Chapter 2
Robot Kinematics and Dynamics Modeling

Abstract The robotic kinematics is essential for describing an end-effector's position, orientation as well as motion of all the joints, while dynamics modeling is crucial for analyzing and synthesizing the dynamic behavior of robot. In this chapter, the kinematics and dynamics modeling procedures of the Baxter robot are investigated thoroughly. The robotic kinematics is briefly reviewed by highlighting its basic role in analyzing the motion of robot. By extracting the parameters from an URDF file, the kinematics model of the Baxter robot is built. Two experiments are performed to verify that the kinematics model matches the real robot. Next, the dynamics of robot is briefly introduced by highlighting its role in establishing the relation between the joint actuator torques and the resulting motion. The method for derivation of the Lagrange–Euler dynamics of the Baxter manipulator is presented, followed by experimental verification using data collected from the physical robot. The results show that the derived dynamics model is a good match to the real dynamics, with small errors in three different end-effector trajectories.

2.1 Kinematics Modeling of the Baxter® Robot

2.1.1 Introduction of Kinematics

The robotic kinematics studies the motion of a robot mechanism regardless of forces and torque that cause it. It allows to compute the position and orientation of robot manipulator's end-effector relative to the base of the manipulator as a function of the joint variables. Robotic kinematics is fundamental for designing and controlling a robot system. In order to deal with the complex geometry of a robot manipulator, the properly chosen coordinate frames are fixed to various parts of the mechanism and then we can formulate the relationships between these frames. The manipulator kinematics mainly studies how the locations of these frames change as the robot joints move.

Kinematics focuses on position, velocity, acceleration, and an accurate kinematics model must be established in order to investigate the motion of a robot manipulator. Denavit–Hartenberg (DH) notations are widely used to describe the kinematic model

C. Yang et al., *Advanced Technologies in Modern Robotic Applications*,
DOI 10.1007/978-981-10-0830-6_2

of a robot. The DH formation for describing serial-link robot mechanism geometry has been established as a principal method for a roboticist [1]. Standard DH notations are used to create a kinematics model of robot. The fundamentals of serial-link robot kinematics and the DH notations are well explained in [2, 3].

In DH convention notation system, each link can be represented by two parameters, namely the link length a_i and the link twist angle α_i. The link twist angle α_i indicates the axis twist angle of two adjacent joints i and $i-1$. Joints are also described by two parameters, namely the link offset d_i, which indicates the distance from a link to next link along the axis of joint i, and the joint revolute angle θ_i, which is the rotation of one link with respect to the next about the joint axis [4]. Usually, three of these four parameters are fixed while one variable is called joint variable. For a revolute joint, the joint variable is parameter θ_i, while for a prismatic joint, it will be d_i. The DH convention is illustrated in Fig. 2.1.

The four parameters of each link can be specified using DH notation method. With these parameters, link homogeneous transform matrix which transforms link coordinate frame $i-1$ to frame i is given as Eq. (2.1).

$$
\begin{aligned}
{}^{i-1}A_i(\theta_i, d_i, a_i, \alpha_i) &= R_z(\theta_i)T_z(d_i)T_x(a_i)R_x(\alpha_i) \\
&= \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ c\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.1}
$$

where $\sin(\theta)$ is abbreviated as $s\theta$ and $\cos(\theta)$ as $c\theta$, R_j denotes rotation about axis j, and T_j denotes translation along axis j. For an n-link robot arm, the overall arm transform namely forward kinematics, in terms of the individual link transforms can

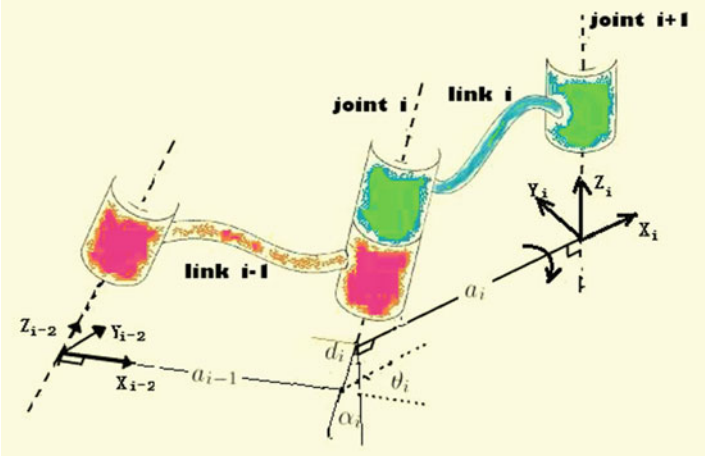


Fig. 2.1 Definition of standard DH link parameters

be expressed in Eq. (2.2). The Cartesian position of the end-effector can be calculated from Eq. (2.3).

$$^{0}A_{n} = {}^{0}A_{1}{}^{1}A_{2} \ldots {}^{n-1}A_{n} \tag{2.2}$$

$$^{n}X_{0} = {}^{0}A_{n}X_{n} \tag{2.3}$$

where $X = [x, y, z, 1]^{T}$ is an augmented vector of Cartesian coordinate.

Using the DH description of a robot and the Robotics Toolbox as introduced in Sect. 1.1, we can easily simulate the motion of a robot and calculate the kinematics, e.g., joints configuration for a certain pose and the Jacobian matrix [4].

2.1.2 Kinematics Modeling Procedure

Recalling Sect. 1.1, the Baxter robot consists of a torso based on a movable pedestal and two 7 DOF arms installed on left/right arm mounts, respectively. Next, we will build the kinematics model, the Baxter robot. The modeling procedure of kinematics of the Baxter® robot includes the following steps [5]. First, we perform analysis of mechanical structure of the Baxter® robot, and the main elements in URDF file and a 3D visual model will be described. Second, the method to obtain the DH parameters from a link of the left arm is presented. Finally, we will tabulate the DH parameters and create the kinematic model of dual arms with corresponding DH parameters. This completes the kinematic modeling of Baxter® robot.

2.1.2.1 Structural Analysis

Baxter® robot has two arms and a rotational head on its torso. The arm is shown in Fig. 2.2. The arm of Baxter robot comprises a set of bodies, called links, in a chain and connected by revolute joints. There are seven rotational joints on the arm, namely s0, s1, e0, e1, w0, w1, w2, respectively. Each joint has 1DOF. The *"arm mount"* assembly, is fixed on the *"torso"* assembly. Joint s0 is connected to the *"arm mount"* assembly. Between joint s0 and s1, there is *"upper shoulder"* assembly. *"lower shoulder"* assembly is located between joint s1 and joint e0, and so on. This structural connection is illustrated in Fig. 2.2.

