Kinematic and Dynamic analysis of a Remote Center of Motion for KUKA LWR in Robot-Assisted Minimally Invasive Surgery

Master's Thesis

SAPIENZA
UNIVERSITÀ DI ROMA

Ahmed Elhelow

Control Engineering

Supervised by
Prof. Marilena Vendittelli

October 24, 2019

Abstract

This project is about the application of a robot assistance in minimally invasive surgeries. While many methods were proposed for minimally invasive surgery, most of them have mechanical constraints limiting the robot's movement by using mechanical constraints to keep the Remote Center of Motion (RCM) coincide with the trocar (incision point) on the patient's body, through which the medical tool can be inserted. We inherit a different approach, where we obtain a programmable RCM that can be easily mapped in the future to be used also for open surgery and it would also be easier to extend the task that the robot can execute in a more dynamic way. An architecture for the control of a moving RCM will be introduced, which allows the insertion of a medical tool through the trocar, where a small mechanical cylinder is put to constrain the movement of the medical tool by minimizing the error between the RCM and the trocar positions. The RCM lives on the surgical tool and can dynamically change based on how much the tool should enter the patient's body. Another important aspect that should be taken into consideration is the fact that the robot should consider that the reference point will be affected by the patient's breathing and heart beats and this is achieved by estimating the external forces acting on the system. After building the kinematic model, dynamic model and control law for the KUKA LWR robot, we ran several experiments to see how the different parameters are behaving on our specific task.

Contents

Chapter 1

Introduction

This chapter will outline the modern technologies in the surgery room precisely the contribution of robots in minimally invasive surgery (MIS). It gives a review of the recent medical robot systems, including minimally invasive surgery robot systems as they are the primary subject of this thesis.

1.1 Background

By the 1990's robots had been used into a several areas of medicine including surgery and specifically minimally invasive surgery, which means performing surgery through "keyholes". MIS was found to have many advantages over certain types of operations performed by open surgery. These advantages included lower patient stress level, a less-time hospital stay after the surgical operation allowing the patient to resume their daily activities earlier, and preferable recovery quality from the cosmetic point of view. As MIS advanced and became more familiar, the more the use of robots was introduced into the field.

1.1.1 Laboratory robots

Robots were initially used in the hospital laboratory for sample preparation in the early 1980's. One of them was the Zymate system. It allowed movement in four directions for zoom, pan, actuator, swivel and tilt. The positional accuracy, defined as how accurate posterior movements from a reference point, was 2.5 mm and the repeatability, defined as reaching a reference point a repeated number of times, was nearly 1 mm. The arm could be used with numerous end effectors and had the ability to lift 1.5 Kg. The end effectors were a clamping hand, syringe and gripper.

Introducing automation helped with precision in the time-consuming and complex testing procedures in the laboratory. The robots conducted automatic tasks routinely and can be reprogrammed to do different tasks if needed. Recently Laboratory robots are cost effective, safe, efficient and flexible. Also they help to enhance accuracy, allow task standardization and increase sample throughput.

1.1.2 Rehabilitation robots

Rehabilitation robots development followed The development of laboratory robots. Rehabilitation robots attempt to give disabled people the required tools to enhance both productivity of their work and their quality of life. A several types of robots are used in prosthetics; those that either increase the ability of the individuals or replace a part of them; and tele-robotics where the robot works at a distance from the user. Examples of the sorts of roles of rehabilitation robots include every day living activities such as eating, educational tasks, game playing, and personal hygiene, and manipulative functions in a structured environment (laboratory test procedures, handling paper in the workplace, etc.).

Robots function as a partner with the patient to conduct rehabilitation tasks. For this goal, the instantaneous relationship between the robot and the patient must be very safe for people in the environment in general and specifically for the patient. To this end, the designing requirements of rehabilitation robots may be different from those of industrial robots which work in a structured environment away from humans.

The forms of tasks rehabilitation robots are desired to conduct are different to those done by laboratory and industrial robots, with several tasks either differing each time they are conducted or being done only once. Therefore rehabilitation robots needed the development of sensors to lead the robot in a dynamic changing environment which is different from the industrial robots static environment. Hence, the robot autonomy required to be bigger in order to adjust itself to new situations with respect to both the disabled person needs and the required task. When using rehabilitation robots the patient may have a supervisory role only. For example, a robot with highly specialized roles for physically impaired people such as the objects sensing and recognizing objects visually may only need higher intervention if the robot fail. The robot control in such more complicated environment as rehabilitation, may be conducted using a specific programming. The patient directs the task by, for example, aiming the robot through manual controls, advance sensory interfaces and task or object level descriptions which frees the patient from focusing on how the robot will execute the task.

1.1.3 Robots in surgery

Over the past 15 years, MIS has introduced powered master-slave instruments in which the surgeon is the controller and gives the commands, and the robot (slave) responds accordingly using different types of actuators. Master-slave robots were introduced by adding a computer within the device, allowing the user to conduct a task with higher accuracy and precision. Further development of these robots granted the giving commands from a distance. The surgeon would operate the robot and perform surgical operations from a distance using Internet as a high-speed communication system. A further step would be the development of autonomous robots programmed to perform specific operations with complete independence from the surgeon.

1.1.4 Robots in MIS

MIS imposes certain challenges and restrictions over open surgery. The surgeon operates through a nearly 1 cm in diameter set of holes. A video camera laparoscope gives a view

of the internal operating field. Long handled instruments grip and cut tissue within the body. Unlike open surgery, MIS avoids long incisions. Studies have also shown that compared with open surgery, MIS patients recover more quickly, have reduced discomfort, improved healing, reduced convalescence and hospitalisation costs, and spend less time away from productive work.

The dexterity required in MIS is different to that required by open surgery. Having only a few fixed points of entry for the instruments, MIS limits the type of movement so for example, lateral movement is not possible. The fulcrum effect has been widely identified as one of the main challenges of MIS. The fixed entry point constrains the instrument causing a fulcrum effect, which is resulting in a reversed movement of the instruments inside the abdomen, therefore making the operation more challenging.

Further difficulties arise in MIS due to the position of the video monitor which is often located on the far side of the patient. This results in a difference in orientation between the instruments in use and the surgeon's viewpoint. Surgery therefore becomes mentally taxing thus increasing the total workload of the operation Finally, haptic feedback (the contact force perception) greatly diminished in MIS makes difficult to identify consistencies and fragilities in tissue handling.

Over a relatively short period robotic manipulators were developed to resolve some of these challenges. The tasks carried out by these robots ranged from holding a camera, previously held by an assistant, to a complex robotic systems such as the Da Vinci (Intuitive Surgical, California USA) capable of doing several tasks such as camera guiding, instrument grasping and cutting, and suturing tissue.

In order to perform surgery, the robot must be able to perform several tasks that may include planning, registration, navigation and control.

1.2 Problem statement

Minimally invasive surgery requires introducing a tool inside the patient's body, where the tool can be a needle, an endoscope, or any other tool needed for a specific operation. Robots assistants in the medical field, especially in minimally invasive surgery were introduced to provide convenience to the medical doctors during operations, increase the precision and reduce the small faults. In minimally invasive surgery, the doctor should navigate inside the patient's stomach for example preventing any harm to the body, which requires a high level of expertise, training and several other factors such as the doctor's comfort, to be able to perform a high quality operation. In minimally invasive surgery, the constraints are even harder, because the tool is constrained to the trocar, which is the incision point, placed on the patient's body, where the surgical tool placed on the robot's end effector can only translate on the axis aligned with it and rotate about the same axis, while respecting the constraint on the RCM.

The operating modality of Minimally Invasive Surgery (MIS) consists in introducing the surgical tools inside the patient's body through small incisions and in guiding them from the outside of the body. In addition to the tools necessary to perform the surgery, a camera is introduced into the body to provide the surgeon with the view of the surgery site. Though highly advantageous for the patients, these operative conditions constrain the manipulation and are less natural and more tiring for the surgeon with respect to open surgery.

The introduction of robots in the operative room is aimed at recovering the conditions of the open surgery while providing an important support in terms of accuracy and comfort for the surgeon during the accomplishment of the surgical task. Robotic arms are used for moving the endoscopic camera only or the camera and the surgical tools. Whatever the setup of the robot-assisted surgery system is, the used manipulators must posses a special kinematics obtained either through mechanical design or by appropriate control of the manipulator joints. More specifically, the robot link constrained to move through the incision point (usually the link holding the surgical tool), can only translate along its axis and rotate about the incision point. The manipulator motion is then constrained with respect to a point on the link axis known as Remote Center of Motion (RCM).

1.3 State of the Art

Many different sorts of mechanically constrained RCM manipulators have been introduced in the literature. However, using a programmable RCM by constraining the movement of a serial robotic arm represents a more space-efficient and flexible solution which allows the use of the same manipulator for both open and for minimally invasive surgery.

Pham presented an architecture for controlling a moving remote center of motion in addition to the end-effector motion during robotic surgery [1]. It allowed both the incision point and the end effector to follow a trajectory, finds a Jacobian matrix that satisfies the velocity constraints in both the end-effector and the incision point frames. A redundant robot with eight revolute joints was considered, so that six DoF were used for controlling the end effector and the remaining two DoF were used to control the incision point. The standard body Jacobian matrix gave the mapping from the joint velocities to the end-effector velocities in body coordinates. The entry point was to follow a trajectory in the xy-plane. The used open-loop control law could be computed only when the Jacobian matrix had full rank, which means the robot had to be away from singular configurations. A well-known solution to deal with the problem of inverting the Jacobian matrix in the vicinity of a singularity is given by the so-called Damped Least-Squares (DLS) inverse method which allows the robot end effector to escape from the singularities by deviatation from the desired reference trajectory. Similar to this paper, our work uses the jacobian matrix for controlling the end effector and the constrained two DoF are used to control the incision point. However, our work doesn't need to know the trocar trajectory in the xy-plane in prior, only the initial position of the trocar point. Our work deals with moving that comes from breathing using force estimation and reaction. Our work deals with singularities, however, we used the pseudo inverse technique.

Sandoval presented a dynamically consistent control approach for manipulators with Cartesian Admittance Control (CAC) while guaranteeing a Remote Center of Motion (RCM) constraint [2]. It used the null space method as a general solution for a redundant robot which allowed to add an extended task to the robot. Compliance at the end-effector was added, since unexpected movements of the patient's body, such as breathing, involuntary movements or internal organ motions, can be dynamically compensated. Cartesian Admittance Control was used to implement a desired dynamic behavior to the end-effector, known as a mechanical admittance and characterized by a mass, damping and stiffness. The minimization of the forces applied by the surgical tool to the patient's body in the RCM was essential to avoid causing lacerations to

the patient. Similar to this paper, we used null-space motion to implement the control law to achieve the RCM constraint. Our work extends the control law to achieve an extended task in the null space of the RCM constraint control law. However, our work doesn't include Cartesian Admittance Control, which can be done in the future work. We used instead residual force estimation technique then simple reaction technique so that the end effector doesn't apply high force on the skin since admitance control is still applying some force to the skin.

In another paper, Sandoval provided a new generalized kinematic formulation of the Remote Center of Motion (RCM) constraint for serial robotic arms to be useful for implementation in torque-controlled robots [3]. The generalized formulation associated the task-space with the RCM constraint coordinates, which were described by the minimum distance between the surgical tool and the trocar. Two control methods were proposed and compared, one using a projection in the null-space, the other by combining both tasks using a task-space augmentation method position. The RCM constraint was presented in which a Jacobian matrix of the kinematic constraint was defined only as a function of the joint positions of the robot. Other tasks could be performed in lower priority levels, using the proposed Jacobian matrix of the kinematic constraint to calculate the null-space projectors. A second method to apply the RCM constraint in the task-space was presented, through a task-space augmentation method. In this method, a new task can be defined as an extended task, composed by the task concerning the tool-tip trajectory and the RCM constraint task. Similar to this paper, our work define the RCM constraint in a Jacobian matrix as a function of the joint positions of the robot. Also, Our control law minimizes the distance between the surgical tool and the trocar. However, we used separated distance variables for x, y and z axes. Moreover, similarly, we are using the null-space method to force the RCM constraint, however, differently, we are using the task-space augmentation method within the same control law to ensure the insertion angle.

Marinho presented a comparison of Remote Center-of-Motion Generation Algorithms [4]. He mentioned that the most well-known software-based RCM generation techniques were divided into two groups: the RCM was described from the viewpoint of a world frame, The second group aggregated the formulations that used the trocar point as the reference frame. When performing the same simulated trajectory, world-frame-based techniques might have algorithmic singularities, whereas trocar-frame-based techniques did not. The RCM control could be assigned as first priority, and the task assigned a secondary priority by using the null-space projection method. Algorithmic singularities are not the same as mechanical singularities. Mechanical singularities are joint configurations in which J, the regular Jacobian, loses rank. algorithmic singularities are postures in which the task Jacobian is singular, but J is full rank. Similar to this paper, our work uses the RCM is described from the viewpoint of a world frame. However, we haven't addresses the problem of algorithmic singularities in our work, which can be done in the future work. Similarly, we used the null-space projection method to prioritize the tasks and maintain the RCM constraints. Differently, we used the task augmentation within the null space method to control the insertion angle.

Sandoval presented another paper which provided a generalized Framework for Control of Redundant Manipulators [5]. He proposed a joint compliance strategy, by exploiting the Jacobian null-space. The proposed control framework dealt simultaneously with the surgical tool-tip trajectory, the RCM constraint and collisions in the robot's body. The surgical task performed by the tool-tip was in the first level of priority, by implementing a Cartesian compliance control approach. A kinematic formulation of the RCM constraint was exploited in the second level of

priority, in order to constraint the tool movements by the trocar position. The third level of priority was exploited to implement a joint compliance strategy dealing with collisions with the robot's body. This paper proposed to use a compliance control strategy based on the definition of the potential function of a virtual spring and added to a damping term to define the cartesian compliance control. A joint compliance control allowed to deal with collisions with the robot's body, and once the contact disappeared, the desired joint configuration was recovered. Similarly to this paper, our work consider dealing with collisions, however, we haven't used the cartesian compliance strategy. We dealt with the collisions by residual force estimation technique, then simple reaction. Differently, we assigned the RCM constraint to be the first priority task. Joint compliance strategy can be used in the future work to add more elasticity to the robot if needed.

