Simulación del control de bajo nivel de un manipulador industrial

Julio Salvador Lora Millán

Juan Alejandro Castaño Peña

1. SIMULIK: Simulación y diseño basado en Modelos

Simulink® es un entorno de diagramas de bloque para la simulación multidominio y el diseño basado en modelos. Admite el diseño y la simulación a nivel de sistema, la generación automática de código y la prueba y verificación continuas de los sistemas embebidos.

Simulink ofrece un **editor gráfico**, bibliotecas de bloques personalizables y ***solvers*** para modelar y simular sistemas dinámicos. Se integra con MATLAB®, lo que permite incorporar algoritmos de MATLAB en los modelos y exportar los resultados de la simulación a MATLAB para llevar a cabo más análisis.

Capacidades de Simulink®

- Creación del Modelo: Crea modelos de subsistemas jerárquicos con bloques de biblioteca predefinidos. En Simulink, el término *Modelo* se suele utilizar en sustitución de lo que se entiende como "programa". Un modelo es un programa realizado en Simulink.
- Simulación del Modelo: Simula el comportamiento dinámico del sistema y presenta los resultados a medida que se ejecuta la simulación.
- Análisis de resultados de la simulación
- Gestión de proyectos
- Conexión con hardware: Conecta el modelo con hardware para realizar pruebas en tiempo real y desplegar sistemas embebidos.

En la web de Simulink se puede encontrar toda la documentación técnica, aplicaciones y ejemplos.

https://es.mathworks.com/products/simulink.html

2. Primeros pasos con Simulink

2.1. Abrir Simulink

Simulink se puede abrir de varias maneras diferentes:

Tecleando desde la ventada de comandos de Matlab:

```
>> simulink
```

- Desde el botón *Simulink* de Matlab

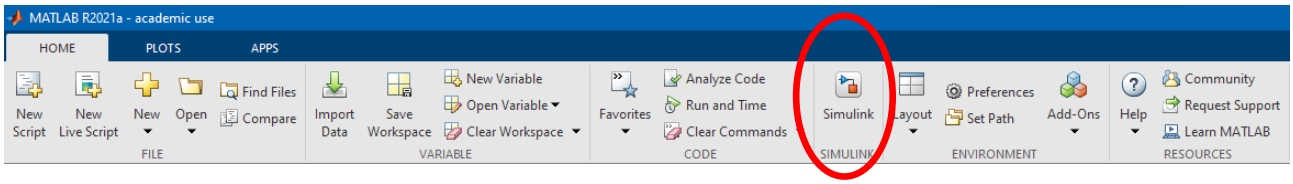


Figura 1. Cómo arrancar Simulink

- Haciendo doble click en un archivo de simulink (.mdl, .mdlx)

Al abrir Simulink se obtiene una ventana cómo la que aparece en la Fig.2, que permite seleccionar entre varias opciones de trabajo:

- *Blank Model:* crea un "modelo" (un archivo de programación gráfico con bloques) en blanco para comenzar a programar. Es la opción recomendada para programas de tamaño y/o complejidad no muy alta.
- *Blank Subsystem*: permite crear un modelo que pueda ser utilizado como subsistema ("bloque de código") en otro modelo (o programa).
- *Blank Library*: permite crear un conjunto de modelos (bloques) propios, organizados en una librería integrable con la librería de modelos de Simulink.
- *Blank Project*: Un proyecto es una manera de organizar programas complejos, con dependencias entre otros programas y extensiones, tanto de Simulink como de terceros.
- *Folder to Project:* permite crear un proyecto a partir de programas Simulink que se encuentren dentro de una carpeta (y subcarpetas).
- *Project from Git:* permite descargar un Proyecto desde un repositorio en internet y hacer una copia local para trabajar sobre él.
- *Project from SVN:* permite descargar un Proyecto desde un repositorio SVN (con control de versiones online) en internet y hacer una copia local para trabajar sobre él.
- Code Generation: permite compilar en código de bajo nivel (C, C#, etc) los programas hechos con Simulink para ser integrados en controladores.

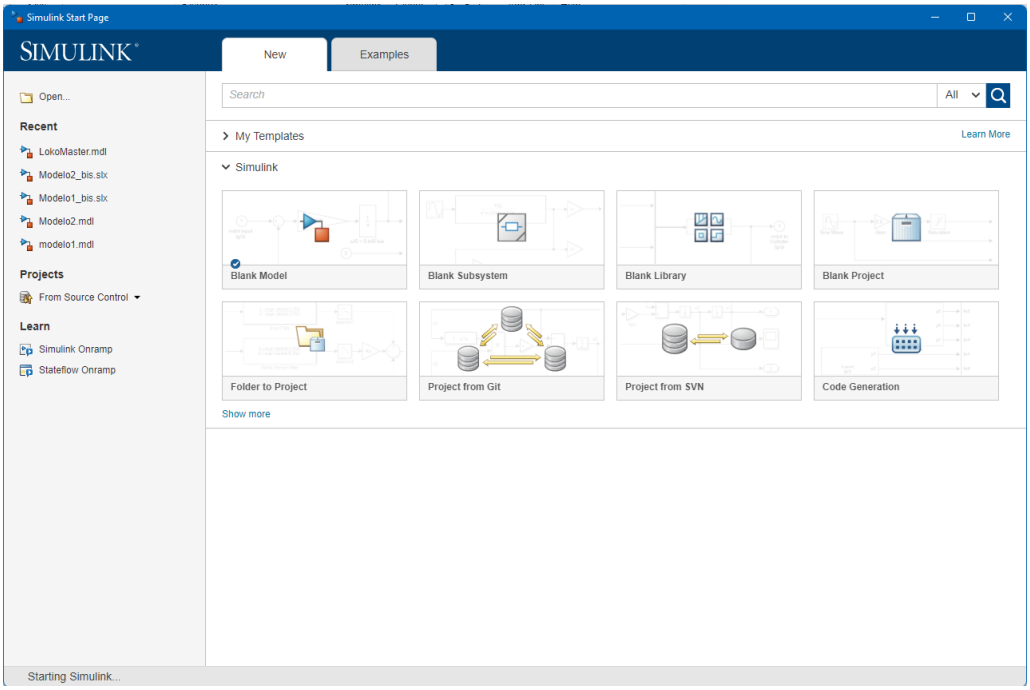


Figura 2 Entorno de Simulink

Una vez abierto un modelo en blanco, se obtiene la pantalla de la figura 3, que muestra el lienzo donde se programa el modelo, ubicando bloques y flechas. Dichos bloques se obtienen desde la librería de Simulink. Para acceder a ella, se pulsa en en "*Library Browser*", y se obtiene el navegador de librerías (Fig. 4). En esta ventana se muestran las librerías disponibles, cada una de ellas contiene bloques elementales con los que modelar los sistemas. El número de librerías depende de la versión del programa y de las *toolboxes* (extensiones) instaladas.

