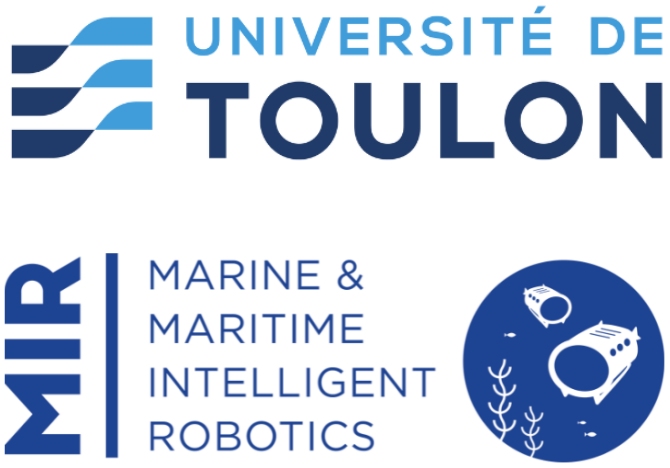

Modeling & Control of a Hexabot leg

By

Mohamed Magdy Atta

ID: 22405169

Geometric, kinematic and dynamical modeling of robotic systems

Submitted to: Prof. Cedric Anthierens

Universite de Toulon

November 7, 2024

Contents

1 Introduction

Hexabots are one of the well-known locomotion mechanisms in Robotics. They use several robotic arms to construct their legs which are connected to a base known as the robot body.

The hexabot Leg is composed of 3 revolute joints; the hip (q1), the knee (q2), and the ankle (q3). The footprint is considered as punctual. This kinematic chain can move the foot tip along an elliptic path relative to the ground frame, so it would be able to generate steps.

The following sections will discuss the kinematic model of a single hexabot leg. In addition, the control of the hexabot leg foot to generate desired motions and velocities will be discussed with the help of MATLAB & SIMULINK and a Simscape Multibody Simulation.

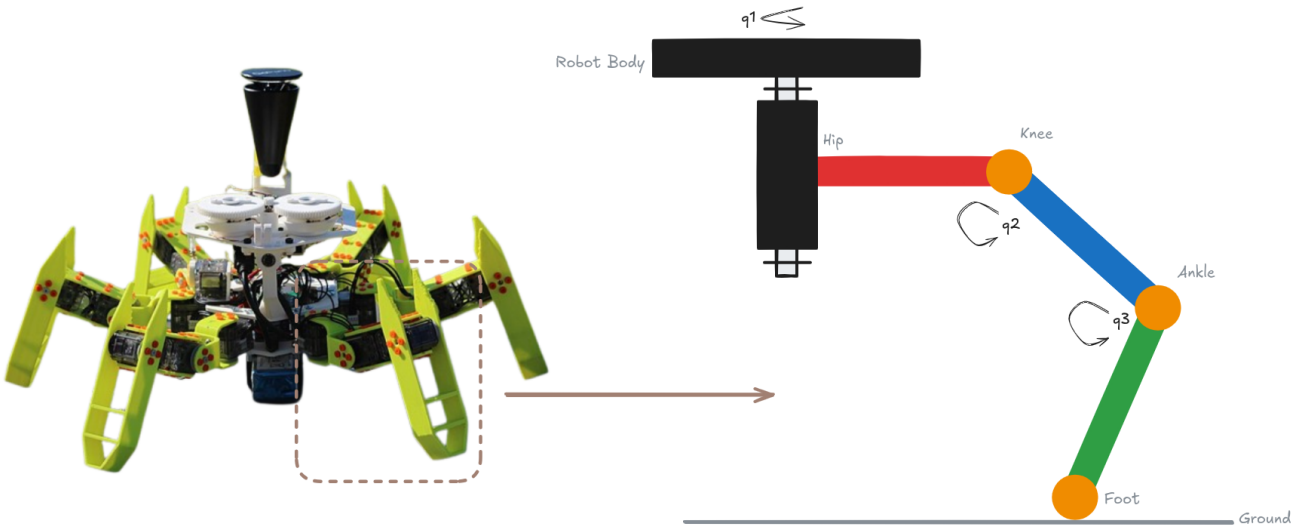

Figure 1: Full Hexabot and one of its legs

2 Foot Degrees of Freedom

To understand how the hexabot leg is moving, it is important to know how many degrees of freedom of the foot can be controlled. The hexabot leg is a spatial mechanism. Its degrees of freedom can be calculated using Grübler's formula for spatial mechanisms which take into consideration the number of links including ground (N), number of joints (J), and number of degrees of freedom (f) provided by each single joint:

