Şahin YILDIRIM
Erciyes University
Engineering Faculty

Erdem ARSLAN
Erciyes University
Engineering Faculty

ODE BASED JOINT CONTROLLER DESIGN FOR LEGGED ROBOTS

Keywords: *ODE(Open Dynamics Engine), Legged robot, Controller Design, Joint Controller*

Abstract

On legged robot controller design researchers mostly use a dynamic model for adjusting the controller parameters. The dynamic model of legged mobile robots has highly complex and nonlinear equations. With physics simulators like Open Dynamics Engine (ODE) this modeling can be accomplished simply by using iterative methods without the need for analytic solution of complex equations. In this investigation a real-time model based structure is preferred for legged robot control. The dynamic model required for model-reference adaptive control is designed as discrete-time by using ODE. This paper focuses on proving that the joint controllers used in model-based control work correctly when ODE is selected as reference model. In addition to this, the developed ODE based identical joint controllers for six legged robot has explained. The results of proposed discrete time P, PI, PID controllers are given in details on graphs.

Introduction

Nowadays legged locomotion is an applicable alternative because of the improvements on efficient actuator designs. Even if legged robots energy consumption is higher compare to wheeled locomotion the legged locomotion is an obligation in non-paved surfaces. According to project that researchers made for Logistical vehicle off-road mobility[1], %50 earth surface is unreachable for

[1]U.S.A.T.C.D. Agency, *Logistical Vehicle Off-road Mobility: Final Report: U.S. Army*, Transportation Combat Developments Agency, 1963.

wheeled vehicles. Because of this reason legged locomotion researches are important to open this %50 percent of earth surface to the service of mankind[2].

The main problems of legged locomotion are the actuator design and the control issue. Artificial legged locomotion systems have not done well energetically because of inappropriate choices of system configuration[3]. Legged mobile robots by their nature are not working as efficiently as wheeled mobile robots. Also because of the leg designs are more fragile compared to wheeled mechanism; wheeled locomotion is more preferred in locomotion.

The controllers developed for legged mobile robots are designed in a layered structure. These control layers are joint controllers, stability controllers and gait controllers, respectively. In the joint control layer, the current positions and velocities of the joints are controlled to be separate for each joint. Depending on the actuator, the controller types used in this control layer may vary, but in general, basic controllers such as P, PI and PID controllers are preferred.

There is a balance control layer on top of the joint control layer. In the stability control layer, the objective of control is to determine the joint variables required for the robot to be able to stance without tumbling. There is a gait controller on top of the balance control layer. In the gait control layer, the leg movement sequence is determined to take the robot from one point to another. Before this study, we tried to determine the required posture positions for the optimum balance point using the guided control algorithm developed on ODE[4].

Open Dynamics Engine is rigid-body dynamics implementation for robot simulation platforms. The most widely used robot simulation platforms Gazebo[5] and Webots[6] uses ODE as physics engine. The ODE can be used also as game engine. The game engine is computer software that changes the positions and velocities of the objects within the virtual world, taking into account physical laws, making the reactions look realistic. A lot of video games (Resident Evil: The Umbrella Chronicles, Mario Strikers Charged, Call of Juarez) also uses

---

[2] A. Morecki and K. J. Waldron, *Human and Machine Locomotion*: Springer Vienna, 2014.

[3] Yildirim Ş., Arslan E., "*Estimation of Contact Forces on Real-time Six Legged Mobile Robot with ODE (Open Dynamics Engine*)", International Conference on Advances in Mechanical Engineering ICAME 2016, ISTANBUL, TURKEY, 10-13 Mayıs 2016,

[4] Yildirim Ş., Arslan E., *Altı Bacaklı Mobil Robot için ODE Destekli Denge Kontrolü*, Ulusal Makina Teorisi Sempozyumu 2015, IZMIR, TURKEY, 14-17 Haziran 2015

[5] N. Koenig and A. Howard,*"Design and use paradigms for Gazebo, an open-source multi-robot simulator"*, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), 2004, pp. 2149-2154 vol.3

[6] O. Michel, "Cyberbotics Ltd. WebotsTM: Professional Mobile Robot Simulation " International Journal of Advanced Robotic Systems, vol. 1, pp. 39-42, 2004

ODE as physics engine under their infrastructure. The ODE can be use in military training applications as well[7].

