UNIVERSITY COLLEGE LONDON (UCL)

COMP0127 – ROBOTIC SYSTEMS ENGINEERING

Coursework 3: Actuators, Mechanisms and Robot Dynamics

Department
Computer Science

Name
Wing Chung Law

Student Number
18067536

Submission Date
January 10, 2022

Contents

Actuators and Mechanisms

Question 1

1.a

If we assume the manipulator is able to pick up the candy from directly above, it will only require 3 degree of freedom in the Cartesian space, for XYZ translation to avoid redundancy. This is because the candies are quasi-spherical shaped, so it doesn't require an additional degree of freedom to change the orientation of the candies.

1.b

A parallel manipulator such as the Delta robot can be used as it has a high speed and accuracy. A vacuum gripper should be used to avoid breaking the candies. A vacuum gripper also doesn't require an additional actuator to rotate the orientation of the end effector.

1.c

As the candies are very small, high degree of accuracy and performance are required. Servomotors should be used as they have high accuracy with position and velocity feedback. Gear transmission should be used for high robustness.

1.d

Rotary encoders are used to measure the speed and position of the servo motor. A pressure sensor is used to measure the the pressure in the vacuum gripper. A camera can be used to determine the correct object.

1.e

The joints are prone to wear from repeated task execution, caused by constant friction and heat generation. The wear can be minimised by applying grease and lubricant to the joints. Wear could also be minimised by avoiding loading that exceeds the maximum capacity of the robot.

1.f

A camera can be used to identify the specified colour of the candies. One way to determine the weight of the candies is to measure the required pressure in the vacuum gripper, heavier candies require higher suction force to pick up. Hence, candies that can be picked up with a suction force below the minimum threshold are discarded. Another way to determine the weight of the candies is by measuring the inertia of the end effector when moving the candies around. Furthermore, active thermography can also be used, which shines a laser to the candy and examines the dissipation of thermal heat at the surface of the candy with a thermal camera. The thermal data is then fed to a machine learning model to estimate the volume and weight of the candy[1].

Robot Dynamics

Question 2

2.a

The `get_jacobian_centre_of_mass` function was edited to compute the Jacobian at the centre of mass for the iiwa14 manipulator. As the iiwa14 manipulator has 7 joints, an empty 6x7 matrix was created, where the Jacobian will have a form as shown below

$$J = \begin{bmatrix} J_{P_1} & J_{P_2} & J_{P_3} & J_{P_4} & J_{P_5} & J_{P_6} & J_{P_7} \\ J_{O_1} & J_{O_2} & J_{O_3} & J_{O_4} & J_{O_5} & J_{O_6} & J_{O_7} \end{bmatrix} \tag{1}$$

where J_{P_i} and J_{O_1} are 3×1 vectors given by

$$\begin{aligned} J_{P_j}^{(l_i)} &= \begin{cases} z_{j-1} & \text{if joint } j \text{ is prismatic} \\ z_{j-1} \times (p_{l_i} - p_{j-1}) & \text{if joint } j \text{ is revolute} \end{cases} \\ J_{o_j}^{(l_i)} &= \begin{cases} 0 & \text{if joint } j \text{ is prismatic} \\ z_{j-1} & \text{if joint } j \text{ is revolute} \end{cases} \end{aligned} \tag{2}$$

where z_{j-1} is the unit vector of axis z of frame $j-1$, p_{j-1} is the position vector of the origin of frame $j-1$, and p_{l_i} is the position vector of the centre of mass of the $i-th$ link.

We then created a for loop to compute the transformation matrix ${}^0T_i$ up to joint i with the `forward_kinematics` function and the centre of mass transformation matrix ${}^0T_{G_i}$ using the `forward_kinematics_centre_of_mass` up to joint $i+1$. p_{l_i} is then given as the first three elements of the fourth column of ${}^0T_{G_i}$, where z_{j-1} and p_{j-1} are given as the first three elements of the third and forth column of ${}^0T_i$, respectively. $J_{P_j}^{(l_i)}$ and $J_{o_j}^{(l_i)}$ are then calculated with (2) and are populated in the jacobian matrix.