2.1.2.2 URDF Description

URDF is a file in XML format which describes a robot, detailing its parts, joints, dimensions, and so on. It is a fundamental robot model file in ROS, which is a flexible framework for developing robot software. As introduced in Sect. 1.5, ROS is a collection of tools, libraries, and conventions that aims to simplify the task of creating

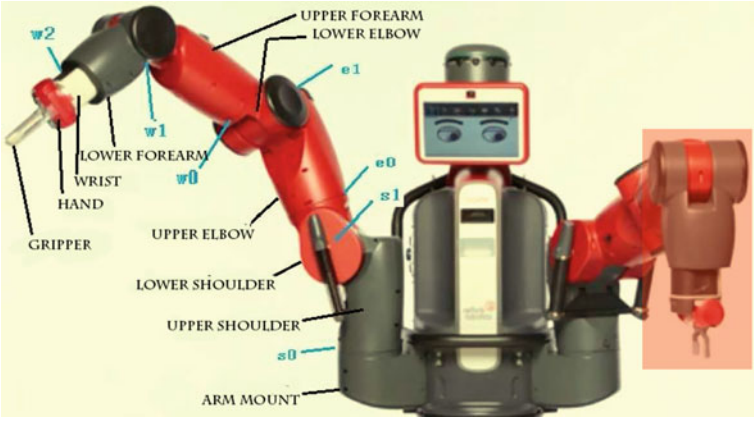


Fig. 2.2 The arms of Baxter robot, on which there are seven joints

complex and robust robot behavior across a wide variety of robotic platforms [6, 7]. The way ROS uses a 3D model of a robot or its parts to simulate them or to simply help the developers in their daily work is by the means of the URDF files. In ROS, a 3D robot such as the Baxter is always associated with a URDF file which describes the kinematic information of the robot.

The Baxter® robot URDF file consists of a series of frames and transform description between frames. A package about Baxter® robot mechanics construction can be found in [8]. This package includes all files that we need to analyze key factors of Baxter® robot, such as URDF file and other necessary files.

There are two major elements in an URDF file which describe the geometry of a robot: "link" and "joint." These two elements are not the same as joints and links normally used in DH notation system. In fact, a "link" in URDF represents a frame. The "link" code section means a frame predefined in the robot: it describes a rigid body with *inertia*, *visual* features, and so on. Below is an example of a link element named *left_upper_shoulder*:

```
<link name="left_upper_shoulder">
 <visual>
  <origin rpy = "0,0,0" xyz = "0,0,0"/></visual>
  <inertial>
  <origin rpy="0,0,0"
   xyz="0.01783,0.00086,0.19127"/>
  <mass value="5.70044"/>
  <inertia
   ixx="0.04709102262"...izz="0.03595988478"/>
 </inertial>
</link>
```

The *visual* property of the link specifies the shape of the body (such as box, cylinder) for visualization purposes. *Origin* section in *visual* indicates the reference frame of the visual element with respect to the reference frame of the link.

A "joint" in URDF represents a transform between two frames. It does not necessarily mean a motional (translational or rotational) joint. In fact, most of these "joints" are fixed. For example, the "joint" named *torso_s0* and *left_torso_arm_mount* are both fixed, as described above. Below is an example of a "joint" element.

```
<joint name="left_s0" type="revolute">
 <origin rpy="0,0,0" xyz="0.055695,0,0.011038"/>
 <axis xyz="0,0,1"/>
 <parent link="left_arm_mount"/>
 <child link="left_upper_shoulder"/>
 <limit lower="−1.7" upper="1.7" />
</joint>
```

The *type* property specifies the type of "joint": revolute, prismatic, or fixed, whereas the former two types are motional joint. The *origin* property is the transform from the parent link to the child link. The "joint" is located at the origin of the child link, "above. *rpy*" represents the rotation around the axis of the parent frame: first roll around x, then pitch around y and finally yaw around "z. *xyz*" represents the position offset from the "origin.*lower*" and "origin.*upper*" describe the minimum and maximum joint angles in radians. In Fig. 2.3, the first link has name *"base"* which is represented by a rectangle, also representing a coordinate frame named *"base"*. The ellipse represents a coordinate transformation between two frames, for example, through *torso_t0*, frame *base* can be translated to frame *torso*. In order to get the transform between frames, we will use the visual properties of link, and we can get a transform path of left arm, which is depicted in Fig. 2.3 in a bold line.

2.1.2.3 Kinematic Model

Now, we are ready to create a forward kinematics model of Baxter® robot. We will take left arm as an example, and create the left arm simulation model using DH notation.

From the frame of *base*, the "joint" *torso_s0* and *left_torso_arm_mount* are fixed, the first motional joint we can find is *left_s0*, so we consider the *left_s0* as the first joint and *left_upper_shoulder* as the first link, *left_s1* and *left_lower_shoulder* as the second joint and link, respectively. Let us continue in this manner until the seventh joint *left_w2*, the last rotational joint, and link *left_wrist*, are calculated.

The first four frame transformations can be found in URDF file (refer to Fig. 2.3). Here is the description of *left_s0*, *left_s1*, *left_e0*, and *left_e1* in URDF:

```
<joint name="left_s0" type="revolute">
 <origin rpy="0 0 0" xyz="0.056 0 0.011"/>
<joint name="left_s1" type="revolute">
 <origin rpy="−1.57 0 0" xyz="0.069 0 0.27"/>
<joint name="left_e0" type="revolute">
 <origin rpy="1.57 0 1.57" xyz="0.102 0 0"/>
<joint name="left_e1" type="revolute">
 <origin rpy="−1.57 −1.57 0"xyz="0.069 0 0.262"/>
```

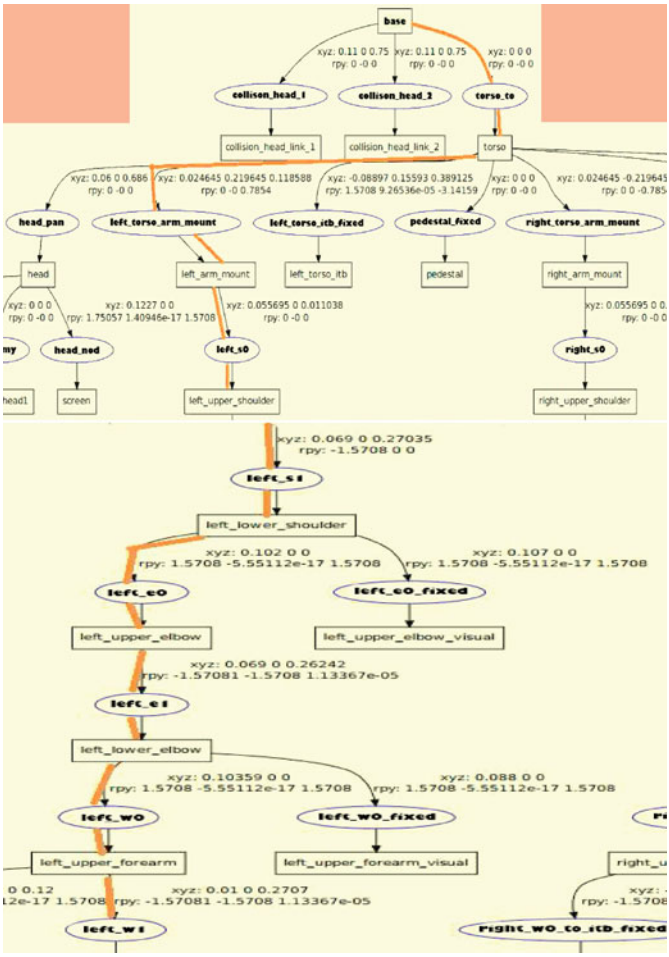


Fig. 2.3 Parts of frames and transform relationship between frames in Baxter® robot, generated from ROS instruction *$rosrun urdfdom urdf_to_graphiz baxter.urdf* [7]. Rectangle("link") represents a frame and ellipse("joint") represents a translation/rotation between frames

These frame transforms are shown in Fig. 2.4.

From *left_s0* to *left_s1*, the axis of joint rotates $-\frac{\pi}{2}$ along X_0 axis (all revolution abides by right-hand law), so link twist $\alpha_1 = -\frac{\pi}{2}$ .The translation of *left_s1* in URDF file is $x = 0.069\,\text{m}$ and $z = 0.27\,\text{m}$, so $d_1 = 0.27\,\text{m}$ and $a_1 = 0.069\,\text{m}$, let $\theta_1 = 0$ just because this is a revolute joint.

