Concepts and Software Design for CPS

Lab 4: Basics of control theory and PID controller design

Tomasz Kloda Gero Schwäricke Debayan Roy

Chair of Cyber-Physical Systems in Production Engineering
Technical University of Munich
Munich, Germany

Lab 4: Overview

Basics on Control Theory

PID Controller design in Simulink

Assignment 4 (due: 17.06.2021)

Basics on Control Theory

Additional Material

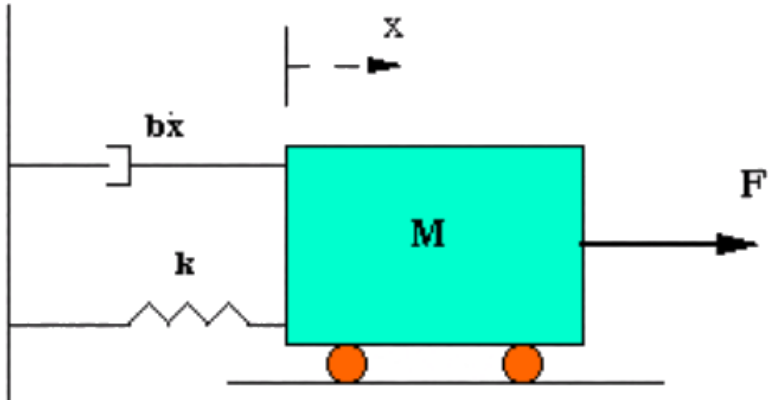
You can find related material at the following sources:

- Lecture 4
- Lecture 5
- Simulink tutorial
- Control Tutorials for MATLAB and Simulink:

  `http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID`

  `http://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=ControlPID`
- Empirical PID gain tuning (Kevin Lynch)

Mass-spring-damper example



m = 1 kg
b = 10 N s/m
k = 20 N/m
The desired
set-point = 1.0

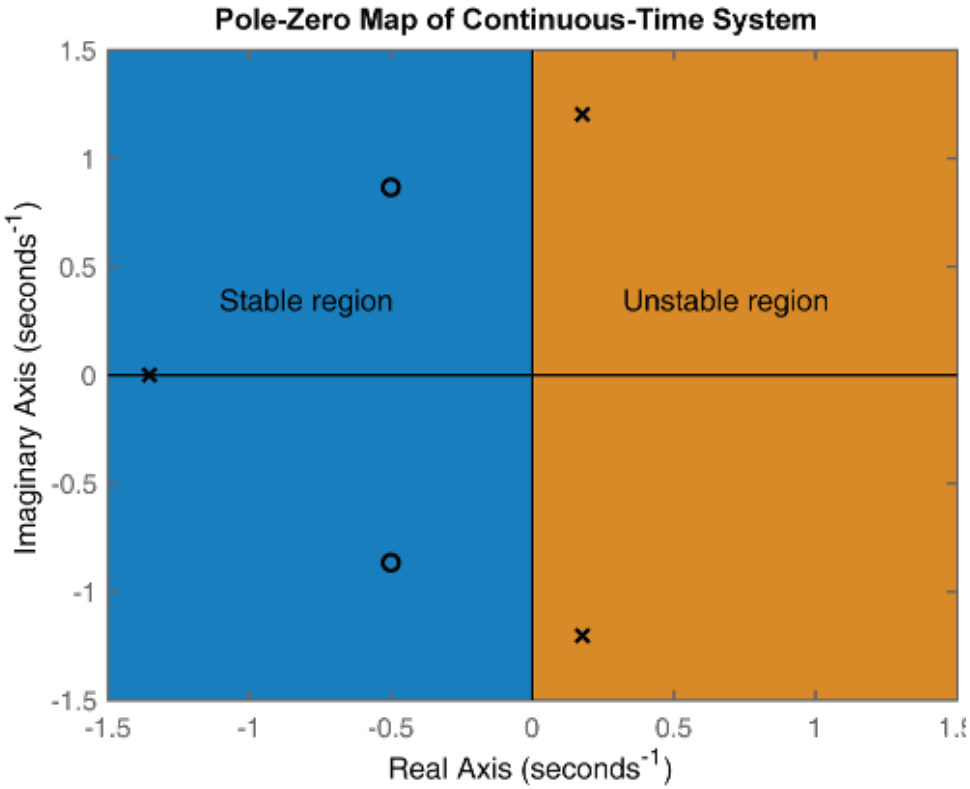
$$F(x) = m\ddot{x} + b\dot{x} + kx$$

$$F(s) = ms^2X(s) + bsX(s) + kX(s) \quad \textit{(Laplace transform)}$$

$$\frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k}$$

For more information please refer to Lecture 4 slide 10 and Control Tutorial for MATLAB and Simulink

