

# Design and Implementation of Decentralised Control Strategy for 8-DoF Quadruped Robot

Rushikesh Deshmukh  
Robotics Engineering  
Worcester Polytechnic Institute  
Worcester, MA, USA  
rdeshmukh@wpi.edu

Krishna Madhurkar  
Robotics Engineering  
Worcester Polytechnic Institute  
Worcester, MA, USA  
ksmadhurkar@wpi.edu

Kunal Nandanwar  
Robotics Engineering  
Worcester Polytechnic Institute  
Worcester, MA, USA  
kgnandanwar@wpi.edu

**Abstract**—Quadruped robots have been of interest to the robotics community as they have high maneuverability and they can be designed to be very robust to the environment they are placed in. The use of quadrupeds as service robots is being realized by the community. Understanding the dynamics of a quadruped robot is a complex task as there are many factors that need to be taken into consideration while designing the robot. This project intends to develop an eight degree of freedom (DOF) sprawling-type quadruped robot. Control of high DOF robots can be highly challenging and computationally heavy. This project intends to work on bringing down the complexity of the robot control by decoupling the dynamics of the robot into four separate dynamic systems, i.e one for each leg, allowing us to achieve a decentralized control of the robot. Gazebo simulation of the robot is achieved with a few gait implementations using a decentralized control strategy.

**Index Terms**—Quadruped, Spot, Decentralized Control, Feedback Linearization Control, Robust Control, Crawling bot

## I. INTRODUCTION

Quadrupeds are advantageous for locomotion for two major reasons. Firstly, for rapid travel over various terrains and obstacles and secondly, for discrete foothold. Even though they have multiple advantages they are still used in meager applications. The main reason for this is due to their poor energy efficiency and complex design due to multiple closed-loop kinematic chains. The problem becomes much more complex with an increase in the number of degrees of freedom (DOF) of each leg [1] [2].

A lot of research developments have been made over the past years towards the study of Quadruped kinematics, dynamics, and control. Few researchers have used Denavit-Hartenberg (DH) parameters [3] to derive the kinematics of the system. Some have used the Power of Exponentials method for the same. Studies have also been done in developing the dynamics of the robot with various assumptions for various terrains and ground-foot impact. Some research focuses on developing models based on single leg dynamics [4].

A more recent study used a decentralized control system where the quadruped is considered as coupled bipeds and is controlled individually. This decentralized control strategy was employed to reduce the computational complexity of the robot

All authors have contributed equally to this paper, the names are written in the order of the Last Name of the authors

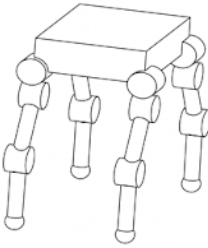
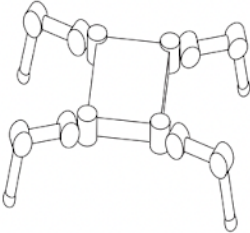
Topology	Characteristics
	(a) Mammal-type robot (b) The joint torque is small when the leg is bent, and there is almost no joint torque when standing upright. The footprint is small, and it is easy to pass through narrow spaces
	(a) Sprawling-type robot (b) The front and rear swings are large. The center of gravity can be adjusted to a low position. The body can be used as a support point while standing

Fig. 1. Types of Quadrupeds [8]

[5]. Different methods used for the formulation of dynamics of quadrupeds are the recursive Newton-Euler method [6] and Lagrange-Euler's method [7] [5]. For this project, the kinematic of a single leg of the quadruped is calculated using DH parameters and the Power of Exponentials (PoE) method. For dynamics, we plan to formulate the equations based on the recursive Newton-Euler method.

## II. METHODOLOGY

### A. Quadruped

Quadrupedal robots are a type of legged robots with four legs. Each leg is designed with certain DOFs, morphology, link lengths, and material. Based on the leg morphology, quadrupeds can be classified into two types- sprawling-type and articulate mammal-type quadrupeds. Articulate mammal-type quadrupeds like Spot Mini by Boston Dynamics are gaining popularity. These can perform a huge range of motions like walking, trotting, running, backflips. These quadrupeds are very robust to external conditions and can perform high-

speed motions. They can also adjust their height to avoid obstacles. Here, each leg can connect to the ground at various angles depending on the gait [8]. This type of quadruped is thus difficult to model. In this project we work towards our goal of achieving a decentralized control strategy. To reduce project complexity we choose a sprawling type quadruped. Sprawling type quadruped is relatively easy to model. They have a low center of gravity, complete symmetry, and simpler gait trajectories.

