Impedance Control on Delta Robots

Dingkun Guo
dingkung@andrew.cmu.edu

Gaoyue Zhou
gaoyuez@andrew.cmu.edu

Zhizhuo Zhou
zhizhuoz@andrew.cmu.edu

December 10, 2021

Abstract

Position control on robots are easy to implement, however, not controlling the motor torque can lead to dangerous high impact situations. We implement an impedance controller in PyBullet physics simulator. We show the ability of our impedance controller to soften impact on free-falling robots with delta legs. Furthermore, we demonstrate the flexibility of our impedance controller by modifying spring constant K and damping factor D. Finally, we show cool visualizations in PyBullet.

1 Introduction

Delta robots are cheap and versatile, offering 6 degrees of freedom in a dome shaped working area with 3 motors. They are often deployed for packaging and sorting applications. One naive way to control delta robots is by controlling the motor angles, θ_i. However, only controlling the motor angles may result in high impact forces and dangerous situations when the robot comes into contact with obstacles or human limbs. This impact situation occurs when the position controller tries to maintain or reach a position with maximum available power of the motor despite hitting an obstacle. This is especially dangerous if robots coexist with humans in a shared workspace; if an industrial grade motor hits a human, the human may experience catastrophic damage. Therefore, it is crucial to develop robotic motor controllers that limit the maximum force a motor can exert and demonstrate flexibility under impact.

Impedance control introduces a dynamic way to manipulate robots that relates force and position. In electromagnetism, impedance is the amount of opposition to current in an alternating current circuit and represents the ratio of voltage to current in very simple terms. Drawing analogies electric impedance, impedance control focuses around the the ratio of force output to motion input. The impedance controller follows a spring mass damper system and regulates the relationship between force and position, and velocity and acceleration. The main benefit for impedance control is that we can use constants to adjust the amount of force motors exerts in reaction to external situations and use it to effectively soften sudden impacts. In our experiments, we are able to reduce the maximum force exerted by our delta robot during a drop scenario. We elaborate on our robotic setup and also the mathematics of impedance control in later sections.

2 Mathematics of Impedance Control

Impedance control imposes a desired dynamic behavior to the interaction between robot end-effector and environment. The goal is having motors acting like a virtual mass-spring-damper system as described in Figure 1 and Equation 1 below:

$$M\ddot{x} + D\dot{x} + Kx = F \tag{1}$$

where M is mass, D is damping coefficient, and K is stiffness coefficient. We can control how the robot behaves during an interaction with the environment by defining its stiffness and damping coefficient, K and D.

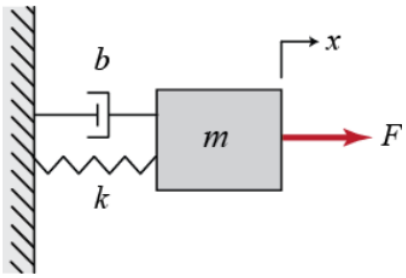

Figure 1: The mass-spring-damping system consists of an object with mass and connected via a srping and a damper. [2]

Then, we change the mass-spring-damper system into the imposition of a dynamic impedance model, shown in Equation 2 below:

$$M(\ddot{x} - \ddot{x}_{des}) + D(\dot{x} - \dot{x}_{des}) + K(x - x_{des}) = F_{ext} \tag{2}$$

where F_{ext} is external force, whose desired magnitude is specified through a generalized dynamic impedance.

