Christian Schilling
Jakob Ø. Hansen
Kim G. Larsen
Martijn Goorden
Max Tschaikowski
Milad Samim

AALBORG UNIVERSITY
DENMARK

Models and Tools for Cyber-Physical Systems
Exercise sheet 9
WITH SOLUTIONS

Exercise 1: Parking a Car

Consider the car model from the course

$$\frac{d}{dt}x(t) = v(t) \qquad \qquad \frac{d}{dt}v(t) = -\frac{k}{m}v(t) + \frac{1}{m}F(t),$$

where x is the position, v the velocity and F the force applied in case of mass m and friction coefficient k. Starting at position $x(0) = -100$, we wish to bring the car to the point 0. To this end, we want to construct a proportional controller K_p for $m = 1000$ (kg) and $k = 50$.

(a) Assuming that we have no feedforward control and measure only the position of the car, provide a continuous-time component for the parking model in block diagram form.

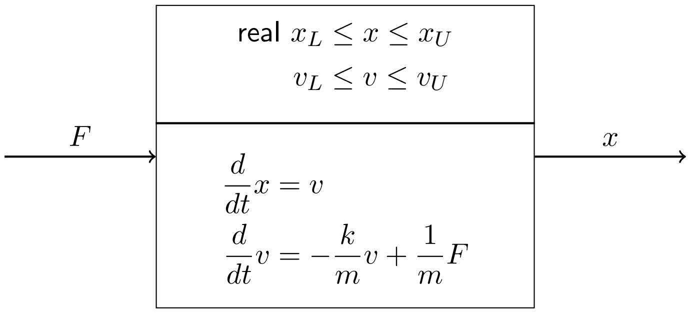
(b) Create a continuous-time component of the proportional controller in block diagram form and indicate how the controller component is connected to the car model component. Finally, combine them together into a single continuous-time component.

(c) Using your numerical solver from Exercise sheet 8, tune your controller, i.e., find appropriate values for K_p, such that the car is parked at $x = 0$ within $t = 100$ (s). Describe the quality of your controller in terms of steady state error, settling time, overshoot and rise time.

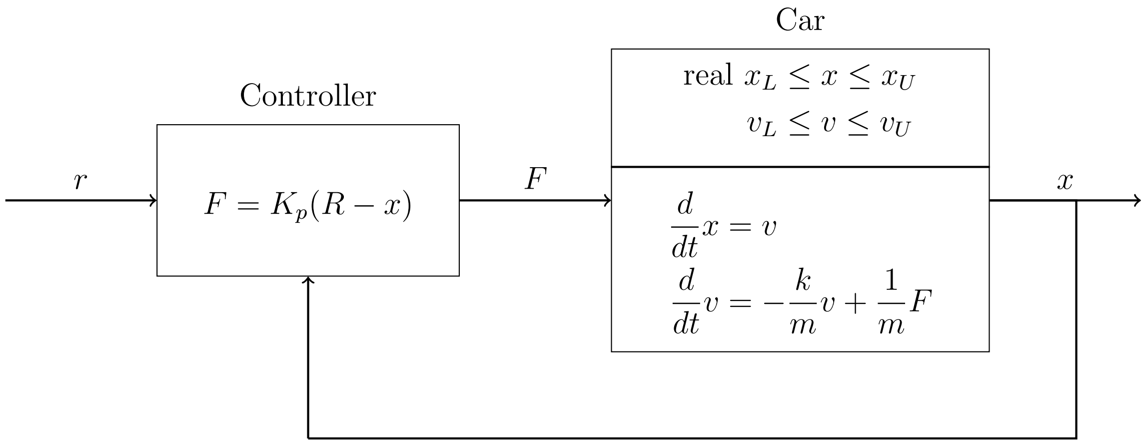
(d) Derive a formula for the Proportional-Derivative (PD) controller that brings the car to the state $(x, v) = (100, 0)$. Here a PD controller is a PID controller whose integral component has coefficient zero, i.e., $K_i = 0$. Moreover, implement the PD car controller using your numerical solver and tune the values of K_p and K_d. Describe the quality of your controller in terms of steady state error, settling time, overshoot and rise time.

. . . . Solution sketch . . . .

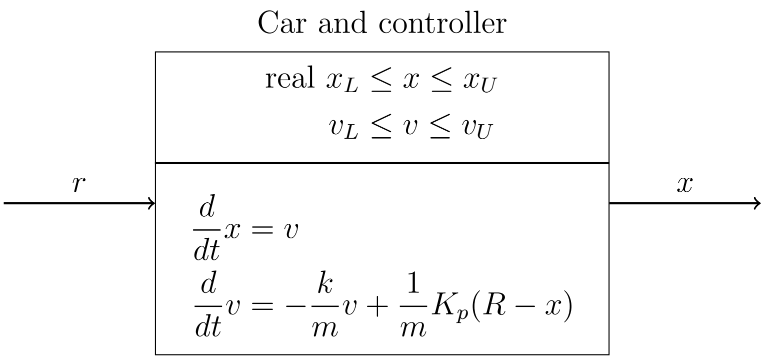
(a) The model is already given as a set of first-order differential equations, so we do not have to do any rewriting. The continuous-time component of the car is shown below.



