École Centrale de Nantes

Master ARIA - ROBA

Thesis Report

Definition of an active deformation model for manipulation of flexible objects

Author:
Vyshakh Palli Thazha

Supervisors:
Sébastien Briot
Abdelhamid Chriette
Philippe Martinet

President:
Olivier Kermorgant

July 20, 2016

Contents

List of Figures

1 Acknowledgments

The research described in this thesis was carried out from November 2015 to July 2016 as a part of the Robotics (ARMEN) team of the "Institut de Recherche en Communications et Cybernétique de Nantes" (**IRCCyN**) in Nantes. The robotics master was financially supported by the European Union, under the **Erasmus Mundus program** called **Interweave**.

First of all, I am deeply indebted to my supervisor Professor **Sébastien BRIOT** for pushing and motivating me to produce results even when I was going through a tough time, my co-supervisors Professor **Philippe MARTINET**, and Professor **Abdelhamid CHRIETTE** for their encouragement, valuable advises, continuous and indispensable assistance, and their great help. I would like to thank them for their helpful guidance and great care. The numerous discussions and the motivating investment in the research surrounding my thesis is more than greatly appreciated. It is really my pride to perform my Masters research thesis under their supervision.

I would like also to express my gratitude to Professor **Olivier KERMORGANT** to have honored me for presiding the jury and for his suggestions allowing me to enforce certain perspectives in my research subject. Also I would like to thank Professor **Ina TARALOVA** for her suggestions and critical remarks on my work which helped me improve my report and the work that was put into it.

I would like to thank my colleagues who have, all throughout my thesis, helped with various aspects of my work and for supporting me to succeed in the whole research experience and enjoying it at the same time.

I would like to deeply thank my friends in Nantes, and around Europe, especially from University of Genoa and from around the world for the unforgettable moments we spent together. Finally, I owe a huge debt of gratitude and sincere thanks to my parents and family, they have lost a lot due to my research abroad. Without their encouragement and support it would have been impossible for me to finish this work. Thank you.

Vyshakh PALLI THAZHA

2 Abbreviations

MRI - Magnetic resonance imaging

CT - Computed Tomography

NURBS - Non Uniform Rational B-Splines

DoF - Degrees of Freedom

FEM - Finite Element Method

BEM - Boundary Element Method

CV - Computer Vision

CoM - Center of Mass

A* - A-Star

PRM - Probabilistic Roadmap Method

3 Abstract

Manipulation of flexible objects plays an important rule in the industry. Any industry that involves the use of a deformable object will, at one certain point of time, require robots to handle these objects. Be it handling of food items in the food industry, folding or sorting of clothes in the textile industry, or handling of cables in various industries etc.. every where there arises a need to study how these objects could be manipulated. So modeling of objects has to be studied, using the developed model to plan trajectories and motions for a robot to handle this object has to be studies, vision based control to correctly manipulate the object using the developed motion planning strategies has to be developed. This master thesis is a first step into the study of an articulated object and its manipulation.

The modeling of the object is studied which is used to find the equilibrium configurations and hence the configuration space of the object. The configuration space is studied and path planning algorithms are used to generate trajectories from one equilibrium configuration to another.

The manipulation of the object using two 2R planar robots is studied. For this, task-priority based control of redundant manipulators is studies and implemented. Simulations and results are documented.

4 Introduction

Contents

4.1 Topic of Research

Robots and robotic technologies have been evolving at a fast pace over the past few decades and there is a wide range of robots specialized for different application like manipulation, navigation, surveillance, rescue, medical treatment and the list goes on. Manipulation of real world objects has been an important area of research for years and robots which can manipulate rigid objects are very common in industries around the globe. Increased productivity, lowered cost and better precision is obtained by using robots. But when it comes to manipulation of soft or flexible objects, the robots are still not capable enough and the technologies which enable to manipulate a soft object with the dexterity of a human hand is still to be developed. Because of this inability of perfect manipulation, the research topic of developing the perfect technology for doing the same has been picking up pace and is a hot topic for developments at this point of time. Several methods have been proposed which try to solve this and although many of them succeed in its own way, they have their own drawback making their application to a wide variety of objects impossible. Some methods even though provides good results are computationally expensive while some others have some other constraints.

Deformable objects are found in a wide range of industries like food industry, garments, packaging, manufacturing, industries involving elastic objects and the list goes on. Deformable objects are hard to manipulate because of their low stiffness and they are susceptible to large deformations on application of force. They change their shape and volume during handling. Thus a static model of the object cannot be used like in the cases of rigid objects. Model based and model-free methods have been studied for manipulation of deformable objects. The state of the art will be detailed in the coming sections.

4.2 Organization of the report

The rest of the report is organized as follows. In Section 4.3, the general techniques that are used to model deformable objects are specified. In Section 4.4, the state of the art in deformable object manipulation is discussed. In Section 5 the object that is studied is described and elaborated. In Section 6 the modeling of the already defined object and Lagrange formulations of modeling is elaborated. In Section 7 and Section 8, the modeling technique is extended to objected with 3 and 4 connected links respectively. In Section 9 the configuration space of the modeled object is defined and in section 10 the configuration space is represented as a grid map. In Section 11, trajectory generation techniques are studied and applied to the object manipulation. Shortest path algorithms like A-star algorithm and Probabilistic Roadmap Method is elaborated. In Section 12 the effect on gravity angle to the equilibrium state of the object is studied. In Section 13 the modeling of robots, redundant manipulation and application of 2R robots for manipulating the proposed object is studied. In Section 14 a task-priority approach for redundant manipulators is studied and results are plotted for different conditions and gains are tuned.

4.3 Modeling of deformable objects

Deformable objects can be one modeled either in one dimension (lines and curves), two dimensions (surfaces) or three dimensions (solid objects). These models are used for different applications such as animations, image segmentation, 3D reconstruction of bones and organs from MRI or CT scans, haptics, mechanical simulations, surgery planning etc. [16]

All these applications are diverse and the utilization of one single modeling method for all these is not appropriate and over the years a variety of methods have been developed and implemented for the same. The deformable models can be classified into three categories. Heuristic methods, Continuum-mechanical models, and hybrid methods which combine both the previous categories.

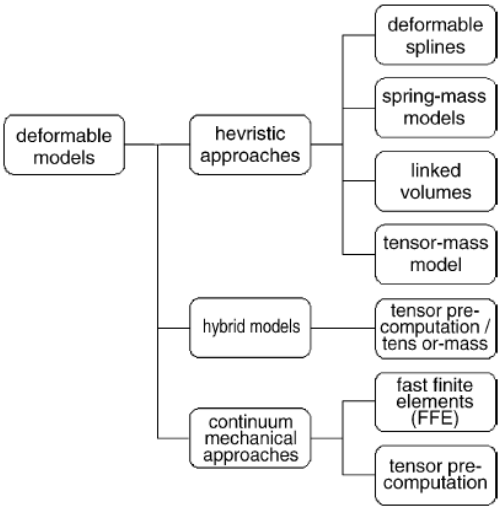


Figure 1: Different categories of deformable object modeling - Adopted from [16]

The heuristic approaches use straightforward techniques of geometry of the deformable objects including their elastic properties. In the beginning these methods used to model solid objects as hollow shells because most of the applications focused on deformation on the surface level. This reduced the size to less than a hundredth of its initial size. But this approximation neglects the conservation of its volume. So later, methods like spring-mass models were extended to include changes in volume - linked volume algorithms, tensor-mass model which used tetrahedral elements were developed.

The continuum-mechanical approaches are more exact but they are far too complex and computationally costly. This category has methods like Finite Element method (FEM) which involves a discretization of the entire object and Boundary Element

(BEM) approach which maps the calculations of the interior of the object to its surface, thereby requiring discretization of only the surface.
The hybrid approach divides the object into parts and finds the best possible approach for each part bases on how this part is interacted with.

4.4 Manipulation of Deformable objects

A successful attempt of manipulating deformable object using indirect positioning is presented in [23]. They use a crude model of the system and a robust control system to nullify the inadequacies of the model.

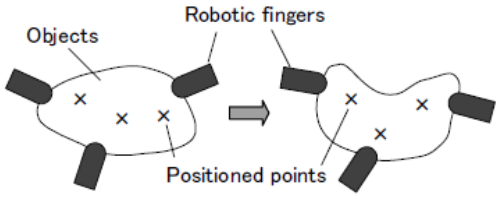


Figure 2: Indirect positioning of deformable objects adopted from [23]

Model is made for two dimensional objects by discretizing into mesh-points and each mesh point being connected by horizontal, vertical and diagonal springs as shown in Fig.3. The model describes translation, orientation and deformation of the object. Position vector of the (i, j)-th mesh point is defined as $p_{i,j} = [x_{i,j}, y_{i,j}]^T$ $(i = 0, \ldots, M; j = 0, \ldots N)$. Coefficients k_x, k_y, k_θ are spring constants of horizontal, vertical, and diagonal springs. Assuming that no moment is exerted on each mesh point. Then, the resultant force exerted on mesh point $p_{i,j}$ can be described as eg.4.1.

$$F_{i,j} = \sum_{k=1}^{8} F_{i,j}^{k} = -\frac{\partial U}{\partial p_{i,j}} \tag{4.1}$$

U denotes whole potential energy of the object. Then, function U can be calculated by sum of all energies of springs. U can be calculated by:

$$\frac{\partial(r_m, r_n, r_p)}{\partial r_m} - \lambda = 0 \tag{4.2}$$

$$\begin{bmatrix} \frac{\partial(r_m, r_n, r_p)}{\partial r_p} \\ \frac{\partial(r_m, r_n, r_p)}{\partial r_n} \end{bmatrix} = 0 \tag{4.3}$$

Vector r_m is defined as a vector that consists of coordinate values of the manipulated points. Vectors r_p and r_n are also defined for positioned and non-target points in the similar way. Vector λ denotes a set of forces exerted on the object at the manipulated points r_m by robotic fingers.

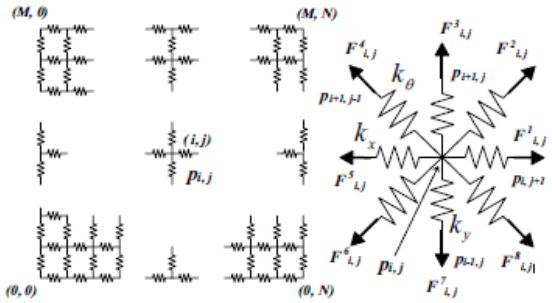


Figure 3: Spring model of deformable object [23]

The mesh points $p_{i,j}$ into the following three categories(see Fig 4) in order to formulate indirect simultaneous positioning:

- **Manipulated points:** are defined as the points that can be manipulated directly by robotic fingers.
- **Positioned points:** are defined as the points that should be positioned indirectly by controlling manipulated points appropriately.
- **Non-target points:** are defined as the all points except the above two points.

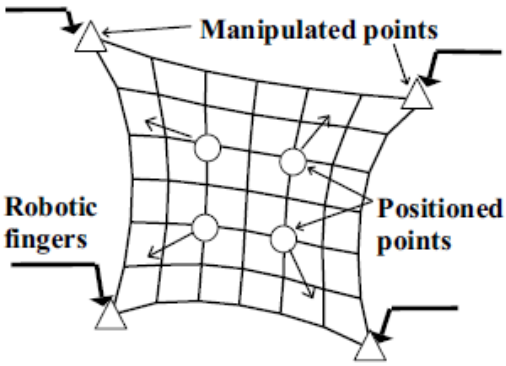


Figure 4: Classification of mesh points adopted from [23]

4.5 Control Loop

A relation between positioned points and manipulated points are obtained by linearizing 4.2 about an equilibrium point $r0 = [r_{m0}^T, r_{p0}^T, r_{n0}^T]^T$. The following equation is obtained:

$$A\delta r_m + B\delta r_n + C\delta r_p = 0 \tag{4.4}$$

where

$$A = \begin{bmatrix} \frac{\partial^2 U}{\partial r_m \partial r_p} \\ \frac{\partial^2 U}{\partial r_m \partial r_n} \end{bmatrix} |_{r0} \in R^{(2p+2n)\times 2m} \tag{4.5}$$

$$B = \begin{bmatrix} \frac{\partial^2 U}{\partial r_n \partial r_p} \\ \frac{\partial^2 U}{\partial r_n \partial r_n} \end{bmatrix} |_{r0} \in R^{(2p+2n)\times 2n} \tag{4.6}$$

$$C = \begin{bmatrix} \frac{\partial^2 U}{\partial r_p \partial r_p} \\ \frac{\partial^2 U}{\partial r_p \partial r_n} \end{bmatrix} |_{r0} \in R^{(2p+2n)\times 2p} \tag{4.7}$$

Vector $\delta r_m$is defined as an infinitesimal deviation of the manipulated points from their equilibrium points. Vectors δr_n and δr_p are defined in the similar way. 4.4 can be transformed as

$$F \begin{bmatrix} \delta r_m \\ \delta r_n \end{bmatrix} = -C\delta r_p \tag{4.8}$$

where F= [A B].