After that, we add feedback control in the impedance model, and by applying an analytical Jacobian, we convert force to torque, which can be directly used to control motors, as shown in Equation 3 [1] below:

$$u_\tau = J_a^T(M\ddot{x} + D\dot{x} + Kx - F_{ext}) \tag{3}$$

where J_a^T is the analytical Jacobian. Or we can instead use external torques directly to find control signal as described in Equation 4 [1] below:

$$u_\tau = M\ddot{x} + D\dot{x} + Kx - F_{ext} \tag{4}$$

3 Implementation in PyBullet

We conduct our robotic simulations using the open-source PyBullet [] simulator. To accelerate the process of development and focus on implementing the controller, we use an existing tridelta robot template. The robot can be seen in Figure 2. The robot has three independent delta legs. While delta arms are generally used for grasping, sorting, or 3D printing, we demonstrate our impedance controller in a free fall drop setup. Our free fall drop simulation consists of three main parts: robot URDF, impedance controller, and experimental data logging.

3.1 Robot URDF

We use an existing tripedal delta robot configuration as a base for demonstrating impedance control. In short, the URDF file consists of definitions for a group of links connected by joints of different types. In our case, the robot has four parts: three separate delta robots and the base connecting all the delta robots at the top via joints.

As shown in Figure 3, each delta robot is roughly composed of three gray upper legs, purple parallel lower legs, and the joints connecting them with positions shown in the red links. Among all the joints, only the three joints connecting the

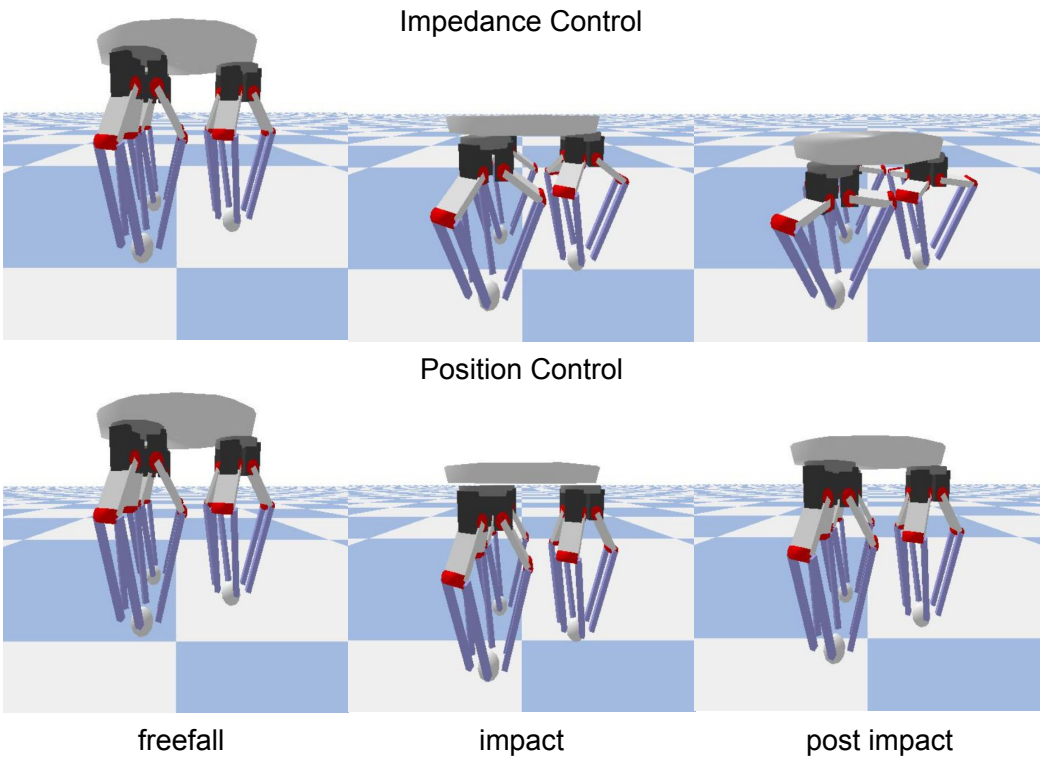

Figure 2: Visualization of tridelta robot during free fall, right after impact, and a longer time after impact. The top row shows the robot under impedance control and the bottom row shows the robot with pure position control. Notice how the impedance controller folds the legs to dampen the sudden impact.

upper legs and the lower legs are of the revolute type. The revolute joints act as motors, which allow us to effectively control the delta robot by applying forces to these joints.