Su presented a safety enhanced constraint which was applied on the compliant null space motion [6]. This control approach integrated an adaptive fuzzy compensator to guarantee the accuracy of the surgical tasks during the uncertain human-robot interaction. A redundant robot (LWR4+, KUKA, Germany) was torque-controlled through the Fast Research Interface (FRI), which provides direct low-level real-time access to the robot controller. External torque could be computed from external torque sensors. Since the hand force applied on the robot arm was uncertain and time-variable, the influence of human-robot interaction on the surgical tip was uncertain. Fuzzy adaptive nonlinear controller could compensate the unknown time-varying periodic disturbances from human-robot interaction. Similar to this paper, we compute the external torque, and we do so through the residual force estimation technique. Our work doesn't include a fuzzy adaptive nonlinear controller to compensate the unknown disturbances from human-robot interaction, which can be done in the future work.

In another paper, Su presented a hierarchical operational space formulation, which was considered to achieve whole-body impedance control of a 7-DoF serial robot [7]. Its strategy was achieving RCM constraint in the 1st Null-space of the Jacobian matrix of surgical tip, then adding a safety-enhanced compliant arm behavior in the 2nd Null-space. A decoupled fuzzy compensator was introduced to compensate the disturbance achieving human-robot collaborative control. Similarly, our work uses null space method to achieve hierarchical operational space formulation, however, we have the RCM constraint as the 1st priority. differently, our work doesn't include a decoupled fuzzy compensator to compensate the disturbance from human-robot collaborative control, which can be done in the future work.

Aghakhani et al. proposed a RCM constraint formalization that models explicitly the translation along the link axis and therefore allows control directly of a variable representing a link penetration into a patient's body [8]. We followed the approach in [8] and implemented on the V-REP with KUKA LWR IV.

1.4 Proposed Solution

The main contribution of the paper is to implement the method proposed in [8] in V-REP using the dynamic model of the KUKA LWR IV.
The whole algorithm was developed in C++, simulations were done in V-REP, where we could actually visualize the KUKA robot performing the task and were able to plot several variables to see the behavior of our algorithm.
We added a world to base frame transformation to express the external forces in the V-REP world frame. We also fixed the jacobian calculation to be calculated through more general

approach. Moreover, we used the dynamic parameters form [9] in the C++ code and in the V-REP so that we have a very close simulation to the real robot. Furthermore, we introduced the idea of using γ as a penetration angle. Also, we used a secondary task to ensure the joints are in range. Another contribution is using the force and torque transformation matrix to transform the external force to the force sensor position to validate our work. Moreover, we used the parallel axis theorem and rotation matrices to obtain inertia matrices in V-REP world frame.

Chapter 2

Kuka LWR

2.1 Kinematic Model

For this project we decided to use the KUKA LWR 4+, which is a 7-DoF robot having a spherical shoulder with 3-DoF, an elbow with 1-DoF and a spherical wrist with 3-DoF. Previously, in another work, the kinematic model was calculated by hand for another robot, but in our project, we calculated it by building the homogeneous transformation matrices between the different reference frames of the robot, which provided us with the complete kinematic model of the robot. This way we could prevent possible mistakes in calculating the kinematic model and saved us time and the burden of doing all the calculations by hand, where all the calculations were done inside the C++ and matlab codes. After obtaining the full kinematic model of the robot, we could use it to build the kinematic control law and execute our first task, which is driving the RCM on the surgical tool to the specified position that is equal to the trocar position in this case. To build the kinematic model, we followed the convention provided by Claudio et. al [1]. In V-REP, the reference frames were inconsistent with the convention provided in the paper, so we changed them following the DH table.

$$ {}^{i-1}A_i = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

$$ \text{s. t.} \quad c = cos, \quad s = sin \tag{2.2} $$

After we did the adjustments in V-REP, we obtained the same model as the one provided by Claudio **Figure 2.1, 2.2**. The calculated transformation matrices can be seen below:

$$ {}^{0}A_1 = \begin{bmatrix} c1 & 0 & s1 & 0 \\ s1 & 0 & -c1 & 0 \\ 0 & 1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}A_2 = \begin{bmatrix} c2 & 0 & -s2 & 0 \\ s2 & 0 & c2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}A_3 = \begin{bmatrix} c3 & 0 & -s3 & 0 \\ s3 & 0 & c3 & 0 \\ 0 & -1 & 0 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_4 = \begin{bmatrix} c4 & 0 & s4 & 0 \\ s4 & 0 & -c4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{4}A_5 = \begin{bmatrix} c5 & 0 & s5 & 0 \\ s5 & 0 & -c5 & 0 \\ 0 & 1 & 0 & d5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}A_6 = \begin{bmatrix} c6 & 0 & -s6 & 0 \\ s6 & 0 & c6 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{6}A_7 = \begin{bmatrix} c7 & -s7 & 0 & 0 \\ s7 & c7 & 0 & 0 \\ 0 & 0 & 1 & d7 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

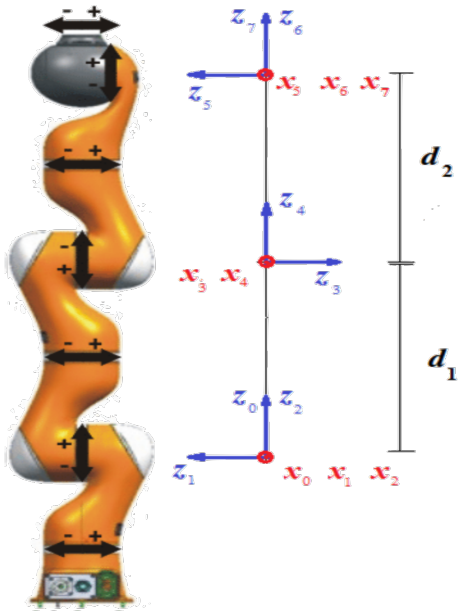


Figure 2.1: New reference frames

i	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	q_1
2	0	$-\pi/2$	0	q_2
3	0	$-\pi/2$	d_1	q_3
4	0	$\pi/2$	0	q_4
5	0	$\pi/2$	d_2	q_5
6	0	$-\pi/2$	0	q_6
7	0	0	0	q_7

Figure 2.2: DH-table parameters

Where d1 and d2 shown in Figure 2.1 are equal to d3 and d5 in the calculated matrices and d1 in the transformation matrices was set to 0.
After having the transformation matrices calculated using the Denavit-Hartenberg parameters, we could multiply the transformation matrices to obtain the full kinematic model:

$$ {}^{0}\mathbf{T}_7 = {}^{0}\mathbf{A}(q_1)_1.{}^{1}\mathbf{A}(q_2)_2.{}^{2}\mathbf{A}(q_3)_3.{}^{3}\mathbf{A}(q_4)_4.{}^{4}\mathbf{A}(q_5)_5.{}^{5}\mathbf{A}(q_6)_6.{}^{6}\mathbf{A}(q_7)_7 \tag{2.3} $$

$$ {}^{0}\mathbf{T}_7 = \begin{bmatrix} & {}^{0}\mathbf{R}_7 & & {}^{0}\mathbf{p}_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

Parameter	Value	Parameter	Value
d_1	0	d_3	0.4
d_5	0.39	d_7	$0.078 + 0.23$

Table 2.1: DH parameters values

Having computed the direct kinematics of the robot, we can proceed to calculate the jacobian matrix. Jacobian matrix is the relation between motion (velocity) in joint space and motion (linear/angular velocity) in task space. We Differentiate the position with respect to all of the joints to finally obtain the jacobian matrix, which is a block matrix having the upper block dedicated to the linear velocity and the lower block describes the angular velocity that relates the joints velocities to the velocity of the end effector. We calculated the geometric jacobian of the 6^{th} and 7^{th} links, where the geometric jacobian consists of linear part, and angular part as following:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = \begin{bmatrix} J_L(q) \\ J_A(q) \end{bmatrix} \dot{q} = J(q)\dot{q} \tag{2.5}$$

$$J_L(q) = \begin{bmatrix} J_{L1}(q) & ... & J_{Ln}(q) \end{bmatrix} \tag{2.6}$$

$$J_A(q) = \begin{bmatrix} J_{A1}(q) & ... & J_{An}(q) \end{bmatrix} \tag{2.7}$$

Geometric jacobian doesn't require calculating derivatives

$$\mathbf{J}_{Li}(q) = Z_{i-1} \times \mathbf{P}_{i-1,E} \tag{2.8}$$

$$Z_{i-1} = {}^{0}\mathbf{R}_1(q_1)...{}^{i-2}\mathbf{R}_{i-1}(q_{i-1}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.9}$$

$$\mathbf{P}_{i-1,E} = \mathbf{P}_{0,E}(q_1, ..., q_n) - \mathbf{P}_{0,i-1}(q_1, ..., q_{i-1}) \tag{2.10}$$

and

$$\mathbf{J}_{Ai}(q) = Z_{i-1} \tag{2.11}$$

where J_{Li} is the linear block of the jacobian matrix at the i^{th} column, which resembles the i^{th} joint, and J_{Ai} is the angular block of the jacobian matrix at the i^{th} column, which resembles the i^{th} joint. Here we are interested in the velocity of the RCM, which lies on the axis between the end effector and the tool tip. In particular, we need to compute the linear velocity components of the jacobian, since the last link can only translate along the axis aligned with the last link, which was subsequently used in the development of the control law. Moreover, it is worth to note that we are only considered the revolute joint case for calculating the jacobian since our robot KUKA LWR IV has only revolute joints.

2.2 Dynamics

2.2.1 Dynamic Model

Although the LWR robot is equipped with harmonic drives and displays non-negligible joint elasticity, thanks to the torque control mode of the KUKA controller, it is possible to by-pass the elasticity in the dynamic model and consider only the link dynamics which is described by

$$B(q)\ddot{q} + c(q, \dot{q}) + g(q) = \tau \tag{2.12}$$

where $B(q)$ is the inertia matrix, $c(q, \dot{q})$ is the Coriolis and centrifugal vector, $g(q)$ is the gravity vector, and τ is the vector of joint torques downstream the elasticity.
For a rigid robot with N links, the link masses, the position of their centers of mass, and the elements of the inertia matrices of the links build up a vector of dynamic parameters. In KUKA LWR, $N = 7$, and denote by m_i be the mass of link i, for $i = 1, \ldots, 7$. The position of the center of mass of link i with respect to the $i - th$ link frame is

$$r_{i,ci} = \begin{bmatrix} c_{ix} \\ c_{iy} \\ c_{iz} \end{bmatrix} \tag{2.13}$$

Similarly, the link inertia matrix relative to the center of mass of link i is the (possibly full) symmetric matrix

$$I_i = \begin{bmatrix} I_{ixx} & I_{ixy} & I_{ixz} \\ I_{ixy} & I_{iyy} & I_{iyz} \\ I_{ixz} & I_{iyz} & I_{izz} \end{bmatrix} \tag{2.14}$$

The values of all dynamic parameters of the KUKA LWR are given in Table 2.2

Parameter	Value	Parameter	Value
m_1	4.1948152162	I_{2yy}	0.05
m_2	4.2996847737	I_{2zz}	0.001601901
m_3	3.658530333	I_{2xy}	−0.00000621
m_4	2.3846673548	I_{2xz}	0.0001166457
m_5	1.7035567183	I_{2yz}	−0.0009141575
m_6	0.4000713156	I_{3xx}	0.0469510749
m_7	0.6501439811	I_{3yy}	0.0008344566
c_{1x}	−0.0216387515	I_{3zz}	0.05
c_{1y}	0.01	I_{3xy}	−0.000271431
c_{1z}	−0.0376881829	I_{3xz}	$4.09E-008$
c_{2x}	0.0003284751	I_{3yz}	−0.000577228
c_{2y}	−0.0041132249	I_{4xx}	0.0124233226
c_{2z}	0.0823647642	I_{4yy}	0.0072708907
c_{3x}	0.0002593328	I_{4zz}	0.0099884782
c_{3y}	0.1137431845	I_{4xy}	0.000000225
c_{3z}	−0.000100257	I_{4xz}	−0.0005187982

Table 2.2: Values of dynamic parameters

Parameter	Value	Parameter	Value
c_{4x}	-0.0014648843	I_{4yz}	-0.0005484476
c_{4y}	-0.0000461	I_{5xx}	0.006322648
c_{4z}	0.148580959	I_{5yy}	0.0012020203
c_{5x}	-0.0003791484	I_{5zz}	0.0070806218
c_{5y}	-0.0553526131	I_{5xy}	-0.0002163196
c_{5z}	-0.0101255137	I_{5xz}	0.00000652
c_{6x}	0.0020739022	I_{5yz}	-0.005
c_{6y}	0.0586184696	I_{6xx}	0.0005278646
c_{6z}	-0.044799983	I_{6yy}	0
c_{7x}	-0.0004601303	I_{6zz}	0.0034899625
c_{7y}	0.0014789221	I_{6xy}	0.0000483
c_{7z}	0.0715608282	I_{6xz}	-0.0000375
I_{1xx}	0.01	I_{6yz}	-0.0010605344
I_{1yy}	0.0018932828	I_{7xx}	0
I_{1zz}	0.01	I_{7yy}	0.0000323
I_{1xy}	0.01	I_{7zz}	0.0001187527
I_{1xz}	0.01	I_{7xy}	-0.000000577
I_{1yz}	0.01	I_{7xz}	0
I_{2xx}	0.0474108647	I_{7yz}	0

Table 2.3: Values of dynamic parameters

2.2.2 Newton-Euler Approach

The Newton-Euler approach is based on a balance of all the forces acting on the generic link of the manipulator. This leads to a set of equations whose structure allows a recursive type of solution; a forward recursion is performed for propagating link velocities and accelerations, followed by a backward recursion for propagating forces.