The guided control algorithm is designed as a model-reference adaptive control. The key point in the guided control algorithm is the use of the ODE dynamic model for the reference model. When a standard model reference control is performed, the relationship between the input and output of the system must be fully modeled. And this is impossible for a six legged mobile robot with non-holonomic constraints.With the use of ODE, the reference model can be kept update easily by using the data from the sensors and this allows the model-based control for a six-legged mobile robot to be carried out easily without dealing with mathematical complexity.

In guided control algorithm, we tried to control the robot's posture in accordance with the real-time platform inclination values measured from the IMU sensor. The joint controllers' works under guided control algorithm was P type at that time. Because of the aim of that research is explaining the stabilization algorithm the development procedure of the joint controller was ignored before.

In this study, a discrete time feedback joint controllers used under guided control algorithm was explained. In addition to P type joint controller we examine the effect of PI and PID controllers which was developed as discrete time feedback system. The results, which prove the ODE can be used as a reference system when the reference model control is selected, are given in the graphs.

Open Dynamics Engine

Most of the multibody dynamics analysis software like MD-ADAMS, COMSOL or ANSYS uses physics engines in their infrastructure. The ODE platform has been developed by Russell Smith[8]. There are also a lot of physics engine (BULLET, NVDIA PhysX, Havok, MuJoCo) exists on market. Especially in computer games, considering the dynamic effects such as gravity or inertial effects of bodies gives to the game players more realistic experience.

In this study, an open source ODE code is selected instead of using packaged software like MD-ADAMS® because of the continuous data required to be sent to the motors on the robot in real time. ODE uses fixed difference method under its structure and works as a discrete time solver. Depending on step size the solver performance changes. Minimizing the step size gives realistic analysis

[7] P. McDowell, R. Darken, J. Sullivan, and E. Johnson, "*Delta3D: A Complete Open Source Game and Simulation Engine for Building Military Training Systems*," The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, vol. 3, pp. 143-154, 2006

[8] R. Kooijman, "*Evaluation of open dynamics engine software*," Eindhoven University of Technology2010.

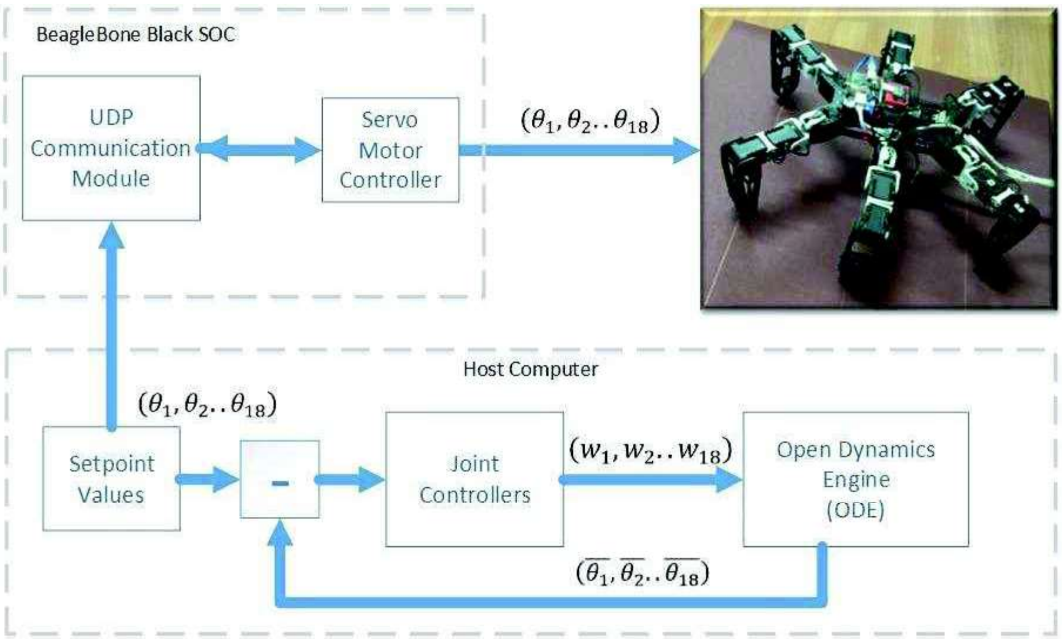
results but prolongs the time required for the solution. In this study we chose 5 ms as the size of the step to provide the real-time working condition.