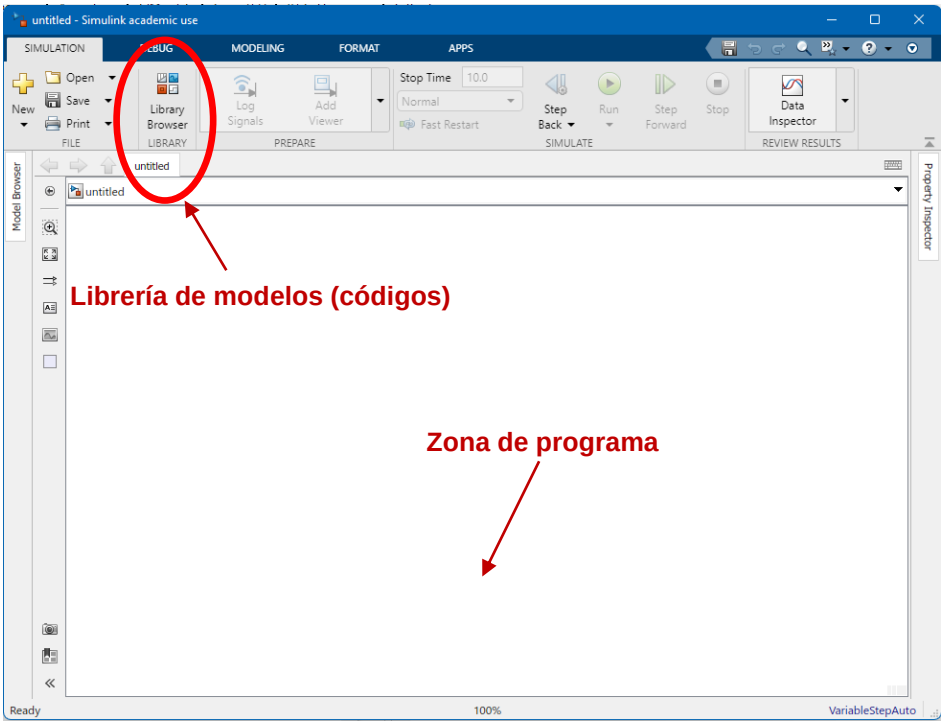


Figura 3 Entorno de programación

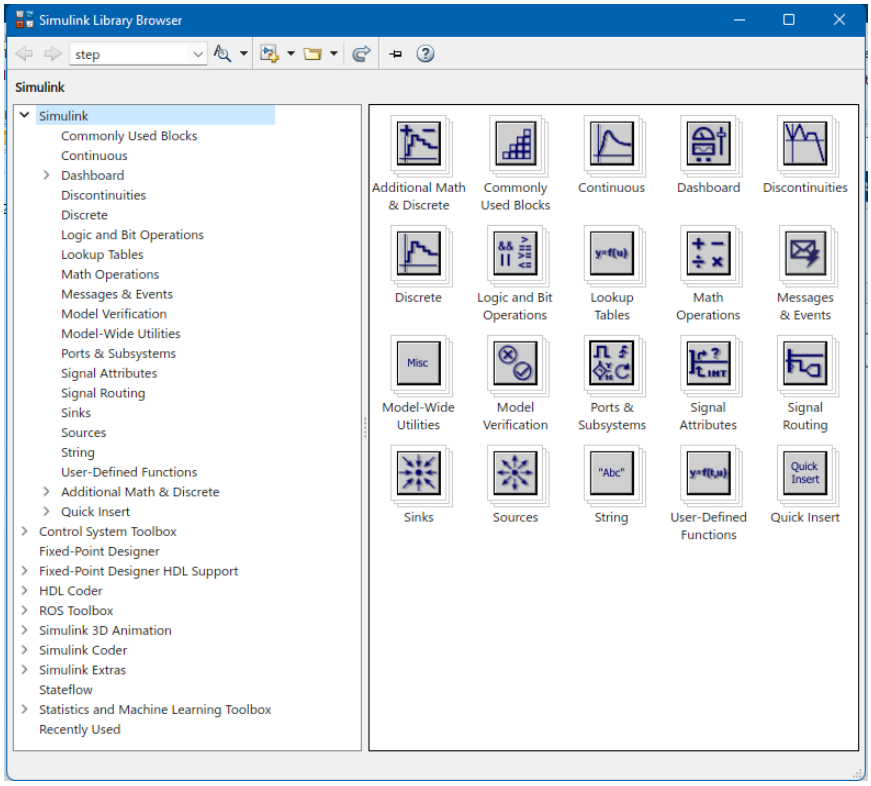


Figura 4 Librerías de Simulink

2.3. Bloques en Simulink

Como se observa en la figura 4, los bloques de Simulink se estructuran en distintas categorías. A continuación, se van a describir brevemente algunas de las categorías más comunes:

- Continuos: Representan sistemas continuos por su relación entrada-salida. Entre los bloques que podemos encontrar en esta categoría tenemos (ver Fig.5):
  - Derivative (bloque derivador: la salida es derivada de la entrada).
  - Integrator (bloque integrador: la salida es la integral de la entrada).
  - Transfer Fcn (función de transferencia en s expresada como cociente de polinomios).
  - Zero-Pole (función de transferencia en s expresada como cociente en forma factorizada)

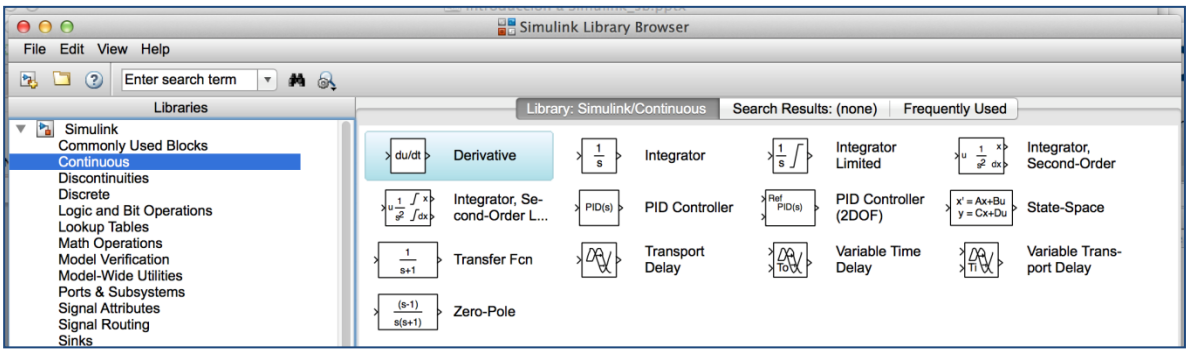


Figura 5 Entorno de Simulink: Categoría Continuos

- *Sources*: Entradas o "fuentes" de señales (ver Fig.6):
  - *Step* (Escalón)
  - *Ramp* (Rampa)
  - *Sine Wave* (Función sinusoidal).
  - *Pulse generador* (Tren de pulsos)
  - *From workspace* (Lectura de datos desde la memoria de trabajo o Workspace de Matlab)
  - *To workspace* (Mandar datos hacia la memoria de trabajo o Workspace de Matlab)

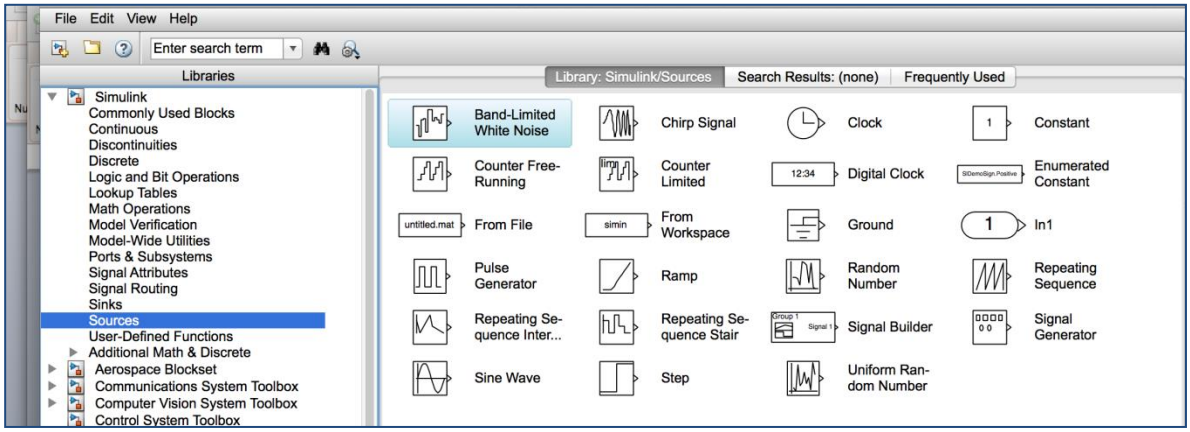


Figura 6 Entorno de Simulink: Categoría Sources

- *Sinks*: Salidas o dispositivos de visualización/almacenamiento de las variables del sistema (ver Fig.7):
  - *Scope* (osciloscopio). Permite observar el contenido de las señales que se transmiten entre bloques
  - *Display* (indicador numérico)
  - *To workspace* (Mandar datos hacia la memoria de trabajo o Workspace de Matlab)