### B. Decentralized Control

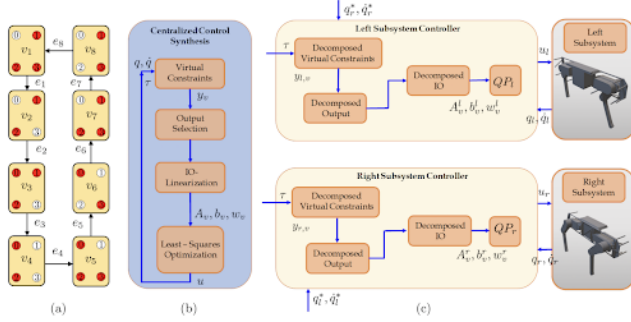


Fig. 2. Centralized control vs Decentralized Control Strategy [5]

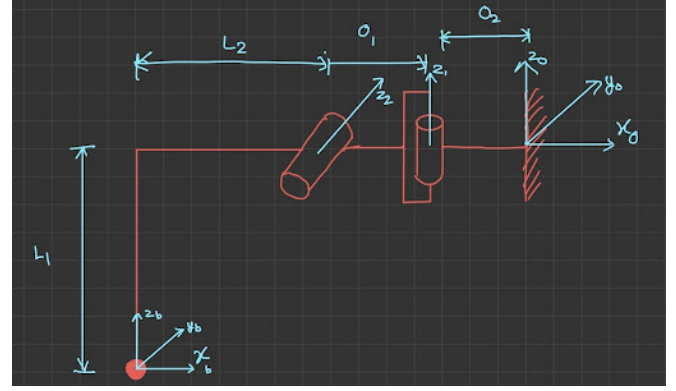
Usually, a robot system is modeled as a whole with kinematics and dynamics. To control such a model, many mathematical operations are performed including state-matrix multiplications and inversions. These operations are often computationally heavy [5]. So, recently a few researchers have started to use a decentralized control strategy. In this control strategy, the model of the robot is broken down into smaller subsystems. The dynamics and kinematics of these sub-systems are then calculated and each of these sub-system is individually controlled as can be seen in Fig[2]. There is one central controller that handles high-level control like gaiting, motion planning, etc. Each subsystem can be placed in a separate controller for increased efficiency. Thus, we selected our control to be a decentralized control strategy for control of our sprawling-type quadruped.

### C. Dynamics

Working towards the goal of decentralized control, the next step was to formulate the robot dynamics. To find the dynamics of the model we are making the following assumptions:

- 1) There is no slip at the point of contact.
- 2) The terrain is always flat.
- 3) There is no deformation of the body at the point of impact.
- 4) The foot will touch the ground only as a point contact.
- 5) There is no rebound on collision.

We have two scenarios on when the leg is in swing motion and the other, when the leg has collided with the ground. Therefore, we have two equations. We formulate the two equations with the above assumptions in consideration for the system using the Recursive Newton Euler(RNE) method as



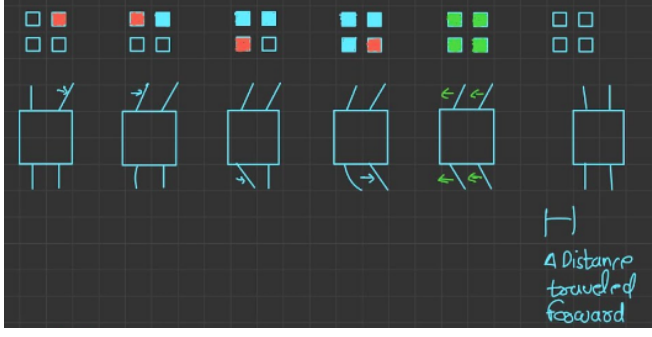


Fig. 5. Creeping gait. Here the colors indicate the following, Red : forward swing trajectory, Blue : Standing, Green : When the legs push the body forward

seen in figure 6. The model is passed with the torque values using a effort controller of the robot to control the robot.