While creating a dynamic model on ODE, firstly dynamic parameters such as mass, inertial tensor, joint positions must be defined. When the ODE is run for each step time, it is tried to find out positions and velocities of the parts after the limitations such as joint links and contact points are applied. If the parameters of ODE model are not set correctly, the solution is diverging or being too slow. Depending on the complexity of the dynamic model this parameter must be determined properly. In this study, these values were determined using the Russell Smith's limitation formulas described in [8].

Description of Experimental System

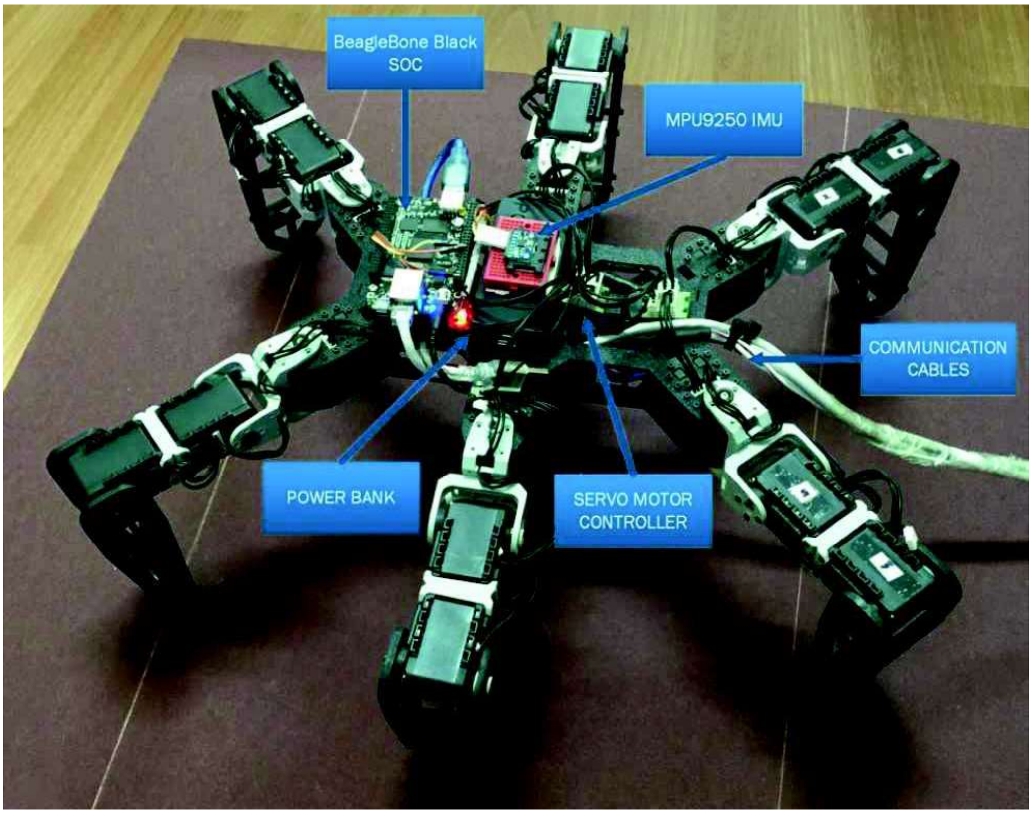
As we mentioned before we used the guided control algorithm for legged robot stability control. The structure of the joint controllers can be seen in the guided control algorithm given in figure 1. Guided control algorithm shown in Figure 1 consists of two different parts. In the first part of the algorithm, there is a computer program that can handle five simultaneous processes at the same time. This program developed by us, consists of 5 separate threads. These are; dynamic analysis, visual drawing, actuators communication, sensors communication and real-time graphical plotting process, respectively.