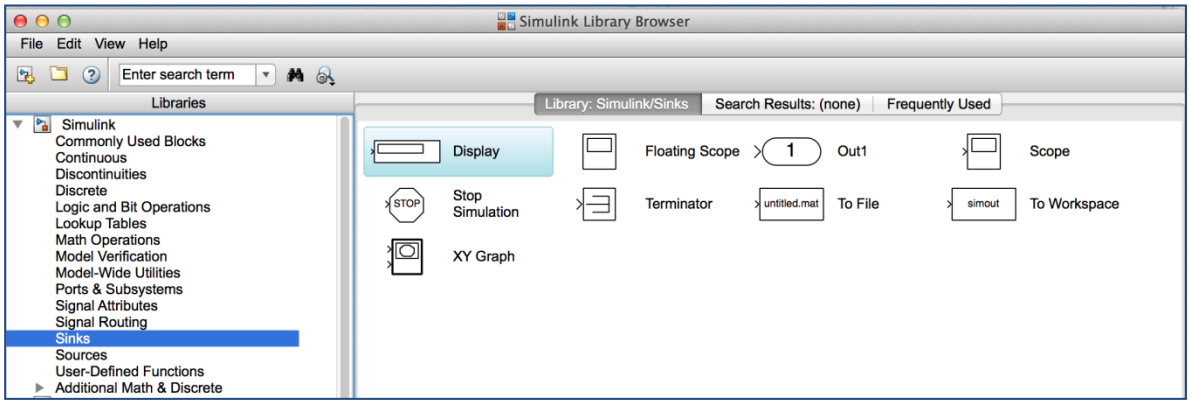


Figura 7 Entorno de Simulink. Categoría Sinks

Los bloques de *Math Operators* permiten realizar operaciones matemáticas sobre señales (Fig.8).

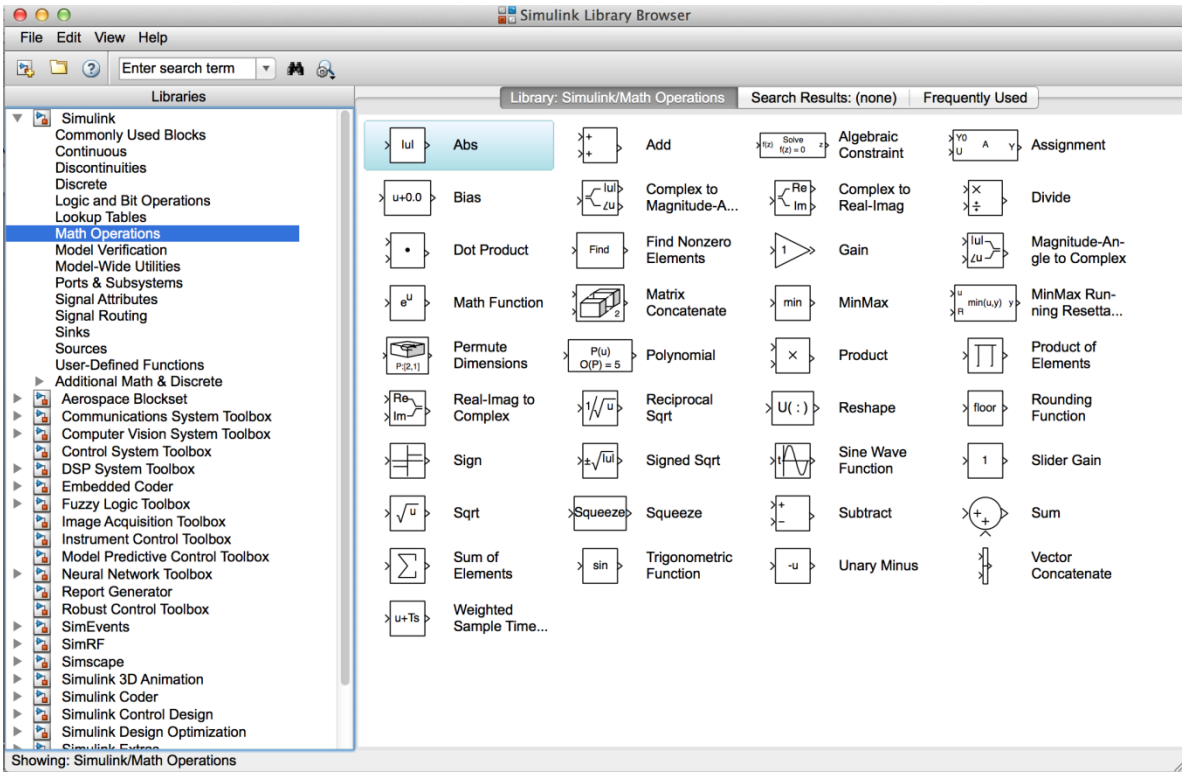


Figura 8 Entorno de Simulink. Categoría Math Operators

Los bloques de *Signal Routing* permiten realizar conexiones entre señales (Fig.9).

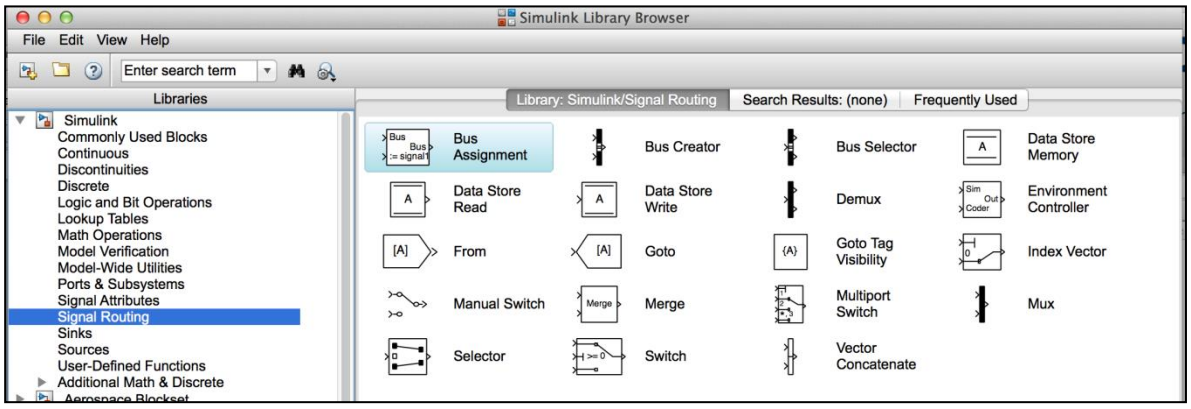


Figura 9 Entorno de Simulink Categoría Signal Routing

Los bloques de *User-Defined Functions* (Fig. 10) permiten utilizar código propio de Matlab (utilizado para los scripts .m) o C dentro del modelo simulink

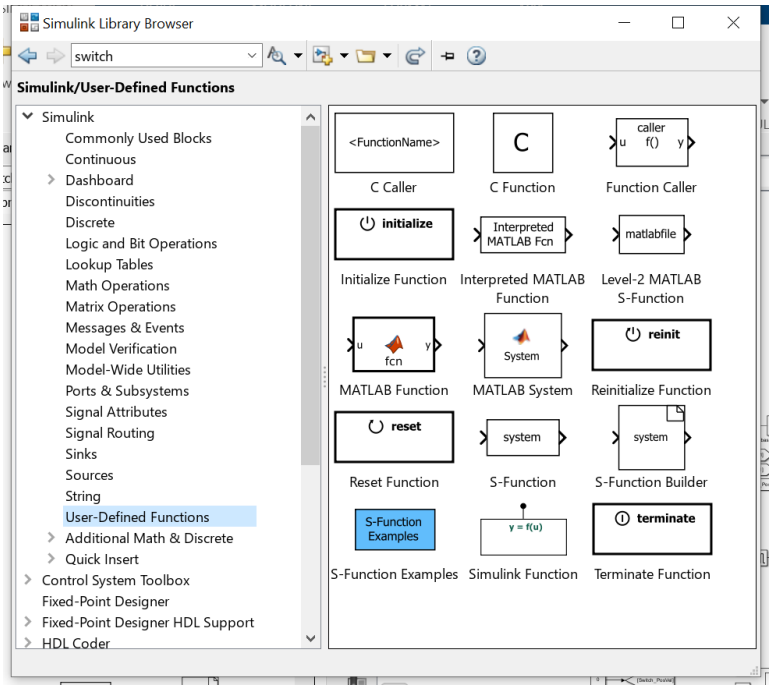


Figura 10: Entorno de Simulink Categoría User-defined functions.

