, en este caso español de América Latina, pero si desea las demás instrucciones y en si el programa en otro idioma puede hacerlo.

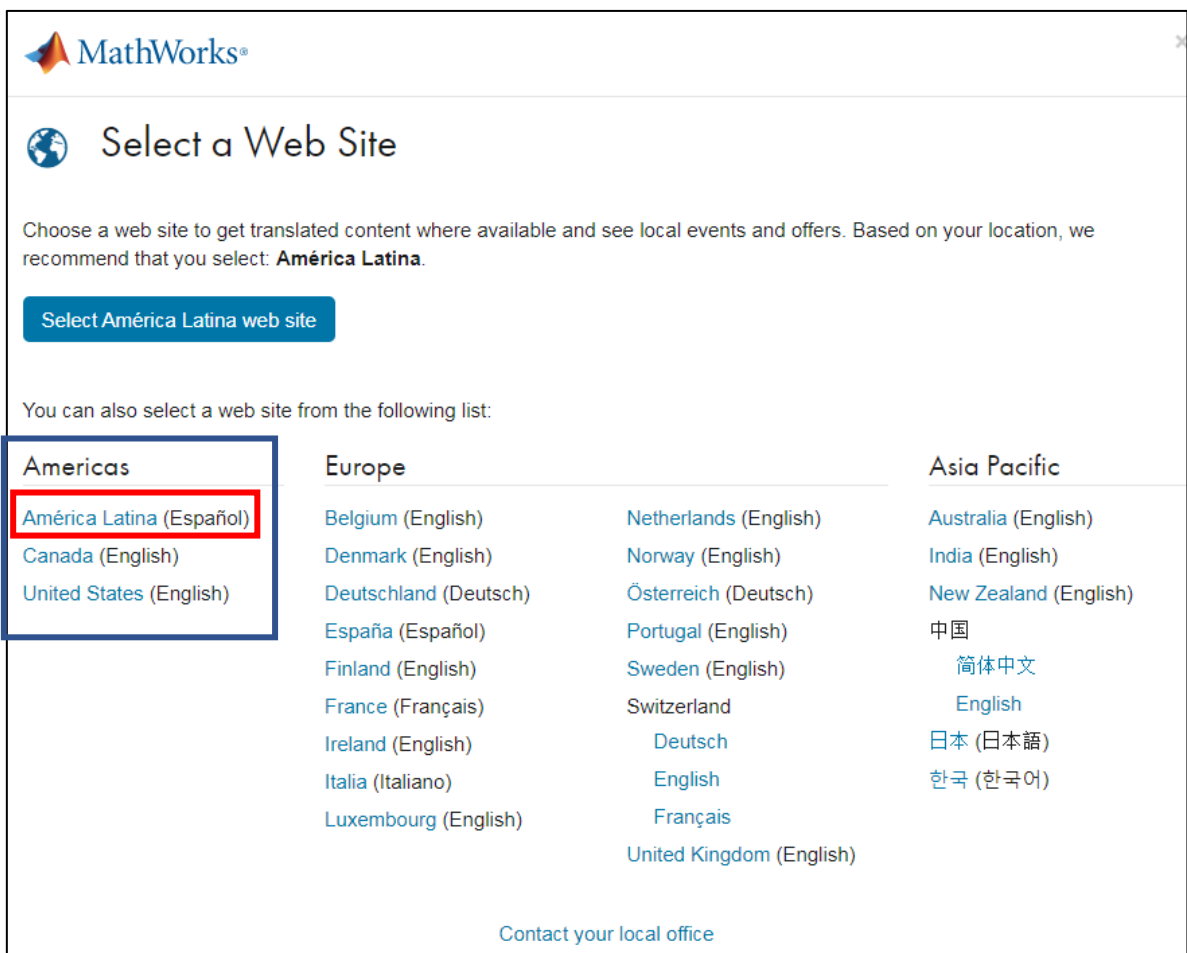


Figura 6-9: Idioma según ubicación MathWorks.

(Fuente: Propia)

7. En la página se verifica el acceso exclusivo que se tiene para personal y alumnos de la ESCUELA POLITÉCNICA NACIONAL.



Figura 6-10: Acceso para personal y alumnos EPN.

(Fuente: Propia)

8. En la misma página se encuentra el acceso para la descarga de MATLAB el programa que interesa para este trabajo y SIMULINK, complemento de MATLAB que permite la programación por medio de bloques, se dará click en “Inicie sesión para empezar a utilizarlos”.

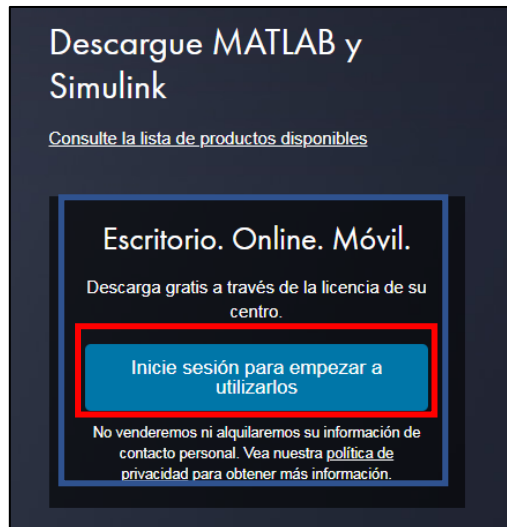


Figura 6-11: Descarga de MATLAB y SIMULINK.

(Fuente: Propia)

9. En los siguientes campos se tiene que introducir los datos personales correspondientes a la institución, como se observa en las siguientes figuras se debe iniciar sesión con el correo institucional y con la clave correspondiente, esto debido a que MATHWORKS detecta que es un correo de la ESCUELA POLITÉCNICA NACIONAL y permite descargar el programa con la licencia gratuita.

Figura 6-12: Ingreso de correo institucional

(Fuente: Propia)

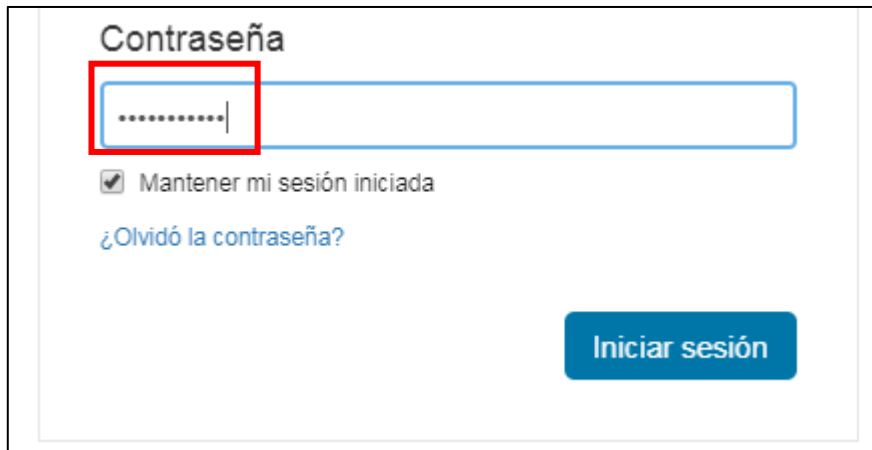


Figura 6-13 Ingreso de la clave del correo institucional.

(Fuente: Propia)

- Una vez ingresadas las credenciales correspondientes, se inicia sesión en la página oficial de MATHWORKS para empezar con el proceso de descarga del instalador con su respectiva licencia, se debe dar click en la pestaña “Consiga MATLAB”.

