CORO IMARO M2
Advanced Robotics

AMORO Lab Report

November 8, 2020

Kinematics and Dynamics Of a Biglide

Submitted by
Gowri UMESH
Niranjana Subha RAJEEV

Contents

List of Figures

Chapter 1

Introduction

In this lab,a Biglide robot is studied. The Geometric, Kinematic and Dynamic models of the robot are created in MATLAB and then compared with the results obtained with ADAMS. Controllers are designed to track trajectories in simulation.

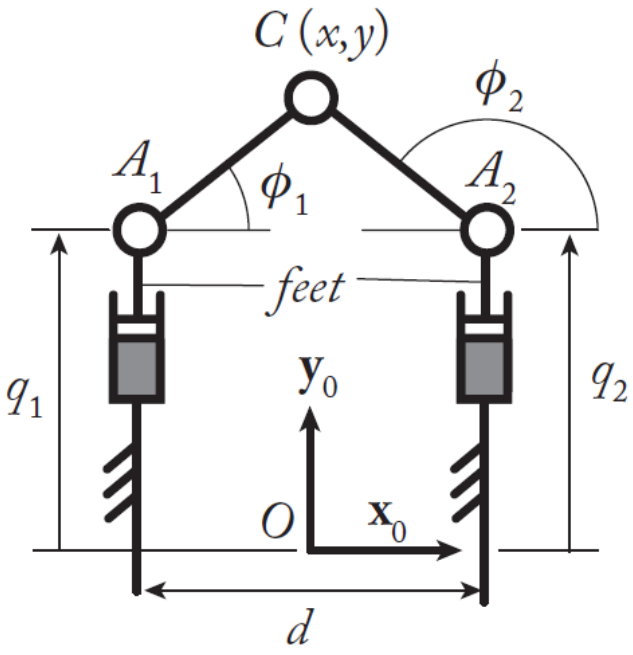

FIGURE 1.1: *The Biglide Robot*

The data already given are as follows:
For the mechanism of the ADAMS mock-up, the geometric parameters are:

- Bar length (all bars have equal length) l = 0.3606 m
- Distance between the two active joints: d = 0.4 m

The two prismatic joints (q1 and q2) are actuated.
The base dynamic parameters are:

- mp = 3 kg the mass of the end-effector
- mf = 1 kg the mass of each foot

Chapter 2

Organisation of Files

The zip folder *RajeevUmeshLab4Files.zip* contains the deliverables and are organised as follows.

- Report - *RajeevUmeshAMOROReport.pdf*
- Sub Folder -*ModelVerification*
  - Adams File : *Biglide.bin*
  - Simulink file :*BiglideModel.slx*
  - MATLAB plant : *ControlsPlantModel.m*
- Sub Folder - *KinematicControl*
  - Adams File :*BiglideKinematicControl.bin*
  - Simulink file : *KinematicControl.slx*
  - MATLAB plant : *ControlsPlantKinematicControl.m*
- Sub Folder - *ComputedTorqueControl*
  - Adams File :*BiglideTorqueControl.bin*
  - Simulink file : *ComputedTorqueControl.slx*
  - MATLAB plant :*ControlsPlantTorqueControl.m*

Chapter 3

Mathematical expressions of the Biglide Mechanism

This chapter explains the computation of mathematical expressions of the geometric,kinematic and dynamic models of the Biglide mechanim.

3.1 Direct Geometric Model

The Direct Kinematic Model of the Biglide is given as follows:

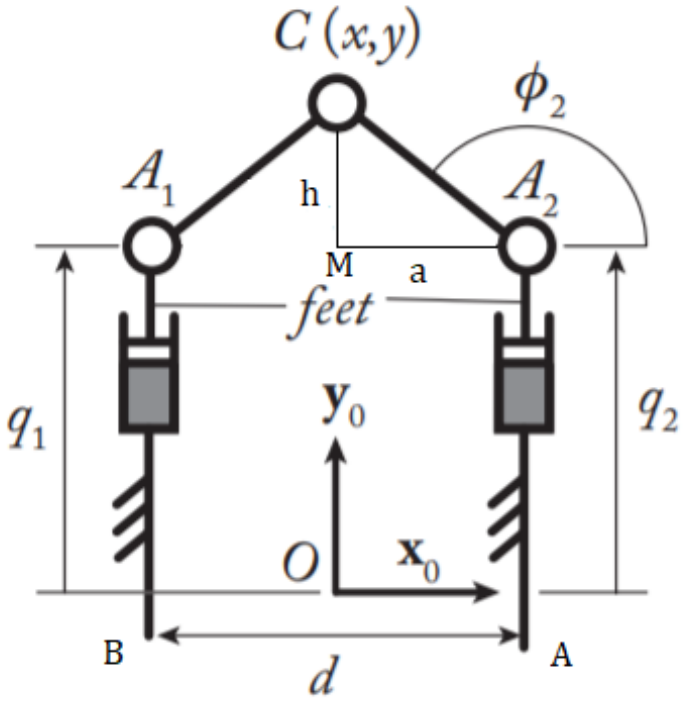

FIGURE 3.1: *Kinematic model of the Biglide*

The direct geometric model of the biglide computes the end-effector position (x,y) and is formulated as follows:
From figure 3.1 we know that ,

$$\mathbf{OC} = \mathbf{OA} + \mathbf{AA_2} + \mathbf{A_2M} + \mathbf{MC} \tag{3.1}$$

$$\mathbf{OC} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \mathbf{OA} = \begin{bmatrix} \frac{d}{2} \\ 0 \end{bmatrix} \qquad \mathbf{AA_2} = \begin{bmatrix} 0 \\ q_2 \end{bmatrix}$$

$$\mathbf{A_2M} = 0.5(\mathbf{A_2A_1})$$

$$\mathbf{A_2M} = 0.5(-\mathbf{AA_2} - \mathbf{OA} + \mathbf{OB} + \mathbf{BA_1}) \tag{3.2}$$

where

$$\mathbf{OB} = \begin{bmatrix} -\frac{d}{2} \\ 0 \end{bmatrix}, \mathbf{BA_1} = \begin{bmatrix} 0 \\ q_1 \end{bmatrix}$$

so

$$\mathbf{A_{2M}} = \begin{bmatrix} \frac{d}{2} \\ \frac{q_1}{2} - \frac{q_2}{2} \end{bmatrix}$$

$$\mathbf{MC} = \gamma \frac{h}{a} E\mathbf{A_2M}$$

with

$$a = norm(\mathbf{A_2M}) \qquad h = \sqrt{l^2 - a^2}$$

3.2 Passive Angle Calculation

Passive joint angles ϕ_1, ϕ_2 is computed in this section.In order to compute the passive joints equations, we must write the vector **OC** from the left and right limbs respectively. We get

$$x = \frac{d}{2} + lcos\phi_2$$
$$y = q_2 + lsin\phi_2$$

which gives

$$\phi_2 = tan^-1(\frac{y - q_2}{x - \frac{d}{2}})$$

similarly

$$\phi_1 = tan^-1(\frac{y - q_1}{x + \frac{d}{2}})$$

3.3 Direct Kinematic Model

From the Direct Geometric Model , The equations can be rewritten as:

$$\mathbf{OC} = -\frac{d}{2}\mathbf{i} + q_1\mathbf{U_1} + l\mathbf{V_1} \tag{3.3}$$

$$\mathbf{OC} = \frac{d}{2}\mathbf{i} + q_2\mathbf{U_2} + l\mathbf{V_2} \tag{3.4}$$

To obtain the first order kinematic model, we differentiate the loop-closure equations

$$\dot{\mathbf{C}} = \dot{q_1}\mathbf{U_1} + lE\dot{\phi_1}\mathbf{V_1} \tag{3.5}$$

$$\dot{\mathbf{C}} = \dot{q_2}\mathbf{U_2} + lE\dot{\phi_2}\mathbf{V_2} \tag{3.6}$$

where E = $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ is the rotation matrix to show rotation by 90°in a plane.