En las figuras 11 y 12, se muestra un ejemplo de una función escrita en código de Matlab para un modelo Simulink. Nótese que, por defecto, las variables se inicializan cada vez que se ejecuta la función, por lo que para que el valor se mantenga para ejecuciones sucesivas, las variables deben declararse de tipo *Persistent*

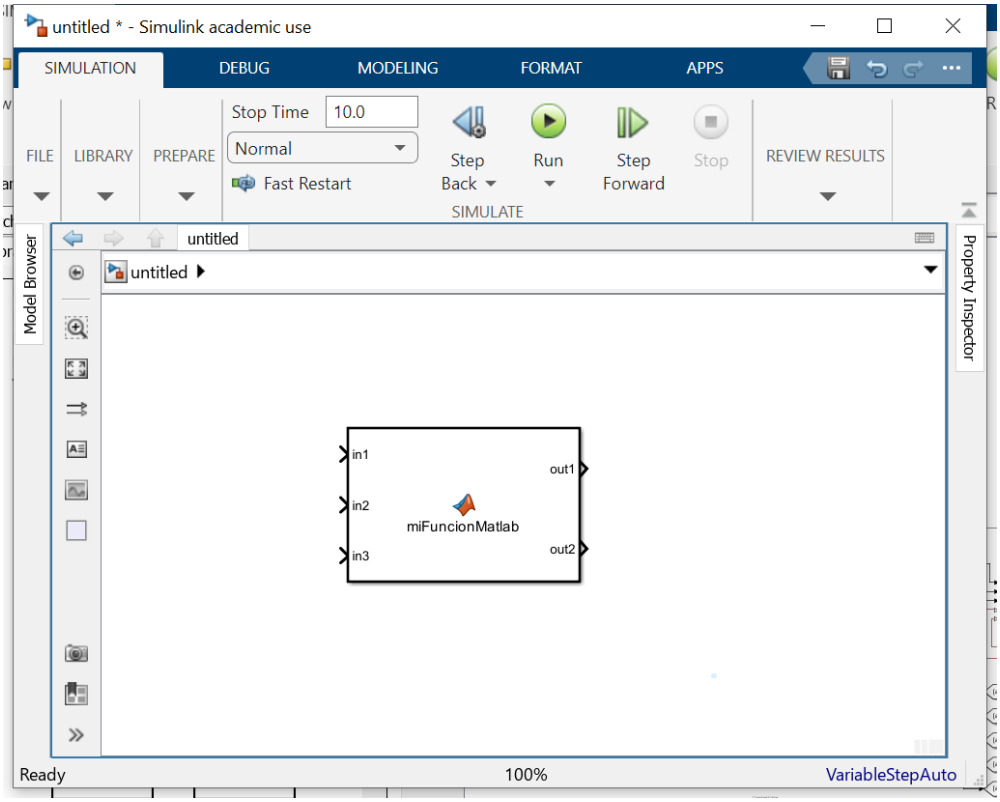


Figura 11: Ejemplo de función Matlab definida por el usuario

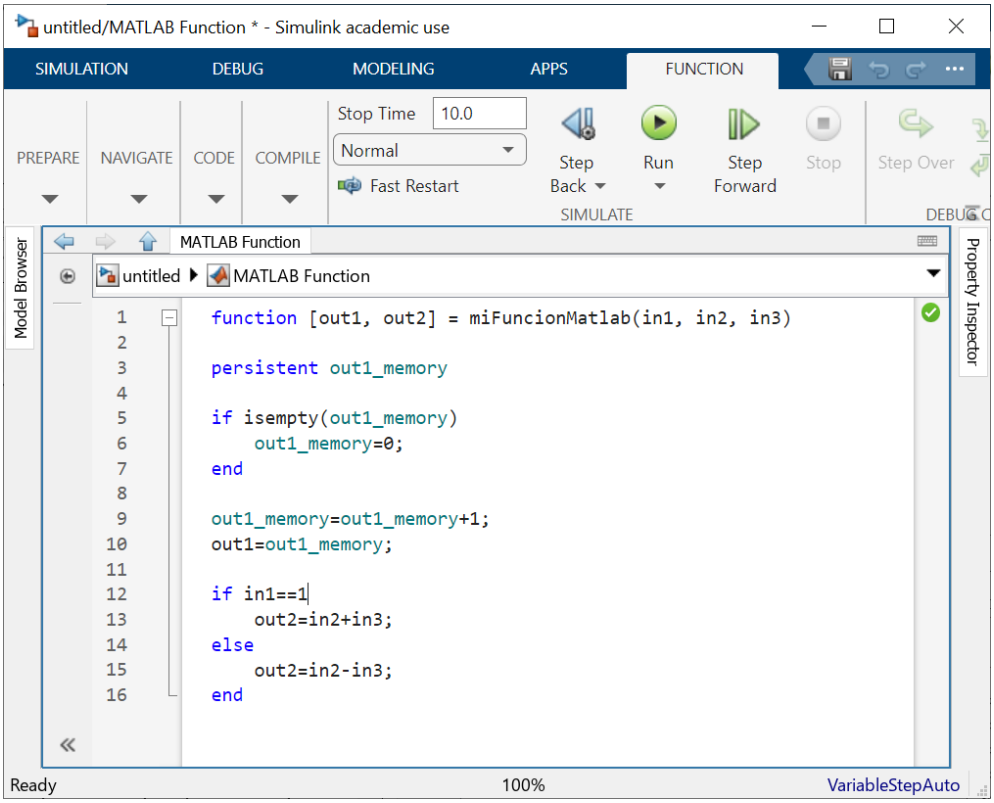


Figura 12: Código de script para una función definida por el usuario.

2.4. Creación de un Modelo

En este ejemplo se va a crear un modelo que nos permita generar la función *sen(t)* y representar sus valores.

2.4.1. Creación del modelo.

Se debe partir de un modelo en blanco (Figura 3) y la librería de bloques de programa ("*Library Browser*") para buscar y seleccionar los bloques necesarios para el programa (Figuras 4 a 9)

2.4.2. Introducción de bloques en el modelo.

Los elementos se introducen arrastrando con el ratón desde la ventana que contiene el listado de todos los bloques hacia la ventana de diseño. En nuestro caso requeriremos los siguientes bloques:

- Dentro de la categoría ***Source***, el bloque ***Sine Wave***.
- Dentro de la categoría ***Sinks***, el bloque ***Scope***. Será el que utilicemos para visualizar la señal.

Una vez situados los elementos en la ventana de diseño, se establecen las conexiones entre ellos, para ello se arrastra con el ratón la conexión de la salida de un bloque a la entrada del otro.

2.4.3. Configuración de los parámetros de los bloques

Todos los bloques de Simulink permiten modificar sus parámetros. En particular, el bloque correspondiente a la función seno se puede configurar la amplitud, frecuencia, fase, etc. Para ello se debe hacer doble clic sobre el bloque, con lo que aparecerá una ventana de introducción de parámetros. Para este ejemplo, la configuración del bloque *Sine* será la que puede verse en la Figura 11:

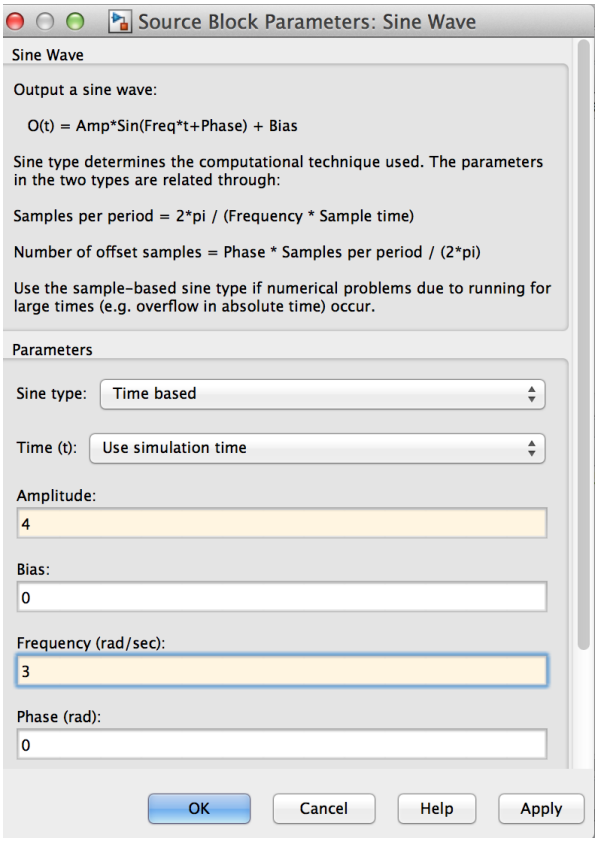
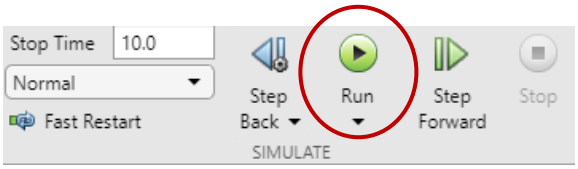