Consider the generic Link i, its dynamics is described by two equations:

- Newton dynamic equation:
  - Sum of forces = Variation of linear momentum

$$\sum f_i = \frac{d}{dt}(mv_c) = m\dot{v}_c \tag{2.15}$$

  - Forces:
    - f_i force applied from link $i-1$ on link i
    - f_{i+1} force applied from link i on link $i+1$
    - $m_i g$ gravity force

    All vectors expressed in the same reference frame RF_i
  - Newton equation

$$f_i - f_{i+1} + m_i g = m_i a_{ci} \tag{2.16}$$

    where a_{ci} is the linear acceleration of link i center of mass

- Euler dynamic equation:

- Sum of torques = Variation of angular momentum

$$\sum \mu_i = \frac{d}{dt}(I\omega) = I\dot{\omega} + \omega \times I\omega \tag{2.17}$$

- Torques:
  - τ_i torque applied from link $i-1$ on link i
  - τ_{i+1} torque applied from link i on link $i+1$
  - Torque due to f_i w.r.t. $C_i : f_i \times r_{i-1,ci}$
  - Torque due to $-f_{i+1}$ w.r.t. $C_i : -f_{i+1} \times r_{i,ci}$
  - Gravity force gives no torque at C_i

  All vectors expressed in the same reference frame RF_i
- Euler equation

$$\tau_i - \tau_{i+1} + f_i \times r_{i-1,ci} - f_{i+1} \times r_{i,ci} = I_i\dot{\omega}_i + \omega_i \times (I_i\omega_i) \tag{2.18}$$

  where $\dot{\omega}_i$ is the angular acceleration of body i

Note that the principle of action and reaction says that forces/torques applied by body i to body $i+1$ equal minus forces/torques applied by body $i+1$ to body i.

Newton–Euler equations of motion are not in closed form, since the motion of a single link is coupled to the motion of the other links through the kinematic relationship for velocities and accelerations. Once the joint positions, velocities and accelerations are known, one can compute the link velocities and accelerations, and the Newton-Euler equations can be utilized to find the forces and moments acting on each link in a recursive fashion, starting from the force and moment applied to the end-effector.

Note that we only consider the revolute joints case here since KUKA LWR all joints are revolute.

Algorithm:

- Forward recursion: Computing velocities and accelerations

$$\begin{aligned} \omega_i &= {}^{i-1}R_i^T[\omega_{i-1} + \dot{q}_i z_{i-1}] \\ \dot{\omega}_i &= {}^{i-1}R_i^T[\dot{\omega}_{i-1} + \ddot{q}_i z_{i-1} + \dot{q}_i \omega_{i-1} \times z_{i-1}] \\ a_i &= {}^{i-1}R_i^T a_{i-1} + \dot{\omega}_i \times {}^i r_{i-1,i} + \omega_i \times (\omega_i \times {}^i r_{i-1,i}) \\ a_{ci} &= a_i + \dot{\omega}_i \times r_{i,ci} + \omega_i \times (\omega_i \times r_{i,ci}) \end{aligned} \tag{2.19}$$

- Backward recursion: Computing forces and torques

$$f_i = f_{i+1} + m_i(a_{ci} - {}^i g) \tag{2.20}$$

$$\tau_i = \tau_{i+1} - f_i \times (r_{i-1,i} + r_{i,ci}) + f_{i+1} \times r_{i,ci} + I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i) \tag{2.21}$$

- At last:

$$u_i = \tau_i^{T\,i} z_{i-1} + \eta_i \dot{q}_i \tag{2.22}$$

  where $\eta_i \dot{q}_i$ is the viscous friction

If the equations of motion are obtained with Newton-Euler formulation, it is possible to compute direct dynamics by using a computationally efficient method. We can compute the gravity terms by calling Newton-Euler formulation with $\dot{q} = 0$ and $\ddot{q} = 0$. Also, we can compute the centrifugal and coriolis terms by calling Newton-Euler formulation with $g = 0$ and $\ddot{q} = 0$. Furthermore, we can compute the $i-th$ column of the inertia matrix by calling Newton-Euler formulation with $g = 0$, $\dot{q} = 0$ and $\ddot{q} = e_i$ s.t. $e_i : i-th$ column of identity matrix. Finally, we can compute the inverse dynamics by calling Newton-Euler formulation with $q = q_d$, $\dot{q} = \dot{q}_d$ and $\ddot{q} = \ddot{q}_d$ as in Table 5.3.

Required	Newton-Euler Function
Gravity terms	$g(q) = NE_g(q, 0, 0)$
Centrifugal and Coriolis terms	$c(q, \dot{q}) = NE_0(q, \dot{q}, 0)$
$i-th$ column of the inertia matrix	$B_i(q) = NE_0(q, 0, e_i)$ $s.t.$ $e_i : i-th$ column of identity matrix
Inverse Dynamics	$\tau_d = B(q_d)\ddot{q}_d + c(q_d, \dot{q}_d) + g(q_d)$ $= NE_g(q_d, \dot{q}_d, \ddot{q}_d)$

Table 2.4: Newton-Euler Approach

2.2.3 External forces dynamics

Taking into account the external forces coming from breathing and heart beating, we added external forces to the dynamic model

$$B(q)\ddot{q} + c(q, \dot{q}) + g(q) = \tau + J_c^T(q) f_{ext} \tag{2.23}$$

Where $J_c(q)$ is the jacobian matrix computed at the contact point (i.e. RCM point)

$$J_c(q) = J_{RCM}(q) \tag{2.24}$$

We need to detect and estimate these external forces. Then, we include them into the dynamical model to eliminate their effect on following a desired trajectory.

Chapter 3

Kinematic Control

In this work we followed the approach developed in [8], a task control with Remote Center of Motion constraint for minimally invasive surgery, with a few modification to implement it on V-REP with KUKA LWR IV.
In robots assisted minimally invasive surgery, the task is the restriction of the feasible motion of the RCM point $\boldsymbol{P}_{RCM}$, which can be located anywhere on the last link added to the end effector, who's position should be equal to the position of the trocar point $\boldsymbol{P}_{trocar}$, that is the incision point in the patient's body through which the surgical tool can be inserted. The rotation and translation of $\boldsymbol{P}_{RCM}$ on the surgical tool is always restricted to the trocar point as shown in **Figure 3.1** and this problem should be carefully treated in order to prevent any harm, or undesired motion of the surgical tool during the operation.
Several kinds of mechanically constrained RCM manipulators were proposed, but here we worked on obtaining a programmable RCM using a 7-DoF serial manipulator, which makes it easier to extend the task performed by the robot and reduces the complexity of mapping the usage of the same manipulator for open surgeries and other tasks as well.

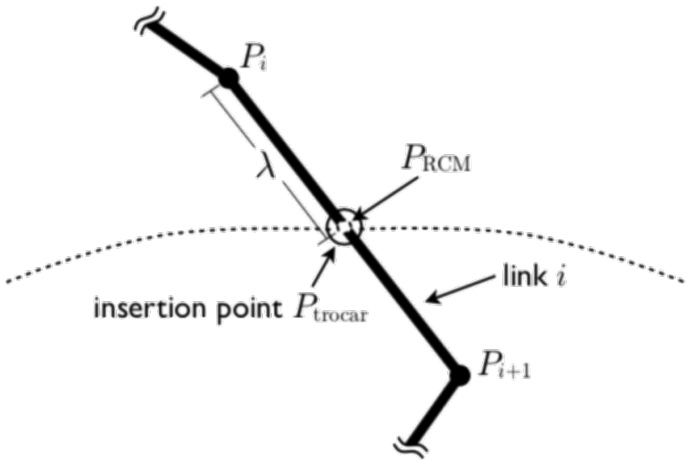


Figure 3.1: Tool insertion through the trocar point

We consider a 7-DoF serial manipulator of which all the joints are outside the patient's body, so the RCM can be positioned on the last link, which is the surgical tool, where we have:

(1) The vector of joints variables

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{bmatrix}^{\mathrm{T}} \tag{3.1}$$

(2) The cartesian coordinates of points $\mathbf{P}_6$ and $\mathbf{P}_7$

$$\mathbf{P}_6(\mathbf{q}(t)), \quad \mathbf{P}_7(\mathbf{q}(t)) \tag{3.2}$$

(3) The coordinates of $\boldsymbol{P}_{RCM}$

$$\boldsymbol{P}_{RCM}(\mathbf{q}(t)) \tag{3.3}$$

(4) The part of the surgical tool that is above $\boldsymbol{P}_{trocar}$

$$0 \leq \lambda(t) \leq 1 \tag{3.4}$$

Having defined all of the above and taking into consideration that $\mathbf{P}_{\mathrm{RCM}}$ can be anywhere on the last link and its coordinates can change over the time, we obtain:

$$\boldsymbol{P}_{RCM}(\mathbf{q}(t)) = \mathbf{P}_6(\mathbf{q}(t)) + \lambda(t)(\mathbf{p}_7(\mathbf{q}(t)) - \mathbf{p}_6(\mathbf{q}(t))) \tag{3.5}$$

And differentiate it with respect to time to obtain the the velocity of $\boldsymbol{P}_{RCM}$

$$\dot{\boldsymbol{P}}_{RCM} = \begin{bmatrix} \mathbf{J}_6 + \lambda(\mathbf{J}_7 - \mathbf{J}_6) & \mathbf{P}_7(\mathbf{q}(t)) - \mathbf{P}_6(\mathbf{q}(t)) \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\lambda} \end{bmatrix} = \mathbf{J}_{RCM}(\mathbf{q}, \lambda) \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\lambda} \end{bmatrix} \tag{3.6}$$

Since the RCM position should always be equal to the position of the trocar point, their derivative with respect to time should be equal to zero. We calculate the derivative and obtain:

$$\dot{\boldsymbol{P}}_{RCM} = \mathbf{J}_{RCM}(\mathbf{q}, \lambda) \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\lambda} \end{bmatrix} = 0 \tag{3.7}$$

The control law that allows us to control the robot and achieve the desired task:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\lambda} \end{bmatrix} = \mathbf{J}^{\dagger}_{RCM} \mathbf{K}_{RCM} \mathbf{e} + (I - \mathbf{J}^{\dagger}_{RCM} \mathbf{J}_{RCM}) w \tag{3.8}$$

Where $\mathbf{J}_{\mathrm{RCM}}$ is the jacobian of the $\mathbf{P}_{\mathrm{RCM}}$, $\mathbf{J}^{\dagger}_{RCM}$ is the pseudo inverse of $\mathbf{J}_{\mathrm{RCM}}$. If $\mathbf{J}_{\mathrm{RCM}}$ is full (row) rank, then

$$\mathbf{J}^{\dagger}_{RCM} = \mathbf{J}^{T}_{RCM} (\mathbf{J}_{RCM} \mathbf{J}^{T}_{RCM})^{-1} \tag{3.9}$$

else, it is computed numerically using the SVD (Singular Value Decomposition) of $\mathbf{J}_{\mathrm{RCM}}$. If the task is feasible, there will be no task error.

$\mathbf{K}_{\mathrm{RCM}}$ is a constant 3x3 matrix representing the gain and the first part of **eq: 3.9** represents the original task that should be achieved by the robot. $\mathbf{e}$ is the error of the original task that should be achieved by the robot, which is driving the end effector to the position where $\mathbf{P}_{\mathrm{RCM}}$ coincides with $\mathbf{P}_{\mathrm{trocar}}$. In order to be able to extend the task including a new one and still satisfying the original task while giving it the highest priority and achieving the new task, we used the null space method which helps us satisfy these conditions. Here ω does not have any noticeable effect before adding the new task. After extending the task, we obtain the extended form of the above formula:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\lambda} \end{bmatrix} = \mathbf{J}^{\dagger}_{ext} \begin{bmatrix} \mathbf{K}_t & 0_{n_t \times 3} \\ 0_{3 \times n_t} & \mathbf{K}_{RCM} \end{bmatrix} \mathbf{e}_t + (I - \mathbf{J}^{\dagger}_{ext} \mathbf{J}_{ext}) w \tag{3.10}$$

Where $\mathbf{K}_{\mathrm{t}}$ (diagonal gain matrix for the extended task) and $\mathbf{K}_{\mathrm{RCM}}$ are two 3x3 diagonal positive definite matrices and the error becomes that of the extended task

$$\mathbf{e}_t = \begin{bmatrix} t_d - t \\ \mathbf{P}_{RCM} - \mathbf{P}_{trocar} \end{bmatrix} \tag{3.11}$$

$$\mathbf{J}_{ext}^{\dagger} = \begin{bmatrix} \mathbf{J}_t \\ \mathbf{J}_{RCM} \end{bmatrix} \tag{3.12}$$

The jacobian now also refers to the extended task instead of being only the jacobian of the RCM. As previously stated, using the null space method, $\mathbf{w}$ is the residual input projected in the null space of the jacobian, which is used to satisfy additional tasks while assuring the convergence of the original task as well.
It is worth mentioning that we have used the extended task within the e_t, $t_d - t$, to specify the desired position of p_6 which allows to specify the penetration angle γ

$$\gamma = tan^{-1}\left(\frac{P_{6,z} - P_{RCM,z}}{\sqrt{(P_{6,x} - P_{RCM,x})^2 + (P_{6,y} - P_{RCM,y})^2}}\right) \tag{3.13}$$

Hence, the jacobian of the extended task is the jacobian of P_6

$$\mathbf{J}_t = \mathbf{J}_{6,L} \tag{3.14}$$

$$\mathbf{J}_{6,L} = \begin{bmatrix} J_{L1}(q) & \dots & J_{L6}(q) & 0 \end{bmatrix} \tag{3.15}$$

Furthermore, we used the null space method to ensure satisfying the RCM constraints as the primary task, then we are able to add a secondary task which will be executed in the null space of the primary task. We chose the secondary task to be minimizing the distance from the mid points of the joint ranges to ensure that the resulting solution is in joint range

$$\dot{q}_0 = -\nabla_q H(q) \tag{3.16}$$

Consider that joint i has the following joint range

$$q_i \in [q_{m,i}, q_{M,i}] \tag{3.17}$$

$$\bar{q}_i = \frac{q_{M,i} + q_{m,i}}{2} \tag{3.18}$$

then

$$H_{range}(q) = \frac{1}{2N}\sum_{i=1}^{N}\left(\frac{q_i - \bar{q}_i}{q_{M,i} - q_{m,i}}\right)^2 \tag{3.19}$$

In our case, $\bar{q}_i = 0$ for all joints, then

$$\dot{q}_{0,i} = -\frac{q_i}{N(q_{M,i} - q_{m,i})} \tag{3.20}$$

Furthermore, since λ doesn't express a real joint, we included an equation to ensure the minimization of λ in the null space of the primary task

$$\dot{q}_{0,8} = K(\lambda_d - \lambda) \tag{3.21}$$

Other secondary tasks can be used such as maximizing the distance from singularities using $H(q)$ as the manipulability equation, or maximizing the minimum distance to Cartesian obstacles. However, in the case of maximization

$$\dot{q}_0 = \nabla_q H(q) \tag{3.22}$$

Chapter 4

Force Estimation

Physical Human-Robot Interaction (pHRI) is a challenging requirement that comes from the cohabitation and collaboration between the humans and the robot. In particular, this interaction involves forces exchanges, that needs to happen without any issue or concern. We can consider three phases of the collision:

- Detection phase: It is the phase whose binary output denotes whether a robot collision occurred or not. The occurrence of a collision, which may happen anywhere along the robot structure, shall be detected as fast as possible. A major practical problem is the selection of a threshold on the monitoring signals so as to avoid false positives and achieve high sensitivity at the same time. One way to do that is to compare the actual commanded torques with the nominal model-based control law (i.e., the instantaneous torque expected in the absence of collision), with any difference being attributed to a collision.

