Control Theory - Home Work 2

Alina Bogdanova BS18-04

February 2020

1 Introduction

Consider spring-mass-damper system and complete the following tasks:

2 Task 1

Task: design PD-controller that tracks time varying reference states i.e. $\begin{bmatrix} x & \dot{x} \end{bmatrix}^T$ as closely as possible. Test your controller on different trajectories;

First of all, what is PD control for our system?

PD control is a control in form $u = e \cdot k_p + \dot{e} \cdot k_d$, where e is an error function: $e = x^* - x$, where x^* is our desired value.

Thus out system takes form:

$$\dot{X} = AX + B \begin{bmatrix} k_d & k_p \\ 0 & 0 \end{bmatrix} X,$$

where $X = \begin{bmatrix} \dot{x} \\ x \end{bmatrix}$, $B = \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix}$

Let's consider different trajectories of the system.

The trajectory is defined my this set of parameters: m, b, k, x_0, $\dot{x_0}$, desired values (x^* and $\dot{x^*}$) and the controller parameters (k_p and k_d).

Let's try a few sets of parameters:

$x^* = 1$, $\dot{x^*} = 0$ - for all tests

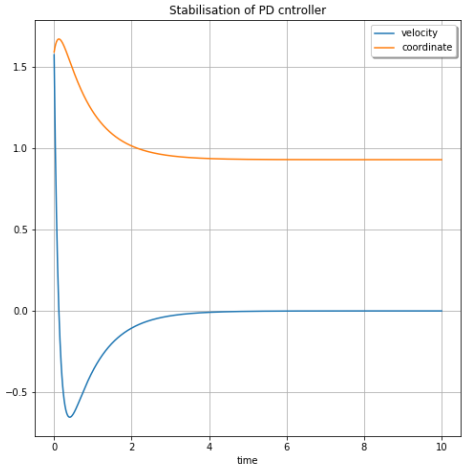

Figure 1: $m = 0.1$, $k = 0.5$, $b = 0.07$, $k_p = 0.9$, $k_d = 0.4$

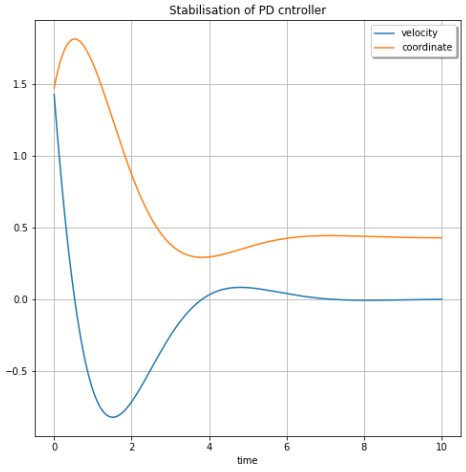

Figure 2: $m = 1$, $k = 0.8$, $b = 0.6$, $k_p = 0.6$, $k_d = 0.8$

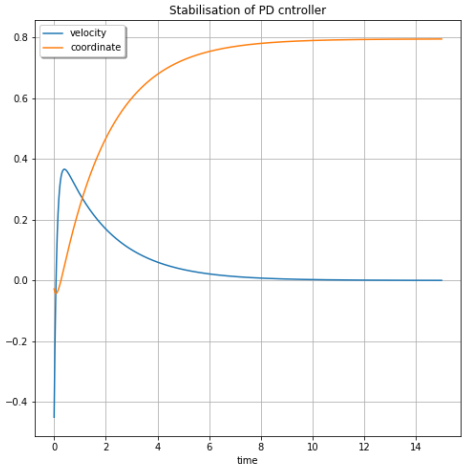

Figure 3: $m = 0.1$, $k = 0.1$, $b = 0.2$, $k_p = 0.4$, $k_d = 0.8$, $x_0 < 0$

At the figure 1 the smooth stopping around the desired values can be seen, at the figure 2 there are a smooth oscillations and the coordinate is quite far from the desired value, while at the graph3 there is a growing without oscillations according to the coordinate and some oscillations of the velocity.

3 Task 2 and 3

Task 2: find k_p and k_d such that there are no oscillations and no overshoot in the system. Prove it on step input signal;
Task 3: prove that spring-mass-damper system is stable with PD-controller for k_p and k_d of your choice.

First of all, lets use some parameters from the previous examples to reach graphs without oscillations and overshooting.

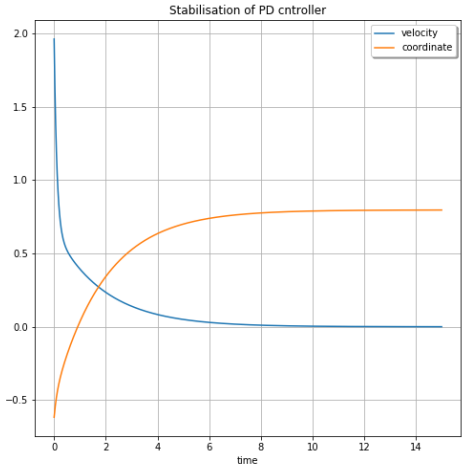

Figure 4: $m = 0.1$, $k = 0.1$, $b = 0.2$, $k_p = 0.4$, $k_d = 0.8$, $x_0 < 0$, $\dot{x^*} > 0$

The parameters on figure 4 are the same as as on graph 3, but the velocity, which is now greater then 0 initially.

Let's check if the system is stable.