2.b

The dynamic component $\boldsymbol{B}(\boldsymbol{q})$ for the iiwa14 manipulator can be calculated using the following equation

$$\boldsymbol{B}(\boldsymbol{q}) = \sum_{i=1}^{n} \left(m_{l_i} \boldsymbol{J}_P^{(l_i)^T} \boldsymbol{J}_P^{(l_i)} + \boldsymbol{J}_O^{(l_i)^T} \mathcal{J}_{l_i} \boldsymbol{J}_O^{(l_i)} \right) = [b_{ij}]_{n \times n} \tag{3}$$

where m_{l_i} is the mass of link i, J_P and J_O are the Jacobian components at the centre of mass, and $\mathcal{J}_{l_i}$ is the moment of inertia of link i in the base frame, given by

$$\mathcal{J}_{l_i} = {}^0R_{G_i} I_{l_i}^{O_i i \|} {}^0R_{G_i}^T \tag{4}$$

where ${}^0R_{G_i}$ is the rotation matrix from the frame G_i to the base frame and $I_{l_i}^{O_i i \|}$ is the moment of inertia of link i.

The `get_B` method is edited to compute $\boldsymbol{B}(\boldsymbol{q})$, which is a 7×7 numpy matrix. A for loop is created to sum up all the components for the 7 links. As the moment of inertia provided in the `Iiwa14DynamicBase` class is defined at the centre of mass

of each link (i.e. $I_{l_i}^{G_i}$), we used the Parallel-Axis theorem and the $x_{Gi}^{i-1}, y_{Gi}^{i-1}, z_{Gi}^{i-1}$ provided to convert it to the parallel inertia frame in joint i (i.e. $I_{l_i}^{O_i i\|}$). The moment of inertial in the base frame $\mathcal{J}_{l_i}$ is then computed using (4), where the rotation matrix is obtained from the rotation component of the transformation matrix ${}^0T_{G_i}$ by calling the `forward_kinematics_centre_of_mass` function. J_P and J_O for each link are obtained from the Jacobian centre of mass and are used to compute $\boldsymbol{B}(\boldsymbol{q})$ for each link with (3)

2.c

The dynamic component $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the product of the Centrifugal and Coriolis component $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and the joint velocity $\dot{\boldsymbol{q}}$, where $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is a 7×7 matrix given by

$$c_{ij} = \sum_{k=1}^{n} h_{ijk}\dot{q}_k \tag{5}$$

where h_{ijk} is the Christoffel symbols, that is a $7\times 7\times 7$ matrix given by

$$h_{ijk} = \frac{\partial b_{ij}(\boldsymbol{q})}{\partial q_k} - \frac{1}{2}\frac{\partial b_{jk}(\boldsymbol{q})}{\partial q_i} \tag{6}$$

To calculate the Christoffel symbols, three for loops were used to calculate the i, j and k components, in order to compute the partial derivatives of the dynamic component $\boldsymbol{B}$. However, as there is no function to calculate the partial derivatives, it can be approximated using the equation below

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h} \tag{7}$$

where h is a very small number, here we used $1\times 10^{-}8$. We then multiplied the Christoffel symbols with the joint velocities of the k^{th} joint to obtain c_{ij}. The final 7×7 $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ matrix is then multiplied by the 7×1 joint velocity matrix, to output a 7×1 matrix.

2.d

The dynamic component $\boldsymbol{g}(\boldsymbol{q})$ for the iiwa14 manipulator is a 7×1 matrix which can be calculated with the expression below

$$g_n = \frac{\partial P(\boldsymbol{q})}{\partial qn} \tag{8}$$

where $P(\boldsymbol{q})$ is the potential energy given by

$$\mathcal{P}(\boldsymbol{q}) = -\sum_{i=1}^{n} m_{l_i}\boldsymbol{g}_0^T\boldsymbol{p}_{l_i} \tag{9}$$

where g_0^T is the gravitational acceleration, m_{l_i} is mass of link i and p_{l_i} is the position vector of the centre of mass of link i. To calculate $\boldsymbol{g}$, we first defined a 7×1 array, and calculated the potential energy in a for loop, where p_{l_i} is given by the first three elements of the fourth column of ${}^0T_{G_i}$. As the $\boldsymbol{g}$ is the partial derivative of the potential energy, we used the same method in (7) to approximate the value, where we also used $h = 1\times 10^{-}8$. A for loop is created to compute the $\boldsymbol{g}$ for each link, and a for loop within is used to compute the potential energy partial derivative.

Question 3

The Huygens-Steiner theorem is used to determine the moment of inertia or the rotational inertial of a rigid body about any axis, given the body's moment of inertia about a parallel axis that passes through the object's centre of gravity, and the perpendicular distance between the axes [2].

The theorem states that the moment of inertia about an axis parallel to and at a distance r from the axis which passes through the object's centre of mass is given by

$$I = I_o + mr^2 \tag{10}$$

where I_o is the moment of inertia of the body about an axis (i.e. A) through the body, and I is the moment of inertia of the body about another axis parallel to axis A, m is the mass of the body, and r is the perpendicular distance between the two axes.

