



*Kinematic Chains - Modeling of Mechanical Systems*

Laboratory Report:  
Modeling of a Hexabot Leg in MATLAB

Maximilian J. Vieweg

Andreas Sitorus

October 2022

## Table of Contents

<b>1. Introduction.....</b>	<b>2</b>
<b>2. Theoretical Modelling of the Hexabot Leg.....</b>	<b>2</b>
<b>3. Simulation of the Hexabot leg.....</b>	<b>6</b>
<b>4. Conclusion .....</b>	<b>7</b>
<b>5. Appendix .....</b>	<b>8</b>

# 1. Introduction

There are numerous approaches to the movement of mobile robotic systems. The Hexabot design, using six legs to move forward, is modeled during this laboratory exercise. First it is described theoretically, after which a simulation of one step of the leg is carried out.

The robotic leg consists of three revolute joints; the footprint is assumed to be punctual. Using the kinematic chain, the foot tip may be moved for the robot to generate steps. A schematic of the setup can be found below.

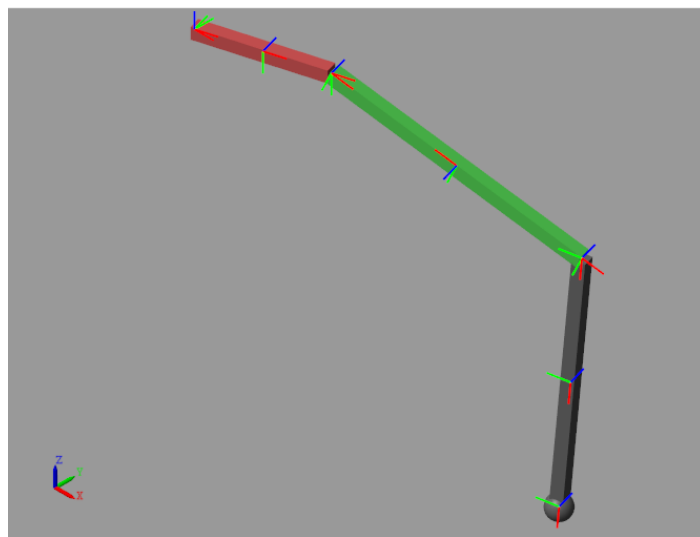


Figure 1: The kinematic model, connecting three rods with revolute joints.

The kinematic model of the system is used to describe the motion of the end-effector. Using inverse kinematics, a simple step is simulated to generate a simple framework for robotic locomotion.

## 2. Theoretical Modelling of the Hexabot Leg

1. According to the mission dedicated to this limb, what are the desired controlled degree of freedom of the foot?

**Answers:**

In total, there are three desired controlled degrees of freedom for the Hexabot leg,  $q_1$  (hip),  $q_2$  (knee), and  $q_3$  (ankle); the minimum number of actuators required for moving in three-dimensional space is three.

2. The robot body may be considered as still (same orientation as the Ground frame), so place all the intermediate frames until the foot. Fill in the table as asked in the first exercise. Write all the homogeneous transform matrices from a frame to the neighboring frame and all those from each frame to the body frame (useful for the next question).

**Answers:**

The Denavit-Hartenberg parameters and Khalil formalism Table is used to calculate the homogenous transform matrices for each joint. The table is constructed as follows:

*Table 1 DH Parameters and Khalil Formalism Table*

Type of Joint	Distance along X	Angle around X	Distance along Z	Angle around Z
$\sigma$	$d$	$\alpha$	$r$	$\theta$
Revolute	0	0	0	$\theta_1$
Revolute	$l_1$	$-90^\circ$	0	$\theta_2$
Revolute	$l_2$	0	0	$\theta_3$
End-Effector	$l_3$	0	0	0

with the coordinate frames chosen below:

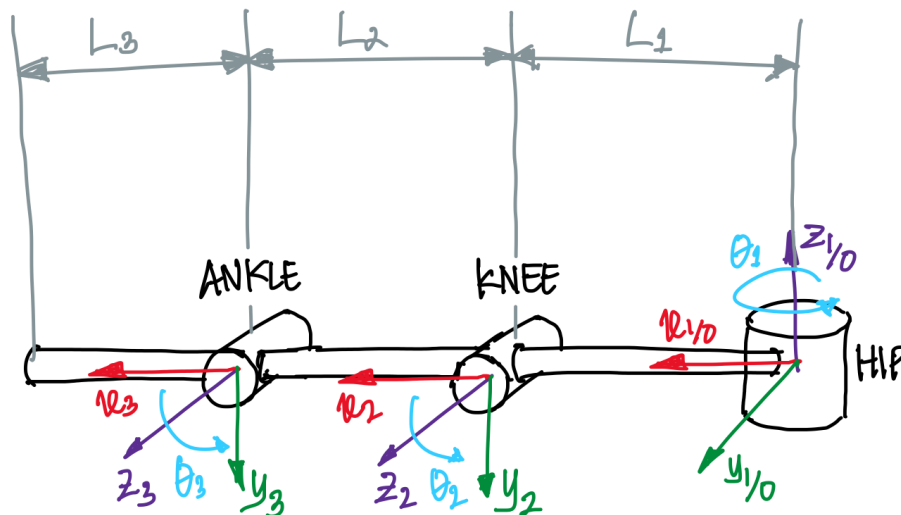


Figure 2: The chosen coordinate frames for the Hexabot leg can be seen here. The x-Axis points towards the next joint, while the z-axis in the direction of the angular velocity vector.

From the table, the homogenous transform matrices for each joint follow as such:

$${}^0_1T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1_2T = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2_3T = \begin{bmatrix} c_3 & -s_3 & 0 & l_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^3_4T = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrices above describe the position and orientation of each joint relative to the previous frame. The motion of the end-effector relative to the base frame can be found by continuously performing the transformations from the body of the robot to the foot. The explicit representation of the joint positions is shown below:

$${}^0_2T = \begin{bmatrix} c_1 c_2 & -c_1 s_2 & s_1 & l_1 c_1 \\ s_1 c_2 & -s_1 s_2 & -c_1 & l_1 s_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_3T = \begin{bmatrix} c_1 c_2 c_3 - c_1 s_2 s_3 & -c_1 c_2 s_3 - c_1 s_2 c_3 & s_1 & l_1 c_1 + l_2 c_1 c_2 \\ s_1 c_2 c_3 - s_1 s_2 s_3 & -s_1 c_2 s_3 - s_1 s_2 c_3 & c_1 & l_1 s_1 + l_2 s_1 c_2 \\ s_2 c_3 + c_2 s_3 & -s_2 s_3 + c_2 c_3 & 0 & l_2 s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_4T = \begin{bmatrix} c_1 c_2 c_3 & -c_1 s_2 c_3 & s_1 & l_1 c_1 + l_2 c_1 c_2 + l_3 c_1 c_2 c_3 \\ s_1 c_2 c_3 & -s_1 s_2 c_3 & -c_1 & l_1 s_1 + l_2 s_1 c_2 + l_3 s_1 c_2 c_3 \\ s_2 c_3 & c_2 c_3 & 0 & -l_2 s_2 - l_3 s_2 c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The last transformation matrix directly shows the position of the end-effector relative to the body of the robot, which we can use in calculating the Jacobian matrix below.

3. By using the direct method, calculate the Jacobian matrix which gives the foot motion relative to the ground frame (or the body frame if it's considered still).

**Answers:**

To calculate the Jacobian Matrix, the position of the end-effector is described using standard kinematics equations: The relative position of the lower leg is derived and then rotated using the rotation matrix.

$$\mathbf{r} = \begin{pmatrix} \cos(q_1) & -\sin(q_1) & 0 \\ \sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_1 + l_3 \cos(q_2 + q_3) + l_2 \cos(q_2) \\ 0 \\ -l_3 \sin(q_2 + q_3) - l_2 \sin(q_2) \end{pmatrix}$$

$$\mathbf{r} = \begin{pmatrix} \cos(q_1) (l_1 + l_3 \cos(q_2 + q_3) + l_2 \cos(q_2)) \\ \sin(q_1) (l_1 + l_3 \cos(q_2 + q_3) + l_2 \cos(q_2)) \\ -l_3 \sin(q_2 + q_3) - l_2 \sin(q_2) \end{pmatrix}$$

As can easily be verified, the location vector matches the transformation from the body to the ground frame.