$$dof = 6(N - 1 - J) + \sum_{i=1}^{J} f_i$$

The hexabot leg has three rigid links, three revolute joints, and one degree of freedom for each joint.

$$dof = 6(4 - 1 - 3) + 3 = 3$$

The leg configuration results in having three controlled degrees of freedom for the foot, which is considered as the end effector of the leg. The control of the leg foot will be possible along the Cartesian X , Y , and Z axes only.

3 Hexabot leg forward Kinematics

The forward kinematics of the robot is used to map from the joint space variables (Θi) to the position and orientation of the end-effector .

Figure 2: Forward Kinematics Mapping

To calculate the forward kinematics, a set of frames are positioned on the robot geometry and a DH parameters table is calculated according to DH/Khalil formalism:

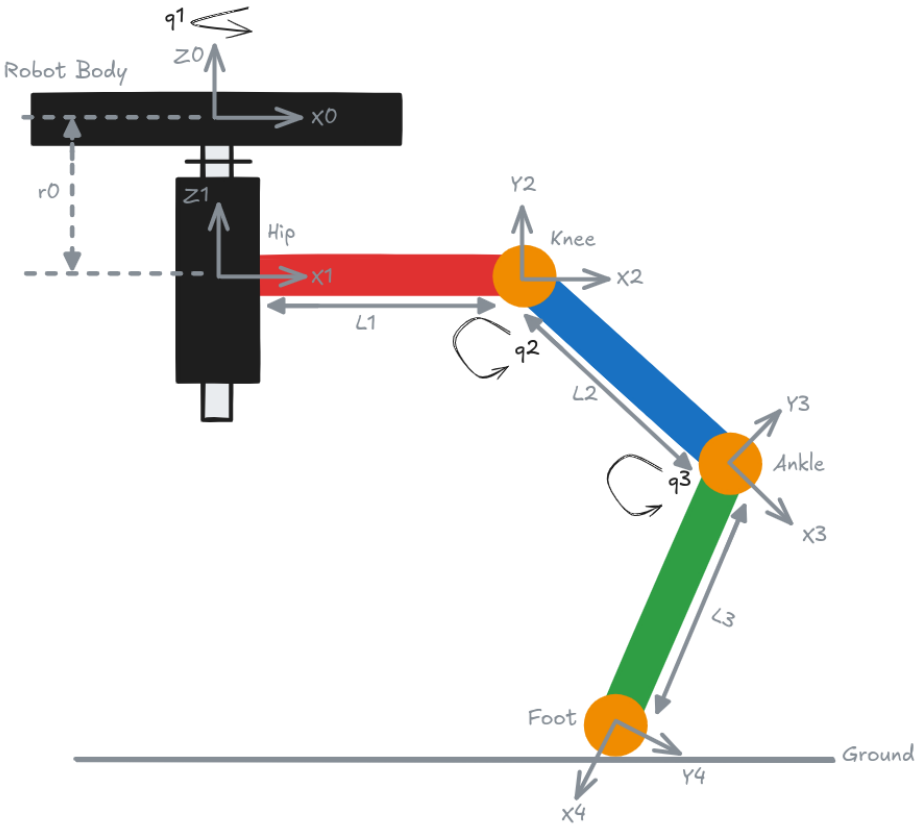

Figure 3: Hexabot leg Coordinate Frames

Frame i	$\boldsymbol{d}$	$\boldsymbol{\alpha}$	$\boldsymbol{r}$	$\boldsymbol{\theta}$
1	0	0	r_0	q_1
2	L_1	$\frac{\pi}{2}$	0	q_2
3	L_2	0	0	q_3
4	L_3	0	0	0

