Impedance Control Under Uncertainty about Environment Properties

A Thesis presented

by

Sravana Sumanth Koneru

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Mechanical Engineering

Stony Brook University

August 2021

Stony Brook University

The Graduate School

Sravana Sumanth Koneru

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis

Dr. Nilanjan Chakraborty, Thesis Advisor
Assistant Professor, Department of Mechanical Engineering

Dr. Shikui Chen, Committee Chair
Associate Professor, Department of Mechanical Engineering

Dr. Shanshan Yao, Committee Member
Assistant Professor, Department of Mechanical Engineering

This thesis is accepted by the Graduate School

Eric Wertheimer
Dean of the Graduate School

Abstract of the Thesis

Impedance Control Under Uncertainty about Environment Properties

by

Sravana Sumanth Koneru

Master of Science

in

Mechanical Engineering

Stony Brook University

2021

For any robot manipulator to perform a task, which involves the manipulation of its surroundings, interaction with the environment is a key concern. Future robot applications, such as use of tools or close cooperation with humans, may be enabled by improved control of mechanical interaction. Successful execution of an interaction task with the environment by using motion control could be obtained only if the task were accurately planned, which is very hard to achieve and this would require an accurate model of both the robot manipulator and the environment. Even though manipulator model can be achieved with enough precision, a detailed description of the environment is difficult to achieve.

Impedance control is a well-established framework to manage the interaction of the end effector of a robot manipulator with the environment. Impedance control has received significant efforts in recent decades in robotics as impedance control aims to achieve the desired mechanical interaction with uncertain environment. The contact forces achieved with the interaction is highly dependent on the stiffness of both the environment and the manipulator. As the environment's stiffness is one parameter which is not in the control of the user, the only parameter which can be altered to perform a successful interaction task is manipulator's stiffness. But choosing this stiffness is not an easy task as it is very task specific and need to be changes when there is a change in the environment stiffness.

This thesis proposes a method to successfully perform impedance control without changing the stiffness of the manipulator for every change in the environment. This thesis uses a moving desired configuration/frame to perform the tasks while keeping the stiffness of the manipulator as constant. Also, Screw Linear Interpolation (ScLERP) is used in choosing the desired configuration to ensure the required end effector trajectory.

The capability of this method is then validated by performing a series of simulations of a 7 degree-of-freedom (DOF) robot (Baxter Arm) performing several household tasks with different types of constraints. Example tasks include pushing buttons and opening doors.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

I would like to express my sincere and heartfelt gratitude to my research advisor, Professor Nilanjan Chakraborty for his guidance and continued support throughout my graduate studies. I would like to thank him for supporting me and help me understand things.

I would also like to thank Professor Shikui Chen and Professor Shanshan Yao for their presence in the committee and providing valuable input.

I am grateful to my parents, family and friends for encouraging me into pursuing my masters and always staying by my side no matter what.

I owe a huge debt of gratitude to the members of the Interacting Robotic Systems Lab, especially Dasharadhan Mahalingam and Anirban Sinha without whom, this work would not have been possible.

Lastly, I would like to thank everyone who helped me or inspired me in someway or the other.

Chapter 1

Introduction

Robots have become an integral part of human's day to day life. From industrial lines to household tasks, robots are now employed to manipulate and interact with environment to perform multiple tasks. The robot manipulators that interact with the surroundings to accomplish certain tasks require to exploit contact forces at the end effector in a safe manner. Tasks that involve interaction and force exchange with the environment cannot be accomplished solely through position control.

Mechanical interaction with objects is one of the fundamentally important robot behaviors. One of the principal necessities for the success of the manipulation task is the ability to handle the interaction between the manipulator and the environment. The quantity that describes the state of interaction more effectively is the $contact\ force$ at the manipulator's end-effector. High values of contact force are generally undesirable since they may stress both the manipulator and the manipulated object and a compliant behaviour is to be ensured at the interaction. The day to day activities are bringing the robots into more unstructured environments where the manipulator is expected to perform tasks while having interaction with the environment. These situations lead us to use methods based on impedance control.

Impedance control [1] is an efficient method for handling robot force control and has been widely studied and utilized because of its low computational complexity and strong robustness. Impedance control includes regulation and stabilization of robot motion by creating a mathematical relationship between the interaction forces and the reference trajectories. The impedance controller imitates a virtual spring mass damper system between the environment and the robot end effector, which imposes a dynamic behaviour to the interaction between the robot end effector and the environment and in turn provides a compliant behaviour between the systems while the interaction takes place.

This study tries to implement the impedance control to do multiple household tasks while proposing a novel method to accomplish task with moving the desired goal pose as needed for the task.

1.1 Motivation

The interaction between the end effector and the environment is highly influenced by their respective compliance features i.e., stiffness etc. The stiffness of the environment is something which cannot be controlled by the user and the stiffness of the manipulator (denoted by a matrix $\mathbf{K_p}$) is the altering parameter for a successful execution of a task. The selection of the elements of the matrix $\mathbf{K_p}$ does follow a defined method and is made by taking in to account the geometry and the mechanical features of the environment.

For a given contact stiffness at the environment, large values are taken for the values of the elements of manipulator stiffness for those directions along which the environment has to comply and small values are taken in those directions where the end effector has to comply. This is mostly a Trial and error scheme, where we choose a specific stiffness in the required direction for a task and if the task is not accomplished, we tune the values accordingly and perform the task again. This method might be ideal for an industrial robot or a robot repeating the same task again and again.

To accomplish household tasks such as opening a door, rotating a door knob or a simple task of pushing a button, the above stated scheme might not be of much use. Consider a task of opening a door, for performing the task, we tune the values of stiffness for the manipulator to successfully perform the task. But the task might not be successfully performed on any other door if it is more or less stiffer than the previous door (or has different mechanical features). The user should be constantly changing or tuning the values of the $\mathbf{K_p}$ according to the task and also according to the features of the task performed which is not desirable.

This problem is taken as a motivation for this thesis and we focus on developing a method where the the stiffness need not be tuned often. This would be helpful for the user to perform the tasks without frequently changing the stiffness and damping values of the manipulator.

1.2 Solution Approach

While the traditional impedance control algorithms have a necessity of choosing the stiffness of the manipulator, it is very difficult to know the stiffness of the environment for each task and the stiffness is to be chosen or altered accordingly if the chosen stiffness does not accomplish the defined task. The impedance control uses the stiffness, mechanical features of the robot and the force, moment wrench applied at the end effector to generate control torques to move the manipulator. In the proposed method, instead of altering the stiffness values, we keep the stiffness value optimum and try to adjust and change the goal pose $\in SE(3)$ in such a way such that the required joint torques are produced to achieve the required goal. Screw Linear Interpolation (ScLERP) is used to obtain the guiding or intermediate desired poses to accomplish the given task.

1.3 Contribution

In this thesis, we put forward a method to accomplish a task using a robot manipulator having interaction with an environment without knowing its Mechanical features (Stiffness etc.). The following contributions are made by this study:

- An algorithm is proposed for impedance control with a dynamic moving goal/desired frame keeping the stiffness parameters of the manipulator constant to accomplish the required task even if we do not have an accurate knowledge of the environment stiffness.
- Screw Linear Interpolation is introduced while choosing the desired poses, so that the achieved trajectory follows the desired trajectory.
- This method is then implemented and simulated on a 7-DOF redundant Baxter Robot arm in MATLAB to carry out the household tasks which include interaction and force exchange with the environment.

1.4 Outline

The thesis is organized as follows. Chapter 2 discusses about the mathematical background necessary for the thesis. In chapter 3, the background, past and related works and a brief overview of the robot dynamics and impedance control are discussed. Chapter 4 illustrates the problem statement and approach to the solution. The simulated experiments and the results from the simulation are presented in chapter 5 and finally the thesis is concluded in chapter 6 with a brief discussion about future work.

Chapter 2

Mathematical Preliminaries

This chapter introduces briefly about the mathematical concepts used in this thesis. We use the following notations and definitions throughout the thesis.

- **Joint Space:** The set of all possible joint displacements (angular or linear) of a manipulator is called the joint space of the manipulator.
- **Task Space:** The set of all possible end effector poses is called the task space or end-effector space.
- $\mathbb{R}^n$: Vector space of dimension n with real entities.

2.1 Rigid Body Motion Group

Rigid body: A rigid body is a collection of particles such that the distance between any two particles is unchanged irrespective of the motion of the body or the forces acting on the body. If $\mathbf{p}, \mathbf{q}$ are any two points on a rigid body B, then according to the above definition, even if body B moves

$$||\mathbf{p}(t) - \mathbf{q}(t)|| = ||\mathbf{p}(0) - \mathbf{q}(0)|| = c$$

The distance between the points **p** and **q** remains a constant c at both time t and time 0.

A rigid motion is defined as a continuous movement of the particles such that the distance between any two particles remains unchanged at all times. A rigid motion can be expressed as a function called rigid transformation. This rigid transformation consists of translation and rotation. A rigid transformation mapping g : $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ should satisfy the following properties:

1. Length is preserved : $\|g(\text{p}) - g(\text{q})\| = \|\text{p} - \text{q}\|$, $\forall$ p, q $\in \mathbb{R}^3$.
2. The cross product is preserved: $g_*{}^*(\text{v} \times \text{w}) = g_*(\text{v}) \times g_*(\text{w})$ $\forall$ v,w $\in \mathbb{R}^3$

The orientation of rigid bodies are represented by rotation matrices. A rotation matrix **R** is a real matrix of order 3×3 and it gives us the orientation of a rigid body in three dimensional space.

2.1.1 Rotation Matrix

The orientation of the rigid bodies in $\mathbb{R}^3$ are represented by a 3 x 3 matrix, called the Rotation matrix. The rotation matrix has the following properties:

- All columns and rows of a rotation matrix are unit vectors.
- All columns and rows of a rotation matrix are orthonormal.
- The inverse of a rotation matrix is its transpose.

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}_{3\times3} \implies \mathbf{R}^T = \mathbf{R}^{-1}$$

- The determinant if rotation matrix is **1**, i.e., $det(\mathbf{R}) = 1$.

All the rotation matrices form a special group called the " Special Orthogonal group" of dimension 3 or $SO(3)$. A special orthogonal group of dimension n is defined as:

$$SO(n) = \{\mathbf{R} \in \mathbb{R}^{n\times n};\ \mathbf{R}\mathbf{R}^T = \mathbf{I}_{3\times 3},\ det(R) = 1\}$$

The other representations of orientation of a rigid body in $\mathbb{R}^3$ such as Euler angles are discussed in later sections.

2.1.2 Exponential coordinates for rotation

Let $\omega \in \mathbb{R}^3$ be a unit vector. Let the rigid body rotate by an angle θ about the axis ω. Then the resulting rotation matrix is given by:

$$\mathbf{R}(\omega, \theta) = e^{\hat{\omega}\theta}$$

From the expansion of the exponential, the following equation is obtained, i.e.:

$$\mathbf{R}(\omega, \theta) = \mathbf{I} + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \cdots\right)\hat{\omega} + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \cdots\right)\hat{\omega}^2$$

$$\mathbf{R}(\omega, \theta) = e^{\omega\theta} = I + \ \hat{\omega} sin\theta + \ \hat{\omega}^2(1 - cos\theta)$$

This equation is called the Rodrigues formula. Using this we can compute the rotation matrix, $\mathbf{R}$, given the rotation angle θ and the axis ω.

Note: The hat operator ($\hat{}$) on a vector is the skew symmetric representation of the vector. In the above case, for $\omega = \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}$, $\hat{\omega} = \begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}$
.

2.1.3 Transformation matrix

As discussed in the previous sections, a rigid body transformation consists of both rotation and translation. The configuration of any rigid body can be described as:

$$g = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix}$$

where, $\mathbf{R} \in SO(3)$ is the rotation matrix of the body frame with respect to the world frame and $\mathbf{p} \in \mathbb{R}$ is the position vector of the origin of the body frame with respect to world frame. 0 is a 1 x 3 vector of zeros. The inverse of this transformation is given as

$$g^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{p} \\ 0 & 1 \end{bmatrix}$$

2.1.4 Exponential coordinates in rigid body transformations

Similar to the exponential representation of a rigid body rotation, we can represent the rigid body transformations as exponential coordinates. Consider a link rotating about a unit axis $\omega \in \mathbb{R}^3$. Consider a point on the axis $q \in \mathbb{R}^3$. We define a 4 x 4 matrix $\hat{\xi}$, where

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}, \text{ for combined translation and rotation}$$

$$\hat{\xi} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix}, \text{ for pure translation}$$

where v = - $\omega \times q$. The set of all 4 x 4 matrices of the above form are defined as the space $se(3)$ defined as:

$$se(3) = \{(v, \hat{\omega}) : v \in \mathbb{R}^3, \hat{\omega} \in so(3)\}$$

Now, the elements in se(3) are converted to elements in SE(3) by exponential map. Given a twist ξ in se(3),

$$e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} \in SE(3) \text{ for } \omega \neq 0 \tag{2.1}$$

$$e^{\hat{\xi}\theta} = \begin{bmatrix} I & v\theta \\ 0 & 1 \end{bmatrix} \in SE(3) \text{ for } \omega = 0 \tag{2.2}$$

These exponential coordinates of the rigid body transformation makes the forward kinematics map of the manipulator easy without complicated calculation. The forward kinematics map is shown in the later sections.

2.2 Euler Angles

As we know, while a point can be described only by its position, bodies require orientation in addition to position to define its configuration. Generally rotation matrices are a subset of SO(3) and are represented by 3x3 matrices. These rotation matrices can be represented by different parameters such as Euler angles, quaternions etc.

Any rotation in space can be defined as a sequence of elementary rotations about the x, y and z axes. These Euler angles can be taken in multiple sequences of elementary rotations about the axes i.e., ZYX, ZYZ, ZXZ etc. Here, we used ZYX Euler angles through out the article.

2.2.1 Mapping between rotation matrix and Euler Angles

Let the frame B be rotated about z-axis by an angle z. The rotated frame is then rotated about y-axis and then about x-axis in that order.

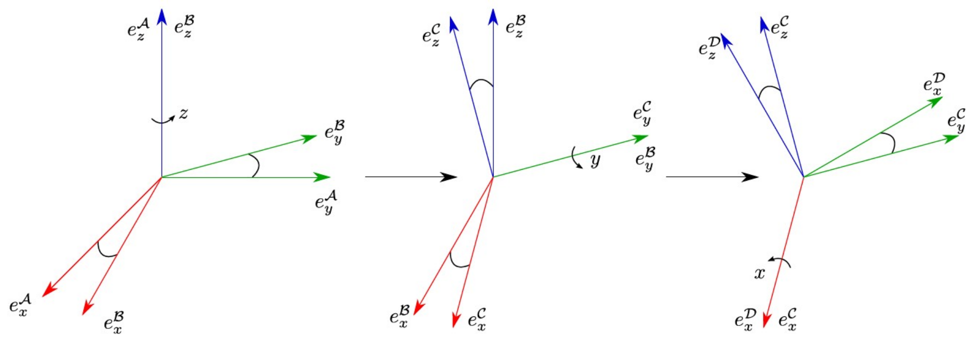


Figure 2.1: ZYX Euler Angles as three successive rotations around z, y, and x - axes.

Then the the whole rotation matrix obtained by the three elementary rotations about Z,Y and X axes is given by:

$$R = R(\phi_z) * R(\phi_y) * R(\phi_x);$$

$$R = \begin{bmatrix} cos(\phi_z) & -sin(\phi_z) & 0 \\ sin(\phi_z) & cos(\phi_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\phi_y) & 0 & sin(\phi_y) \\ 0 & 1 & 0 \\ -sin(\phi_y) & 0 & cos(\phi_y) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi_x) & -sin(\phi_x) \\ 0 & sin(\phi_x) & cos(\phi_x) \end{bmatrix}$$

So, when given the three ZYX euler angles, the rotation matrix is given by:

$$R = \begin{bmatrix} c_y c_z & c_z s_x s_y - c_x s_z & s_x s_z + c_x c_z s_y \\ c_y s_z & c_x c_z + s_x s_y s_z & c_x s_y s_z - c_z s_x \\ -s_y & c_y s_x & c_x c_y \end{bmatrix} \tag{2.3}$$

where $c_x = cos(\phi_x)$ and $s_x = sin(\phi_x)$. Similarly, given the rotation matrix

$$R = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

the Euler angles are given by:

$$\begin{bmatrix} \phi_z \\ \phi_y \\ \phi_x \end{bmatrix} = \begin{bmatrix} atan2(c_{21}, c_{11}) \\ atan2(-c_{31}, \sqrt{c_{32}^2 + c_{33}^2}) \\ atan2(c_{32}, c_{33}) \end{bmatrix} \tag{2.4}$$

2.2.2 Derivative of Euler Angles

As discussed above, like a rotation or an angular displacement can be expressed in terms of Euler angles. Similarly, the angular velocity can be expressed in terms of derivatives of Euler angles and vice versa. Given the set of ZYX Euler angles $\begin{bmatrix} \phi_z & \phi_y & \phi_x \end{bmatrix}^T$ and their time derivatives $\begin{bmatrix} \dot{\phi}_z & \dot{\phi}_y & \dot{\phi}_x \end{bmatrix}^T$. The angular velocities are mapped to euler angle derivates as

$$\omega = E_R(\phi_R) * \dot{\phi}_R$$

The mapping between the angular velocities and the derivatives of Euler angles is given by: [2]

$$E_{EulerZYX} = \begin{bmatrix} 0 & -sin(\phi_z) & cos(\phi_y)cos(\phi_z) \\ 0 & cos(\phi_z) & cos(\phi_y)sin(\phi_z) \\ 1 & 0 & -sin(\phi_y) \end{bmatrix} \tag{2.5}$$

The inverse mapping is given by:

$$E_{EulerZYX}^{-1} = J_r = \begin{bmatrix} \frac{cos(\phi_z)sin(\phi_y)}{cos(\phi_y)} & \frac{sin(\phi_y)sin(\phi_z)}{cos(\phi_y)} & 1 \\ -sin(\phi_z) & cos(\phi_z) & 0 \\ \frac{cos(\phi_z)}{cos(\phi_y)} & \frac{sin(\phi_z)}{cos(\phi_y)} & 0 \end{bmatrix} \tag{2.6}$$

2.3 Quaternions

Quaternions are generalizations of complex numbers. Any rotation can be represented in the form of a unit quaternion. A unit quaternion is a quaternion whose magnitude is 1. A quaternion is defined as

$$Q = q_0 + q_1 i + q_2 j + q_3 k$$

where, $q_0 \in \mathbb{R}$ is the scalar part and $\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \in \mathbb{R}^3$ is the vector part with

$$\begin{aligned} ii = jj = kk = ijk = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j \end{aligned}$$

The quaternion is generally represented as $(q_0, \mathbf{q})$. Any scalar $a \in \mathbb{R}$ can be represented as a quaternion with the scalar part $q_0 = a$ and the vector part $\mathbf{q} = \mathbf{0}$. Similarly, any vector $\boldsymbol{v} \in \mathbb{R}^3$ can be represented as a quaternion with the scalar part $q_0 = 0$ and the vector part $\mathbf{q} = \boldsymbol{v}$. Thus a vector v is written as (0,v).