From *left_s1* to *left_e0*, the axis of joint rotates $\frac{\pi}{2}$ along axis Z_1 first and then rotates $\frac{\pi}{2}$ along axis X_1. Therefore, we should let $\theta_2 = \frac{\pi}{2}$ and let link twist $\alpha_2 = \frac{\pi}{2}$. Although the translation of *left_e0* is $x = 0.102\,\text{m}$, we have to assign $d_2 = 0\,\text{m}$ and

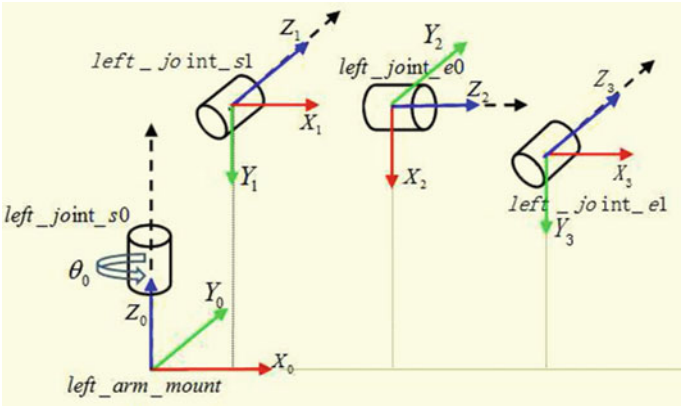


Fig. 2.4 First three joint frame transformational relationship

$a_2 = 0\,\text{m}$ because the $x = 0.102\,\text{m}$ is not the offset along the axis of joint *left_s1*. It is the key to make rotation occur only at the end of a link in this step.

From *left_e0* to *left_e1*, the axis of joint rotates $-\frac{\pi}{2}$ along X_2 axis, so link twist $\alpha_3 = -\frac{\pi}{2}$. The translation of *left_e0* is $x = 0.069\,\text{m}$ and $z = 0.262\,\text{m}$, so $a_3 = 0.069\,\text{m}$. The offset along the revolution axis should be $d_3 = 0.262\,\text{m} + 0.102\,\text{m} = 0.364\,\text{m}$, including the $x = 0.102\,\text{m}$ of joint *left_s1*. Repeating these steps, DH parameter table can be developed as shown in Table 2.1.

From DH notation table, we can get the transform matrix from the frame of *left_arm_mount* to the frame of *left_gripper* (end point). The model of the right arm of Baxter® robot can be created by using the same procedures above. DH notation table of right arm is the same as the left arm except $d_7 = 0.275\,\text{m}$, because in our configuration the gripper on right arm is an electric gripper, and a vacuum cup is installed on the left arm. According to different types of gripper mounted on each arm, the value of d_i would be different. In fact, each Baxter® robot has a unique URDF parameter configuration. Before creating the simulation model, the actual

Table 2.1 DH notation table of the left arm

Link i	θ_i (deg)	d_i (m)	a_i (m)	α_i (rad)
1	q_1	0.27	0.069	$-\frac{\pi}{2}$
2	$q_2 + \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
3	q_3	$0.102 + 0.262$	0.069	$-\frac{\pi}{2}$
4	q_4	0	0	$\frac{\pi}{2}$
5	q_5	$0.104 + 0.271$	0.01	$-\frac{\pi}{2}$
6	q_6	0	0	$\frac{\pi}{2}$
7	q_7	0.28	0	0

parameters should be obtained from frame transformation topic in ROS, rather than using the nominal ideal URDF file from the website.

The transform from frame of *base* to *left_arm_mount* given in Eq. (2.4).

$$^{base}T_{left\_arm\_mount} = R_z\left(\frac{\pi}{4}\right) T_x(0.056\,\text{m}) T_z(0.011\,\text{m}) \tag{2.4}$$

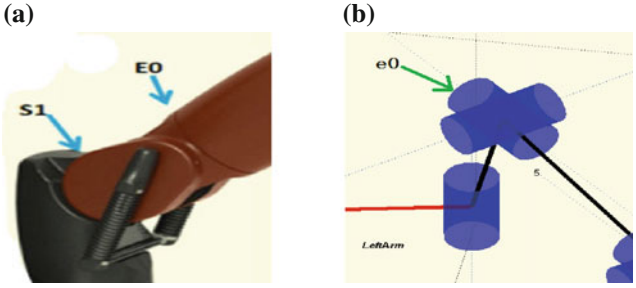
Then, a complete kinematics model of Baxter® robot can be created by using MATLAB commands developed in MATLAB Robotics Toolbox.

In general, the key steps to extract DH parameters from URDF file are summarized below

(i) Figure 2.3 describes the transform relationship of joints and links in the URDF file and can be considered as an inverted tree. To identify DH parameters for Baxter robot, it is necessary to find the first rotational joint ($rpy \neq 0$) from the root of this tree. Mark it as joint 1, and recognize four elements θ_1, d_1, a_1, α_1 and fill them in the DH table. Denote the next link as link 1, and assign properties includes *mass*, *inertia*, and *origin* to the model. Then continue the above procedure for joint 2, joint 3, and so on.

(ii) When the frame rotates not only about X axis (r in *rpy*), but also Z axis(y in *rpy*), an additional rotation ($\frac{\pi}{2}/-\frac{\pi}{2}$) needs to be added to the joint revolution angle θ_i.

(iii) The location of a joint seen on the model built in MATLAB Robotics Toolbox may be different from its location on the physical robot. For example, Baxter robot's joint e0 is located toward the middle of the upper arm link as shown in Fig. 2.5a, while on the model developed by the MATLAB Robotics Toolbox, it is located in such a manner that it shares origin with the previous joint s1, as shown in Fig. 2.5b. This is because that the origin of the coordinate frame associated with joint e0 is same as the origin of coordinate frame associated with joint s1, according to DH convention. Consequently, in the MATLAB Robotics Toolbox model the link offset should be modified.



Joint e0 is in the middle of upper arm in actual robot

Joint e0 is at the end of the upper arm

Fig. 2.5 Joint rotates can only be represented at the end of link in model

2.1.3 Experimental Tests on Kinematics Modeling

Two experimental tests have been carried out to demonstrate the effectiveness of the kinematics model of the Baxter® robot. The first experiment shows that robot postures generated by the kinematic model are same as the real robot. The second experiment demonstrates the accuracy of the kinematic model.

2.1.3.1 Experiment Test 1

Given a Baxter robot manipulator, if all the seven joint angels $(\theta_1, \theta_2, \ldots \theta_7)$ are known, then we can calculate the Cartesian position and orientation of the robot. To test the accuracy of the kinematic model, we create the Baxter® robot arm that are put to several postures, retrieving the joint angular value $(\theta_1, \theta_2, \ldots \theta_7)$ and the position and orientation in Cartesian space (x_r, y_r, z_r). Then, input joint angles to our mathematical model and calculate Cartesian position (x_m, y_m, z_m) using forward kinematics. The postures of the robot and generated by model are shown in Fig. 2.6. It can be seen that when same joint angles are given, the model has the same posture as the robot.