Multiplying with $\mathbf{V_i}^T$on 3.3 AND 3.4 we get

$$\begin{bmatrix} \mathbf{V_1}^T \\ \mathbf{V_2}^T \end{bmatrix} \dot{\mathbf{C}} = \begin{bmatrix} \mathbf{V_1}^T\mathbf{U_1} & 0 \\ 0 & \mathbf{V_2}^T\mathbf{U_2} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} \tag{3.7}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = A^{-1}B \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} \tag{3.8}$$

Where

$$A = \begin{bmatrix} \mathbf{V_1}^T \\ \mathbf{V_2}^T \end{bmatrix}$$

and

$$B = \begin{bmatrix} \mathbf{V_1}^T\mathbf{U_1} & 0 \\ 0 & \mathbf{V_2}^T\mathbf{U_2} \end{bmatrix}$$

3.4 Passive Angle velocity

Multiplying with E$\mathbf{V_i}^T$ on 3.3 and 3.4 we get

$$\begin{bmatrix} (E\mathbf{V_1})^T \\ (E\mathbf{V_2})^T \end{bmatrix} \dot{\mathbf{C}} - \begin{bmatrix} (E\mathbf{V_1})^T\mathbf{U_1} & 0 \\ 0 & (E\mathbf{V_2})^T\mathbf{U_2} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} l & 0 \\ 0 & l \end{bmatrix} \begin{bmatrix} \dot{\phi_1} \\ \dot{\phi_2} \end{bmatrix} \tag{3.9}$$

Then, the above equations can be re-written in a vector-matrix form as follows,

$$\mathbf{J_t}\dot{\mathbf{P}} = \mathbf{J_{ta}}\dot{\mathbf{q_a}} + \mathbf{J_{td}}\dot{\mathbf{q_d}} \tag{3.10}$$

$$\dot{q}_d = J_t^{-1}(J_t\dot{P} - J_{ta}\dot{q}_a) \tag{3.11}$$

The above equation gives the passive angle velocity. where

$$\mathbf{J_t} = \begin{bmatrix} (E\mathbf{V_1})^T \\ (E\mathbf{V_2})^T \end{bmatrix}, \mathbf{J_{ta}} = \begin{bmatrix} (E\mathbf{V_1})^T\mathbf{U_1} & 0 \\ 0 & (E\mathbf{V_2})^T\mathbf{U_2} \end{bmatrix}, \mathbf{J_{td}} = \begin{bmatrix} l & 0 \\ 0 & l \end{bmatrix}$$

3.5 Second Order Kinematics

The second-order kinematic equations can be obtained by differentiating equations 3.3 and 3.4

$$\ddot{C} = \ddot{q_i}\mathbf{U_i} + lE\ddot{\phi_i}\mathbf{V} + lEE\ddot{\phi_i}^2\mathbf{V_i} \tag{3.12}$$

Multiplying with $\mathbf{V_i}^T$ on 3.12 we get

$$\mathbf{V_i}^T\ddot{C} = \ddot{q_i}\mathbf{V_i}^T\mathbf{U_i} + +l\mathbf{V_i}^TE^2\ddot{\phi_i}^2\mathbf{V_i} \tag{3.13}$$

$$\mathbf{V_i}^T\ddot{C} = \ddot{q_i}\mathbf{V_i}^T\mathbf{U_i} - l\phi_i{}^2 \tag{3.14}$$

$$\begin{bmatrix}\mathbf{V_1}^T\\ \mathbf{V_2}^T\end{bmatrix}\ddot{\mathbf{C}} = \begin{bmatrix}\mathbf{V_1}^T\mathbf{U_1} & 0\\ 0 & \mathbf{V_2}^T\mathbf{U_2}\end{bmatrix}\begin{bmatrix}\dot{q_1}\\ \dot{q_2}\end{bmatrix} - \begin{bmatrix}l & 0\\ 0 & l\end{bmatrix}\begin{bmatrix}\dot{\phi_1}^2\\ \dot{\phi_2}^2\end{bmatrix} \tag{3.15}$$

3.6 Passive Angle Acceleration

Multiplying with E$\mathbf{V_i}^T$ on 3.12 we get

$$(E\mathbf{V_i})^T\ddot{C} = \ddot{q_i}(E\mathbf{V_i})^T\mathbf{U_i} + l(E\mathbf{V_i})^TE\mathbf{V_i}\ddot{\phi_i} + l(E\mathbf{V_i})^TE^2\mathbf{V_i}\ddot{\phi_i}^2 \tag{3.16}$$

$$(E\mathbf{V_i})^T\ddot{C} = \ddot{q_i}(E\mathbf{V_i})^T\mathbf{U_i} + l\ddot{\phi_i}^2 \tag{3.17}$$

Which can be written as:

$$\begin{bmatrix}l & 0\\ 0 & l\end{bmatrix}\begin{bmatrix}\ddot{\phi_1}\\ \ddot{\phi_2}\end{bmatrix} = \begin{bmatrix}(E\mathbf{V_1})^T\\ (E\mathbf{V_2})^T\end{bmatrix}\ddot{\mathbf{C}} - \begin{bmatrix}(E\mathbf{V_1})^T\mathbf{U_1} & 0\\ 0 & (E\mathbf{V_2})^T\mathbf{U_2}\end{bmatrix}\begin{bmatrix}\ddot{q_1}\\ \ddot{q_2}\end{bmatrix} \tag{3.18}$$

3.7 Dynamic Model

Few assumptions are made in the derivation of the dynamic equations:

- The Potential Energy **U** is zero
- Body 3 and 4 are not taken into account as their masses are zero

For Body 1:

$$\mathbf{v_1} = \dot{q_1}\mathbf{y_1} \tag{3.19}$$

$$\boldsymbol{\omega_1} = 0 \tag{3.20}$$

For Body 2:

$$\mathbf{v_2} = \dot{q_2}\mathbf{y_2} \tag{3.21}$$

$$\boldsymbol{\omega_2} = 0 \tag{3.22}$$

The kinetic energies for both bodies are:

$$E_1 = \frac{1}{2} m_1 \dot{q_1}^2 \tag{3.23}$$

$$E_2 = \frac{1}{2} m_1 \dot{q_2}^2 \tag{3.24}$$

The total Kinetic Energy is

$$E = \frac{1}{2}(m_1 \dot{q_1}^2 + m_2 \dot{q_2}^2) \tag{3.25}$$

Now by the Lagrange equation ,The Lagrangian L = E-U, but as the potential energy U is zero here it reduces to L = E .