At the bottom of each delta robot, we create a constraint via a sphere connecting all the parallel lower legs. This constraint simulates the end effector of the real-world delta robot where a gripper can be attached.

Last but not least, in our demonstration of free fall impact, we create another constraint that fixes the x and y-axis of our tripedal delta robot for stability purposes.

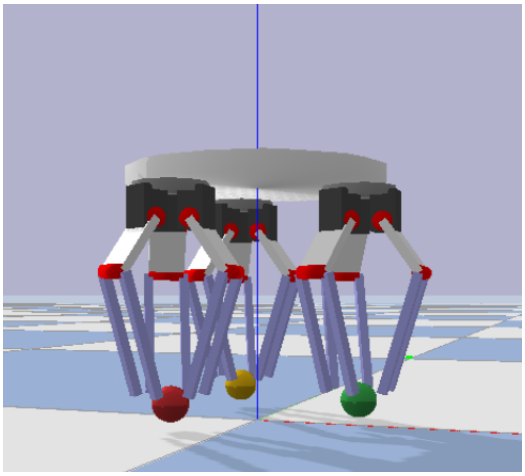

Figure 3: Visualization of tridelta robot in resting state. The base is the gray disk on the top connecting all three delta robots. For each delta robot, it mainly consists of three gray upper legs, six purple parallel lower legs, and joints in the position of the red links. The three sphere at the bottom serve as a constraint connecting all lower legs for each delta robot.

3.2 Impedance Controller

There are multiple control modes available in PyBullet, for example, position control, torque control, velocity control, and PD control. Since the amount of torque is the input to the system for impedance control, we use PyBullet's torque control mode as our control interface, where we need to specify the joint to exert the torque and the desired amount of torque to be exerted. To serve as a comparison to impedance control, we also control the robot via position control, which takes in the desired joint position and maximum torque allowed as inputs. Both control modes are achieved by calling the function *setJointMotorControl2*.

3.3 Data Logging

In order to have detailed comparisons in our experiments, we need to access and log the states of the robot and its joints which are available in the simulator. More specifically, we focus on the torque exerted on the motors, the position (in joint angles) and velocity of the motors, and the z-coordinate of the entire robot. We use the function *getJointState* to obtain the current position, velocity, and torque of

the target joint. Note that for impedance control where we use torque control, the torque exerted on the joints is instead given by our calculation from Equation 4 at each time step. As for the robot position, we use *getBasePositionAndOrientation* to access the robot base position.

4 Experiments

We conduct the free fall drop experiment using the same tridelta robot with different motor controls and various constants in the case of impedance controller. The ultimate goal of impedance controller is to make robots safer in an environment with humans; thus we especially care about the maximal force exercted by the motors.

4.1 Free Fall Impact

Figure 4 shows images taken at various timesteps of the free fall experiment with various motor controllers. First, we notice that the baseline (B) robot with position control has very rigid legs and an overall rigid reaction to the ground impact under gravity. The baseline robot comes in contact with the ground for a relatively short period of time right after impact, and as a result, the motors experience a higher impact torque for a short amount of time. In contrast, all impedance controlled robots (K) come in contact with the ground for a longer time after impact and experiences lower maximum impact torque. This follows the principles of linear momentum and force. Right before impact, the robot experiences momentum mv where m is the mass of the robot and v is the velocity right before impact. As the robot comes to a stop, the robot has to exert impulse, which is force times time, equal to the inital momentum. We can see that the impulse is the area under the graph of force and time. Notice that the graphs in Figure 4 shows the torque exerted by one of the 9 motors and is not exactly the linear force in the direction of gravity, however, the relationship that the area under of torque-time graph to bring the robot to a stop is the same regardless of the controller. In position control, we experience high torque for a shorter duration. In impedance control, we can control how we want to slow down the robot. With a higher spring constant K, the robot is able to exert higher force and come to a stop quicker. With a low spring constant K, the robot exerts lower torque and instead takes a longer time to come to a stop. Here we notice that impedance control with a high enough spring constant K is effectively position control.