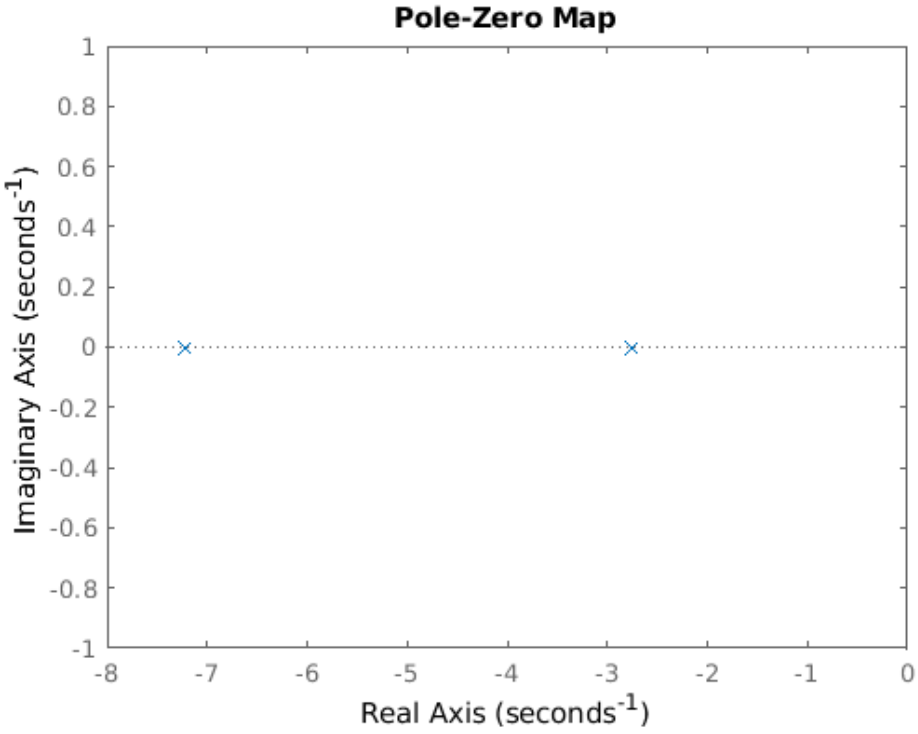
Stability



The poles on the complex s-plane must be in the left-half plane (blue region) to ensure stability.

The system is marginally stable if distinct poles lie on the imaginary axis, that is, the real parts of the poles are zero.

Mass-spring-damper (open loop): Stability

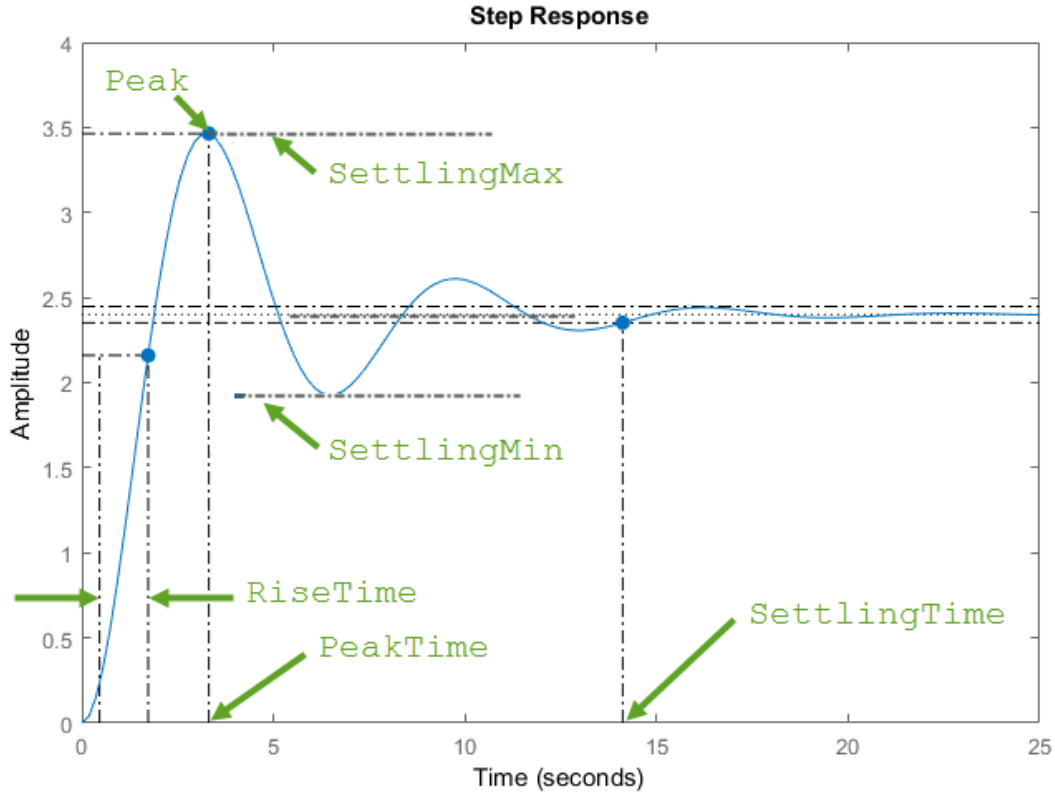



```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
pzmap(P)
```