The input torques/forces can be obtained by the following equation:

$$\tau_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q_i}}) - (\frac{\partial L}{\partial q_i}) \tag{3.26}$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q_1}}) = m_1 \ddot{q_1}, \frac{d}{dt}(\frac{\partial L}{\partial \dot{q_2}}) = m_2 \ddot{q_2}, (\frac{\partial L}{\partial q_1}) + (\frac{\partial L}{\partial q_2}) = 0$$

Which gives:

$$\tau_1 = m_1 \ddot{q_1} \tag{3.27}$$

$$\tau_2 = m_2 \ddot{q_2} \tag{3.28}$$

$$\tau_a = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \tag{3.29}$$

Above equation gives the inverse dynamic model of the system. Also, we know that for the end-effector platform force is given by

$$\tau_p + B^T \lambda_p = 0$$

$$\tau_p = -B^T \lambda$$

$$A^T \lambda_p = m_p \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$$

$$\lambda = m_p A^{-T} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$$

$$\tau_p = -m_p B^T A^{-T} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$$

$$\tau_{total} = \tau_a + \tau_p$$

Which gives:

$$\tau_{total} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} - m_p B^T A^{-T} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{3.30}$$

The above equations gives us inertia matrix M and the vector of coriolis and Centrifugal effects c.

3.8 Inverse Geometric Model

The IGM of the Biglide mechanism is derived below

From the geometric constraint equation we know that,

$$(x + \frac{d}{2})^2 + (q_1 - y)^2 = l^2 \tag{3.31}$$

$$(x - \frac{d}{2})^2 + (q_2 - y)^2 = l^2 \tag{3.32}$$

From above equations we joint position equations,

$$q_1 = y \pm \sqrt{l^2 - (x + \frac{d}{2})^2} \tag{3.33}$$

$$q_2 = y \pm \sqrt{l^2 - (x - \frac{d}{2})^2} \tag{3.34}$$

Chapter 4

ADAMS Plant

Primiarily for the purpose of co simulation we have to create the ADAMS plant that can be used with MATLAB. The following steps are followed in order to create an ADAMS plant

- Create the state variables for the output
- Impose a motion on active joint variables q1 and q2 (0.1*sin(time) on q1 and -0.1*sin(time) on q2). Note that the exact same input should be given in the SIMULINK scheme to verify the model.
- Create plant output
- Export the plant to MATLAB (involves fake input creation)

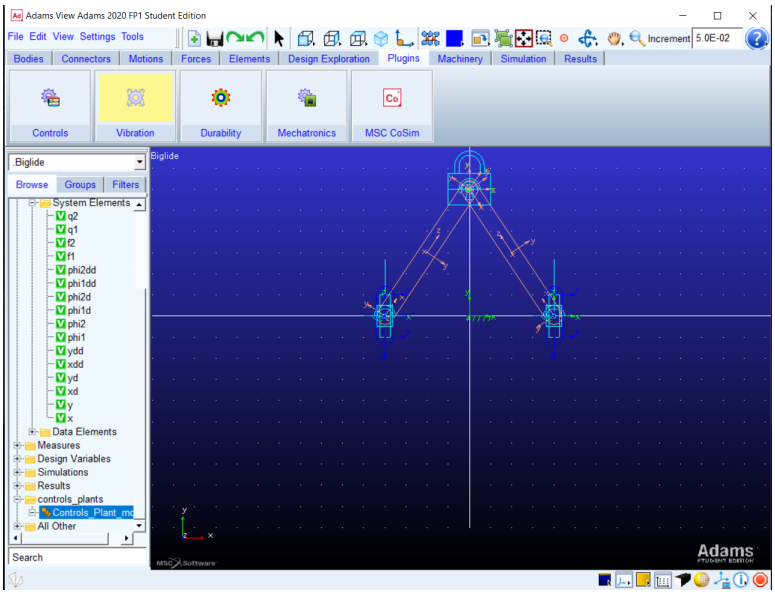

FIGURE 4.1: *Adams environment with State variables and Reference marker*

The above figure shows that state variables and a reference marker created in ADAMS environment.The reference marker was created to ease the measurement between the ground and end effector. It is to be noted that the state variables are added and the inputs (motion/force etc.,)are changed based on the objective in further sections and a different binary file has been saved.

Chapter 5

Direct Kinematic analysis of the Biglide mechanism

The mathematical expressions derived for the Biglide mechanism are compared to the adams' output using Matlab and Simulink. A single simulink template is created to compare the Adams model with the theoretical model. The inputs q1 and q2 are taken as 0.1 sin(time) and -0.1sin(time) in both the environments and then the error between the corresponding outputs are computed for the purpose of comparison.

5.1 Geometric analysis

Equations established in section 3.1 and 3.2 are written into MATLAB function block Direct Geometric model and Passive Angle and the results (x,y) and (phi1,phi2) respectively are compared with the values obtained from ADAMS simulation.

Figure 5.1 shows the error signal of Direct Geometric Model and Figure

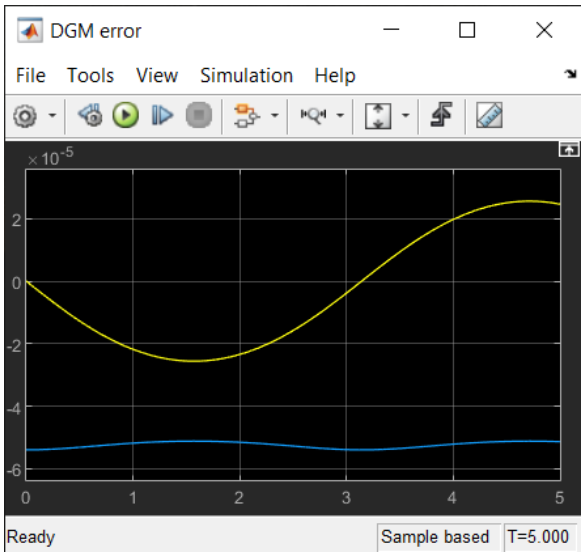

FIGURE 5.1: *Error Signal of Direct Geometric Model*

5.2 shows the error signal of passive angles . It can be seen that the error is very minute and is in the order of 10^{-5} for DGM and 10^{-4} for passive angles

which is negligible . This error could be caused due to the communication delay between MATLAB and ADAMS.