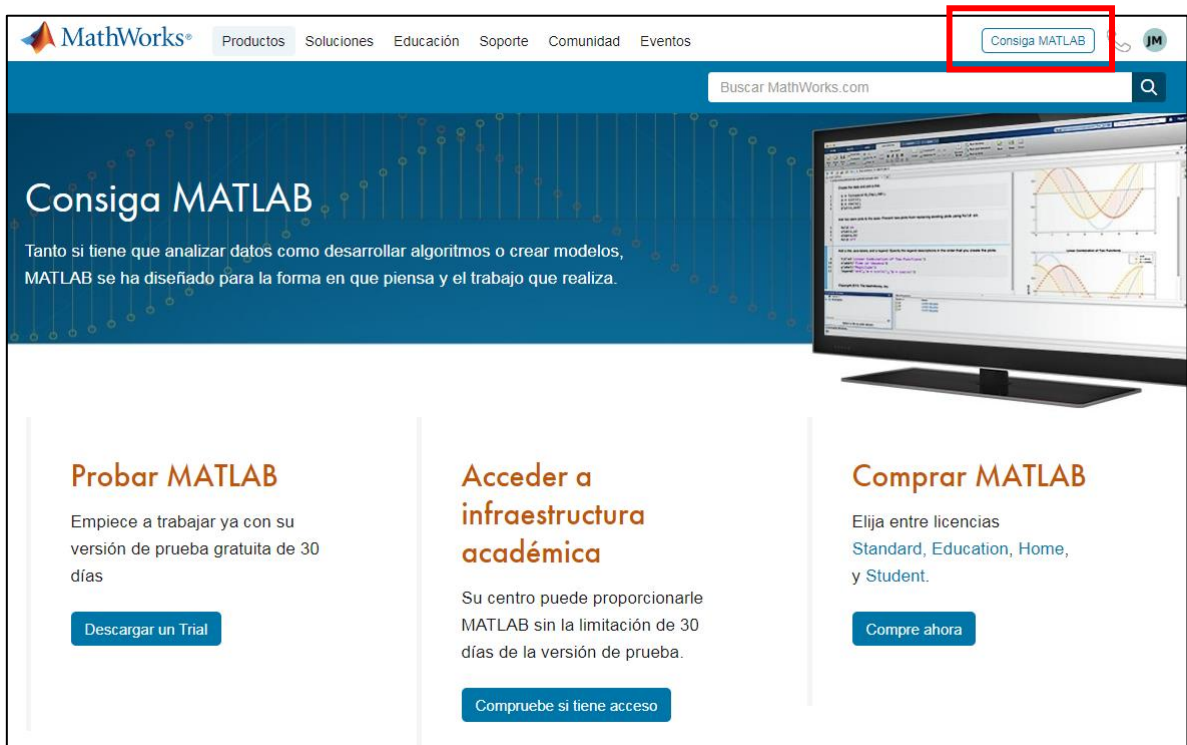


Figura 6-14: Página y registro oficial en MATHWORKS.

(Fuente: Propia)

11. Una vez seleccionada esa opción se tiene que dar click en la opción “Descargar” con ello se redirige a la pestaña con la versión más reciente del programa MATLAB.



Figura 6-15: Última versión MATLAB.

(Fuente: Propia)

12. A continuación, se descarga la última versión hasta el momento que es la R2020a y la que se utiliza para el presente trabajo.



Figura 6-16: Versión R2020a de MATLAB.

(Fuente: Propia)

13. La siguiente opción viene dada por el tipo de dispositivo que posee en función de su sistema operativo, para este caso MATLAB presenta tres opciones, para sistemas operativos Windows, MacOs y Linux; para el caso del proyecto actual se utiliza para sistema Windows 10 de 64 bits.

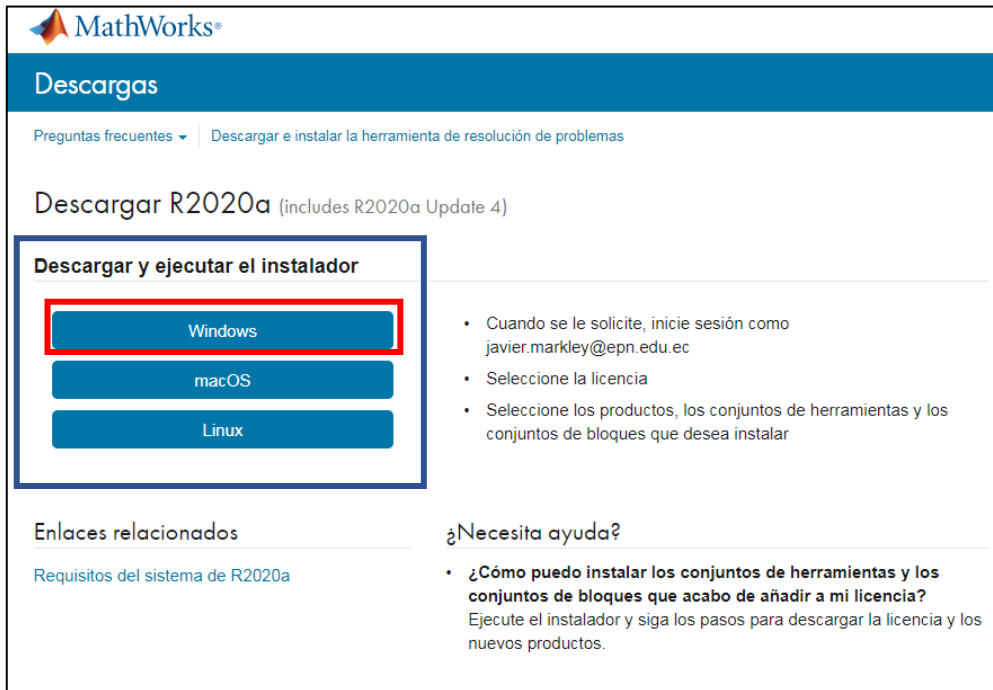


Figura 6-17: Sistema operativo.

(Fuente: Propia)

14. Después de haber seleccionado el sistema operativo, empieza la descarga del instalador, que se posiciona en la sección de descargas del navegador de su preferencia.

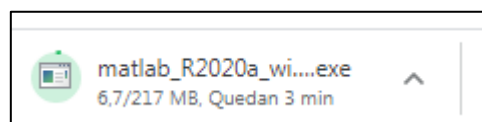


Figura 6-18: Proceso de descarga instalador MATLAB.

(Fuente: Propia)

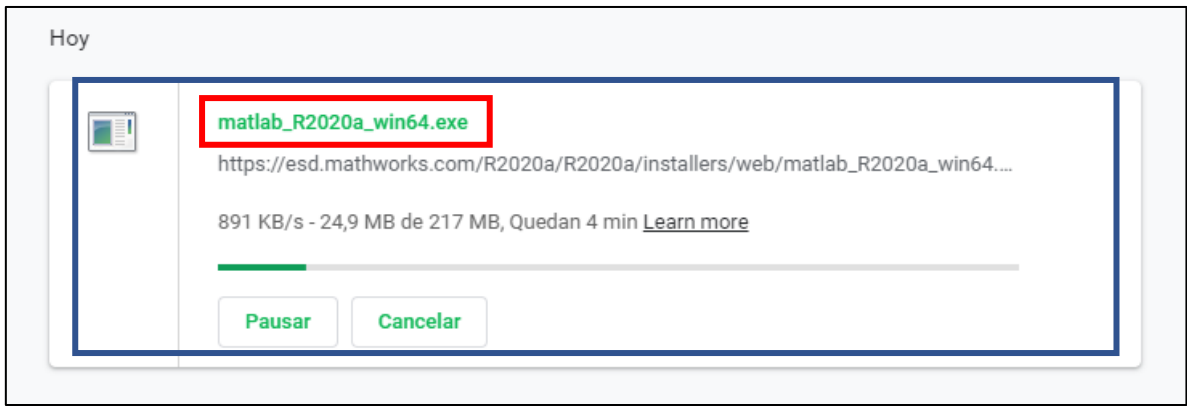


Figura 6-19: Sección de descargas navegador.

(Fuente: Propia)

15. Una vez finalizada la descarga del instalador de MATLAB, dar click en mostrar carpeta y se debe redirigir al instalador ubicado en la sección descargas de su computador.

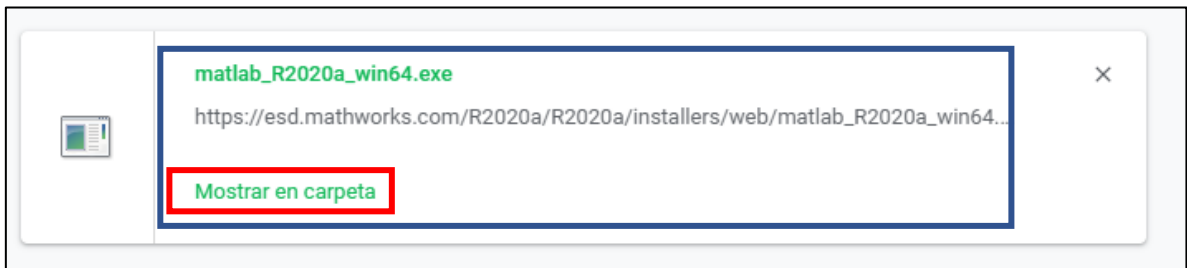


Figura 6-20: Descarga finalizada.

(Fuente: Propia)

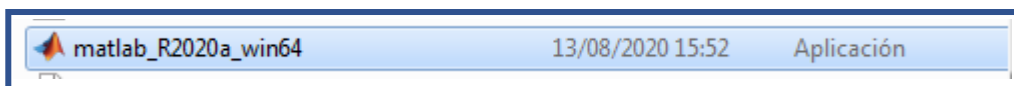


Figura 6-21: Instalador sección descargas computador.

(Fuente: Propia)

INSTALACIÓN MATLAB

1. Una vez que se haya descargado y abierto la carpeta donde se encuentra el instalador de MATLAB, se debe dar click derecho escoger la opción “Ejecutar como administrador”.

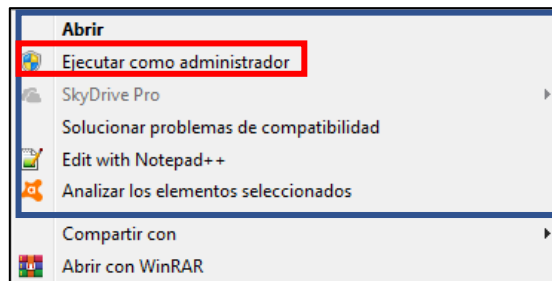


Figura 6-22: Ejecución del instalador de MATLAB.