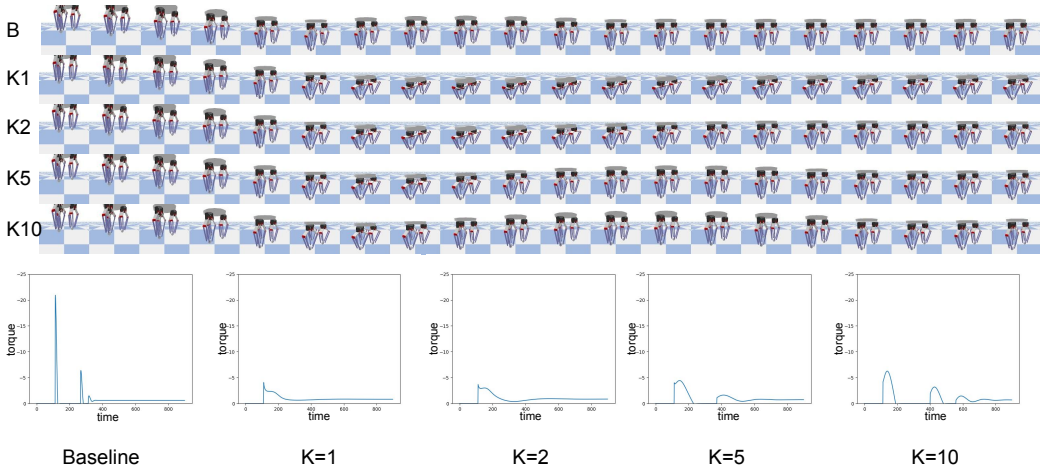

Figure 4: Visualization showing free fall and impact of our tridelta robot in PyBullet with position control baseline and impedance controller (with varying spring constant K). Plots of motor torque with respect to time is shown below for all the visualizations above. B is the baseline position controller. All K samples are impedance controllers with the specified constant K and the damping factor D held constant at 0.1. For example, K1 is impedance controller with K = 1 and D = 0.1. When the legs are not touching the ground, the toqrue is 0. In the end, when the robot is standing on the ground, the torque stabilizes to roughly 1 newton meter.

We effectively showed that impedance controller can help reduce the maximal force exerted by the motors in an impact situation. Furthermore, we demonstrated the flexibility of an impedance controller. We see that a higher spring constant K leads to a higher peak force and shorter impact time, and vice versa. This fine grained control over motor behavior is incredibly useful when designing robots to interact with objects and people in dynamic settings while factoring safety.

4.2 Varying Stiffness

In our free fall impact setting, with fixed impedance, the system behaves as a spring-mass system. The resting state of the spring is the predetermined desired

joint position under the effect of gravity and the spring's stiffness. From Equation 2 we can see that both K and D are tunable parameters that determine the property of the system. In this section, we explore the effect of K, the stiffness of the spring, and see how the robot behaves differently when free-falling from the same initial height. Varying the stiffness of the system is crucial since it determines how drastic the system would respond when external forces are exerted. For example, when manipulating fragile objects, we may want to lower the stiffness of the system so we can sacrifice control accuracy in order to minimize contact forces.

In Figure 5, we plot the z-coordinate of the robot after releasing it from the initial position with $K = 1$, 10, and 100 respectively, while fixing D, the damping factor, to be at a constant 0.01. Since higher K means we have a stiffer string, the system will have more potential energy once the joint is moved away from its desired joint angle. Therefore, we expect to have larger total energy in the system for higher K. We can see from Figure 5 that system oscillates at a higher frequency as K increases, which matches our expectation since the frequency $f = \sqrt{\frac{K}{m}}$ where m is the mass of the spring. Also, we can see that the magnitude of the oscillation dies off slower as K increases.