(b) The continuous-time component of the controller is shown below. Note that a proportional controller does not have an internal state.



The combined continuous-time component of the car and the controller is shown below.



(c) You can use the solution code from Exercise sheet 8 where you define function `f` to be as follows.

```
def f(x, t):
    m = 1000
    k = 50
    Kp = 10
    r = 0
    der_x = x[1]
    der_v = -k / m * x[1] + Kp * (r - x[0]) / m
    return [der_x, der_v]
```

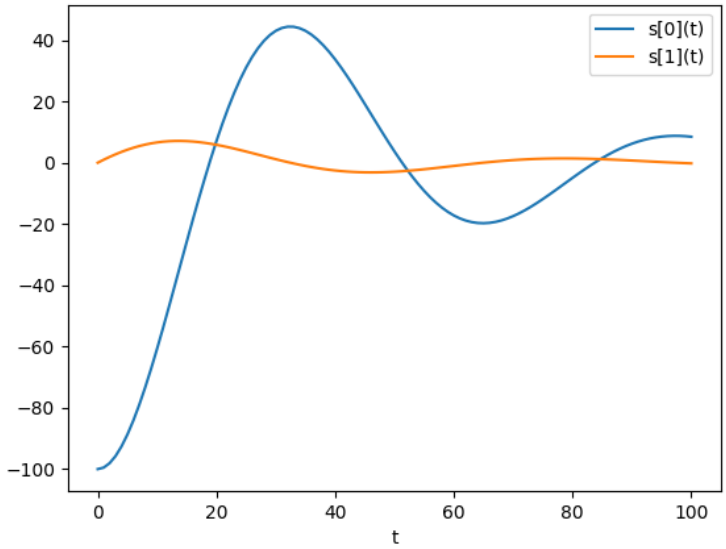


Figure 1: Car model state trajectories with only proportional controller $K_p = 10$.

Figure 1 shows a plot of position x (blue) and velocity v (orange) for a gain of $K_p = 10$.

By changing the value of Kp you can observe that low values of K_p result in a controller that has no overshoot but a very long settling time, while high values of K_p result in a fast rise time but significant overshoot and not really a fast settling time. So a proportional controller does not really have the desired performance.

(d) We first note that the error is $e = (r - x) = (100 - x)$. Hence, $u_p = K_p(100 - x)$ and $u_d = K_d \frac{d}{dt}(100 - x) = -K_d v$. With this, the overall control is thus $F = u = u_p + u_d = K_p(100 - x) - K_d v$.[1]

Note that picking PD gains is equivalent to picking a gain matrix K for a fully observable system. This correspondence holds true in general for single input single output systems (SISO), that is, for any PID gains, there exist a gain matrix giving rise to the same control and vice versa. However, finding good PID gains is easier than finding good gain matrices (or eigenvalues).

A possible Python code for the PD controller is given below. The resulting trajectories of the car's state variables are shown in Figure 2 for $K_p = 20$ and $K_d = 200$ (the values in the script below).

```
def f(x, t):
  m = 1000
  k = 50
  Kp = 20
  Kd = 200
  r = 0
  F = Kp * (r - x[0]) - Kd * x[1]
  der_x = x[1]
```

[1] Note that in this example the derivative of the error can be expressed directly using state variables of the system. In general, it would be possible to implement a PD controller by approximating $\frac{d}{dt}e$ via the differential quotient. To this end, one would need to know the values of t and e of the foregoing step.

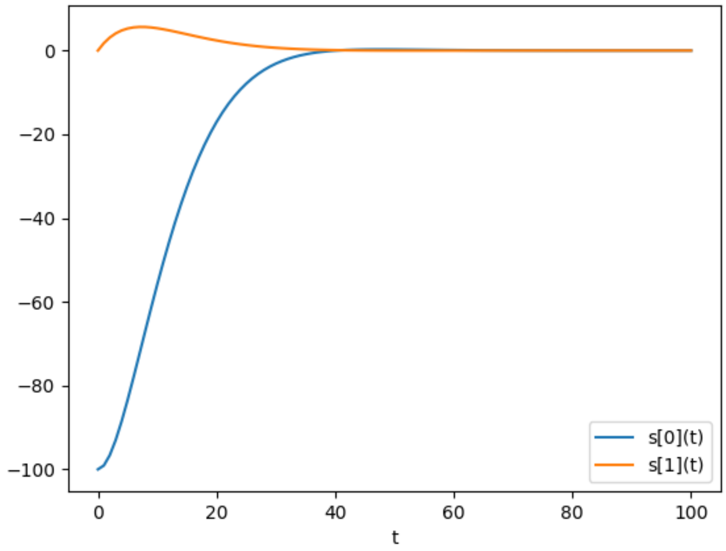


Figure 2: Car model state trajectories with a PD controller using $K_p = 20$ and $K_d = 200$.

```
    der_v = -k / m * x[1] + F / m
    return [der_x, der_v]
```