2.3.1 Quaternion Operations

In this section, the operations which can be performed on the quaternions are introduced. Let $Q = (q_0, \mathbf{q})$ and $P = (p_0, \mathbf{p})$ be two quaternions, the following operations can be performed on these quaternions.

Magnitude of a Quaternion

The magnitude of a quaternion Q is given by

$$||Q|| = Q \otimes Q^* = q_0^2 + q_1^2 + q_2^2 + q_3^2$$

Quaternion Conjugate

The conjugate of the quaternion Q is given by

$$Q^* = (q_0, -\mathbf{q})$$

Scalar Product

For any scalar, $s \in \mathbb{R}$, the scalar product of the quaternion Q with the scalar s is given by

$$sQ = (sq_0, s\mathbf{q})$$

Quaternion Addition

The sum of the quaternions Q and P is given by

$$Q + P = (q_0 + p_0, \mathbf{q} + \mathbf{p})$$

Quaternion Multiplication

The product of the quaternions Q and P is given by

$$Q \otimes P = (q_0 p_0 - \mathbf{q}.\mathbf{p}, q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{q} \times \mathbf{p})$$

2.3.2 Exponential coordinates and unit quaternions

Let $\mathbf{R}$ a rotation matrix with axis ω and angle θ, i.e.,

$$\mathbf{R} = e^{\hat{\omega}\theta},\ \theta \in \mathbb{R},\ \omega \in \mathbb{R}^3,\ ||\omega|| = 1.$$

Then the corresponding quaternion is given by

$$Q = \left(\cos\frac{\theta}{2}, \boldsymbol{\omega}\sin\frac{\theta}{2}\right)$$

Conversely, for a given unit quaternion, $Q = (q_0, \mathbf{q})$, we can determine the exponential coordinates as

$$\theta = 2\arccos q_0, \quad \boldsymbol{\omega} = \frac{\mathbf{q}}{\sin\frac{\theta}{2}}$$

2.4 Dual Numbers

A dual number is defined as

$$D = a + \epsilon b,\ \epsilon \neq 0,\ \epsilon^2 = 0$$

here, a $\in$ $\mathbb{R}$ is the real part and b $\in$ $\mathbb{R}$ is the dual part.

2.4.1 Operations on Dual numbers

Let $D_1 = a_1 + \epsilon b_1$ and $D_2 = a_2 + \epsilon b_2$ be two dual numbers. The following operations can be defined for dual numbers:

1. **Addition:** The addition of two dual numbers is given by

$$D_1 + D_2 = (a_1 + a_2) + \epsilon(b_1 + b_2)$$

2. **Multiplication:** The product of two dual numbers is given by

$$D_1 \otimes D_2 = a_1a_2 + \epsilon(a_1b_2 + a_2b_1)$$

3. **Inverse:** The inverse of a dual number D = a + ϵb is given by

$$D^{-1} = \frac{1}{a}\left(1 - \epsilon\frac{b}{a}\right), \text{assuming } a \neq 0$$

4. **Conjugate:** The conjugate of a dual number D = a + ϵb is denoted by D^* and is given by

$$D^* = a - \epsilon b$$

   Also, $D \otimes D^* = D^* \otimes D = a^2$

2.5 Dual Vector

A dual vector is a dual number whose real part and dual part are both vectors i.e., elements of $\mathbb{R}^n$.

$$\mathbf{D} = \mathbf{a} + \epsilon\mathbf{b},\ \mathbf{a}, \mathbf{b} \in \mathbb{R}^n$$

for $\mathbb{R}^3$, the dual vector is form of the form:

$$\mathbf{D} = \mathbf{a} + \epsilon\mathbf{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \epsilon \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_1 + \epsilon b_1 \\ a_2 + \epsilon b_2 \\ a_3 + \epsilon b_3 \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix}$$

2.5.1 Operations on Dual Vectors

Let $E_i = e_{ir} + \epsilon e_{id}$ and $F_i = f_{ir} + \epsilon f_{id}$, i = 1,2,3 be dual numbers, where r is subscript denotes the real part and d denotes the dual part. The following operations are defined on dual vectors.

1. **Addition:**

$$\mathbf{E} + \mathbf{F} = \begin{bmatrix} E_1 + F_1 \\ E_2 + F_2 \\ E_3 + F_3 \end{bmatrix}$$

2. **Product of dual number with a dual vector:**

$$D\mathbf{E} = \begin{bmatrix} D \otimes E_1 \\ D \otimes E_2 \\ D \otimes E_3 \end{bmatrix}$$

3. **Dot Product of two dual vectors:**

$$\mathbf{E}.\mathbf{F} = E_1 \otimes F_1 + E_2 \otimes F_2 + E_3 \otimes F_3$$

4. **Cross Product of two dual Vectors**

$$\mathbf{E} \times \mathbf{F} = \begin{bmatrix} E_2 \otimes F_3 - E_3 \otimes F_2 \\ E_3 \otimes F_1 - E_1 \otimes F_3 \\ E_1 \otimes F_2 - E_2 \otimes F_1 \end{bmatrix}$$

2.6 Dual Quaternions

Dual quaternions are dual numbers with both real and dual part being quaternions.

A Dual Quaternion can be represented as $\mathbf{A} = \mathbf{P} + \epsilon\mathbf{Q}$

Where, $\mathbf{P} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$ and $\mathbf{Q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ are quaternions

2.6.1 Operations on Dual Quaternions

Let $\mathbf{A} = \mathbf{P} + \epsilon\mathbf{Q}$ and $\mathbf{B} = \mathbf{U} + \epsilon\mathbf{V}$ be two dual quaternions, where $\mathbf{P}, \mathbf{Q}, \mathbf{U}, \mathbf{V}$ are quaternions. The following operations are defined for dual quaternions:

1. **Addition:**
$$\mathbf{A} + \mathbf{B} = (\mathbf{P} + \mathbf{U}) + \epsilon(\mathbf{Q} + \mathbf{V})$$

2. **Multiplication:**
$$\mathbf{A} \otimes \mathbf{B} = \mathbf{P} \otimes \mathbf{U} + \epsilon(\mathbf{Q} \otimes \mathbf{U} + \mathbf{P} \otimes \mathbf{V})$$

   It is to be noted that the $\otimes$ on the left represents dual quaternion multiplication whereas $\otimes$ on the right represents quaternion multiplication.

3. **Conjugate:** The conjugate of $\mathbf{A} = \mathbf{P} + \epsilon\mathbf{Q}$ is denoted by $\mathbf{A}^*$ defined by
$$\mathbf{A}^* = \mathbf{P}^* + \epsilon\mathbf{Q}^*$$

2.6.2 Unit Dual Quaternion

A dual quaternion $\mathbf{A} = \mathbf{P} + \epsilon\mathbf{Q}$ is a unit dual quaternion if $\mathbf{A} \otimes \mathbf{A}^* = \mathbf{1}$ or

$$p_0^2 + p_1^2 + p_2^2 + p_3^2 = 1 \text{ and}$$

$$p_0q_0 + p_1q_1 + p_2q_2 + p_3q_3 = 0$$

Any rigid body transformation can be represented as a unit dual quaternion.

2.6.3 Dual Quaternions and Rigid Body Transformations

As stated above, any rigid body transformation can be represented as a unit dual quaternion. Let $\mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$ be a rigid body transformation and $\mathbf{A} = \mathbf{A}_r + \epsilon \mathbf{A}_d$ be the corresponding dual quaternion.

Here, $\mathbf{A}_r$ is a unit quaternion corresponding to the rotation matrix $\mathbf{R}$.

$$\mathbf{A}_r = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2}$$

$\mathbf{A}_d$ is another quaternion such that $\mathbf{A}_r^T \mathbf{A}_d = 0$. $\mathbf{A}_d$ is given by

$$\mathbf{A}_d = \frac{1}{2}\mathbf{p} \otimes \mathbf{A}_r$$

The unit dual quaternion representing the transformation $\mathbf{G}$ is given by

$$\mathbf{A} = \mathbf{A}_r + \frac{1}{2}\epsilon\mathbf{p} \otimes \mathbf{A}_r$$

For pure translation along the axis $\mathbf{v}$ by a magnitude d,

$$\mathbf{A}_r = \begin{bmatrix} 1 \\ \mathbf{0}_{3\times 1} \end{bmatrix} \quad \mathbf{A}_d = \begin{bmatrix} 0 \\ \frac{d}{2}\mathbf{v} \end{bmatrix}$$

2.6.4 Power of a Unit Dual Quaternion

Let $\mathbf{A} = \mathbf{A}_r + \epsilon\mathbf{A}_d$ be a dual quaternion corresponding to a rigid body displacement.

$$\mathbf{A}_r = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2}$$
$$\mathbf{A}_d = \frac{1}{2}\mathbf{p}\otimes\mathbf{A}_r$$

The power of A i.e., $\mathbf{A}^\tau$ can be computed as follows

Determine θ and $\mathbf{u}$ from

$$\mathbf{A}_r = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2}$$
$$\mathbf{p} = 2\mathbf{A}_d\otimes\mathbf{A}_r^*$$

- **Case 1 :** $\theta \neq 0$

$$d = \mathbf{p}.\mathbf{u}$$
$$\mathbf{m} = \frac{1}{2}\left(\mathbf{p}\times\mathbf{u} + (\mathbf{p} - d\mathbf{u})\cot\frac{\theta}{2}\right)$$
$$\mathbf{A}_r^\tau = \begin{bmatrix}\cos\frac{\tau\theta}{2}\\ \mathbf{u}\sin\frac{\tau\theta}{2}\end{bmatrix} \quad \mathbf{A}_d^\tau = \begin{bmatrix}-\frac{1}{2}\tau d\sin\frac{\tau\theta}{2}\\ \mathbf{m}\sin\frac{\tau\theta}{2} + \frac{1}{2}\tau d\mathbf{u}\cos\frac{\tau\theta}{2}\end{bmatrix}$$

- **Case 2 :** $\theta = 0$ (Pure translation)

$$\mathbf{A}_d \text{ is of the form } \mathbf{A}_d = \begin{bmatrix}0\\ \frac{d}{2}\mathbf{v}\end{bmatrix}$$

Determine $\mathbf{u}$ and d from $\mathbf{A}_d$

$$\mathbf{A}_r^\tau = \begin{bmatrix}\cos\frac{\tau\theta}{2}\\ \mathbf{v}\sin\frac{\tau\theta}{2}\end{bmatrix} \quad \mathbf{A}_d^\tau = \begin{bmatrix}-\frac{1}{2}\tau d\sin\frac{\tau\theta}{2}\\ \frac{1}{2}\tau d\mathbf{v}\cos\frac{\tau\theta}{2}\end{bmatrix}$$

2.7 Screw Linear Interpolation (ScLERP)

Screw Linear Interpolation (ScLERP) is a method of interpolating between two rigid body transformations in $SE(3)$ and is analogous to the straight line interpolation between two point in the Euclidean space. Using ScLERP results in a screw motion between two poses in $SE(3)$. ScLERP can be performed between any two poses in $SE(3)$ by performing dual quaternion interpolation.

Screw Linear Interpolation between two rigid body transformations can be performed by using dual quaternion interpolation. Let $\mathbf{A} = \mathbf{A}_r + \epsilon\mathbf{A}_d$ and $\mathbf{B} = \mathbf{B}_r + \epsilon\mathbf{B}_d$ by two unit dual quaternions representing the two rigid body transformations between which we want to perform screw linear interpolation. Let $\tau \in [0, 1]$ be the interpolation parameter.

Any pose on the screw path between $\mathbf{A}$ and $\mathbf{B}$ is given by

$$\mathbf{C}(\tau) = \mathbf{A} \otimes (\mathbf{A}^* \otimes \mathbf{B})^\tau \tag{2.7}$$

Algorithm 1: Screw Linear Interpolation

```
  input : Initial Pose $\mathbf{G}_i \in SE(3)$
          Final Pose $\mathbf{G}_f \in SE(3)$
          Interpolation Parameter $\tau \in [0, 1]$
  output: The interpolated pose $\mathbf{G}_\tau \in SE(3)$
1 procedure ScLERP($\mathbf{G}_i, \mathbf{G}_f, \tau$):
2     Determine the Unit Dual Quaternion representations $\mathbf{D}_i$ and $\mathbf{D}_f$
        of $\mathbf{G}_i$ and $\mathbf{G}_f$ respectively
3     Determine the Interpolated Pose $\mathbf{D}_\tau = \mathbf{D}_i \otimes (\mathbf{D}_i^* \otimes \mathbf{D}_f)^\tau$
4     Convert $\mathbf{D}_\tau$ into its $SE(3)$ representation $\mathbf{G}_\tau$
5     return $\mathbf{G}_t$
```

2.8 Adjoint Matrix

Any moving rigid body has a linear and angular velocity. We represent both the linear velocity **v** and the angular velocity ω in a single 6 x 1 vector **V** is called the *generalized velocity* of the rigid body, it is also called as *twist* denoted as:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix}$$

The matrix that converts velocity twists in one reference frame to another reference frame is called the Adjoint transformation. For any $g \in SE(3)$

$$g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

the adjoint of g, represented as Ad_g as a mapping Ad_g is given by:

$$Ad_g = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix}$$

The inverse mapping is the inverse if the adjoint matrix of g which is same as the adjoint of the inverse of the g. It is given by:

$$Ad_g^{-1} = \begin{bmatrix} R^T & -R^T\hat{p} \\ 0 & R^T \end{bmatrix} = Ad_{g^{-1}} \qquad (2.8)$$

Chapter 3

Background

The previous chapter provided a brief introduction to the mathematical preliminaries required for this thesis. This chapter provides the necessary background and previous research work which are relevant to this thesis. In

3.1 Forward Kinematics of a manipulator

Given a configuration of the robot i.e., the joint angles of the robot the forward kinematics helps to find the configuration of the end-effector frame with respect to base frame. Consider a serial chain manipulator with n joints and n-1 links. The joint space is defined as the set of all possible joint displacements (angular for revolute or linear of prismatic joints).

Let the transformation between the world frame {S} and the body frame {T} be $g_{st}(0)$. Let $\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}$ be the twist corresponding to rigid body transformation and θ be the angle of rotation so that the frame {T} transforms to {T'}. Therefore the transformation between the world frame {S} and {T'} is given as:

$$g_{st}(\theta) = e^{\hat{\xi}\theta} g_{st}(0)$$

This can be used to find the forward kinematics map of a serial chain manipulator. The individual joint motions can be combined together to obtain the forward kinematics map. Given the joint angles q, the forward kinematics map which maps joint space to end-effector space is given by,

$$g_{st}(q) = e^{\hat{\xi_1}\theta_1} e^{\hat{\xi_2}\theta_2} e^{\hat{\xi_3}\theta_3} \ldots e^{\hat{\xi_n}\theta_n} g_{st}(0) \tag{3.1}$$

where $g_{st}(0)$ is the transformation between tool and world frame at reference configuration.

3.2 Velocity Kinematics of a manipulator

The velocity kinematics of a serial chain manipulators relates joint angle rates of the manipulator to the end effector velocity.

3.2.1 Manipulator Jacobian

Consider a n-DOF manipulator. Let $\Theta = [\theta_1\, \theta_2 \ldots \theta_n]^T$ be the n×1 vector of joint angles of the manipulator and $\dot{\Theta} = [\dot{\theta}_1\, \dot{\theta}_2 \ldots \dot{\theta}_n]^T$ be the n×1 vector of joint angle rates of the manipulator. The relation between the joint angular rates and the velocity of the end effector is given by:

$$V = J(\Theta)\dot{\Theta}$$

where $J(\Theta)$ is a 6×n matrix called the manipulator jacobian.