(Fuente: Propia)

2. Depende de las características del computador en su rapidez de respuesta, para empezar a descomprimir todos los archivos que dispone la carpeta, si se realizó paso a paso la instalación no debe dar problema hasta este paso, pero en el caso que no se haya comprobado que el sistema operativo es de 64 bits salta el siguiente error.

Error de instalación

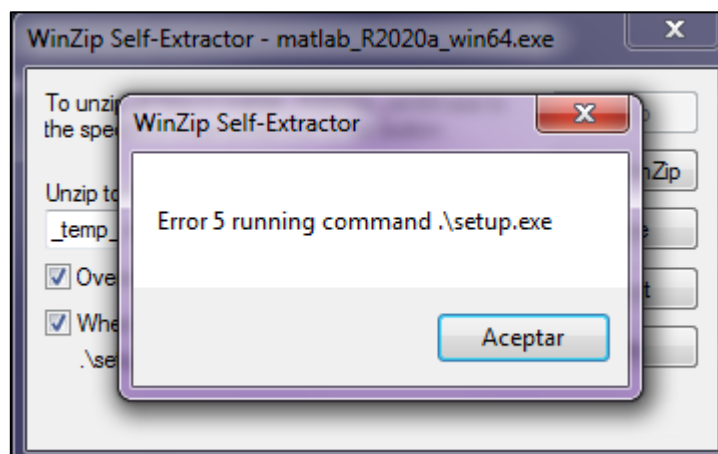


Figura 6-23: Error de instalación.

(Fuente: Propia)

Instalación correcta

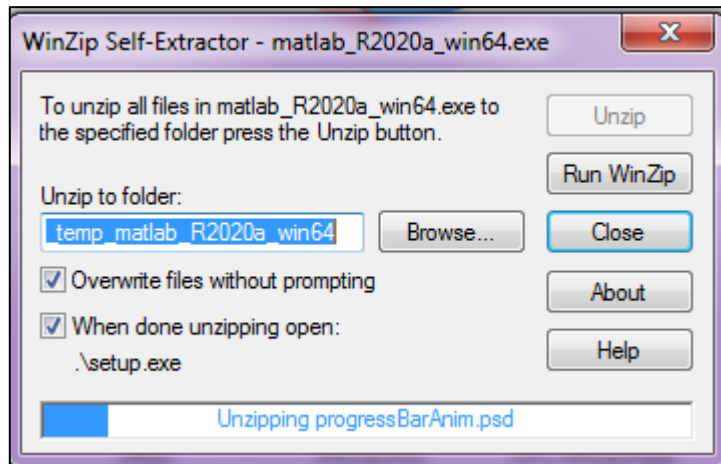


Figura 6-24: Descompresión sin error de los archivos MATLAB.

(Fuente: Propia)

3. Una vez se descompriman todos los elementos, se debe presionar “Permitir acceso” para que empiece la descarga.

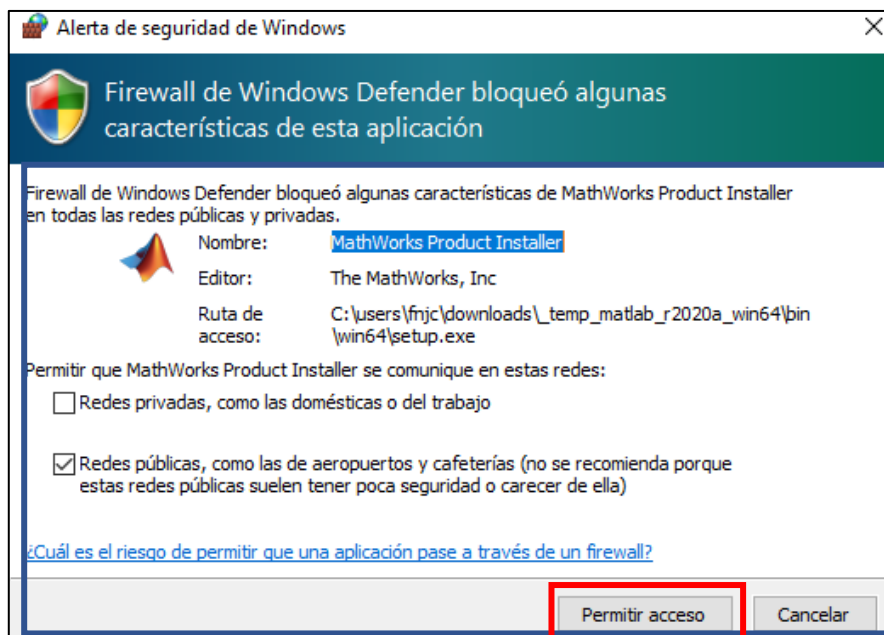


Figura 6-25: Permitir acceso aplicación MATLAB.

(Fuente: Propia)

4. Para seguridad de MathWorks se solicita de nuevo las credenciales institucionales ingresadas con anterioridad estas son correo y clave.

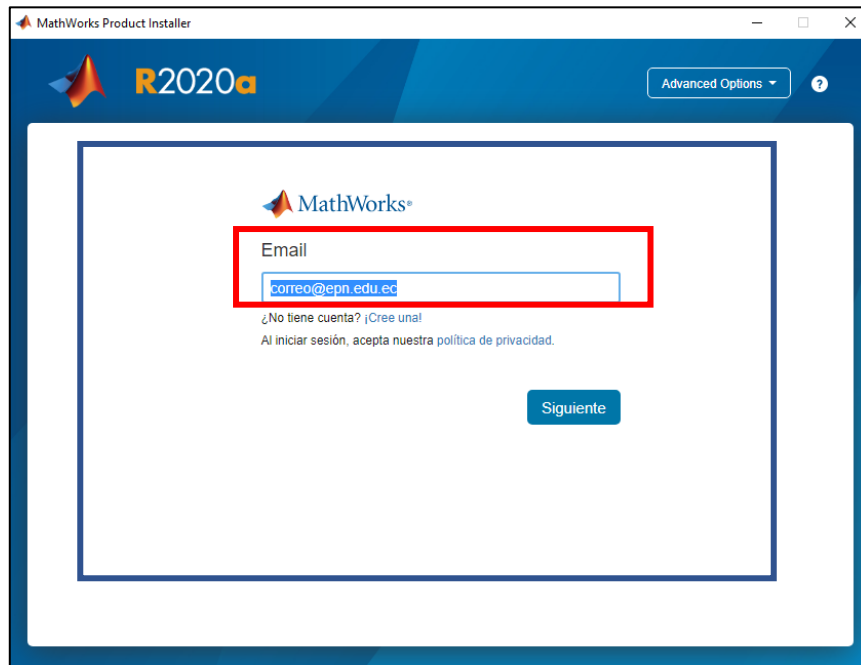


Figura 6-26: Credenciales dentro de MATLAB (Correo institucional).

(Fuente: Propia)

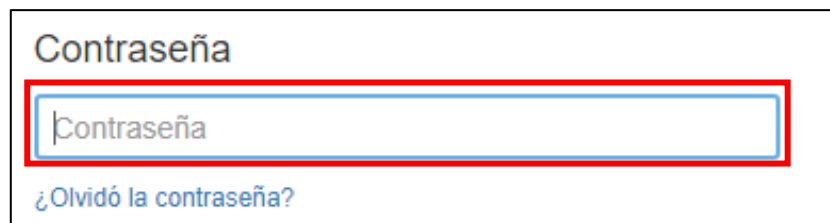


Figura 6-27: Credenciales dentro de MATLAB (Contraseña).

(Fuente: Propia)

5. En este paso únicamente se debe aceptar los términos y condiciones de MATHWORKS y se presiona el botón de “Next”.

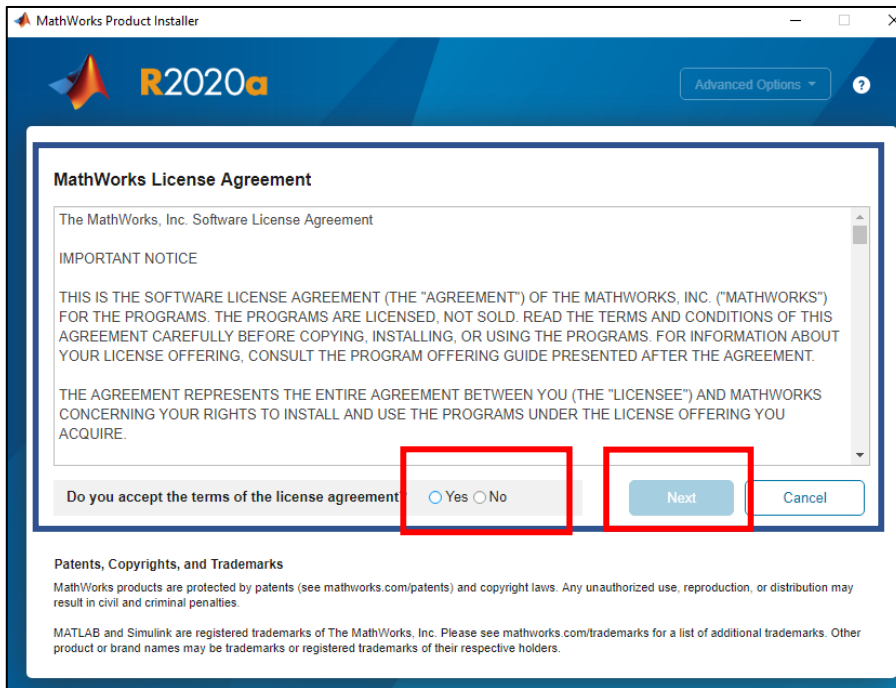


Figura 6-28: Aceptación de términos y condiciones.