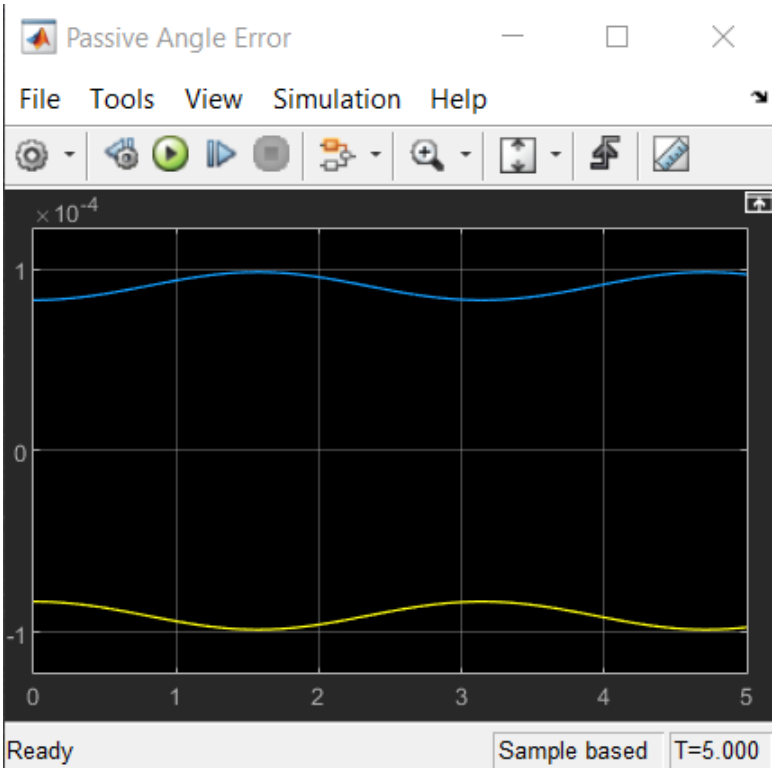

FIGURE 5.2: *Error Signal of Passive Angles*

5.2 Velocity analysis

Equations established in section 3.3 and 3.4 are written into MATLAB function block Direct Kinematic Model and Passive Angle Kinematics and the results (xd,yd) and (phi1d,phi2d) respectively are compared with the values obtained from ADAMS simulation.

- The input to the above mentioned blocks are the joint velocities q1d and q2d which are also corresponding to the Adams input.

- Figure 5.3 shows the error signal of Direct Kinematic Model and Figure 5.4 shows the error signal of passive angle Kinematics . The error in both the figures is extremely close to zero.

5.3 Acceleration Analysis

Equations established in section 3.5 and 3.6 are written into MATLAB function block Direct Kinematic Model 2 and Passive Angle Acceleration and the results end effector and passive angle accelerations (xdd,ydd) and (phi1dd,phi2dd)

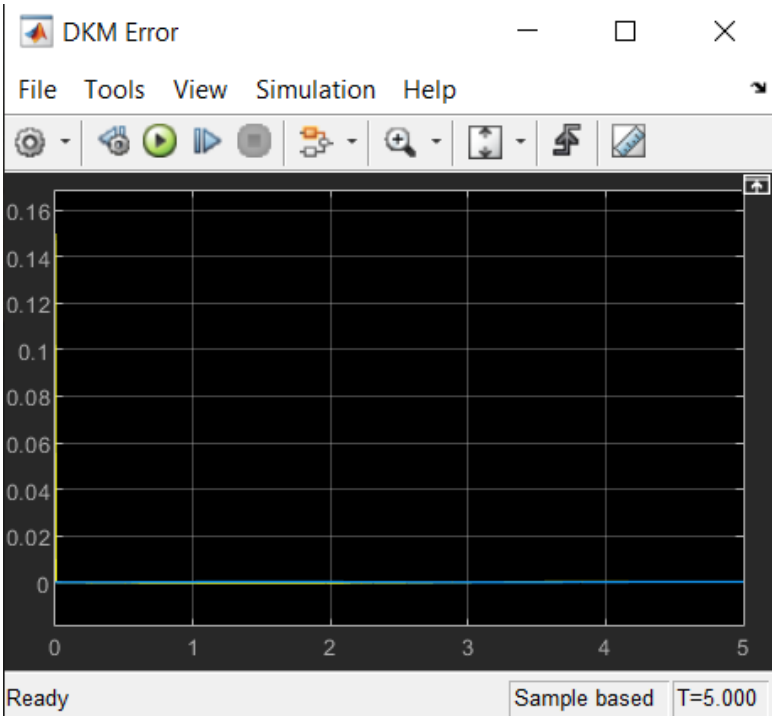

FIGURE 5.3: *Error Signal of Direct Kinematic Model*

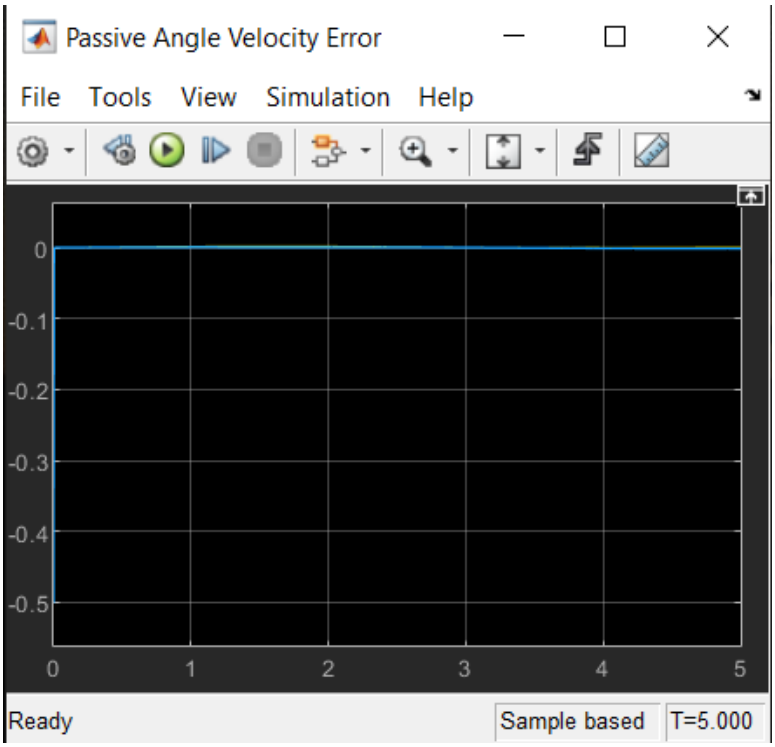

FIGURE 5.4: *Error Signal of passive angle kinematics*

respectively are compared with the values obtained from ADAMS simulation.

- The input to the above mentioned blocks are the joint accelerations

q1dd and q2dd which are also corresponding to the Adams input.

- Figure 5.5 shows the error signal of end effector acceleration and Figure 5.6 shows the error signal of passive angle Kinematics . As seen previously the error in both the signals are extremely close to zero.

