Assignment 4
Inverse Dynamics
Due Oct 19

In this assignment, you will implement an inverse dynamics controller with redundancy resolution. Your will plan the end-effector positions for the robot using from potential field. The final output is the joint torques of the robot.

1 Steps

1.1 Pull the latest packages for Kortex

Once you pull the lastest packages, you should be able to run
`> roslaunch kortex_gazebo gen3_dynamics.launch`

1.2 Read the URDF filename with ROS parameter

Use ROS parameter to read the URDF filename, which is stored in `/gen3/urdf_file_name`.

1.3 Potential Field Planner

Plan your trajectory for the robot end-effector positions using potential field. You can ignore the orientations in this assignment.

- Write a ROS service so the user can enter the target position $\dot{\mathbf{x}}^{tar}$
- Desired task-space velocity: choose your own $\mathbf{K}_{attr}$

$$\dot{\mathbf{x}}^{ref} \in \mathbb{R}^3 = \mathbf{K}_{attr}(\mathbf{x}^{tar} - \mathbf{x}^{fbk})$$

  Scale this vector down if the norm exceeds the velocity limit
- Desired task-space position

$$\mathbf{x}^{ref} \in \mathbb{R}^3 = \mathbf{x}^{fbk} + \dot{\mathbf{x}}^{ref} \triangle t$$

- Desired task-space accelerations

$$\ddot{\mathbf{x}}^{ref} \in \mathbb{R}^3 = \frac{\dot{\mathbf{x}}^{ref} - \dot{\mathbf{x}}^{fbk}}{\triangle t}, \text{ or set it to } 0$$

1.4 Task-space Controller

- Step 1: Read the reference task-space motion $\dot{\mathbf{x}}^{ref}, \dot{\mathbf{x}}^{ref}, \ddot{\mathbf{x}}^{ref}$ from potential field
- Step 2: Compute the feedback task-space positions and velocities
  - Get the joint feedback by subscribing to the topic `/gen3/joint_states` (same as A3)
  - Compute the task-space position $\mathbf{x}^{fbk}$ (same as A3)
  - Compute the task-space velocity $\dot{\mathbf{x}}^{fbk} = \mathbf{J}\dot{\mathbf{q}}^{fbk}$
- Step 3: Compute $\mathbf{J}, \dot{\mathbf{J}}, \mathbf{M}, \mathbf{h}$ from Pinocchio. You can find examples in `test/test-pinocchio.cpp` in `robot_model` package. We will control the translation only and leave the orientation. The dimensionalty of those quantities should be

$$\mathbf{J}^{tra} \in \mathbb{R}^{3\times 7}, \dot{\mathbf{J}}^{tra} \in \mathbb{R}^{3\times 7}, \mathbf{M} \in \mathbb{R}^{7\times 7}, \mathbf{h} \in \mathbb{R}^{7\times 1}$$

- Step 4: Implement the task-space inverse dynamics controller

$$
\begin{aligned}
\ddot{\mathbf{x}}^{cmd} &= \ddot{\mathbf{x}}^{ref} + \mathbf{D}_{task}(\dot{\mathbf{x}}^{ref} - \dot{\mathbf{x}}^{fbk}) + \mathbf{K}_{task}(\mathbf{x}^{ref} - \mathbf{x}^{fbk}) && \in \mathbb{R}^{3\times 1} \\
\mathbf{\Lambda} &= (\mathbf{J}^\top)^\dagger \mathbf{M}\mathbf{J}^\dagger && \in \mathbb{R}^{3\times 3} \\
\boldsymbol{\eta} &= (\mathbf{J}^\top)^\dagger \mathbf{h} - \mathbf{\Lambda}\dot{\mathbf{J}}\dot{\mathbf{q}} && \in \mathbb{R}^{3\times 1} \\
\boldsymbol{\mathcal{F}}^{cmd} &= \mathbf{\Lambda}\ddot{\mathbf{x}}^{cmd} + \boldsymbol{\eta} && \in \mathbb{R}^{3\times 1} \\
\boldsymbol{\tau}_{task} &= \mathbf{J}^\top \boldsymbol{\mathcal{F}}^{cmd} && \in \mathbb{R}^{7\times 1}
\end{aligned}
$$