(Fuente: Propia)

- Para este punto también se acepta la licencia asignada por defecto y se pulsa en el botón "Next".

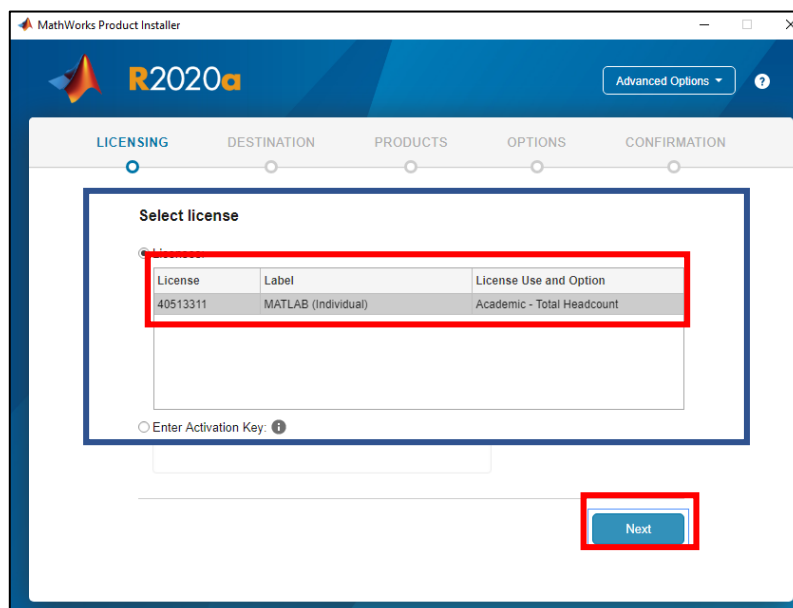


Figura 6-29: Asignación de licencia MATLAB.

(Fuente: Propia)

- Dentro de la siguiente sección se selecciona la ubicación donde se desea que se instale el programa.

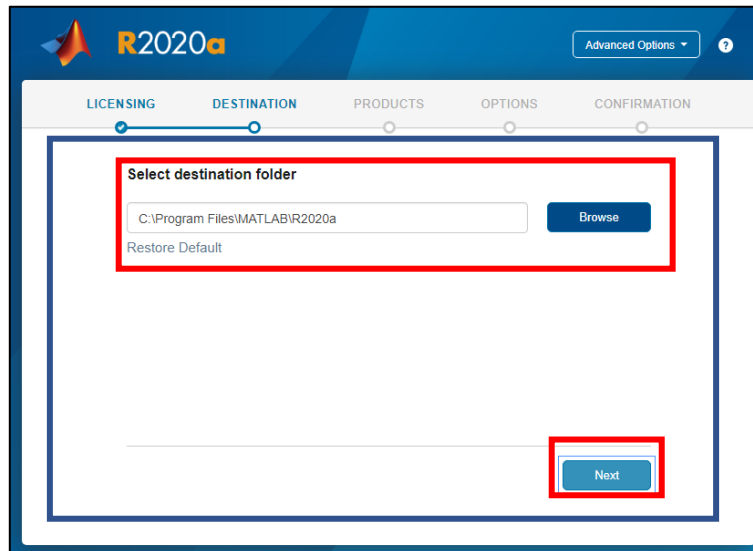


Figura 6-30: Ubicación de instalación.

(Fuente: Propia)

- El siguiente punto el programa da a escoger entre varios productos que posee, muchos de ellos son para aplicaciones muy puntuales y más específicas, si desea el programa completo con todos los productos puede seleccionar todos, pero para el presente trabajo se descargará únicamente los que vienen por defecto.

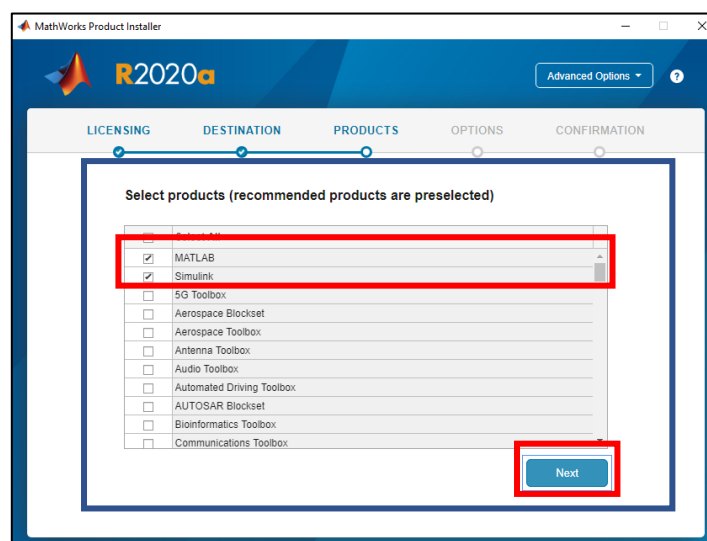


Figura 6-31: Productos MATLAB.

(Fuente: Propia)

9. En la siguiente pestaña MATHWORKS ofrece dos opciones, la primera es para añadir un acceso directo al escritorio y la segunda es un permiso para que MATLAB envíe información sobre sus productos, se selecciona la que desee el usuario.

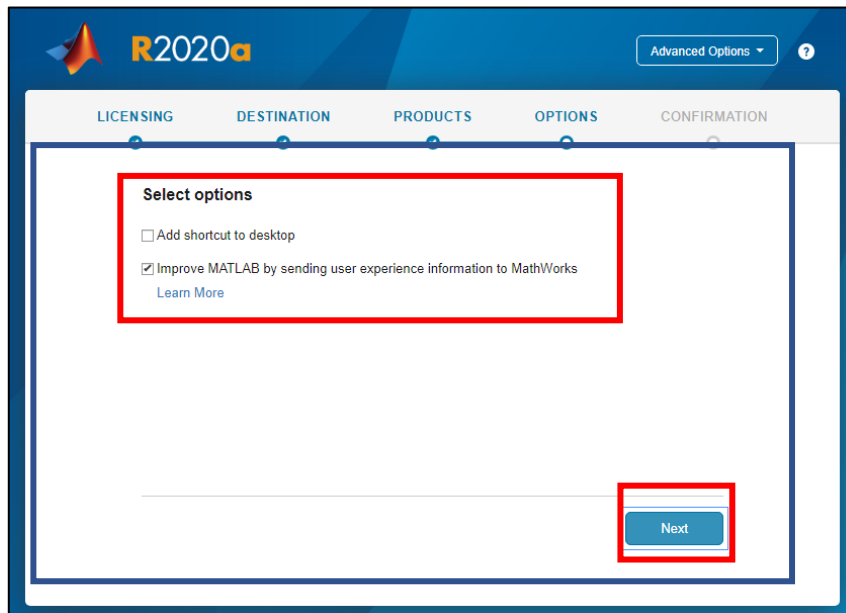


Figura 6-32: Opciones MATLAB

(Fuente: Propia)

10. A continuación, se presenta la información general de todos los ajustes realizados, una confirmación de todas las configuraciones.

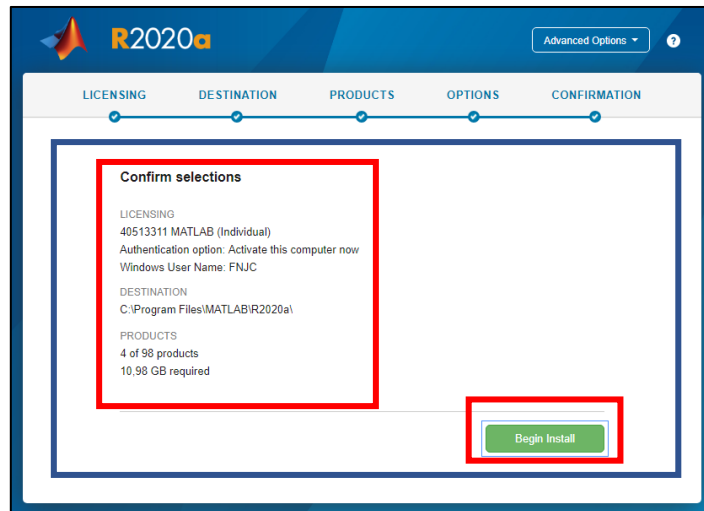


Figura 6-33: Confirmación MATLAB.