Figura 13 Parámetros de la función seno

2.4.4. Ejecución del programa (simulación)

En Simulink, la ejecución del código generado se conoce como *simulación.* Hay varias maneras de lanzar la simulación:

- Pulsar en el icono *Run* dentro de la pestaña *simulate*



- Acceso rápido: crtl+t

Para poder conocer el resultado de la simulación, se debe abrir el scope pulsando dos veces con el ratón (figura 11)

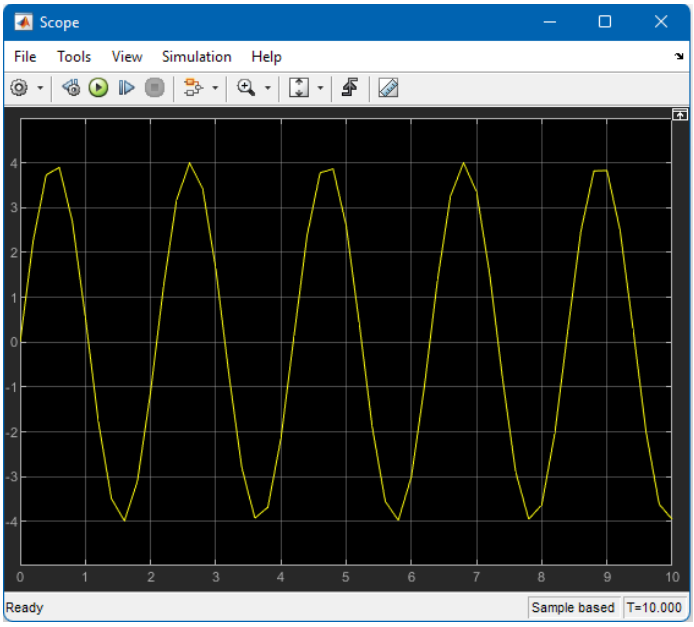
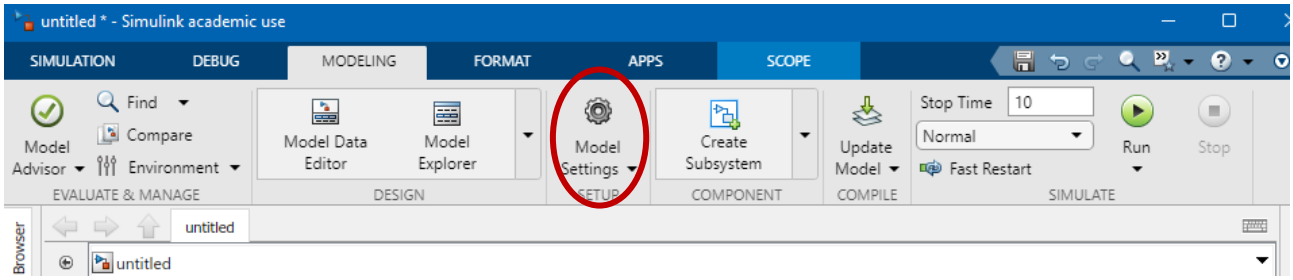


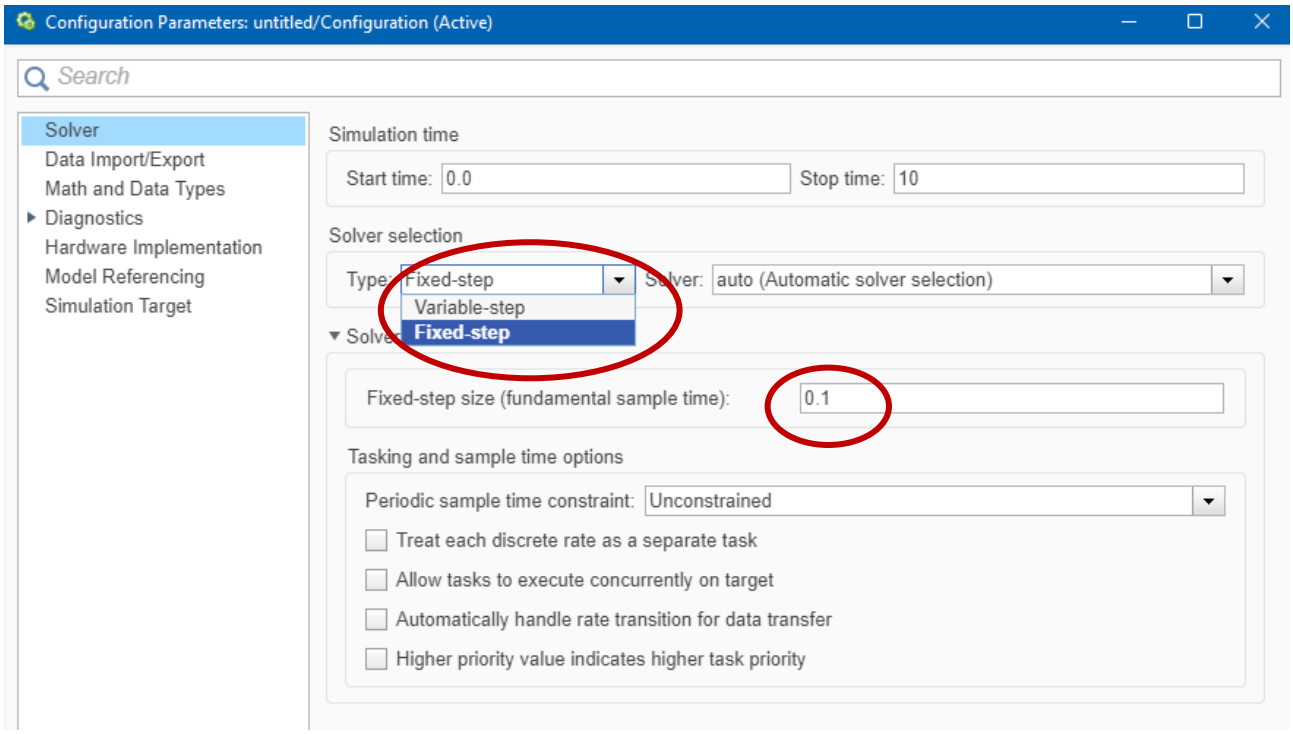
Figura 14: Resultado de la simulación

En la figura seguramente aprecies que la onda del seno no es suave, como debería corresponder a la función matemática. Esto es debido a la configuración del *Solver* de Simulink. El Solver aplica un método de numérico para resolver las ecuaciones -habitualmente diferenciales- presentes en el modelo. En este momento, el *Solver* está configurado de manera automática, por tanto, Simulink selecciona y configura el *Solver* equilibrando la velocidad de resolución con la precisión del mismo.

Para lograr una representación más suave de la curva, vamos a configurar el *Solver* para que resuelva las ecuaciones utilizando un intervalo temporal fijo, de manera que se obtengan más puntos en la resolución de las ecuaciones. Puedes acceder desde la pestaña *Modeling,* o mediante el acceso rápido Crtl+t



Una vez dentro de las opciones de configuración, vamos a seleccionar *Solver type => Fixed Step*. Accedemos a *Solver details*, e introducimos en *Fixed-step size* el valor 0.1



Con estas opciones, Simulink realizará 10 iteraciones por segundo.

Ejercicio 1. Utilización de Matlab Function.