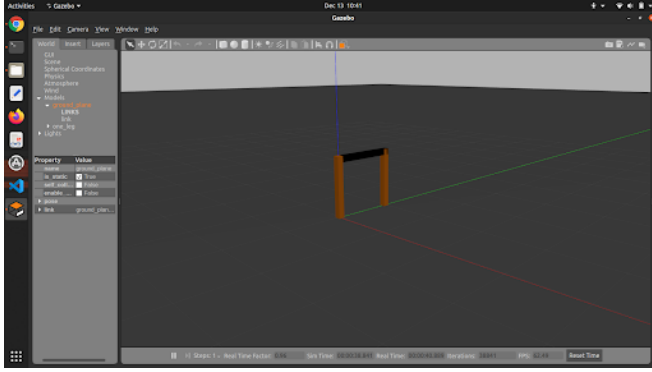


Fig. 6. One Leg Gazebo Model

The next phase of our project was to develop Bot robot model in Gazebo to simulate our control results. This model being a Bot has to have horizontal rotation in the z axis of the robot body for its shoulder joint and a revolute joint along the x axis rotation. Figure 7 shows this model. As a stretched

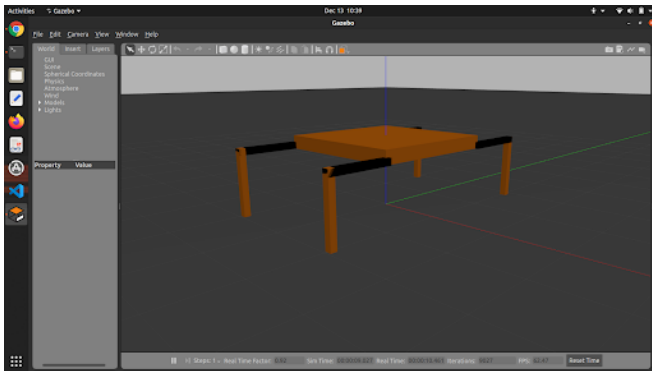


Fig. 7. Creepy Bot

goal, a model of spot done next which was similar to the Bot robot in terms of joint two but now had a shoulder joint along the x axis of the robot, taking away the ability of the Bot robot

to always stay stable and parallel to the ground plane as can be seen in Figure 8.

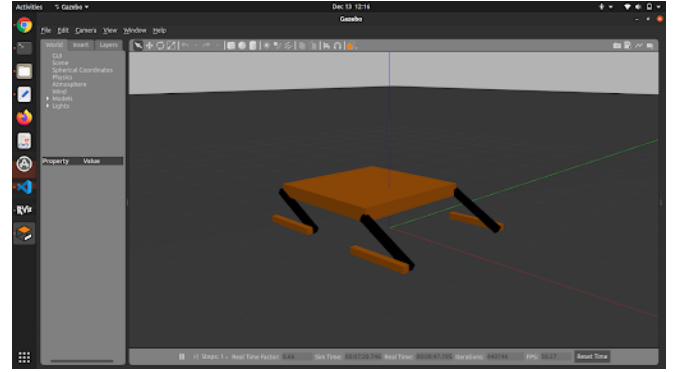


Fig. 8. Creepy SPOT

### E. Matlab Implementation for Controls

The control scripts for feedback linear control and robust control were developed in MATLAB for each leg. The dynamic calculations for the Bot robot individual leg is as follows:

$$\begin{aligned}
 v_x &= (l1 * \sin(q1) * q1dot) + (r2 * ((- \sin(q2) * \sin(q1) * q1dot) + (\cos(q2) * \cos(q1) * q2dot))) \\
 v_y &= ((l1 * \cos(q1) * q1dot) + (r2 * ((\sin(q2) * \cos(q1) * q1dot) + (\cos(q2) * \sin(q1) * q2dot)))) \\
 v_z &= (-r2 * \sin(q2) * q2dot) \\
 v_2 &= (v_x^2 + v_y^2 + v_z^2) \\
 K1 &= 0.5 * m1 * r1 * r1 * q1dot^2 + 0.5 * I1 * q1dot^2 \\
 K2 &= 0.5 * m2 * (v_2) + 0.5 * I2 * (q1dot^2 + q2dot^2) \\
 K &= K1 + K2 \\
 P1 &= 0 \\
 P2 &= m2 * g * (l2 - l1 * \cos(q2)) \\
 P &= P1 + P2 \\
 L &= simplify(K - P) \\
 q &= [q1; q2] \\
 qd &= [q1dot; q2dot] \\
 qdd &= [q1ddot; q2ddot] \\
 U &= [u1; u2]
 \end{aligned}$$

where:

$v_x$  is the velocity component of joint 2 in x direction  
 $v_y$  is the velocity component of joint 2 in y direction  
 $v_z$  is the velocity component of joint 3 in z direction  
 $K1$  is the kinetic energy contributed by the joint 1  
 $K2$  is the kinetic energy contributed by the joint 2  
 $P1$  is the potential energy contributed by the joint 1  
 $P2$  is the potential energy contributed by the joint 2  
 $L$  is the Lagrangian

Final  $U$  gives the dynamics of the system in terms of  $u1$  and  $u2$ . After getting  $u1$  and  $u2$  we then used these to implement feedback and robust control.

1) *Feedback Linearization Control*: Feedback Linearization Control Law:

$$v = [\text{desired}q1\ddot{\phantom{x}} \text{desired}q2\ddot{\phantom{x}}]' - K * e;$$

Here,  $e$  is the error dynamics representing the difference in current states and desired states  $K$  is control tuning parameter.

$$v = [\text{desired}q1\ddot{\phantom{x}}; \text{desired}q2\ddot{\phantom{x}}] - K * e$$

$$\tau = M * v + C * [q1\dot{\phantom{x}}; q2\dot{\phantom{x}}] + G;$$

2) *Robust Control*: Dynamics model of one leg approximates the impact of central body on each leg. This approximation can affect the trajectory control. Robust Control algorithm considers a maximum error in dynamics as  $\rho$ .  $\rho$  value is tuned to get better results

```

if  $\phi > 0$ 
  if  $\text{norm}(X.' * P * B) > \phi$ 
     $vr = -\rho * (X.' * P * B) / \text{norm}(X.' * P * B);$ 
  else
     $vr = -\rho * (X.' * P * B) / \phi;$ 
  end
else
  if  $\text{norm}(X.' * P * B) == 0$ 
     $vr = 0;$ 
  else
     $vr = -\rho * (X.' * P * B) / \text{norm}(X.' * P * B);$ 
  end
end

```

$$v = [\text{desired}q1\ddot{\phantom{x}}; \text{desired}q2\ddot{\phantom{x}}] - K * e + vr'$$

### III. RESULTS

In this section, we analyze and discuss the control performance on the simulation of both, the Crawler Bot robot model and the spot robot model for which we derived the dynamic equations as well as implemented control scripts in the previous section.

#### A. Creepy Bot

For the following trajectory equations and eigen values, we have implemented the feedback trajectory control:

$$\begin{aligned} \text{desired}q1 &= a0 + a1*t + a2*t.^2 + a3*t^3 + a4*t^4 + a5*t^5; \\ \text{desired}q2 &= b0 + b1*t + b2*t^2 + b3*t^3 + b4*t^4 + b5*t^5; \end{aligned}$$

where,

$$a0 = 0;$$

$$a1 = 0;$$

$$a2 = 0;$$

$$a4 = -7.8540;$$

$$a5 = \pi;$$

$$b0 = 0;$$

$$b1 = 0;$$

$$b2 = 0;$$

$$b3 = 6.3540;$$

$$b4 = -9.2810;$$

$$b5 = 3.7124;$$

$$\text{EigenValues} : [-15, -8, -10, -6]$$

For the following trajectory equations and values, we have implemented the robust control:

$$\begin{aligned} \text{desired}q1 &= a0 + a1*t + a2*t.^2 + a3*t^3 + a4*t^4 + a5*t^5; \\ \text{desired}q2 &= b0 + b1*t + b2*t^2 + b3*t^3 + b4*t^4 + b5*t^5; \end{aligned}$$

where,

$$a0 = 0;$$

$$a1 = 0;$$

$$a2 = 0;$$

$$a4 = -7.8540;$$

$$a5 = \pi;$$

$$b0 = 0;$$

$$b1 = 0;$$

$$b2 = 0;$$

$$b3 = 6.3540;$$

$$b4 = -9.2810;$$

$$b5 = 3.7124;$$

$$\text{EigenValues} : [-1, -1, -2, -3]$$

$$Q = \text{eye}(4)*2;$$

$$\phi = 0.05;$$

$$\rho = 5;$$

Figure 9 shows the model performance for following the above trajectory under feedback linearization control where as Figure 10 shows the same performance under robust control.