Fig.1. Block diagram of guided control algorithm



The second part of the algorithm works on a SOC element where the Debian operating system is installed. The six legged robot built for simultaneous work with platform and system elements shown in Fig. 2.

Fig.2. Six legged mobile robot and the system elements



In the middle of the robot shown in Fig. 2, the IMU sensor used to measure its orientation is placed. This sensor is used to measure the angular change that the robot made with respect to the ground plane by utilizing gravity acceleration. In addition to this sensor on robot there are BeagleBone Black SOC controller, a power-bank for operating the card, Usb2Dynamixel used for servo motors and communication cables for connection between main program and BeagleBone Black SOC controller.

Control System Design

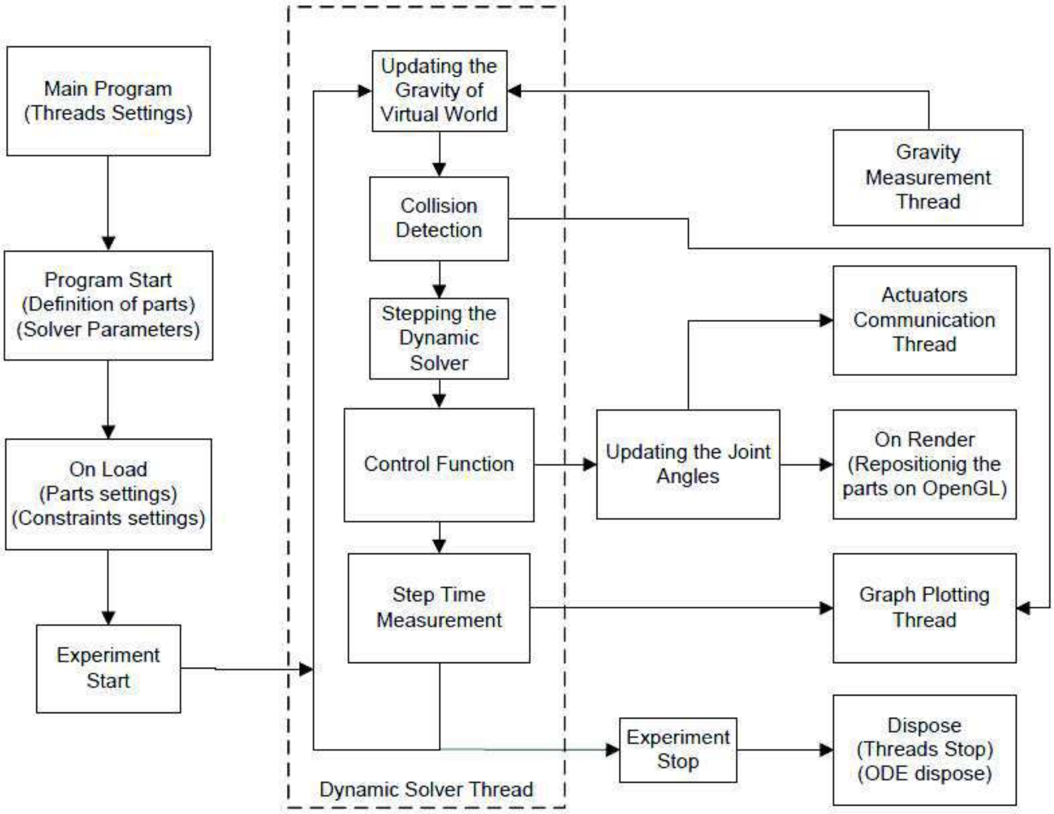
While designing a dynamic model on ODE, a virtual world must be created first. Parameters such as gravity, ground plane, contact parameter, solver parame-ters (ERP, CFM), step time size should be determined on this virtual world. After defining the dynamic parameters such as the inertia tensor and mass for the parts, the constraints in the dynamic modeling such as the joint points and the contact plane must be defined also. The outline of proposed control system is shown in fig. 3. In this control system, the test process is started after the dynamic parameters on the simulator are defined to be identical to the robot used in the test system. When the experiment is started, the five independent processes, dynamic solver operation, graph plotting, gravity measurement, communication with the actuators, and on render process are initiated simultaneously. In this system, acts as a discrete time controller, dynamic solver thread runs in an infinite loop.