Spatial Jacobian:

The spatial jacobian provides the relation between joint angle velocities and spatial velocity of the end effector. The spatial velocity of end effector is denoted by $V^s = \begin{bmatrix} v^s \\ \omega^s \end{bmatrix}$, where ω^s is the angular velocity of the tool frame with respect to spatial frame, expresses in spatial frame. The linear velocity v^s is of a point on the end-effector that is located at the origin of the world frame at any particular instant.

The relation between spatial velocity of the end effector V^s and the joint velocities $\dot{\theta}$ is given by:

$$V^s = J^s(\Theta)\dot{\Theta}$$

Let $\xi_1, \xi_2, \xi_3, \ldots, \xi_n$ be the twists of n joints of the manipulator at reference configuration. The spatial jacobian is given by

$$J^s(\Theta) = [\xi_1\ \xi_2'\ \xi_3' \ldots \xi_n'], \text{ where, } \xi_i' = Ad_{g1,i-1}\xi_i\ i = 2, 3, \ldots, n. \tag{3.2}$$

where $Ad_{g1,i-1}$ is the adjoint matrix corresponding to $g_{1,i-1}$ and can be obtained by using exponential coordinates as shown in forward kinematics section.

Analytic Jacobian

The Analytic jacobian provides the relation between the joint angle velocities and derivatives of the Euler angles.

$$J_A = A_r.A.J^s$$

where,

$$A = \begin{bmatrix} \mathbf{I}_{3\times3} & -\hat{p} \\ 0 & \mathbf{I}_{3\times3} \end{bmatrix}$$

and

$$A_r = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & J_r \end{bmatrix} \tag{3.3}$$

where, Jr is the mapping between the Euler angle derivatives and the angular velocity ω.

3.3 Dynamics of a manipulator

Dynamic model of a manipulator plays an important role for the design of control algorithm and simulation of motion. This simulation of motion of manipulator allows us to test the control algorithm without the need to use a physical system. In this section, we obtain the dynamic parameters for the implementation of the control algorithm.

The dynamic model of a manipulator provides a description of the relationship between the joint actuator torques and the motion of the structure. With Lagrange formulation, the equations of motion can be derived in a systematic way independently of the reference coordinate frame.

Lagrangian of the mechanical system can be defined as a function of the generalized coordinates:

$$L = T - U$$

where T and U respectively denote the total kinetic energy and potential energy of the system. The Lagrange equations are expressed by:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \xi_i, \quad i = 1, 2, \cdots, n$$

where ξ_i is the generalized force associated with the generalized coordinate q_i. Calculating the kinetic and potential energies and using the Lagrange equations, the dynamic model of a manipulator can be written in compact joint space model as:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau - \mathbf{J}^T(q)h \tag{3.4}$$

where B is the (n $\times$ n) symmetric positive definite inertia matrix, C$\dot{q}$ is the (n $\times$ 1) vector of Coriolis and centrifugal torques, g is the (n $\times$ 1) vector of gravitational torques, τ is the (n $\times$ 1) vector of driving torques, and h is the (6 $\times$ 1) vector of contact forces and moments at the end effector. The derivation and detailed explanation of this method is given in [2]. Even though we obtain a proper dynamic model of the manipulator through Lagrange formulation, it is very difficult to compute the energies and formulate the model for manipulators with more links and joints. So, we use the newton Euler formulation, which is computationally efficient to obtain the dynamic model.

The Newton-Euler formulation is in the form of a computationally efficient algorithm which makes it easier for computer implementation compared to Lagrange method. For robots with higher degrees of freedom, the differentiation involved in the Lagrange method makes it harder to obtain the dynamic equations. The Newton-Euler method is based on the dynamics of a single rigid body.

Consider a manipulator with n links. Consider a body fixed reference frame {i} is attached to the center of mass of each link i, i = 1, 2, . . . , n. The base frame {0} is taken at the base of the manipulator, and a frame {n+1} is taken at the end effector.

At the home configuration, we denote the configuration of the frame {j} in {i} as $M_{i,j}$ and the configuration of frame {i} in base is given as $M_{0,i} = M_i$.

The twist of joint i in base frame is S_i, $S_i = \begin{bmatrix} -\omega \times q \\ \omega \end{bmatrix}$, where ω is the axis if the joint and q is a point on the axis. A_i is the twist of the joint i expressed in the link frame. Both are related as

$$A_i = Ad_{M_i^{-1}} S_i \tag{3.5}$$

Where Ad_g is the adjoint matrix of g. Let $T_{i,j} \in$ SE(3) be the configuration of the frame {j} in {i}, at any given joint angle θ, then the configuration of the frame {i-1} in the frame {i} at given joint variable θ_i is given by:

$$T_{i,i-1} = e^{-A_i \theta_i} M_{i,i-1} \tag{3.6}$$

The velocity twist of link frame {i}, expressed in frame {i} is denoted by $V_i = (v_i, \omega_i)$.

Let $G_i \in \mathbb{R}$, denote the spatial inertia matrix of link i, expressed in link frame {i}. Let m_i be the mass of the link i and I_i be the (3x3) rotational inertia matrix of link i. Therefore,

$$G_i = \begin{bmatrix} m_i I & 0 \\ 0 & I_i \end{bmatrix} \tag{3.7}$$

where I is a (3x3) identity matrix. We now recursively calculate the accelerations and velocities of each link from base to tip. The twist of link i is the sum of the twist caused by link (i-1) and the twist obtained due to the joint rate $\dot{\theta}_i$.

$$V_i = A_i \dot{\theta}_i + Ad_{T_{i,i-1}} V_{i-1} \tag{3.8}$$

By taking the time derivatives of this equation, we obtain the accelerations of the link.

$$\dot{V}_i = A_i \ddot{\theta}_i + [Ad_{T_{i,i-1}}] \dot{V}_{i-1} + \frac{d}{dt}[Ad_{T_{i,i-1}}] V_{i-1} \tag{3.9}$$

simplifying,

$$\frac{d}{dt}[Ad_{T_{i,i-1}}] V_{i-1} = \frac{d}{dt} \begin{bmatrix} R_{i,i-1} & \hat{p} R_{i,i-1} \\ 0 & R_{i,i-1} \end{bmatrix} V_{i-1}$$

$$= \begin{bmatrix} -\omega\hat{\dot{\theta}} R_{i,i-1} & -v\hat{\dot{\theta}} R_{i,i-1} - \omega\hat{\dot{\theta}} \hat{p} R_{i,i-1} \\ 0 & -\omega\hat{\dot{\theta}} R_{i,i-1} \end{bmatrix} V_{i-1}$$

$$= \begin{bmatrix} -\widehat{\omega\dot{\theta}} & -\widehat{v\dot{\theta}} \\ 0 & -\widehat{\omega\dot{\theta}} \end{bmatrix} \begin{bmatrix} R_{i,i-1} & \hat{p} R_{i,i-1} \\ 0 & R_{i,i-1} \end{bmatrix} V_{i-1}$$

$$= [ad_V] A_i \theta_i$$

Therefore,

$$\dot{V}_i = A_i\ddot{\theta}_i + [Ad_{T_{i,i-1}}]\dot{V}_{i-1} + [ad_{V_i}]A_i\dot{\theta}_i \tag{3.10}$$

Where $ad_V = \begin{bmatrix} \hat{\omega} & \hat{v} \\ 0 & \hat{\omega} \end{bmatrix}$. This $\dot{V}_i$ is the acceleration of the link i.

Once the link twists and accelerations are determined, we calculate the joint forces and torques in a similar way by moving inward from tip.

The total wrench acting on link i is the sum of the wrench F_i obtained through joint i and the wrench applied to the link through joint (i+1), expressed in frame i.

$$Gi\dot{V}_i - ad_{V_i}^T(G_iV_i) = F_i - Ad_{T_{i+1,i}}^T(F_{i+1}); \tag{3.11}$$

The torque at the joint i is then obtained by:

$$\tau_i = F_i^T A_i \tag{3.12}$$

These recursive equations are stated below in the form of an algorithm.

Algorithm

Attach frames {1} to {n} to the center of mass of each links. Attach a frame {0} to the base and {n+1} to the end effector. Consider $M_{i,i-1}$ to be the configuration of frame {i-1} in {i} at home configuration. A_i is the twist of joint i expressed in frame {i}. G_i is the spatial inertia matrix of link i. V_0 is the velocity of the base. In this case, we consider V_0 to be zero, $V_0 = 0$. The acceleration at the base is taken as $\dot{V}_0 = \dot{v}_0, \dot{\omega}_0 =$ {-g, 0}. F_{n+1} is the force at the tip, i.e., $F_{n+1} = F_{tip}$.

Forward Iterations Given $\theta, \dot{\theta}, \ddot{\theta}$, for $i = 1$ to n do

$$T_{i,i-1} = e^{-A_i\theta_i}M_{i,i-1},$$

$$V_i = A_i\dot{\theta}_i + Ad_{T_{i,i-1}}V_{i-1},$$

$$\dot{V}_i = A_i\ddot{\theta}_i + [Ad_{T_{i,i-1}}]\dot{V}_{i-1} + [ad_{V_i}]A_i\dot{\theta}_i.$$

Backward Iterations For $i = n$ to 1 do

$$F_i = G_i\dot{V}_i - ad_{V_i}^T(G_iV_i) + Ad_{T_{i+1,i}}^T(F_{i+1})$$

$$\tau_i = F_i^T A_i$$

3.3.1 Obtaining the dynamic parameters

Now we express the Newton Euler inverse dynamics in a closed form set of dynamic equations in the form: $\tau = B(\theta)\ddot{\theta} + C(\dot{\theta}, \theta) + G$. For this the following stacked vectors are defined.

$$V = \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \in \mathbb{R}^{6n}$$

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} \in \mathbb{R}^{6n}$$

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & A_n \end{bmatrix} \in \mathbb{R}^{6n\times n}$$

$$Gi = \begin{bmatrix} G_1 & 0 & \cdots & 0 \\ 0 & G_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & G_n \end{bmatrix} \in \mathbb{R}^{6n\times 6n}$$

$$[ad_V] = \begin{bmatrix} [ad_{V_1}] & 0 & \cdots & 0 \\ 0 & [ad_{V_2}] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & [ad_{V_n}] \end{bmatrix} \in \mathbb{R}^{6n \times 6n}$$

$$[ad_{A\dot{\theta}}] = \begin{bmatrix} [ad_{A_1\dot{\theta}_1}] & 0 & \cdots & 0 \\ 0 & [ad_{A_2\dot{\theta}_2}] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & [ad_{A_n\dot{\theta}_n}] \end{bmatrix} \in \mathbb{R}^{6n \times 6n}$$

$$W = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ [Ad_{T_{21}}] & 0 & \cdots & 0 & 0 \\ 0 & [Ad_{T_{32}}] & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & [Ad_{T_{n,n-1}}] & 0 \end{bmatrix} \in \mathbb{R}^{6n \times 6n}$$

$$V_{base} = \begin{bmatrix} Ad_{T_{10}}(V_0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{6n}$$

$$\dot{V}_{base} = \begin{bmatrix} Ad_{T_{10}}(\dot{V}_0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{6n}$$

$$F_{tip} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ Ad^T_{T_{n+1,n}}(F_{n+1}) \end{bmatrix} \in \mathbb{R}^{6n}$$

With these definitions, the Newton Euler recursive inverse dynamic equations can be assembled into matrix form in the following way:

$$V = WV + A\dot{\theta} + V_{base}$$

$$\dot{V} = W\dot{V} + A\ddot{\theta} - [ad_{A\dot{\theta}}](WV + V_{base}) + \dot{V}_{base}$$

$$F = W^T F + Gi\dot{V} - [ad_V]^T + F_{tip}$$

$$\tau = A^T F$$

defining $L = (I - W)^{-1} = I + W + \cdots + W^{n-1}$,
The above equations can be now reorganized using L as follows:

$$V = L(A\dot{\theta} + V_{base})$$

$$\dot{V} = L(A\ddot{\theta} + [ad_{A\dot{\theta}}](WV + V_{base}) + \dot{V}_{base})$$

$$F = L^T(Gi\dot{V} - [ad_V]^T GiV + F_{tip})$$

$$\tau = A^T F$$

Then the Dynamic parameters are obtained by:
Mass Matrix:

$$B(\theta) = A^T.L^T.Gi.L.A \tag{3.13}$$

Coriolis Matrix:

$$C(\dot{\theta}) = -A^T.L^T.(Gi.L.[ad_{A\dot{\theta}}].W + [ad_V]^T.Gi).L.A.\dot{\theta} \tag{3.14}$$

Gravity Term

$$G = A^T.L^T.Gi.L.\dot{V}_{base} \tag{3.15}$$

3.4 Impedance control

When the robot's end effector is in contact with the environment, a suitable compliant behaviour has to be ensured. One of the most reliable strategies to manage the interaction is impedance control. The goal is to keep the contact force limited both during transient and at steady state by acting on the end-effector displacement.

Impedance control includes computing the joint control torques guaranteeing a given mechanical impedance behaviour when the end effector interacts with the environment, by feeding back end-effector motion and force variables. The impedance equation can be chosen so as to enforce an equivalent mass-damper-spring behavior for the end-effector position displacement under the contact force h_e acting on the end effector. The impedance control scheme can be imagined as a virtual spring mass damper system between the current and the desired end effector configurations as shown in Figure 3.1.

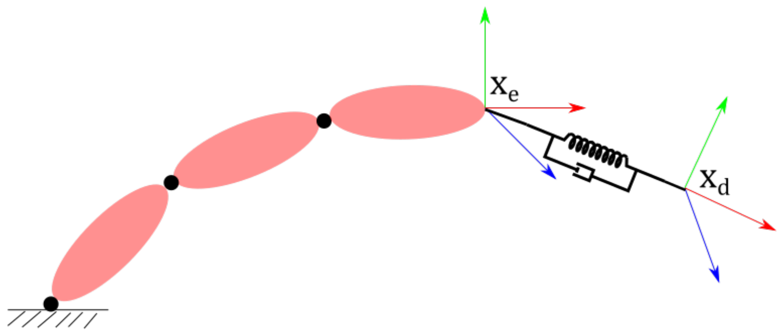


Figure 3.1: representation of impedance control scheme

To achieve the desired mass spring damper model, we impose the following model on the manipulator.

$$M_d(\ddot{x}_d - \ddot{x}_e) + K_D(\dot{x}_d - \dot{x}_e) + K_p(x_d - x_e) = h_A \tag{3.16}$$

$$M_d\ddot{\widetilde{x}} + K_D\dot{\widetilde{x}} + K_p\widetilde{x} = h_A \tag{3.17}$$

where, M_d, K_d and K_p are the Mass, damping and stiffness matrices of the manipulator respectively. In Cartesian space, these are generally 6×6 matrices. and

$$h_A = \bar{A}^{-T}h_e$$

where, $\bar{A}$ is the mapping between spatial and Analytic jacobian and in case of quaternions, $h_A = h_e$.

Here we define x as a (6x1) vector, where the first three elements of the vector represent the position of the configuration and the next three elements represent the orientation of the configuration through either Euler angles or the vector part of unit quaternions. [3]

Here, $\tilde{x} = x_d - x_e$, where x_d is the desired configuration of the end effector and x_e is the current configuration of the end effector. We use the similar notation for $\dot{\tilde{x}}$ and $\ddot{\tilde{x}}$. Therefore from the equation

$$M_d(\ddot{x}_d - \ddot{x}) + K_D(\dot{x}_d - \dot{x}) + K_P(x_d - x) = h_A$$

$$M_d\ddot{\widetilde{x}} = M_d\ddot{x}_d + K_D\dot{\widetilde{x}} + K_p\widetilde{x} - h_A$$

$$M_dJ_A\ddot{q} + M_d\dot{J}_A\dot{q} = M_d\ddot{x}_d + K_D\dot{\widetilde{x}} + K_P\widetilde{x} - h_A$$

Therefore, we choose the control input to be,

$$y = J_A^{-1}M_d^{-1}(M_d\ddot{x}_d + K_D\dot{\widetilde{x}} + K_P\widetilde{x} - M_d\dot{J}_A\dot{q} - h_A) \tag{3.18}$$

For redundant manipulators, the inverse of Jacobian does not exist. So, we use a pseudo inverse $J^\dagger$ of the Jacobian instead of the inverse. For redundant manipulators, the pseudo inverse,

$$J^{\dagger} = J^T(JJ^T)^{-1}$$

We then use this control input on the dynamic model of the manipulator to obtain the control torque as shown below:

$$u = By + C + G + J^T h_e \tag{3.19}$$

When the manipulator's end-effector is in contact with an environment, a part of the actuation torques are used to balance the torques induced at the joints by the contact forces and makes the manipulator stiff . So, by choosing y to be the control input as (7) a compliant behaviour is conferred on the manipulator.