The stiffness also affects the final equilibrium position of the system. The three pictures in the right column in Figure 5 show the equilibrium position for each K respectively. Although the desired joint angle is the same in all cases, since we have gravity acting downward, it counteracts the force exerted by the joints from impedance control. As K increases, the equilibrium position becomes closer to the predetermined desired joint angle.

4.3 Varying Damping Factor

The damping factor describes how oscillations in a system decay after some disturbance. In real-world applications, introducing a damping factor is a way to limit vibrations in the system, and it is essential for protecting the system in which it operates. Damping removes energy from the system through resistance to motion.

In Figure 6, we plot the z-coordinate of the robot with time with $D = 0.001$, 0.01, and 0.1 while fixing K to be 10. From Equation 2 we can see that the damping force counteracting the elastic force increases linearly as the velocity of the system. In the plots, we obtain the expected behavior that the oscillation decays faster as D increases.

4.4 Compare with Position Control

In the free-fall experiment, as indicated in Figure 4, one downside for position control during the collision is the huge torque exerted on the motors, which can cause severe damage to the environment and the robot itself. What if we change the position control parameter so that the magnitude of the maximum torque allowed is also below 7.5 just like impedance control with K between 1 to 10? Figure 7 shows the comparison of the torque exerted on joint, z-coordinate of the robot, joint position, and joint velocity for a position-controlled robot with maximum torque limited to 7.5 (left column) and an impedance-controlled robot with $K = 10$ and $D = 0.1$ (right column). As shown in the z-coordinate plots, the robot using position control can also jump back up a few times during the free fall, and the z-coordinate of the two robots are pretty similar. However, the robot using impedance control has a smoother oscillation that resembles a sinusoidal wave. Moreover, for the torque exerted on the joint, we can see that for position control, the torque has been clipped at the 7.5 maximum allowed torque with many abrupt changes, while impedance control gives more continuous torque changes. This can potentially leave lighter pressure on the control system and resulting in more natural behavior.

5 Conclusion

In conclusion, we implemented an impedance controller in PyBullet and showed that an impedance controller is very flexible. We can adjust the constants K and D to control the behavior of the motors. In a situation where a robot is interacting physically with a human, we can use impedance controller to make the motor act like a spring mass damping system to soften any sudden contacts.

References

[1] Alessandro De Luca. Impedance control. 3

[2] ProgrammerWorld says:. Design spring mass damping system in simulink, Apr 2020. 2