In this loop, first the value measured from the gravity sensor is equalized to the gravity variable of the simulator. Then the contact points of the moving parts are determined and drawn as contact polygon on the result graph screen.

In the next step, the dynamic solver is executed to calculate the position and velocity of the parts after a step time. After solver has been operated for one step time, the control function is called which will calculate the speed values to be sent to the joints according to those joint angle errors. For a discrete-time controller it is very important to determine how much time each cycle spends. Because of this reason, as a final step in the cycle, the time spent in the loop is measured.

A standard feedback PID control structure is preferred while the control function is designed. In equation 7, the joint position error is calculated by taking the difference between the desired angle value of the joint and the angle value at that moment.

Fig.3. Six legged mobile robot and the system elements



$$e_i = \theta_{d_i} - \theta_{s_i} \tag{1}$$

In equation 2, the integral error required for I effect in controller is found by summing the current integral error and the current position error. It is necessary to calculate the slope of the error to account for the D effect in the loop.

$$T_I = T_I + e_i \tag{2}$$

To find the slope value, in equation 3, the difference between the current error and error value in the previous cycle is divided by the time spent between the two cycles.

$$s_i = (e_i - e_{i-1})/t_i \tag{3}$$

As a result, the formula in which the control signal is calculated in total is given in equation 4.

$$u_i = k_p . e_i + k_i . T_I + k_d * s_i \tag{4}$$

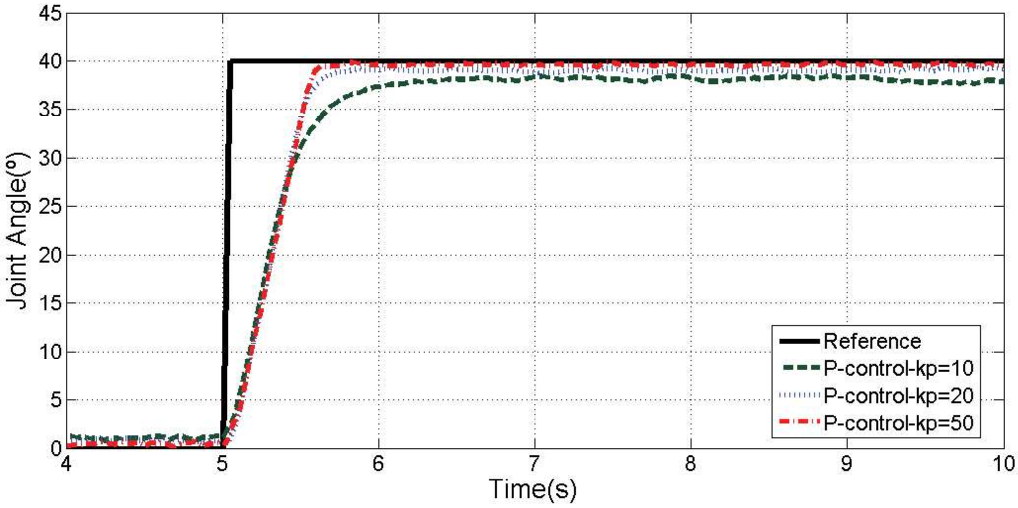
In equation 10, k_p represents the proportional control coefficient, the k_i represents the integral effect coefficient, and k_d represents the derivative effect coefficient.