2.1.3.2 Experiment Test 2

The pose that Baxter is shipping in or in power-off state is referred to as "shipping pose" [9] (Fig. 2.6d). Baxter's arms should be un-tucked (Fig. 2.6b) before subsequent movements. During the period when robot's arms move from tucked pose to un-tucked pose, and when it moves back to tucked pose, we retrieved a series of joint angles and Cartesian positions (x_r, y_r, z_r) and orientations $(\eta_{Rr}, \eta_{Pr}, \eta_{Yr})$, and calculate our Cartesian position (x_m, y_m, z_m) and orientations $(\eta_{Rm}, \eta_{Pm}, \eta_{Ym})$ using Eqs. (2.1)–(2.4) based on the model built. The deviation between them can be calculated from Eq. (2.5).

$$\begin{aligned} \Delta X &= (\Delta x, \Delta y, \Delta z) = (x_m, y_m, z_m) - (x_r, y_r, z_r) \\ \Delta \eta &= (\Delta \eta_R, \Delta \eta_P, \Delta \eta_Y) = (\eta_{Rm}, \eta_{Pm}, \eta_{Ym}) - (\eta_{Rr}, \eta_{Pr}, \eta_{Yr}) \end{aligned} \tag{2.5}$$

Figure 2.7 shows the position and orientation of left end point during the period of arm is tucking and untucking. Position and orientation of the end point of model (red curve) follows the trajectory of the actual robot (light gray curve) very closely.

The maximum error (unit of *x/y/z* is mm and unit of roll/pitch/yaw is rad) of position between our model and the real robot is sufficiently small and the maximum of errors (absolute values) for right arm is listed as below

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{max} = [4.2, 3.3, 6.8, 5.6, 7.6, 7.3]$$

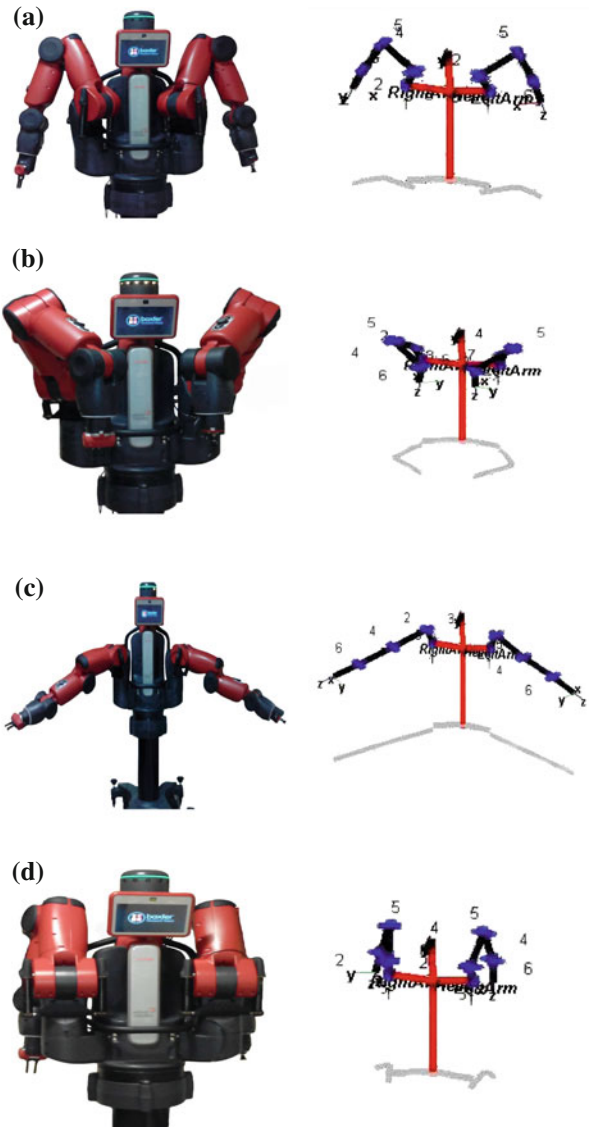


Fig. 2.6 Comparison of the posture of robot generated from model when same joint angles are given. The *'err'* is $(e_x, e_y, e_z, e_R, e_P, e_Y)$. They are calculated from Eq. (2.5), **a** [ready, $err = (0.6, 0.4, -2.4, 2.8, 0.6, 1.3)$], **b** [untacked, $err = (1.2, -0.8, -3.1, -1.0, 1.6, 1.8)$], **c** [extended, $err = (0.6, 0.7, -3.9, -2.8, 0.2, -1.8)$], **d** [tacked, $err = (0.5, 0.4, -1.6, -0.9, 2.3, 0.2)$]

for left arm is

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{max} = [4.4, 3.8, 6.1, 5.1, 7.2, 7.6]$$

The average error calculated from the data for right arm is

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{avg} = [0.7, -0.6, -3.8, 2.9, -3.2, -1.2]$$

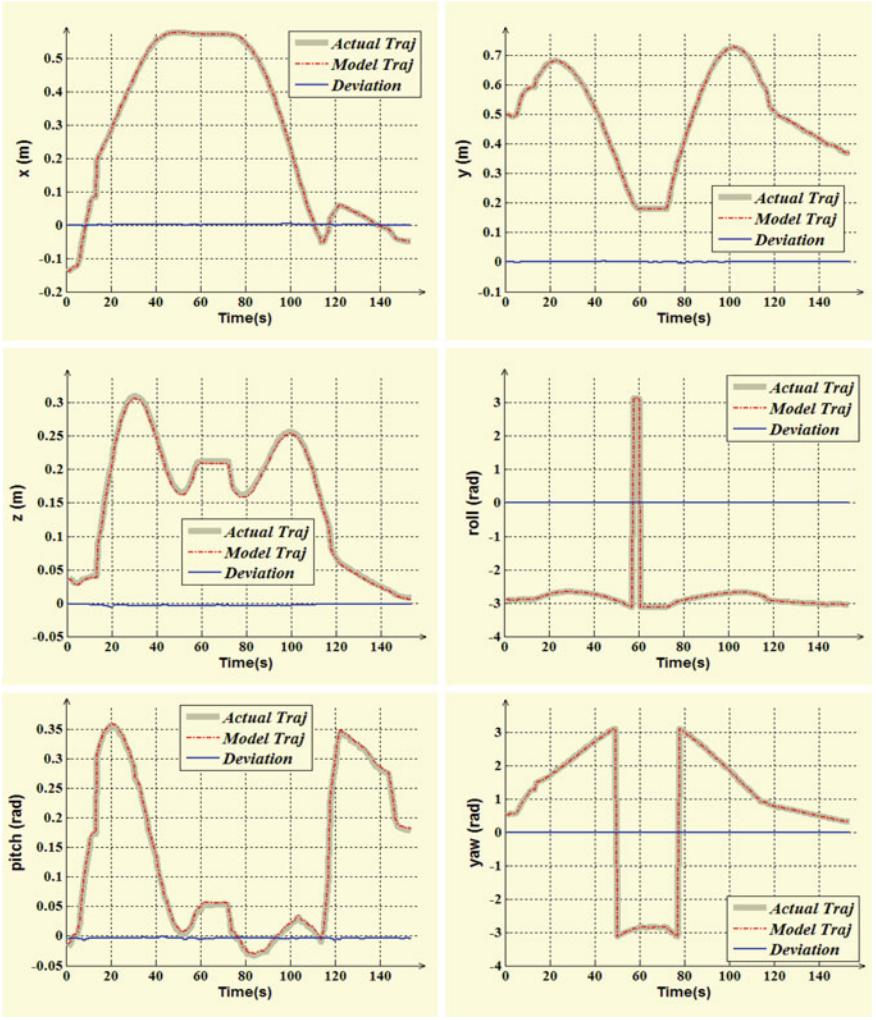


Fig. 2.7 Comparison of the trajectories of robot and those generated from model during the robot's left arm is tucked/un-tucked

for left arm is

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{avg} = [1.1, 0.6, -2.8, -1.6, -3.5, 1.1]$$

Thus, the accuracy of the kinematic model is satisfactory for simulation purpose.