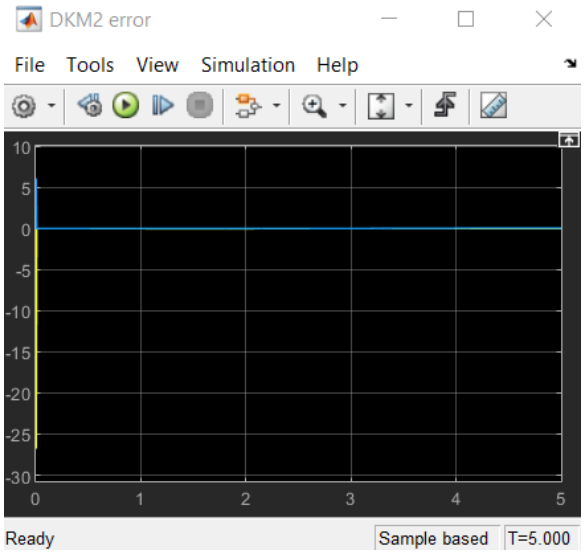

FIGURE 5.5: *Error Signal of Second Order Direct Kinematic Model*

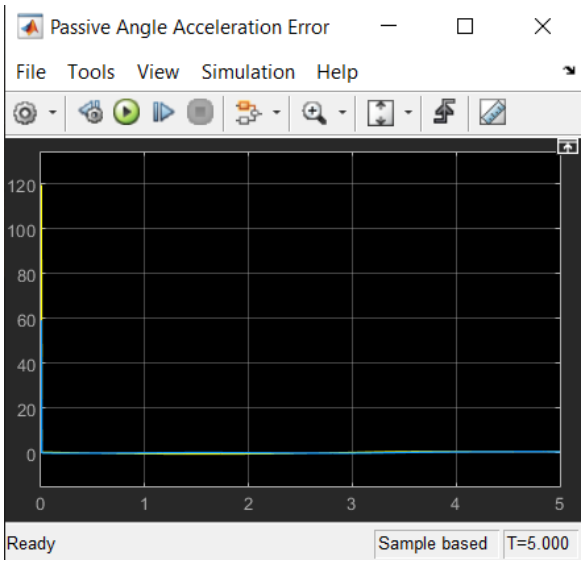

FIGURE 5.6: *Error Signal of Second Order Passive Angle Kinematics*

5.4 Dynamic Analysis

The dynamic behaviour of the Biglide model is simulated and compared with MATLAB model.Equations established in section are written into MATLAB function block Dynamic Model and the output forces obtained (f1,f2)

are compared with the values obtained from ADAMS simulation.
Figure 5.7 shows the force error . As seen previously the error in both the signals are extremely close to zero.

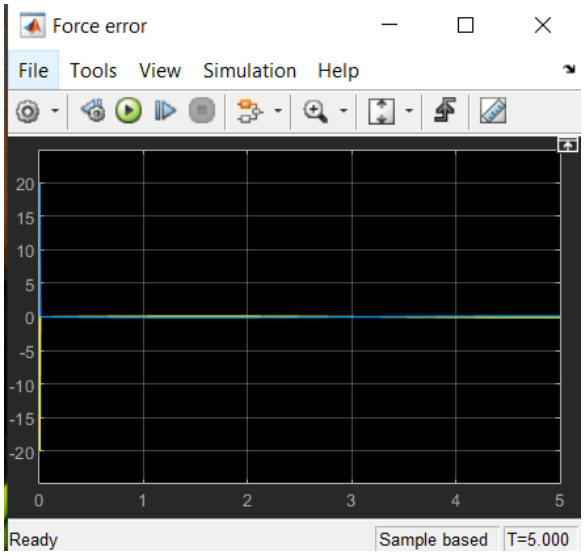

FIGURE 5.7: *Force Error*

Chapter 6

Control co-simulation

Control law is implemented on the Biglide system to follow a trajectory using both Kinematic control and computed torque control.The changes are made in the ADAMS binary file accordingly to create a new MATLAB control plant for the purpose of co-simulation.

6.1 Kinematic Control Law

The objective of the Kinematic control is to track a desired trajectory in Cartesian space. The desired trajectory is expressed by x_t and the tracking error is the difference between the actual trajectory x and the desired trajectory

$$e = x - x_t \tag{6.1}$$

this error has to converge to zero and hence the closed loop behaviour is

$$\dot{e} = -\lambda e \tag{6.2}$$

where λ is the positive proportional control gain.The desired control input is

$$\dot{q} = -B^{-1}A(\dot{x_t} - \lambda e) \tag{6.3}$$

with A and B taken from first order Kinematic equations To implement this Kinematic Control law the following steps are followed:

- Creation of two input variables joint velocities q1d and q2d and effect them as control of the joint motion.The corresponding ADAMS binary file is *BiglideKinematicControl.bin*
- Create a new plant with input q1d and q2d and output the joint position q1 and q2. Corresponding MATLAB plant is named as *Controls-PlantKinematicControl.m*
- Creating a SIMULINK scheme to apply a kinematic control.The corresponding SIMULINK file is *KinematicControl.slx*. Figure 6.1 shows the SIMULINK template used to implement control law

MATAB function block *Trajectory Generation* generates a 3^{rd} order polynomial trajectory x defined with initial and final position and velocities.

$$x = a_1t^3 + a_2t^2 + a_3t + a_4 \tag{6.4}$$

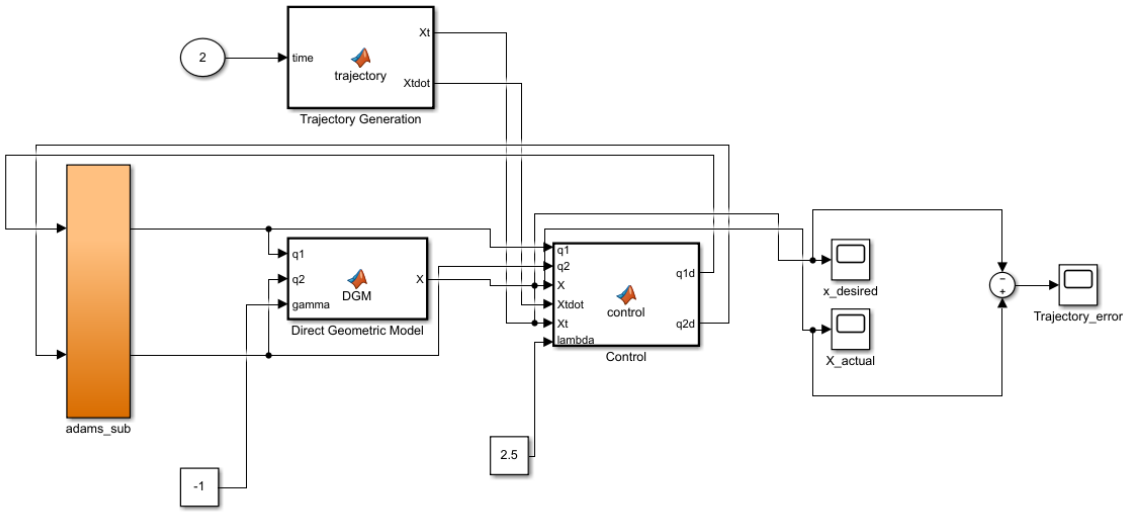

FIGURE 6.1: *SIMULINK template for Kinematic Control*

The initial and final conditions are expressed as

$$\begin{aligned} x(t_i) &= a_1 t_i^3 + a_2 t_i^2 + a_3 t_i + a_4 \\ \dot{x(t_i)} &= 3a_1 t_i^2 + 2a_2 t_i + a_3 \\ x(t_f) &= a_1 t_f^3 + a_2 t_f^2 + a_3 t_f + a_4 \\ \dot{x(t_f)} &= 3a_1 t_f^2 + 2a_2 t_f + a_3 \end{aligned} \tag{6.5}$$

The above equations are solved to obtain polynomial co efficients and are utilized in the block to generate it.

For the purpose of error calculation actual trajectory traced by the robot is being computed in the MATLAB function block *Direct Geometric model* from the ADAMS joint positions q1 and q2. The Control law is implemented in the MATLAB function block *Control* and the trajectory error is plotted.