Sobre el modelo anterior:

1. Modifica el tiempo de paso de la simulación para que el modelo se ejecute 1000 veces por segundo.
2. Crea una función en Matlab que reciba como entrada la señal seno, y tenga dos salidas (Salida1 y Salida2) de manera que:
    a. Salida1 cuente el tiempo que la señal seno tenga valor positivo, sin perder el conteo y mostrando el último valor de la cuenta cuando la señal seno sea negativa.
    b. Salida2 muestre el resultado de e^{-t} durante los 10 primeros segundos de simulación y de $(t-10)^2$ a partir de los 10 segundos.
3. Representa la señal de entrada y ambas señales de salida de la función en un mismo *Scope.*

4. Simulación del control de bajo nivel de un manipulador industrial.

El propósito de esta práctica es simular el control de bajo nivel de un manipulador industrial. Para ello vamos a hacer uso de la *toolbox Simscape Multibody*™ (anteriormente *SimMechanics*™, https://es.mathworks.com/products/simscape-multibody.html), la cual proporciona un entorno de simulación multicuerpo para sistemas mecánicos en 3D, como, por ejemplo, sistemas robóticos móviles o multiarticulares. Esta *toolbox* permite modelar sistemas multicuerpo utilizando bloques que representan sólidos, articulaciones, restricciones, elementos de fuerza y sensores. Simscape Multibody formula y resuelve las ecuaciones de movimiento de todo el sistema mecánico. Además, puede importar a su modelo montajes CAD completos, incluidas todas las masas, inercias, articulaciones, restricciones y geometría en 3D. Una de sus grandes ventajas es que genera una animación en 3D automáticamente que permite visualizar el movimiento y la dinámica del sistema.

La siguiente figura, muestra el entorno de simulación con el manipulador industrial con el que vamos a trabajar.

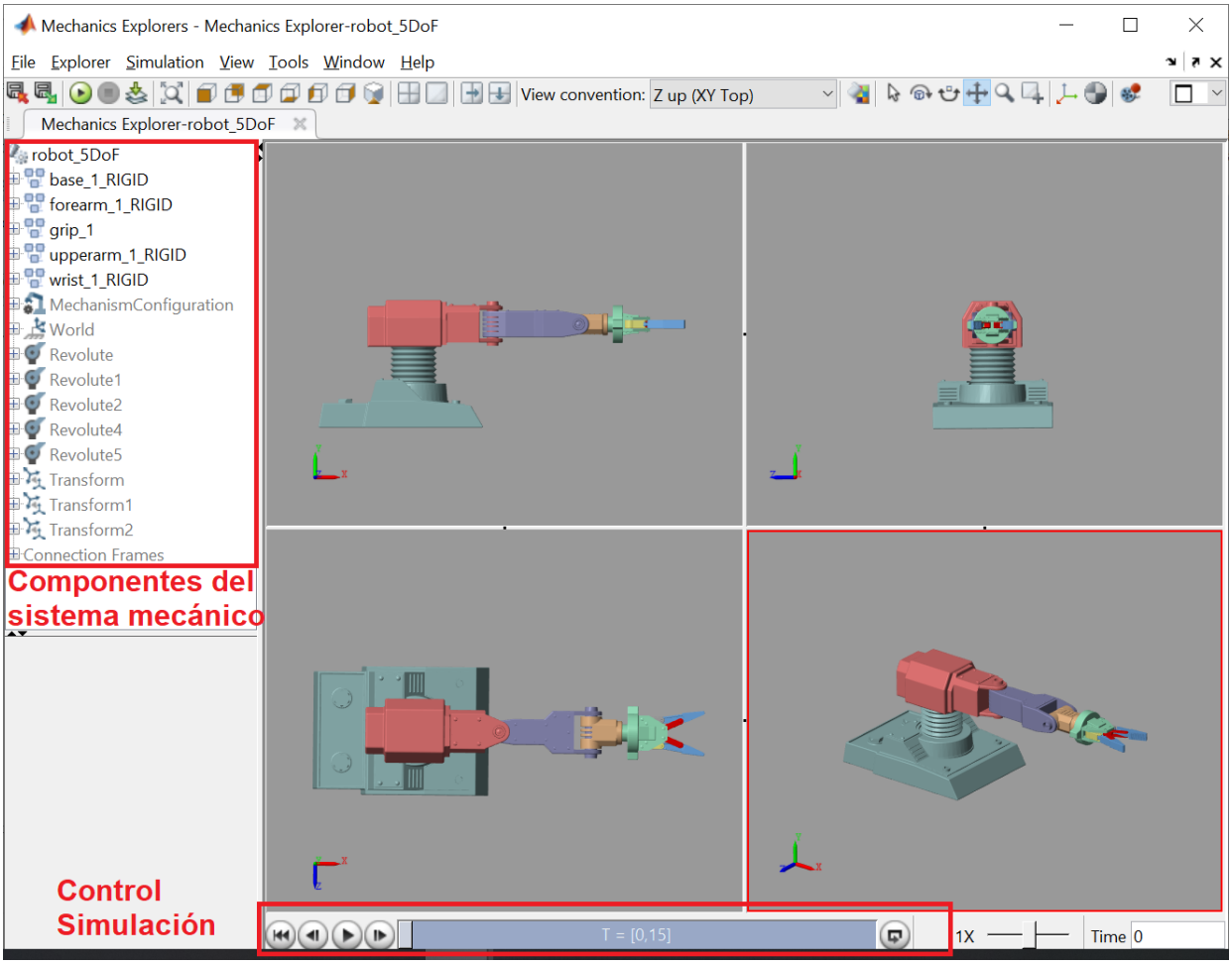


Figura 15: Entorno se simulación de Simscape Multibody.

Se trata de un robot industrial de 5 grados de libertad, todos rotacionales, que tiene las siguientes dimensiones:

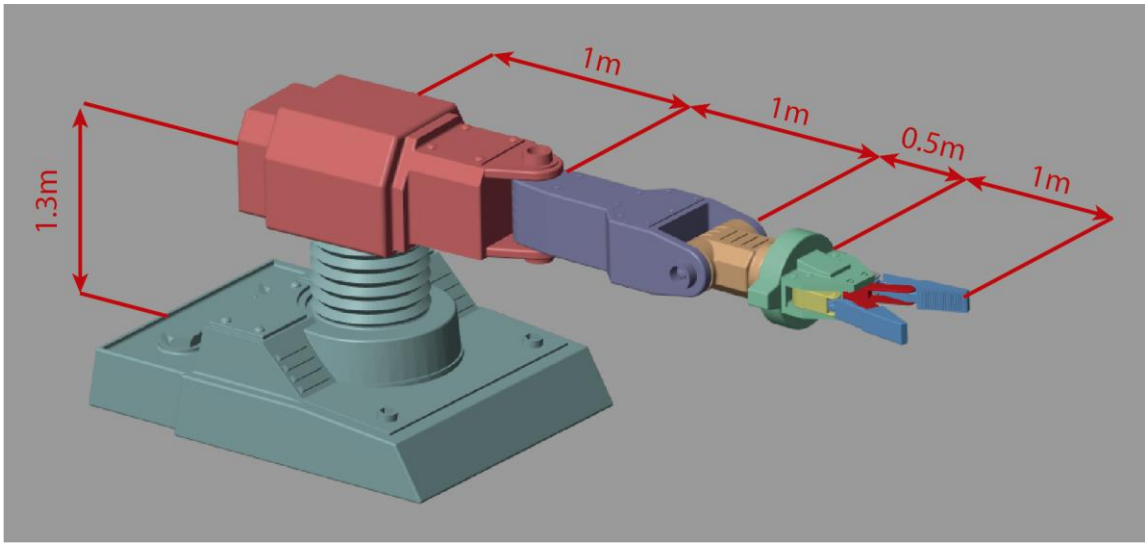


Figura 16: Dimensiones del robot manipulador de 5 grados de libertad

Este sistema robótico, está modelado a través del siguiente modelo de simulink (archivo *robot_5DoF_archivOriginal_R2021a.slx*) que podrás encontrar disponible en el aula virtual. Para cargar los parámetros del modelo, es necesario que ejecutes previamente el archivo *sm_robot_5DoF_DaraFile.m*, también disponible en el aula virtual.