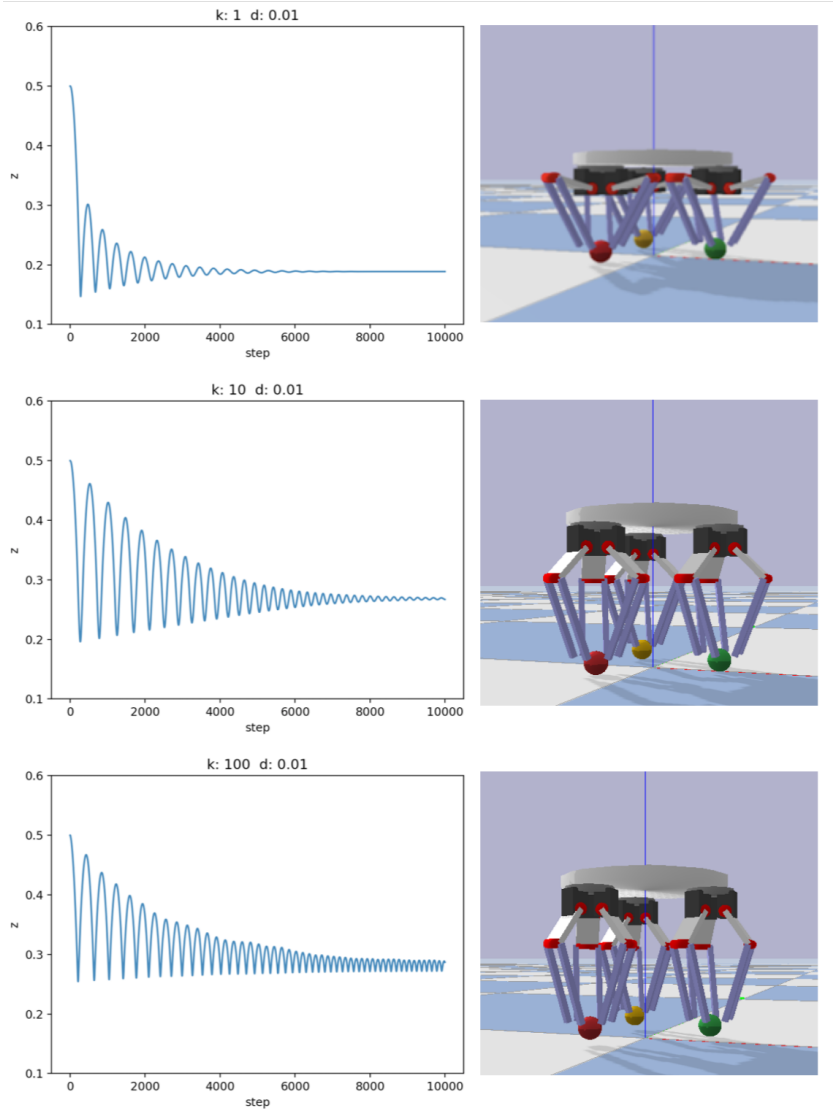

Figure 5: Plots of the robot's z-coordinate with varying stiffness K and fixed damping factor D along with the equilibrium positions in each case. The robot oscillates faster at a higher frequency as K increases and comes to a resting position with joint states closest to the desired ones.

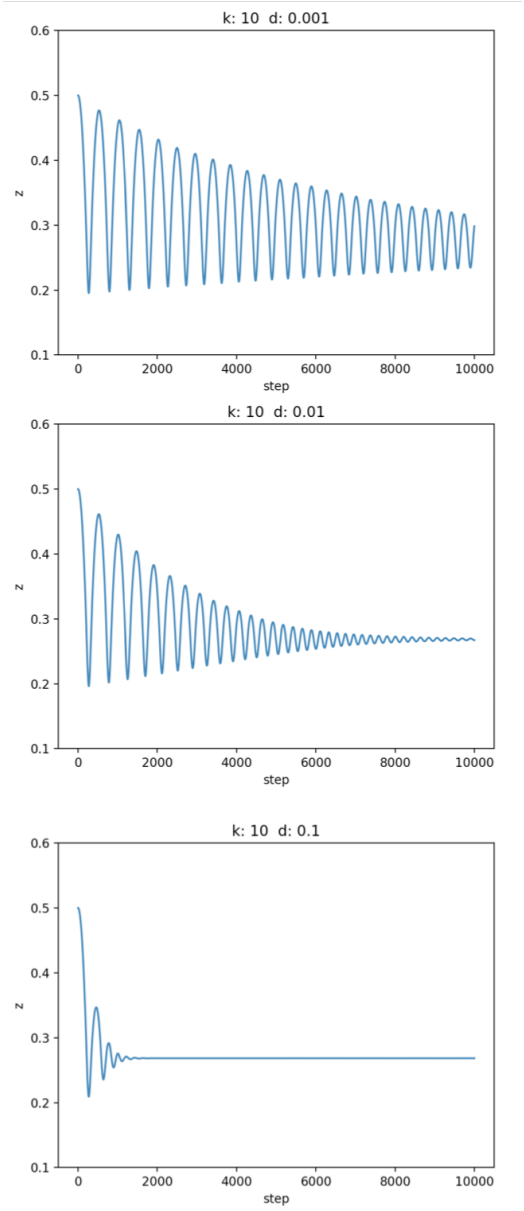

Figure 6: Plots of the robot's z-coordinate with varying damping factor D and fixed stiffness K. The oscillation decays faster as D increases.