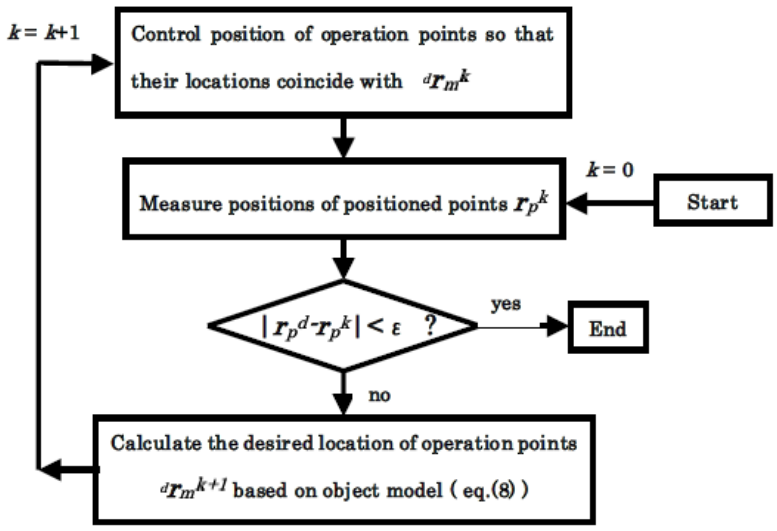


Figure 5: Flow of control method adopted from [23]

An iterative control law is developed based on this linearized equation.

$$\delta r_m = -S_U F^{-1} C \delta r_p \tag{4.9}$$
$$\delta r_n = -S_L F^{-1} C \delta r_p \tag{4.10}$$

where $S_U = [I_m \quad 0_{m\times n}]$ and $S_L = [0_{n\times m} \quad I_n]$. Let r_m^k, r_n^k and r_p^k be positions of manipulated points, those of non-target points, and those of positioned points at k-th iteration, respectively. Replacing the deviations in the previous equation with these, it becomes:

$$r_m^{k+1} = r_m^k - dS_U F_k^{-1} C_k (r_p^d - r_p^k) \tag{4.11}$$
$$r_n^k = r_n^{k-1} - dS_L F_{k-1}^{-1} C_{k-1} (r_p^{k-1} - r_p^{k-2}) \tag{4.12}$$

These equations are used in an iterative control loop using a vision sensor to control the manipulated points and bring the positioned points to the desired position.

5 Object under study

This master thesis will undertake the study of a particular type of object to understand how it behaves, and what can be achieved for manipulating it with a dual arm robot. Equilibrium configurations, which we will define later, of the object under different conditions will be studied and the same will be used to generate trajectories of the manipulation points on the object to conform it to a particular configuration.

The object under study will be an articulated object which has n links of equal length attached together using revolute joints.

As can be seen from Fig. 8 ln is the length of each link and lc is the distance of the Center of Mass (CoM) of each link from the end-point of the previous link. q_1 to q_n are the angle of orientation of each link of the object till the n^{th} link. The relative angle between two links of the object is denoted by tr. $tr1$ denotes the relative angle between the first and the second link, $tr2$ denotes the relative angle between the second and the third link and so on. The angle of gravity with respect to the object is denoted by psi.

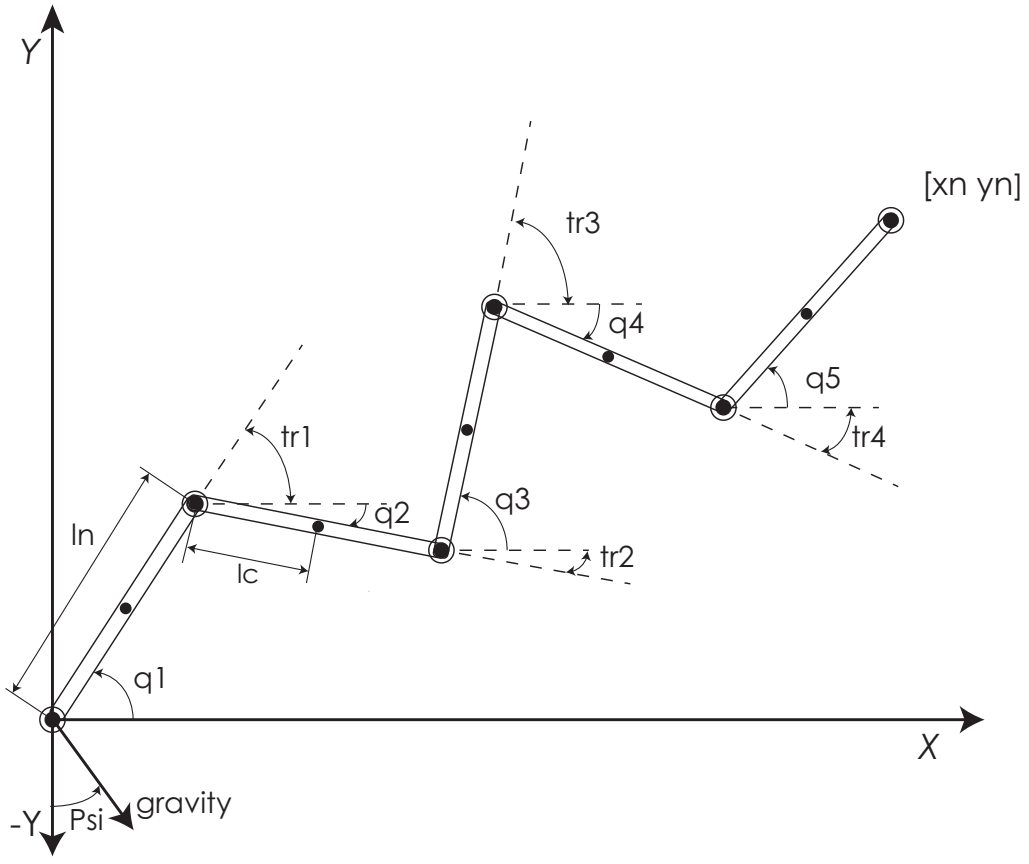


Figure 6: Object Under Study: Articulated Object

6 Modeling of the Articulated Object

Contents

6.1 Modeling

Regarding the n-link planar object shown in Fig. 8, θ_k $(k = 1, 2, \ldots, k)$ is defined to be the angle measured counterclockwise from the positive x-axis. We also define q_k to be the relative angle between links k and $k-1$, and link 0 is defined to be the positive x -axis. For the k^{th} link, let m_k, l_k and l_{ck} be its mass, its length, its distance from joint k to the center of mass (COM) of the link k respectively.
To express the kinematic and potential energy of the robot, for simplicity of derivation, we use the following notation [12] related to the lengths of links:

$$l_{ki} = \begin{cases} l_i, & i < k, \\ l_{ck}, & i = k, \\ 0, & i > k \end{cases} \tag{6.1}$$

We express (x_k, y_k), the coordinate of COM of link k, as

$$x_k = \sum_{i=1}^{n} l_{ki} cos\theta_i, \qquad y_k = \sum_{i=1}^{n} l_{ki} sin\theta_i. \tag{6.2}$$

Now we know the basic equation for computing the potential energy of a system is the product of its mass, acceleration due to gravity and and height of the COM in the direction of gravity. So for our system we use the same expression and substitute the height using the above found expressions of the COM. If the gravity angle is zero, the height would be the same as the y coordinate of the COM and we obtain the potential energy as:

$$P = \sum_{k=1}^{n} m_k g y_k = \sum_{i=1}^{n} \left(\sum_{k=1}^{n} m_k l_{ki} g \right) cos\theta_i \tag{6.3}$$

Now in a situation where the manipulation of the system leads to changes in position and orientation of both ends of the object, the gravity vector will not be in the same direction as that of the y-axis of the previously described reference frame of the object. This change in the orientation of the object with respect to gravity

is incorporated into the system using rotation matrix for a rotation of ψ (the gravity angle) about the z axis of the reference frame. The rotation matrix takes the following form:

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.4}$$

From the above expression, we calculate the new y coordinate as:

$$y_{new} = x \cdot sin\psi + y \cdot cos\psi \tag{6.5}$$

Now this expression for the new y, which is the height of the system according to the new gravity angle is used to compute the expression for the potential energy of the system using the equation as before. The more the number of links the potential energy is updated accordingly.

6.2 Lagrange Formulation

The Lagrange formulation describes the behavior of a system in terms of work and energy stored in the system. The Lagrange equations are expressed in the form:

$$\tau = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}}\right)^T - \left(\frac{\partial L}{\partial q}\right)^T \tag{6.6}$$

where

- τ is the vector of generalized forces applied on the system, which are equal to the input joint torques or forces,
- q is the vector of generalized coordinates, which in our case will be the link angles
- $\dot{q}$ is the vector of generalized velocities, in our case this will be zero as the system is under static conditions.
- L is called the Lagrangian

  $$L = K - P \tag{6.7}$$

  K is the total Kinetic energy and P the total Potential energy (due to gravity, deformations, etc.). Now in our study the system is under static conditions and the formulation becomes

  $$\tau = \left(\frac{\partial P}{\partial q}\right)^T \tag{6.8}$$

6.3 Equilibrium Using Lagrange Multipliers

For closed-loop robots, such as parallel robots, to which our system is analogous to, the expression of kinetic and potential energies can be obtained using all the joint variables including the active (q_a) and the passive joint coordinates (q_d). The variables q_a and q_d are not independent and are linked to each other using the geometric equations described in Section 7.

$$G(q_a, q_d) = 0 \tag{6.9}$$

and

$$A(q_a, q_d)\dot{q}_d + B(q_a, q_d)\dot{q}_a = 0 \tag{6.10}$$

where $\mathbf{A} = \left[\frac{\partial G}{\partial q_d}\right]$ and $\mathbf{B} = \left[\frac{\partial G}{\partial q_a}\right]$ are two matrices depending on q_a and q_d.

Using these, the Lagrange equations can be rewritten using the Lagrange multipliers λ as

$$\tau + \mathbf{B}^T\lambda = \tau_a, \text{where}, \tau_a = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_a}\right)^T - \left(\frac{\partial L}{\partial q_a}\right)^T \tag{6.11}$$

$$\mathbf{A}^T\lambda = \tau_d, \text{where}, \tau_d = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_d}\right)^T - \left(\frac{\partial L}{\partial q_d}\right)^T \tag{6.12}$$

These set of equations derived using the Lagrange formulation and the geometry equations can be solved for all sets of $[xn\ yn]$ for obtaining the equilibrium angles and hence the configurations.

7 Study of 3-link object

Contents

7.1 Geometric Model for 4 link object

To begin with we study an object with 3 links. Studying the geometry of the object, we obtain two equations linking the link angles θ_1, θ_2 and θ_3 and the end position of the final link given by xn and yn.

$$xn - ln \cdot cos\theta_1 - ln \cdot cos\theta_2 - ln \cdot cos\theta_3 = 0 \tag{7.1}$$

$$yn - ln \cdot sin\theta_1 - ln \cdot sin\theta_2 - ln \cdot sin\theta_3 = 0 \tag{7.2}$$

This is similar to the Direct Geometric model for a 3R(Revolute) planar robot.

7.2 Equilibrium Configurations

From the model of the 3 link object, we obtain θ_2 and θ_3 in terms of θ_1, xn and yn.

$$\theta_2 = f(\theta_1, xn, yn) \tag{7.3}$$
$$\theta_3 = f(\theta_1, xn, yn) \tag{7.4}$$

This is done using the *solve* function in matlab. These expressions for θ_2 and θ_3 are substituted in the potential energy equation to obtain an equation of potential energy purely in terms of θ_1, xn and yn.
So now we have:

$$P = f(\theta_1, xn, yn) \tag{7.5}$$

$$P = \sum_{k=1}^{3} m_k g y_k = \sum_{i=1}^{3} \left(\sum_{k=1}^{3} m_k l_{ki} g \right) cos\theta_i \tag{7.6}$$

This modeling for the potential of the object has been validated using simulink and the error plot is as follows:

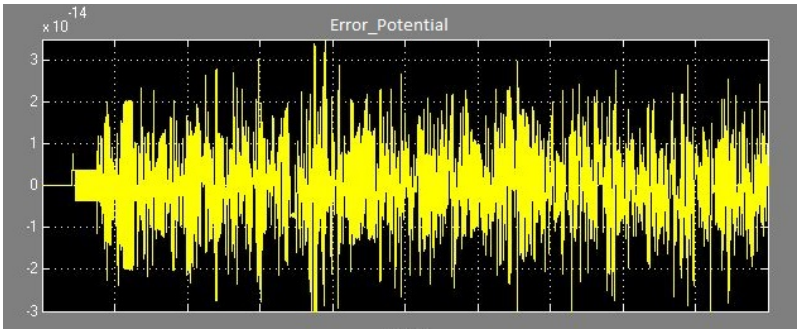


Figure 7: Error plot for the potential function

The error is of the order of 10^{-14} hence validating the potential energy function.

Now, to compute the angles at equilibrium we have to use an equilibrium condition which has to be satisfied. At Static Equilibrium, partial differential of the potential energy with respect to the link angles should be zero.

$$Eqn1 = \frac{\partial P}{\partial \theta_1} = 0 \tag{7.7}$$

Now we have an equation that is solved for given value of xn and yn to obtain the angle of the first link, θ_1 at equilibrium at this particular position for the end point of link 3. This is done using *vpasolve* function of MATLAB which is a numerical solver and gives out the values of θ_1 which satisfies the equation in a particular interval. In our case this interval is $[0\ 2\pi]$.