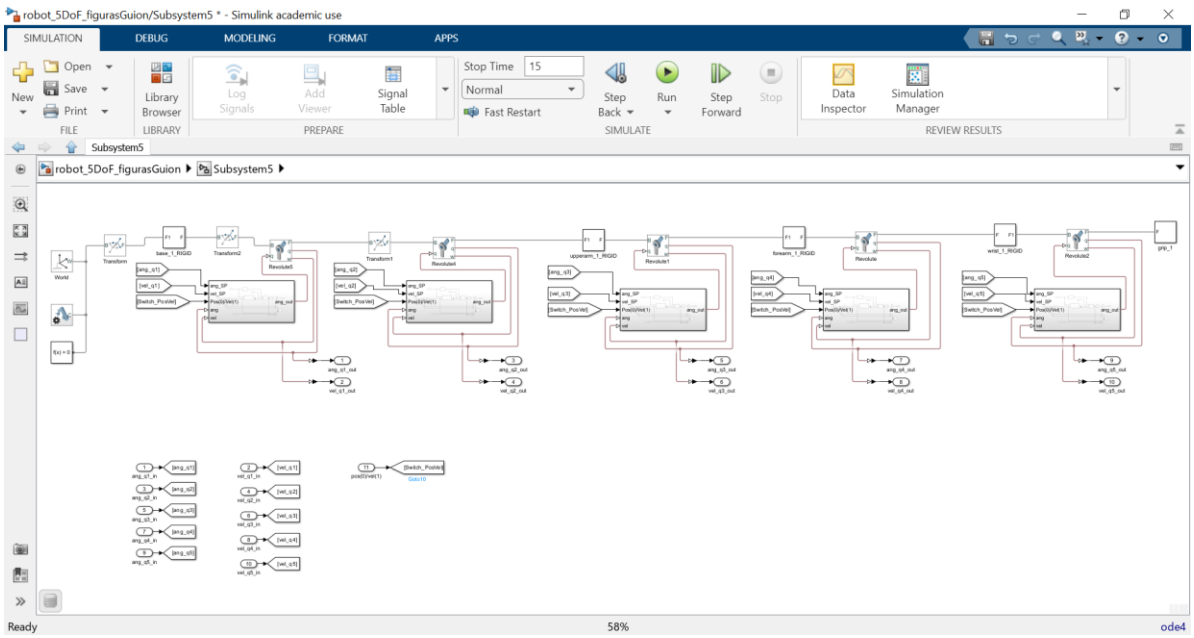


Figura 17: Modelo simulink del manipulador

En lo que respecta al alcance de esta práctica, la dinámica y el controlador de las articulaciones del robot será transparente para nosotros. De esta manera, nosotros consideraremos el modelo del robot como una caja negra, con 11 entradas y 10 salidas:

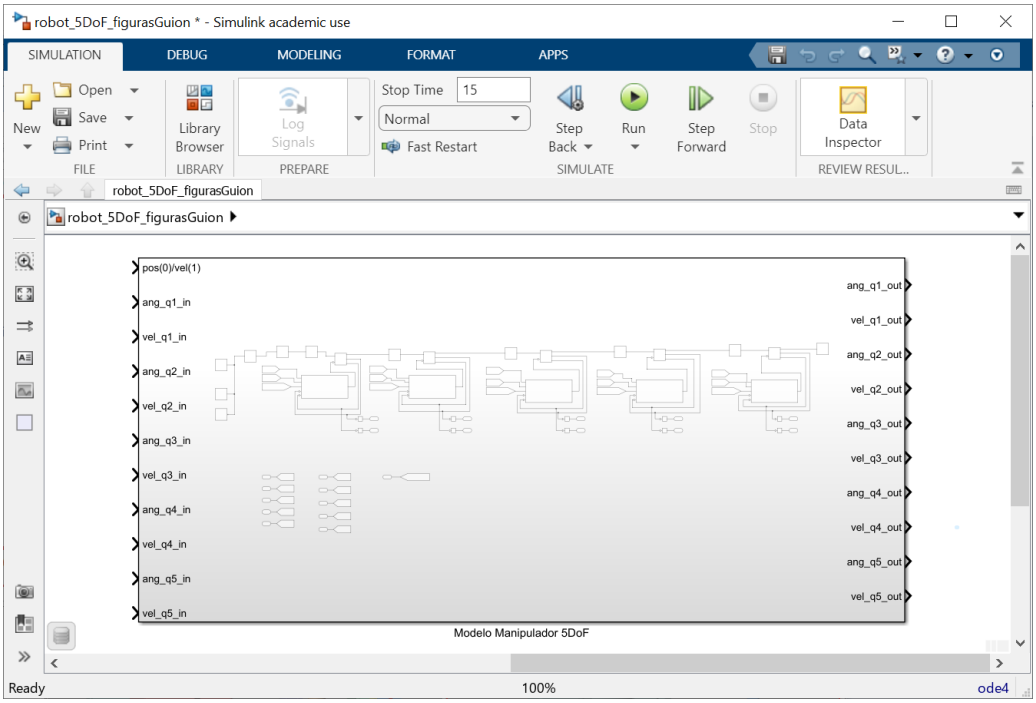


Figura 18: Modelo de caja negra del manipulador

El modelo considera una entrada común a todas las articulaciones:

- *pos(0)/vel(1):* esta entrada determina si las articulaciones serán controladas en posición o en velocidad. Para controlar las articulaciones en posición, esta entrada debe estar recibiendo una constante igual a '0'. En caso de querer controlarlas en velocidad, la entrada ha de recibir una constante de valor '1'.

El resto de entradas y salidas dan información del comportamiento de cada articulación. De manera que para una articulación 'x' (con x de 1 a 5), tendremos:

- *ang_qx_in:* referencia de posición para la articulación *'x'*. Se trata de la posición a la que se comanda llegar a la articulación. Sólo se utiliza en el control de posición del robot.
- *vel_qx_in:* referencia de velocidad para la articulación *'x'*. Se trata de la velocidad de giro a la que se comanda girar la articulación. Es válida tanto para el control de posición como para el control de velocidad del robot.
- *ang_qx_out:* posición instantánea de la articulación del robot.
- *vel_qx_out:* velocidad instantánea de giro de la articulación del robot.

Ejercicio 2. Control articular del robot manipulador.

Comprueba el funcionamiento de los controladores de posición y velocidad del robot. Utilizando los bloques del menú *Sources* envía diferentes consignas de posición y velocidad al robot de manera que:

1. Comanda en velocidad la primera articulación, manteniendo fija la posición del resto. Utiliza una señal seno con frecuencia 0.5Hz, amplitud 5 rad/s y sin offset. Representa la posición y velocidad instantánea de la articulación y comprueba que el sistema se comporta según lo esperado.

2. Genera las señales de entrada para que todas las articulaciones vayan de la posición 0 rad (posición inicial de las articulaciones) a pi/2 rad. Utilizando siempre la misma velocidad en todas las articulaciones, simula el comportamiento ante tres velocidades diferentes (2.5 rad/s, 5 rad/s y 10 rad/s). Representa las posiciones y velocidades instantáneas del robot y comprueba que el funcionamiento es coherente con lo esperado.
3. Comanda movimiento en todas las articulaciones, realiza movimientos de 0 rad a pi/2 rad en las articulaciones impares (1, 3 y 5) y de 0 a pi rad en las articulaciones pares (2 y 4). ¿Qué ocurre cuando se utiliza la misma velocidad en todas las articulaciones?

De esta manera, como se ha podido comprobar, es posible realizar el movimiento descoordinado del robot: el movimiento se inicia a la vez en todas las articulaciones, pero no termina en todas al mismo tiempo. En las aplicaciones reales, este no es un funcionamiento válido para el robot. En su lugar, el movimiento debería ser coordinado, de manera que el movimiento en todas las articulaciones dure el mismo tiempo, empezando y terminando en todas las articulaciones a la vez.

Ejercicio entregable 1 (2 puntos). Control coordinado del robot en el espacio articular.

Crea una función de Matlab en el modelo Simulink del manipulador que configure los comandos de posición y velocidad en cada articulación para que se realice el movimiento de manera coordinada. La función debe tener como entrada el conjunto de coordenadas articulares y la velocidad de crucero del movimiento. Esta velocidad es la máxima admisible para todos los movimientos en todas las articulaciones. Por este motivo, las velocidades en todas las articulaciones deberán ajustarse para que en ningún caso se supere este límite.

Una vez conseguido el movimiento coordinado, modifica la función que has construido para que se pueda llevar el robot a cuatro posiciones consecutivas una vez alcanzada la anterior. Puesto que se tarda un cierto tiempo en alcanzar una posición determinada, se debe monitorizar la posición actual del robot y compararla con la posición objetivo (con un cierto margen de error) antes de comandar la siguiente.