Table 1: Denavit-Hartenberg Parameters Table

Next, homogeneous transformation matrices are calculated based on the D-H parameters table. The approach used starts by applying a translation and rotation about the x-axis then a translation and rotation about z-axis. The goal is to get the final position and orientation of the end effector (foot frame) relative to the base frame (robot body) from (T_4^0).

$$T_i^{i-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & d \\ \sin\theta\cos\alpha & \cos\theta\cos\alpha & -\sin\alpha & -r\sin\alpha \\ \sin\theta\sin\alpha & \cos\theta\sin\alpha & \cos\alpha & r\cos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^0 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & r_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_2^1 = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & L_1 \\ 0 & 0 & -1 & 0 \\ \sin q_2 & \cos q_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} \cos q_3 & -\sin q_3 & 0 & L_2 \\ \sin q_3 & \cos q_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_4^3 = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = T_1^0 * T_2^1 * T_3^2 * T_4^3 = \begin{bmatrix} \cos(q_1)\cos(q_2+q_3) & -\cos(q_1)\sin(q2+q3) & \sin(q_1) & L_1\cos(q_1) + L_2\cos(q_1)\cos(q_2) + L_3\cos(q_1)\cos(q_2+q_3) \\ \sin(q_1)\cos(q_2+q_3) & -\sin(q_1)\cos(q_2+q_3) & -\cos(q_1) & L_1\sin(q_1) + L_2\sin(q_1)\cos(q_2) + L_3\sin(q_1)\cos(q_2+q_3) \\ \sin(q_2+q_3) & \cos(q_2+q_3) & 0 & r_0 + L_2\sin(q_2) + L_3\sin(q_2+q_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4 Hexabot leg Jacobians

The Jacobian matrix J(θ) of the robot is used to map from the joint velocities to the robot end effector velocities.

Figure 4: Jacobian Matrix Mapping

The general equation for the relation between the joint velocities and the end-effector velocities is:

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \\ \dot{\phi_x} \\ \dot{\phi_y} \\ \dot{\phi_z} \end{bmatrix} = J(\theta) \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \vdots \\ \dot{q_n} \end{bmatrix}$$

The hexabot leg has three joint angles and three controlled degrees of freedom, so the equation becomes:

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \end{bmatrix} = J(\theta) \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix}$$

To calculate the jacobian matrix, we need first to get the position of the end effector from the T_4^0 matrix:

$$P_x\text{=}T_4^0\text{[1,4]=}L_1\cos(q_1) + L_2\cos(q_1)\cos(q_2) + L_3\cos(q_1)\cos(q_2+q_3)$$
$$P_y\text{=}T_4^0\text{[2,4]=}L_1\sin(q_1) + L_2\sin(q_1)\cos(q_2) + L_3\sin(q_1)\cos(q_2+q_3)$$
$$P_z\text{=}T_4^0\text{[3,4]=}r_0 + L_2\sin(q_2) + L_3\sin(q_2+q_3)$$

Next, we need to find the time derivatives of the positions obtained. Then we can construct the jacobian matrix by separating the joint velocities from the derivatives in a matrix form:

$$\begin{aligned}\dot{p_x}=&-L1\sin(q_1)\dot{q_1} - L2\cos(q_2)\sin(q_1)\dot{q_1} - L2\cos(q_1)\sin(q_2)\dot{q_2} - L3\cos(q_2+q_3)\sin(q_1)\dot{q_1}\\ &- L3\cos(q_1)\sin(q_2+q_3)\dot{q_2} - L3\cos(q_1)\sin(q_2+q_3)\dot{q_3}\\ \dot{p_y}=&L1\cos(q_1)\dot{q_1} + L2\cos(q_2)\cos(q_1)\dot{q_1} - L2\sin(q_1)\sin(q_2)\dot{q_2} + L3\cos(q_2+q_3)\cos(q_1)\dot{q_1}\\ &- L3\sin(q_1)\sin(q_2+q_3)\dot{q_2} - L3\sin(q_1)\sin(q_2+q_3)\dot{q_3}\end{aligned}$$

