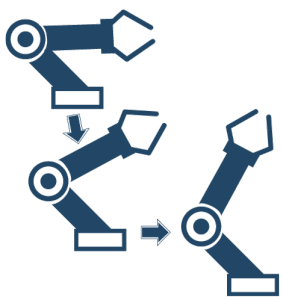
Práctica 2

Controlador cinemático.
Interpolación de trayectorias

V-REP permite simular la dinámica de un robot, con lo que se puede simular de manera más realista cualquier controlador. En esta práctica enlazaremos Matlab con V-REP para probar un controlador cinemático articular básico.



Práctica 2

Control cinemático. Interpolación de trayectorias

Objetivos

En esta práctica se desarrollan algunos de los controladores cinemáticos vistos en las clases de teoría de la asignatura. Previo al diseño e implementación de los controladores, se desarrollará un nuevo interpolador de trayectorias basado en la interpolación con un polinomio cúbico. Para ello, se utilizará Matlab y la Robotics toolbox desarrollada por Peter Corke. En esta toolbox ya se encuentran implementados una serie de interpoladores básicos. Una vez desarrollada la función que devolverá la evolución de una articulación, tanto en posición como en velocidad y aceleración, se implementará el controlador cinemático para el robot Kinova Mico, comunicando Matlab con V-Rep. Así, los objetivos concretos que se buscan alcanzar con la práctica son los siguientes:

- Comprender el funcionamiento de los interpoladores polinómicos.
- Entender la comunicación entre Matlab y V-Rep para poder simular cualquier controlador diseñado en Matlab.
- Comprender los controladores cinemáticos y saber diseñarlos y ajustar sus ganancias.

Robotics toolbox

Descripción del software

La Robotics Toolbox desarrollada por Peter Corke (http://petercorke.com/wordpress/toolboxes/robotics-toolbox) es una librería de Matlab que proporciona una serie de funciones útiles para el estudio, diseño y simulación de la robótica clásica de manipuladores como, por ejemplo, la cinemática, la dinámica y la generación de trayectorias. La toolbox contiene funciones y clases para representar la orientación y posición en el espacio Cartesiano 3D como matrices de rotación o cuaternios. La toolbox también proporciona funciones para manipular y convertir entre tipos de datos tales como vectores, transformaciones homogéneas y cuaternios unitarios, que son necesarios para representar la posición y orientación en el espacio 3D.

La toolbox utiliza un método muy general para representar la cinemática y la dinámica de los robots manipuladores como objetos de MATLAB: el usuario puede crear objetos de robot para cualquier manipulador y se proporcionan varios ejemplos para robots modernos conocidos, como ciertos modelos de Kinova, Universal Robotics, Rethink, así como robots clásicos como el Puma 560 y el brazo de Stanford.

La toolbox admite también robots móviles con funciones para modelos de robots móviles (monociclo, bicicleta), algoritmos de planificación de ruta (transformación de distancia, D*,

PRM), localización (EKF, filtro de partículas), construcción de mapas (EKF) y localización y mapeo simultáneos (EKF), así como un modelo Simulink de un vehículo no holonómico. La toolbox también incluye un modelo Simulink detallado para un dron.

Las principales ventajas de la toolbox son que:

- Tras 10 revisiones en 20 años, el código está ya depurado y proporciona un punto de comparación para otras implementaciones de los mismos algoritmos.
- Las rutinas generalmente se escriben de manera directa, lo que permite una fácil comprensión, quizás a expensas de la eficiencia computacional.
- Dado que el código fuente está disponible, se potencia la comprensión de las distintas implementaciones.

Una vez instalada la toolbox, basta con editar cualquier archivo para ver su código, con lo que es posible aprender de qué forma se implementan funcionalidades como la cinemática directa o inversa de un robot, o un interpolador de trayectorias quíntico como el definido en tpoly.m.

V-Rep

Descripción del software

V-REP (Virtual Robot Experimentation Platform) es un simulador de robótica con una extensa capacidad, funcionalidades y APIs (conjunto de funciones y métodos para ser utilizado por otro software).