- Isolation phase Knowing which robot part(e.g., which link) is involved in the collision is an important information that can be exploited for robot reaction. Collision isolation aims at localizing the contact point x_c, or at least the which link i_c out of the n-body robot collided. To do so, we can use acceleration estimates for torque prediction and comparison, which inherently introduces noise. The common drawback of this method is that the effect of a collision on a link propagates to other link variables or joint commands due to robot dynamic couplings.

- Identification phase The direction and intensity of the generalized collision force, either in terms of the acting Cartesian wrench f_{ext} at the contact or of the resulting external joint torque τ_{ext} during the entire physical interaction event, are very important information for later reaction phase. This information characterizes the collision event.

4.1 Estimation of τ_{ext} via momentum observer

We followed the work in [8], and implemented the Residual method to detect and estimate the external forces in the V-REP with KUKA LWR IV. This monitoring method based on the generalized momentum, and it was motivated by the desire of avoiding the inversion of the robot inertia matrix, decoupling the estimation result, and also eliminating the need of an estimate of joint acceleration.

Residual vector contains directional information of the torque at the robot joints resulting from the link collision. We can confirm that there is a collision in an area up to the i-th link if residual vector has zero values starting from the (i+1)th element

$$r = [* \quad \dots \quad * \quad * \quad 0 \quad \dots \quad 0]^T \tag{4.1}$$

Which is considered as Detection and Isolation.

Residual vector can be calculated by

$$r(t) = K_0\Big[p(t) - \int_0^t \Big(\tau_m + C^T(q,\dot{q})\dot{q} - g(q) + r\Big)ds - p(0)\Big] \quad s.t. \quad r(0) = 0 \tag{4.2}$$

where $K_0 = diagK_{0,i} > 0$ is the diagonal gain matrix of the observer.

In ideal condition, the dynamic relation between the external joint torque will be τ_{ext} and r

$$\dot{r} = K_0(\tau_{ext} - r) \tag{4.3}$$

In other words, the filter equation is a stable, linear, decoupled, first-order estimation of the external collision joint torque τ. Also, the monitoring vector r is sensitive to collisions even when $\dot{q} = 0$.

For sufficiently large gains

$$r \approx \tau_{ext} \tag{4.4}$$

This nice feature makes the momentum observer a kind of virtual sensor for external joint torques acting along the robot structure

$$\tau_{ext} = J_c^T(q) f_{ext} \tag{4.5}$$

$$f_{ext} = (J_c^T(q))^{\#}\tau_{ext} \tag{4.6}$$

$$f_{ext} \approx (J_c^T(q))^{\#} r \tag{4.7}$$

External forces are computed w.r.t. the base frame. In order to compare these results with the values come from the force sensor for validation, we need to do a transformation to get the external forces computed w.r.t. the sensor frame.

In general, force and torque transformation can be computed by

$$\begin{bmatrix} {}^2f_2 \\ {}^2\tau_2 \end{bmatrix} = \begin{bmatrix} {}^2R_1 & 0 \\ -{}^2R_1\,{}^1P_{12} & {}^2R_1 \end{bmatrix} \begin{bmatrix} {}^1f_1 \\ {}^1\tau_1 \end{bmatrix} \tag{4.8}$$

${}^1P_{12}$ is a 3×3 skew-symmetric matrix of ${}^1p_{12}$ such that ${}^1p_{12}$ is the displacement of RF_2 w.r.t. RF_1, expressed in RF_1.

The sensor has the same frame as the frame of the last joint.

$${}^7f_{ext} = {}^0R_7^T\,{}^0f_{ext} \tag{4.9}$$

We can compute residual vector using discrete Euler integration

$$\begin{aligned} beta &= n - \dot{B} \\ sumDyn &= sumDyn + (tau - beta) * deltaTime \\ sumRes &= sumRes + output * deltaTime \\ output &= K * (B * qd - sumDyn - sumRes) \end{aligned} \tag{4.10}$$

Such that n is the coriolis, centrifugal, and gravity terms.

Chapter 5

Implementation

5.1 V-REP

Since the control law that we are using is a velocity control law, and V-REP simulation controls the KUKA LWR robot using position control, we integrated the velocity control law using runge kutta numerical method using odeint library, which is a modern C++ library for numerically solving Ordinary Differential Equations.

We have used V-REP to conduct the simulations. V-Rep (virtual robot experimentation platform) is a virtual reality simulation software that allows the simulation of many robotic systems. All objects and models can be individually controlled using, for instance, embedded routines, plug-in's, ROS nodes, or remote servers connected through API's. These routines can be implemented using C/C++, Python, Java, Lua, Matlab, Octave, or Urbi.

The KUKA LWR model in V-REP has some differences than the real robot. Then, we need to fix the dynamic parameters of the V-REP model to be equal to the values estimated and test by Claudio. First, the position of the center of mass of link i with respect to the ith link frame, ${}^{i}r_{i,ci}$. We need to express ${}^{i}r_{i,ci}$ in the world frame since V-REP dynamic parameters can be assigned w.r.t. world frame.

$$ {}^{i}r_{i,ci} = \begin{bmatrix} c_{ix} & c_{iy} & c_{iz} \end{bmatrix}^{T} , \qquad i = 1, ..., 7 \tag{5.1} $$

$$ {}^{i}r_{i,ci,hom} = \begin{bmatrix} c_{ix} & c_{iy} & c_{iz} & 1 \end{bmatrix}^{T} , \qquad i = 1, ..., 7 \tag{5.2} $$

$$ {}^{w}r_{i,ci,hom} = {}^{w}T_{i}\ {}^{i}r_{i,ci,hom} , \qquad i = 1, ..., 7 \tag{5.3} $$

The resulting values are in Table 5.1

Second, the link inertia matrix relative to the center of mass of link i

$$ {}^{i}I_{l_i} = \begin{bmatrix} I_{ixx} & I_{ixy} & I_{ixz} \\ I_{ixy} & I_{iyy} & I_{iyz} \\ I_{ixz} & I_{iyz} & I_{izz} \end{bmatrix}^{T} , \qquad i = 1, ..., 7 \tag{5.4} $$

We used the parallel axis theorem and rotation matrices then dividing by mass to obtain the link massless inertia matrix expressed in the world frame, which can be assigned in the V-REP environment.

$$ {}^{w}\bar{I}_{l_i} = {}^{i}I_{l_i} + m_i[({}^{i}r_{i,ci}^{T}\,{}^{i}r_{i,ci})E_3 - {}^{i}r_{i,ci}\,{}^{i}r_{i,ci}^{T}] , \qquad i = 1, ..., 7 \tag{5.5} $$

$$ {}^{w}I_{l_i} = \frac{1}{m_i}\,{}^{w}R_i\,{}^{w}\bar{I}_{l_i}\,{}^{w}R_i^{T} , \qquad i = 1, ..., 7 \tag{5.6} $$

Parameter	Value	Parameter	Value
${}^{w}c_{1x}$	0.0215581	${}^{w}c_{2x}$	-0.000409175
${}^{w}c_{1y}$	-0.03758	${}^{w}c_{2y}$	0.00422142
${}^{w}c_{1z}$	0.3205	${}^{w}c_{2z}$	0.392865
${}^{w}c_{3x}$	-0.000340033	${}^{w}c_{4x}$	0.00138418
${}^{w}c_{3y}$	0.000208457	${}^{w}c_{4y}$	0.0001543
${}^{w}c_{3z}$	0.596757	${}^{w}c_{4z}$	0.859081
${}^{w}c_{5x}$	0.000298448	${}^{w}c_{6x}$	-0.0021546
${}^{w}c_{5y}$	-0.0100173	${}^{w}c_{6y}$	-0.0585103
${}^{w}c_{5z}$	1.04515	${}^{w}c_{6z}$	1.0557
${}^{w}c_{7x}$	0.00037943		
${}^{w}c_{7y}$	-0.00137072		
${}^{w}c_{7z}$	1.17206		

Table 5.1: Values of dynamic parameters I

The resulting values are

Parameter	Value	Parameter	Value
${}^{w}I_{1xx}$	$3.979 * 10^{-3}$	${}^{w}I_{2xx}$	$1.783 * 10^{-2}$
${}^{w}I_{1yy}$	$3.162 * 10^{-3}$	${}^{w}I_{2yy}$	$1.841 * 10^{-2}$
${}^{w}I_{1zz}$	$2.508 * 10^{-3}$	${}^{w}I_{2zz}$	$3.894 * 10^{-4}$
${}^{w}I_{1xy}$	$-1.694 * 10^{-3}$	${}^{w}I_{2xy}$	$-9.225 * 10^{-8}$
${}^{w}I_{1xz}$	$-2.393 * 10^{-3}$	${}^{w}I_{2xz}$	$-7.734 * 10^{-8}$
${}^{w}I_{1yz}$	$2.413 * 10^{-3}$	${}^{w}I_{2yz}$	$-1.262 * 10^{-4}$
${}^{w}I_{3xx}$	$2.577 * 10^{-2}$	${}^{w}I_{4xx}$	$2.729 * 10^{-2}$
${}^{w}I_{3yy}$	$2.660 * 10^{-2}$	${}^{w}I_{4yy}$	$2.513 * 10^{-2}$
${}^{w}I_{3zz}$	$2.281 * 10^{-4}$	${}^{w}I_{4zz}$	$4.191 * 10^{-3}$
${}^{w}I_{3xy}$	$1.118 * 10^{-4}$	${}^{w}I_{4xy}$	$-2.173 * 10^{-4}$
${}^{w}I_{3xz}$	$-1.872 * 10^{-4}$	${}^{w}I_{4xz}$	$-2.177 * 10^{-4}$
${}^{w}I_{3yz}$	$-1.463 * 10^{-4}$	${}^{w}I_{4yz}$	$2.230 * 10^{-4}$
${}^{w}I_{5xx}$	$6.878 * 10^{-3}$	${}^{w}I_{6xx}$	$6.760 * 10^{-3}$
${}^{w}I_{5yy}$	$7.341 * 10^{-3}$	${}^{w}I_{6yy}$	$2.011 * 10^{-3}$
${}^{w}I_{5zz}$	$1.435 * 10^{-3}$	${}^{w}I_{6zz}$	$1.216 * 10^{-2}$
${}^{w}I_{5xy}$	$-5.037 * 10^{-6}$	${}^{w}I_{6xy}$	$-1.585 * 10^{-6}$
${}^{w}I_{5xz}$	$1.356 * 10^{-4}$	${}^{w}I_{6xz}$	$8.245 * 10^{-7}$
${}^{w}I_{5yz}$	$-3.220 * 10^{-3}$	${}^{w}I_{6yz}$	$-5.157 * 10^{-5}$
${}^{w}I_{7xx}$	$5.123 * 10^{-3}$	${}^{w}I_{7xy}$	$6.034 * 10^{-7}$
${}^{w}I_{7yy}$	$5.170 * 10^{-3}$	${}^{w}I_{7xz}$	$-1.493 * 10^{-8}$
${}^{w}I_{7zz}$	$1.839 * 10^{-4}$	${}^{w}I_{7yz}$	$-3.005 * 10^{-7}$

Table 5.2: Values of dynamic parameters II

Finally, we need to reassign the link masses to be equal to the values estimated by Claudio.

However, the given inertia matrix values given by the KUKA LWR model gives a little better results

in the simulation environment. Hence, we have changed only the position of the center of mass and the mass value of each link in the V-REP.