Fig 6.2 shows the Desired trajectory and fig 6.3 shows the actual trajectory . From fig 6.4 it can be said that the trajectory tracking is successfully implemented using Kinematic control law as the error signal is only in the order of 10^{-5}. This error could be due to the proportional gain tuning and the communication delays between ADAMS and MATLAB.

6.2 Computed Torque Control law

As mentioned in the earlier lab sessions computed torque control is based ion the feedback linearization of the system through the inverse dynamic model.The dynamic model takes the generic form generic form

$$\tau = M\ddot{q_a} + c \tag{6.6}$$

Where M is the inertia matrix c is the vector of coriolis and Centrifugal effects.

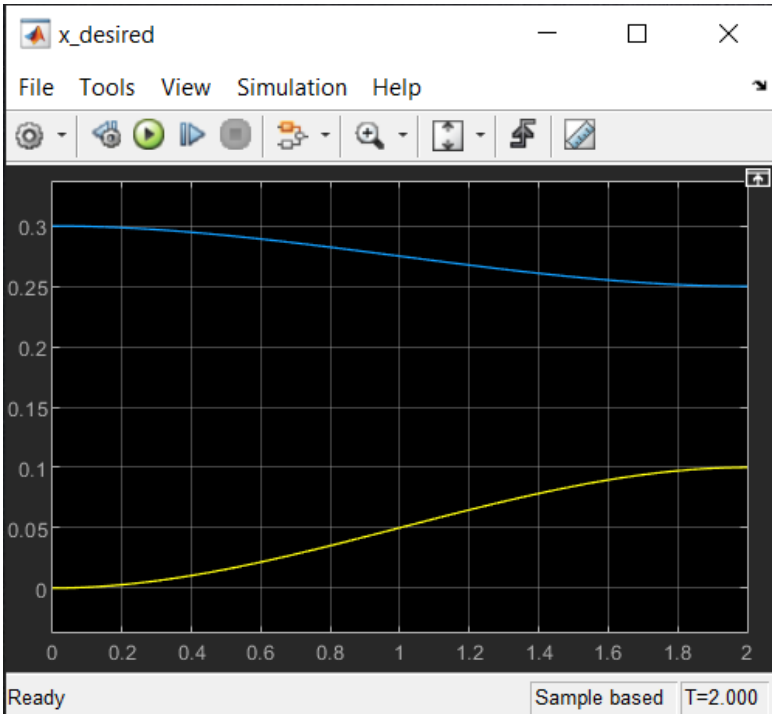

FIGURE 6.2: *Desired Trajectory for Kinematic Control*

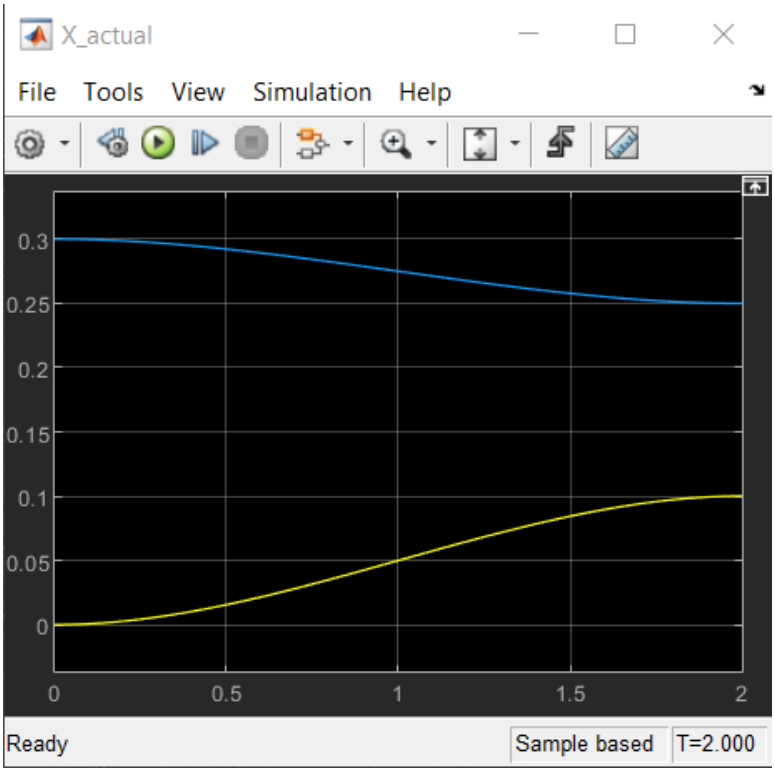

FIGURE 6.3: *Actual Trajectory for Kinematic Control*

Considering τ is the input to the system the auxiliary control can be defined as

$$v = M^{-1}(\tau - c) \tag{6.7}$$

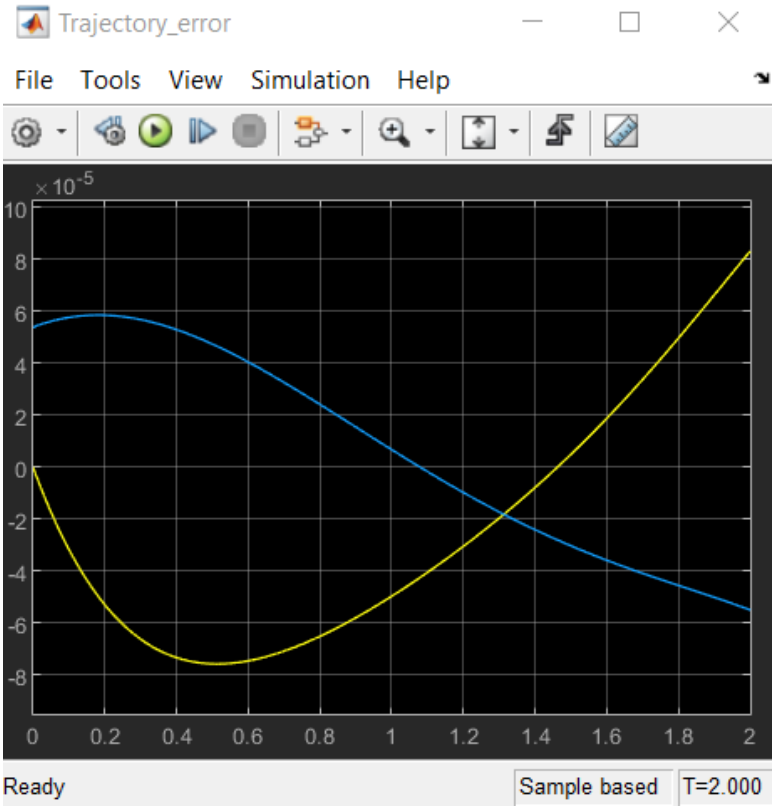

FIGURE 6.4: *Kinematic Control Trajectory error*

Then the auxiliary input corresponds to the robot acceleration.

$$v = \ddot{q}_a \tag{6.8}$$

Fig 6.5 shows the controller scheme.

