Universidad de Ingeniería y Tecnología

FUNDAMENTOS DE ROBÓTICA

Prof. Oscar E. Ramos Ponce

Guía de Laboratorio 7:

Control Dinámico

Lima - Perú

2017 - 1

Laboratorio 7:
Control Dinámico

1. Objetivos

- Utilizar RBDL (*Rigid Body Dynamics Library*) para obtener el modelo dinámico de un robot a partir de su modelo URDF.
- Implementar un controlador PD con compensación de gravedad en el espacio articular para controlar un robot de 7 grados de libertad.
- Implementar un controlador por torque calculado en el espacio articular para controlar un robot de 7 grados de libertad.

2. RBDL

En la práctica no es usual calcular el modelo dinámico de un robot de manera analítica debido a que los robots pueden llegar a ser demasiado complejos a medida que se incrementa el número de grados de libertad. Lo que se suele utilizar es el método de Newton-Euler implementado de manera eficiente en algún paquete computacional. En este laboratorio se utilizará el paquete denominado RBDL (*Rigid Body Dynamics Library*), desarrollado en la Universidad de Heidelberg, que implementa la dinámica de manera numérica con algoritmos como RNEA, CRBA y ABA.

Pregunta

1. (0.5 pt) ¿Qué significan las siglas *RNEA*, *CRBA* y *ABA*?

2.1. Instalación de RBDL

RBDL está implementado en C++ y posee un plug-in para leer modelos URDF. Además, posee una interface para utilizar las funciones de C++ desde Python, la cual fue realizada utilizando *Cython*. Debido a que en este laboratorio se desea utilizar los *bindings* de Python, se debe instalar Cython en la máquina con el siguiente comando (en caso de no estar aún instalado):

```
$ sudo apt-get install cython
```

Luego, RBDL puede ser descargado desde su página oficial: `https://rbdl.bitbucket.io`, en la sección *Downloads*. Para mayor facilidad, se puede descargar del aula virtual el archivo denominado `rbdl.zip` que contiene el paquete. Dicho archivo debe ser descomprimido dentro de `lab_ws/src`. Luego, ingresar a la carpeta `rbdl` y ejecutar los siguientes comandos:

```
$ cd ~/lab_ws/src/rbdl
$ mkdir build; cd build
$ cmake .. -DRBDL_BUILD_PYTHON_WRAPPER=ON -DCMAKE_INSTALL_PREFIX=
  ~/lab_ws/install -DRBDL_BUILD_ADDON_URDFREADER=ON
$ make
$ make install
```

De no existir errores durante la compilación e instalación, se debe indicar a Python dónde se encuentran las librerías instaladas. Para esto se añade

```
$ echo "export LD_LIBRARY_PATH=~/lab_ws/install/lib/x86_64-linux-gnu:
  $ LD_LIBRARY_PATH" >> ~/.bashrc
$ echo "export PYTHONPATH=~/lab_ws/install/lib/python2.7/site-packages:
  $ PYTHONPATH" >> ~/.bashrc
$ source ~/.bashrc
```

Para probar que Python reconoce a rbdl, abrir Python dese el terminal (escribir `python`) y dentro de Python ejecutar: `import rbdl`. De no existir ningún mensaje de error se puede proseguir con las siguientes partes.

3. Modelo Dinámico del Robot

Dadas las coordenadas articulares $\boldsymbol{q}$ de un robot manipulador, su modelo dinámico está dado por

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{1}$$

donde $M(\boldsymbol{q})$ representa la matriz de inercia, $C(\dot{\boldsymbol{q}}, \boldsymbol{q})$ la matriz de fuerzas centrífugas y de Coriolis, $\boldsymbol{g}(\boldsymbol{q})$ el vector de fuerzas debidas a la aceleración de la gravedad, y $\boldsymbol{\tau}$ el vector de torques generalizados que actúan sobre cada articulación. De manera compacta, este modelo puede ser representado como

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{b}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \boldsymbol{\tau} \tag{2}$$

donde el vector $\boldsymbol{b}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$ es a veces denominado vector de efectos no lineales.

La mayoría de paquetes computacionales brindan una función que calcula los torques como en (1), o equivalentemente en (2). Supongamos que dicha función se denomina `NE` y que su formato es `tau = NE(q, dq, ddq)`, donde `dq` representa las velocidades articulares y `ddq` las aceleraciones articulares.

Preguntas. Utilizando solamente la función `NE` responder las siguientes preguntas (brindar las condiciones necesarias para hallar lo solicitado). Justificar en cada caso la respuesta dada.

1. (0.5 pt) ¿Cómo se puede obtener el vector de gravedad $\boldsymbol{g}(\boldsymbol{q})$?
2. (0.5 pt) ¿Cómo se puede obtener el vector de efectos no lineales?
3. (0.5 pt) ¿Cómo se puede obtener la matriz $C(\dot{\boldsymbol{q}}, \boldsymbol{q})$? ¿La solución será única?