This obtained control torque is then applied on the manipulator through joint actuators to get the joint acceleration as shown below.

$$\text{new } \ddot{q} = B^{-1}(u - C - G - J^T h_e) \tag{3.20}$$

The new joint velocities and joint angles can now be obtained by discrete time integration:

$$\text{new } \dot{q} = \dot{q} + dt.\text{new } \ddot{q}$$

$$\text{new q } = q + dt.\dot{q}$$

The new end effector position is obtained by the forward kinematics and velocity kinematics. The new force/moment vector h_e is obtained by the manipulators interaction with the environment. With these new variables, the process is continued till the desired task is achieved. The block scheme representing impedance control scheme is shown in figure 3.2

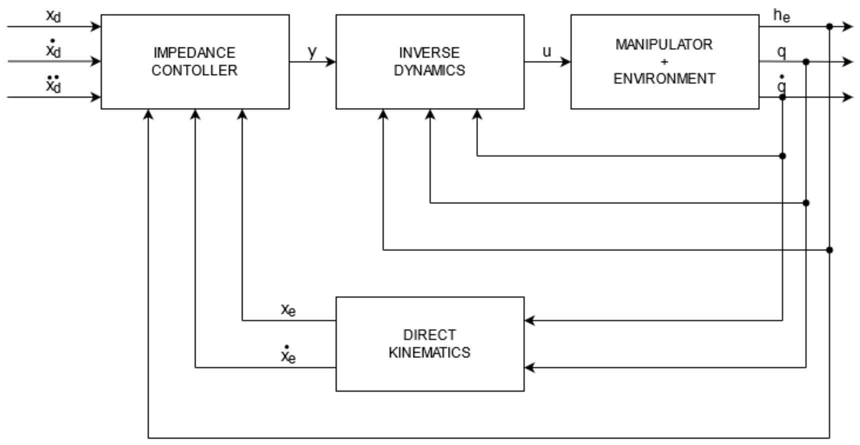


Figure 3.2: Block scheme of impedance control

3.4.1 Orientation error

As discussed in the previous sections, the orientation of a rigid body can be represented in various representations. In 3D space the position error can be easily determined by the Cartesian distance between the points. For rotation, the error depends on the choice of the orientation description. In case of Euler angles the orientation error is simply,

$$\Delta\phi_{de} = \phi_d - \phi_e$$

where ϕ_d and ϕ_e denote the Euler angles representing desired and current orientations, respectively. The unit quaternions for mutual orientation can be extracted directly from $\mathbf{R_2^1}$, or computed by composition of the Euler parameters $(q_{o1}, -\mathbf{q_1})$ and $(q_{o2}, \mathbf{q_2})$ that can be extracted from R_1^T and R_2, respectively, i.e.

$$\mathbf{q}_{21}^1 = q_{o1}\mathbf{q}_2 - q_{o2}\mathbf{q}_1 - S(\mathbf{q}_1)\mathbf{q}_2$$

Let $(q_{od}, \mathbf{q}_d)$ and $(q_{oe}, \mathbf{q}_e)$ represent the unit quaternions associated with $\mathbf{R}_d$ and $\mathbf{R}_e$, respectively. The mutual orientation can be expressed in terms of the unit quaternions $(q_{ode}, \mathbf{q}^e_{de})$. The end effector orientation is given as: [3]

$$\mathbf{q}_{de} = \mathbf{R_e}\mathbf{q}^e_{de}$$

3.4.2 Derivative of Jacobian

As discussed in the previous sections, the velocity mappings provide the relationship between the end effector velocity and the joint velocities. Acquisition of the acceleration of the end effector requires the computation of the higher order Jacobian derivatives, in this case, we need the second order derivative of the Jacobian. We know that the mapping between the end

effector velocity and the joint rates is given by

$$\dot{x} = J_A\dot{q}$$

where $\dot{q}$ is an (n x 1) vector of joint angle velocities and $\dot{x}$ is a 6 x 1 vector of velocities with the first three elements representing linear velocity of the end effector frame with respect to the base frame and the second three elements representing the rate of change of Euler angles. It can be observed that the

acceleration at the end effector can be easily obtained by differentiation of the above equation.

$$\ddot{x} = J_A\ddot{q} + \dot{J}_A\dot{q}$$

For this we need the expression for the derivative of the Analytic jacobian. We know, $J_A = A_r.A.J_s$, let $\bar{A} = A_r.A = \begin{bmatrix} I & -\hat{p} \\ 0 & J_r \end{bmatrix}$.

Therefore,

$$J_A = \bar{A}J_s$$

By taking a time derivative of the above equation, we obtain,

$$\dot{J}_A = \dot{\bar{A}}J_s + \bar{A}\dot{J}_s$$

where, $\dot{\bar{A}} = \begin{bmatrix} 0 & -\dot{\widehat{p}} \\ 0 & \dot{J}_r \end{bmatrix}$ and $\dot{J}_r = \frac{d}{dt}J_r =$

$$\begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix}$$

where,

$J_{11} = \frac{cos(y)[cos(z)cos(y)\dot{y} - sin(z)\dot{z}sin(y)] + cos(z)sin^2(y)\dot{y}}{cos^2(y)}$

$J_{11} = \frac{cos(y)[sin(y)cos(z)\dot{z} + cos(y)\dot{y}sin(z)] + sin(z)sin^2(y)\dot{y}}{cos^2(y)}$

$J_{11} = 0$

$J_{11} = -cos(z)\dot{z}$

$J_{11} = -sin(z)\dot{z}$

$J_{11} = 0$

$J_{11} = \frac{-cos(y)sin(z)\dot{z} + cos(z)sin(y)\dot{y}}{cos^2(y)}$

$J_{11} = \frac{cos(y)cos(z)\dot{z} + sin(z)sin(y)\dot{y}}{cos^2(y)}$

$J_{11} = 0$

A method to obtain the derivative of the Spatial Jacobian is given in [4]

3.5 Related Work

Impedance control is a control scheme suitable for dealing with the mechanical interaction tasks with contact process. The impedance refers to the dynamic relation between the motion variables of manipulator and the contact force. The origin of impedance control is from the observation and investigation of the co-activation effect of antagonist muscles of human arm [5] [6]. Hogan [5] stated that it is not adequate to regard muscle as simply a generator of force, but a mechanical impedance adjuster of human arm. The function of muscles is to change the impedance of the human arm to achieve elaborate manipulations.

Based on the physical analysis of human manipulations, the concept of impedance control was introduced to other research fields, and gradually derived a large number of specific implementation strategies. The implementation methods can be categorized into two main classes: software-based approach or Active impedance control and hardware-based approach called as Passive impedance control.

The hardware-based approach requires not only a proper controller but also well-designed hardware or special devices to perform particular tasks, such as RCC (Remote Center of Compliance). T. Morita and S. Sugano [7] introduced such a new joint mechanism called "Impedance Adjuster" to adjust the impedance of the robot finger joint. These are elastically compliant mechanical devices, suitably designed to achieve maximum decoupling between translation and rotation, that are interposed between the manipulator last link and the end-effector. Hogan [1] stated that it is possible to modulate the end-point impedance without feedback by exploiting the intrinsic properties of the hardware, which is essentially done in the Passive impedance control. The inconvenience of such devices is their low versatility to different operating conditions and generic interaction tasks, namely, whenever a modification of the compliant mechanical hardware is required. Laurin-Kovitz et al. [8] achieved the programmability of robot compliance by adopting the mechanical elements whose stiffness and damping are programmable. Their approach is called programmable passive impedance (PPI).

The aim of impedance control is that of achieving a suitable active impedance that can be easily modified acting on the control software so as to satisfy the requirements of different interaction tasks [9].The software-based approach is a method to implement impedance control by merely adopting control algorithms without any modification on hardware of system. The designed controller is capable of imposing the desired behavior defined by impedance on the original complicated behavior of the end-effector.

The main task in using a active impedance control is to choose the stiffness matrix for the manipulator. These stiffness matrices are usually taken as diagonal matrices with each element representing the stiffness in the respective direction [2]. Fabrizio Caccavale, Bruno Siciliano, and Luigi Villani [10] have presented a method to implement impedance control with non-diagonal stiffness matrix where the off diagonal elements are considered namely, coupling forces with orientation displacements and coupling moments with position displacements. Another method with a time varying stiffness is given by Federica Ferraguti, Cristian Secchi, Cesare Fantuzzi [11]. Even though there are multiple methods to choose the stiffness matrix for the manipulator, it is not an easy task to choose these, as it greatly affected by the surroundings.

All the above given methods and approaches paved the way for the implementation of the impedance control. In this thesis, a method to implement the impedance control with a moving goal pose is proposed, so that we need not choose these stiffness matrices for every task. Also, ScLERP is introduced in choosing these desired poses to ensure the end effector follows the desired trajectory.

Chapter 4

Problem Statement and Solution Approach

In the previous chapters, we have discussed about the dynamics of the manipulator, and the traditional impedance control algorithm. In this chapter, we will give a formal definition to the problem we are trying to solve and the approach to solve this problem.

4.1 Problem Statement

In this thesis we are trying to solve the following problem: For a manipulator to perform a task involving the manipulation of the surroundings with force exchange at the end effector and to obtain a compliant behaviour between the end effector and the environment, we chose impedance control. For successful accomplishment of the task, we have to choose the stiffness and damping matrices (($\mathbf{K_p}$) and ($\mathbf{K_d}$) respectively) of the manipulator carefully depending on the task. The selection of these elements is not an easy task, since it is highly affected by the characteristics of the environment. In this thesis, we intend to come up with a method which helps in accomplishing a given task and can be repeated to do the same task with different environments without any adjustments in the above given parameters.

In the impedance control, the key aspect that drives the manipulator to perform the task is the joint torques that moves the manipulator to the goal pose/frame in SE(3). These control torques are obtained from the control law, which is given in Equation 3.19. The key inspiration here is to obtain the control torques necessary for completing the task without the change of the properties of the robot manipulator.

Consider a task of pushing a spring loaded push button, the task in place here is to push the spring loaded push button with the end effector of the manipulator till the end of the button so that the button is activated. After the button is activated, the manipulator should retract, in turn releasing the button. The manipulator should be able to perform the task no matter what the stiffness or orientation of the button.

To demonstrate the working of the proposed method against a rotational constraint, we consider a spherical door knob with loaded torsional spring. Here the task is, the end effector should reach the knob, grip it and then rotate the door knob against the torsional stiffness, till it reaches the end of the rotation. Again, the critical task here is, the manipulator should be able to perform the task with various knobs with various stiffness's without the need to change the stiffness parameters of the manipulator.

Similarly, we take a look at the task of opening a door with a spring loaded hinge. In this task, the manipulator should first reach the knob of the door, hold it and open the door which is hinged at the end with a door closer. Here, the door should be pulled against the torsional stiffness of the door closer.

From the above examples, we see that the main problem we try to solve here is the manipulator should accomplish the task if we choose a stiffness and should be able to accomplish the task even if the task's stiffness etc. change while also making sure that the required trajectory is achieved.

4.2 Solution Approach

As discussed above, the control torque obtained from the impedance control law tries to drive the manipulator towards the desired pose $\in SE(3)$. The control torque is given by:

$$u = By + C + G + J^T h_e$$

and the control input y is given by:

$$y = J_A^{-1} M_d^{-1}(M_d \ddot{x}_d + K_D \dot{\widetilde{x}} + K_P \widetilde{x} - M_d \dot{J}_A \dot{q} - h_A)$$

We know that, for tasks such as pressing a button, the stiffer the button is, the higher the joint torques are needed to be to accomplish the task and from the above equations, we can observe that, the control torques are dependent on the control input y. Traditionally, the values of $\mathbf{K_p}$ are altered according to the nature of the environment of the task such that, for a given environment stiffness $\mathbf{K}$, according to the prescribed interaction task, large values of the elements of $\mathbf{K_p}$ are chosen for those directions along which the environment has to comply and small values of the elements of $\mathbf{K_p}$ are chosen for those directions along which the manipulator has to comply. Contrary to this, in the proposed method, we keep the $\mathbf{K_p}$ constant to an optimum value and try to change and move the $\widetilde{x}$ to alter the control torque i.e. the desired configuration $\in SE(3)$ is changed, such that the task is accomplished. As compared to choosing the values of $\mathbf{K_p}$, it is easier to choose a moving desired configuration $\in SE(3)$ such that the end effector moves along the desired trajectory, accomplishing the task.

For a task to be successfully performed using the stated method, the critical step is to choose the desired configuration, $x_d \in SE(3)$. For this we choose multiple x_d's along the trajectory which act as guiding poses along the trajectory.

This method consists of two critical steps:

- Choosing a final desired pose, x_d, to perform the required task.
- Choosing the intermediate desired (guiding) poses, to make sure that the end effector stays on the desired trajectory, satisfying all constraints.

To choose the intermediate guiding poses, we use Screw Linear Interpolation (ScLERP) to interpolate between the initial and desired poses, thus satisfying the constraints and performing the required task.

4.3 Implementation

In this section we go through the implementation of the proposed method for doing the required tasks.

4.3.1 Pressing a spring loaded push button

Consider a task of pushing a spring loaded push button using the method proposed. The task here is to push a spring loaded push button using the Baxter arm till it reaches the end and activates the button and then to retract back in turn releasing the push button. This activity can be divided into three phases.

Let the end effector be anywhere in the task space. In phase one, we want the end effector reach a point somewhere near the close proximity to the button and the z axis of the end effector aligned with the axis of the button, so that the end effector is ready to push the button. So, we chose x_d to be near the button's rest position. When the end effector reaches this position, we move to phase two. In phase two, the end effector should push the button till a point where it cannot be pushed any further. Traditionally, We can directly choose a point at the end point of the button, but, generally we do not know the end point of the button or the stiffness of the button. So, if we choose a point somewhere at the end of the button and the stiffness is non chosen properly, the required task will not be performed, the comparisons of this with the proposed method are discussed in the results section. So, we choose x_d such that the end effector moves along the axis of the button, pushing it in. It can be chosen as:

$$x_{dp} = x_{ep} + \lambda(\omega_b) \tag{4.1}$$

where, λ is any scalar value and ω_b is the button axis. x_{dp} and x_{ep} represent the desired and current end effector position respectively. For a chosen stiffness, the end effector starts moving in the direction such that it pushes the button. If the stiffness of the button is more such that the end effector cannot move any further due to its stiffness, we increase the scalar constant (λ) gradually which in turn moves the x_d gradually in the direction the axis of the button and the end effector moves towards it. This can be increased until the joint torque limits are reached. When the end effector reaches the end of the button, it senses a higher force and we move to phase three. In phase three, the end effector should retract back releasing the push button. So, we choose the desired frame x_d as the initial x_d. This method of choosing the desired configuration can be used to perform any task involving a linear motion or a prismatic joint. This approach is shown in figures 4.1, 4.2, 4.3, 4.4.

4.3.2 Rotating a spring loaded spherical door knob

Similar to the previous task, consider a task of pushing a spring loaded spherical door knob.Here, the task is to grip the door knob, and rotate it against the torsional stiffness of the knob till it cannot be rotated anymore. This can be divided to three phases.

The phase one consists of end effector reaching the knob in such a way

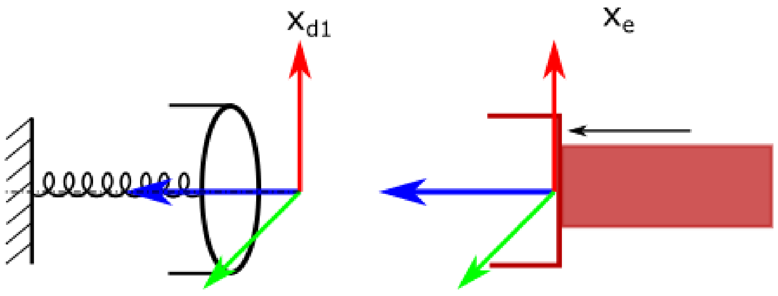


Figure 4.1: Button, Phase 1, x_d is chosen to be a configuration in a pre pressing condition.

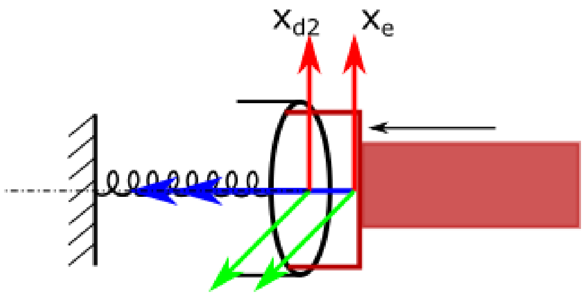


Figure 4.2: Button, Phase 2, x_d moves along the axis of the button.