$$\dot{p_z}=L2\cos(q_2)\dot{q_2} + L3\cos(q_2+q_3)\dot{q_2} + L3\cos(q_2+q_3)\dot{q_3}$$

$$\begin{bmatrix}\dot{p_x}\\ \dot{p_y}\\ \dot{p_z}\end{bmatrix} = \begin{bmatrix}\frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3}\\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3}\\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3}\end{bmatrix}\begin{bmatrix}\dot{q_1}\\ \dot{q_2}\\ \dot{q_3}\end{bmatrix}$$

$$J(\theta) = \begin{bmatrix}\frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3}\\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3}\\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3}\end{bmatrix}$$

$$J(\theta) = \begin{bmatrix}-L_1\sin(q_1) - L_2\cos(q_2)\sin(q_1) - L_3\cos(q_2+q_3)\sin(q_1) & -L_2\cos(q_1)\sin(q_2) - L_3\cos(q_1)\sin(q_2+q_3) & -L_3\cos(q_1)\sin(q_2+q_3)\\ L_1\cos(q_1) + L_2\cos(q_2)\cos(q_1) + L_3\cos(q_2+q_3)\cos(q_1) & -L_2\sin(q_1)\sin(q_2) - L_3\sin(q_1)\sin(q_2+q_3) & -L_3\sin(q_1)\sin(q_2+q_3)\\ 0 & L_2\cos(q_2) + L_3\cos(q_2+q_3) & L_3\cos(q_2+q_3)\end{bmatrix}$$

The jacobian matrix can be used to find the singularity configurations of the hexabot leg. To find the joint variables that cause the singularity configurations, the roots of the following equation are calculated:

$$Determinant(J(\theta)) = 0\ .$$

The hexabot leg has a singularity and a loss of one degree of freedom when the joint variables $(q_1, q_2, q_3) = 0$ or π .

5 Foot Motion Control

Controlling and planning the motion of manipulators can be done by different techniques. Motion control can be done in either joint or task space. The robot jacobians can be used to control the robot end-effector speed and position in the cartesian task space.

In my real-world applications, it is required for the robot end-effector to move in a straight line in the cartesian space. Resolved-rate motion control is a Jacobian-based motion control method that can be used to generate a strictly forward foot step without generating any other motion to the whole robot body. The method works without having to compute the inverse kinematics at each time step. Instead, the inverse of the Jacobian matrix alone is recomputed at each time step to account for the updated joint variables as shown below:

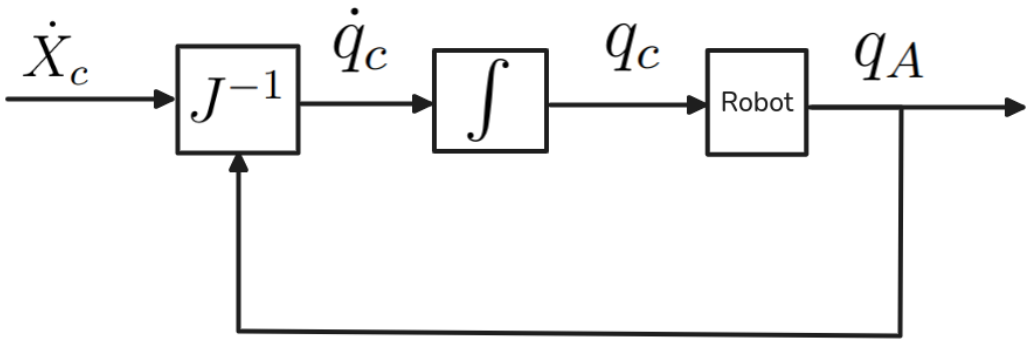

Figure 5: Resolved-Rate Motion Control Algorithm

Simscape and Multibody ,which are 2 toolboxes of Matlab / Simulink, are used to model the hexabot leg as well as control its motion.