El simulador de robótica V-REP dispone de una interfaz o entorno de desarrollo (IDE) en la que cada modelo u objeto está basado en una arquitectura de control distribuido, de modo que cada objeto se puede controlar individualmente por un script, un plug-in, un nodo de ROS, una aplicación remota a través de su API, o una solución externa. Estos controladores se pueden desarrollar en diversos lenguajes, como LUA (lenguaje propio de V-REP), C, C++, Java, Python o MATLAB.

La siguiente lista son sólo una de las posibles aplicaciones de uso de V-REP:

- Simulación de sistemas de fabricación automatizados.
- Monitorización remota.
- Control de hardware.
- Prototipado rápido y verificación.
- Monitorización de seguridad.
- Enseñanza de robótica.
- Presentación de producto.

V-REP puede ser utilizado como una aplicación individual (a través de su IDE) o puede ser incrustada en una aplicación cliente principal. Dispone de un intérprete de LUA (lenguaje de programación) que lo hace muy versátil, con la capacidad de combinar funcionalidades de alto y bajo nivel para obtener resultados espectaculares con pocas líneas de código. V-Rep puede descargarse desde el siguiente enlace (http://www.coppeliarobotics.com).

Una de las grandes ventajas de V-REP es que permite la programación con cinco técnicas diferentes, que pueden utilizarse al mismo tiempo. Estas técnicas son:

- Scripts embebidos: el lenguaje propio de V-REP es LUA. Con él implementa un script principal que controla la funcionalidad general del sistema, llama a las distintas funciones y a los child scripts, asociados a un objeto concreto de la escena y que pueden registrar publicadores/suscriptores de ROS.
- Add-ons: V-REP permite add-ons a través de scripts LUA. Se pueden utilizar como funciones independientes (para la escritura de los importadores/exportadores, por ejemplo), o como código ejecutado de forma recursiva.
- Plug-ins: se pueden utilizar para hacer simulaciones con comandos LUA desde un script embebido o para extender la funcionalidad de un objeto. La interfaz de ROS, por ejemplo, se implementa a través de un plug-in.
- Aplicaciones remotas, a través de su API: este método permite a una aplicación externa conectarse a V-REP con comandos remotos a través de un socket y tener el mismo código cargado en un robot y en el simulador, esta es la técnica que utilizaremos para comunicar V-Rep con Matlab.
- Nodos de ROS: V-REP implementa los nodos de ROS con un plug-in que permite a ROS invocar comandos de V-REP a través de servicios de ROS o transmitir información a través de publicadores/suscriptores de ROS. Los publicadores/suscriptores se pueden habilitar con una llamada de servicio o directamente desde V-REP, a través de un script.