The first way to deal with it is to check eigenvalues of the resulted matrix.

$$\dot{X} = AX + B \begin{bmatrix} k_d & k_p \\ 0 &) \end{bmatrix} X$$

$$\dot{X} = (A + B \begin{bmatrix} k_d & k_p \\ 0 &) \end{bmatrix})X$$

$$A = \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix}$$

Now let's try to find eigenvalues of a resulted matrix:

$$Det(\begin{bmatrix} -2 - k_d - \lambda & k_p \\ 1 & -\lambda \end{bmatrix}) = \lambda^2 + \lambda(2 - k_d) - k_p + = 0$$

$$k_p = 0.4, k_d = 0.8$$

$$D = 1.2^2 + 1.6 - 4 \approx -1$$

$$\lambda \approx \frac{-1.6 \pm i}{2}$$

Real part of lambda is less then zero, thus, the given system is stable.

Another way to check the stability of the system is Bode plot:

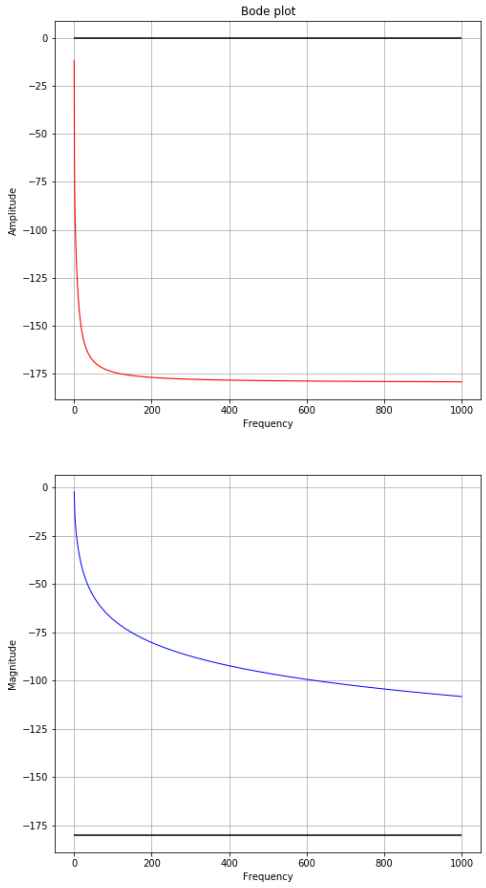

Figure 5: Bode plot of the given spring-mass-dumper system

The plot was made using the transfer function.

$$m\ddot{x} + b\dot{x} + kx = u$$

$$\ddot{x} + \frac{b}{m}\dot{x} + \frac{k}{m}x = \frac{k_d}{m}(\dot{x^*} - \dot{x}) + \frac{k_p}{m}(x^* - x)$$

$$p^2x + px(\frac{b}{m} + \frac{k_d}{m}) + (\frac{k}{m} + \frac{k_p}{m})x = \frac{k_d}{m}\dot{x^*} + \frac{k_p}{m}x^*$$

$$x = \frac{k_d \dot{x^*} + k_p x^*}{mp^2 + p(b + k_d) + (k + k_p)}$$

4 Task 4

Task: implement PI/PID controller and compare it to PD controller. What are the differences between PI, PID and PD controller? The general form of input is $u = k_p e + k_d \dot{e} + k_i \sum e_i dt$, which is PID controller. Let's change the parameters a bit and look how will our system behave: For PID controller the parameters are: $k_i = 0.001, k_p = 20, k_d = 4$, for PI controller the parameters are the same as for PID, but $k_d = 0$, ad fo PD the $k_i = 0$.

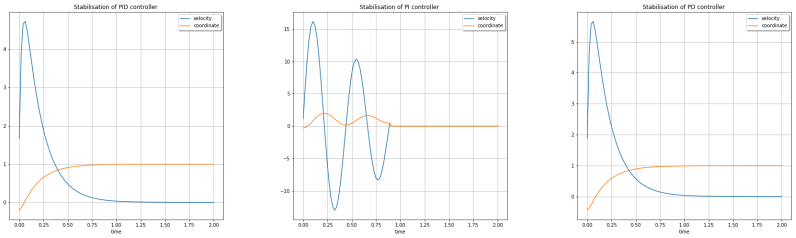

Figure 6: Different type of controllers, applied to the same initial conditions

As it can be seen, the PID is now similar to PD. It should not happen this way.

If the k_i parameter will be increased (figure 7), the graph will show the computational errors due to the ode solver, used in the task. Thus, PID is probably better method, but it should be used with some different tools.

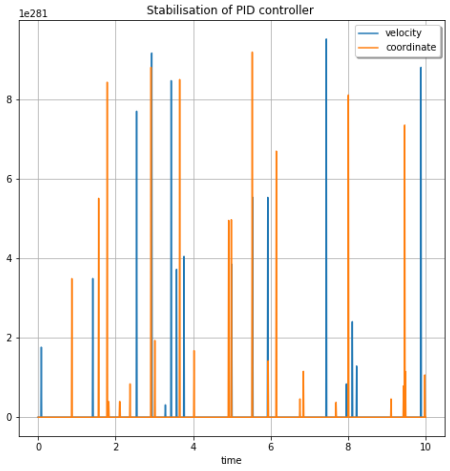

Figure 7: PID control with $k_i = 0.01$

Why does it happen?

"odeint" solver runs method for computations a few times for the same time unit, that is why, total error increases dramatically. In addition, there is a piece of random in odeint method (for multiple executions it shows different results)