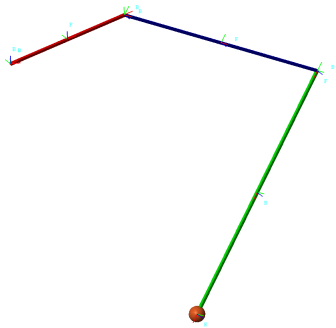

Figure 6: Hexabot leg model created using Simscape Multibody

Provided the main robot axis (longitudinal axis to move forward) is lining with X-ground, we can control the leg to generate a strictly forward step in the main robot axis (X-axis) by inputting velocity values along x-axis only to the inverse jacobian function, while keeping the velocity values along y and z-axes as zeros. For testing the method, a sine function with an amplitude of 70 is provided as the x-axis velocities during the simulation. The outputs of the position and speed of the foot indicate clearly that the motion is executed along the x-axis only:

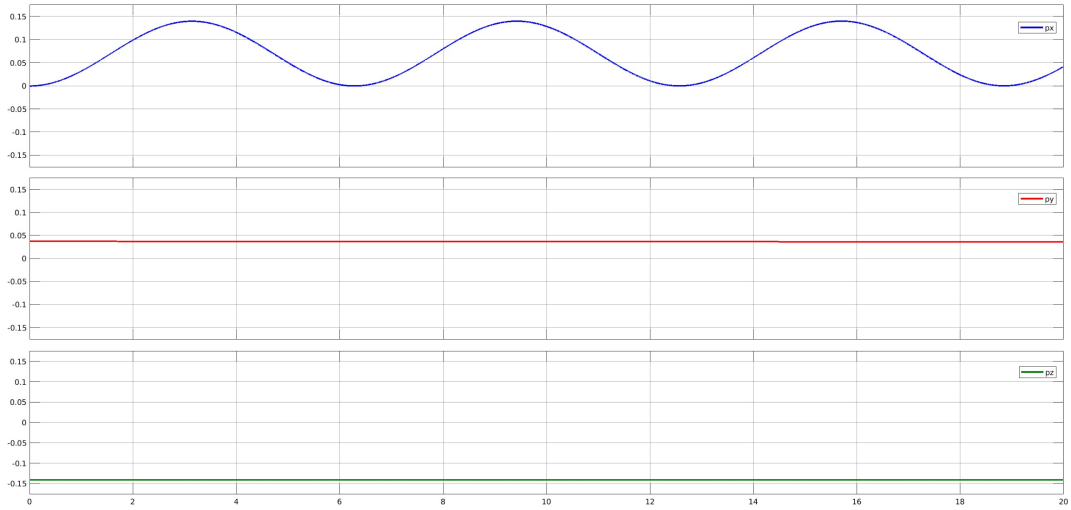

Figure 7: Hexabot leg foot positions during simulation