2.2 Lagrange–Euler Dynamics Modeling of the Baxter Robot

2.2.1 Introduction of Dynamics

The dynamic model of a robot studies the relation between the joint actuator torques and the resulting motion. An accurate dynamics model of a robot manipulator is useful in many ways: for the design of motion control systems, analysis of mechanical design, simulation of manipulator motion, etc. Many control algorithms, such as computed torque control [10], predictive control [11] and sliding mode control [12] normally require an accurate model of the manipulator dynamics, commonly in the Lagrangian form:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau \tag{2.6}$$

where q denotes the vector of joint angles; $M(q) \in \Re^{n \times n}$ is the symmetric, bounded, positive definite inertia matrix, and n is the degree of freedom (DoF) of the robot arm; $C(q, \dot{q})\dot{q} \in \Re^n$ denotes the Coriolis and Centrifugal force; $G(q) \in \Re^n$ is the gravitational force, and $\tau \in \Re^n$ is the vector of actuator torques. In this form, the kinetic energy of the manipulator is described within $M(q)\ddot{q} + C(q, \dot{q})$, and the potential energy represented in the gravity term $G(q)$. This can then be used to calculate either the forward dynamics (useful for simulation), where the manipulator motion is calculated based on a vector of applied torques, or the inverse dynamics (useful for control design) where the torques for a given set of joint parameters can be calculated.

There are two commonly used methods for formulating the dynamics in Eq. (2.6), based on the specific geometric and inertial parameters of the robot: the Lagrange–Euler (L–E) formulation and the Recursive Newton–Euler (RN–E) method [13]. Both are equivalent, as both describe the dynamic behavior of the robot motion, but are specifically useful for different purposes.

The L–E method is based on simple and systematic methods to calculate the kinetic and potential energies of a rigid body system. The works of Bajeczy [14, 15], show that the equations of dynamic motion for a robot manipulator are highly nonlinear, consisting of inertial and gravity terms, and are dependent on the link physical parameters and configuration (i.e., position, angular velocity and acceleration). This provides the closed form of the robot dynamics, and is therefore applicable to the analytical computation of robot dynamics [16], and therefore can be used to design joint-space (or task-space, using transformation via the Jacobian) control strategies.

The L–E formulation may also be used for forward and inverse dynamic calculation, but this requires the calculation of a large number of coefficients in $M(q)$ and $C(q, \dot{q})$ from Eq. (2.6), which may take a long time. This makes this method somewhat unsuitable for online dynamic calculations, especially as other methods such as RN–E (described in the next section), or Lee's Generalized d'Alembert Equations (GAE) [17] produce more simple, albeit messy, derivations which are much faster [13]. A recursive L–E method has also been described [18] which greatly reduces

the computational cost of the L–E formulation and brings it into line with RN–E methods.

The N-E formulation is based on a balance of all the forces acting on the generic link of the manipulator; this forms a set of equations with a recursive solution [19], and was developed. A forward recursion propagates link velocities and accelerations, then a backward recursion propagates the forces and torques along the manipulator chain. This is developed as a more efficient method than L–E, and is based on the principle of the manipulator being a serial chain; when a force is applied to one link, it may also produce motion in connected links. Due to this effect, there may be considerable duplication of calculation [20], which can be avoided if expressed in a recursive form. This reduction in computational load greatly reduces calculation time, allowing the forward and inverse dynamics calculations to be performed in real time; therefore, it can enable real-time torque control methods of robot manipulators.

2.2.2 *Dynamics Modeling Procedure*

To establish the dynamics model of a robot, as stated above, two common methods have been developed: one is the Lagrange formulation, which gives a closed form of the dynamic equations for ease of theoretical analysis; and the other is the Newton–Euler method, which uses a recursive form with the easy-to-compute merit. In this section, a formulation of the Lagrange–Euler (L–E) equations representing the dynamics of the Baxter manipulator is presented for analysis the dynamics parameters in controller design.

Next, we will show the dynamics modeling of the Baxter robot which is introduced in Sect. 1.1. To accomplish the dynamics modeling of the Baxter Robot, some parameters are specified at first. The common inertial and geometrical parameters of the Baxter robot are shown in Table 2.2.

Table 2.2 Nomenclature

n	Degrees of freedom (DoF) of the manipulator
$q, \dot{q}, \ddot{q} \in \Re^{n\times 1}$	Vector of joint position, angular velocity and acceleration, respectively
a, d, α, θ	Variables denoting the Denavit–Hartenberg parameters
$I_i \in \Re^{3\times 3}$	Inertia tensor of link i
m	Link mass
$\bar{r}_i \in \Re^{4\times 1}$	Center of mass of link i
${}^{i}T_j \in \Re^{4\times 4}$	Homogeneous transform from link i to j

2.2.2.1 Parameters Specification

The D–H parameters and link masses of the Baxter manipulator are given in Table 2.3 and are derived from the Universal Robot Descriptor File (URDF) [21]. These parameters describe the configuration of the links, and form the basis of the Lagrange–Euler formulation. The homogeneous link transform matrices are formed from the D–H parameters as such below:

$$
{}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}
$$

where ${}^0T_i = {}^0T_1\ {}^1T_2 \ldots {}^{i-1}T_i$.

The center of mass (CoM) for each link is given in Table 2.4, which forms the homogeneous column vector $\bar{r}_i = [\bar{x}_i\ \bar{y}_i\ \bar{z}_i\ 1]^T$. The inertia tensors of each joint are given in Table 2.5, represented by the inertias working in each axis I_{xx}, I_{yy}, I_{zz} and cross-talk inertia between axes I_{xy}, I_{yz}, I_{xz}. Here, it is represented as a row vector, but is also commonly found in $I^{3\times3}$ symmetric matrix form. These information were obtained from https://github.com/RethinkRobotics

Table 2.3 D–H parameters of the Baxter robot

Link	θ	d (m)	a (m)	α (rad)	m (kg)
1	θ_1	0.2703	0.069	$-\pi/2$	5.70044
2	θ_2	0	0	$\pi/2$	3.22698
3	θ_3	0.3644	0.069	$-\pi/2$	4.31272
4	θ_4	0	0	$\pi/2$	2.07206
5	θ_5	0.3743	0.01	$-\pi/2$	2.24665
6	θ_6	0	0	$\pi/2$	1.60979
7	θ_7	0.2295	0	0	0.54218

Table 2.4 Center of mass (all units in m)

Link	$\bar{x}$	$\bar{y}$	$\bar{z}$
1	−0.05117	0.07908	0.00086
2	0.00269	−0.00529	0.06845
3	−0.07176	0.08149	0.00132
4	0.00159	−0.01117	0.02618
5	−0.01168	0.13111	0.0046
6	0.00697	0.006	0.06048
7	0.005137	0.0009572	−0.06682

Table 2.5 Link inertia tensors (all units kg $\times$ m^2)