One problem in implementing Newton Euler approach is determining ${}^{i}r_{i-1,i}$ since it needs to be obtained graphically by drawing the frames. In our case, it has the following values

Parameter	Value	Parameter	Value
${}^{1}r_{0,1}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	${}^{2}r_{1,2}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
${}^{3}r_{2,3}$	$\begin{bmatrix}0 & -d3 & 0\end{bmatrix}^T$	${}^{4}r_{3,4}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
${}^{5}r_{4,5}$	$\begin{bmatrix}0 & d5 & 0\end{bmatrix}^T$	${}^{6}r_{5,6}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
${}^{7}r_{6,7}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$		

Table 5.3: Values of ${}^{i}r_{i-1,i}$

In the case that we want to express the external forces in the world frame

$$ {}^{w}f_{ext} = {}^{w}R_0 \, {}^{0}f_{ext} \tag{5.7} $$

Which will be needed so that the reaction strategy doesn't violate the RCM constraints. Hence, we will express the reaction forces as a change in the position of the trocar w.r.t. the world frame so that we can use the same velocity control law, which uses the null space method, to achieve the reaction strategy.

5.2 Kinematic Control System

The algorithm was implemented in C++ and Matlab. We used V-REP to visualize the realistic simulation of the C++ algorithm running on the available KUKA LWR robot. The same simulation was also done in matlab, and we used matlab to visualize the simulation results.

In V-REP, as already stated, the reference frames were not consistent with the paper, so we changed them inside the simulator to obtain the same convention.

Moreover, we have to consider the transformation between the world frame and the robot base frame since the trocar point is defined w.r.t. the world frame. The transformation matrix between the world frame and the robot base frame in our V-rep simulation is

$$ {}^{w}A_0 = \begin{bmatrix} -1 & 0 & 0 & -0.0000807 \\ 0 & -1 & 0 & 0.0001082 \\ 0 & 0 & 1 & 0.3105 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8} $$

After fixing the reference frames, calculating the kinematic of the robot and defining the control law, we set the desired position of the RCM at [0.5 0.5 0].

For simplicity, we removed the collision between the robot and the floor in V-REP, in order to be able to simulate the patient's body (i.e., we used the floor as the patient's stomach for example to specify $\mathbf{P}_{\text{RCM}}$ position, which is equal to $\mathbf{P}_{\text{trocar}}$). After removing the collision, the robot can go to the specified point and then translate along the axis aligned with the tool attached to the end effector simulating the entry of the tool through the incision point. As seen in **Figure 5.1**, the robot starts at a singularity, which is the initial configuration $\mathbf{q}_0$ and have to go to the desired position, which is achieved as seen in the numerical data in **Figures 5.2** and **5.3**
Below is the result obtained with our implementation using V-REP simulation for the extended task.

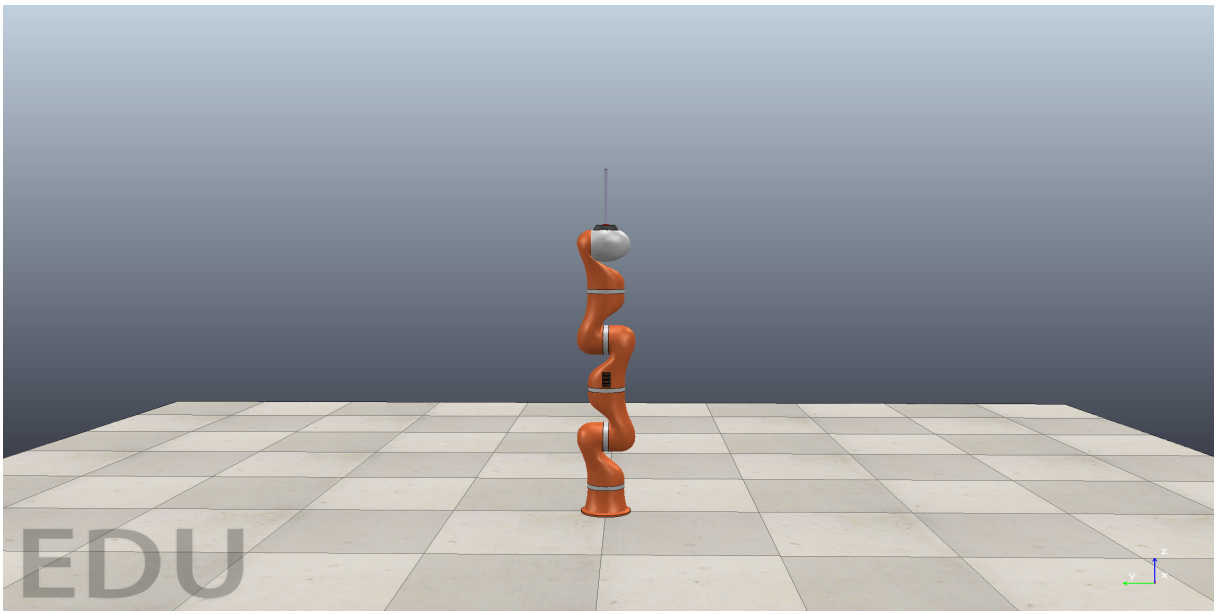


Figure 5.1: Initial position of the robot

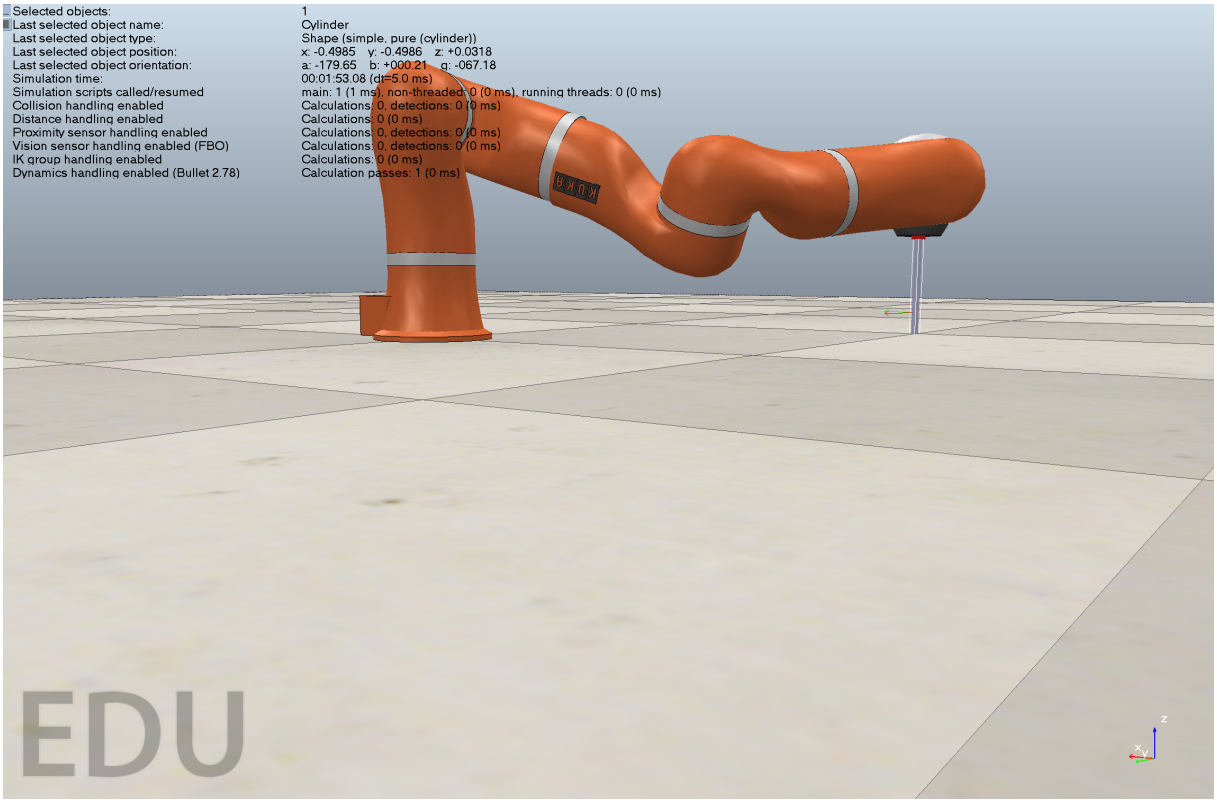


Figure 5.2: final position of the robot

In **Figure 5.5**, we can see the simulation in matlab
As shown in the previous figures, the primary task was achieved as well as second task, where the robot changed its configuration to achieve the desired one and then translated to simulate the surgical tool insertion in the human body. This was achieved thanks to the control law that assured the convergence of the error and the usage of the null space method to extend the task, which also assured the convergence of the second task while satisfying the primary one.

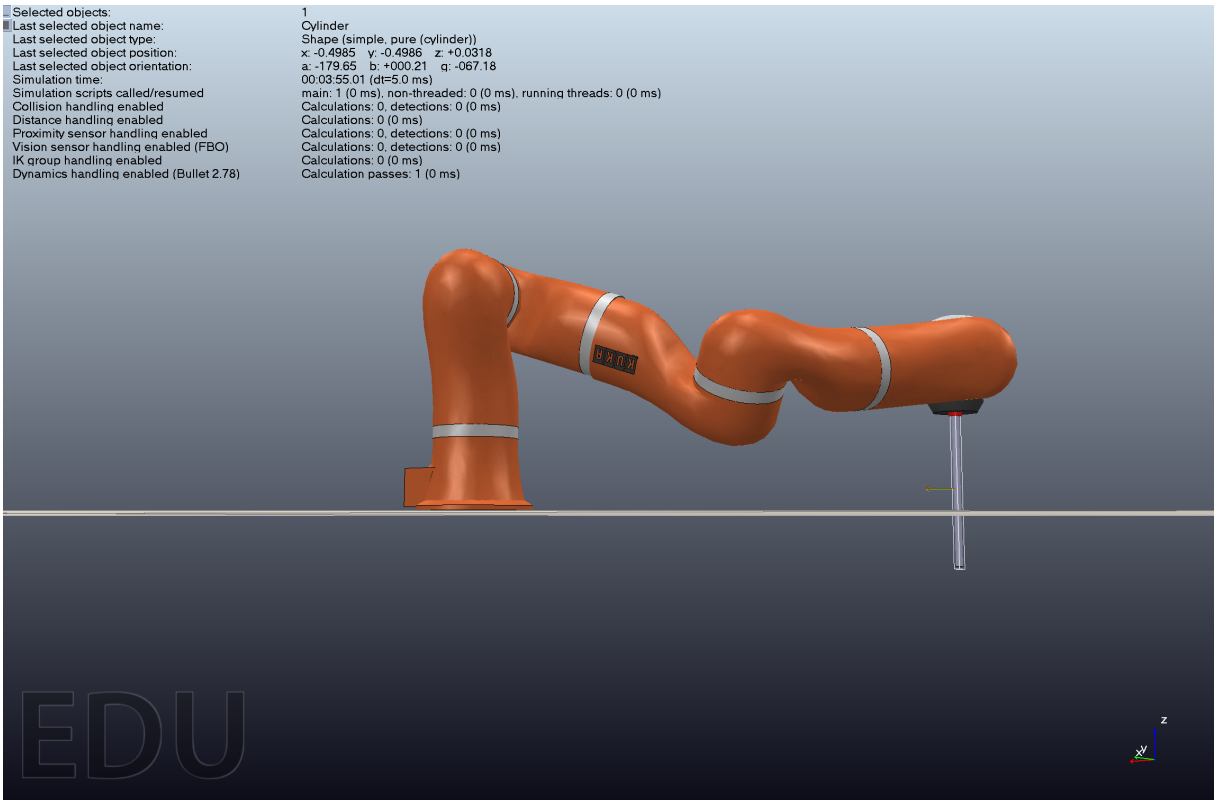


Figure 5.3: final position of the robot showing the tool inside the patient's body

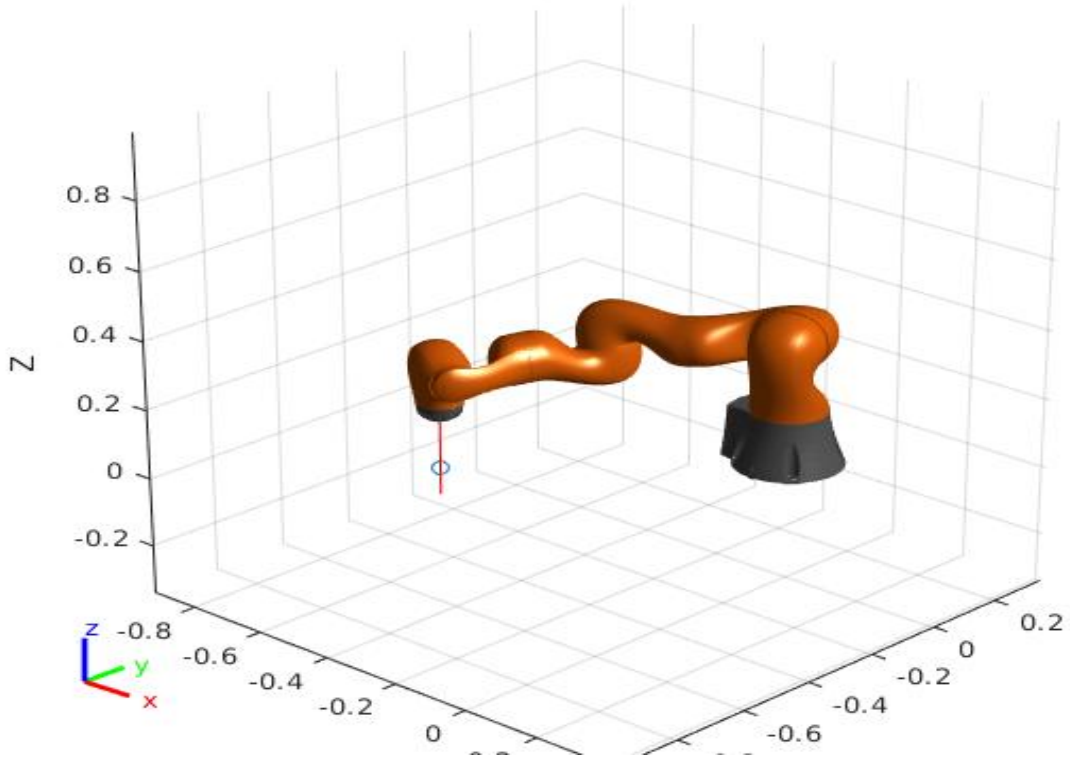


Figure 5.4: final position of the robot simulated in matlab

5.3 External Forces Estimation

Building on the previously defined kinematic architecture, we added the external forces and tested the algorithms both in matlab and C++ getting the same results, which provides a strong base for future works, where we would like to run it also in V-REP.