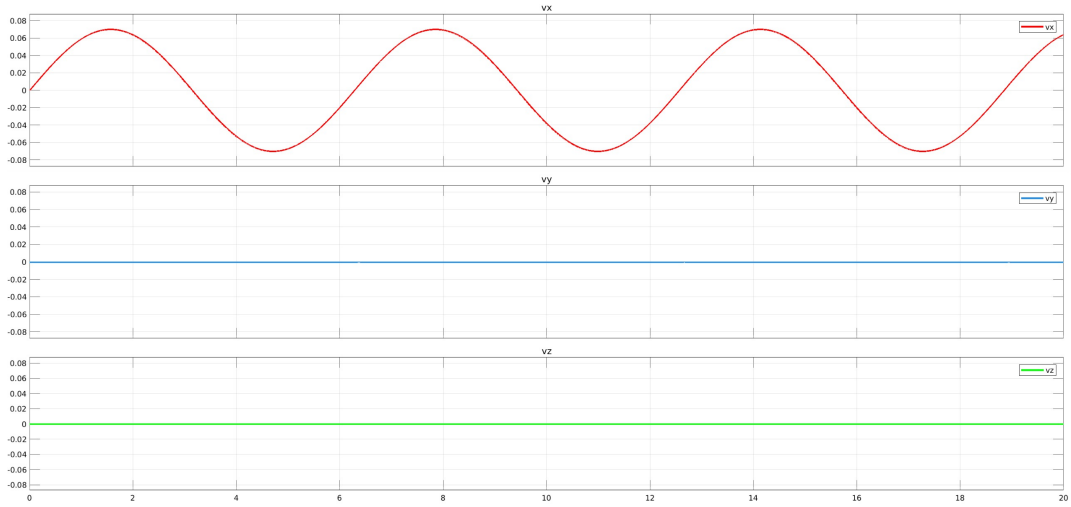

Figure 8: Hexabot leg foot velocities during simulation

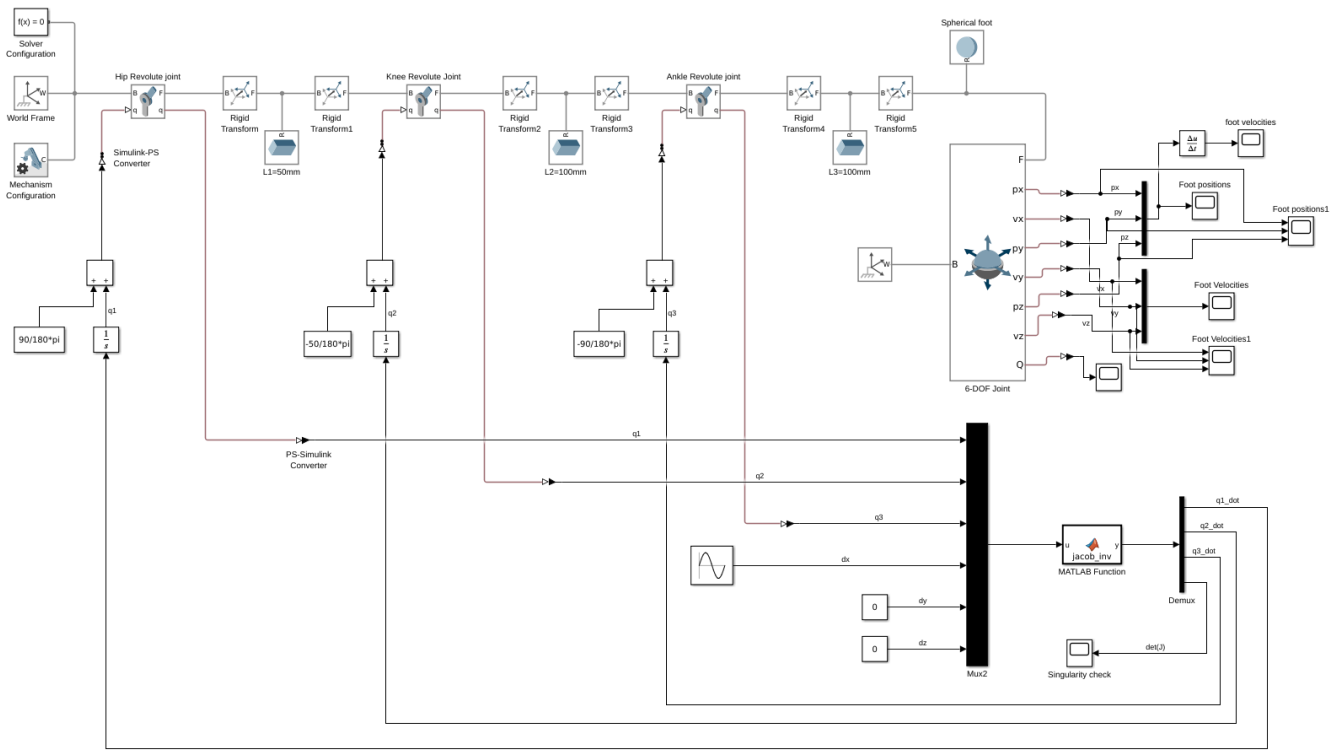

Figure 9: Simulink Model of the Hexabot Leg

Another important aspect that can be studied using the jacobian matrix during simulation is the robot singularities. To detect which configurations are near the singularity, the determinant of the jacobian matrix is calculated at each time step. The robot is having a singularity configuration when the determinant is zero. The portions of the simulation with determinant values near the zero are considered important as it could be studied for improving the path and motion planning techniques used in order to be avoid singularities in the future.