To prove the theorem, we first assume a rigid body with mass m, where its centre of mass lies at the origin of the body and is rotated about the z axis. The moment of inertia relative to the z axis is given by

$$I_{\text{cm}} = \int \left(x^2 + y^2\right) dm \tag{11}$$

If the rigid body is now rotated about a new axis z', that is parallel to z and at a distance r away along the x axis, we can modify (12) to obtain the moment of inertia relative to the z' axis as

$$I = \int \left[(x-r)^2 + y^2\right] dm \tag{12}$$

expanding the brackets to give

$$I = \int \left(x^2 - 2xr + r^2 + y^2\right) dm \tag{13}$$

which can be further written as

$$I = \int \left(x^2 + y^2\right) dm + r^2 \int dm - 2r \int x\,dm \tag{14}$$

where the first term is same as (12), and $\int x\,dm$ of the third term is the x centre of mass $x_c m$ of the object. As the centre of mass is at the origin, $x_c m$ is 0, and the third term of (14) is 0. Hence the expression can be modified to

$$I = I_{cm} + mr^2 \tag{15}$$

The Huygens-Steiner theorem is important in robotics applications as the theorem can be generalised to calculations for the inertia tensor, which is very useful as sometimes the tensor of inertia about different axes are required. For example, the inertia tensor of each link of a manipulator is calculated in the frame at the centre of mass, which is usually around the middle of the link. However, this is not useful as the frames of a manipulator are defined at the joints on both ends of the links. Hence, the Huygens-Steiner theorem can be used to change the position of the reference frame from the centre of mass to the joint.

Question 4

Forward dynamics can be used to calculate the acceleration vector as well as the possible end-effector forces, when the joint torques are known, and is given by

$$\ddot{q} = B^{-1}(q)(\tau - C(q,\dot{q})\dot{q} - g(q)) \tag{16}$$

where τ is the joint torque, $\ddot{q}$, $\dot{q}$ and q are the joint acceleration, joint velocity and joint position, respectively. B is the inertial components, $C(q,\dot{q})$ is the Coriolis and centrifugal components, and $g(q)$ is the gravitational components. Forward dynamics is useful in robot motion simulations and are also used in musculoskeletal simulations, which can be used to study human locomotion [3]. However, modelling realistic human motion from internal forces with forward dynamics can be challenging due to a large number of controllable segments in the human body. For example, it is very difficult to predict how a change in moment of force in the hip joint might influence the ground reaction force and thus the acceleration of the centre of mass. Furthermore, predicting the movements that result from a change in moment of force in the rest of the body is very challenging, unless the model is heavily constrained [4].

Inverse dynamics is used to compute the torque function $\tau(q,\dot{q},t)$, given the joint accelerations, velocities and position vector, which is given by

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau(q,\dot{q},t) \tag{17}$$

Inverse dynamics is used in real systems for online control of robot motions and forces, as it can be used to calculate the torques required by the robot's motors to deliver a movement. Similar to forward dynamics, inverse dynamics is also used in the field of biomechanics, as muscle forces for a desired joint motion can be calculated. However, inverse dynamics simulation for human motion is also very challenging and is burdened with high inaccuracies, as the models often rely on multiple assumptions related to musculotendon paths, effective attachment points of the tendons, muscle activations etc. [5]. Another challenge faced by inverse dynamics is the external forces, such as ground contact forces, which affect the motions but are not directly observable from the kinematic motion.

Question 5

5.a

The bag file contains 1 message of type trajectory_msgs/JointTrajectory. The message contains the joint names of the iiwa14 robot, along with 3 sets of joint positions, velocities and accelerations, where the velocities and accelerations are 0. The message also contains "effort" and "time from start" for each joint position, which describes the torque of each joint and the time it takes for the robot to reach the described joint position, respectively. The messages from the bag file can be read using the `bag.read_messages` function, where the joint data and time from start are appended to the corresponding list.

5.b

As we are required to compute the joint accelerations $\ddot{q}$ of the robot, this is an inverse dynamics problem. As the joint position, velocity and torque can be obtained with a subscriber, the inertial components, Coriolis and centrifugal components and the Gravitational components can be found, which can be used to calculate the joint acceleration as given in (16).

5.c

To publish the trajectory to see the robot moving in simulation, we first defined a publisher with `rospy.Publisher`, which publishes the "JointTrajectory" of the robot. A `trajectory` function is then created, which takes the "positions" and the "time_from_start" from the bag file and defines the "JointTrajectoryPoint", the 3 trajectory points are then appended to the "points" of the "JointTrajectory" message through a for-loop. The timestamp and joint names of the message are also defined as the current time and the joint names from the bag file, respectively.