Controller Results

As a test scenario, a stance is preferred in which the robot's 5 legs are in contact with ground to provide a static equilibrium condition. In order to observe the performance of the controller on a single joint, the angles between the body-coxa and tibia-femur parts in the moving leg were held constant. In the 5th second of the scenario a 40° angle change was applied to joint between coxa and tibia to lift the leg.

The robot used in the experiment has 18 joints in total. The controller we have developed has been used repeatedly 18 times to be separate for each joint. P type control was performed by setting the ki and k_d parameters to zero. Fig. 4 shows the change of the joint angle with respect to time when unit step input is applied for 3 different k_p values.

Fig.4. P control results of joint angle



In Fig. 5, PI control results for 3 different k_i values are given, keeping the k_d parameter at zero for the same motion scenario.

Fig.5. PI control results of joint angle

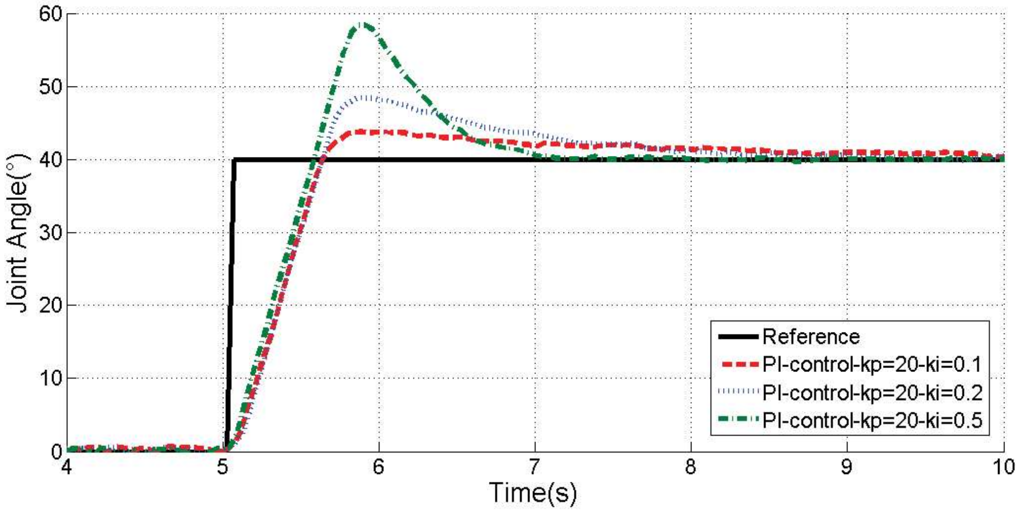
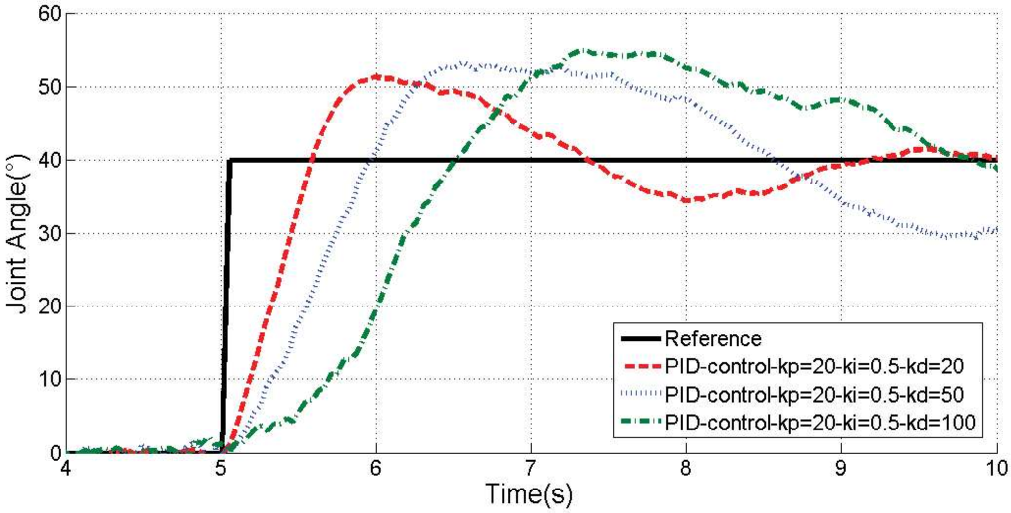