The Jacobian of the system can be found by two different approaches. During this laboratory the following formula for directly calculation was used:

$$\mathbf{J}(\mathbf{r}) = \begin{bmatrix} \frac{\partial r_1}{\partial q_1} & \frac{\partial r_1}{\partial q_2} & \frac{\partial r_1}{\partial q_3} \\ \frac{\partial r_2}{\partial q_1} & \frac{\partial r_2}{\partial q_2} & \frac{\partial r_2}{\partial q_3} \\ \frac{\partial r_3}{\partial q_1} & \frac{\partial r_3}{\partial q_2} & \frac{\partial r_3}{\partial q_3} \end{bmatrix}$$

$$\mathbf{J}(\mathbf{r}) = \begin{pmatrix} -\sin(q_1) (l_1 + l_3 \cos(q_2 + q_3) + l_2 \cos(q_2)) & -\cos(q_1) (l_3 \sin(q_2 + q_3) + l_2 \sin(q_2)) & -l_3 \sin(q_2 + q_3) \cos(q_1) \\ \cos(q_1) (l_1 + l_3 \cos(q_2 + q_3) + l_2 \cos(q_2)) & -\sin(q_1) (l_3 \sin(q_2 + q_3) + l_2 \sin(q_2)) & -l_3 \sin(q_2 + q_3) \sin(q_1) \\ 0 & -l_3 \cos(q_2 + q_3) - l_2 \cos(q_2) & -l_3 \cos(q_2 + q_3) \end{pmatrix}$$

The resulting matrix can then be used to compute the velocity of the foot, given a change in the joint rotation.

A second method for calculation can be found by doing a time-derivation of each element of the translational part in the final homogenous transformation matrix ( ${}^0_4T$ ), described by  $P_x$ ,  $P_y$ ,  $P_z$ . The process will give us the (3x1) vector containing velocities of the end-effector relative to the base frame. By decomposing the vector into its angular speed components, the Jacobian matrix is obtained.

4. Provided the main robot axis (longitudinal axis to move forwards) is lining with the ground, can we control the leg to generate a strictly forward step (without any other motion)? What would it imply about the robot locomotion?

**Answers:**

Since the derived Jacobian does not have any singularities, the end-effector may be moved to any desired point in the cartesian workspace of the Hexabot leg. During the multi-body simulation, the leg could be moved in a circular motion in the ground plane to generate a forward step. The coordinated action of the six legs can translationally move the main body forward, thus not generating any other rotational motion during the stepping process.

### 3. Simulation of the Hexabot leg

First the mechanical system is modeled in Simulink using the multibody toolkit. Revolute joints connect the three-dimensional robotic limbs. Since each body exists in its own coordinate system, they must each be transformed to the position and orientation of the previous joint. Then each frame is moved towards the following joint using the transformation matrices described in Section 2. The current state of the robotic leg, described by  $(q_1, q_2, q_3)$  is used as an input by the inverse Jacobian function, which also considers the desired  $(x, y, z)$  cartesian movement of the foot. The result is then used for moving the joints in the desired position. Since the return value of the Inverse Jacobian describes an angular velocity of the joint, it must first be integrated, after which it is added to the initial state of the Hexabot leg.

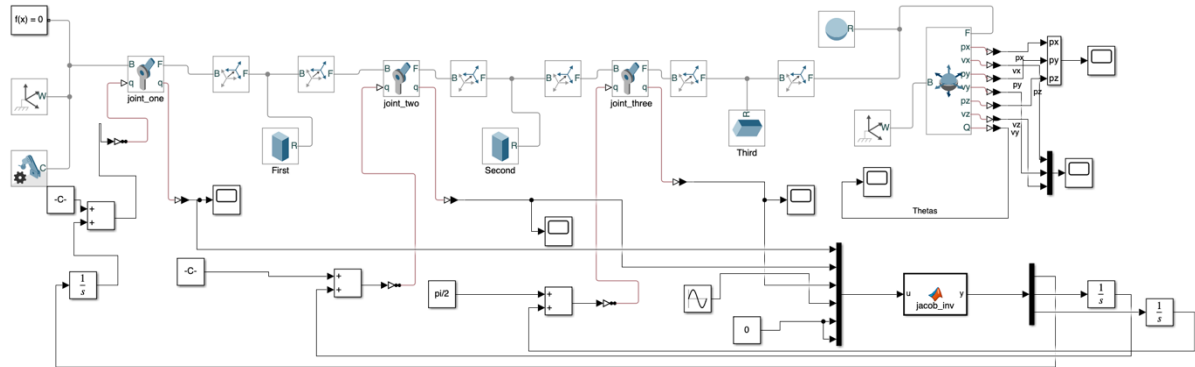


Figure 3: Simulink model connecting the kinematic model of the robot with the control of the joints through a custom designed MATLAB control function. The positions of each joint are fed into the function, as well as the desired next trajectory.