Using this value of θ_1 and substituting in the expressions for θ_2 and θ_3 we obtain all the equilibrium angles for this pair of xn and yn.

7.3 Equilibrium of 3-link object using Lagrange Multipliers

Using the Lagrange multipliers explained in Section 6.3, we obtain the equations of equilibrium for the 3-link object as:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial G}{\partial \theta_{2,3}} \end{bmatrix} \tag{7.8}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial G}{\partial \theta_1} \end{bmatrix} \tag{7.9}$$

$$\tau + \mathbf{B}^T\lambda = \tau_a, \text{where}, \tau_a = \left(\frac{\partial P}{\partial \theta_1}\right)^T \tag{7.10}$$

$$\mathbf{A}^T\lambda = \tau_d, \text{where}, \tau_d = \begin{bmatrix} \frac{\partial P}{\partial \theta_2} \\ \frac{\partial P}{\partial \theta_3} \end{bmatrix} \tag{7.11}$$

During equilibrium conditions, τ will be zero. So from the above equations, λ is solved by taking inverse of $\mathbf{B}^T\tau_a$. Multiplying the whole equation by determinant of B^T will help avoid problems with finding the solution. After finding λ, it is substituted in 7.11 to get the equations which will be solved for finding the equilibrium angles. The equations along with 7.2 will be used for finding the solution of the four equilibrium angles for different values of $[xn\ yn]$

8 Study of 4-link object

Contents

8.1 Geometric Model for 4 link object

Now we take up the study of an object with 3 links. Studying the geometry of the object, just like the 3-link object, we obtain two equations linking the link angles θ_1, θ_2, θ_3 and θ_4 and the end position of the final link given by xn and yn.

$$xn - ln \cdot cos\theta_1 - ln \cdot cos\theta_2 - ln \cdot cos\theta_3 - ln \cdot cos\theta_4 = 0 \tag{8.1}$$
$$yn - ln \cdot sin\theta_1 - ln \cdot sin\theta_2 - ln \cdot sin\theta_3 - ln \cdot sin\theta_4 = 0 \tag{8.2}$$

This is similar to the Direct Geometric model for a 4R(Revolute) planar robot.

8.2 Equilibrium Configurations

From the model of the 3 link object, we obtain θ_3 and θ_4 in terms of θ_1, θ_2, xn and yn.

$$\theta_3 = f(\theta_1, \theta_2, xn, yn) \tag{8.3}$$
$$\theta_4 = f(\theta_1, \theta_2, xn, yn) \tag{8.4}$$

This is done using the *solve* function in matlab. These expressions for θ_3 and θ_4 are substituted in the potential energy equation to obtain an equation of potential energy purely in terms of θ_1, θ_2, xn and yn.
So now we have:

$$P = f(\theta_1, \theta_2, xn, yn) \tag{8.5}$$

For a four link object the expression for the potential energy is given by Equation 8.6

$$P = \sum_{k=1}^{4} m_k g y_k = \sum_{i=1}^{4} \left(\sum_{k=1}^{4} m_k l_{ki} g \right) cos\theta_i \tag{8.6}$$

This modeling for the potential of the object has been validated using simulink and the error plot is as follows:

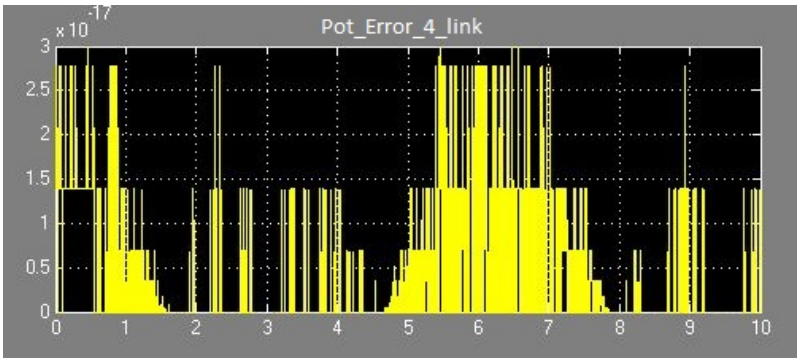


Figure 8: Error plot for the potential function for object with 4 links

The error is of the order of 10^{-17} hence validating the potential energy function.

Now, to compute the angles at equilibrium we have to use an equilibrium condition which has to be satisfied. At Static Equilibrium, partial differential of the potential energy with respect to the link angles should be zero.

$$Eqn1 = \frac{\partial P}{\partial \theta_1} = 0 \tag{8.7}$$

$$Eqn2 = \frac{\partial P}{\partial \theta_2} = 0 \tag{8.8}$$

Now we have an equation that is solved for given value of xn and yn to obtain the angle of the first link, θ_1 and the second link θ_2 at equilibrium at this particular position for the end point of link 4. This is done using *vpasolve* function of MATLAB which is a numerical solver and gives out the values of θ_1 and θ_2 which satisfies the equation in a particular interval. In our case this interval is $[0\ 2\pi]$.

Using this value of θ_1 and θ_2 and substituting in the expressions for θ_3 and θ_4 we obtain all the equilibrium angles for this pair of xn and yn.

8.3 Equilibrium of 4-link object using Lagrange Multipliers

Using the Lagrange multipliers explained in Section 6.3, we obtain the equations of equilibrium for the 4-link object as:

$$\mathbf{A} = \left[\frac{\partial G}{\partial \theta_{3,4}}\right] \tag{8.9}$$

$$\mathbf{B} = \left[\frac{\partial G}{\partial \theta_{1,2}}\right] \tag{8.10}$$

$$\tau + \mathbf{B}^T\lambda = \tau_a, \text{where}, \tau_a = \begin{bmatrix} \frac{\partial P}{\partial \theta_1} \\ \frac{\partial P}{\partial \theta_2} \end{bmatrix} \tag{8.11}$$

$$\mathbf{A}^T\lambda = \tau_d, \text{where}, \tau_d = \begin{bmatrix} \frac{\partial P}{\partial \theta_3} \\ \frac{\partial P}{\partial \theta_4} \end{bmatrix} \tag{8.12}$$

During equilibrium conditions, τ will be zero. So from the above equations, λ is solved by taking inverse of $\mathbf{B}^T\tau_a$. Multiplying the whole equation by determinant of B^T will help avoid problems with finding the solution. After finding λ, it is substituted in 8.12 to get the equations which will be solved for finding the equilibrium angles. The equations along with 8.2 will be used for finding the solution of the four equilibrium angles for different values of $[xn\ yn]$

9 Obtaining the Configuration Space of the object

Now that the modeling of the object is done and a methodology to find equilibrium configuration has been elaborated, this model is used to generate the link angles at equilibrium for all achievable positions for the end-point of the object in Cartesian space. For a link length of l, the reachable points will be inside a circle of radius $3l$. So for each point inside this circle, the corresponding equilibrium link angles are found and are stored in an array.

Now to represent each configuration we use the relative angles between two successive links so that the configuration space can be depicted in a two-dimensional image. Hence,

$$tr1 = \theta_1 - \theta_2 \tag{9.1}$$
$$tr2 = \theta_2 - \theta_3 \tag{9.2}$$

where $tr1$ and $tr2$ are the two relative angles for the 3R object.

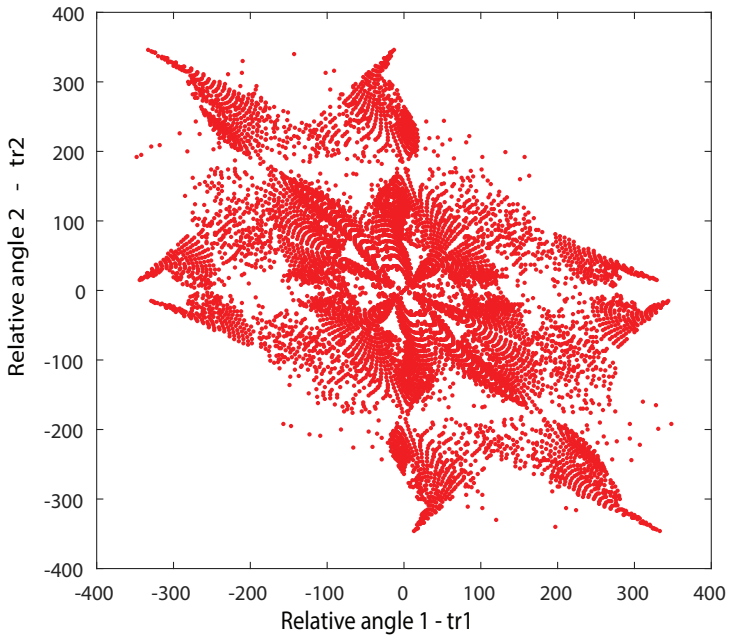


Figure 9: Configuration space of the object

The configuration space for a four-link object will be a 3 dimensional space as in Fig ?? and 11. And the configuration space of a five-link object will be a

4-dimensional space and so on. As the degrees of freedom increases, it becomes difficult to show graphically the configuration space.

Now to represent each configuration we use the relative angles between successive links so that the configuration space can be depicted in a three-dimensional image. Hence,

$$tr1 = \theta_1 - \theta_2 \tag{9.3}$$
$$tr2 = \theta_2 - \theta_3 \tag{9.4}$$
$$tr3 = \theta_3 - \theta_4 \tag{9.5}$$
$$\tag{9.6}$$

where $tr1$, $tr2$ and $tr3$ are the three relative angles for the 4R object.

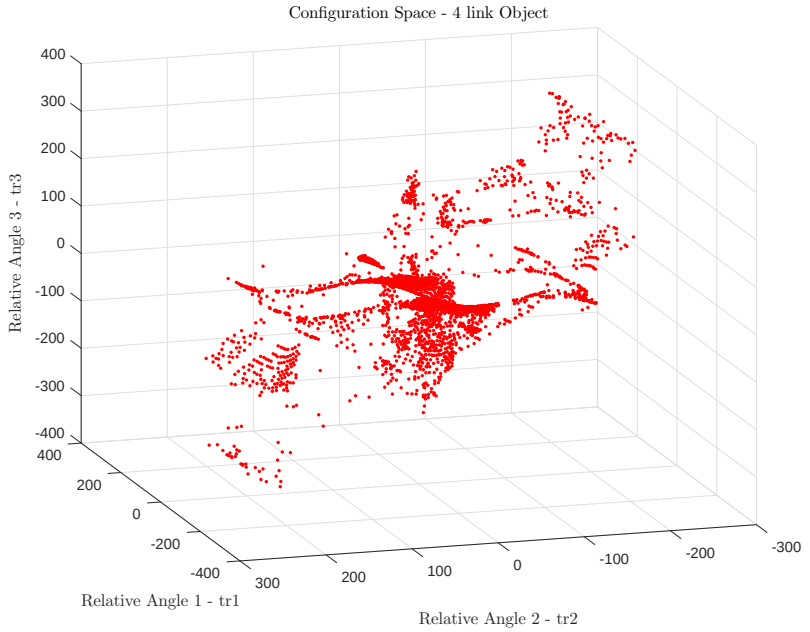


Figure 10: Configuration space of a 4 link object

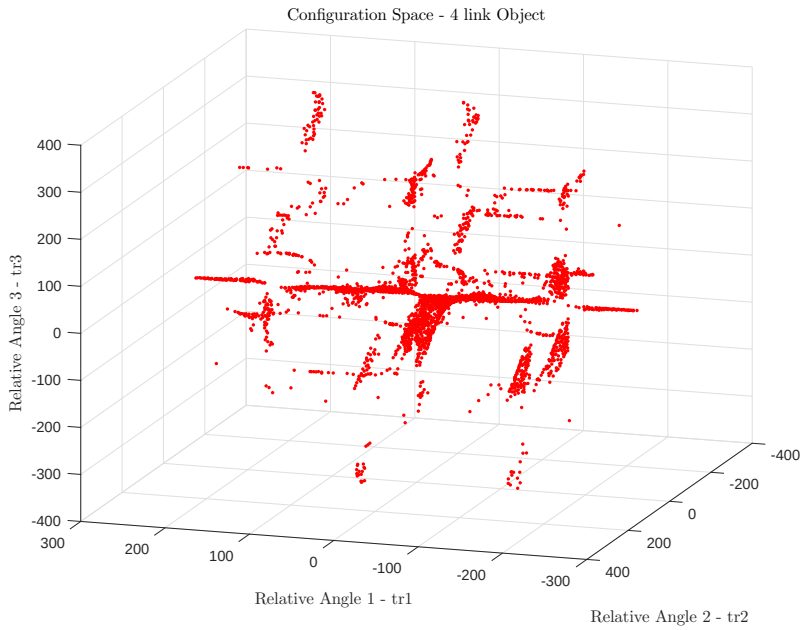


Figure 11: Configuration space of a 4 link object

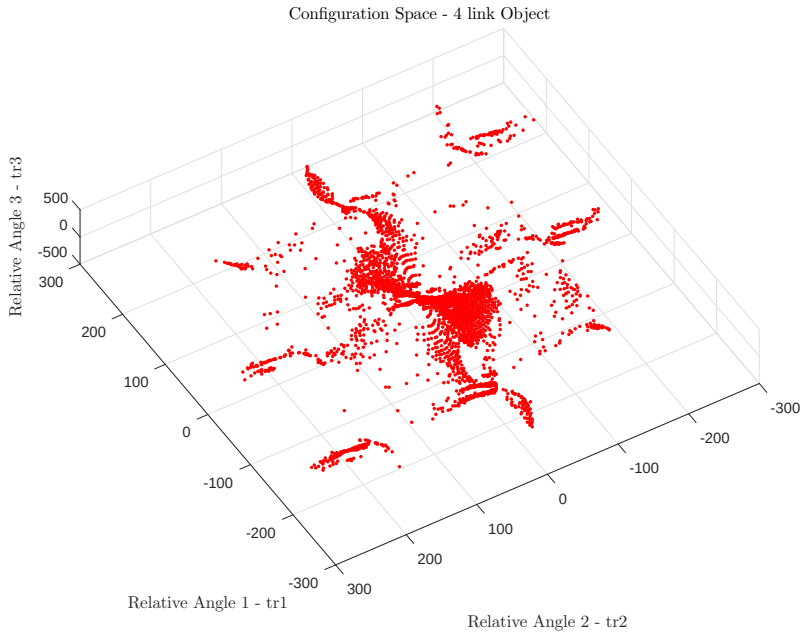


Figure 12: Configuration space of a 4 link object

10 Configuration Space: Represented as a map

To find a trajectory to move from one configurations of the links to another, we represent the configuration space of the object in the form of a grid map with squares representing each pair of relative angles. The squares marked black have values zero and corresponds to the configurations which are attainable by the object in equilibrium and the squares marked white have a value of 1 and corresponds to the configurations which are not attainable by the object in equilibrium. A part of the big grid map is shown below:

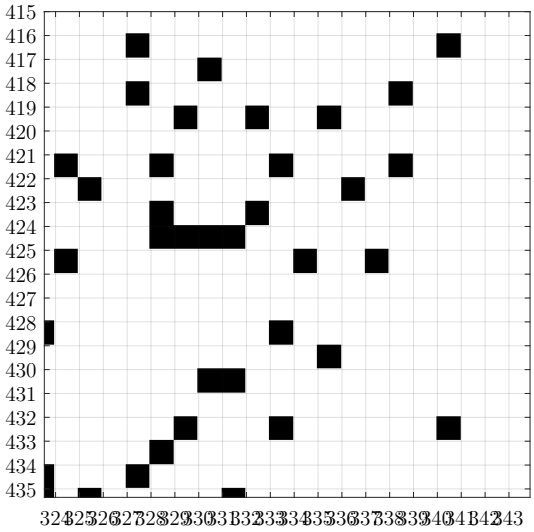


Figure 13: Configuration space represented as a grid map

For a four-link object the same will be a 3D space with black and white cubes. This method is just to help visualize how the algorithm works.

11 Trajectory Generation

Contents

11.1 Objective

The objective of this section is to define a methodology for finding a trajectory to be followed to move from one equilibrium configuration of the object to another one, enabling a dual arm robot to handle the object just like a human could. Now that we have defined a grid map for the configurations, the same could be treated as a path finding problem. The goal will be to move from one square to another in the grid map. Both corresponding to the initial and final configuration. But because all squares are not connected to each other, a small change in the concept of moving between squares will be taken up in the following sections.

11.2 Shortest path Algorithms

Shortest path problems are the ones where one finds a path between two nodes or vertices in a graph which minimizes the sum of the weight of the path followed. The most important algorithms for solving this problem are:

- **Dijkstra's algorithm** → solves the single-source shortest path problem.
- **BellmanFord algorithm** → solves the single-source problem if edge weights may be negative.
- **A* search algorithm** → solves for single pair shortest path using heuristics to try to speed up the search.
- **FloydWarshall algorithm** → solves all pairs shortest paths.
- **Johnson's algorithm** → solves all pairs shortest paths, and may be faster than FloydWarshall on sparse graphs.
- **Viterbi algorithm** → solves the shortest stochastic path problem with an additional probabilistic weight on each node.

11.3 A* (A-Star) Algorithm

A-Star algorithm is a shortest path finding algorithm that finds a locally optimal path when it is possible. Each state is estimated with the following formula that encourages the algorithm to select the state with the closest distance to the goal:

$$F = G + H \tag{11.1}$$

where:

- F is the estimated cost of the state
- G is the exact distance from the starting state to the current node
- H is the estimated cost from a given node to the goal

H is the heuristic function that is used to approximate distance from the current location to the goal state. This function is distinct because it is a mere estimation rather than an exact value. The more accurate the heuristic the better the faster the goal state is reach and with much more accuracy.

Algorithm 1 Pseudocode for A* algorithm

```
Put the initial state in the open list
Sb ⇐ initial state
repeat
  Put the state Sb in the close list (or mark as explored)
  for each reachable state Sr from Sb do
    if the reachable state is not yet in the list then
      Estimate the cost (F=G+H);
      Add the reachable state Sr in the open list
      Mark Sb as the previous state of Sr
    else if the new estimation (F=G+H) is better than the previous then
      update data in the list
    end if
  end for
  Find the best state Sb in the open list
until Sb is the final state
```

From now on, obstacles will be the squares that represent the unattainable configurations of the object. In our case, instead of moving single square by square

and checking for obstacles, which would lead to no result, as all the squares are not connected with each other, we do the search by considering motions from a square region to another in the grid.
So if there are only obstacles in a given region it is a state that we cannot go to. And if in a region there is atleast one square which is not an obstacle, that region corresponds to a state that we can move to.

In every iteration, the algorithm checks the four square regions that it can move to from the current position and checks if the region is an obstacle and if has already been checked and is in the list. If the region has attainable configurations in it, it chooses the closest to the goal and adds it to the list and marks the current state as the previous state of the reachable state. End the end of the iteration, the best state is found from the list and is chosen as the current state.

The structure of the list is as follows:

X	Y	G	H	F	Previous	Explored(CloseList)

X and Y are the coordinates of the current state, G, H and F are the A* paramenters as explained before. Previous is the position of the previous state in the list as a whole. Explored is either 1 or 0 based on if the state has been explored or not.

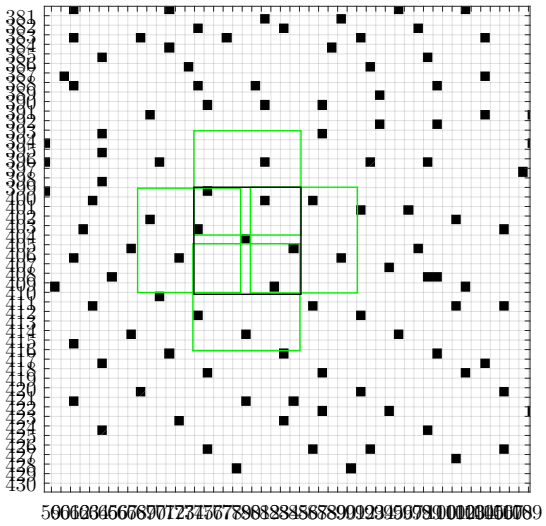


Figure 14: Increments of square areas

Finally when the current state is equal to the goal, the trajectory is found by tracking the path back to the start by using the previous states. The trajectory generated is shown below:

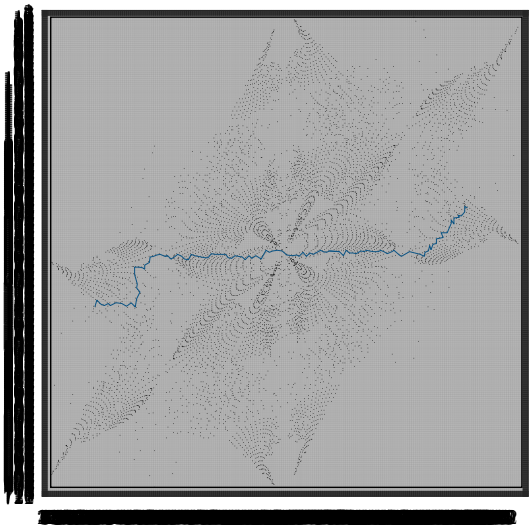


Figure 15: Representation of the trajectory

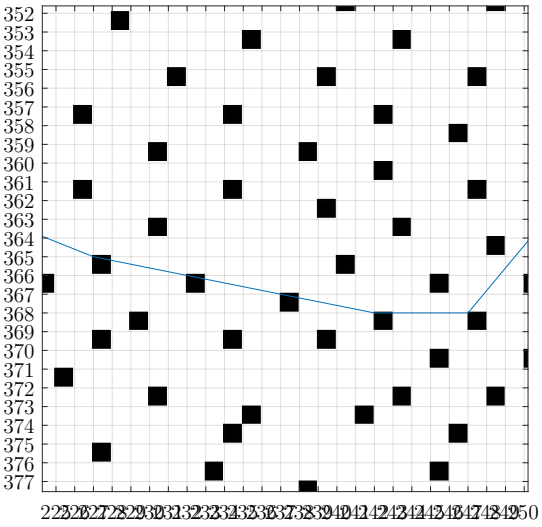


Figure 16: Representation of the trajectory (Zoomed)

11.4 A-Star applied to higher dimension

The List in the case of a four link object will be as follows:

X	Y	Z	G	H	F	Previous	Explored(CloseList)

The search will be done in three dimensions. So analogous to the algorithm for the three-link object, we will move in regions but this time not a square, it will be a cube as it is of three dimensions.
To test the algorithm we adopt two points like in Figure 17 as Start and End configurations.

$$Start = [-60 - 20 - 8]$$
$$Goal = [9\ 17\ 34]$$

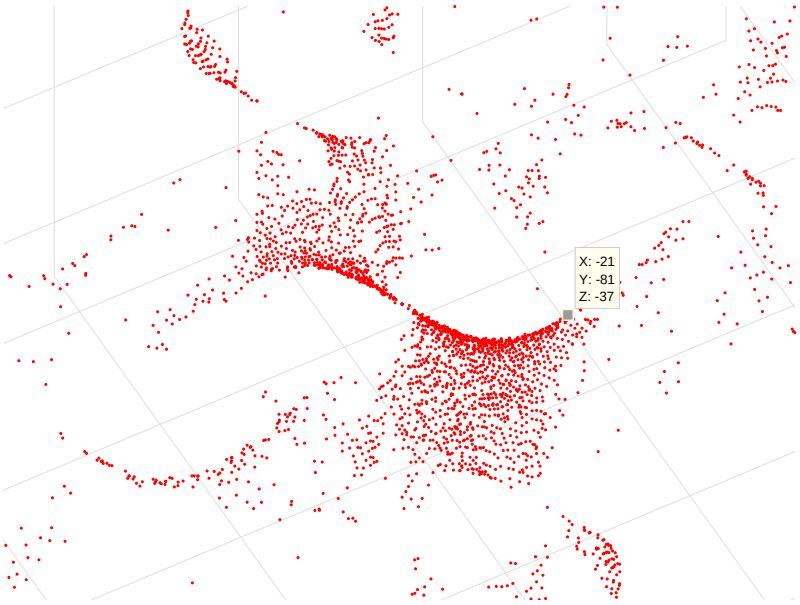


Figure 17: Choosing Start and End Configurations from the Configuration Space

The algorithm outputs the trajectory as an array of the configurations through which it has to pass through to reach from the starting configuration to the end configuration.

tr1	tr2	tr3
9	17	34
3	7	19
-6	7	10
-6	-2	10
-15	-2	1
-15	-11	1
-24	-11	1
-24	-20	1
-33	-20	1
-42	-20	1
-51	-20	1
-60	-20	-8

Table 1: Trajectory in terms of relative angles $tr1$, $tr2$ and $tr3$

The trajectory generated in the configuration space is shown in Figure 18 and 19. This configuration space is for a four-link object under zero gravity angle.