Link	I_{xx}	I_{yy}	I_{zz}
1	0.0470910226	0.035959884	0.0376697645
2	0.027885975	0.020787492	0.0117520941
3	0.0266173355	0.012480083	0.0284435520
4	0.0131822787	0.009268520	0.0071158268
5	0.0166774282	0.003746311	0.0167545726
6	0.0070053791	0.005527552	0.0038760715
7	0.0008162135	0.0008735012	0.0005494148
Link	I_{xy}	I_{yz}	I_{xz}
1	−0.0061487003	−0.0007808689	0.0001278755
2	−0.0001882199	0.0020767576	−0.00030096397
3	−0.0039218988	−0.001083893	0.0002927063
4	−0.0001966341	0.000745949	0.0003603617
5	−0.0001865762	0.0006473235	0.0001840370
6	0.0001534806	−0.0002111503	−0.0004438478
7	0.000128440	0.0001057726	0.00018969891

2.2.2.2 Lagrange–Euler Formulation

The Lagrange–Euler equations of motion for a conservative system [13] are given by

$$L = K - P, \qquad \tau = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \tag{2.8}$$

where K and P are the total kinetic and potential energies of the system, respectively, $q \in \Re^n$ is the generalized robot coordinates equivalent to θ in Table 2.3, and τ is the generalized torque at the robot joints [13]. The kinematic and potential energies are given by:

$$\begin{aligned} K &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i}\left[\mathrm{Tr}\left(U_{ij}\ J_i\ U_{ik}^T\right)\dot{q}_j\dot{q}_k\right] \\ P &= \sum_{i=1}^{n} -m_i\ \mathbf{g}\left({}^0T_i\ \bar{r}_i\right) \end{aligned} \tag{2.9}$$

which, when substituted into Eq. (2.8), gives the expression:

$$\tau_i = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q}$$

$$= \sum_{j=i}^{n} \sum_{k=1}^{j} \mathrm{Tr}(U_{jk} J_j U_{ji}^T) \ddot{q}_k - \sum_{j=i}^{n} m_j \mathbf{g} U_{ji} \bar{r}_j$$
$$+ \sum_{j=i}^{n} \sum_{k=1}^{j} \sum_{m=1}^{j} \mathrm{Tr}(U_{jkm} J_j U_{ji}^T) \dot{q}_k \dot{q}_m. \tag{2.10}$$

This can be expressed more simply in the form given in Eq. (2.6), as a sum of the inertia, Coriolis/centrifugal and gravity terms. The elements of the symmetric matrix $M(q)$ are given by

$$M_{i,k} = \sum_{j=\max(i,k)}^{n} \mathrm{Tr}(U_{jk}\ J_j\ U_{ji}^T) \quad i,k = 1,2,\ldots n, \tag{2.11}$$

the Coriolis/centrifugal force vector $C(q, \dot{q})$

$$C_i = \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm} \dot{q}_k \dot{q}_m$$
$$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} \mathrm{Tr}(U_{jkm}\ J_j\ U_{ji}^T) \tag{2.12}$$

and the gravity vector $G(q)$

$$G_i = \sum_{j=i}^{n} (-m_j \mathbf{g} U_{ji} \bar{r}_j) \tag{2.13}$$

where $\mathbf{g} = [0,\ 0,\ -9.81,\ 0]$ is the gravity row vector. The matrix U_{ij} is the rate of change of points on link i relative to the base as the joint position q_j changes

$$U_{ij} \equiv \frac{\partial T_i^0}{\partial q_j} = \begin{cases} {}^0T_{j-1}\ Q_j\ {}^{j-1}T_i & j \le i \\ 0 & j > i \end{cases} \tag{2.14}$$

which allows derivation of the interaction effects between joints, U_{ijk}

$$U_{ijk} \equiv \frac{\partial U_{ij}}{\partial q_k}$$
$$= \begin{cases} {}^0T_{j-1}\ Q_j\ {}^{j-1}T_{k-1}\ Q_k\ {}^{k-1}T_i & i \ge k \ge j \\ {}^0T_{k-1}\ Q_k\ {}^{k-1}T_{j-1}\ Q_j\ {}^{j-1}T_i & i \ge j \ge k \\ 0 & i < j \text{ or } i < k \end{cases} \tag{2.15}$$

where, for Baxter, as all joints are revolute,

$$Q_j = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.16}$$

The J_i matrices are independent of link position or motion, and therefore only needs to be calculated *once* from the inertia tensors, link masses and link CoMs:

$$J_i = \begin{bmatrix} \frac{-I_{xxi}+I_{yyi}+I_{zzi}}{2} & I_{xyi} & I_{xzi} & m_i\bar{x}_i \\ I_{xyi} & \frac{I_{xxi}-I_{yyi}+I_{zzi}}{2} & I_{yzi} & m_i\bar{y}_i \\ I_{xzi} & I_{yzi} & \frac{I_{xxi}+I_{yyi}-I_{zzi}}{2} & m_i\bar{z}_i \\ m_i\bar{x}_i & m_i\bar{y}_i & m_i\bar{z}_i & m_i \end{bmatrix} \tag{2.17}$$

This concludes the calculations required to form the L–E dynamics of the Baxter arm.

2.2.3 Experimental Studies

To collect data from the Baxter robot, a PID position controller is employed in a dual loop configuration, as shown in Fig. 2.8. A desired velocity $\dot{q}_d$ is generated from the outer loop

$$\begin{aligned} e &= q_r - q, \quad \dot{e} = \frac{\mathrm{d}}{dt}e \\ \dot{q}_d &= K_p e + K_d \dot{e} \end{aligned} \tag{2.18}$$

which is then used to generate torque in the inner loop

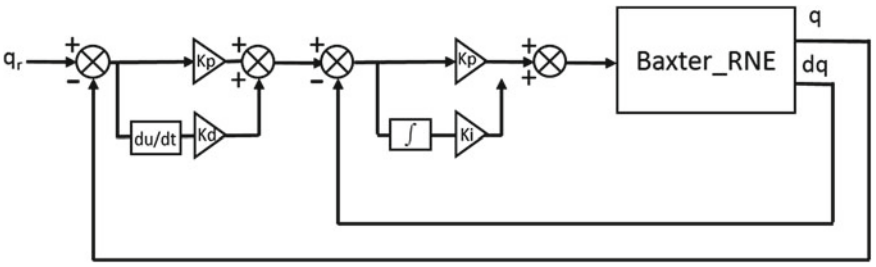


Fig. 2.8 Block diagram of torque control system

$$\dot{\epsilon} = \dot{q}_d - \dot{q}_r, \quad \epsilon = \int \dot{\epsilon}\, \mathrm{d}t$$
$$\tau_r = K_p \dot{\epsilon} + K_i \epsilon. \tag{2.19}$$

The trajectories of q_r were created in two ways: generated using sine and cosine patterns or using a touchpad input, both in Cartesian space. Inverse kinematics are performed using the inverse Jacobian method, that is

$$\dot{q}_r = J^{\dagger}(q)\dot{x}_r \tag{2.20}$$

where $\dot{x}_r$ is the reference Cartesian velocity and $J^{\dagger}$ is the pseudo-inverse of the Jacobian matrix. The selected test trajectories in Fig. 2.9, show the actual Cartesian test trajectories x which are calculated from $x = F(q)$, where $F(q)$ is the forward kinematics of the robot. For the experiment, the right-hand manipulator of the Baxter was driven through these three trajectories and data collected at 50 Hz, including joint positions and velocities $q, \dot{q}$, Cartesian position x and the estimated torques applied to the motors τ. This is calculated on board of the Baxter robot from the deflection of the internal springs (and large external springs at joint 2), summed with the automatic gravity compensation. The joint accelerations $\ddot{q}$ were estimated through first order numerical differentiation of $\dot{q}$, which accounts for the noise in the calculated results.