Tareas. En RBDL, la función que implementa la dinámica inversa (equivalente a la función `NE` mencionada anteriormente) se denomina `InverseDynamics`. Descargar el archivo comprimido `lab7.zip` y descomprimirlo dentro de `lab_ws/src`. Dentro de `lab7/src` utilizar el archivo llamado `modelo_din.py`. La ayuda sobre cada función de RBDL puede encontrarse en `https://rbdl.bitbucket.io/`, dentro de *Modules/Dynamics*. Para ejecutar el archivo de python utilizar la forma interactiva: `ipython -i modelo_din.py`

1. (0.75 pt) Utilizando las respuestas de las preguntas anteriores, obtener los vectores de gravedad, de efectos no lineales y de la matriz de Coriolis y fuerza centrífuga a partir de la función `InverseDynamics` que es equivalente a la función `NE` mencionada anteriormente. Incluir en el reporte los vectores y matrices obtenidos (para un mejor formato se puede usar la función `np.round` al mostrar los valores).

2. (0.75 pt) Usando la función `CompositeRigidBodyAlgorithm` obtener directamente la matriz de inercia. De manera semejante, usando la función `NonlinearEffects` encontrar los efectos no lineales. Verificar si los valores en ambos casos sean coherentes con los anteriormente encontrados.

3. (0.5 pt) Verificar que las matrices calculadas satisfacen la expresión (2) (realizando de manera explícita las multiplicaciones) para los valores articulares utilizados.

Adjuntar el programa de python utilizado o las líneas usadas.

4. Control Dinánico

En esta sección se implementará algunos controladores que utilizan el modelo dinámico del robot. Se hará uso de los valores obtenidos anteriormente.

4.1. Control PD+Compensación de la Gravedad

En clase se vio la forma que toma el controlador de tipo Proporcional-Derivativo con compensación de la gravedad en el espacio articular. Aquí se utilizará la implementación de dicho controlador para mover al robot de siete grados de libertad desde una configuración inicial a una configuración final; todo en el espacio articular. Para esta parte se utilizará el script Python llamado `control_pdg`.

Tareas

1. (2 pts) Implementar el controlador PD con compensación de gravedad utilizando matrices K_p y K_d diagonales con todos los elementos de la diagonal siendo iguales (de manera equivalente, se puede usar solamente escalares para k_p y k_d dado que se utilizará valores iguales). Encontrar algunos valores de ganancia proporcional y derivativa que permitan alcanzar los valores articulares deseados. Adjuntar el código desarrollado o copiar la parte relevante en el reporte.

2. (1 pt) Cuando el sistema converge al valor final, graficar la evolución temporal de cada una de las articulaciones comentando el tipo de comportamiento que se observa.

3. (2 pt) Graficar la evolución temporal de las coordenadas x, y y z del efector final del robot e indicar si convergen al valor Cartesiano deseado (a partir de la configuración deseada). ¿Se observa un comportamiento similar al observado en el espacio articular? ¿Por qué sucede eso?

4. (2 pt) Utilizar dos configuraciones articulares deseadas, diferentes a la anterior, e indicar si las ganancias encontradas anteriormente brindan el mismo tipo de respuestas para cada caso (analizando el comportamiento de las articulaciones).

4.2. Control por Dinámica Inversa

En esta sección se busca implementar un controlador que utilice dinámica inversa en el espacio articular (también llamado controlador de torque calculado). Copiar el script utilizado para la sección anterior y llamarlo `control_dininv` (asegurarse de que es ejecutable). En las siguientes tareas, utilizar dicho script.

Tareas.

1. (1 pt) ¿Qué hace un controlador por dinámica inversa? ¿En qué casos podría brindar resultados no muy precisos?¿Cómo se podrían resolver dichos casos?

2. (2 pt) En el script `control_dininv` implementar un controlador por dinámica inversa en el espacio operacional, donde la dinámica del error sea de segundo orden. Adjuntar el código desarrollado o copiar la parte relevante del controlador en el reporte.

3. (2 pt) Determinar valores de ganancia adecuados para obtener un comportamiento aproximadamente críticamente amortiguado para el error. Graficar la evolución temporal de cada una de las articulaciones comentando el tipo de comportamiento obtenido. Comentar las semejanzas/diferencias con la respuesta obtenida usando el controlador PD+compensación de gravedad.

4. (1 pt) Graficar la evolución temporal de las coordenadas x, y y z del efector final del robot. ¿Presentan un comportamiento similar al del espacio articular?

5. (1 pt) Utilizar dos puntos deseados adicionales semejantes a los usados en la sección anterior e indicar si las ganancias usadas anteriormente brindan el mismo tipo de respuestas para cada caso. De igual manera, indicar las semejanzas/diferencias con los resultados obtenidos en la sección anterior.

6. (1 pt) Indicar cómo realizaría control de un punto deseado en el espacio Cartesiano.

5. Conclusiones

(1 pt) Elaborar algunas conclusiones sobre el laboratorio desarrollado comparando principalmente los resultados obtenidos con los diversos métodos utilizados.