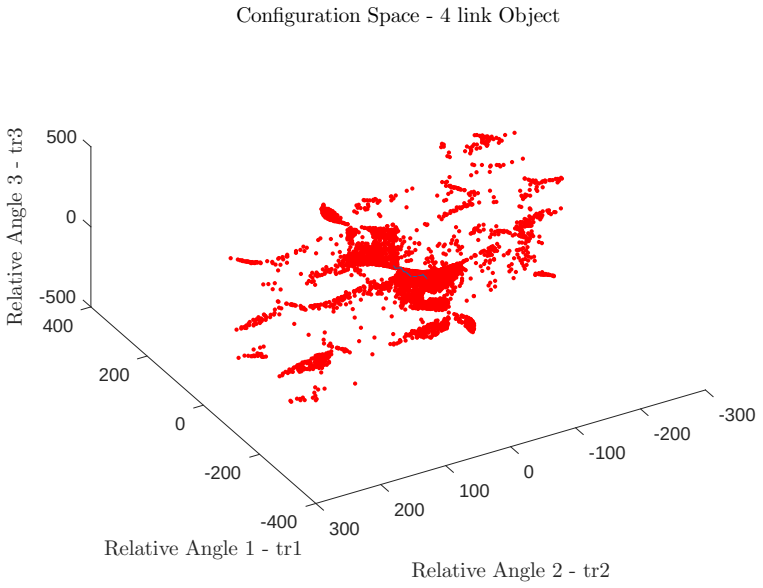


Figure 18: Trajectory generated for 4-link object shown in blue

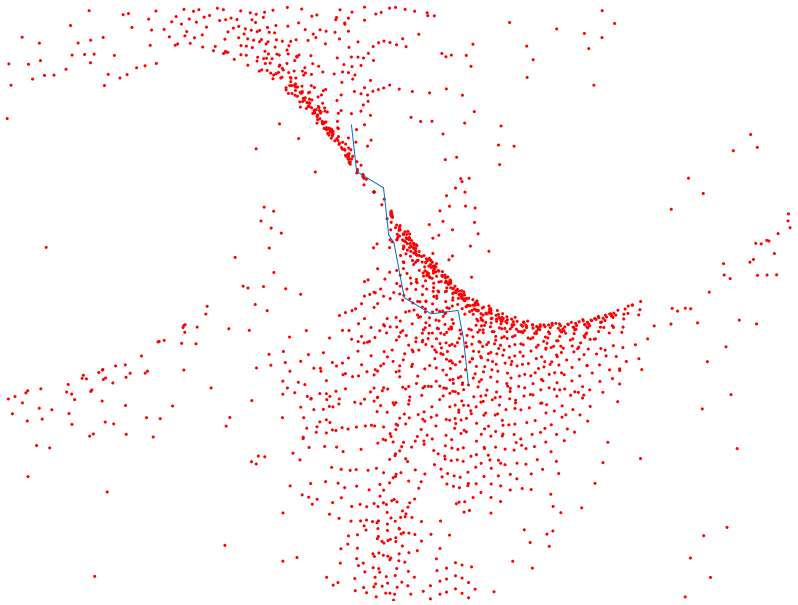


Figure 19: Trajectory generated for 4-link object shown in blue

11.5 Probabilistic Road-map Method (PRM)

For configuration spaces of higher degrees of freedom, it is possible to use another method called as Probabilistic Roadmap Method. Probabilistic Roadmap planners solve motion planning problems where the robot's configuration space, C is of high degrees and the environment is defined by thousands of such configurations, defining the free space F. PRM planner builds only an extremely simplified representation of F, called a probabilistic roadmap. A roadmap is a graph whose nodes are configurations sampled from F according to a suitable probability measure and whose edges are simple collision-free paths, e.g. straight-line segments, between sampled configurations. [9].

In most cases, PRM is used for motion planning algorithms for finding collision free trajectory for a robot. The configuration could be the position of various points on its body and the free space will be the real world with obstacles. In our case, the configuration space is formed the relative angles of the articulated object. So if an object has N links, then the configuration space will be of the dimension $N-1$. And we have to find straight line connected paths between two configurations without crossing the obstacles(which in our case are the unreachable configurations) that are there in the configuration space.

PRM planners use two probes based on such algorithms to access geometric information on the configuration space C:

- For any $q \in C$, $FreeConf(q)$ is true if and only if qF, where F is the Free

Space and C is the configuration space.

- For any pair $q, q' \in C$, $FreePath(q, q')$ is true if and only if q and q' can be connected with a straight-line path lying entirely in F.

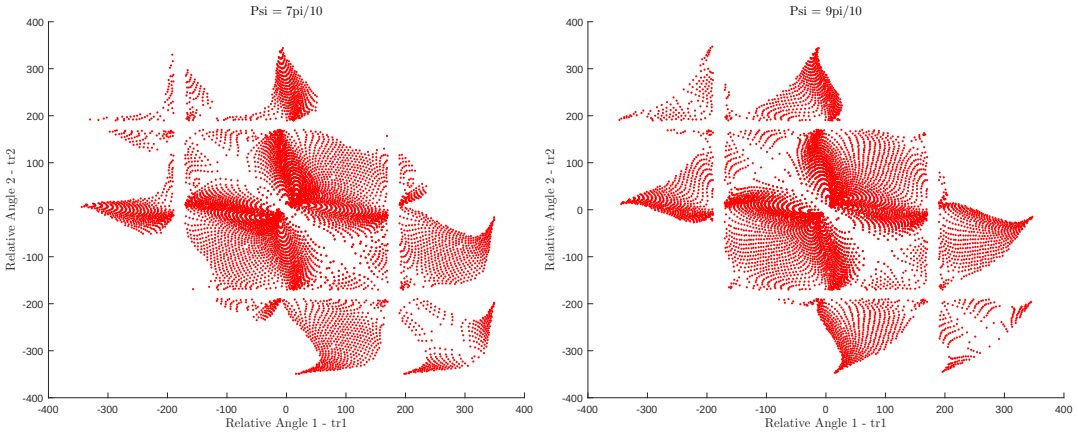
The pseudo-code for the algorithm is given below.

Algorithm 2 Pseudocode for a basic PRM algorithm

if $Freepath(q_1,q_2)$ is true **then**
  Return path between q_1 and q_2
end if
Initialize the roadmap R with two nodes, q_1 and q_2.
repeat
  Sample a configuration q from C uniformly at random.
  if $FreeConf(q)$ is true **then**
    add q as a new node of R.
  end if
  for every node v of R such that $v \neq q$ **do**
    if FreePath(q, v) is true **then**
      add (q, v) as a new edge of R.
    end if
  end for
until q_1 and q_2 are in the same connected component of R or R contains N + 2 nodes.
if q_1 and q_2 are in the same connected component of R **then**
  return a path between q1 and q2.
else
  return NoPath.
end if

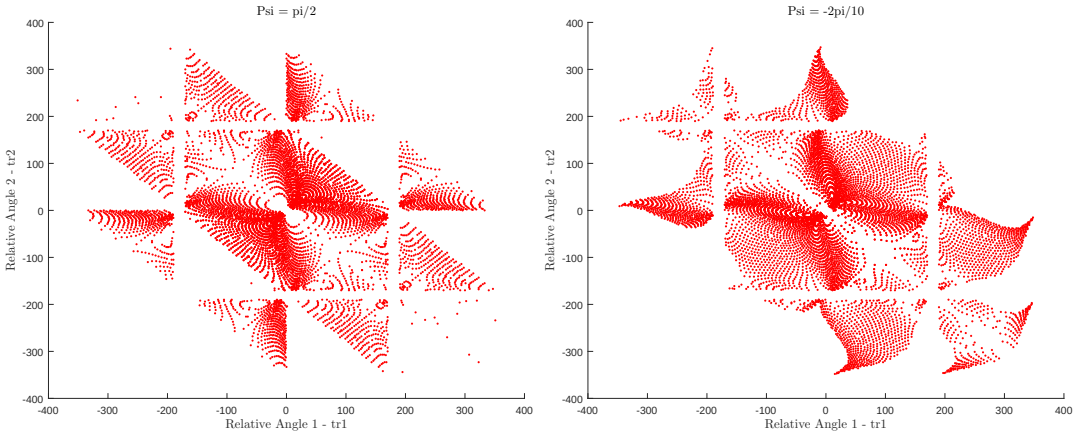
12 Effect of gravity on Object Configuration

To study the effect of gravity on the object configuration, configuration space corresponding to different gravity angles *Psi* are studied. It can be seen that the configuration space is not the same and that particular configurations can be attained in certain gravity angles while it cannot be attained in certain others.



(a) Configuration Space for a gravity angle of $7\pi/10$

(b) Configuration Space for a gravity angle of $9\pi/10$



(c) Configuration Space for a gravity angle of $\pi/2$

(d) Configuration Space for a gravity angle of $-2\pi/10$

Figure 20: Configuration Space for different gravity angles Psi

Below is shown the configuration space for two different gravity angles, Psi = 4pi/10 and Psi = 6pi/10.

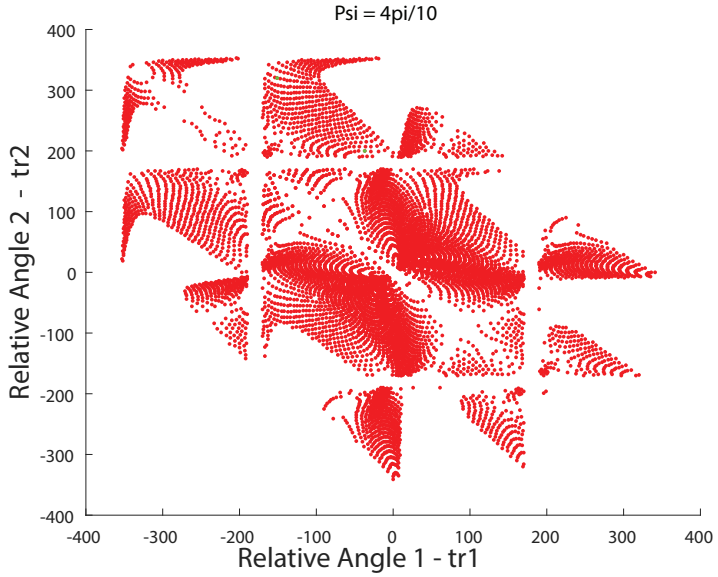


Figure 21: Configuration Space for a gravity angle of $4\pi/10$

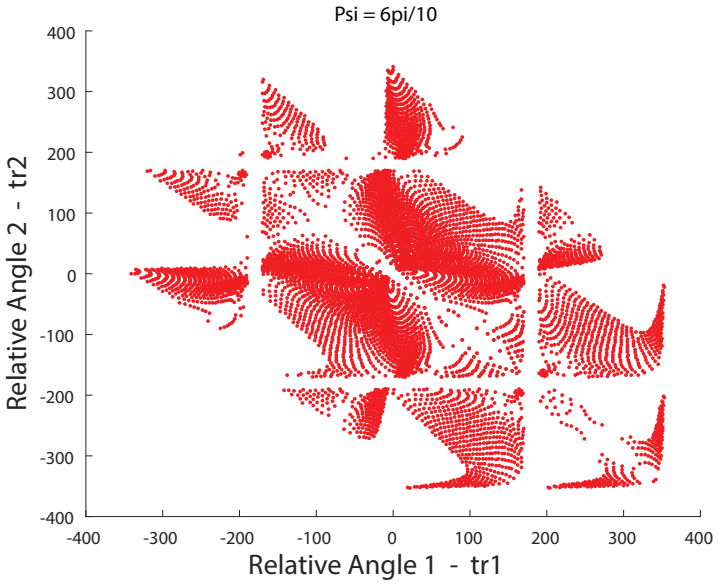


Figure 22: Configuration Space for a gravity angle of $6\pi/10$

Now to understand the effect we zoom in to the same part of the configuration spaces for both gravity angles. It can be seen that, to move from the configuration [-150,320] to [-40,200] is not possible in the case when the gravity angle Psi = 6pi/10 but the same is possible when the gravity angle is 4pi/10. The configuration is given

my [tr1,tr2] where tr1 and tr2 are the relative angles between the successive links of the object.

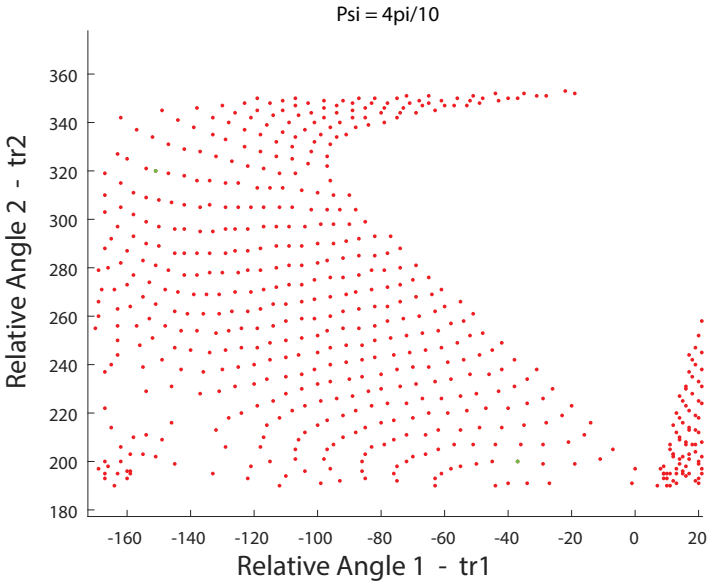


Figure 23: Configuration Space for a gravity angle of $4\pi/10$ (Zoomed)

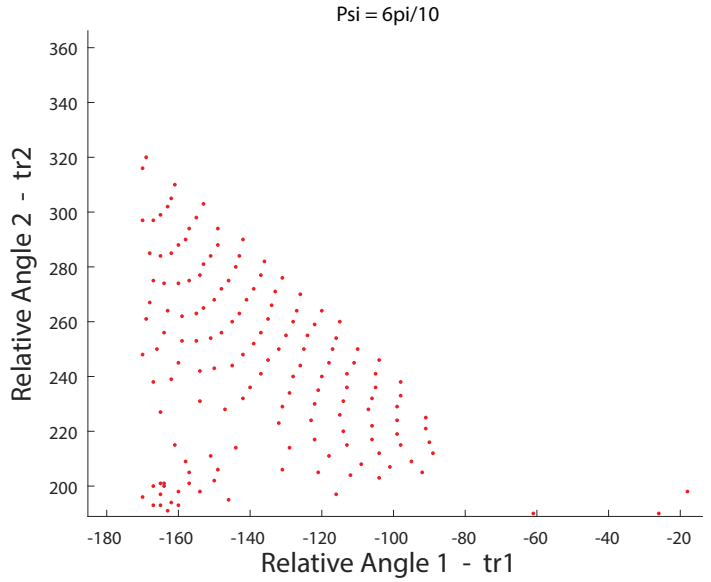


Figure 24: Configuration Space for a gravity angle of $6\pi/10$ (Zoomed)

13 Object manipulation using two 2R-planar robots

Contents

13.1 Objective

Now that the modeling, trajectory generation and study of the equilibrium conditions under different gravity angles have been completed, we try to implement this manipulation task using two 2R-robots. This will be done by manipulating the two end-points of the object using the two robot end-effectors. This is co-manipulation and even though there are four-DoFs for the two control points of the object, ultimately because of two robots manipulating the same object, it becomes 2DoF, i.e; the relative positioning of the two end-effectors.