NOTA PARA LA ENTREGA: se deberá subir al aula virtual un vídeo que refleje el funcionamiento del modo de control pedido. En dicho video deberá aparecer la simulación del robot, los *Scopes* que se consideren necesarios y una narración en la que se explique el procedimiento seguido. Además, se subirá el modelo .slx configurado de manera que al ejecutarse se reproduzca directamente el control pedido.

En este momento, es posible controlar y monitorizar el estado del robot en el espacio articular. Es decir, se puede controlar con coordenadas articulares y conocer la rotación de cada link con respecto a su antecesor. Sin embargo, aún no existe una correspondencia entre el espacio articular y el espacio de la tarea. Utilizando las herramientas vistas en las sesiones de teoría, es posible calcular la cinemática directa e inversa del robot, utilizando la información de la Figura 16.

Ejercicio entregable 2 (2 puntos / 3 puntos con extra). Modelado cinemático directo e inverso del

robot.

Calcula el modelo cinemático del robot manipulador. Para ello:

1. Construye su tabla siguiendo el método de Denavit-Hatemberg y calcula la cinemática directa.
2. Crea una función de Matlab en el modelo Simulink del manipulador que, partiendo de las coordenadas articulares, te indique las coordenadas del *end-effector* en el espacio de la tarea (coordenadas cartesianas X, Y, Z y vectores de orientación $\vec{o}$ y aproximación $\vec{a}$). Esta función debe ser capaz de transformar en tiempo real las coordenadas articulares del robot a coordenadas en el espacio de la tarea.
3. Calcula el modelo cinemático inverso del robot, únicamente para la posición del TCP, sin tener en cuenta su orientación. Para este modelado y de ahora en adelante, se considerará que la articulación 2 y la articulación 5 quedan fijas a 0 rad.
4. Crea una función en un script de Matlab (.m) que calcule las posiciones de las articulaciones para llegar a una posición determinada.
    a. (EXTRA 1 punto). Considera la posición actual del robot para proponer la configuración del robot de manera que la propuesta sea la más próxima posible a la actual.

NOTA: para la resolución de este entregable, considera utilizar previemante las herramientas de cálculo simbólico de MATLAB para facilitar la resolución. Esta toolbox incluye funciones para simplificar expresiones simbólicas (*simplify:* *https://es.mathworks.com/help/symbolic/simplify-symbolic-expressions.html*) y para resolver ecuaciones y sistemas de ecuaciones simbólicas sencillas (*solve*, *https://es.mathworks.com/help/symbolic/sym.solve.html?s_tid=doc_ta*). Ten en cuenta que Simulink no es capaz de utilizar cálculo simbólico. Por ello, es recomendable utilizar un live-script que resuelva la cinemática inversa (haciendo uso de cálculo simbólico si fuera necesario), y, una vez resuelta la cinemática inversa, escribir una función que sí pueda ser ejecutada desde Simulink.

NOTA PARA LA ENTREGA: se deberá subir al aula virtual un documento que detalle los sistemas de referencia utilizados para modelar la cinemática directa del robot, la resolución de las cinemáticas directas e inversas, y las funciones pedidas en este entregable.

Una vez conocidas las cinemáticas directa e inversa del robot, podemos pasar a controlar el movimiento del robot en el espacio de la tarea. El movimiento más simple es aquel que se comanda desde un punto inicial a un punto final sin definir la forma de la trayectoria que habría de seguir el TCP en el proceso, este movimiento es el equivalente al *MoveJ* del controlador de ABB.

Ejercicio entregable 3 (1.5 puntos). Ejecución de movimiento entre puntos.

Utilizando el trabajo que ya has desarrollado para los entregables 1 y 2, crea una función de Matlab en el modelo Simulink del manipulador que ejecute el movimiento consecutivo del robot, iniciando es su posición inicial y pasando por los puntos:

- P1 = (-0.2, 2.4, 2.4)

- P2 = (2.8, -1, 1.8)
- P3 = (1.4, -1.4, 0)

Estos movimientos se realizarán de manera continua, sin detenerse y sin considerar el recorrido intermedio del robot (comportamiento equivalente a la ejecución de tres instrucciones *MoveJ* consecutivas*).* No es necesario que consideres la orientación de la pinza durante el movimiento.

NOTA PARA LA ENTREGA: se deberá subir al aula virtual un vídeo que refleje el funcionamiento del modo de control pedido. En dicho video deberá aparecer la simulación del robot, los *Scopes* que se consideren necesarios y una narración en la que se explique el procedimiento seguido. Además, se subirá el modelo .slx configurado de manera que al ejecutarse se reproduzca el control pedido.

Si para la aplicación de funcionamiento de nuestro robot es necesario definir trayectorias en línea recta dentro del espacio de trabajo (instrucción *MoveL* del controlador ABB), existen dos formas de conseguir este comportamiento. La más burda pero también más simple, consistiría en dividir la trayectoria objetivo en un vector de puntos muy cercanos y ejecutar el movimiento consecutivo entre ellos. Para conseguir una buena aproximación es necesario una resolución suficiente en el vector de puntos, así como garantizar con suficiente precisión (±0.1º aproximadamente) que se ha alcanzado cada punto intermedio antes de comandar la siguiente posición.

Ejercicio entregable 4 (1 punto). Ejecución de movimiento en línea recta mediante vector de puntos.

Partiendo del trabajo ya desarrollado, desarrolla una función de Matlab en el modelo Simulink del manipulador que calcule y ejecute una trayectoria en línea recta entre los puntos P4=(3.25, 0, 1.3) y P5=(0.5, -2.25, 0.1). Analiza el efecto que tiene aumentar y disminuir la resolución en el cálculo de los puntos intermedios. No es necesario que consideres la orientación de la pinza durante el movimiento. Nótese que el punto P4 es muy próximo a la posición completamente extendida del robot.

NOTA PARA LA ENTREGA: se deberá subir al aula virtual un vídeo que refleje el funcionamiento del modo de control pedido. En dicho video deberá aparecer la simulación del robot, los *Scopes* que se consideren necesarios y una narración en la que se explique el procedimiento seguido. Además, se subirá el modelo .slx configurado de manera que al ejecutarse se reproduzca el control pedido.

La otra opción para la ejecución de una trayectoria en línea recta implicaría el control directo de la velocidad de giro de cada articulación. De esta manera es posible obtener una velocidad lineal constante del TCP en la dirección de la trayectoria, lo que describiría una trayectoria en línea recta. Para ello, es preciso previamente calcular la cinemática diferencial del robot.

Ejercicio entregable 5 (2 puntos / 2.5 puntos con extra). Ejecución de movimiento en línea recta mediante cinemática diferencial:

1. Partiendo del trabajo ya desarrollado, calcula la cinemática diferencial del robot (1 punto).

2. Desarrolla una función de Matlab en el modelo Simulink que, partiendo de la cinemática diferencial inversa del robot, controle la velocidad de las articulaciones para que el TCP describa una línea recta entre la posición inicial del robot y el punto P5=(0.5, -2.25, 0.1) No es necesario que consideres la orientación de la pinza durante el movimiento (1 punto).
    a. (EXTRA 0.5 punto). Estudia las posibles configuraciones singulares del robot, analiza su significado y justifica si supone algún problema en la generación de trayectorias rectas.

NOTA: para la resolución de este entregable, considera utilizar las herramientas de cálculo simbólico de MATLAB para facilitar la resolución. Esta toolbox incluye funciones para derivar expresiones simbólicas (*diff:* https://es.mathworks.com/help/symbolic/diff.html) y para el cálculo de la matriz jacobiana (*jacobian*, https://es.mathworks.com/help/symbolic/sym.jacobian.html?s_tid=doc_ta). Recuerda que el cálculo simbólico no es compatible con Simulink, sino sólo una herramienta previa al desarrollo de la función ejecutada en la simulación.

NOTA PARA LA ENTREGA: se deberá subir al aula virtual un vídeo que refleje el funcionamiento del modo de control pedido. En dicho video deberá aparecer la simulación del robot, los *Scopes* que se consideren necesarios y una narración en la que se explique el procedimiento seguido. Además, se subirá el modelo .slx configurado de manera que al ejecutarse se reproduzca el control pedido.