(Fuente: Propia)

11. La instalación del programa empieza sin problemas y debe finalizar sin ningún error, como se muestra en las siguientes figuras.

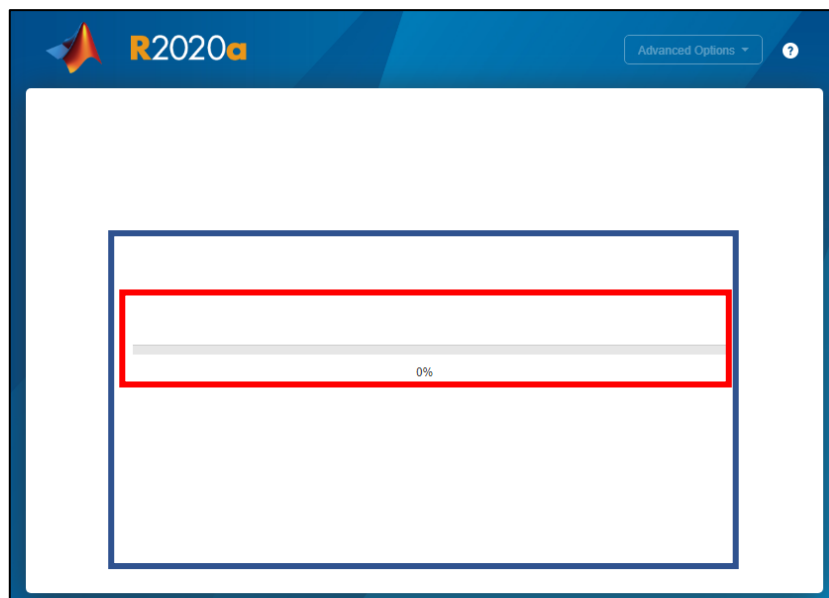


Figura 6-34: Inicio de la instalación.

(Fuente: Propia)

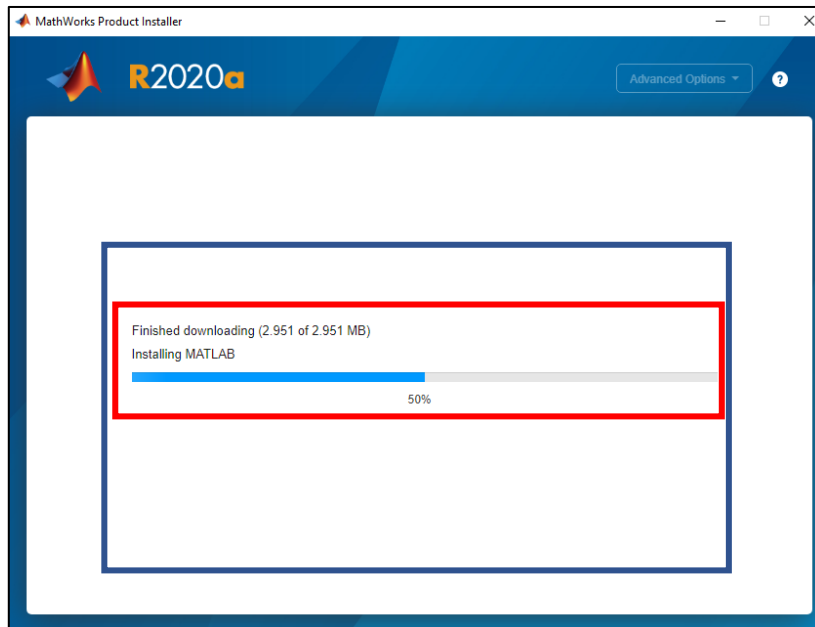


Figura 6-35: Instalación 50%.

(Fuente: Propia)

12. Cuando se finalice la instalación se deben cerrar todas las pestañas correspondientes al programa de MATLAB.

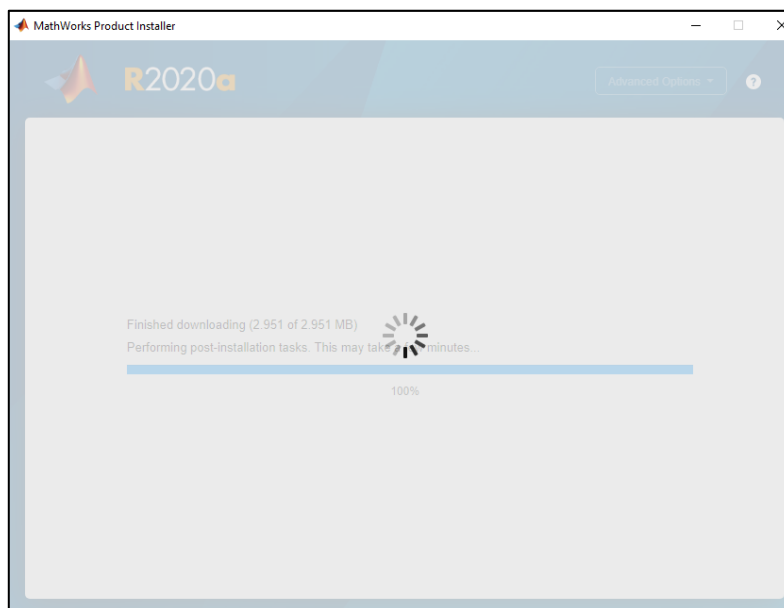


Figura 6-36: Fin de la instalación.

(Fuente: Propia)

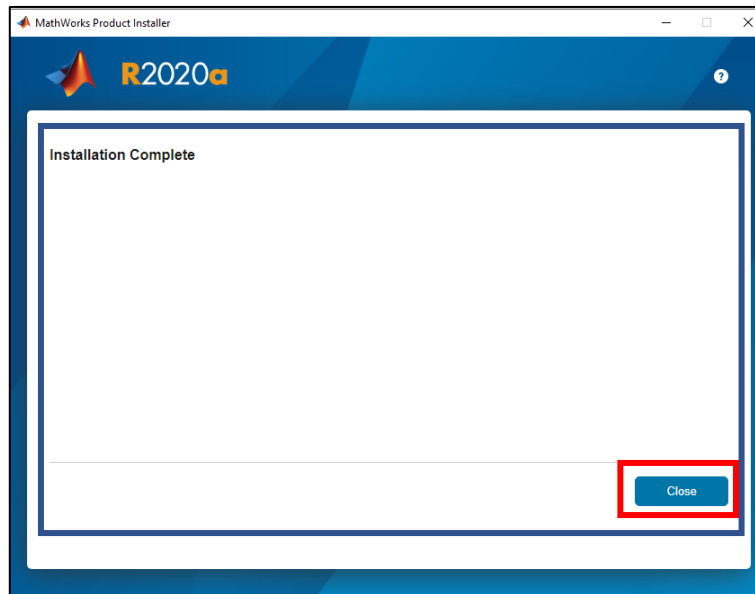


Figura 6-37: Instalación completa.

(Fuente: Propia)

13. Buscar en el escritorio el acceso directo si se escogió esa opción previamente o colocar la palabra MATLAB en el buscador del computador, una vez seleccionada se inicia, esto se muestra en las siguientes figuras.

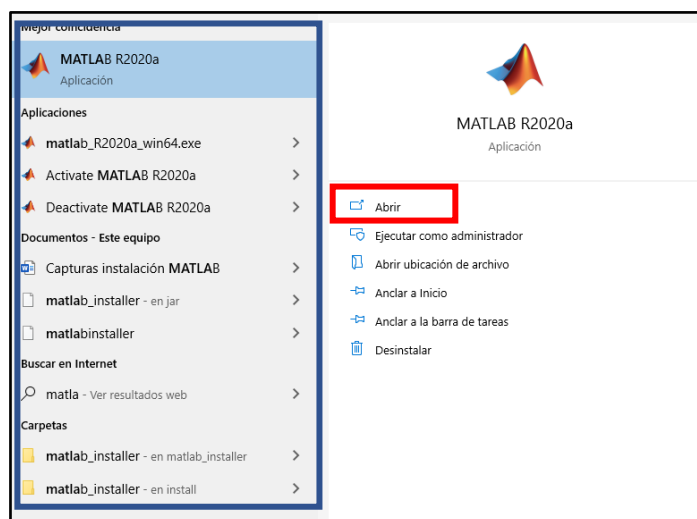


Figura 6-38: Abrir MATLAB.