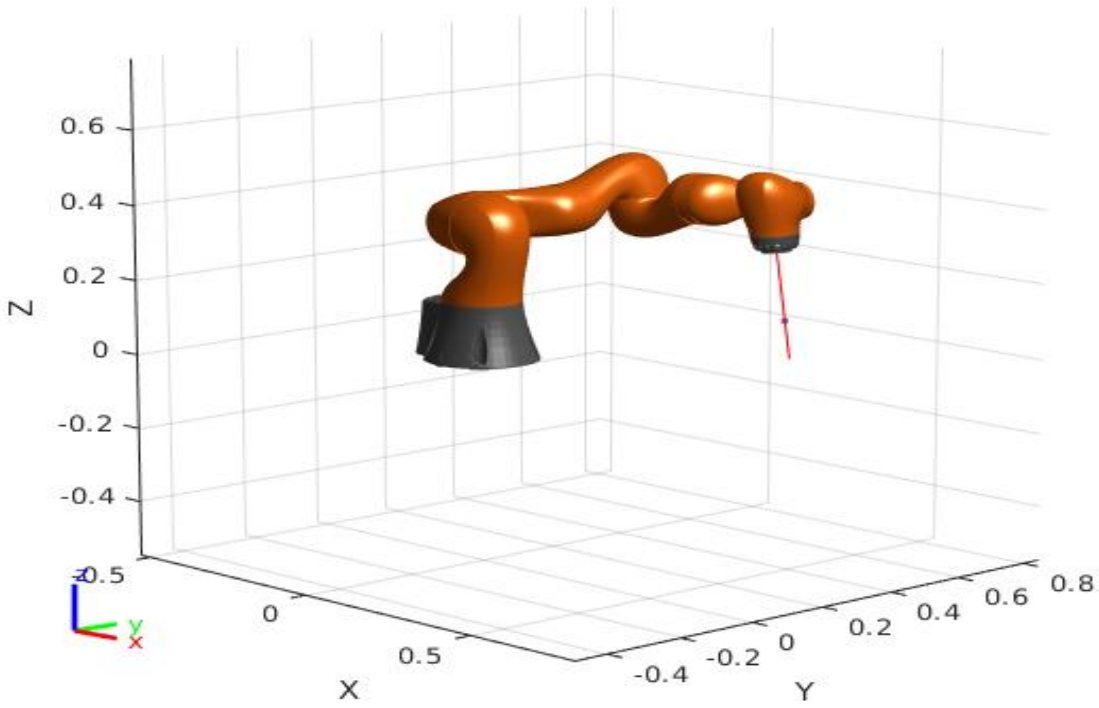


Figure 5.5: External force estimation in matlab

Chapter 6

Results

6.1 Force Estimation Results

6.1.1 Matlab

In this section we are going to see the results of the estimated external forces calculated using the residual method with respect to the true external forces we added to simulate the patient breathing and heart beating.

Knowing that x-axis is the number of simulated iterations, and y-axis is the value of external torque applied vs. the estimated residual in Newton.