A PD control law can be implemented and the control law looks like the

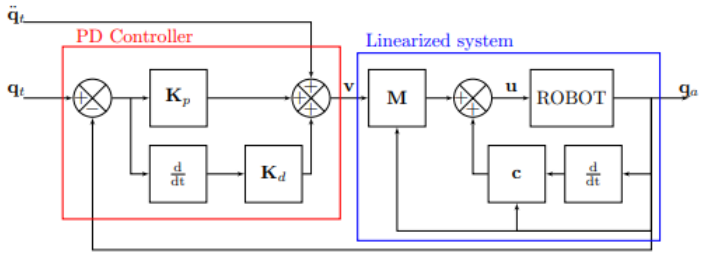

FIGURE 6.5: *Computed Torque Control Scheme*

below

$$v = \ddot{q}_t + K_d(\dot{q}_t - \dot{q}_a) + K_p(q_t - q_a) \tag{6.9}$$

With Kp and Kd being definite positive and hence the control law takes the

form

$$\tau = M(\ddot{q}_t + K_d(\dot{q}_t - \dot{q}_a) + K_p(q_t - q_a)) + c \tag{6.10}$$

In our particular case ,

$$M = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \tag{6.11}$$

and

$$c = m_p B^T A^{-T} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{6.12}$$

To implement the Computed torque Control law the following steps are followed

- Creation of two input variables joint forces f1 and f2 . Remove the motion imposed on the active joints and apply the forces driven by f1 and f2.
- Create a new plant with input f1 and f2 and output the joint position q1 and q2. Corresponding MATLAB plant is named as ControlsPlant-TorqueControl.m
- Creating a SIMULINK scheme to apply the control.Figure 6.6 shows the SIMULINK template used to implement the control law

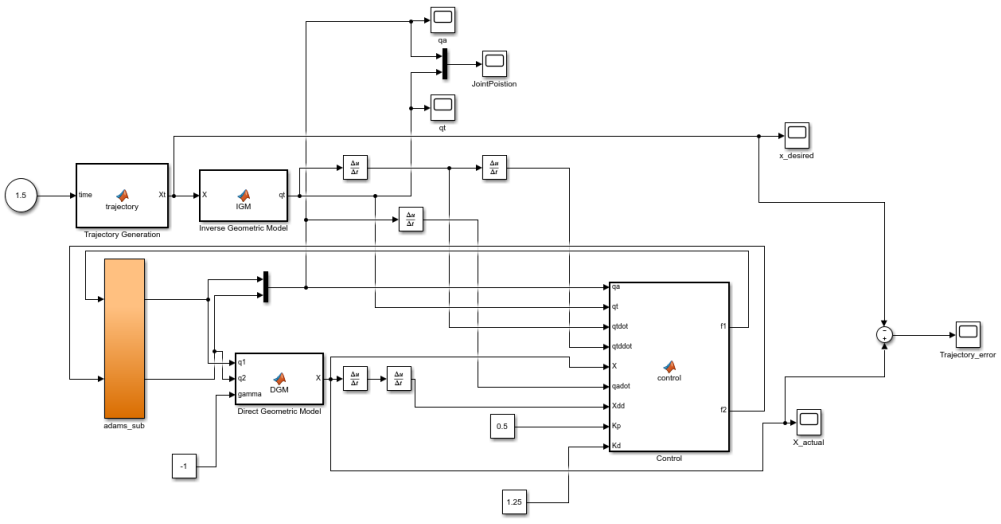

FIGURE 6.6: *SIMULINK template for Computed Torque Control*

Trajectory is generated similar to the previously mentioned way and is implemented in the MATLAB function block *trajectory generation* ,however the comparison is going to be with the joint angles and not the trajectory and

so an Inverse Geometric Model is implemented to retrieve the desired joint angles from the desired trajectory. IGM established in section 3.8 is implemented in the MATLAB function block *Geometric Model*.

A MATLAB function block *Control* implements the control law and the trajectory error is plotted along with the actual and desired trajectories. Fig 6.8 shows the Desired trajectory and fig 6.9 shows the actual trajectory. We can notice from fig 6.10 that the trajectory error is minimal and is in the order $10^{-}3$ which is negligible. The error could be due to the improper tuning of the controller or the communication delay between ADAMS and MATLAB

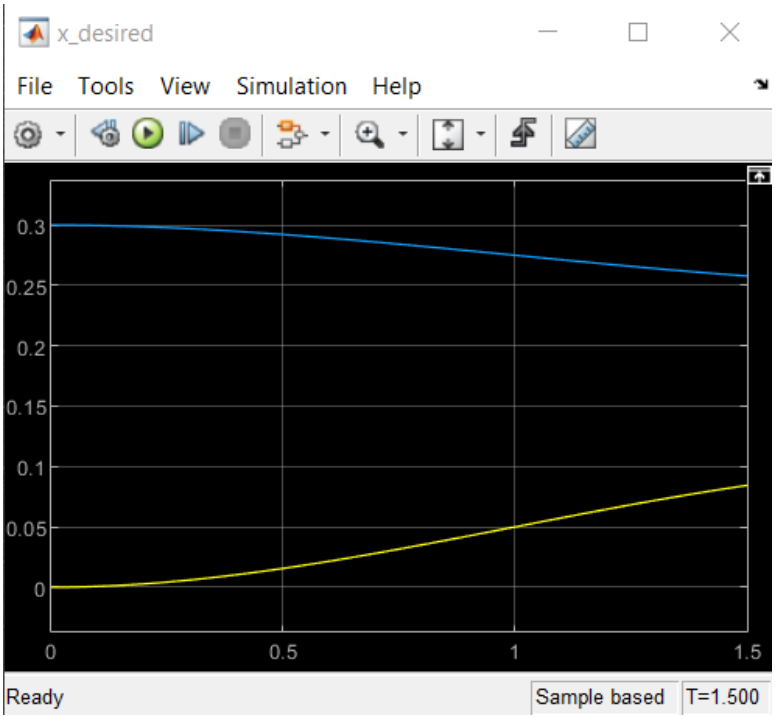

FIGURE 6.7: *Computed Torque Control Trajectory error*

In Figure 6.10 we can notice that the joint angles converge with the desired and actual values