#### B. Creepy SPOT

For the following trajectory equations and values, we have implemented the feedback control:

$$\begin{aligned} \text{desired}q1 &= a0 + a1*t + a2*t.^2 + a3*t^3 + a4*t^4 + a5*t^5; \\ \text{desired}q2 &= b0 + b1*t + b2*t^2 + b3*t^3 + b4*t^4 + b5*t^5; \end{aligned}$$

where,

$$a0 = 0;$$

$$a1 = 0;$$

$$a2 = 0;$$

$$a4 = -7.8540;$$

$$a5 = \pi;$$

$$b0 = 0;$$

$$b1 = 0;$$

$$b2 = 0;$$

$b3 = 6.3540;$   
 $b4 = -9.2810;$   
 $b5 = 3.7124;$

*EigenValues* :  $[-3 - 2i, -3 + 2i, -3, -2.5]$

For the following trajectory equations and eigen values, we have implemented the robust control:

$desiredq1 = a0 + a1*t + a2*t.^2 + a3*t^3 + a4*t^4 + a5*t^5;$   
 $desiredq2 = b0 + b1*t + b2*t^2 + b3*t^3 + b4*t^4 + b5*t^5;$

where,

$a0 = 0;$   
 $a1 = 0;$   
 $a2 = 0;$   
 $a4 = -7.8540;$   
 $a5 = \pi;$

$b0 = 0;$   
 $b1 = 0;$   
 $b2 = 0;$   
 $b3 = 6.3540;$   
 $b4 = -9.2810;$   
 $b5 = 3.7124;$

Eigen Values:  $[-1, -1, -2, -3]$

$Q = eye(4)*3;$

$\phi = 0.05;$

$\rho = 4;$

Figure 11 shows the model performance for following the above trajectory under feedback linearization control where as Figure 12 shows the same performance under robust control.

#### IV. CONCLUSIONS

Sprawling Type quadruped was selected for the project because of its stable gait implementations. It is also an ideal candidate for decentralized control as it is symmetric around both the x and y-axis thus we can work with one leg dynamics. Using PoE we calculated the forward kinematics and used Moore-Penrose Pseudo Inverse method to calculate inverse kinematics. We calculated inverse dynamics using Recursive Newton-Euler and we got the torque values for each joint. We simulated the method in MATLAB to check its connectivity and it worked. We then used the torque values in gazebo simulation to control the leg.

For further work, the dynamics to push the robot forward with all four legs needs to be implemented after the control of all the legs has been done. For this a centralized controller needs to be implemented which will control the gating of the robot with all four arms.

As a stretched goal we have also implemented the mammal type quadruped robot model in gazebo. Its one leg model equations were calculated using Lagrangian formulation.

Dynamics Model were successfully designed for single leg of creepy Bot and creepy spot. Here dynamic models took into account the impact of 1/4th of centre body. Single Leg gazebo models were able to follow the trajectory successfully.

In Creeper Bot, a centralized controller was designed to give trajectory trigger values to the single leg control modules. A position controller was implemented to show the working of this. Creepy Bot was successfully able to follow the trajectory and move each leg to forward position. We were unable to add friction element to the dynamics of single leg and hence we couldn't push the center body forward.

In Creepy SPOT model was designed in gazebo. We tried to make the model stand on its own. But the center body of the robot applied uneven weight gravity forces on each leg. This kind of uncertainty in the dynamics of single leg is hard to predict and compensate for. Hence the robot is not able to stand properly. Future we can try to send the gravity forces applied by the center body to each leg by the central controller along with trajectory triggers. This will help in getting better dynamics of each leg.

#### GAZEBO VIDEOS

Kindly find the links to the relevant Gazebo animations:

- One Leg Robust: [Click Here](#)
- Creepy Bot: [Click Here](#)
- Creepy SPOT, attempting to stand: [Click Here](#)
- Creepy SPOT, attempting to crawl: [Click Here](#)

#### ACKNOWLEDGEMENTS

We would like to thank Prof. Farzan instructor of the RBE 502 controls course at the Robotics Engineering Department of Worcester Polytechnic Institute for supporting our project during the August-December 2021 semester. We owe our deep gratitude to him for taking keen interest in our project and guiding us, by providing all the necessary information and suggestions during this project work.