pzmap creates a pole-zero plot of the continuous or discrete-time dynamic system model

tf Construct transfer function or convert to transfer function

Step Response

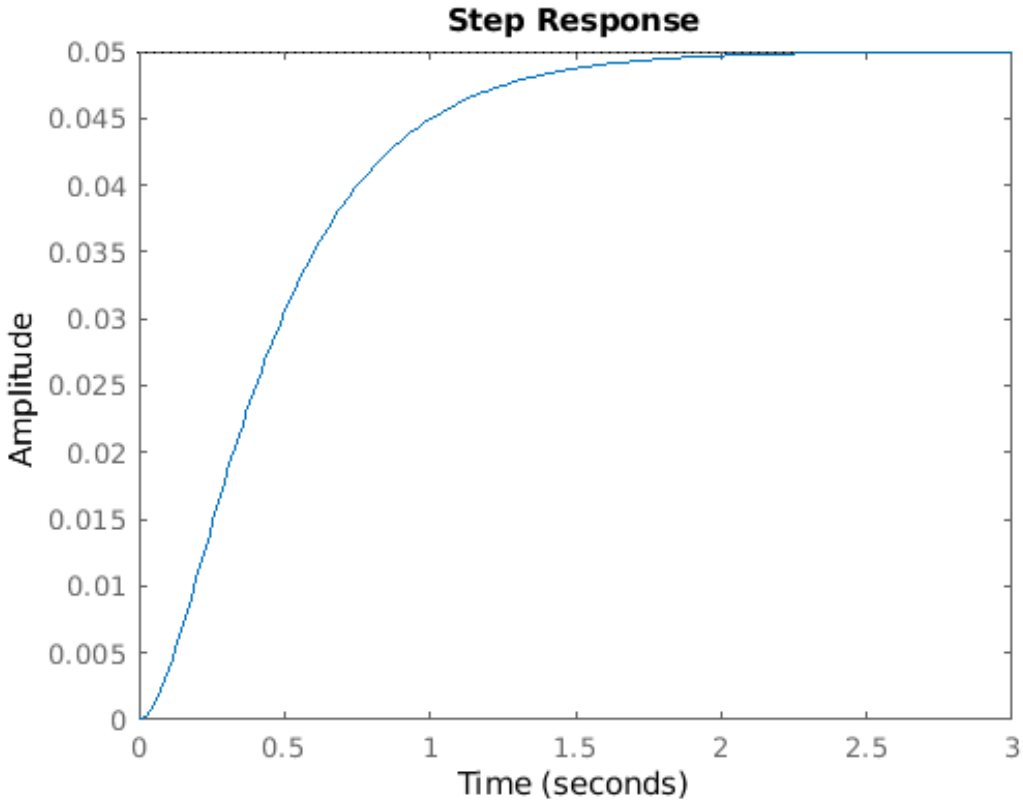


Steady-State Error: the difference between the input (command) and the output of a system in the steady state.

Rise Time: time it takes for the response to rise from 10% to 90% of the steady-state response.