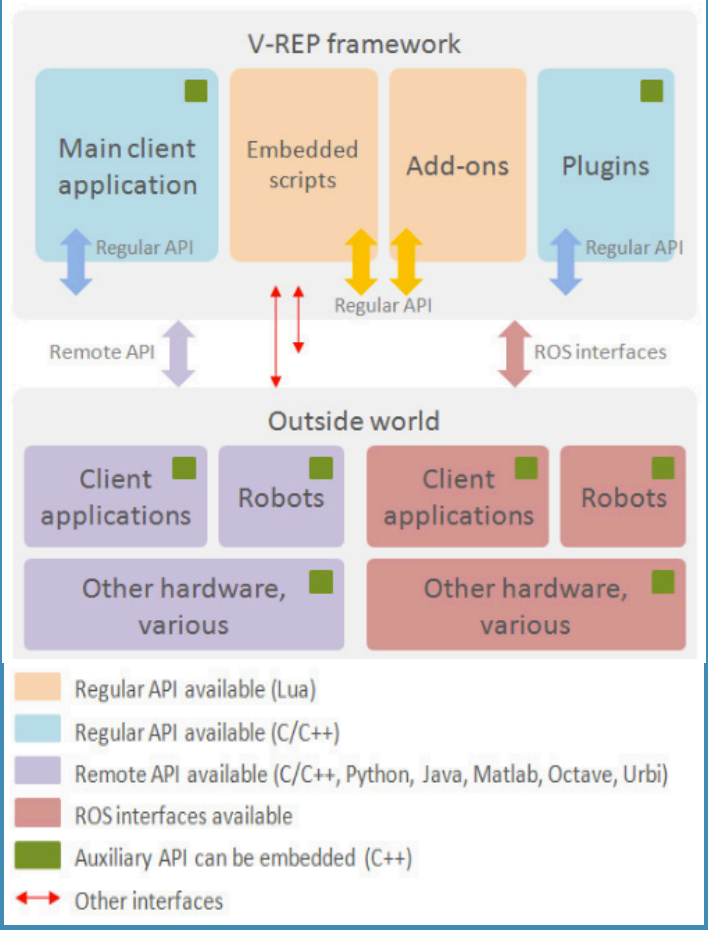


Figura 1. Framework de V-REP.

Fuente: Coppelia Robotics

Tipos de objetos en V-REP

- Articulaciones: elementos que unen dos o más objetos de la escena con uno o tres grados de libertad (pueden ser prismáticos, rotacionales o esféricos). Pueden funcionar en distintos modos, como son el modo fuerza o el modo de cinemática inversa, entre otros.
- Formas: son engranajes triangulares y se utilizan para simular y visualizar cuerpos rígidos. Se pueden optimizar para calcular la respuesta dinámica a una colisión, por ejemplo. Sirven de base para otros cálculos de otros objetos de la escena.
- Sensores de proximidad: calculan la distancia mínima exacta a la parte de una forma contenida en un cuerpo, en lugar de hacer la detección basada en rayos.
- Sensores de visión: permiten extraer información de una escena en aspectos como colores, formas, mapas de profundidad, etc. Funcionan a partir de aceleración de *hardware* (OpenGL).
- Sensores de fuerza: representan uniones rígidas entre formas que pueden grabar fuerzas o pares aplicados y detectar cuándo se ha superado un umbral definido.
- Gráficos: pueden recoger datos de diversos tipos y mostrarlos individualmente o combinados con otras métricas en gráficos X/Y o en curvas 3D.
- Cámaras: permiten visualizar la escena cuando se asocian a un *viewport*.
- Luces: iluminan la escena u objetos concretos de la escena y son visibles tanto para las cámaras como para los sensores de visión.
- *Paths*: permiten configurar la definición de movimiento en el espacio (sucesión de traslaciones, rotaciones y pausas). Se utilizan para crear trayectorias de seguimiento y guiado de robots o para generar el movimiento de una cinta transportadora.
- *Dummies*: son estructuras de referencia que se pueden utilizar para diversas operaciones, generalmente en conjunción con otros objetos de la escena, por lo que se pueden considerar "comodines".
- *Mills*: representan cuerpos convexos configurables con los que se pueden hacer cortes en objetos.

Inclusión de un robot (UR3)

V-REP incluye diversos modelos de robot ya creados, tanto móviles como no móviles. Algunos son robots comerciales muy conocidos, de fabricantes como ABB o KUKA, y otros son modelos genéricos ya predefinidos, como manipuladores de 7GDL, drones o seguidores de línea. Cualquiera de estos modelos viene ya ensamblado, con sus dimensiones reales y materiales. Es posible, también, crear un robot desde cero a partir de formas básicas y articulaciones, e ir ensamblando las partes. Algunos de los modelos disponibles se pueden ver en la Figura 3.

La Figura 4 muestra un UR5 en una escena vacía de V-REP.