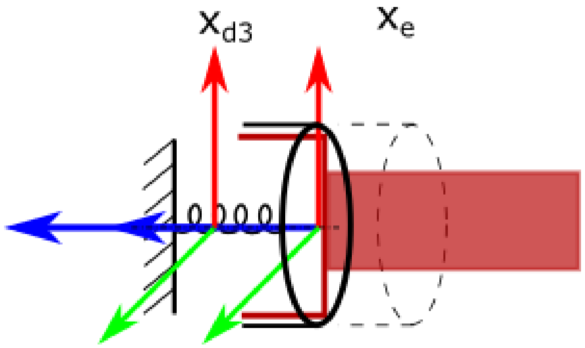


Figure 4.3: Button, Phase 2, when x_e comes to rest, λ is increased in turn moving x_d more.

so that the gripper grips the door knob. In this phase, we choose x_d so that the end effector moves to a pre-grasp pose. We move on to phase two after the end effector reaches this x_d. In phase two, the end effector should rotate the door knob about its axis. This can be considered as a

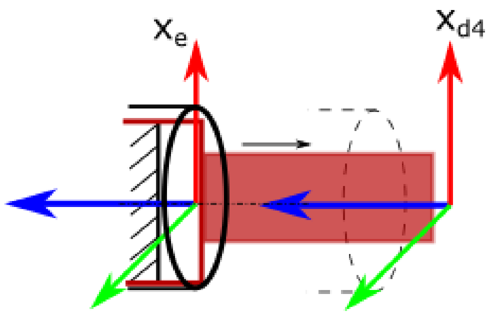


Figure 4.4: Button, Phase 3, end effector reaches the end of the button, senses higher force, and x_d is chosen to retract the end effector.

task with pure rotation constraint. In this phase, we choose a set of x_d's i.e., ($x_d(1)$,$x_d(2)$,$x_d(3)$,....,$x_d(i)$) by interpolating between the initial and a final desired configuration using ScLERP. Similar to the previous case, the final x_d can be chosen directly, but we cannot be sure that the constraints are satisfied. Also, we need to know the stiffness of the torsional spring in the knob to choose the final x_d directly. So, in the proposed method, we choose the guiding poses from the set of x_d's i.e. x_d(i) and when end effector reaches this pose, we will choose the next x_d i.e., x_d(i+1). In this way, we can make sure that the end effector satisfies the constraints. If the stiffness of the end effector is not enough to rotate the knob and remains stationary, we automatically choose the next x_d from the interpolated set. We perform this until the door knob cannot be rotated anymore and the end effector senses a higher torque. Once this state is reached, we move to phase three. In phase three, the end effector pulls back the knob while holding it i.e., keeping its orientation constant.

The implementation of this task is represented in figure 4.5, we here we can see that a number of guiding x_d's are chosen along the path using screw linear interpolation, represented by dotted lines in the figure 4.5. When the torque is not enough to turn the knob anymore, the next x_d from the guiding x_d is chosen.

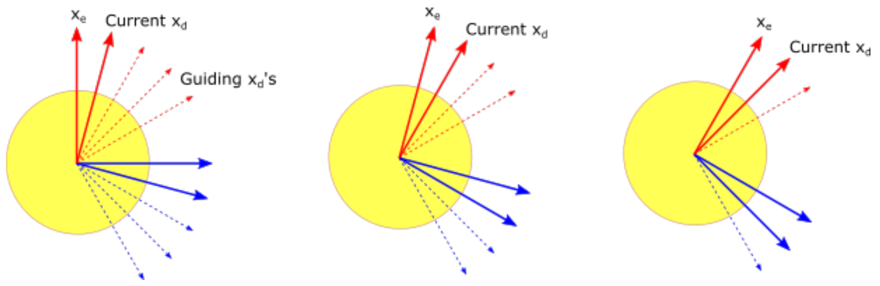


Figure 4.5: Door knob, implementation and choosing x_d

4.3.3 Opening a spring loaded door

Consider the task of opening a door loaded with a torsional spring at its hinge. In this task, the manipulator has to grasp the door at its knob and open it against the torsional spring loaded at its hinge. Similar to the previous tasks, this task can also be accomplished in two phases: 1. reaching and grasping the door and 2. opening the door.

In phase one, the end effector should reach the door knob such that it can grasp the knob. So, in phase one the desired pose x_d is chosen at a point on the door knob. Phase two involves opening the door while grasping the knob against the torque provided by the torsional spring loaded hinges. The intermediate goal poses here are chosen along the circular path created when opening the door. This circular path is discretized into a sequence of straight lines using Screw Linear Interpolation (ScLERP) with axis of the screw present at the door hinge as $(x_d(1),x_d(2),x_d(3),....,x_d(i))$. When end effector reaches the first intermediate goal pose $x_d(1)$, $x_d(2)$ is taken to be current x_d and this process is continued till the door is opened to the required angle. When the manipulator cannot pull the door any further due to its stiffness, we choose a temporary x_d, $x_d(temp)$ further along the direction of the discretized path and stop when the end effector reaches the x_d on the discretized path. The choosing of this $x_d(temp)$ can be done using Screw

Linear Interpolation using:

$$\mathbf{C}(\tau) = \mathbf{A} \otimes (\mathbf{A}^* \otimes \mathbf{B})^{\tau} \tag{4.2}$$

where, λ is any scalar value. x_{dp} and x_{ep} represent the desired and current end effector position respectively. This λ is increased gradually when the end effector comes to rest and does not move anymore. Like in the case of the push button, this constant can be increased till the joint torque limits are reached. This is continued till the door is opened to the required angle and then terminated.

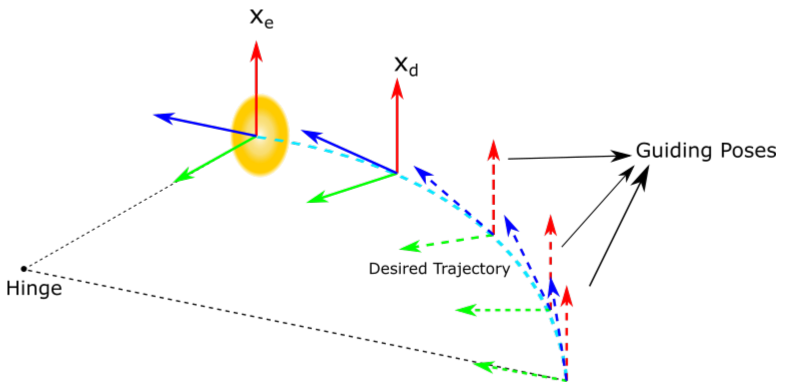


Figure 4.6: Door, implementation, the x_d's are chosen along the path of the opening door using Screw Linear Interpolation

The implementation of this task is shown in figures 4.6, 4.7, 4.8. The cyan dotted line in the figures show the desired trajectory. In figure 4.6, the x_d is chosen as a configuration on the path of the opening door using ScLERP. When the end effector cannot move anymore, we choose a temporary x_d along the direction of the vector between x_e and x_d shown in figure 4.7. When the x_e reaches the original x_d we choose the next x_d along the path, shown in figure 4.8.

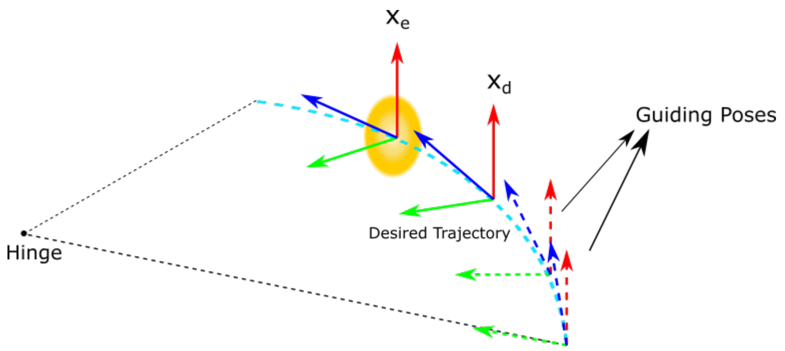


Figure 4.7: Door , implementation, when the end effector cannot move anymore, we move the x_d along the vector between x_e and x_d.

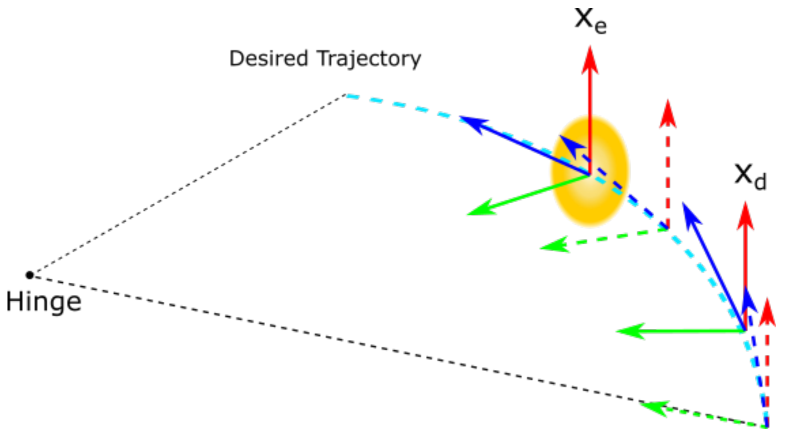


Figure 4.8: Door , Implementation, Once the current x_d is reached, the next x_d becomes the current x_d

4.3.4 Algorithm for implementation

The algorithm to implement the impedance control is given below:

The algorithm is first initialized using $\ddot{q}$, $\dot{q}$, q, $\dot{x_d}$, $\ddot{x_d}$, g, dt (time step) and force at the end effector h_e. The current end effector configuration can be obtained from forward kinematics and Velocity kinematics using q, $\dot{q}$ respectively shown in step 2.

The dynamic parameters B, C, G are obtained using the joint angles, velocities and accelerations as shown in 3. The desired pose x_d is now chosen using ScLERP as shown in Eq (Eq 4.1, Eq 4.2).

The control input y and control torque are now obtained using the Algorithm 3. Using the obtained control torque, the new joint accelerations are obtained using the equation shown in step 8.

Using the discrete time integration, the new joint velocity $\dot{q}$ and new joint angles q are obtained as shown in step 10. The force at the end effector is now obtained with the end effector interaction with the environment. Now the old q, $\dot{q}$, $\ddot{q}$ are replaced with the new q, new $\dot{q}$ and new $\ddot{q}$ respectively and this process is continued till the convergence is reached.

Algorithm 2: Implementation of Impedance control

```
1  Initialize $\ddot{q}$, $\dot{q}$, $q$ , $\dot{x}_d$, $\ddot{x}_d$, g, dt (time step) and $h_e$ is the force at the
     end effector.
2  Obtain $x_e$ from $x_e = FK(q)$, and $\dot{x}_e = J_A\dot{q}$.
3  for $i =$ ti:dt:tf do
4      Obtain the dynamic parameters B, C, G using $\ddot{q}$, $\dot{q}$, $q$.
5      Obtain the dynamic parameters with noise $B_{noisy}, C_{noisy}, G_{noisy}$
         from the noisy $q$ and $\dot{q}$ sensor data.
6      Choose $x_d$ according to the proposed method and the required
         task (Eq 4.1, Eq 4.2).
7      Obtain control input y and control torque u from
         FUNCTION(Impedance Control)
8      Using the obtained control torque, obtain the new joint
         accelerations $\ddot{q}$.

             $\text{new } \ddot{q} = B^{-1}(u - C - G - J^T h_e)$

9      new $\dot{q} \longleftarrow \dot{q} + dt(\text{new } \ddot{q})$ ;
         new $q \longleftarrow q + dt(\dot{q})$ ;
10     $x_e \longleftarrow FK(\text{new } q)$;
         $\dot{x}_e \longleftarrow J_A.(\text{new } \dot{q})$;
11     Obtain force $h_e$ at the end effector from interaction with the
         environment.
12     $q = \text{new } q$;
         $\dot{q} \longleftarrow \text{new } \dot{q}$;
         $\ddot{q} \longleftarrow \text{new } \ddot{q}$;
13     if convergence is achieved then
14         stop;
15     end
16 end
```

```
Algorithm 3: FUNCTION: Impedance Control
  Input: $M_D$,$K_D$,$K_P$,$\ddot{x}$, $\dot{\widetilde{x}}$, $\widetilde{x}$ ,$h_A$
  Output: Control input, y and Control Torque u
1
      $$y = J_A^{\dagger} M_d^{-1}(M_d\ddot{x}_d + K_D\dot{\widetilde{x}} + K_P\widetilde{x} - M_d\dot{J}_A\dot{q} - h_A)$$
2 where,
      $$h_A = \bar{A}^{-T}(h_e + h_{noise})$$
    and
      $$J^{\dagger} = J^T(JJ^T)^{-1}$$
    ,$\bar{A}$ is the mapping between spatial and Analytic jacobian.
3 Using the control input, the control torque can be obtained by:
      $$u = B_{noisy}y + C_{noisy} + G_{noisy} + J^T(h_e + h_{noise})$$
    where, $B_{noisy}, C_{noisy}, G_{noisy}$ are the dynamic parameters of the
    manipulator model obtained from the noisy $q$ and $\dot{q}$ sensor data.
```

Chapter 5

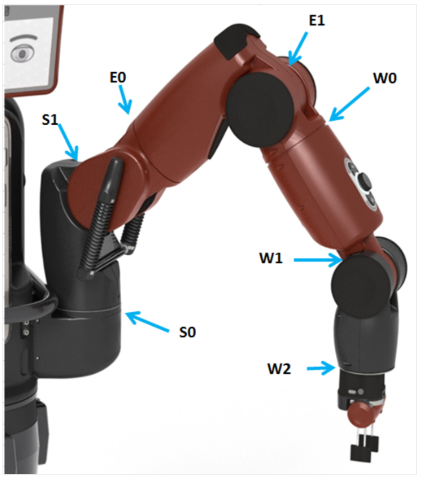
Simulation and Results

This chapter of the thesis discusses about the experimental setup used to demonstrate the given scheme. We use simulations in MATLAB for all the experiments done. A Baxter robot is used as the reference robot for simulation. A brief description of the Baxter robot, the specifications, parameters are given in this chapter. The simulation environment and the results of the experiments are also discussed in the later part of this chapter.

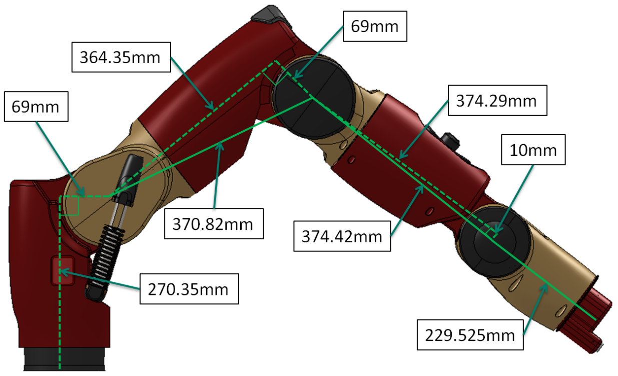
5.1 Baxter Robot

The simulation performed in this thesis is based on Baxter robot[12] by Rethink Robotics. The Baxter is a humanoid, anthropomorphic robot with two seven degree of freedom arms including force, position, and torque sensing and control at every joint. The Baxter SDK allows users to develop custom software for Baxter. The seven DOF arms provide kinematic redundancy which greatly improves manipulability.

The joints are named from the base to end in the following order : S0, S1, E0, E1, W0, W1, W2. The labelled figure of the baxter arm is shown below (S=Shoulder, E=Elbow, W=Wrist). The link lengths of the baxter arm used are also shown in the figure 5.1.



(a) Joint names



(b) Link Lengths

Figure 5.1: Parameters of the Baxter arm

For obtaining the dynamic model of the Baxter, we use the following parameters of the robot arm.

Hardware specifications:

The hardware specifications of the Baxter robot arm are given below [13]. The maximum payload for the Baxter is 5lbs/2.2 Kg.

Link Masses

The Masses of links measured in Kgs, numbered from base to end effector are below Table 5.1.

Link	Mass(Kg)
1	5.700440
2	3.226980
3	4.312720
4	2.072060
5	2.246650
6	1.609790
7	0.350930

Table 5.1: Link Masses

Center of Mass

The center of mass of each link, measured in mm, referred in the reference frame of the link are shown below in table 5.2:

Link	$\bar{x}$	$\bar{y}$	$\bar{z}$
1	0.0178	0.0009	0.1913
2	0.0684	0.0027	-0.0053
3	-0.0028	0.0013	0.1809
4	0.0261	0.0016	-0.0112
5	-0.0017	0.0046	0.1395
6	0.0604	0.0070	0.0060
7	0.0020	0.0013	0.0186

Table 5.2: Link Center of Mass (m)

Inertia

The rotational inertial of each link in $kg.m^2$, expressed in the link frame, Table 5.3

Link	I_{xx}	I_{yy}	I_{zz}	I_{yz}	I_{xz}	I_{xy}
1	0.2556	0.2480	0.0378	-0.0002	-0.0133	0.0000
2	0.0119	0.0431	0.0359	-0.0001	0.0032	-0.0009
3	0.1677	0.1695	0.0125	0.0001	0.0061	0.0003
4	0.0074	0.0149	0.0107	-0.0002	0.0014	0.0003
5	0.0605	0.0605	0.0038	-0.0021	0.0007	0.0002
6	0.0040	0.0129	0.0115	0.0001	-0.0008	-0.0011
7	0.03742e-3	0.3910e-3	0.3093e-3	-0.0133e-3	-0.0145e-3	0.0049e-3

Table 5.3: Inertia of the links ($kg.m^2$)

Peak Torque

The peak torque specification refers to the maximum amount of torque that can be applied to each joint. The joint torque limits for baxter arm are shown in table 5.4

Joint	Peak Torque (Nm)
S0, S1, E0, E1	50
W0, W1, W2	15

Table 5.4: Peak Torque per joint

5.2 Experiments and Results

We conduct the following experiments in the simulated environment in MATLAB. For these experiments, we use unit quaternion parameterization to represent the orientation. We choose this to avoid the representation singularities which we get when using Euler angles.

5.2.1 Pushing a Spring loaded push button

As stated in the previous sections, the task of pushing a spring loaded button is considered. We consider a button which can be found in the house hold to demonstrate this task e.g., a button on a keyboard, power button on a CPU etc. For this we consider a circular button with 1cm diameter and 5mm depth. We choose several stiffness's for the spring loaded in the button while we perform the simulation.