(Fuente: Propia)

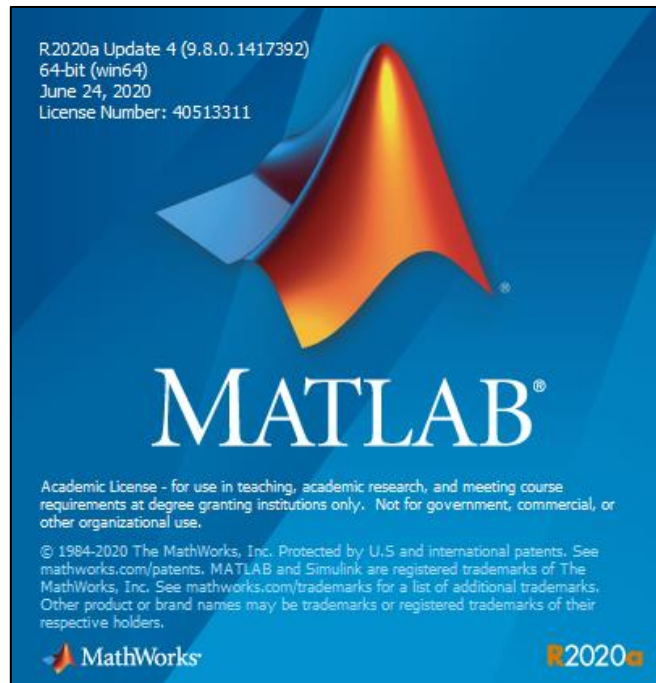


Figura 6-39: Inicialización de MATLAB.

(Fuente: Propia)

14. Una vez que la aplicación abra sin problemas y muestre el espacio de trabajo de MATLAB como se indica en la figura, la instalación del programa está completa, el único punto que falta es agregar la librería correspondiente.

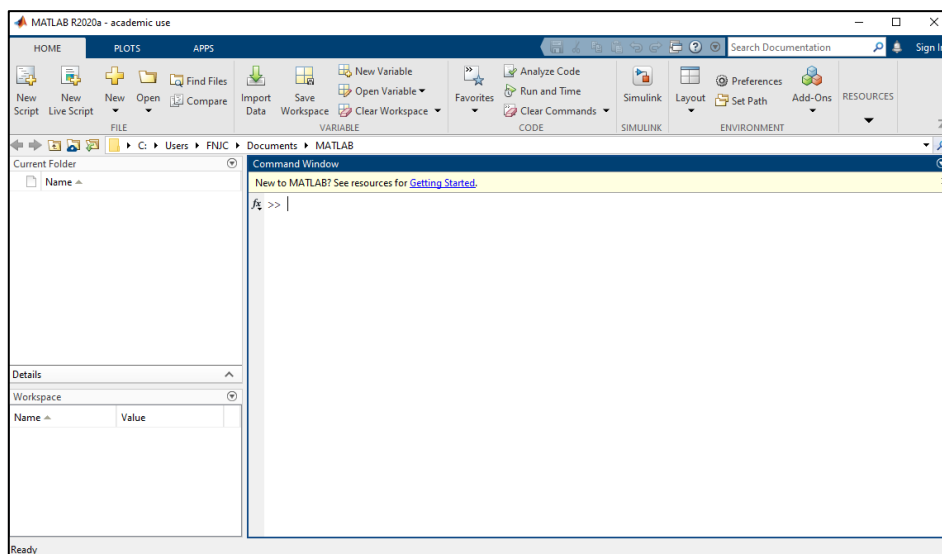


Figura 6-40: Espacio de trabajo de MATLAB.

(Fuente: Propia)

DESCARGA ROBOTICS TOOLBOX

La descarga de Toolbox robotics si se tiene Matlab 2020a, es mucho más sencillo hacerlo en comparación a versiones anteriores. A continuación, se detallan los pasos a seguir:

1. Visitar la página oficial de Peter Corke, creador de la librería o ir al siguiente enlace (<https://bit.ly/3aLRJKa>) que redirige a la mencionada página.

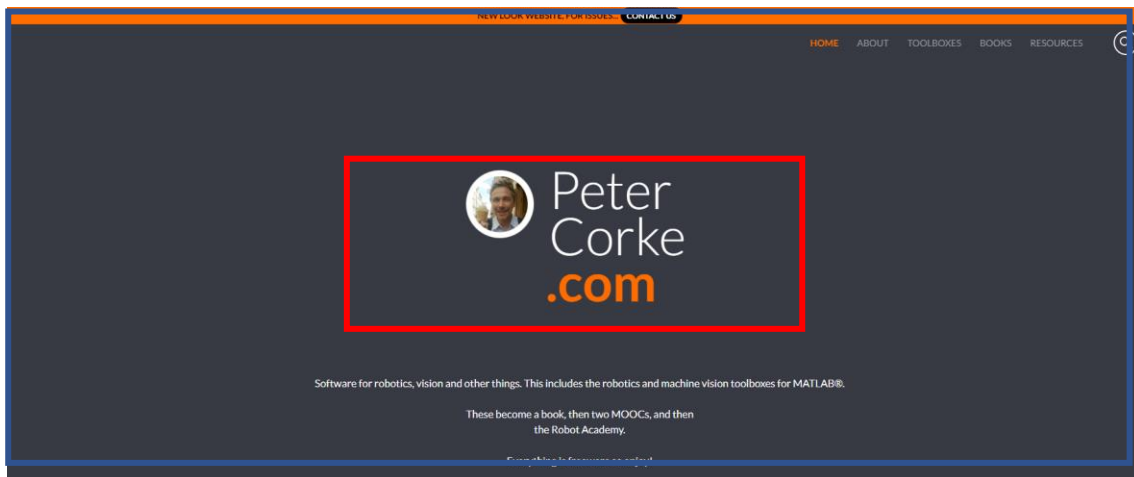


Figura 6-41: Página oficial Peter Corke.

(Fuente: Propia)

2. Una vez en la página oficial, buscar la barra de accesos y dar click en el acceso directo que dice "TOOLBOXES", de ahí se despliega todas las librerías que ofrece Peter Corke, dar click en la de "ROBOTICS TOOLBOX".

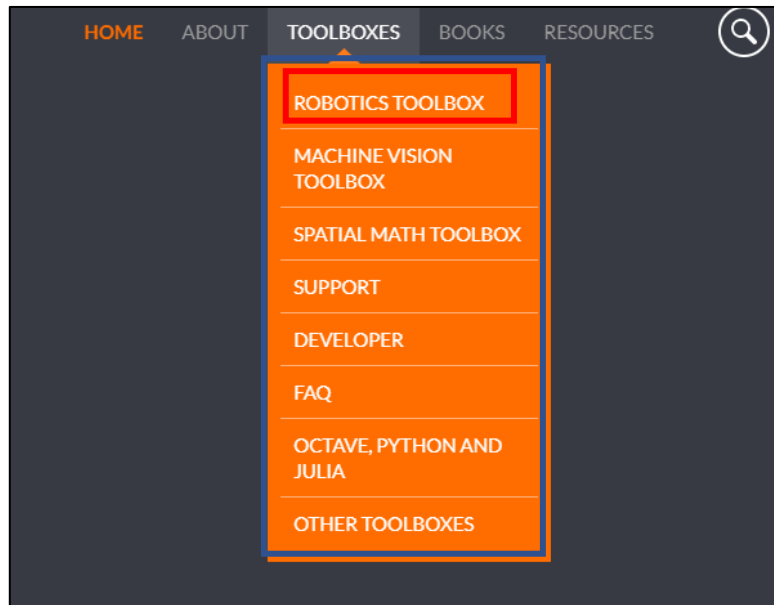


Figura 6-42: Librerías ofrecidas por Peter Corke

(Fuente: Propia)

3. Una vez dentro de la página de la librería, se encuentran varios descargables, entre ellos se dispone de la librería Toolbox Robotics y el manual de como instalarla y probarla, dar click en la librería.

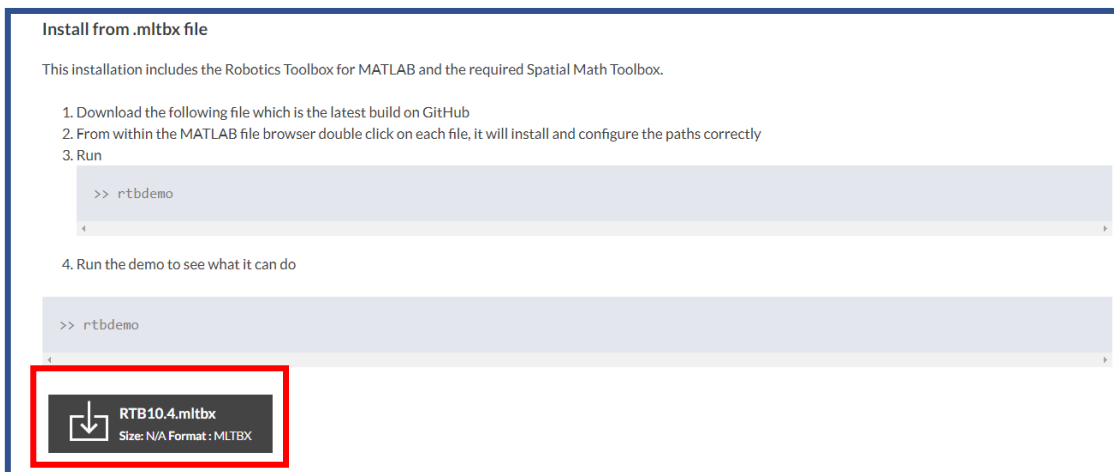


Figura 6-43: Librería RTB10.4.mltbx

(Fuente: Propia)

4. Una vez se de click en la librería, el documento empieza a descargarse, esperar hasta que finalice la descarga, una vez finalizada, dar click en “Mostrar en carpeta” y se debe mostrar donde se guardó la descarga del instalador para que pueda ser guardado en una carpeta de su preferencia.