The Simulink model used to simulate the behavior of the robotic leg can be seen in Fig. 3. In the upper right corner, the change in the position during the simulation can be seen. During the experiment the leg moves in a sinusoidal pattern along the x-axis, while not moving in either the y or z-axis.

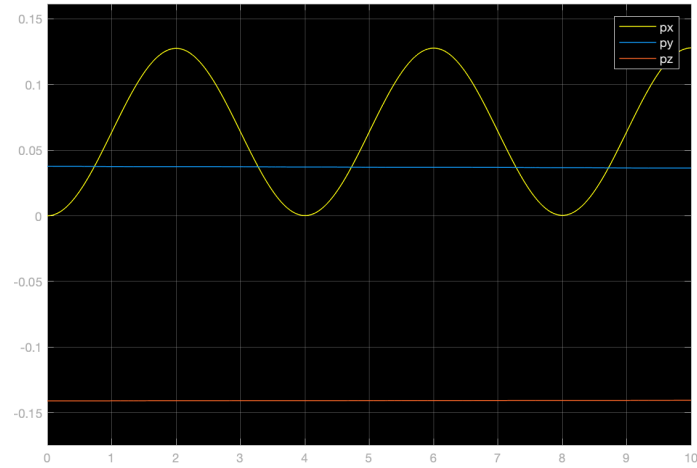


Figure 4: The  $(x, y, z)$  position of the end-effector of the robot are shown. As configured, the effector moves along the  $x$ -direction in a sine motion, while it stays in the  $yz$ -plane.

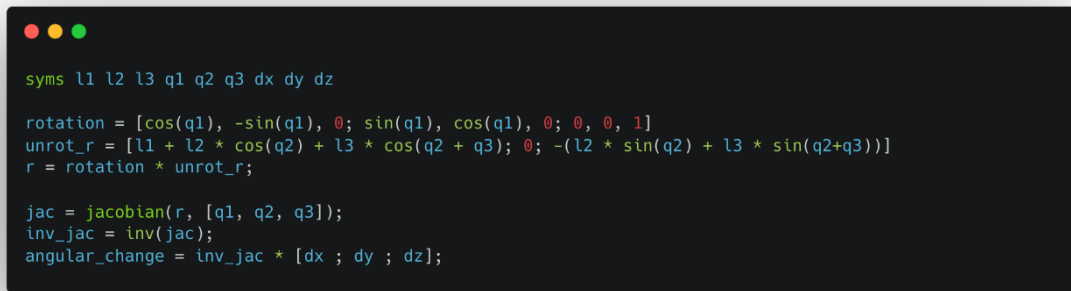
Fig. 4 demonstrates this behavior clearly. The desired movement along the  $x$ -direction does not influence its position in the other directions. This is due to the joints working in unison according to the inverse kinematic model.

## 4. Conclusion

First, we confirmed that it was theoretically possible to generate a forward step, given that the Jacobian matrix of the system is invertible. This analysis could be confirmed during the experiments, when it was possible to visualize and quantify the movement of the end-effector as well as the joint angular velocities.



## 5. Appendix

A screenshot of a MATLAB code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in a light green monospace font. It defines symbolic variables for link lengths (l1, l2, l3), joint angles (q1, q2, q3), and Cartesian coordinates (dx, dy, dz). It then defines a rotation matrix, an unrotated position vector, and the final position vector r. Finally, it calculates the Jacobian matrix, its inverse, and the required angular changes for a given displacement.

```
syms l1 l2 l3 q1 q2 q3 dx dy dz

rotation = [cos(q1), -sin(q1), 0; sin(q1), cos(q1), 0; 0, 0, 1]
unrot_r = [l1 + l2 * cos(q2) + l3 * cos(q2 + q3); 0; -(l2 * sin(q2) + l3 * sin(q2+q3))]
r = rotation * unrot_r;

jac = jacobian(r, [q1, q2, q3]);
inv_jac = inv(jac);
angular_change = inv_jac * [dx ; dy ; dz];
```

Figure 5: The kinematic model described in Section 2 has been implemented in MATLAB code. Using the inverse Jacobian matrix, the movements in the (x, y, z) direction can be used to determine the necessary angular changes for the joints, to generate a forward step.