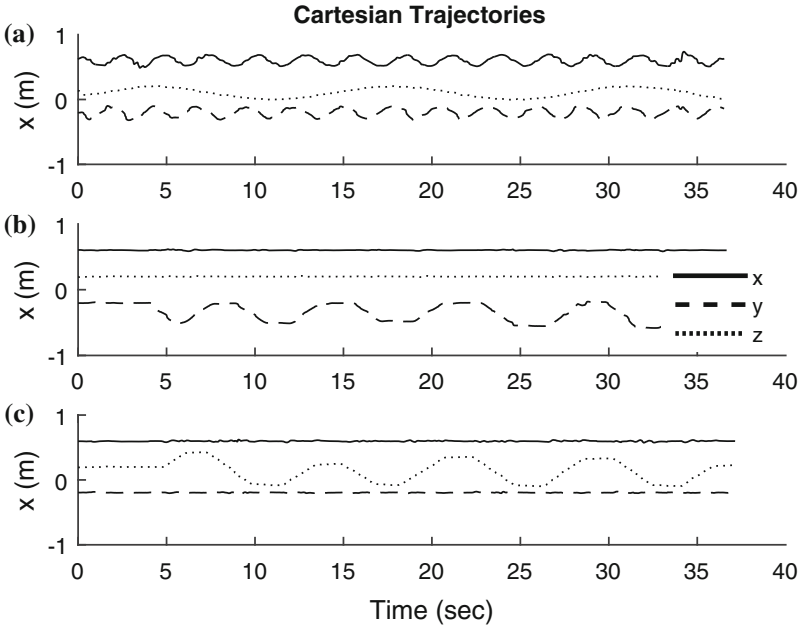


Fig. 2.9 Cartesian trajectories selected for experimentation. In **a** all three dimensions follow a sinusoidal pattern, with a lower frequency in the z-axis. For the trajectories in **b** and **c** the arm was moved only in the y and z-axes, respectively

To find the explicit form of Baxter manipulator dynamics, MATLAB's symbolic toolbox was utilized. In their raw state, the symbolic representations of the elements of $D(q)$, $C(q, \dot{q})$ and $G(q)$ have many coefficients (over half a million), they cannot be printed here.

To confirm the accuracy of the process, a numerical form was also created. Joint positions and velocities were recorded from the Baxter moving in three different trajectories, and the joint accelerations estimated through numerical differentiation, i.e., $\ddot{q}_i = \frac{d\dot{q}_i}{dt}$, where $dt = 0.02$ is the sampling period of the trajectory recorder. The results from the L–E form are compared against torques recorded from the Baxter, and torque trajectories generated using the RN–E method from Peter Corke's Robotics Toolbox [3]. It is possible to generate the analytical representation of Eq. (2.6) using this RN–E method, but only if n is small due to heavy memory consumption. Due to the way Baxter is controlled, the recorded torques are a sum of the actuator torques measured via internal spring deflection and two torque vectors acting on joint 2 (hysteresis and crosstalk) to compensate for the large external springs, mentioned previously. All results that are collected from the right-hand manipulator, with the end-effector aligned with the z-axis. No external forces were applied to the arm during testing.

In Fig. 2.10, the arm was moving in all three planes. It is noticeable that the torques generated from L–E and RN–E are much noisier; this is due to the numerical differentiation of the joint accelerations $\ddot{q}$. This could be reduced by passing it through a low pass filter. However, we can see that the shape of the trajectories is very similar.

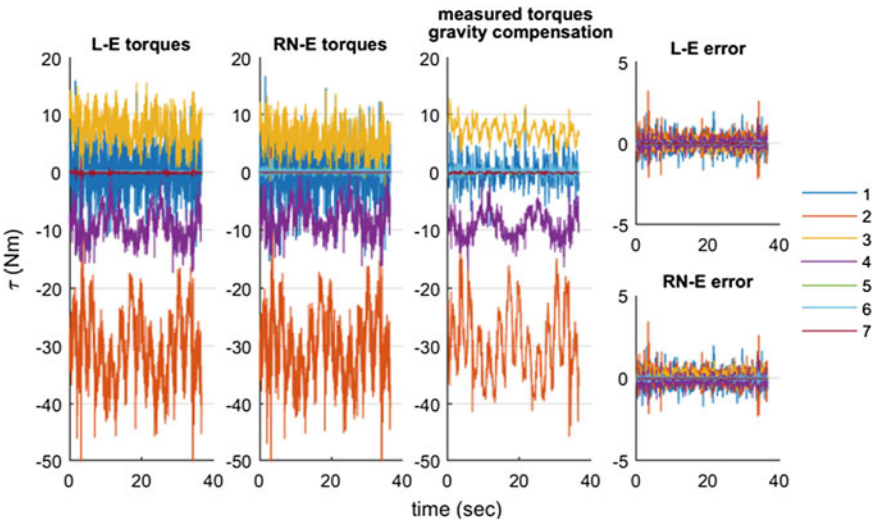


Fig. 2.10 Comparison of torque generated through L–E and RN–E methods with torques recorded from the Baxter robot during the trajectory from Fig. 2.9a. The trajectory for this sample was moving the end-effector in a circular trajectory in the x, y planes and in a cosine pattern in the z-axis, where $x = 0.6 + 0.1\sin(t)$, $y = -0.2 + 0.1\cos(t)$ and $z = 0.1 + 0.1\cos(0.2t)$. The errors (*far right*) are the modeled torques subtracted from the recorded torques

The first joint and distal joints 5–7 only require very small torque input as they are mostly unaffected by gravity. Examining the L–E error plot in Fig. 2.10, the noise dominates the largest errors but it is centered around zero for all joints, confirming that there are no bias errors. The RN–E error plot shows a similar range of error, but it is noticeable that the error for joint 3 has some positive bias.

In Fig. 2.11, we have similar results, with an even smaller error result. In this case, the arm was moved in a way to generate higher accelerations by quickly switching the target position in the y-axis only. This movement is mostly achieved using joint 2 at the shoulder, noticeable in the plots. The low error result in this case confirms a good match for the kinetic part of the dynamic model. Again, looking at the RN–E there is an obvious positive bias in the torques calculated for joint 3.

A slower trajectory was applied to the robot for the results in Fig. 2.12, moving primarily in the z-axis, which is evident by the large changes occurring in joint 4. Noise is reduced due to minimal acceleration in the trajectory. The error results again show no bias errors, and within a good tolerance of around ± 1.5 Nm which mostly can be accounted for by noise from the acceleration trajectory derivation.

A good comparison of the methods is shown in Table 2.6, where the average (and a sum total) error of both methods for each joint are shown side by side. These are calculated from each trajectory set, i.e., set 1 in Table 2.6 corresponds to the first trajectory results in Fig. 2.10. Looking through the table, it can be seen that the average error for each joint is comparable between the methods, apart from in joints 3–4 which have significantly larger errors in every set. This gives a good indication that the L–E method is not only accurate, but also slightly more accurate than the RN–E method.