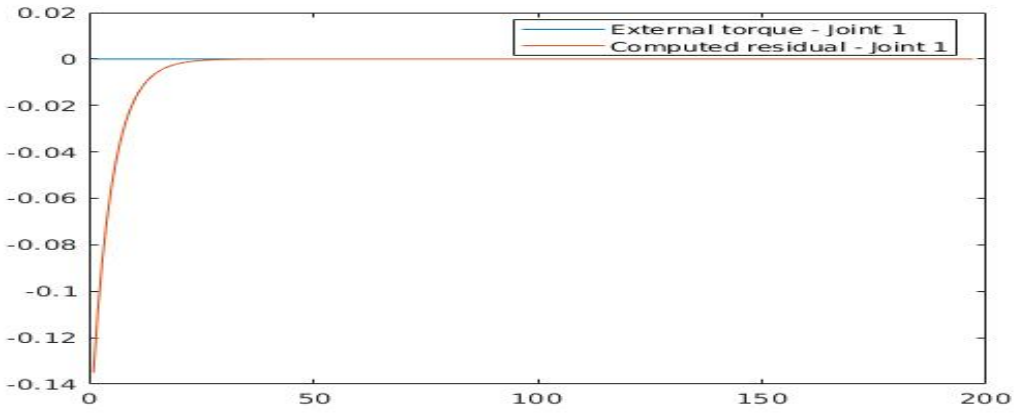


Figure 6.1: External torque vs. Computed residual for Joint 1

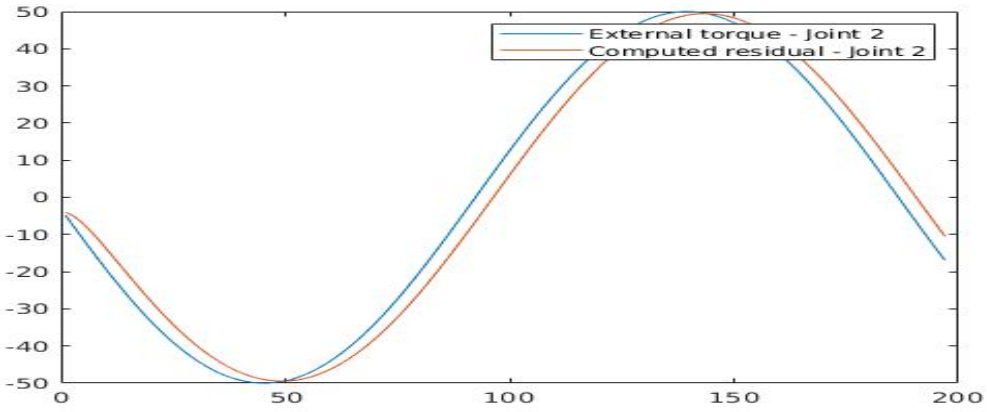


Figure 6.2: External torque vs. Computed residual for Joint 2

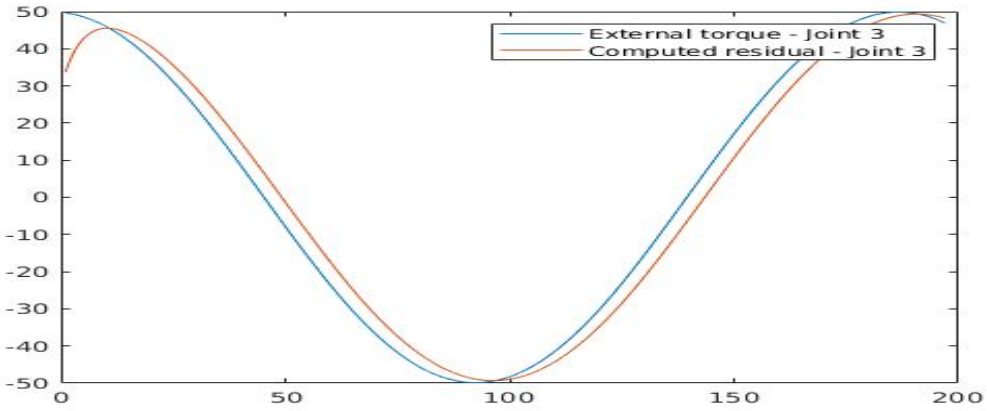


Figure 6.3: External torque vs. Computed residual for Joint 3

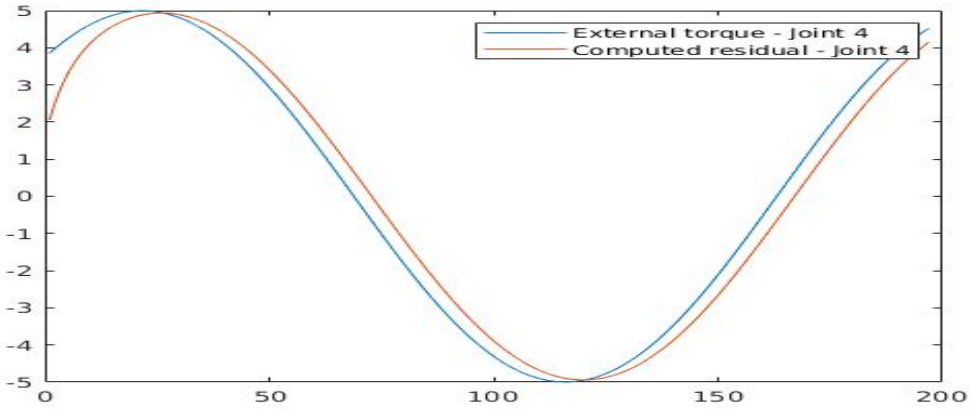


Figure 6.4: External torque vs. Computed residual for Joint 4

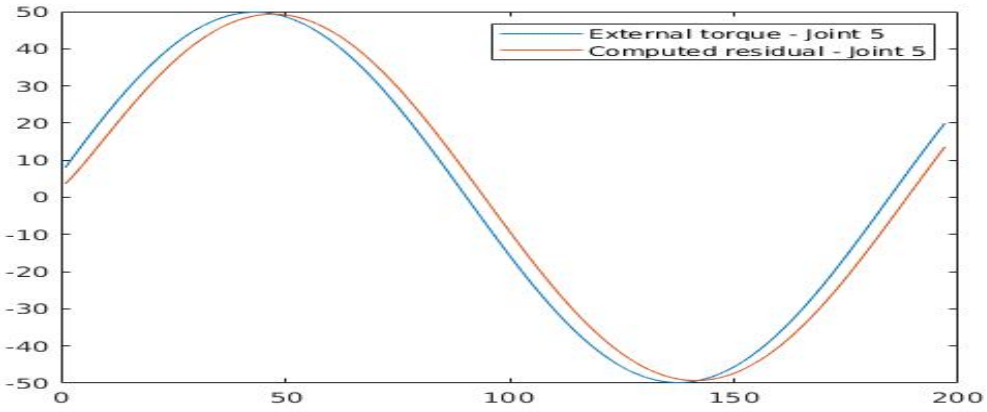


Figure 6.5: External torque vs. Computed residual for Joint 5

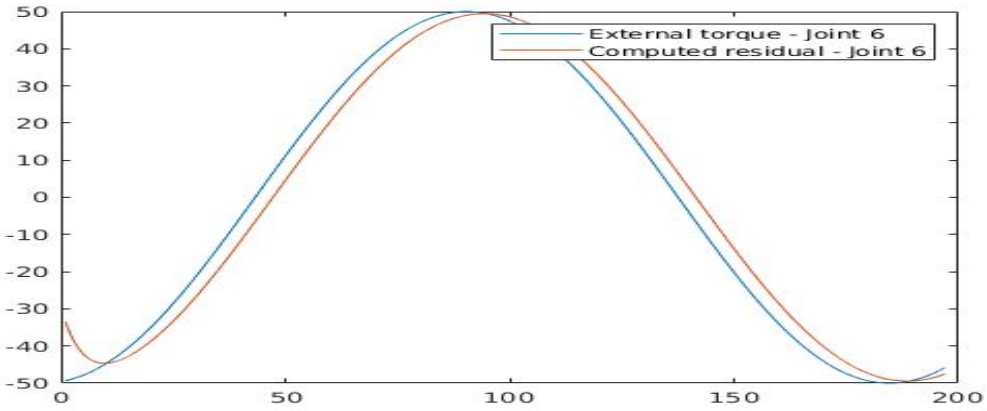


Figure 6.6: External torque vs. Computed residual for Joint 6

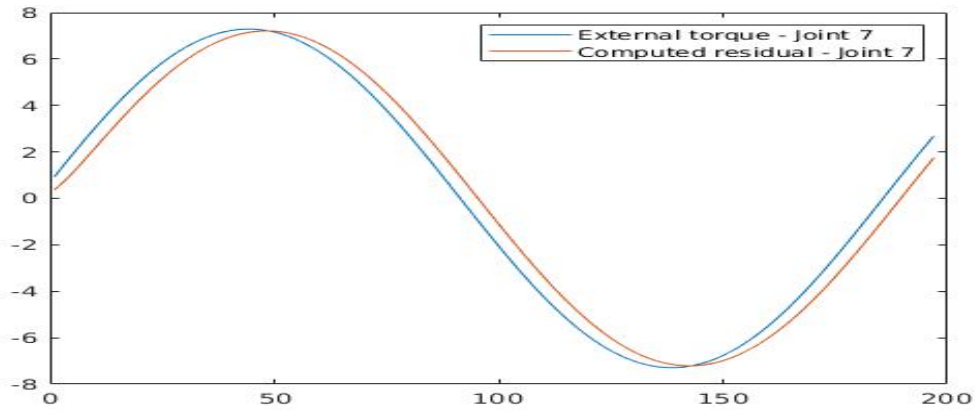


Figure 6.7: External torque vs. Computed residual for Joint 7

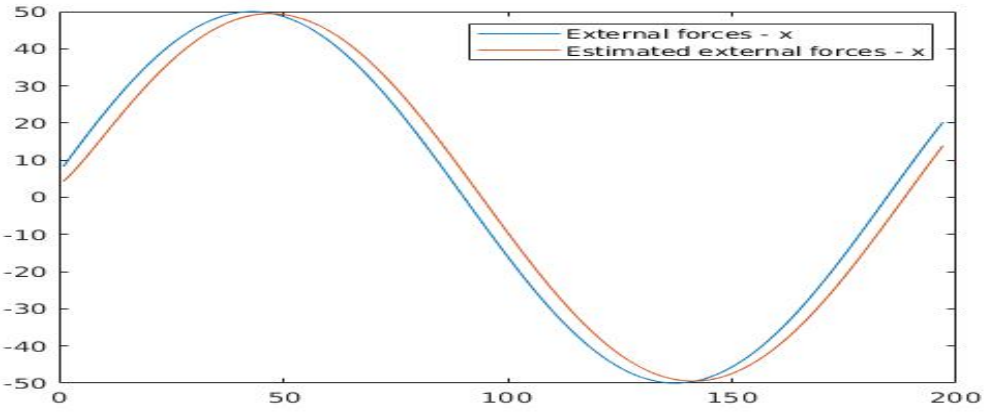


Figure 6.8: External force vs. estimation - x Component

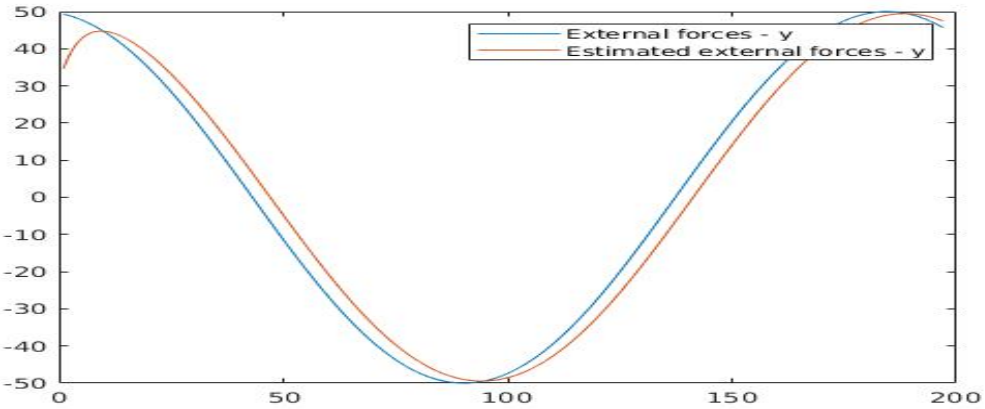


Figure 6.9: External force vs. estimation - y Component

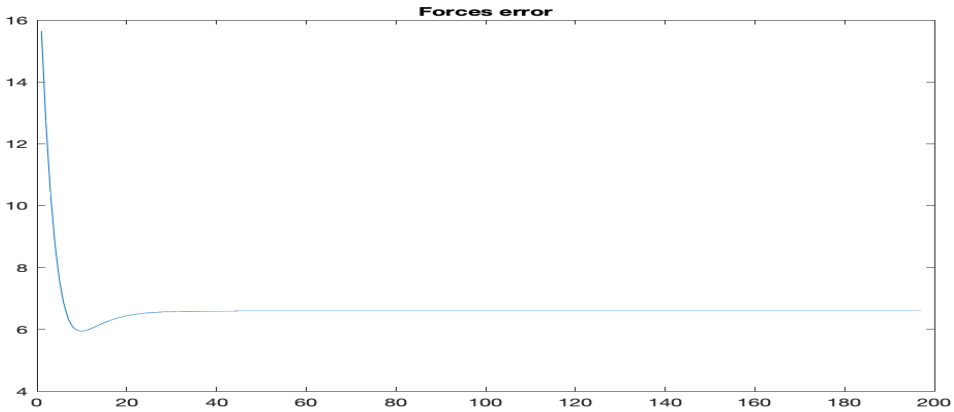
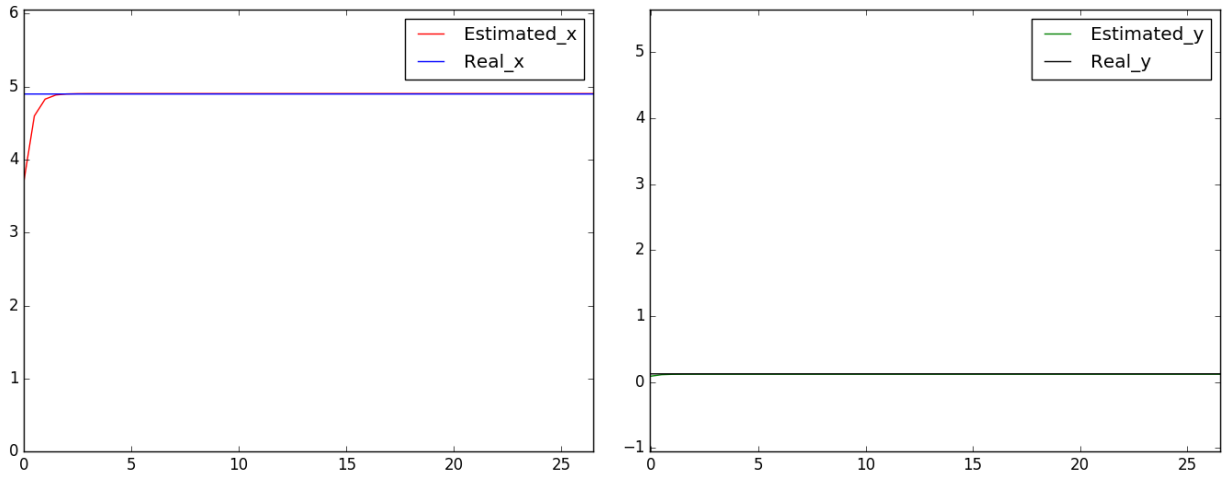


Figure 6.10: Forces error

As we have seen, the estimated external forces calculated by the residual method converged on all of the 7 joints. The initial estimation is a little bit different from the truth value, which is expected since this is an approximation, but after a small time, the residual method gives almost the same force

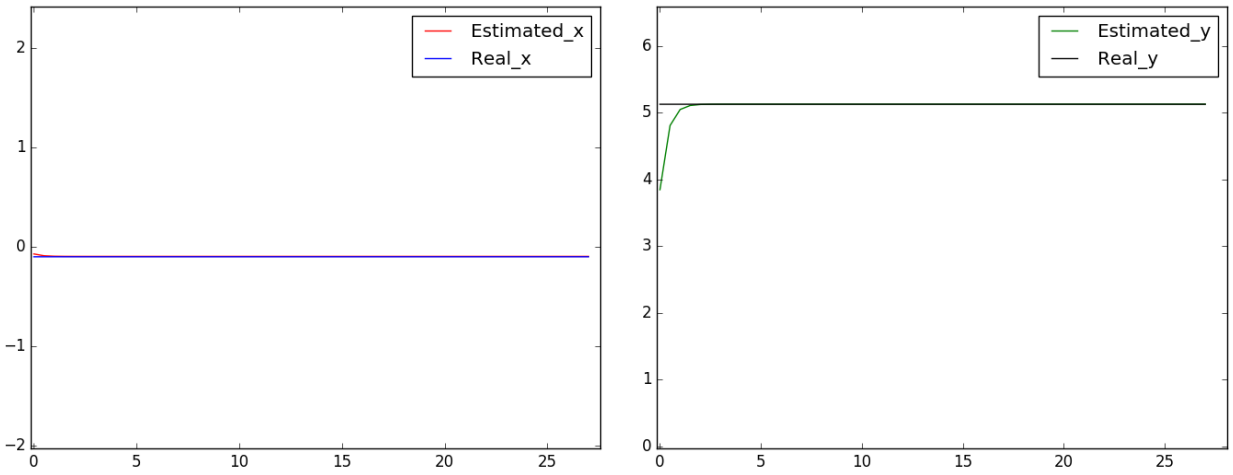
acting on the system, which tells us that the results we got using this method is reliable and can be used for our task with a very high accuracy.