Figure 6 shows the change of joint angle for 3 different kd values where kp and ki parameters are kept constant for the same scenario.

Fig.6. PID control results of joint angle



Conclusions and discussion

In this simulation investigation, a discrete time feedback joint controllers used under guided control algorithm was explained. The guided control algorithm we developed in [3] used only P-type control for the joint controllers. In this research, I and D effects were added to the structure of this P type controller to see if there was any improvement in the performance of the joint controller.

When the P type control was performed in the joint controller, it was observed that the settlement time decreased significantly as the k_p value increased. If the k_p value is randomly selected too high, a high frequency oscillation is observed at the output. When the k_p value is chosen very small, it is seen that there is a steady-state error on the output.

The PI control type is tested with the aim of reducing the steady-state error and increasing the response speed and it is seen that the first overshoot value increases as the k_i parameter is increased.

The PID type control has been tried to reduce the maximum overshoot, but in this case, the response speed has decreased as k_d increased. Although there are many methods for selecting controller parameters (Cohen-coon, Ziegler-Nichols, Tyreus-Luyben), the parameters have not been tried to be ideally selected, because the objective of this paper is to prove the accuracy of ODE based controller.

As a result, it has been proven that the use of the P type controller in the joint controller is more appropriate. Similar results are seen when the k_p, k_i, and k_d parameters are changed in a continuous time feedback PID control example This result shows us that the design of a discrete time ODE based joint controller is made correct.

REFERENCES

[1] U.S.A.T.C.D. Agency, Logistical Vehicle Off-road Mobility: Final Report: U.S. Army, Transportation Combat Developments Agency, 1963.

[2] Morecki and K. J. Waldron, Human and Machine Locomotion: Springer Vienna, 2014.

[3] Yildirim Ş., Arslan E., "Estimation of Contact Forces on Real-time Six Legged Mobile Robot with ODE (Open Dynamics Engine)", International Conference on Advances in Mechanical Engineering ICAME 2016, ISTANBUL, TURKEY, 10-13 Mayıs 2016,

[4] Yildirim Ş., Arslan E., Altı Bacaklı Mobil Robot için ODE Destekli Denge Kontrolü, Ulusal Makina Teorisi Sempozyumu 2015, IZMIR, TURKEY, 14-17 Haziran 2015

[5] N. Koenig and A. Howard,"Design and use paradigms for Gazebo, an open-source multi-robot simulator", 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), 2004, pp. 2149-2154 vol.3

[6] O. Michel, "Cyberbotics Ltd. WebotsTM: Professional Mobile Robot Simulation " International Journal of Advanced Robotic Systems, vol. 1, pp. 39-42, 2004

[7] P. McDowell, R. Darken, J. Sullivan, and E. Johnson, "Delta3D: A Complete Open Source Game and Simulation Engine for Building Military Training Systems," The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, vol. 3, pp. 143-154, 2006

[8] R. Kooijman, "Evaluation of open dynamics engine software," Eindhoven University of Technology2010.