- Step 5: Tune the stiffness $\mathbf{K}_{task}$ and damping $\mathbf{D}_{task}$
  - start with small $\mathbf{K}_{task}$ slowly increase it
  - visually check the effects on the robot

1.5 Null-space Controller

- Step 1: Get the feedback joint-space positions and velocities $\mathbf{q}^{fbk}, \dot{\mathbf{q}}^{fbk}$
- Step 2: Choose a null-space target for the first 7 joints. Here is one that works, but may not be the best one.

$$\mathbf{q}^{tar} = [0.0, 0.0, 0.0, -1.57, 0.0, 0.75, 1.57]$$

- Step 3: Get the reference joint-space motion $\mathbf{q}^{ref}, \dot{\mathbf{q}}^{ref}, \ddot{\mathbf{q}}^{ref}$, which can be another potential field planner

$$
\begin{aligned}
\dot{\mathbf{q}}^{ref} \in \mathbb{R}^7 &= \mathbf{K}_{attr}(\mathbf{q}^{tar} - \mathbf{q}^{fbk}) \\
\mathbf{q}^{ref} \in \mathbb{R}^7 &= \mathbf{q}^{fbk} + \dot{\mathbf{q}}^{ref} \triangle t \\
\ddot{\mathbf{q}}^{ref} \in \mathbb{R}^7 &= \frac{\dot{\mathbf{q}}^{ref} - \dot{\mathbf{q}}^{fbk})}{\triangle t} \text{ or set it to 0}
\end{aligned}
$$

- Step 4: Get $\mathbf{M}, \mathbf{h}$ from Pinocchio
- Step 5: Implement Joint-space Inverse Dynamics Controller

$$
\begin{aligned}
\ddot{\mathbf{q}}^{cmd} &= \ddot{\mathbf{q}}^{ref} + \mathbf{D}_{joint}(\dot{\mathbf{q}}^{ref} - \dot{\mathbf{q}}^{fbk}) + \mathbf{K}_{joint}(\mathbf{q}^{ref} - \mathbf{q}^{fbk}) \\
\boldsymbol{\tau}_{joint} &= \mathbf{M}\ddot{\mathbf{q}}^{cmd} + \mathbf{h}
\end{aligned}
$$

- Step 6: Tune the stiffness $\mathbf{K}_{joint}$ and damping $\mathbf{D}_{joint}$

1.6 Putting together

- Putting together

$$
\begin{aligned}
\mathbf{P} &= \mathbf{I} - \mathbf{J}^\top(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^\top)^{-1}\mathbf{J}\mathbf{M}^{-1} \\
\boldsymbol{\tau}_{null} &= \mathbf{P}\boldsymbol{\tau}_{joint} \\
\boldsymbol{\tau}_{total} &= \boldsymbol{\tau}_{task} + \boldsymbol{\tau}_{null}
\end{aligned}
$$

- Publish the first 7 joints to `/gen3/joint_group_effort_controller/command`

2 Useful Tips

For implementations

- On the ROS side, start by designing your packages, publishers, subscribers, before implmentations
- Use classes whenever possible

- Compile often
- Avoid hard-coding anything and use ROS parameters instead

For A4

- Start by modifying your working code, (e.g., test potential field with your inverse kinematic controller)
- Joint space controller is easier to start
- Use ROS services or ROS parameters for tuning **K** and **D**

3 Evaluation

3.1 How to submit

Push your change to `robotic-coursework-f2022`. Do not create a new repository.

3.2 Test

We will test your implementations as follow:

1. Open a terminal and run
   `> roslaunch kortex_gazebo gen3_dynamics.launch`
2. Open another terminal and run
   `> roslaunch inverse_dynamics_controller a4.launch`
3. Open another terminal and run
   `> rosservice call /planner/move_to 0.5 0.0 0.3`
   The end-effector should move to position (0.5 0.0 0.3). We will try a few different combinations of inputs.

3.3 Marking Scheme (10 points total)

(2 points) Use ROS parameters to read the URDF file name
(2 points) Use potential field to plan the end-effector trajectory
(2 points) Use task-space inverse dynamics controller and the hand reaches the target
(2 points) Use joint-space inverse dynamics controller for redundancy resolution
(2 points) End-effector reaches the target