During simulating the model using a sine function with an amplitude of 70 as a velocity input along x-axis, the robot didn't reach a singularity position during its motion. However, when the amplitude of the sine wave is increased to be 150, a number of singularities happened. Increasing the amplitude of the sine wave input makes the robot explore a wider range of its workspace, which increases the likelihood of encountering singularities as shown below:

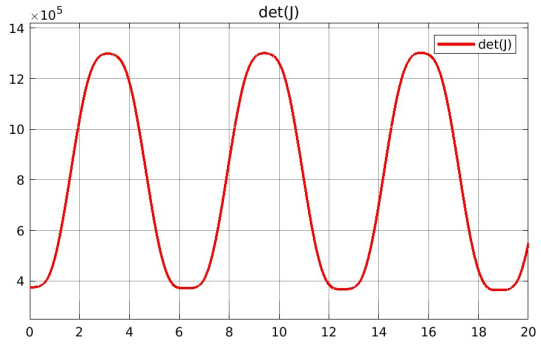

Figure 10: Singularity Check with Sine Amplitude=70

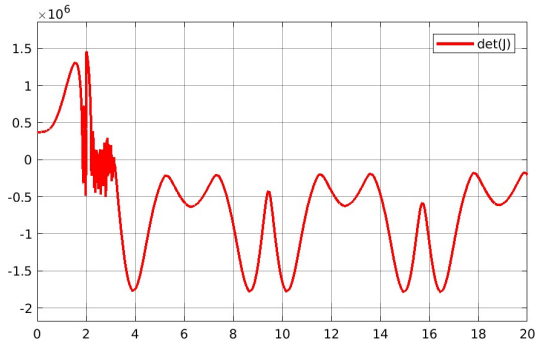

Figure 11: Singularity Check with Sine Amplitude=150

Motion control with the help of Jacobians is an important topic to be considered during working with manipulators. Another method that can be used to actuate the foot in the x-axis direction only using Jacobians is to solve the inverse kinematics problem numerically by iteratively guessing the solution until it becomes within a certain threshold. This is done by formulating the problem as an optimization one. The optimization method aims at minimizing the difference between the robot's end effector current and desired cartesian position. Thus, the cost function will be the euclidean distance between these two position. Gradient descent is used as the optimization algorithm. It iteratively adjusts the joint angles of the robotic arm to minimize the difference between the current end-effector position and the desired end-effector position. In each iteration, the algorithm calculates the gradient of the cost function with respect to the joint angles, and then updates the joint angles by taking a step in the direction of the negative gradient.Once the gradient is calculated, the joint angles can be updated using the following equation: $\theta_i = \theta_i - \alpha \frac{\partial C}{\partial \theta_i}$ where θ is the i_{th} joint angle, α is the learning rate or step size, and $\frac{\partial C}{\partial \theta_i}$ is the partial derivative of the cost function with respect to the i_{th} joint angle:

$$
\begin{aligned}
&\theta \leftarrow \theta initial \\
&loop: \\
&\qquad \frac{\partial f}{\partial \theta} \leftarrow Jacobian \\
&\qquad \frac{\partial C}{\partial \theta} \leftarrow \frac{\partial f^T}{\partial \theta}(f(\theta) - X) \\
&\qquad \theta \leftarrow \theta - \alpha \frac{\partial C}{\partial \theta} \\
&\qquad if\ C(\theta) < threshold: \\
&\qquad\qquad return\ \theta
\end{aligned}
$$

Figure 12: Jacobian-Based Inverse Kinematics Algorithm using gradient descent