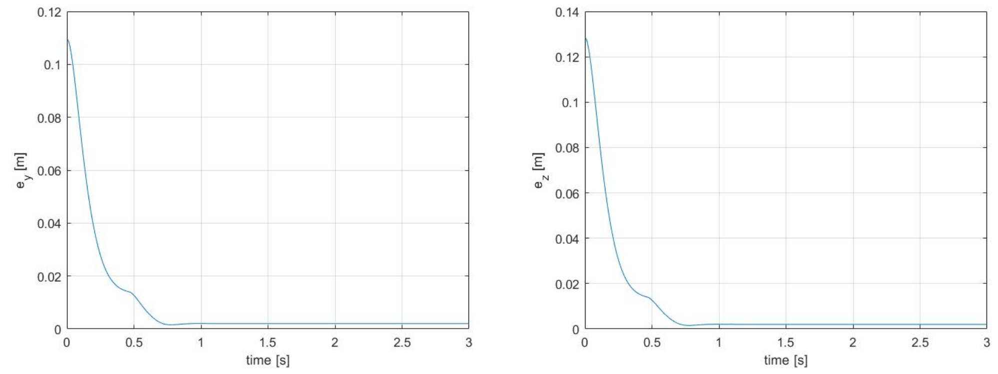
We first perform the task with a stationary desired configuration, x_d and later we use the proposed method with the moving x_d and compare the results. We also use different parameterizations i.e., Euler angles and unit quaternions for the orientation of the end effector and desired frames and the results are compared.

We choose $\mathbf{K_p}$ as a (6 x 6) diagonal matrix with all diagonal elements as 150N/m and $\mathbf{K_d}$ is a (6 x 6) diagonal matrix with all diagonal elements as 22.5 Ns/m. We keep these values constant throughout all the experiments.

Stationary desired pose

As stated in the solution approach section, this task is divided into three phases. First we try to accomplish the task using a stationary desired pose at placed at the end of the button. So, the desired pose x_d is now place at the end of the button, assuming we know the depth of the button. Here, for simulation, we choose the stiffness of the button to be 200N/m. The stiffness for the manipulator is maintained constant through out our experiments as this is the main goal of the experiments.

Trial 1: We choose the stiffness of the button to be 200N/m and the desired pose at the end of the button. The following results are obtained from the experiment.



(a) error in y vs time **(b)** error in z vs time

Figure 5.2: Button, Error in position vs time, stationary x_d, Failed trial

The above Figure 5.2 shows the error in the position of the end effector with respect to the end of the button and it can be observed that the end effector fails to reach the end of the button, in turn failing to activate the button.

The joint torques plot for the above trial are obtained as shown in figure 5.3

Trial 2: As the manipulator couldn't reach the goal pose in the previous trial, we choose to move the x_d more along the direction of axis and choose a frame beyond the end of the button, here we choose the origin of the goal pose to be a point 5cm moved along the direction of the buttons axis. The manipulator retracts back as it reaches the end of the button using position feedback.

The figures 5.4 and 5.5 shows the simulation of this trial.

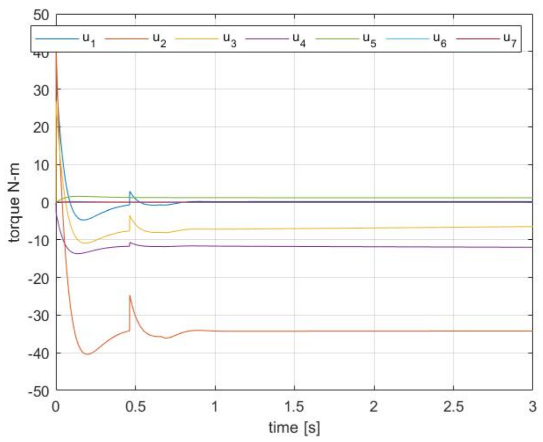
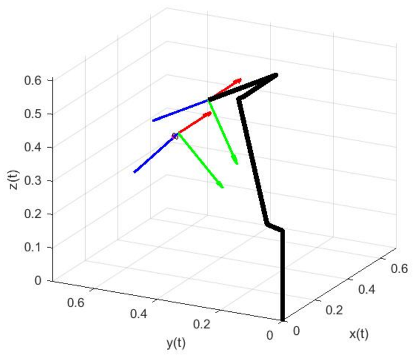
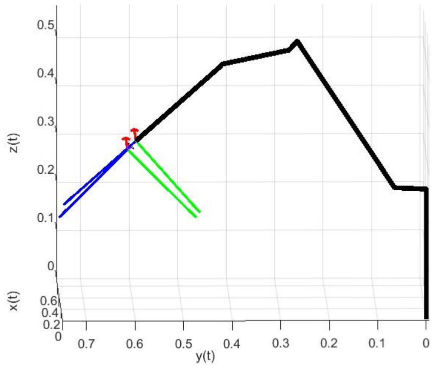


Figure 5.3: Button, Joint torques vs time for trial 1, stationary x_d, Failed trial



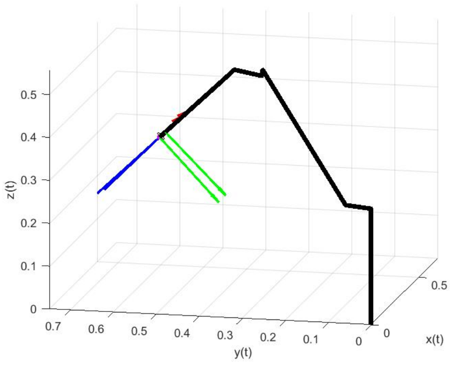
(a) Initial configuration



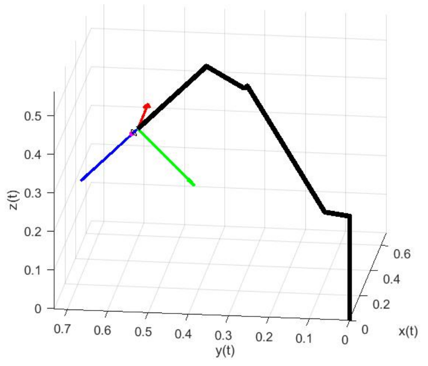
(b) x_d is chosen at the end of the button , beginning of phase 2

Figure 5.4: Button, Error in position vs time for trial 2, stationary x_d

In the error plots shown in figure 5.6, the figures show the error in y and z plotted vs time. In the plots, region between red and blue dotted vertical lines represent phase two. The curve before the red vertical line shows the approach of the end effector toward the x_d in phase one. The pressing of the button is done between the red and the green lines. As the error becomes zero, the phase three starts, i.e. the curve after the blue dotted line. The

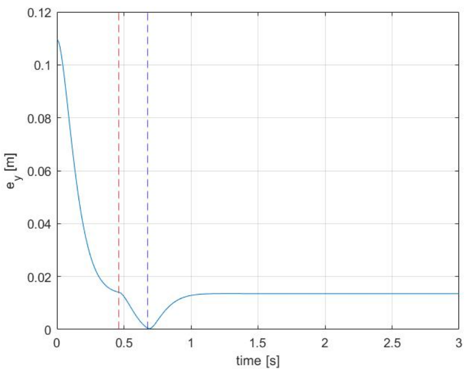


(a) As the phase 3 is initiated, the x_d is chosen as in phase 1

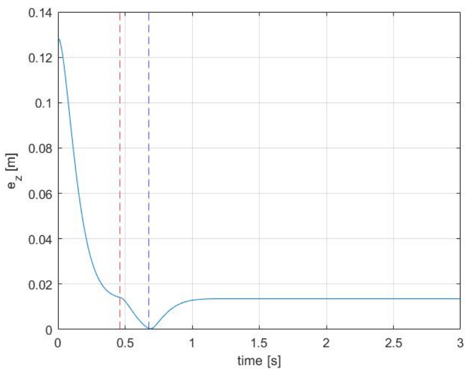


(b) Button, stationary x_d, Final configuration

Figure 5.5: Button, Error in position vs time for trial 2, stationary x_d



(a) error in y vs time

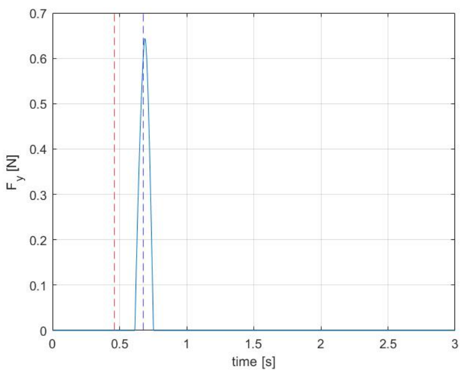


(b) error in z vs time

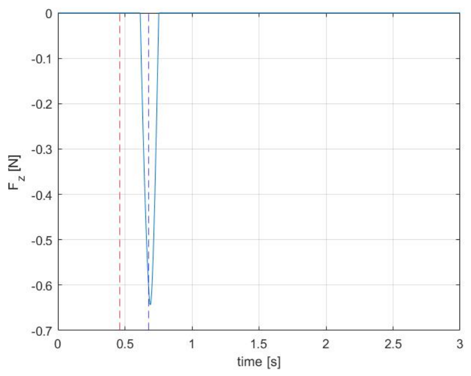
Figure 5.6: Button, Error in position vs time for trial 2, stationary x_d

force vs time plots are as shown in figure 5.7.

Similar to the error plots, the region between the red and blue dotted vertical lines represent phase 2. In phase two, the peak force is observed at the end of the phase two, i.e., the button is totally pressed and the release of the button is shown after the blue dotted line. The joint torques of the manipulator vs time plot is shown below.



(a) Force in y vs time



(b) force in z vs time

Figure 5.7: Button, Force vs time for trial 2, stationary x_d

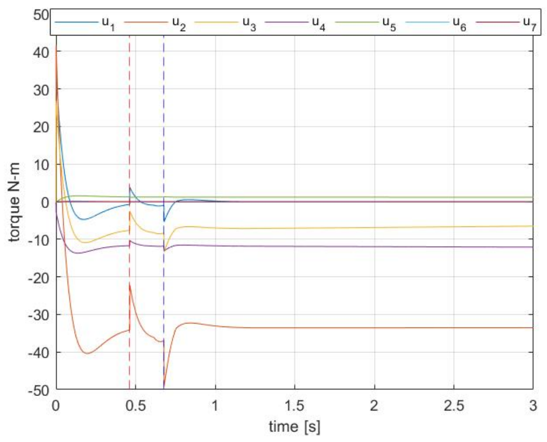
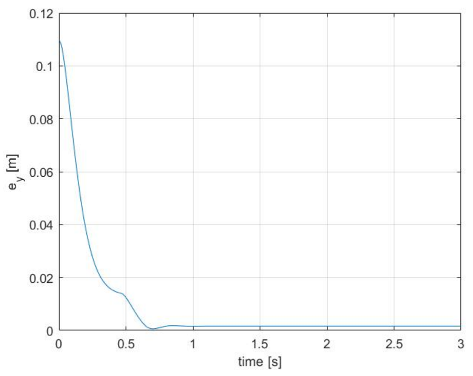


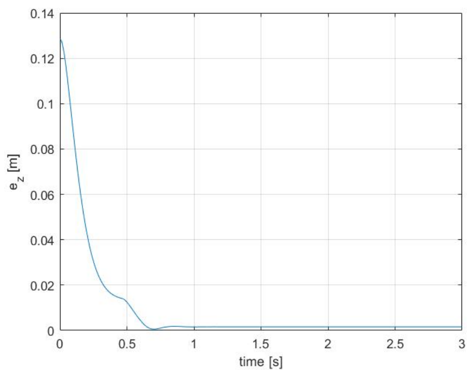
Figure 5.8: Button, Joint torques vs time for trial 2, stationary x_d

Trial 3: As the task is achieved using this method, this method is tested on button with higher stiffness. So, we choose another button with higher stiffness i.e. 400N/m. The results are as follows:

From the error plots shown in figure 5.9, we can observe that the end effector does not reach the end of the button, in turn not activating it. From the above results, it can be observed that, if the stiffness of the button is altered, the stiffness of the manipulator has to be tuned in order to perform



(a) error in y vs time



(b) error in z vs time

Figure 5.9: Button, Error in position vs time for trial 3, stationary x_d, Failed trial

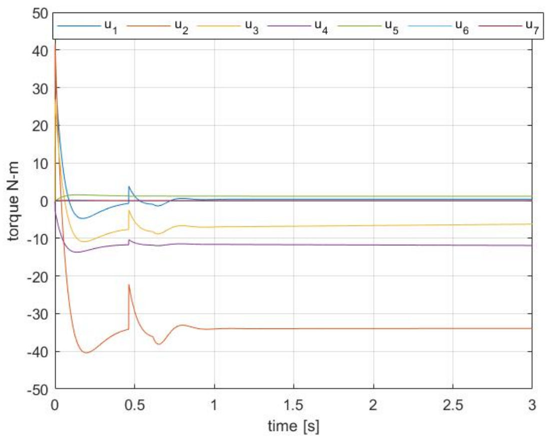


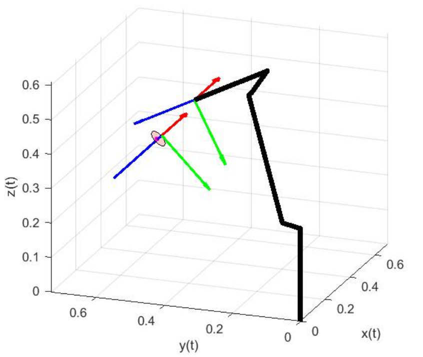
Figure 5.10: Button, Joint torques vs time for trial 3, stationary x_d, failed trial

the task and this tuning differs from button to button and is not feasible to the user in real life. Also, the stiffness or the properties of the button i.e., depth etc. are unknown, performing this task on various buttons would be a problem. So, now we try to perform the same task using the method proposed in this thesis.

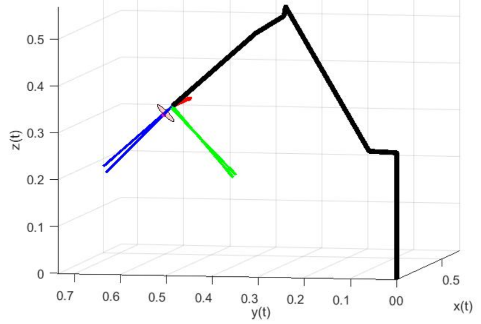
Moving Desired pose

As stated above, we choose a constant stiffness matrix for the manipulator throughout the experiments. In this case, we choose the desired pose x_d as discussed in chapter 4.

Trial 1: The stiffness of the button is chosen as 200N/m and the desired pose is selected as discussed in chapter 4. The following results are obtained from the experiment. Figures 5.11 and 5.12 show the simulation for this trial.



(a) Initiation of phase 1

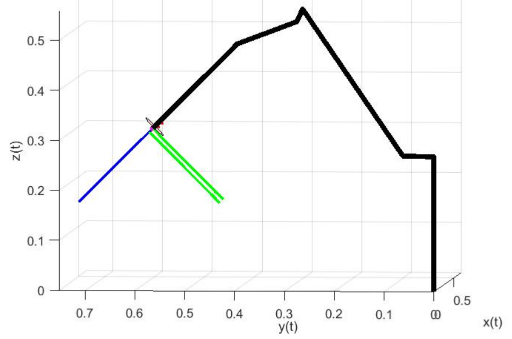


(b) As the end effector reaches x_d in phase 1, the origin of the frame x_d moves along the direction of the axis of the button

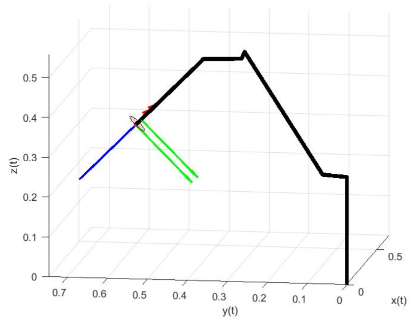
Figure 5.11: Button, moving x_d, Simulation, Trial 1

The plots are shown below. Similar to the previous section, the region between the red and blue vertical dotted lines in the plots represent phase 2. In the error plots 5.13, the initial curve before red line shows the approach of the end effector towards the initial x_d. In phase two, the button is totally pressed and the error becomes zero at the end of the phase two, shown by the blue dotted vertical line. The release is shown by the curve after the blue line.

In the force plots ,fig 5.14, it can be observed that, when a higher force is observed, the end effector retracts back and the peak force is observed at the end of phase 2 (blue line).