Figura 3. Ejemplo de robots disponibles en V-REP

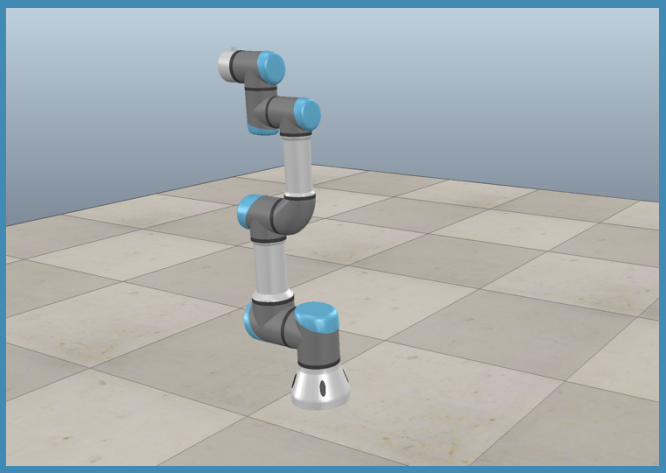


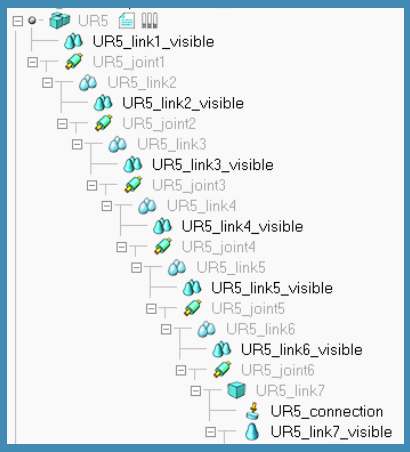
Figura 4.

Robot UR5 en V-REP

Cuando se introduce el robot en la escena, se muestra su estructura en el árbol de jerarquías, con las dependencias de sus eslabones y articulaciones (ver Figura 5).

Figura 5.

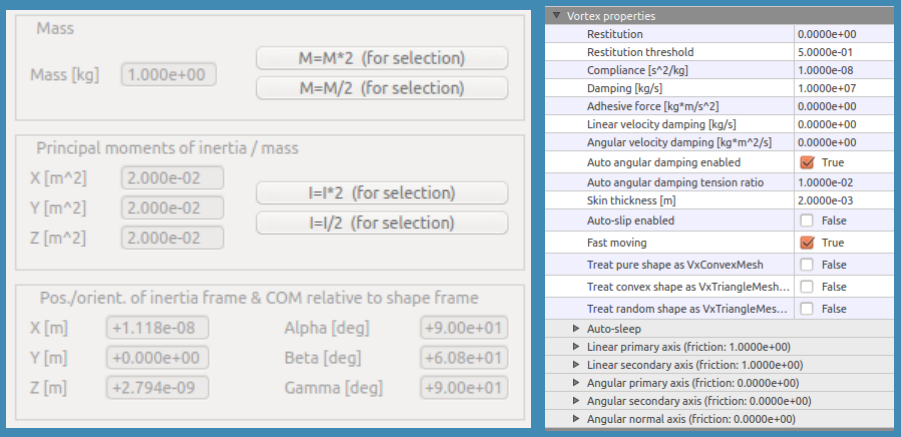
UR5. Partes



Al ser un modelo predefinido, sus partes están ya ensambladas, y entrando en su pestaña de propiedades, se pueden ver todas sus características: masa, inercia y propiedades físicas y dinámicas (ver Figura 6).

Figura 6.

Propiedades UR5



Con estos parámetros ya configurados, la simulación será realista y el robot se comportará como lo haría en un contexto real.

Para su control, V-REP tiene distintos módulos y formas de operación. Si se seleccionan las articulaciones, para cada una de ellas se puede configurar su posición mínima y rango, el modo de funcionamiento, y activar o desactivar el motor, pudiendo definir su par máximo y sus propiedades específicas.

Respecto al modo de operación, V-REP ofrece distintas estrategias, concretamente:

- *Passive mode*: la articulación no tiene controlador y funciona como un eslabón. Aun así, se puede modificar su posición utilizando funciones del API.
- *Inverse kinematics mode*: también funciona como una articulación pasiva, pero se tiene en cuenta en los cálculos de la cinemática inversa.
- *Dependent mode*: la posición de la articulación depende linealmente de otra articulación.

- *Motion mode*: este modo ya no se usa.
- *Torque/force mode*: la articulación se simula a través del *dynamics module* (si está marcada la opción "*dynamically enabled*"). En este caso, puede estar controlada en fuerza/par, en velocidad o en posición.