13.2 Modeling of 2R planar robots

Direct Geometric Model

The direct geometric model for a 2R planar robot is given as follows. L_1 and L_2 are the length of the proximal and the distal links of the robot. q_1 and q_2 are the joint angles and xn and yn are the end-effector coordinates.

$$xn = L_1 \cdot cos(q_1) + L_2 \cdot cos(q_2) \tag{13.1}$$
$$yn = L_1 \cdot sin(q_1) + L_2 \cdot sin(q_2) \tag{13.2}$$
$$\tag{13.3}$$

Inverse Kinematic Model

The task space is given by the vector:

$$X = [xn, yn]^T \tag{13.4}$$
$$\tag{13.5}$$

The joint space is given by the vector

$$q = [q_1, q_2]^T \tag{13.6}$$

The relationship between the joint velocities and the task space velocities is given by

$$\dot{\mathbf{X}} = \mathbf{J} \cdot \dot{\mathbf{q}} \tag{13.7}$$

The jacobian matrix links the joint velocities and the task space.

$$J = \begin{bmatrix} -L_1 \cdot sin(q_1) & -L_1 \cdot cos(q_1) \\ L_1 \cdot cos(q_1) & L_2 \cdot cos(q_2) \end{bmatrix} \tag{13.8}$$

13.3 Redundant Manipulation

We have two 2R-robots manipulating two end-points of the articulated object. So we are controlling two degrees of freedom. i.e; the relative position between the two end-points.

We define the manipulation variables as r and t, the horizontal and vertical distance between the two manipulation points.

$$r = L_1 \cdot cos(q_{12}) + L_2 \cdot cos(q_{22}) - L_1 \cdot cos(q_{11}) - L_2 \cdot cos(q_{21}) \tag{13.9}$$

$$t = L_1 \cdot sin(q_{12}) + L_2 \cdot sin(q_{22}) - L_1 \cdot sin(q_{11}) - L_2 \cdot sin(q_{21}) \tag{13.10}$$

Now that we are controlling two degrees of freedom with a four degrees of freedom of the robots, (two degrees of freedom each) we have two degrees of freedom that can be used to achieve more tasks.

14 Task Priority Approach for Redundant Manipulators

Contents

14.1 Introduction to Task-priority Concept

Robot manipulators are usually designed to have as many number of freedoms as is required by the desired task. But in lots of cases, extra joints are given to make it more dexterous and to add more functionality to the manipulator. They are required to be more flexible and adaptive like a human arm. When the robot end-effector has to follow a specific trajectory, avoiding obstacles, having more degrees of freedom lets it achieve this goal. In [18] a task priority based approach was presented to utilize the extra DoFs of the robot.

This approach deals with the problem of degeneracy, or the situation of not having enough degrees of freedom to complete all tasks at the same time, by assigning priorities to the task. So that the task with higher priority is ensured to be executed while the lower priority task is executed as well as possible.

14.2 Inverse Kinematic Solutions with Order of Priority

In this section, the task priority approach [18] is elaborated. We consider the case with two tasks to be executed by the robot. Various papers exists which detail the

approach with more than two tasks. In our study, as we work with two tasks, we limit the explanation to the task-priority approach for the same.

Considering two sub-tasks defined by the corresponding manipulation variables $T_1 \in R^{m_1}$ and $T_2 \in R^{m_1}$ the relationships between the manipulation variable and the joint-variable $q \in R^n$ are expressed as follows:

$$T_i = f_i(q) \qquad (i = 1, 2) \tag{14.1}$$
$$\tag{14.2}$$

The differential relationships are given by

$$\dot{T}_i = J_i(q)\dot{q} \qquad (i = 1, 2) \tag{14.3}$$
$$\tag{14.4}$$

where $J_i(q) = \partial f_i/\partial q \in R^{m_i \times n}$ is the Jacobian matrix for the i^{th} manipulation variable.

The Inverse Kinematic modeling for redundant robots with two tasks is given by:

$$\dot{q} = J_1^+ \dot{T}_1 + (I_n - J_1^+ J_1) J_2^+ \dot{T}_2 \tag{14.5}$$

where $\dot{T}_1$ and $\dot{T}_2$ are the manipulation variable velocity for the two tasks, I_n an identity matrix of size n, in our case n = 4. J_1^+ and J_2^+ are the pseudo-inverse of the non-square Jacobian matrices J_1 and J_2. A pseudo-inverse matrix of a matrix $A^{m \times n}$ is defined as a matrix $A^{+n \times m}$ satisfying all of the following four criteria:

- $AA^+A = A$
- $A^+AA^+ = A^+$
- $(AA^+)^* = AA^+$
- $(A^+A)^* = A^+A$

where A^* is the Hermitian transpose and A^T is the transpose of matrix A.

14.3 Dexterous Manipulation using two 2R planar robots

We have two 2R planar robots with link lengths $L_1 = 300mm$ and $L2 = 300mm$. And placed 700mm apart along the x-axis as shown in the figure 25.

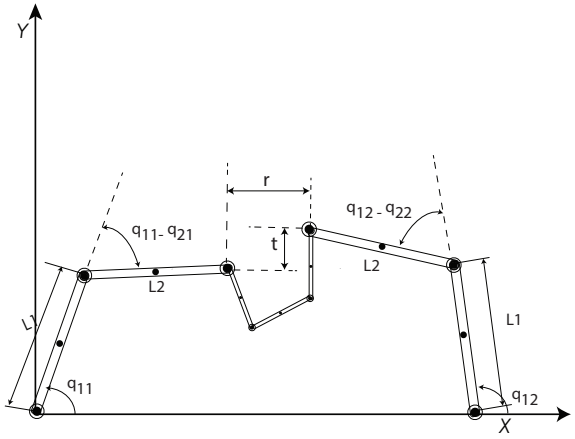


Figure 25: Dexterous manipulation Setup for 3-link object with two 2R planar robots

The joint variable q is defined as the vector of the proximal and the distal joints for both the 2R planar robots.

$$q = \begin{bmatrix} q_{11} \\ q_{21} \\ q_{12} \\ q_{22} \end{bmatrix} \tag{14.6}$$

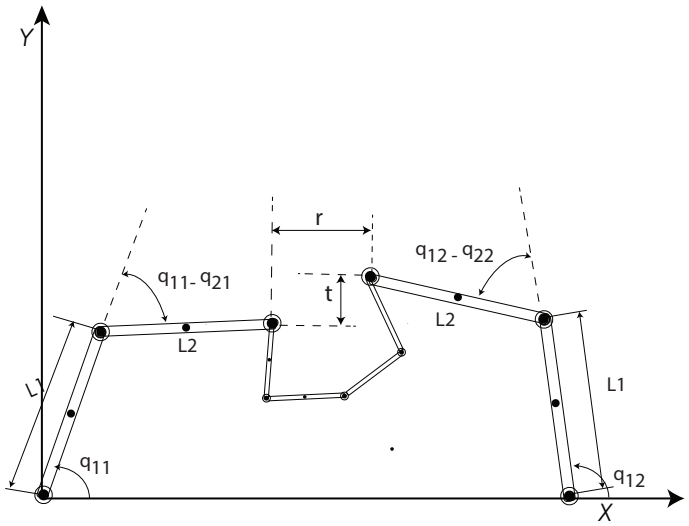


Figure 26: Dexterous manipulation Setup for 4-link object with two 2R planar robots

14.4 Definition of First Task

As defined in Section 13.3 we define our first task using the two manipulation variable r and t

$$r = L_1 \cdot cos(q_{12}) + L_2 \cdot cos(q_{22}) - L_1 \cdot cos(q_{11}) - L_2 \cdot cos(q_{21}) \tag{14.7}$$
$$t = L_1 \cdot sin(q_{12}) + L_2 \cdot sin(q_{22}) - L_1 \cdot sin(q_{11}) - L_2 \cdot sin(q_{21}) \tag{14.8}$$

This is the relative positioning of the two end-effectors of the 2R planar robots. r is the horizontal distance and t is the vertical distance between the end-effectors.

$$T_1 = \begin{bmatrix} r \\ t \end{bmatrix} \tag{14.9}$$

So now we can define the Jacobian of the first task by partial differential of the function defining the manipulation variables of the first task.

$$\dot{T_1} = J_1 \cdot \dot{q} \tag{14.10}$$

$$J_1 = \begin{bmatrix} L_1 \cdot sin(q_{11}) & L_2 \cdot sin(q_{12}) & -L_1 \cdot sin(q_{12}) & -L_2 \cdot sin(q_{22}) \\ -L_1 \cdot cos(q_{11}) & -L_2 \cdot cos(q_{12}) & L_1 \cdot cos(q_{12}) & L_2 \cdot cos(q_{22}) \end{bmatrix} \tag{14.11}$$

This task ensured the trajectory to be followed for the manipulation of the articulated object. Now to ensure the dexterity of the robot we define a second task. As the first task takes up only 2 DoF (r and t), we have another 2 DoFs left for ensuring the dexterity. This will be defined by the relative angle between the two joints of each robot.

14.5 Definition of Second Task

The second task will ensure dexterous manipulation of the object using the two robots. The parameter that defines the dexterity for a 2R planar robot is the angle between its two links. So we can define the second task in the Task-Priority control as follows:

$$T_2 = [q_{11} - q_{21} \ , \ q_{21} - q_{22}]^T \tag{14.12}$$

So now we can define the Jacobian of the second task by partial differential of the function defining the manipulation variables of the second task.

$$\dot{T_1} = J_1 \cdot \dot{q} \tag{14.13}$$

$$J_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{14.14}$$

Velocity control will be adopted to implement the trajectory following by the planar robots. The velocity command for the manipulation variable will be calculated as

$$\dot{T}_i = \dot{T}_i^{\,0}(t) + G_i \cdot (T_i^0(t) - T_i(t)) \tag{14.15}$$

where $\dot{T}_i$ is the velocity command for the i^{th} manipulation variable, $T_i^0(t)$ is the desired trajectory, $\dot{T}_i^{\,0}(t)$ is the derivative of the desired trajectory, and G_i is the scalar feedback coefficient. The values of G_i are determined experimentally.

For task two, the desired trajectory $T_2^0(t)$ can be taken as a constant angle.

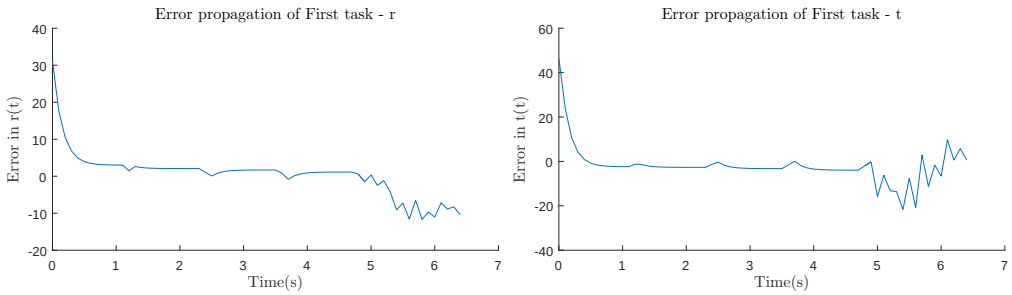
$$T_2^0(t) = [pi/2 \;\; -pi/2] \tag{14.16}$$

The joint velocity command is calculated according to the equation 14.17 as follows:

$$\dot{q} = J_1^+ \dot{T}_1 + (I_n - J_1^+ J_1) J_2^+ \dot{T}_2 \tag{14.17}$$

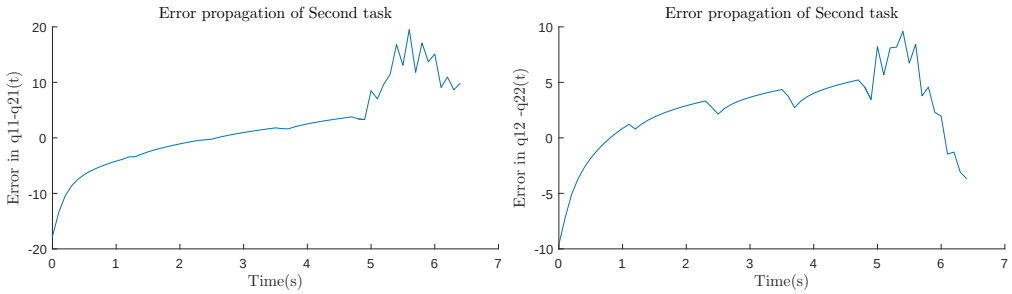
14.6 Results and simulation

Below are the results for initial tests run with the proposed task-priority based control law. The plots show the errors for the manipulation variable and the second task variables. It is seen that the first task is well met for a certain period of time and then at the end it shows larger errors. This is due to improper gain tuning. The gains used are $G_1 = 3$ and $G_2 = 0.3$. Later plots will show a comparison of different gains and its effect on the performance of the control. However, we can see that the task-priority control law always tries to complete the higher priority task, i.e; the first task T_1 while trying to come closer to the second task as can be seen from the error plot for the second task.