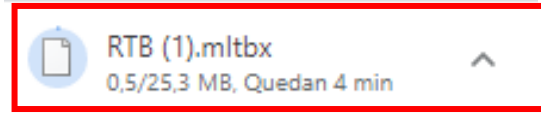


Figura 6-44: Descarga RTB10.4.mltbx

(Fuente: Propia)

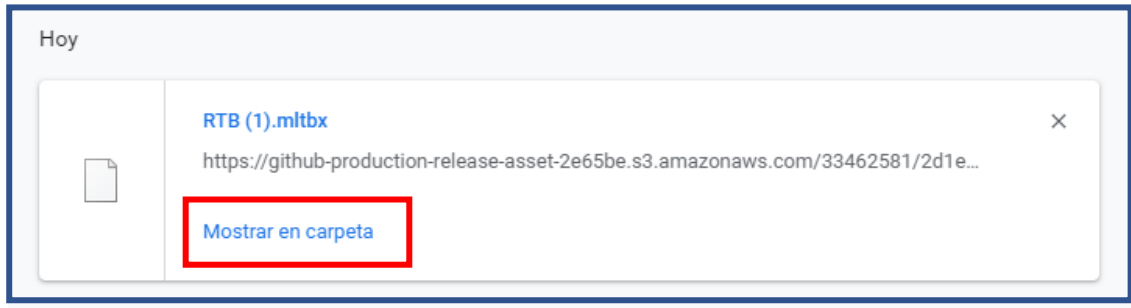


Figura 6-45: Mostrar en carpeta librería RTB10.4.mltbx.

(Fuente: Propia)

5. El siguiente paso se lo debe realizar desde MATLAB, una vez que lo tenga abierto, ir a la sección donde muestra el disco de almacenamiento de la computadora, ahí se tiene que buscar la carpeta donde se haya guardado el instalador de la librería.

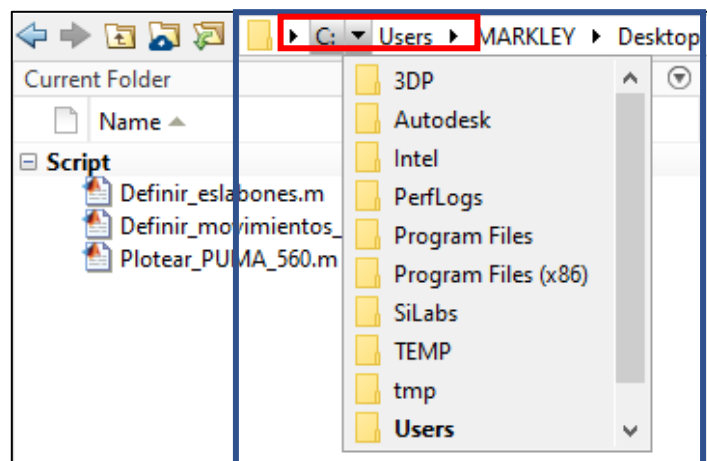


Figura 6-46: Ubicación de la librería RTB10.4.mltbx

(Fuente: Propia)

6. Una vez encontrada la librería RTB10.4.mltbx, se la debe dar click y la instalación debe empezar.

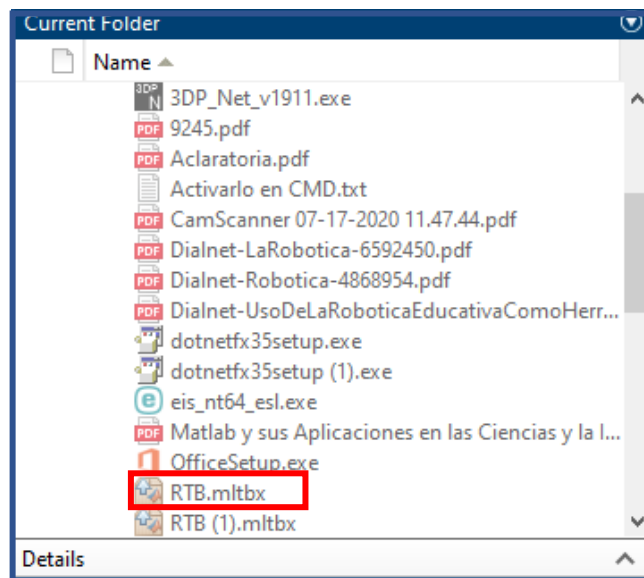


Figura 6-47: Librería RTB10.4.mltbx - Current Folder.

(Fuente: Propia)

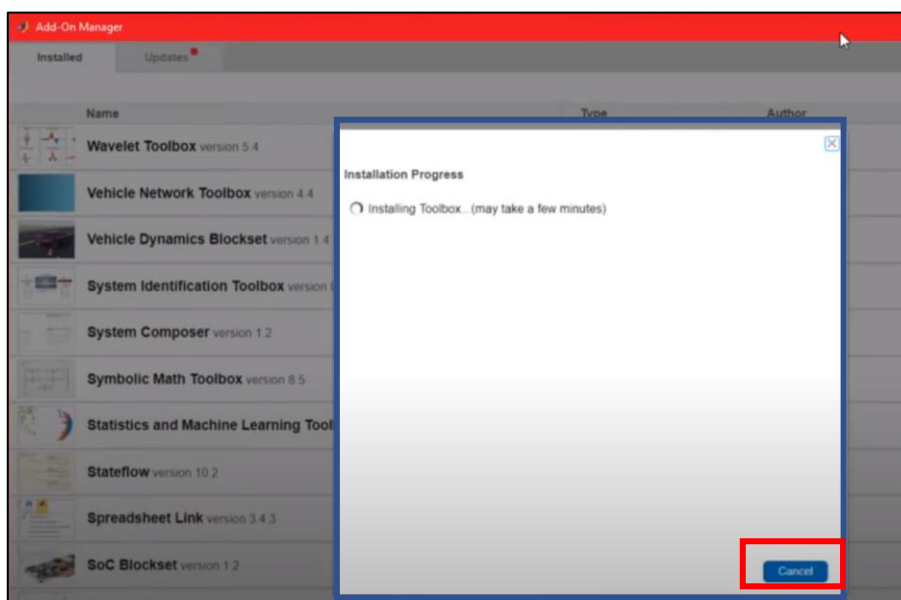


Figura 6-48: Proceso de instalación librería RTB10.4.mltbx.

(Fuente: Propia)

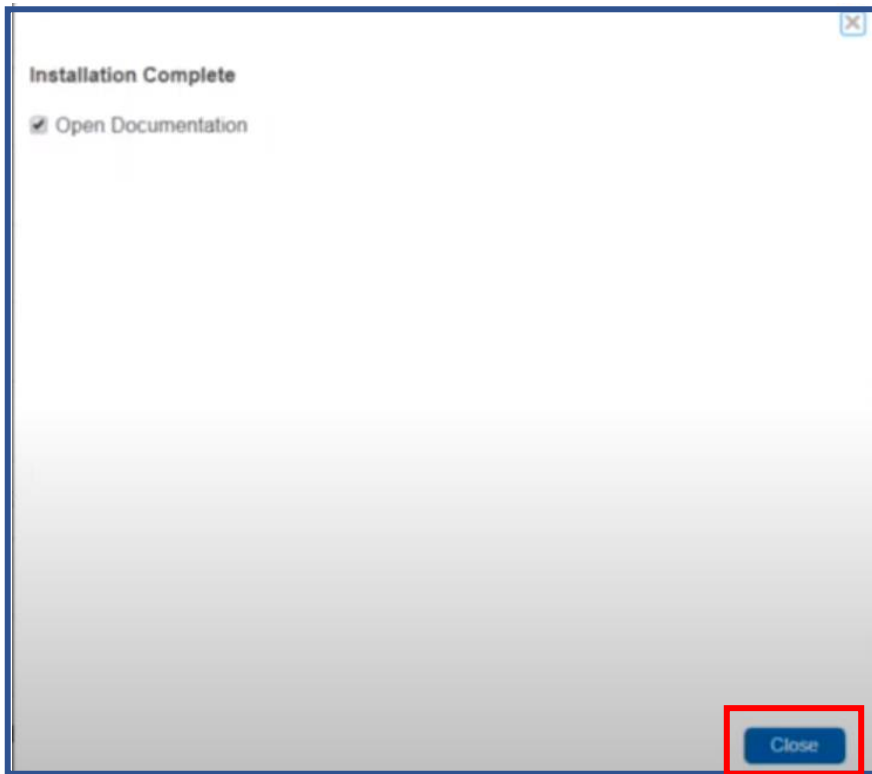


Figura 6-49: Instalación completa.