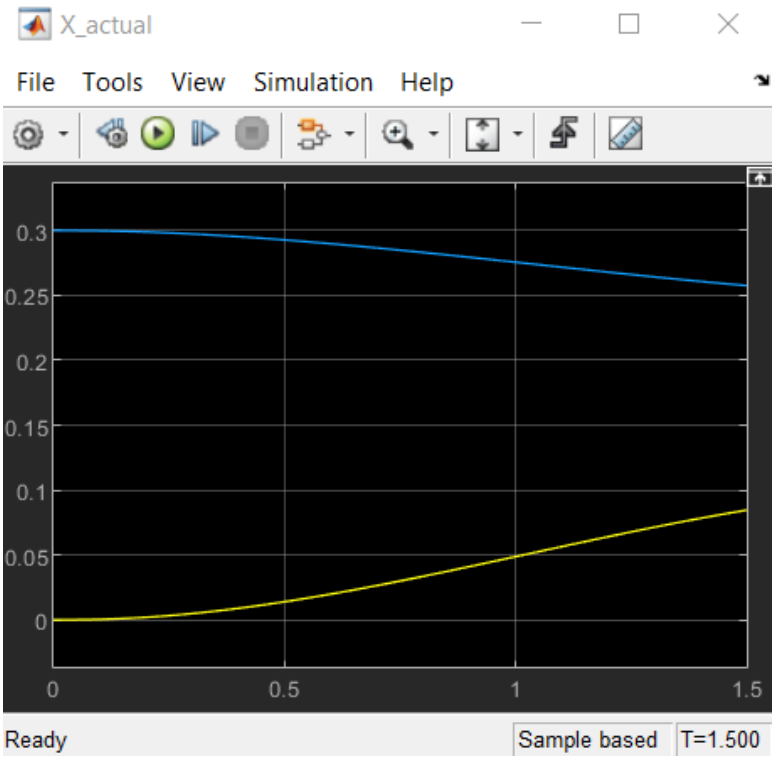

FIGURE 6.8: *Desired Trajectory for Computed Torque Control*

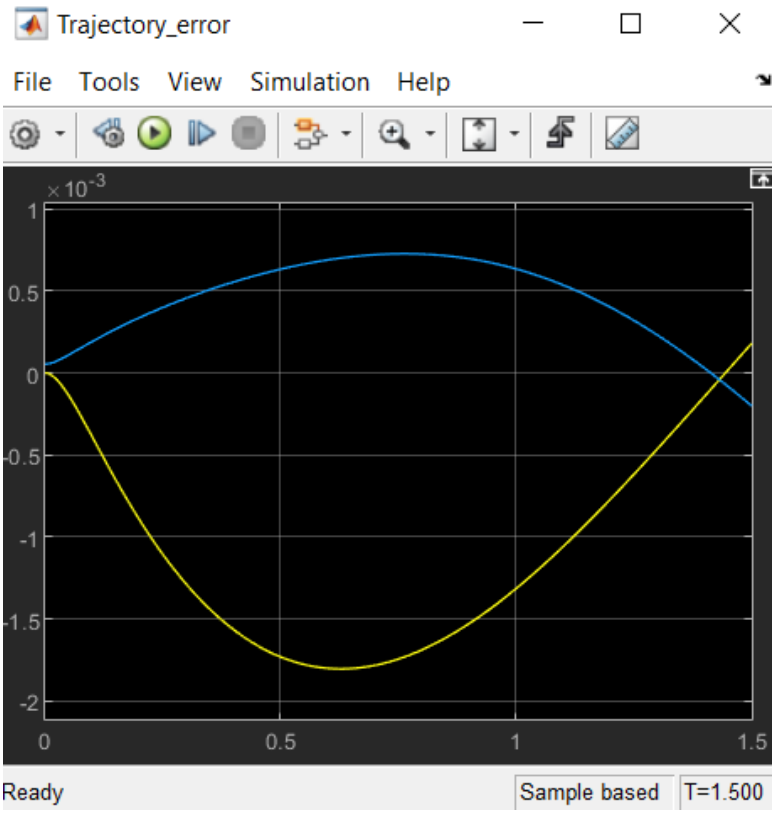

FIGURE 6.9: *Actual Trajectory for Computed Torque Control*

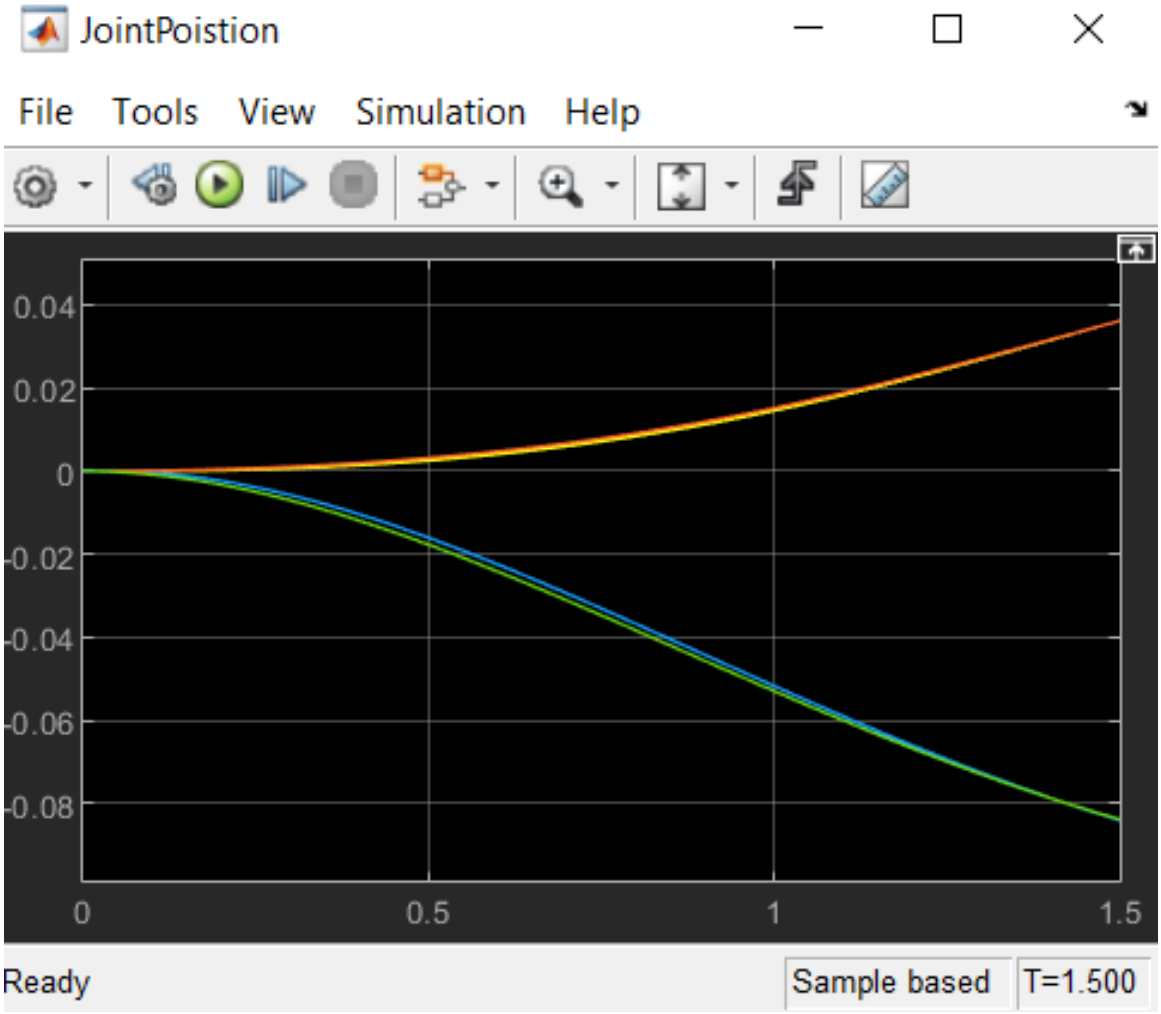

FIGURE 6.10: *Actual and Desired joint positions*

Chapter 7

Conclusion

In conclusion the behaviour of robot was studied,Biglide mechanism's geometric,kinematic and dynamic models are derived and verified.The kinematic and computed control law has been designed and implemented to follow a trajectory. The theoretical models yield close to zero error signal when compared with the ADAMS model.