Configuración de la simulación

En cualquier momento durante la creación y configuración de los modelos y objetos añadidos, se puede simular la escena utilizando su barra de herramientas (ver Figura 7).

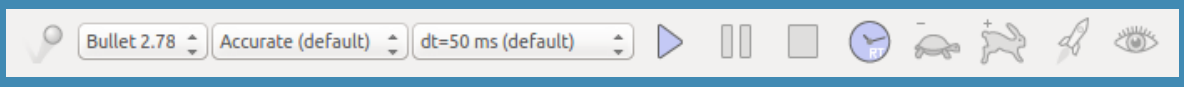


Figura 7. Barra de herramientas de simulación

Con la simulación es posible comprobar el comportamiento dinámico de los elementos, así como observar la interacción entre ellos, incluyendo escenarios en los que haya colisiones, por ejemplo.

Para realizar la simulación dinámica, V-REP se apoya en cuatro motores físicos diferentes, y el usuario puede seleccionar cuál de ellos utilizar e ir cambiándolos para ver cuál responde mejor a las características de la escena.

Cada uno de los cuatro posibles motores físicos será más o menos adecuado dependiendo de si se busca más precisión o más velocidad, y de los elementos que se estén simulando, ya que no todos ellos ofrecen soporte para todos los mecanismos.

Los cuatro motores son:

- *Bullet physics* (versiones 2.78 y 2.83): motor de código abierto que soporta detección de colisiones y simulación dinámica tanto de cuerpos rígidos como blandos, aunque esta última característica no existe en V-REP. Muy utilizado en entornos de videojuegos y efectos visuales en películas.
- *Open Dynamics Engine*: también de código abierto y con dos componentes principales, simulación dinámica de cuerpos rígidos y detección de colisiones.
- *Vortex Dynamics*: motor comercial muy realista, que se utiliza generalmente para simulaciones que requieren mucha precisión y también en entornos de investigación. En la versión gratuita en V-REP se permiten simulaciones de hasta 20 segundos.
- *Newton Dynamics*: es una librería multiplataforma y adecuada para simulación en tiempo real. En V-REP está disponible su versión BETA.

Los scripts para controlar los robots que se simulan con V-Rep se escriben en lenguaje Lua mediante el editor de texto interno que tiene el programa. No obstante, V-Rep también ofrece una API remota para controlar la simulación desde una aplicación externa (como un script de Matlab o un programa en C++) o un hardware remoto (un robot real, otro ordenador conectado a la misma red…).

Para que Matlab pueda conectarse bien con V-Rep habrá que importar algunas funciones. Ve al directorio dónde tengas instalado V-Rep->programming->remoteApiBindings->matlab->matlab y copia todos los ficheros acabados en '.m' que hay dentro. Ahora ve a tu directorio de trabajo y pega los ficheros allí.

Aún necesitas otro fichero. Vuelve al directorio de V-Rep->programming->remoteApiBindings->lib y encontrarás carpetas para Linux, Mac o Windows. Dentro de cada una encontrarás la librería correspondiente a tu sistema operativo. Cópiala y pégala en el directorio de trabajo de Matlab, junto con el resto de archivos que acabas de copiar hace un momento.

Para programar el robot localizado en V-REP, se hará uso del API para Matlab de V-REP. La referencia de las funciones del API para Matlab puede encontrarse en el siguiente enlace:

(http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm)

Es importante que, para que estén bien sincronizados Matlab y V-REP, se utilice el envío de velocidades con las funciones simxSynchronous y simxSynchronousTrigger.

Cuestiones a realizar

En esta práctica de control y programación de robots se trabajará sobre las funciones de la toolbox de Robótica que permiten planificar trayectorias para el movimiento del robot. Se conectará con el simulador V-Rep para simular el comportamiento de un Kinova Mico y se implementará un script en Matlab con un controlador cinemático clásico (realimentación de posición y prealimentación de velocidad).