(Fuente: Propia)

7. Una vez completa la instalación se muestra una pestaña que indica las App instaladas y el Toolbox Robotics correspondiente, se verifica que el autor es Peter Corke, con esto se tiene todo lo necesario para el presente proyecto.

Name	Type	Author
 Triple angle visualizer version 1.0	App	Peter Corke
 Polynomialspirals version 1.0	App	Peter Corke
 Robotics Toolbox for MATLAB version 10.4	Toolbox	Peter Corke

Figura 6-50: Apps y Toolbox instalados.

(Fuente: Propia)

Author
Peter Corke
Peter Corke
Peter Corke

Figura 6-51: Autor: Peter Corke.

(Fuente: Propia)

PRUEBA LIBRERÍA TOOLBOX

En los pasos que se explicaron con anterioridad, se observó la descarga e instalación del software MATLAB con la respectiva librería Toolbox Robotics, necesarios para la modelar un robot, después de tener instalada la librería se debe comprobar su correcta instalación, esto se lo hace de la siguiente manera:

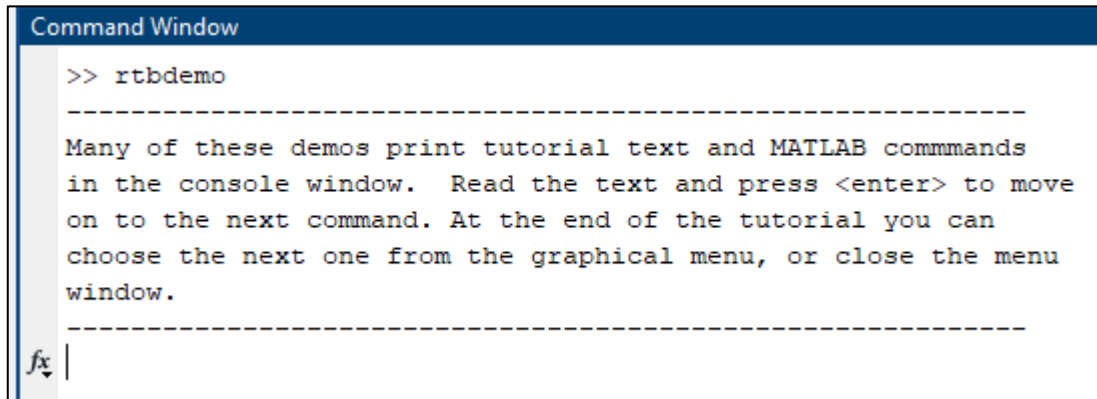
1. Dentro del programa MATLAB, se debe ejecutar el comando *rtbdemo*.



Figura 6-52: Comando *rtbdemo*.

(Fuente: Propia)

2. Si la librería está bien instalada, MATLAB debe reconocer este comando, con esto se despliega el siguiente mensaje indicando que debe presionar “enter” para ejecutar cada comando.



```
Command Window
>> rtbdemo
-----
Many of these demos print tutorial text and MATLAB commands
in the console window. Read the text and press <enter> to move
on to the next command. At the end of the tutorial you can
choose the next one from the graphical menu, or close the menu
window.
-----
fx |
```

Figura 6-53: rtbdemo - Command Window.

(Fuente: Propia)

3. Una vez que se hayan ejecutado los comandos correspondientes se abre una pestaña de MATLAB correspondiente a la interfaz interactiva para probar varias herramientas que posee la librería Toolbox Robotics, se selecciona cualquiera, pero para fines didácticos seleccionar “Create a model” dentro de la sección “Robot”

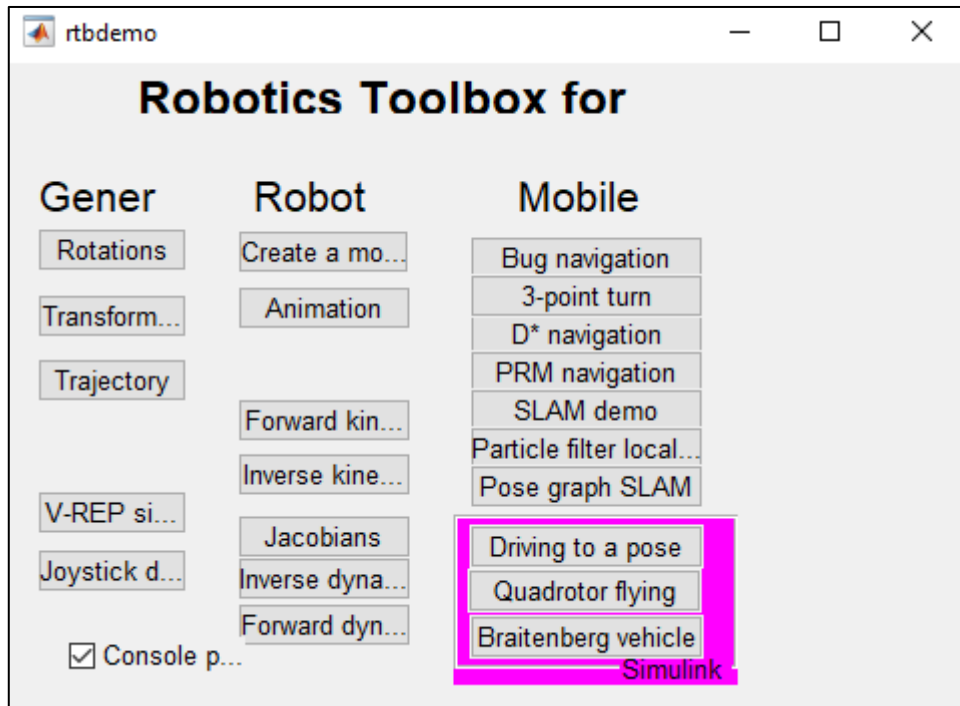


Figura 6-54: Interfaz demo - Robotics Toolbox.

(Fuente: Propia)

4. Una vez seleccionada cualquier opción, se inicia de igual manera varios comandos, cada vez que se indique la palabra *continue?* se debe dar un “enter” para pasar al siguiente comando.

```

Command Window
--- runscript <-- C:\Users\MARKLEY\AppData\Roaming\MathWorks\MATLAB Add-Ons\Toolboxes\Robotics Toolbox for MATLAB\dem

A serial link manipulator comprises a series of links. Each link is described
by four Denavit-Hartenberg parameters.
Let's define a simple 2 link manipulator. The first link is

>> L1 = Link('d', 0, 'a', 1, 'alpha', pi/2)

L1 =
Revolute(std): theta=q, d=0, a=1, alpha=1.5708, offset=0

fx continue?

```

Figura 6-55: continue?

(Fuente: Propia)

5. Una vez se finalice la ejecución de comandos se debe abrir otra pestaña de MATLAB *Figure 1*, mostrando el ejemplo de un robot dentro del espacio tridimensional, con esto se podrá verificar que la librería Toolbox Robotics está bien instalada.

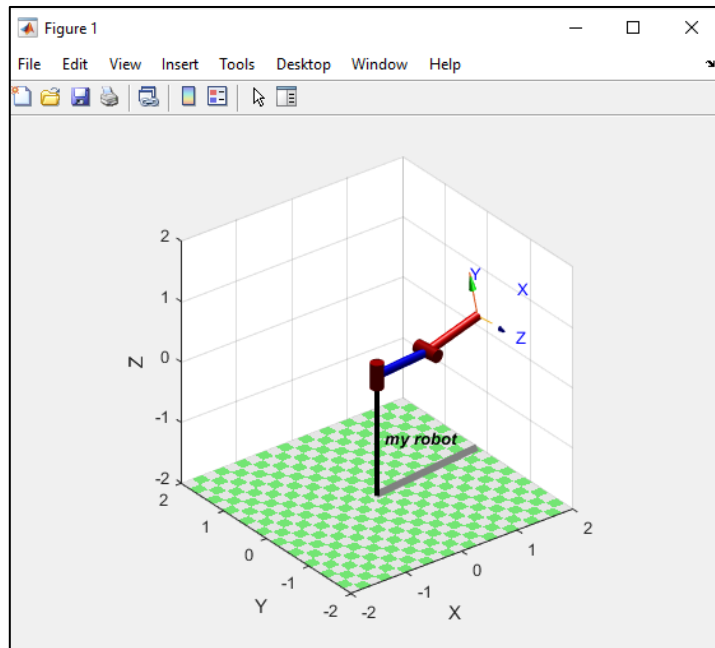


Figura 6-56: Modelado robot demo.

(Fuente: Propia)