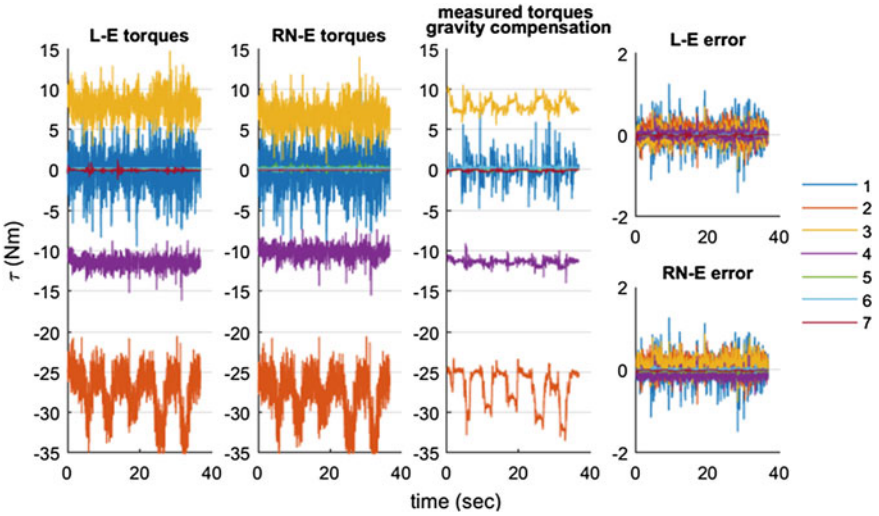


Fig. 2.11 Second comparison of torque trajectories from Fig. 2.9b; for this sample the end-effector is fixed in the x, z plane and is switched quickly between two positions in the y-axis

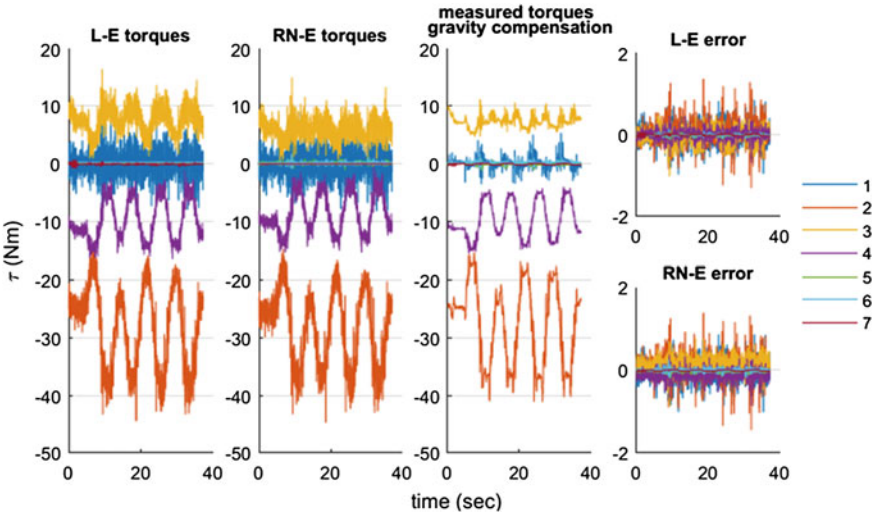


Fig. 2.12 Final comparison of torque trajectories from Fig. 2.9c input. The end-effector was moved up and down in the z-axis, and held constant in the x, y plane

Table 2.6 Averages of calculation errors for L–E and RN–E methods

Joint	Set					
	1		2		3	
	L–E	RN–E	L–E	RN–E	L–E	RN–E
$\lvert\bar{e}_1\rvert$	0.0105	0.0105	0.0148	0.0149	0.0061	0.0058
$\lvert\bar{e}_2\rvert$	0.1002	0.0665	0.0464	0.0931	0.0703	0.0872
$\lvert\bar{e}_3\rvert$	0.0475	0.1382	0.0083	0.1355	0.0367	0.1358
$\lvert\bar{e}_4\rvert$	0.0151	0.1231	0.0079	0.1210	0.0064	0.1148
$\lvert\bar{e}_5\rvert$	0.0068	0.0120	0.0099	0.0145	0.0079	0.0140
$\lvert\bar{e}_6\rvert$	0.0006	0.0103	0.0047	0.0136	0.0003	0.0113
$\lvert\bar{e}_7\rvert$	0.0012	0.0036	0.0003	0.0055	0.0013	0.0024
$\sum_{i=1}^{n} \bar{e}_i$	0.0710	0.0869	0.0527	0.1161	0.0366	0.1089

References

1. Denavit, J.: A kinematic notation for lower-pair mechanisms based on matrices. ASME J. Appl. Mech. **22**, 215–221 (1955)
2. Nethery, J.F., Spong, M.W.: Robotica: a mathematica package for robot analysis. Robot. Autom. Mag. IEEE **1**(1), 13–20 (1994)
3. Corke, P.: A robotics toolbox for matlab. Robot. Autom. Mag. IEEE **3**(1), 24–32 (1996)
4. Corke, P.: A simple and systematic approach to assigning denavit-hartenberg parameters. IEEE Trans. Robot. **23**(3), 590–594 (2007)
5. Ju, Z., Yang, C., Ma, H.: Kinematics modeling and experimental verification of baxter robot, Chinese Control Conference (CCC). **33**, 8518–8523 (2014)

6. About ROS: http://www.ros.org/about-ros/
7. Martinez, A., Fernández, E.: Learning ROS for Robotics Programming. Packt Publishing Ltd, Birmingham (2013)
8. RethinkRobotics baxter common: https://github.com/RethinkRobotics/baxter_common/tree/master/baxter_description
9. Tuck Arms Example: https://github.com/RethinkRobotics/sdk-docs/wiki/Tuck-Arms-Example
10. Uebel, M., Minis, I., Cleary, K.: Improved computed torque control for industrial robots, In: Proceedings of the IEEE International on Conference Robotics and Automation, pp. 528–533 (1992)
11. Poignet, P., Gautier, M.: Nonlinear model predictive control of a robot manipulator. In: Proceedings of the 6th IEEE International Workshop on Advanced Motion Control, pp. 401–406. (2000)
12. Feng, Y., Yu, X., Man, Z.: Non-singular terminal sliding mode control of rigid manipulators. Automatica **38**(12), 2159–2167 (2002)
13. Fu, K., Gonzalez, C.R.C.: Robotics Control, Sensing, Vision, and, Intelligence. McGraw-Hill, New York (1987)
14. Bejczy, A.K.: Robot Arm Dynamics and Control, pp. 33–669. Jet Propulsion Laboratory Technical Memo, Pasadena (1974)
15. Bejczy, A., Paul, R.: Simplified robot arm dynamics for control. In: IEEE 20th Conference on Decision and Control including the Symposium on Adaptive Processes (1981)
16. Megahed, S.M.: Principles of Robot Modelling and Simulation. Wiley, Hoboken (1993)
17. Lee, C., Lee, B., Nigam, R.: Development of the generalized d'alembert equations of motion for mechanical manipulators. In: 22nd IEEE Conference on Decision and Control, pp. 1205–1210 (1983)
18. Hollerbach, J.M.: A recursive lagrangian formulation of maniputator dynamics and a comparative study of dynamics formulation complexity. IEEE Trans. Syst. Man Cybern. **10**(11), 730–736 (1980)
19. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics: Modelling, Planning and Control. Springer, Heidelberg (2009)
20. Mckerrow, P.: Introduction to Robotics. Addison-Wesley Longman Publishing, Boston (1991)
21. Ju, Z., Yang, C., Ma, H.: Kinematic modeling and experimental verification of Baxter robot. In: Proceedings of the 33rd Chinese Control Conference Nanjing, China, pp. 8518–8523. 28–30 July 2014

http://www.springer.com/978-981-10-0829-0

Advanced Technologies in Modern Robotic Applications

Yang, C.; Ma, H.; Fu, M.

2016, XIV, 419 p. 269 illus., 233 illus. in color.,
Hardcover
ISBN: 978-981-10-0829-0