(a) Error propagation for r (b) Error propagation for t

Figure 27: Error Propagation for First Task T_1 - $G_1 = 3$; $G_2 = 0.3$



(a) Error propagation for $q_{11} - q_{21}$ (b) Error propagation for $q_{12} - q_{22}$

Figure 28: Error Propagation for Second Task T_2- $G_1 = 3$; $G_2 = 0.3$

14.7 Gain Tuning for the two Tasks

We take up the study of the effect of changing gains G_1 and G_2 on the performance of the system. We will be testing a trajectory that has been generated using the previously elaborated methods. The goal is to manipulate the object from an initial configuration of [-150,320] to a final configuration of [-40,200], where the values represent the relative angles between the links of the object. The trajectory is obtained first in terms of the x and y coordinates of both the robot end-effectors. Then this is transformed to the r and t coordinates representing the relative positioning of the end-effectors. We have a spline array of values for r and t, which we input to the task-priority control algorithm and also differentiate to find the manipulation variable velocity.

14.7.1 Tuning the gain G_1

First we keep G_2 constant and change G_1. We take four values of G_1

$$G_1 = \begin{bmatrix} 0.5 \\ 1 \\ 2 \\ 5 \end{bmatrix} \tag{14.18}$$

$$G_2 = 0.3 \tag{14.19}$$

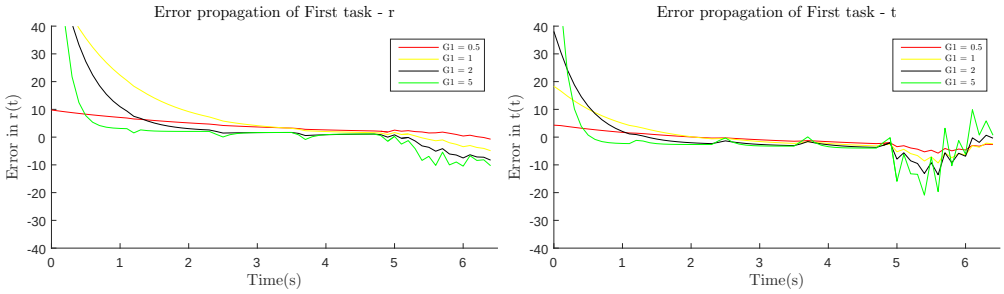
The initial configuration of the robot is:

$$q = \begin{bmatrix} \pi/2 \\ 0 \\ \pi/2 \\ 3\pi/2 \end{bmatrix}$$

This leads to an initial T1 as

$$T1_{stby} = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \tag{14.20}$$

The results for the different values of G_1 are shown below. In Fig 28 the errors of the task 1 T_1 implementation is shown when G_1 is changed while keeping G_2 at a constant value of 0.3. As can be seen the gain in the range of 0.5 - 1 gives good performance and is stable till the end of the task execution. Higher values of gain G_1 shows unfavorable behavior by the end of the simulation.

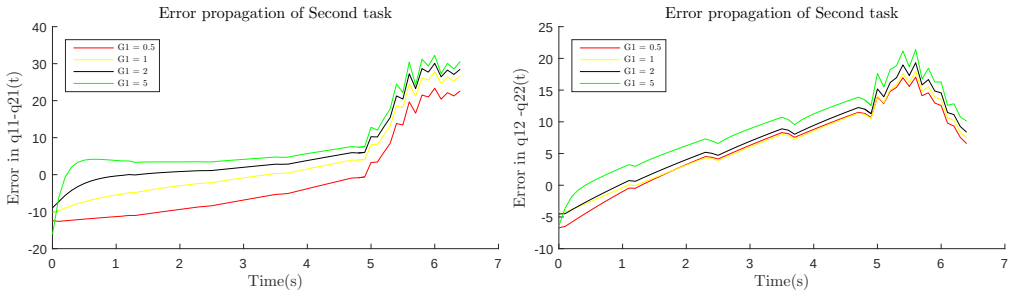


(a) Error propagation for r (b) Error propagation for t

Figure 29: Error Propagation for first task T_1 when gain G1 is changed

Fig 34 shows the error plot for the evolution of the first task T_1 when G_1 is changed while keeping G_2 at a constant value of 0.3. The plots show similar characteristics but slight changes in the value of the error. But it is notable that the

behavior of the error in the case of task 2 is such that the control law tries to bring it closer to the reference values as much as possible.



(a) Error propagation for $q_{11} - q_{21}$ (b) Error propagation for $q_{21} - q_{22}$

Figure 30: Error Propagation for Second task T_2 when gain G1 is changed

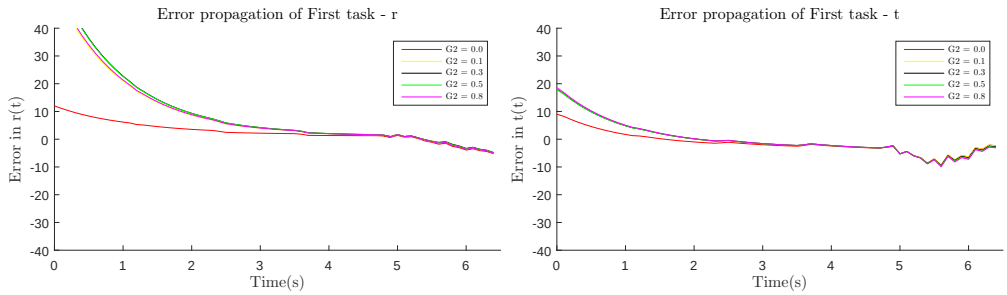
14.7.2 Tuning the gain G_2

Fig 35 shows the error for the first task when G_2 is changed and G_1 is kept at a constant value.

$$G_2 = \begin{bmatrix} 0.0 \\ 0.1 \\ 0.3 \\ 0.5 \\ 0.8 \end{bmatrix} \tag{14.21}$$

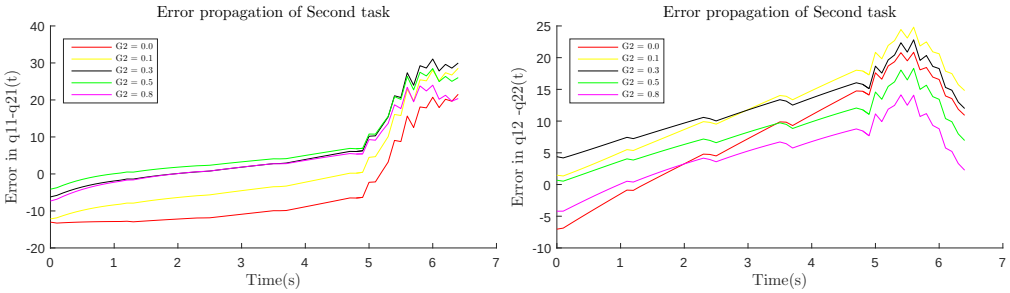
$$G_1 = 1 \tag{14.22}$$

As can be seen from 35 change in the gain G_2 has very little effect on the execution of task 1. it almost remains the same. But from the plot for the evolution of task 2 from the figure 36 we can see that as G_2 is increased there is considerable decrease in the error of in task 2.



(a) Error propagation for r (b) Error propagation for t

Figure 31: Error Propagation for first task T_1 when gain G2 is changed



(a) Error propagation for $q_{11} - q_{21}$ (b) Error propagation for $q_{21} - q_{22}$

Figure 32: Error Propagation for Second task T_2 when gain G2 is changed

14.8 Effect of standby position of robots on the control

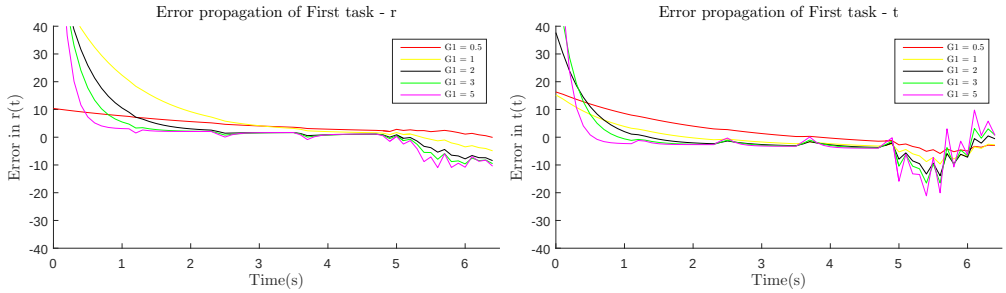
We try the same with another stand-by position for the robot and the results are shown below:

The initial cofiguration of the robot is:

$$q = \begin{bmatrix} \pi/2 \\ \pi/4 \\ \pi/2 \\ 3\pi/4 \end{bmatrix}$$

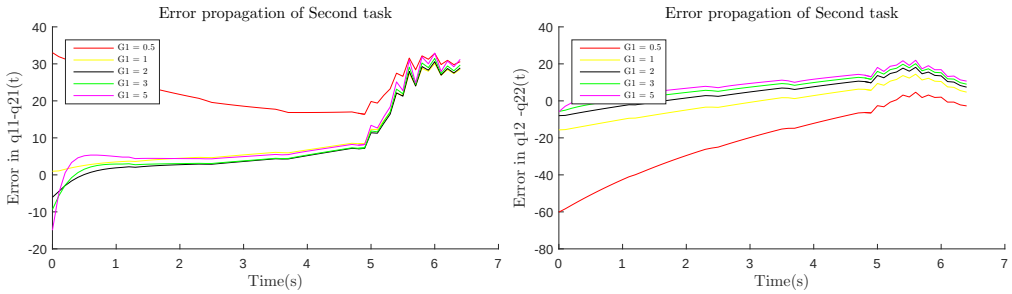
This leads to an initial T1 as

$$T1_{stby} = \begin{bmatrix} 275.7359 \\ 0 \end{bmatrix} \tag{14.23}$$



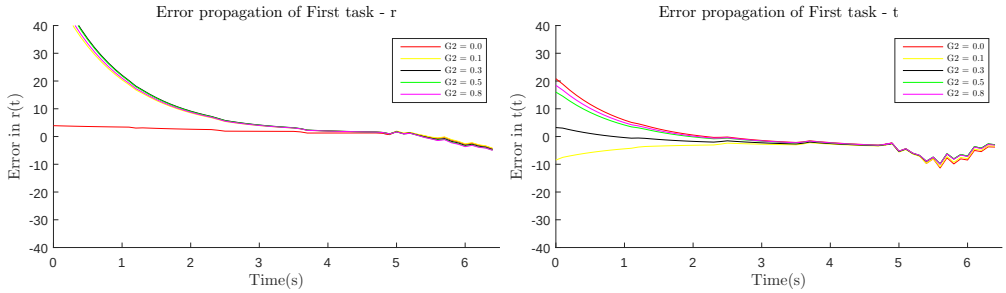
(a) Error propagation for r (b) Error propagation for t

Figure 33: Error Propagation for first task T_1 when gain G1 is changed



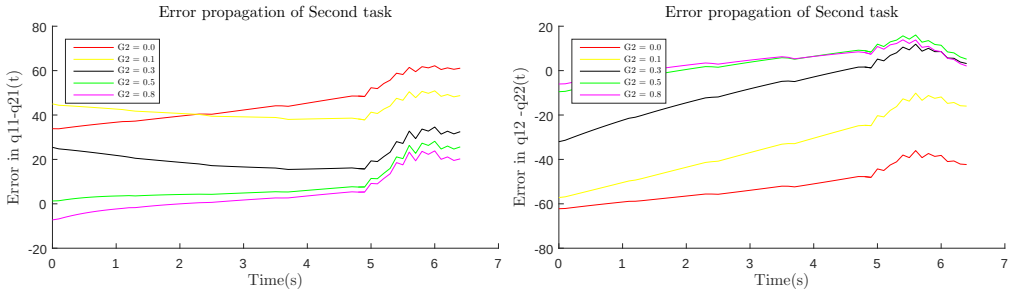
(a) Error propagation for $q_{11} - q_{21}$ (b) Error propagation for $q_{21} - q_{22}$

Figure 34: Error Propagation for Second task T_2 when gain G1 is changed



(a) Error propagation for r (b) Error propagation for t

Figure 35: Error Propagation for first task T_1 when gain G2 is changed

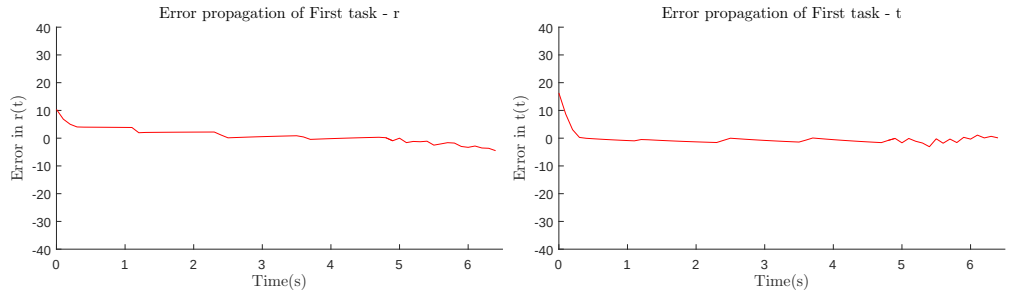


(a) Error propagation for $q_{11} - q_{21}$ (b) Error propagation for $q_{21} - q_{22}$