6.1.2 V-REP



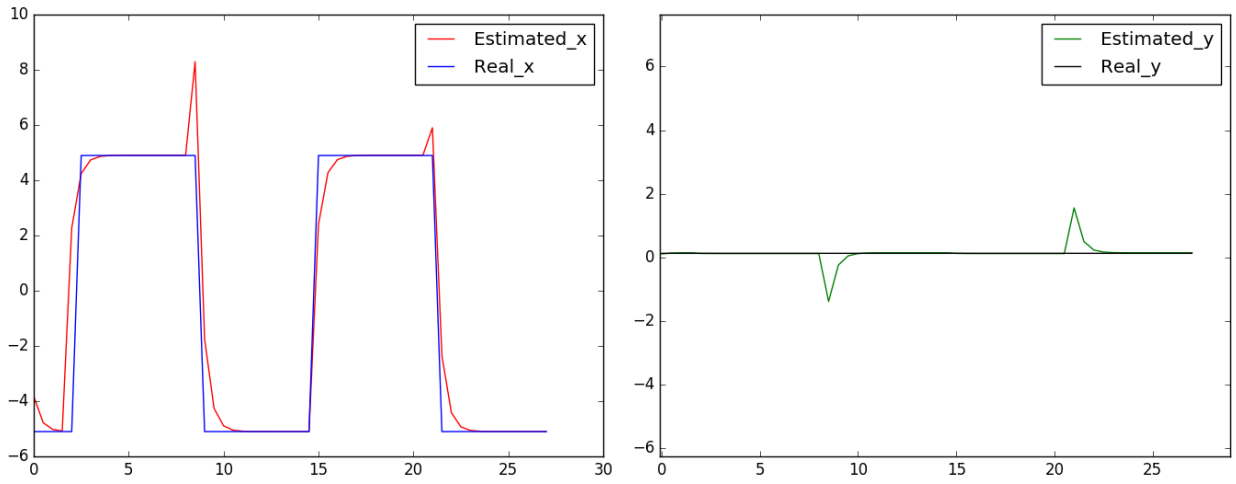
(a) X-component (b) Y-component

Figure 6.11: Estimated vs. real value of constant force on X direction



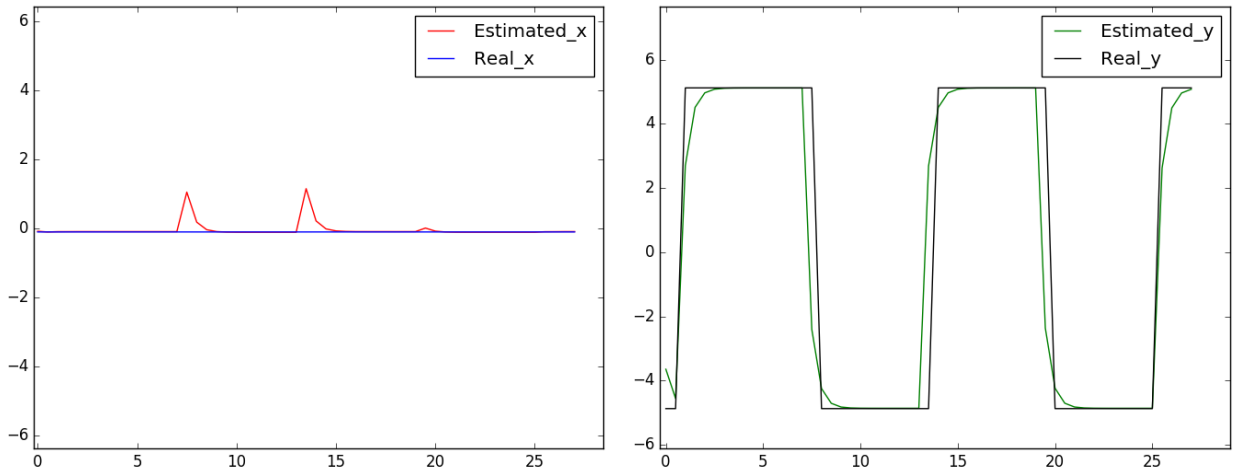
(a) X-component (b) Y-component

Figure 6.12: Estimated vs. real value of constant force on Y direction



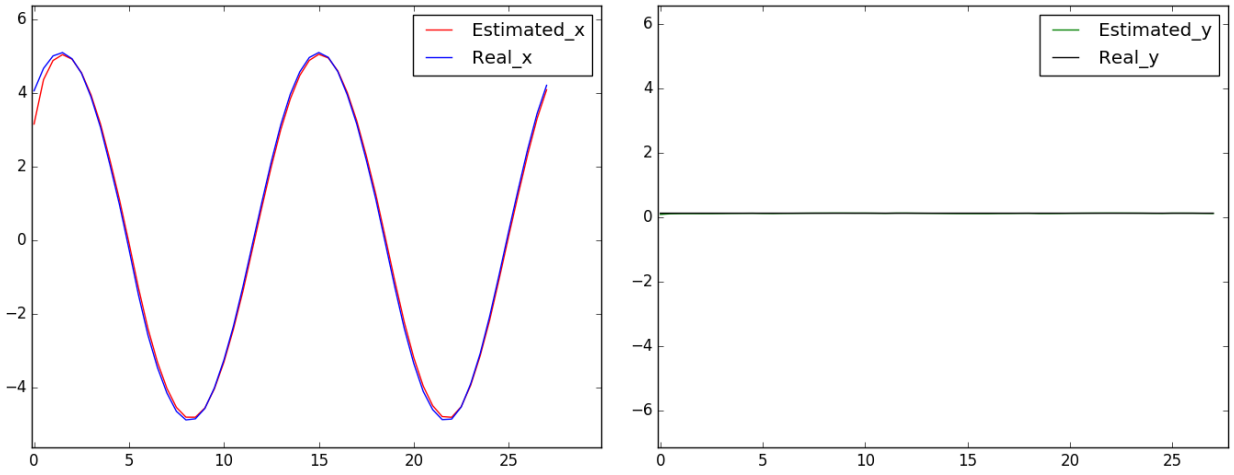
(a) X-component (b) Y-component

Figure 6.13: Estimated vs. real value of square-like force on X direction



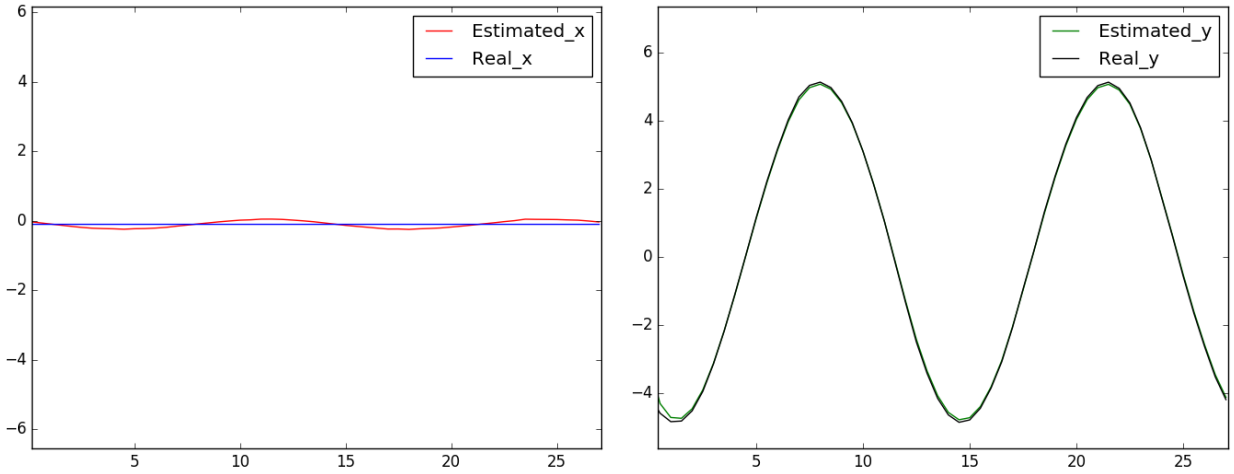
(a) X-component (b) Y-component

Figure 6.14: Estimated vs. real value of square-like force on Y direction



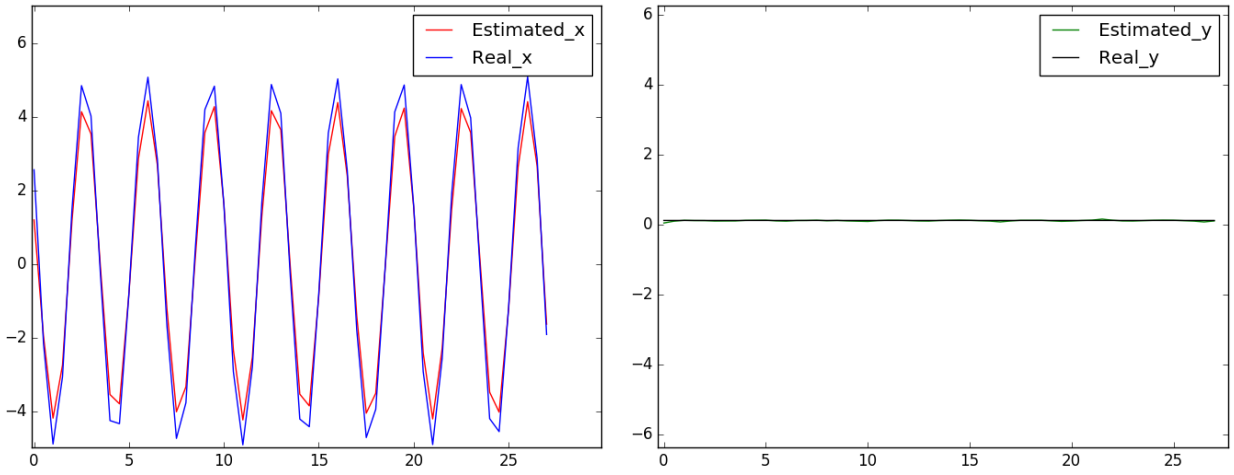
(a) X-component (b) Y-component

Figure 6.15: Estimated vs. real value of sin-like force on X direction



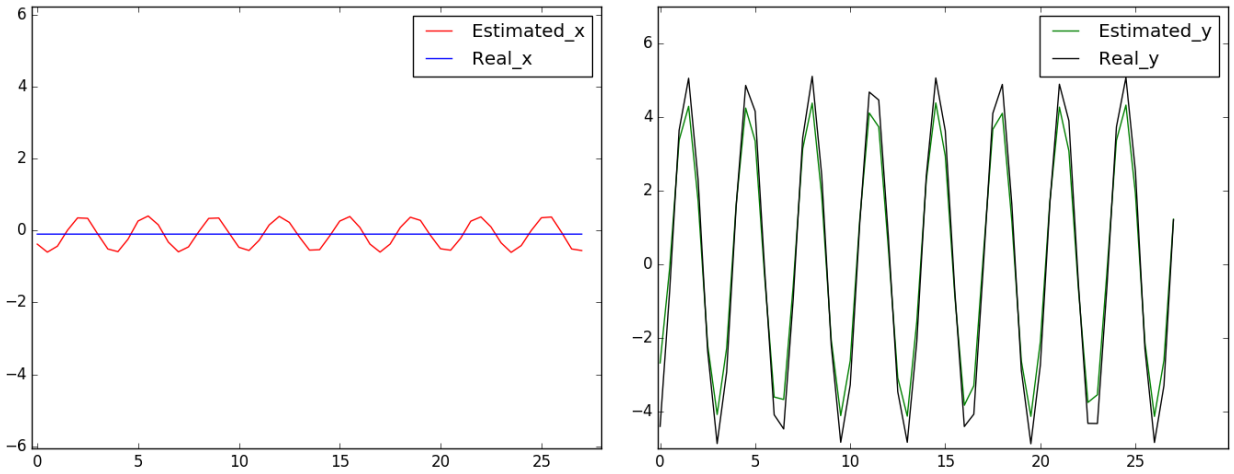
(a) X-component (b) Y-component

Figure 6.16: Estimated vs. real value of sin-like force on Y direction



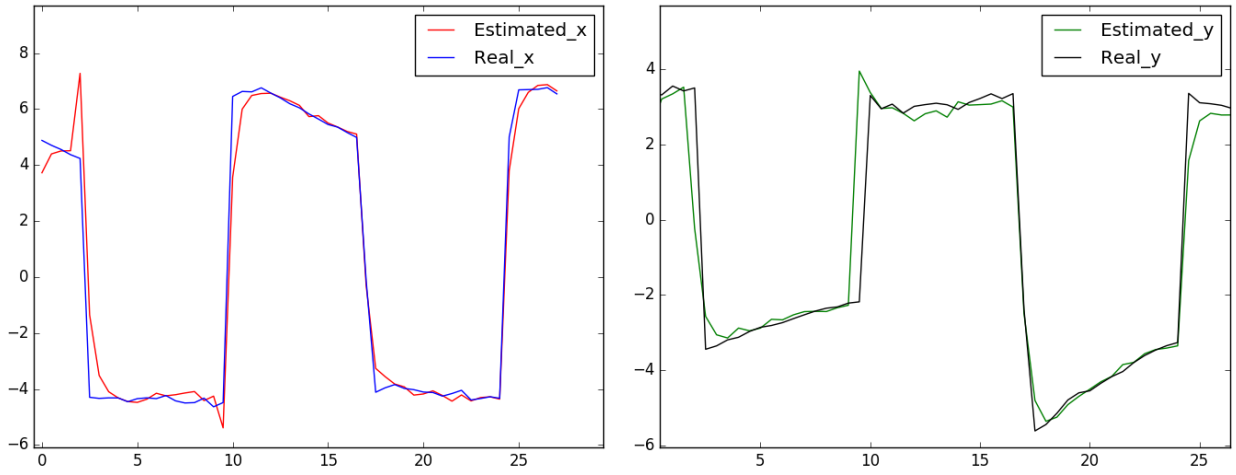
(a) X-component (b) Y-component

Figure 6.17: Estimated vs. real value of HF sin-like force on X direction



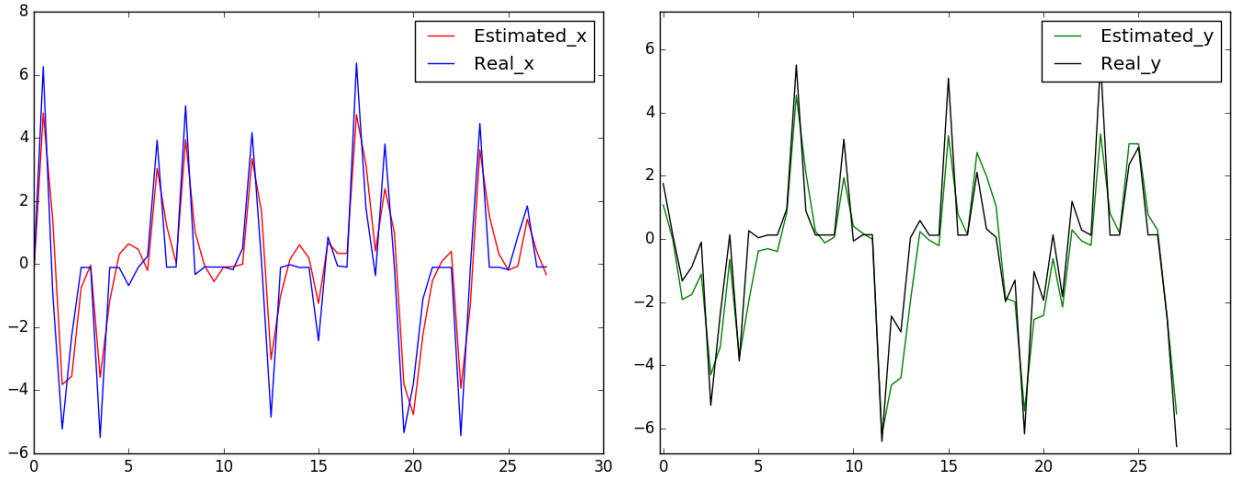
(a) X-component (b) Y-component

Figure 6.18: Estimated vs. real value of HF sin-like force on Y direction



(a) X-component (b) Y-component

Figure 6.19: Estimated vs. real value of forces come from colliding between trocar and surgical tool with no reaction strategy



(a) X-component (b) Y-component

Figure 6.20: Estimated vs. real value of forces come from colliding between trocar and surgical tool with simple reaction strategy

Chapter 7

Conclusions

In this work, we have followed the work in [8] and implemented a kinematic control strategy for robot assistant in minimally invasive surgeries in V-REP using the KUKA LWR 4+.

This project was two folds. The first one was to introduce a kinematically constrained varying remote center of motion (RCM) that lies on the extended link attached to the end effector of the robot, which should always be equal to the incision point (Trocar point). The second one was to build the dynamic model of the robot and detect external forces acting on the system during the operation, such as the heart beat and breathing to compensate these forces, which helps preventing possible damage to the patient's body.

To estimate the external forces, we used the residual method, which takes as input the dynamic data from the dynamic model of the robot and returns the estimated forces rising from the contact with external objects.

After building the model, we evaluated the accuracy of the methods we implemented by analyzing a set of numerical results.

As previously shown, the results we obtained are very good and realize well what we wanted to achieve showing that it is feasible to use a programmable RCM, which is more reliable than mechanically constrained RCM in the sense of using it for minimally invasive surgeries as well as open surgeries and dynamically extending and achieving new tasks.

Using robots in minimally invasive surgeries revolutionized the medical industry, because it reduces the human error by a big factor. Moreover, using robots in the medical field provides a faster and more convenient way of performing surgeries, which increases the number of daily allowed operations while keeping the same high level of efficiency and accuracy.

There is a small drawback in using this approach, which is the cost of setting the environment and training the medical staff, but this is an initial cost that is worth considering given the increasing need of robots assisting the doctors performing increasingly difficult operations and more complex operations.

Bibliography

[1] Pham CD, Coutinho F, Leite AC, Lizarralde F, From PJ, Johansson R. Analysis of a moving remote center of motion for robotics-assisted minimally invasive surgery. In2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2015 Sep (pp. 1440-1446). IEEE.

[2] Sandoval J, Poisson G, Vieyres P. Improved dynamic formulation for decoupled cartesian admittance control and RCM constraint. In2016 IEEE International Conference on Robotics and Automation (ICRA) 2016 May 16 (pp. 1124-1129). IEEE.

[3] Sandoval J, Poisson G, Vieyres P. A new kinematic formulation of the RCM constraint for redundant torque-controlled robots. In2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2017 Sep 24 (pp. 4576-4581). IEEE.

[4] Marinho MM, Harada K, Mitsuishi M. Comparison of remote center-of-motion generation algorithms. In2017 IEEE/SICE International Symposium on System Integration (SII) 2017 Dec 11 (pp. 668-673). IEEE.

[5] Sandoval J, Vieyres P, Poisson G. Generalized framework for control of redundant manipulators in robot-assisted Minimally Invasive Surgery. IRBM. 2018 Jun 1;39(3):160-6.

[6] Su H, Sandoval J, Makhdoomi M, Ferrigno G, De Momi E. Safety-enhanced human-robot interaction control of redundant robot for teleoperated minimally invasive surgery. In2018 IEEE International Conference on Robotics and Automation (ICRA) 2018 May 21 (pp. 6611-6616). IEEE.

[7] Su H, Yang C, Ferrigno G, De Momi E. Improved Human–Robot Collaborative Control of Redundant Robot for Teleoperated Minimally Invasive Surgery. IEEE Robotics and Automation Letters. 2019 Feb 4;4(2):1447-53.

[8] Aghakhani N, Geravand M, Shahriari N, Vendittelli M, Oriolo G. Task control with remote center of motion constraint for minimally invasive robotic surgery. In2013 IEEE international conference on robotics and automation 2013 May 6 (pp. 5807-5812). IEEE.

[9] Gaz C, Flacco F, De Luca A. Extracting feasible robot parameters from dynamic coefficients using nonlinear optimization methods. In 2016 IEEE international conference on robotics and automation (ICRA) 2016 May 16 (pp. 2075-2081). IEEE.

[10] Haddadin S, De Luca A, Albu-Schäffer A. Robot collisions: A survey on detection, isolation, and identification. IEEE Transactions on Robotics. 2017 Oct 5;33(6):1292-1312.

[11] Siciliano B, Sciavicco L, Villani L, Oriolo G. Robotics: modelling, planning and control. Springer Science & Business Media; 2010 Aug 20.