#### REFERENCES

- [1] Mahapatra, A., Roy, S. S., Pratihari, D. K. (2020). Multi-body Dynamic Modeling of Multi-legged Robots. Springer Nature.
- [2] De Santos, P. G., Garcia, E., Estremera, J. (2006). Quadrupedal locomotion: an introduction to the control of four-legged robots (Vol. 1). Verlag, London: springer.
- [3] J. Denavit, R.S. Hartenberg, A kinematic notation for lower-pair mechanisms based on matrices. Trans. ASME J Appl. Mech. 23, 215–221 (1955)
- [4] M.H. Raibert, Legged Robots That Balance, 1st edn. (The MIT Press, Cambridge, 1986)
- [5] Pandala, A., Kamidi, V. R., Hamed, K. A. (2020, October). Decentralized control schemes for stable quadrupedal locomotion: A decomposition approach from centralized controllers. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3975-3981). IEEE.
- [6] Erden, M. S., Leblebicioglu, K. (2007). Torque distribution in a six-legged robot. IEEE Transactions on Robotics, 23(1), 179-186.
- [7] Xu, K., Zi, P., Ding, X. (2019). Gait Analysis of Quadruped Robot Using the Equivalent Mechanism Concept Based on Metamorphosis. Chinese Journal of Mechanical Engineering, 32(1), 1-11.