Figure 36: Error Propagation for Second task T_2 when gain G2 is changed

14.9 Adaptive Gain Change for obtaining minimum error

As is observed from figures 29, the performance changes with respect to time and the values in the spline. This can be overcome by an adaptive gain which changes according to the error evolution. G_1 can change values to obtain the minimum error possible for executing the task T_1. The results with adaptive gain is plotted below. We can clearly see the better error results for task T_1 in Fig.37 and the gain changes has also been shown in 38



(a) Error propagation for r (b) Error propagation for t

Figure 37: Error Propagation for First task T_1 with adaptive gain G_1

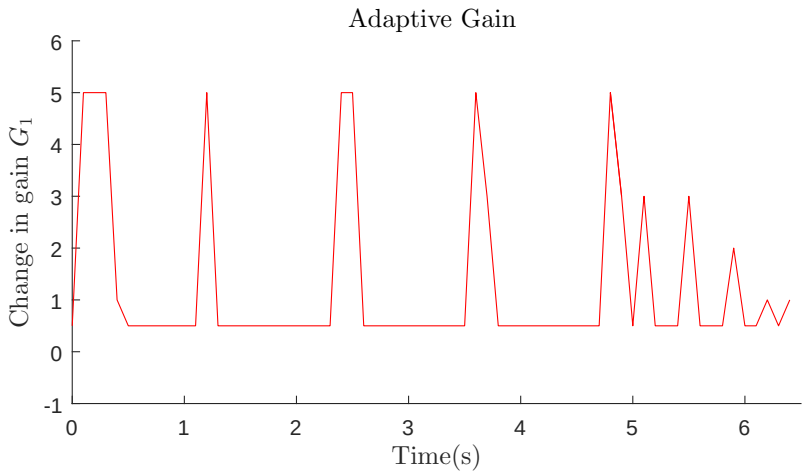
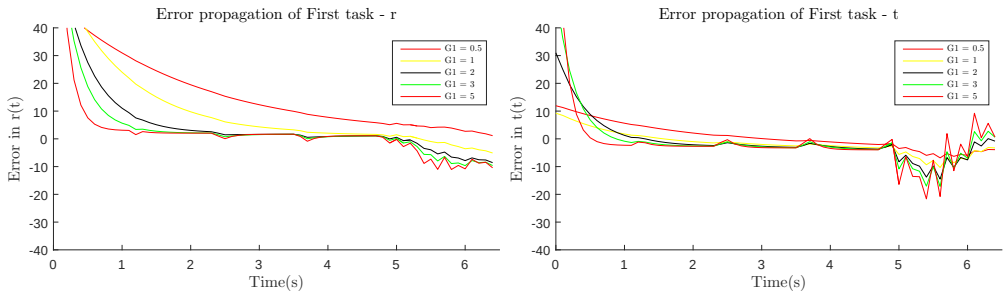


Figure 38: Apdaptive gain G_1

As can be seen, the task errors are reduced and uniform and the gain plot shows that different gains have been adopted for different parts

14.10 Results for task-priority applied to trajectory generated for 4-link objects

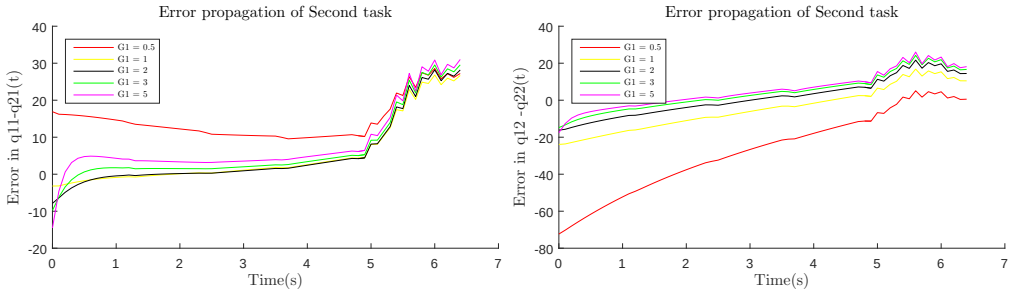
In the figures 39 and 40 the plots show the errors for the tasks T_1 and T_2 execution using different gains G_1. The plots show similar characteristics like that for a 3 link object. The task one converges near to zero and due to high gains in the end part, shows some irregular behavior. Task T_2 in the meanwhile shows also similar results with it coming close to the required value but not completely. But simulations show that the dexterity is maintained and the robot stays away from singularities.



(a) Error propagation for r

(b) Error propagation for t

Figure 39: Error Propagation for First task T_1 for trajectory of 4-link object

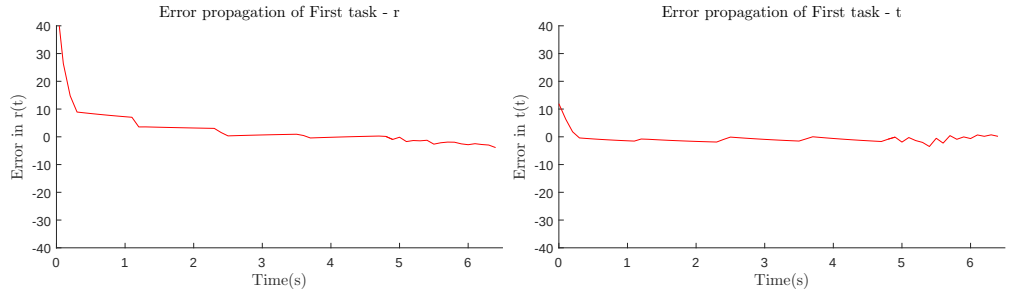


(a) Error propagation for $q_{11} - q_{21}$ (b) Error propagation for $q_{12} - q_{22}$

Figure 40: Error Propagation for First task T_2 for trajectory of 4-link object

14.11 Adaptive Gain for trajectory of Four link object

In this section, the results with changing gain for trajectory generated for four link object is shown. The errors show much more uniform behavior now that this type of gain is used and the results can be seen in 41.



(a) Error propagation for r (b) Error propagation for t

Figure 41: Error Propagation for First task T_1 with adaptive gain G_1 - four link object

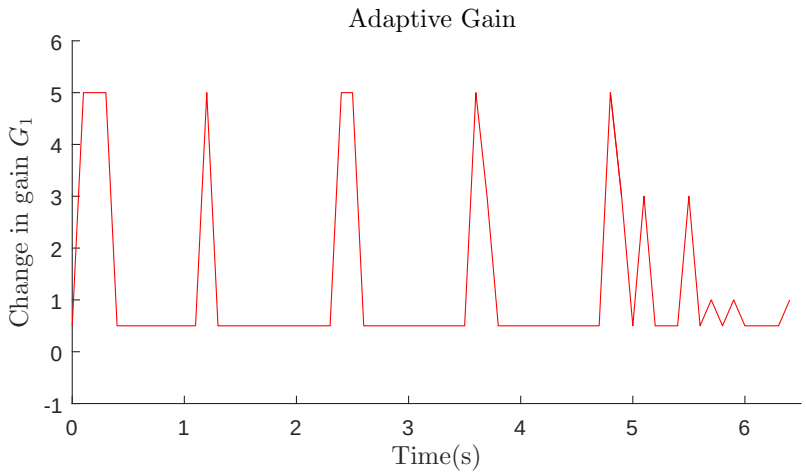


Figure 42: Apdaptive gain G_1 for four link object

In Figure 42, the values of gain adopted are plotted with respect to time.

15 Conclusion and Future Works

A study has been conducted on the modeling of articulated objects, model has been validated using ADAMS and Simulink. The models has been used to generated equilibrium configurations of the object under different possible manipulation positions. Study has been conducted on three and four link objects. These equilibrium configurations has been used to build a configuration space. These configuration spaces has been used to determine trajectory for the manipulation points for achieving different desired configurations without crossing singularity and without losing stability. Path planning / motion planning algorithms has been extended to generate trajectory from the configuration space. A-star and Probabilistic Roadmap methods has been used for this purpose. The generated trajectory has been implemented using two 2R planar robots to manipulated the proposed articulated object. Simulations has been successfully been performed and results documented. Redundant co-manipulation using the two 2R planar robots has been studied using a task-priority approach has been implemented for achieving the same.

15.1 Future Works

As future work, the following steps could be taken:

- Modeling could be extended to cable type objects and other deformable objects
- Study of dynamics of the object during manipulation could be done so that we can manipulate the object in a dynamic way
- Implementation of deformable object manipulation using real robots
- Implementing Eye-to-hand visual servoing on the robots for accurate manipulation

References

[1] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 323–329 vol.1, Oct 1998.

[2] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, Jun 1997.

[3] Jadav Das and Nilanjan Sarkar. Autonomous shape control of a deformable object by multiple manipulators. *Journal of Intelligent and Robotic Systems*, 63:3–27, 2011.

[4] Bernard Espiau and François Chaumette. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[5] F. Flacco, A. De Luca, and O. Khatib. Prioritized multi-task motion control of redundant robots under hard joint constraints. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3970–3977, Oct 2012.

[6] Dominik Henrich and Heinz Wörn. *Robot Manipulation of Deformable Objects*. Springer, London, 2000.

[7] Ayanna M. Howard and George A. Bekey. Recursive learning for deformable object manipulation. *ICAR*, pages 939–944, 1997.

[8] Ayanna M. Howard and George A. Bekey. Intelligent learning for deformable object manipulation. *Autonomous Robots*, 9(1):51–58, 2000.

[9] David Hsu, Jean-Claude Latombe, and Hanna Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Rob. Res.*, 25(7):627–643, July 2006.

[10] Seth Hutchinson, gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[11] Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber. Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks. *9 IEEE International Conference on Robotics and Automation*, 2009.

[12] K. Kobayashi and T. Yoshikawa. Controllability of under-actuated planar manipulators with one unactuated joint. *The International Journal of Robotics Research*, 21(5-6):555–561, 2002.

[13] Yannian Liu, Xin Xin, and Jinglong Wu. On n-link planar revolute robot: Motion equations and new properties. *Proceeding of the 11th World Congress on Intelligent Control and Automation Shenyang, China*, pages 3286–3291, June 2014.

[14] P. Long, W. Khalil, and P. Martinet. Dynamic modeling of cooperative robots holding flexible objects. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 182–187, July 2015.

[15] E. Marchand and François Chaumette. Virtual visual servoing: A framework for real-time augmented reality. *Drettakis, G. and Seidel, H.-P. EUROGRAPHICS 2002 Conference Proceeding,Saarebrün,Germany.*, 21(3):289–298, 2002.

[16] U. Meier, O. López, C.Monserrat, M.C. Juan, and M. Alca niz. Real-time deformable models for surgery simulation: a survey. *Computer Methods and Programs in Biomedicine*, 77:183–197, 2005.

[17] Essahbi Nabil and Alric Matthieu. In process controlling of a multi arms robotic cell for muscle anatomical cutting: focus on mechanical modeling and characterization of soft materials. *ARMS : A multi arms robotic system for muscle separation*, 2014.

[18] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

[19] David Navarro-Alarcón, Yun-Hui Liu, José Gualdalupe Romero, and Peng Li. Model-free visually servoed deformation control of elastic objects by robot manipulators. *IEEE Transactions on Robotics and Automation*, 29(6):1457–1468, 2013.

[20] Georg Nebehay and Roman Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. *Computer Vision and Pattern Recognition*, 2015.

[21] Chendeb S. Chirurgie virtuelle: modélisation temps réel des tissus mous, interactions et systéme haptique dédié. *Doctoral Dissertation, École Nationale Supérieurs des Mines de Paris*, 2007.

[22] Shinichi Tokumoto, Yoshiaki Fujita, and Shinichi Hirai. Deformation modeling of viscoelastic objects for their shape control. *Proceedings of the 1999 IEEE*

International Conference on Robotics and Automation Detroit. Michigan, pages 767–772, 1999.

[23] T. Wada, S. Hirai, H. Mori, and S. Kawamura. Robust manipulation of deformable objects using model based technique. *Articulated Motion and Deformable Objects,Palma de Mallorca, Springer Berlin Heidelberg*, pages 1–14, 2000.

[24] T. Wada, S. Hirai, H. Mori, and S. Kawamura. Robust manipulation of deformable objects by a simple pid feedback. *IEEE International Conference on Robotics and Automation*, pages 85–90, 2001.

[25] Y. Wang, C. Smith, Y. Karayiannidis, and P. gren. Cooperative control of a serial-to-parallel structure using a virtual kinematic chain in a mobile dual-arm manipulation application. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 2372–2379, Sept 2015.

[26] Zhanying Zhang and Ziping Zhao. A multiple mobile robots path planning algorithm based on a-star and dijkstra algorithm. *International Journal of Smart Home*, 8(3):75–86, 2014.