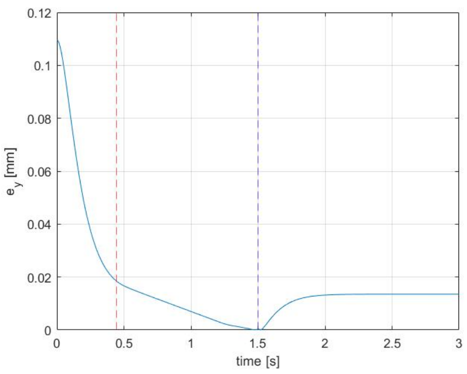


(a) If the x_e comes to rest without reaching x_d, λ is increased

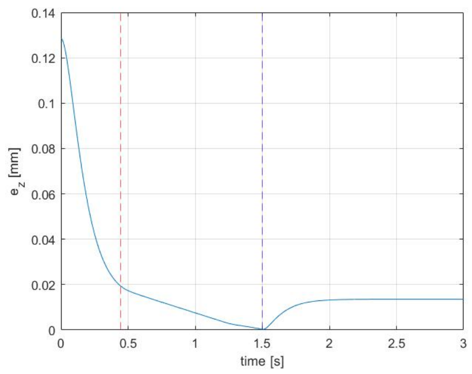


(b) Retraction

Figure 5.12: Button, moving x_d, Simulation, Trial 1



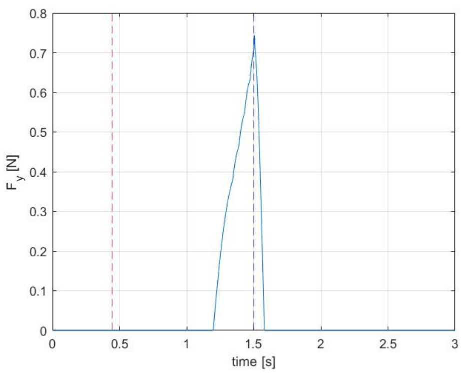
(a) error in y vs time



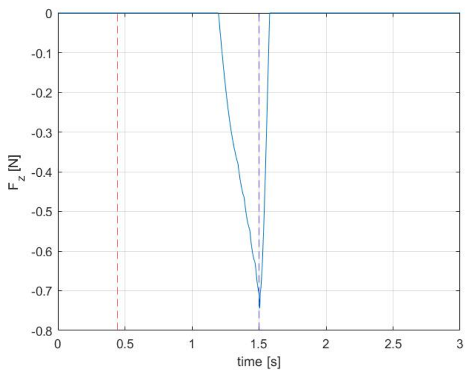
(b) error in z vs time

Figure 5.13: Button, Error in position vs time for trial 1, moving x_d, K = 200 N/m

Figure 5.15 shows the plot of joint torques vs time. Similar to the previous case, we observe a change in the torques at the end of phase 1. The small bumps in the torque plot near the end of phase three is due to the change of λ, which in turn increases the joint torques, moving the origin of x_d further along the axis of the button when the x_e comes to rest. The region after the green line shows the retraction of the end effector.



(a) Force in y vs time



(b) force in z vs time

Figure 5.14: Button, Force vs time for trial 1, Moving x_d

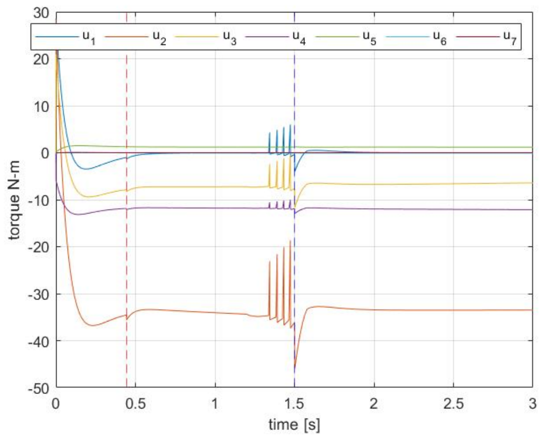
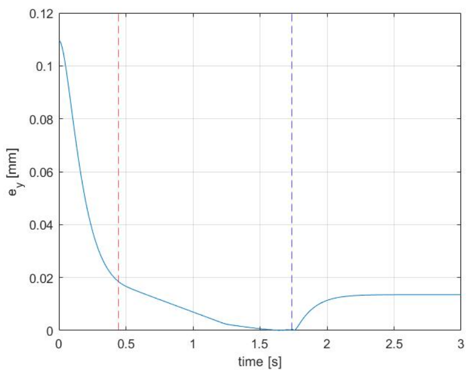


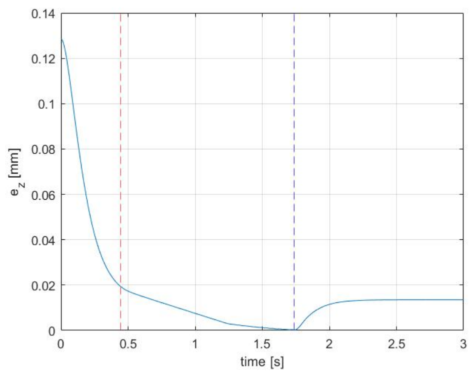
Figure 5.15: Button, Joint torques vs time for trial 1, moving x_d, K = 200N/m

Trial 2: As the previous trial turned out to be successful we increase the stiffness of the button to show that the proposed method would work at various stiffness's. Now, the stiffness of the button is chosen to be 400N/m.

Again, here the curve between the two vertical lines represent phase two. From the error plots (Figure 5.16), we can observe that the error becomes zero even when the stiffness of the button is increased. This shows that this proposed method with the moving desired pose accomplishes the task even



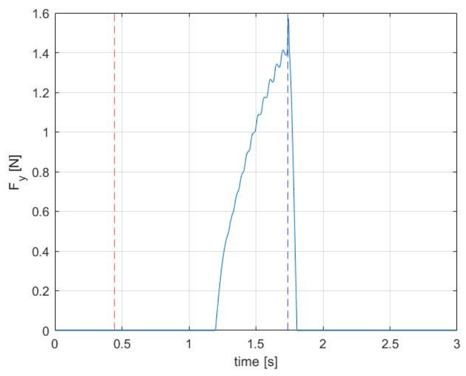
(a) error in y vs time



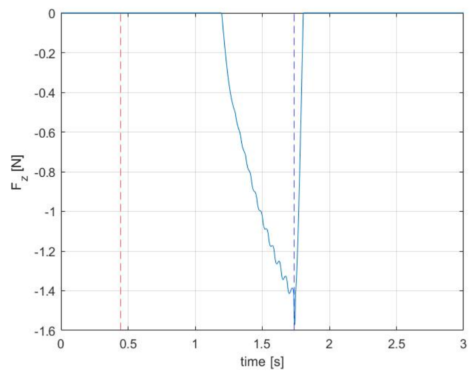
(b) error in z vs time

Figure 5.16: Button, Error in position vs time for trial 2, moving x_d, K = 400 N/m

though the environment variables change. Similarly, the force plots are as follows, figure 5.17:



(a) Force in y vs time



(b) force in z vs time

Figure 5.17: Button, Force vs time for trial 2, Moving x_d

From the force and error plots, it can be observed that the required task is performed i.e., the button is pushed till the end and activated. We can observe from the torque plots (figure 5.18) that this is achieved while the joint torques are still within the limits.

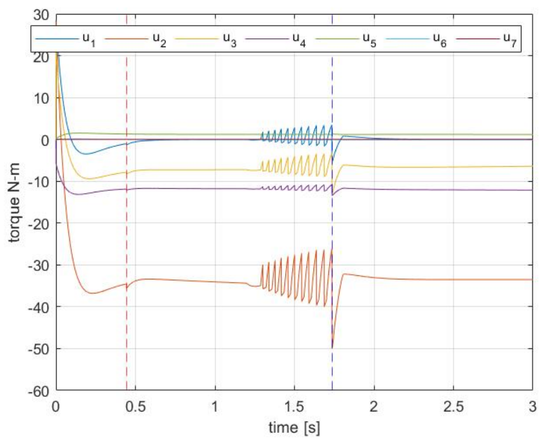


Figure 5.18: Button, Joint torques vs time for trial 2, moving x_d, K = 400N/m

5.2.2 Rotating a Spring loaded spherical door knob

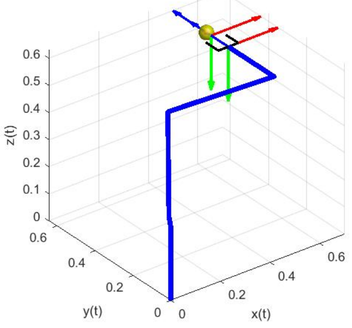
We now move on to a task where where have to manipulate an object with a revolute constraint. So, we use a torsional spring loaded spherical door knob. Similar to the push button task, we compare results of performing the task with a stationary desired configuration and with a moving desired configuration.

Stationary desired configuration

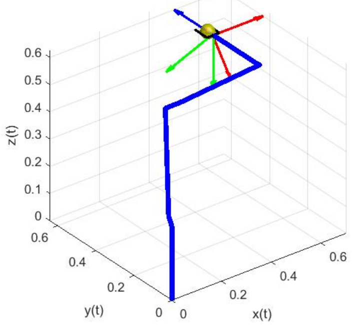
For this task, we consider a knob whose maximum rotation angle is 1.4 radians $\approx 80^o$. We consider the stiffness of the spring loaded in the knob to be 10 Nm/rad. For comparison we first perform the task using a stationary desired configuration.

Trial 1: For this trial, we use a desired configuration as a frame which is rotated about 1.2 radians from the current end effector configuration, x_e and we rotate the end effector about the axis of the door knob. When the error in the orientation with respect to the desired pose becomes zero, the knob is pulled back with the end effector. The stiffness of the knob in this trial is considered to be 10Nm/rad. The simulation is shown in figures 5.19

and 5.20.



(a) Initial Configuration, x_d is chosen to grasp the knob



(b) Phase 2, x_d is chosen as a frame rotated 1.2 radians from the initial pose

Figure 5.19: Knob, Stationary x_d, Simulation

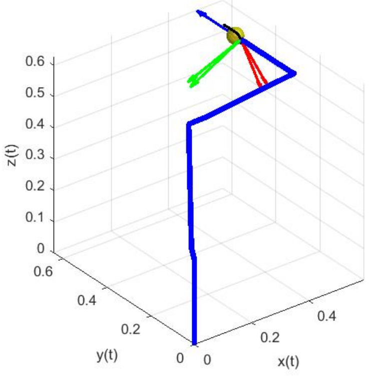
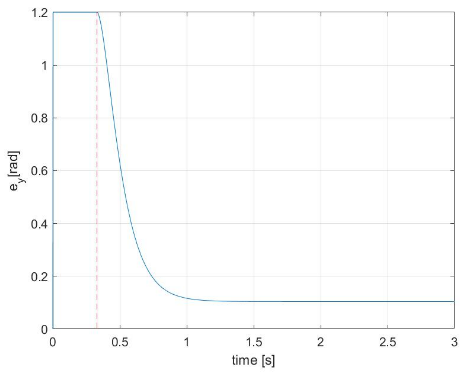


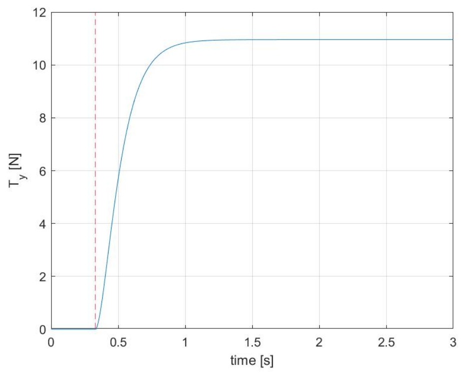
Figure 5.20: Knob, Stationary x_d, Final configuration

The results from the experiment are shown in figure 5.21 and figure 5.22:

For simplicity in simulation, the axis of the knob is taken parallel to y axis. The red dotted vertical line in the plots 5.21 represents the start of phase 2. As we can observe, if we choose the angle to which the knob should be rotated exactly, in this case, 1.2 radians and for given stiffness, the end



(a) Error in knob angle vs time



(b) Torque about y vs time

Figure 5.21: Knob, Stationary x_d, Error in knob angle and torque about y vs time

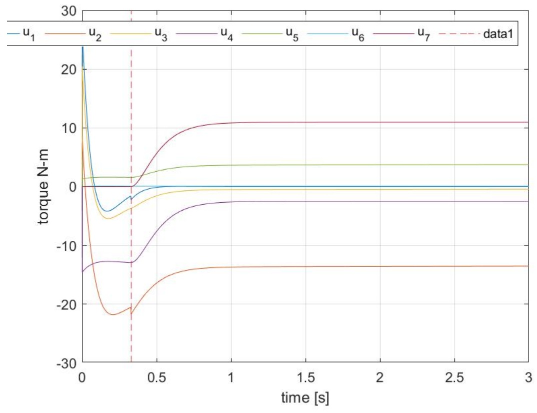
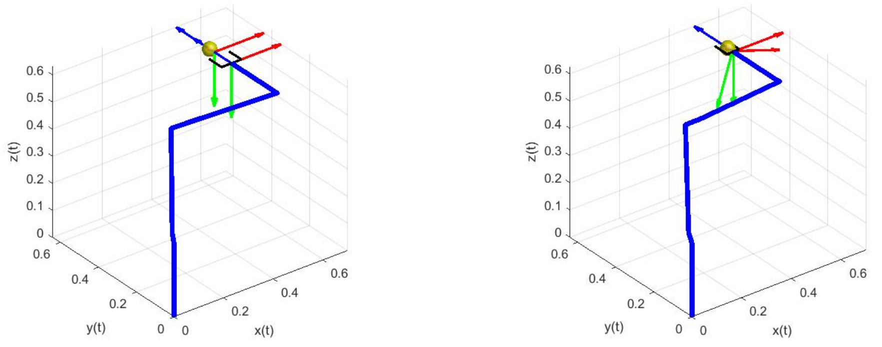


Figure 5.22: Knob, Stationary x_d, Joint torques vs time

effector could not rotate the knob till the end, which can be seen in plot 5.21(a). From these, we can consider that the trial is unsuccessful. So, we now perform this task using the proposed method.

Moving desired configuration

As it can be observed, the previous trial was unsuccessful. Now the same task is performed using the proposed method. A set of guiding x_d's are obtained using ScLERP. When the end effector rotates the knob till the end i.e., till it cannot be rotated anymore and senses high torque, the door is pulled back. The simulation is shown in figures 5.23 and 5.24:

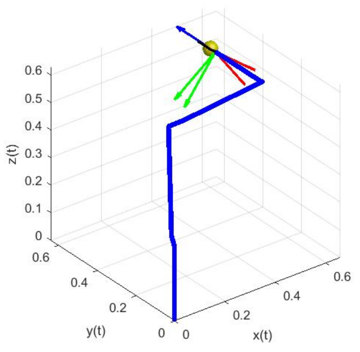


(a) Initial Configuration, x_d is chosen to grasp the knob

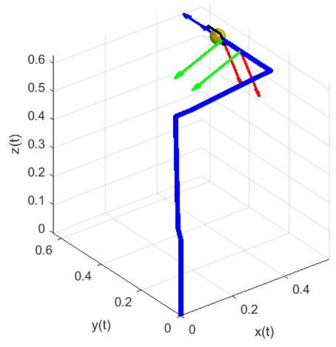
(b) Initialization of Phase 2, x_d be using ScLERP

Figure 5.23: Knob, moving x_d, Simulation

The results from the experiment are as follows: Figure 5.25(a) represents the error in knob angle vs time. It can be observed from the plot that the required angle of rotation for the knob is achieved for the same stiffness used in the previous trial. The region between red and blue vertical dotted lines represent phase 2. We can observe that the error becomes zero at the end of phase 2. When the knob is rotated till the end, a higher torque is observed, which can be seen in figure 5.25(b) and when this point is reached, the knob is pulled back. The plot of the joint torques can be seen in 5.26(b). From the obtained results, it is observed that the required task is performed i.e., the knob is rotated till the end and this task is achieved while the joint torques are within the limits.

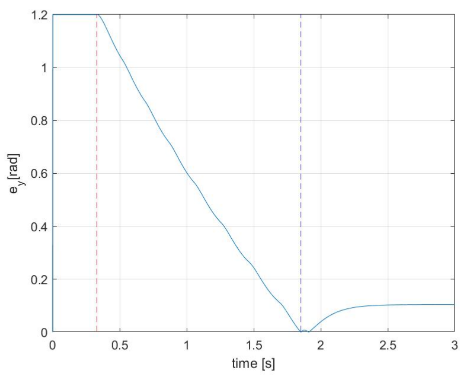


(a) Phase 2, when x_e reaches current x_d, the next x_d from the interpolation is chosen to be current x_d

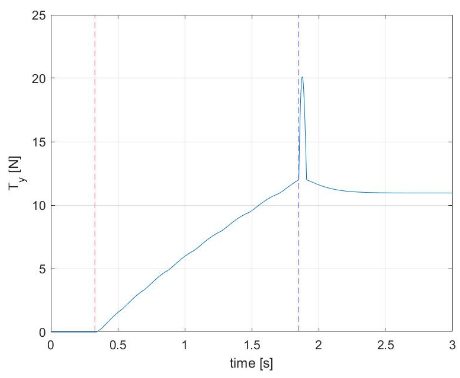


(b) Initialization of Phase 3, x_d is chosen to pull the door back holding the knob in its orientation

Figure 5.24: Knob, moving x_d, Simulation



(a) error in knob angle vs time

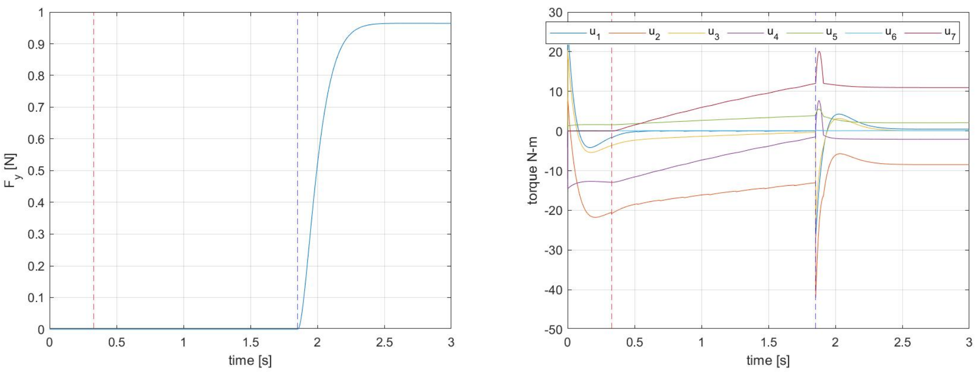


(b) torque in y vs time

Figure 5.25: Knob, Moving x_d, error in knob angle and torque about y vs time

5.2.3 Opening a door

In this experiment, the task of opening a door which is loaded with a spring at the hinge, while holding the knob, is performed. We consider a normal interior door which can be found in any household. The distance from the door knob to the hinge is taken as 70cm. According to ADA, "Interior doors



(a) Force in y vs time **(b)** Joint torques vs time

Figure 5.26: Knob, Moving x_d, force in y and joint torques vs time

should require no more than 5 lbs. of force to open"[14]. Taking this into consideration, for the spring loaded at the hinge of the door, we consider the stiffness to be 10Nm/rad.

