7.3 Appendix: Feedforward control

In this section we design feedforward control action to add to the closed-loop proportional controller.

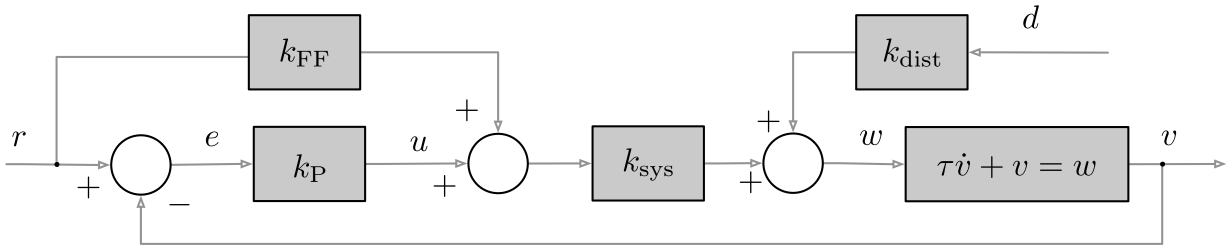

Figure 7.8: Closed-loop P control of the dynamic car velocity model, together with a feedforward control action. Illustration via a block diagram with control block with a feedback loop and a feedforward connection.

- Feedforward control adjusts the input based on (i) a model of the system and (ii) anticipated disturbances, before any error appears and independently of any input.
- Threfore, feedforward control is effective when (i) the model is sufficiently accurate and (ii) disturbances are predictable.

  In other words, the performance of feedforward control depends on proper *calibration*: the model used for the design of the feedforward control should match as much as possible the true system dynamics to avoid under- or over-compensation.
- In practice, feedforward is combined with feedback, which corrects residual errors arising from imperfect calibration or unmodeled effects. In cruise control systems, a feedforward term accounts for known road inclination and simplifyies the task of the feedback loop.

Simulation of proportional feedback + feedforward control for cruise control system

```
import numpy as np; import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
plt.rcParams.update({"text.usetex": True, "font.family": "serif", ...
    "font.serif": ["Computer Modern Roman"] })

# Constants
ksys = 3                    # system gain, we let kdist=ksys
tau = 5                     # system time constant (slow system)
d = 0                       # disturbance
kp = [0.1, 1, 10, 100]      # control gain (multiple values)
kff = .9 / ksys             # feedforward control gain

# Define the ODE for the cruise control system
def cruise_ctrl_ode(t, y, K, tau, ref_speed, d):
    speed = y[0]
    feedback_ctrl = K * (ref_speed - speed)
    feedforward_ctrl = kff * ref_speed
    acceleration = (-speed + ksys * (feedforward_ctrl + ...
        feedback_ctrl + d)) / tau
    dydt = [acceleration]
    return dydt

# Initial conditions: 50 mph.  Time span for simulation
initial_speed = 50;         # initial speed (mph)
ref_speed = 60        # desired speed (mph)
init_cond = [initial_speed]; t_span = (0, 6)

# Create a figure with two subplots
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6.4), sharex=True)
ax1.set_title('Cruise control with disturbance: proportional ...
    controller + feedforward')
colors = ['#752d00', '#a43e00', '#d35000', '#ff6100']

# Solve the ODE and plot the results for each value of K
for i, K in enumerate(kp):
    solution = solve_ivp(cruise_ctrl_ode, t_span, init_cond, ...
        args=(K, tau, ref_speed, d), t_eval=np.arange(0, 6, 0.01), ...
        method='LSODA')
    time = solution.t; speed = solution.y[0]; feedback_ctrl = [K * ...
        (ref_speed - speed[j]) for j in range(len(time))]
    ax1.plot(time, speed, label=f'$k_{{\\mathrm{{p}}}} = {K}$', ...
        color=colors[i]);     ax1.set_ylabel('speed $v(t)$'); ...
        ax1.set_xlim(0, 6);  ax1.set_ylim(35, 65); ax1.grid(True); ...
        ax1.legend();     ax2.plot(time, feedback_ctrl, ...
        label=f'$k_{{\\mathrm{{P}}}} = {K}$', color=colors[i]);     ...
        ax2.set_xlabel('time $t$');  ax2.set_ylabel('control input ...
        $u(t)$'); ax2.set_xlim(0, 6);  ax2.set_ylim(0, 30);  ...
        ax2.grid(True); ax2.legend()

plt.savefig('cruise-control-proportional+feedforward.pdf', ...
    bbox_inches='tight')
```

Listing 7.2: Python script generating Figure 7.9. Available at cruise-control-proportional+feedforward.py

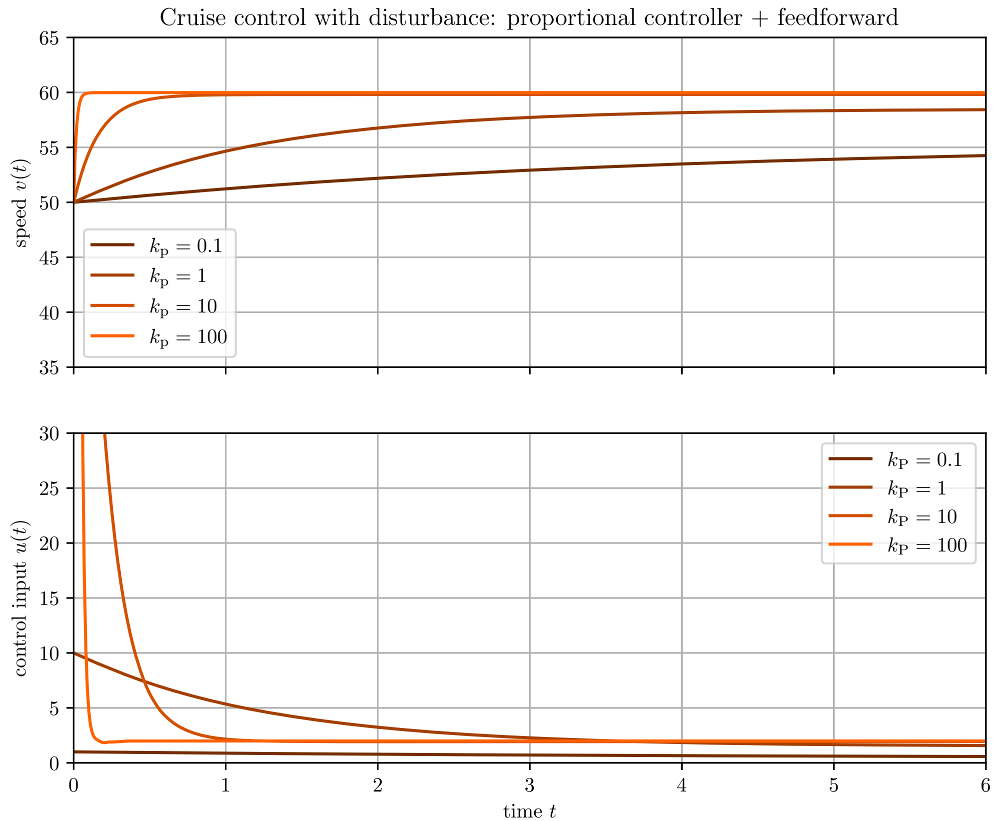

Figure 7.9: Solutions of the cruise control dynamics (7.3): $v(t)$ in the first plot, $u(t)$ in the second plot. The initial velocity is $v(0) = 50$ and the reference velocity is 60.
As before, different values of k_{P} lead to different final values.
Bottom line: The feedforward action helps the behavior considerably, but none of these solution is satisfactory (even without disturbance $d = 0$)