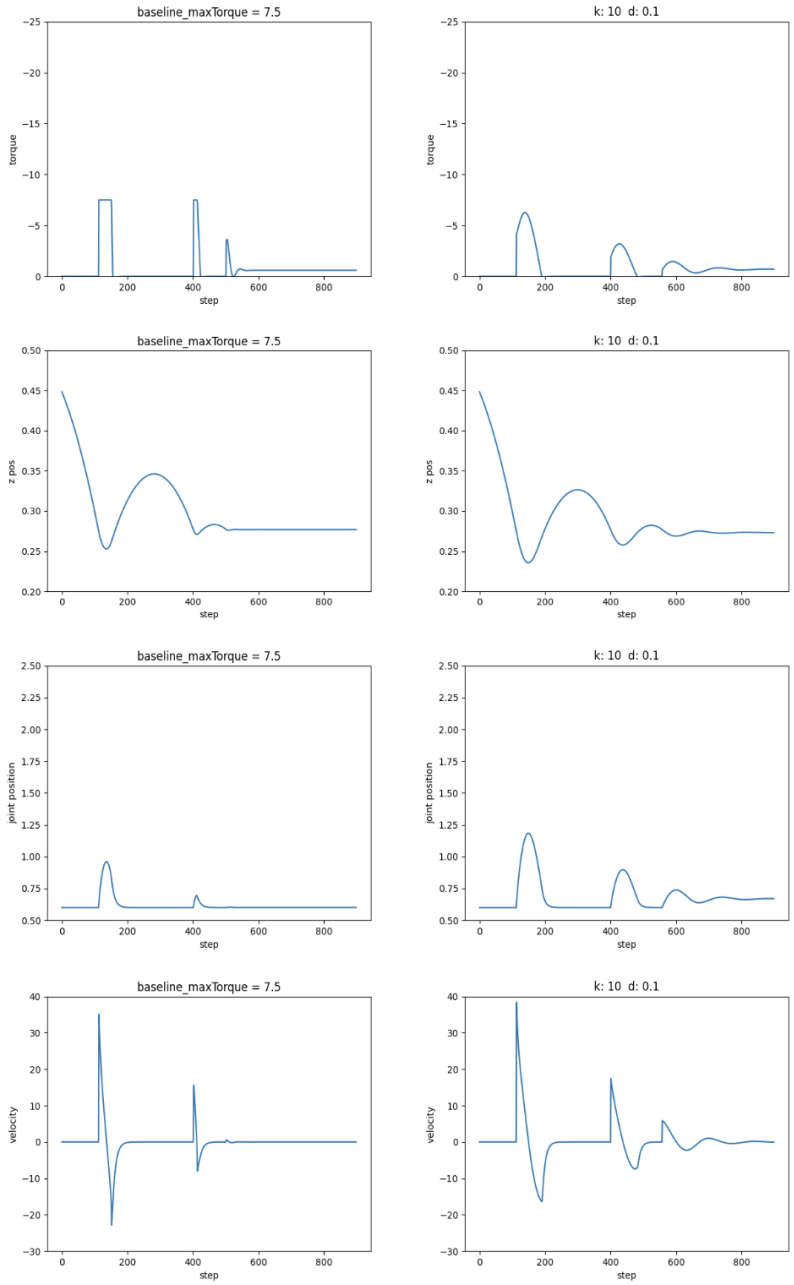

Figure 7: Plots of the robot's z-coordinate with varying damping factor D and fixed stiffness K. The oscillation decays faster as D increases.