For this task, the trial is considered to be successful if the end effector successfully opens the door till the required angle i.e., it should be staying on the required trajectory while holding the door.

Stationary desired configuration, x_d

For this trial, the desired configuration is chosen to be a stationary frame at where the door is opened for a 30^o. The results are as follows:

The achieved trajectory by choosing the x_d as a pose obtained by rotating the door 30^o is shown in Figure 5.27, xy view of the task being done. It can be observed that the end effector does not follow the required trajectory when we choose the desired configuration as given above. Also, the end effector could not reach the x_d even when it is not on the desired trajectory. Therefore, the required constraints are not met if the goal pose is directly set to the end pose. The joint torques obtained are as follows:

From the figure 5.28, we can observe that by choosing a stationary frame

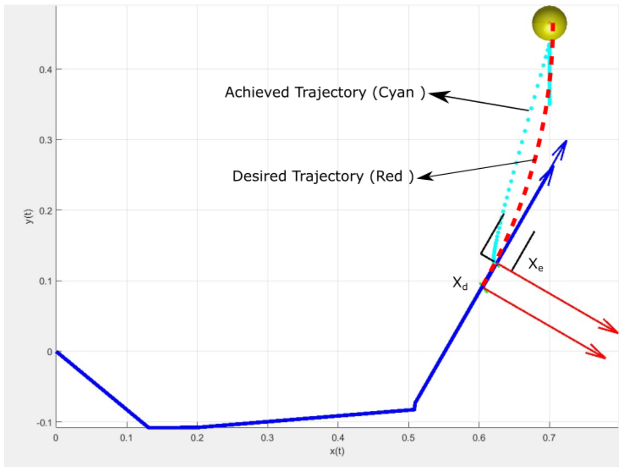


Figure 5.27: Door, Achieved vs desired trajectory, stationary x_d, failed trial

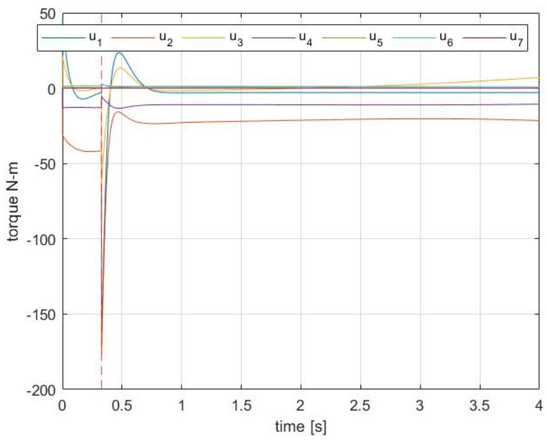
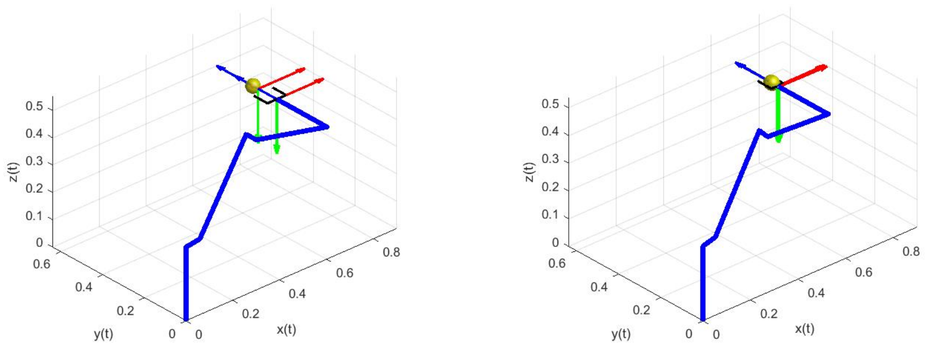


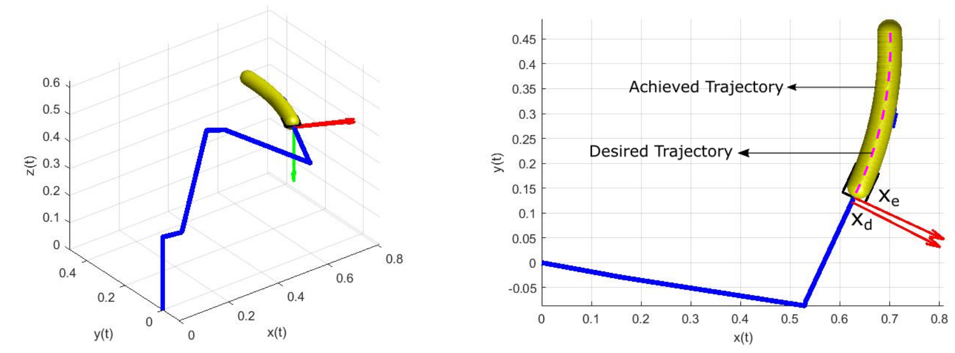
Figure 5.28: Door, Joint Torques vs Time, stationary x_d, failed trial

at the end, the achieved joint torques are very high, beyond the joint torque limits. As it can be clearly inferred from these observations that this method is not feasible for this task, we move on to perform the simulation of performing this task in the proposed method.



(a) Initialization of Phase 1 **(b)** Initialization of Phase 2

Figure 5.29: Door, Moving x_d, Simulation



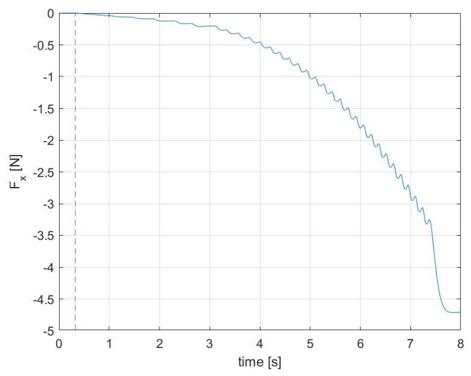
(a) Door, Moving x_d Final Configuration **(b)** Door, Moving x_d, Desired vs achieved Trajectory, successful trial

Figure 5.30: Door, Moving x_d, Simulation and Trajectory

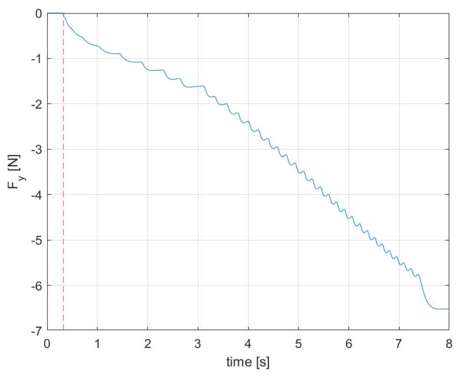
Moving desired configuration, x_d

For this task, the desired configuration x_d is chosen along the path of the movement of the door knob when the door is being opened. The configurations along this path are obtained from Screw Linear Interpolation. For this experiment, we consider the task successful if the door is opened for about 30^o. When the chosen stiffness of the manipulator is not enough to perform the task, thus keeping the end effector in rest. In this situation, we choose the desired configuration as shown in Chapter 4 and we perform the simulation. The simulation performed in MATLAB is shown in figures 5.29,5.30. Figure 5.30b shows the trajectory achieved and we can observe that the achieved trajectory as desired.

The results are as follows:



(a) Force in x vs time



(b) force in y vs time

Figure 5.31: Door, Force vs time for Moving x_d,

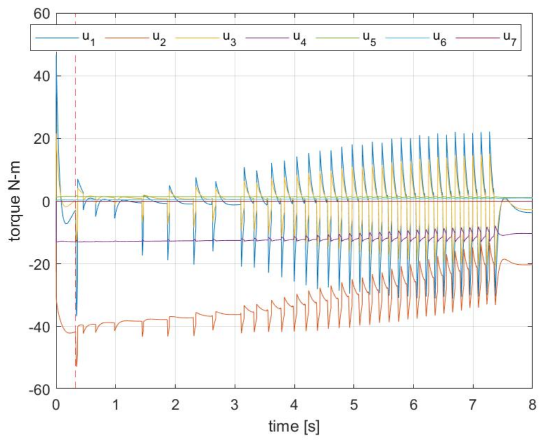


Figure 5.32: Door, Joint Torques vs time, Moving x_d

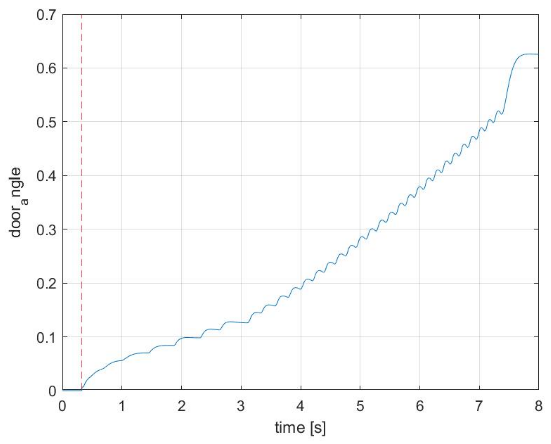
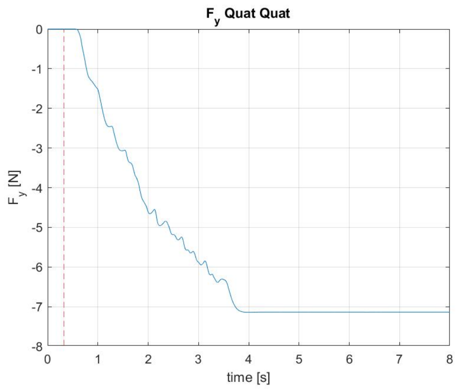


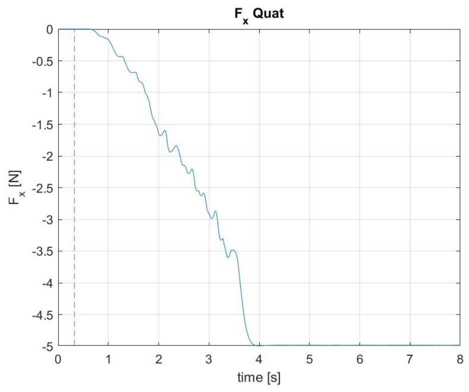
Figure 5.33: Door angle vs time, Moving x_d

The red vertical dotted line in the plots represent the begin of phase 2, i.e., opening the door. The figure 5.33 shows that the door is opened about an angle of 30^o. From figure 5.31 and figure 5.32, it can be inferred that, even though the force at the end effector is constantly changing throughout the task, the impedance controller compensates for it, rendering the task successful. The bumps in the joint torque vs time plot (Figure 5.32) shows the increase or change of the torques whenever the desired pose is moved. Also from the Figure 5.33, we can see that the door is opened to the required angle. From these results, we can infer that by using this method, the task is successfully executed where it is unsuccessful by using the traditional way. To simulate the real world scenario in the experiment, we introduce noise to the force/torque wrench which is pretty common in the force torque sensors at the end effector. We introduce a normally distributed noise with mean 0N and standard deviation 0.01N. The results are as follows:

Similar to the previous trials, the vertical dotted red line indicates the initiation of phase 2. This trial is also successful as the door is rotated till the needed angle and the joint torques are still in limit. Multiple Trials are now performed to check the noise level at which this fails, shown in Tab 5.5.



(a) Force in x vs time



(b) force in y vs time

Figure 5.34: Door, Force vs time for Moving x_d,

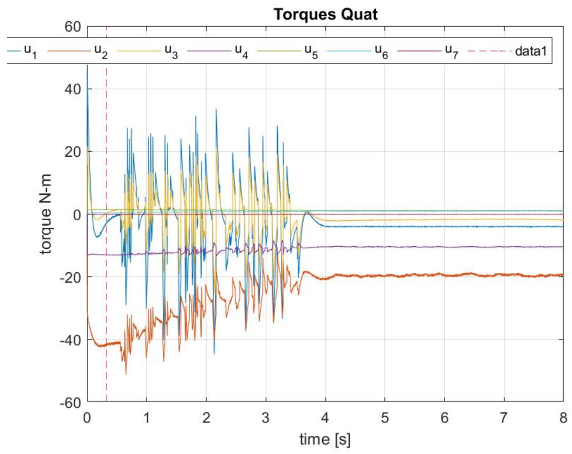


Figure 5.35: Door, Joint Torques vs time, Moving x_d

Trial	Standard deviation	Result	Notes
1	0.01	Pass	-
2	0.02	Pass	-
3	0.05	Pass	-
4	0.09	Fail	Joint torque limit reached

Table 5.5: Results with noisy sensor data

Chapter 6

Conclusion

Impedance control is one of the well-established technique to control the interaction control in robot manipulators. In this thesis we propose a method to accomplish and implement the impedance control algorithm without the need to select and tune the stiffness and damping coefficients for the manipulator for every task.

We consider multiple household tasks as examples for demonstration of this study such as pushing a spring loaded push button, rotating a spring loaded door knob, opening a door. We chose these tasks to show the functionality of this method against tasks with various constraints i.e., tasks with position and orientation constraints.

The approach is summarized as follows: We select an optimum stiffness constant for the manipulator and keep it constant.While performing a task, if the stiffness selected is not enough to produce joint torques enough to accomplish the task, we move the desired goal frame in a way such that the $(x_d - x_e)$ generates enough torque and guide the end effector in way to perform the task. Screw Liner Interpolation(ScLERP) is used to obtain the guiding or intermediate desired poses to make sure that the end effector satisfies the task constraints. This method is performed with different parameterizations for representation of the orientation of end effector i.e. both Euler angles and unit quaternion representations.

This method is then simulated on a Baxter Robot Arm in MATLAB and we were able to successfully perform the given tasks using this method with variations in the mechanical features of the environment (stiffness, position and orientations).

6.1 Future Developments

Since, the experimental part of this is done in simulations, for the future work, the proposed method can be validated on an actual robot. The experiments done only used a constant screw motion. In order to perform those tasks, whose motion does not consist of a single screw, a screw segmentation techniques can be used and the desired posed can be chosen as given in the proposed method.

Bibliography

[1] Neville Hogan. Impedance control: An approach to manipulation. In *1984 American control conference*, pages 304–313. IEEE, 1984.

[2] Siciliano B., Sciavicco L., Villani L., Oriolo G. *Robotics - Modelling, Planning and Control.* Springer, 2009.

[3] Fabrizio Caccavale, Bruno Siciliano, and Luigi Villani. The role of euler parameters in robot control. *Asian journal of control*, 1(1):25–34, 1999.

[4] Andreas Mueller. Higher derivatives of the kinematic mapping and some applications. *Mechanism and Machine Theory*, 76:70–85, 06 2014.

[5] N. Hogan. Adaptive control of mechanical impedance by coactivation of antagonist muscles. *IEEE Transactions on Automatic Control*, 29(8):681–690, 1984.

[6] N. Hogan. Controlling impedance at the man/machine interface. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 1626–1631 vol.3, 1989.

[7] T. Morita and S. Sugano. Design and development of a new robot joint using a mechanical impedance adjuster. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 2469–2475 vol.3, 1995.

[8] K.F. Laurin-Kovitz, J.E. Colgate, and S.D.R. Carnes. Design of components for programmable passive impedance. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1476–1481 vol.2, 1991.

[9] Homayoon Kazerooni. *A robust design method for impedance control of constrained dynamic systems.* PhD thesis, Massachusetts Institute of Technology, 1985.

[10] Fabrizio Caccavale, Bruno Siciliano, and Villani Luigi. Robot impedance control with nondiagonal stiffness. *Automatic Control, IEEE Transactions on*, 45:1943 – 1946, 11 1999.

[11] Federica Ferraguti, Cristian Secchi, and Cesare Fantuzzi. A tank-based approach to impedance control with variable stiffness. In *2013 IEEE International Conference on Robotics and Automation*, pages 4948–4953, 2013.

[12] Rethink Robotics. Baxter overview. `https://sdk.rethinkrobotics.com/wiki/Baxter_Overview`.

[13] Rethink Robotics. Baxter hardware specifications. `https://sdk.rethinkrobotics.com/wiki/Hardware_Specifications`.

[14] The Americans with Disabilities Act (ADA). Adjusting doors for access. `https://adata.org/factsheet/adjusting-doors-access`.