A continuación, se enumeran las cuestiones a realizar en esta segunda práctica:

1. Programar una función (tpoly3.m) que implemente un interpolador cúbico. Aceptará como parámetros de entrada las condiciones iniciales y finales de posición y velocidad. La velocidad por defecto inicial y final será cero. También se le indicará el tiempo en el que se debe realizar el movimiento mediante un vector de tiempos con el que se permitirá introducir el tiempo de paso entre muestras (por ejemplo, "0:0.1:3" permitirá obtener muestras de la interpolación cada 0.1s hasta llegar a 3s). Los parámetros de entrada de la función serán los siguientes:

    - Un escalar con el valor inicial a interpolar.
    - Un escalar con el valor final.
    - Un vector con el tiempo de cada una de las muestras que se quieren obtener.
    - Un escalar con el valor inicial de velocidad (opcional).
    - Un escalar con el valor final de velocidad (opcional).

    Como salida, la función devolverá la evolución del parámetro interpolado (ya sea un valor de posición o de giro de una articulación), así como la evolución de la velocidad y la aceleración. Si no se indica ningún parámetro, se representará en tres gráficas distintas la evolución de estos tres parámetros (posición, velocidad y aceleración) (Esta parte es opcional).
    Evaluar el interpolador con dos valores distintos (inicial y final).

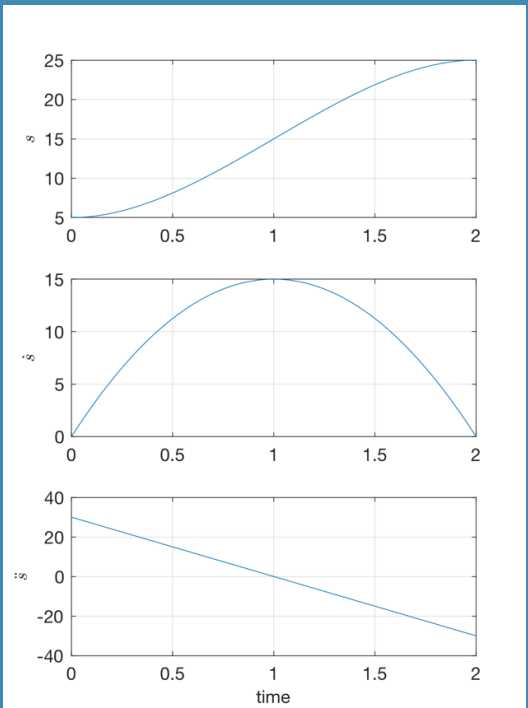


Figura 8. Salida de la función tpoly3.m si no se piden parámetros de salida.

2. Programa el controlador cinemático articular de realimentación en posición y prealimentación en velocidad para la primera articulación del robot Kinova Mico. El controlador se implementará en Matlab, y se simulará en V-REP.
   Introduce un robot Kinova Mico en una escena vacía de V-REP, ajusta su primera articulación en modo de par/fuerza y en el cuadro de diálogo de propiedades dinámicas, desactiva el controlador del motor. Para aceptar peticiones desde un cliente, debemos iniciar el servidor en Lua con:
   simRemoteApi.start(23000,1300,false,true);
   El script de Matlab debe dibujar la gráfica del error en posición articular a lo largo de la trayectoria, la gráfica de la posición articular en cada iteración comparada con la posición articular de referencia, la gráfica de la evolución del par articular leído de la articulación, la gráfica de la evolución de la velocidad articular leída de la articulación, así como del error de velocidad articular enviada al robot y la velocidad articular leída del robot.
3. Genera una trayectoria mediante un polinomio cúbico que alimente al controlador cinemático implementado en la cuestión anterior (para un desplazamiento de 90 grados). Con una ganancia K=20, indica cuál es el tiempo mínimo que permite el seguimiento de la trayectoria sin saturar los pares articulares.
4. Con una trayectoria generada con un polinomio cúbico, donde el robot realice un desplazamiento relativo de 45 grados en 1 segundo, ¿qué ganancia del sistema provoca saturación en el par articular?