5.d

To calculate the joint accelerations throughout the trajectory, we subscribe to the "JointState" of the robot. We then define a `callback` function which is called by the subscriber when data is received. The "JointState" message contains the joint names, joint positions, joint velocities, as well as the joint effort. We can also calculate the joint velocities manually by calculating the difference between the current and previous joint position and dividing by the change in time.

To calculate the joint accelerations, the forward dynamics equation as defined in (16) was used. The joint torque to obtained from the joint effort of the "JointState" message. The dynamic component $\boldsymbol{B}$ is obtained by calling the `get_B` function from the `Iiwa14DynamicKDL` class by passing the current joint position. Similarly, $C(q,\dot{q})\dot{q}$ and $g(q$ are obtained by calling the `get_C_times-qdot` and `get_G` functions, respectively. From the dynamics components, the joint acceleration can be found using the `acceleration` function defined.

5.e

To plot the joint accelerations as a function of time, we first create an empty list for the joint acceleration and modify the `callback` function, which appends the calculated accelerations to the list. We also defined a shutdown condition that shuts down the ROS node when the final position has been reached. This is achieved by calculating the Euclidean norm between the current joint position and the final joint position from the bag file. As the robot takes a long time to reach the exact final position, we will shut down the node when the Euclidean norm goes below 0.0315. This also stops the callback function from recording any more joint acceleration data. By using `rospy.on_shutdown`, the `plot_acceleration` function is called once the node is shut down.

To plot the joint acceleration as a function of time, the joint acceleration for each joint is retrieved from the list and is plotted using `Matplotlib`. The resulting plot is shown

in Fig.1.

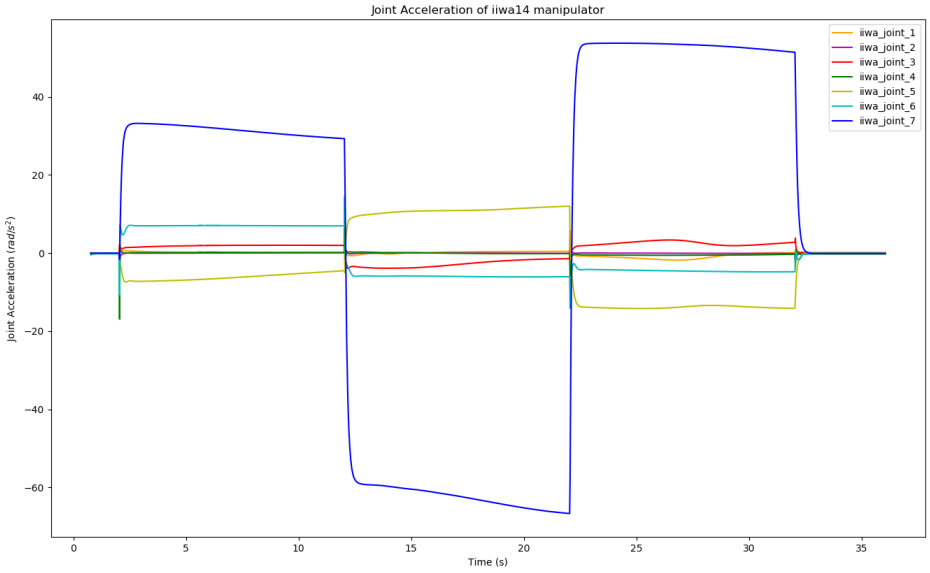

Figure .1: Joint acceleration over time

References

[1] T. Aujeszky, G. Korres, M. Eid, and F. Khorrami, "Estimating weight of unknown objects using active thermography," *Robotics*, vol. 8, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2218-6581/8/4/92

[2] A. R. Abdulghany, "Generalization of parallel axis theorem for rotational inertia," *American Journal of Physics*, vol. 85, no. 10, pp. 791–795, Oct. 2017. [Online]. Available: https://doi.org/10.1119/1.4994835

[3] S. J. Piazza, "Muscle-driven forward dynamic simulations for the study of normal and pathological gait," *Journal of NeuroEngineering and Rehabilitation*, vol. 3, no. 1, Mar. 2006. [Online]. Available: https://doi.org/10.1186/1743-0003-3-5

[4] E. Otten, "Inverse and forward dynamics: models of multi–body systems," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1437, pp. 1493–1500, Aug. 2003. [Online]. Available: https://doi.org/10.1098/rstb.2003.1354

[5] K. Dziewiecki, W. Blajer, Z. Mazur, and A. Czaplicki, "Modeling and computational issues in the inverse dynamics simulation of triple jump," *Multibody System Dynamics*, vol. 32, no. 3, pp. 299–316, Jul. 2013. [Online]. Available: https://doi.org/10.1007/s11044-013-9375-6