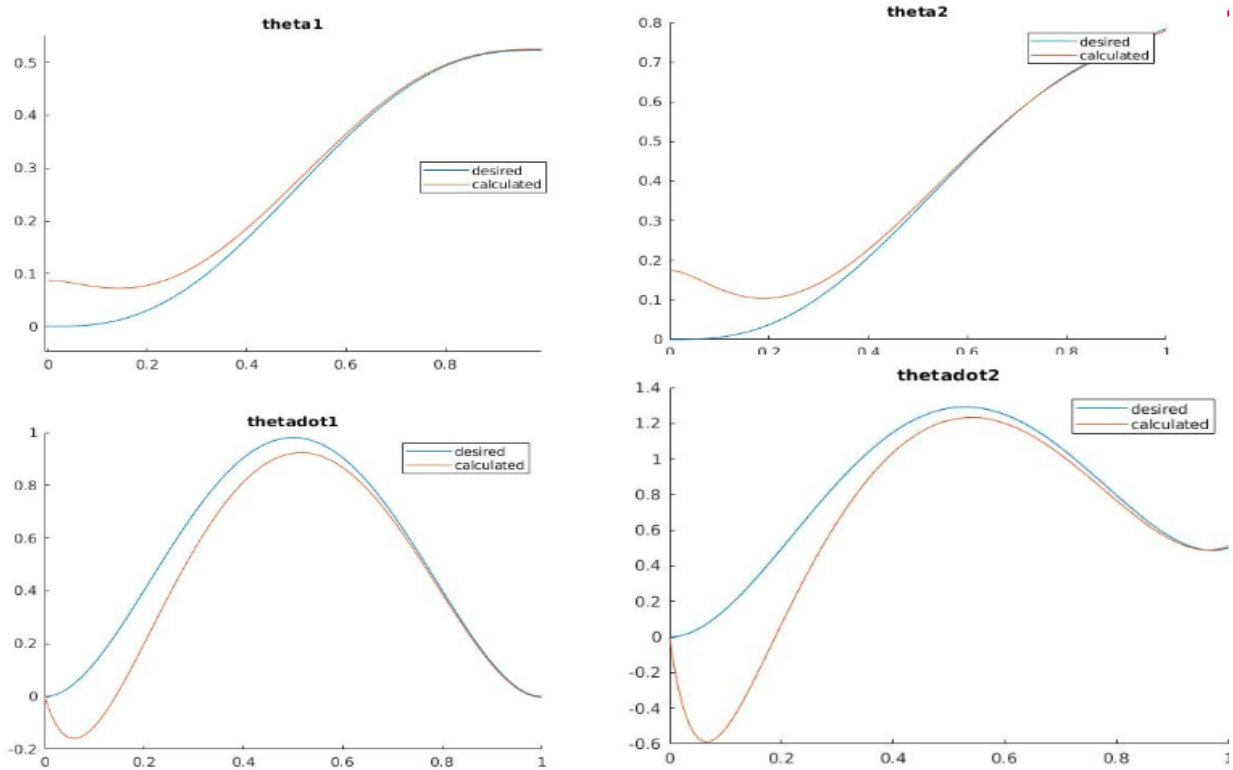


Fig. 9. Feedback Linearization Control for Creepy Bot

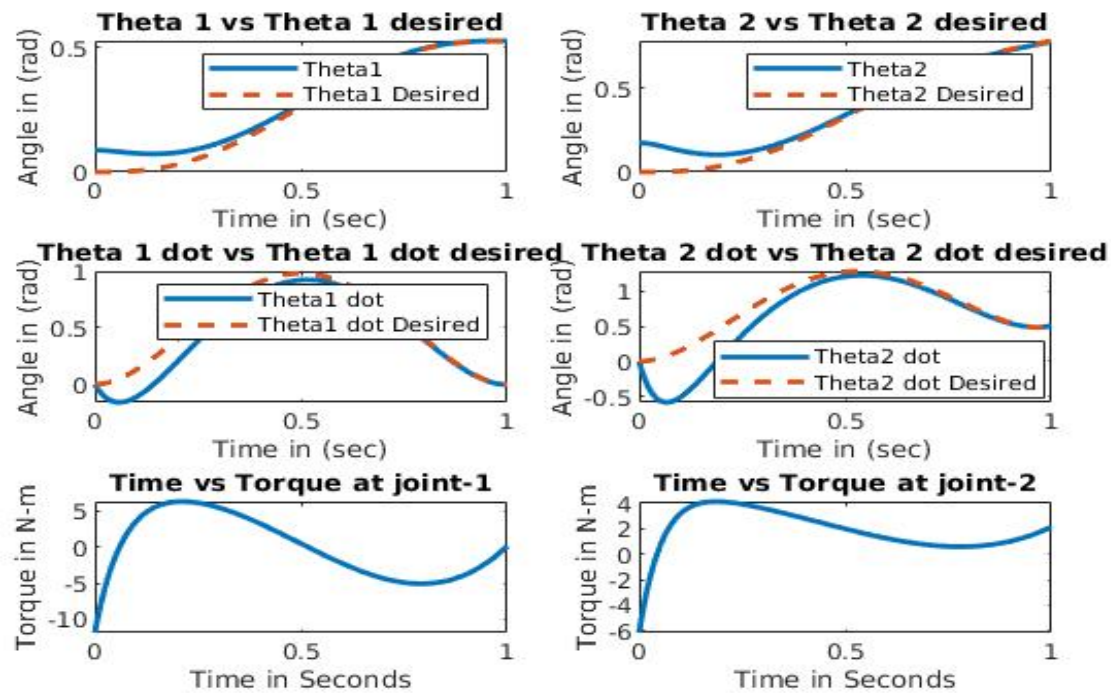


Fig. 10. Robust Control for Creepy Bot

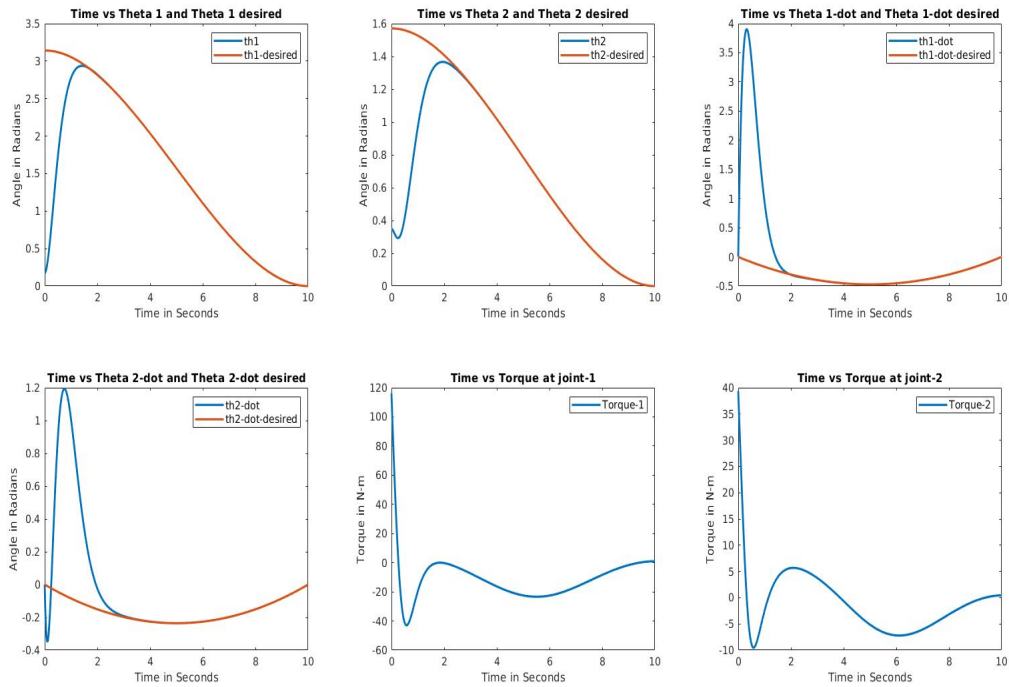


Fig. 11. Feedback Linearization Control for Creepy SPOT

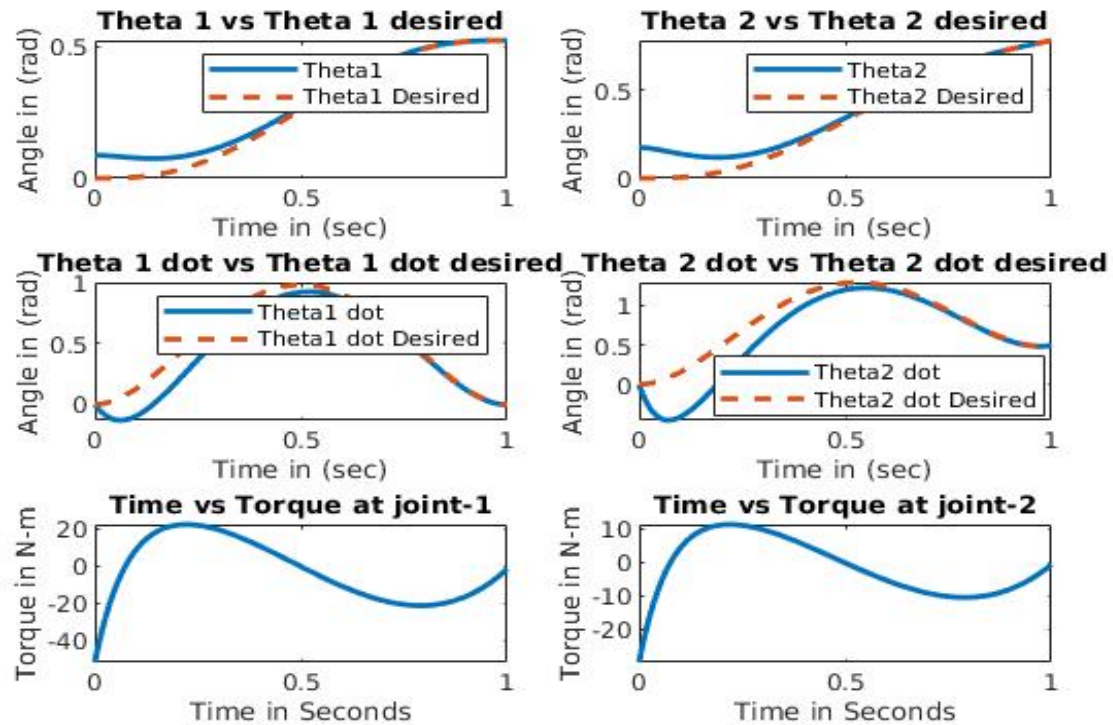


Fig. 12. Robust Control for Creepy SPOT

- [8] Zhong, Y., Wang, R., Feng, H., Chen, Y. (2019). Analysis and research of quadruped robot's legs: a comprehensive review. *International Journal of Advanced Robotic Systems*, 16(3), 1729881419844148.
- [9] De Santos, P. G., Garcia, E., Estremera, J. (2006). *Quadrupedal locomotion: an introduction to the control of four-legged robots* (Vol. 1). Verlag, London: springer.
- [10] Atique, M. M. U., Sarker, M. R. I., Ahad, M. A. R. (2018). Development of an 8 DOF quadruped robot and implementation of Inverse Kinematics using Denavit-Hartenberg convention. *Heliyon*, 4(12), e01053.