Settling Time: Time it takes for the error between the response and the steady-state final response to fall to within 2% of the final steady-state

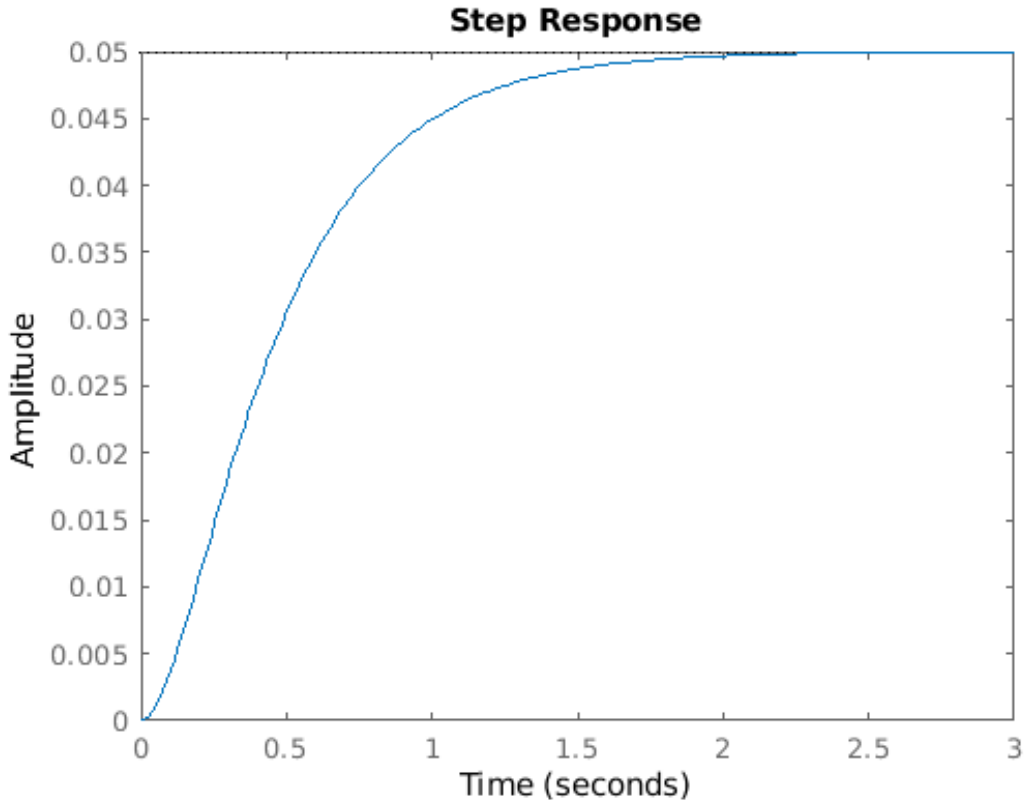
Mass-spring-damper example (open loop): step response




```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
step(P)
```

step Step response of dynamic systems.

Mass-spring-damper example (open loop): step response



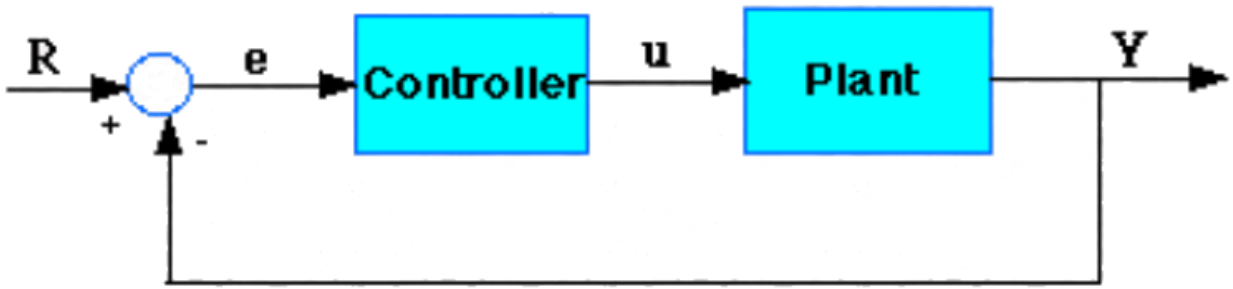
Steady-state error $\approx$ 95 %

The rise time $\approx$ 1.0 s

The settling time $\approx$ 1.5 s

Mass-spring-damper example: PID controller

We will design a PID controller that will reduce the rise time, reduce the settling time, and eliminate the steady-state error for the mass-spring-dumper system.



The feedback controller generates a command signal u based on the error signal e, that is, the difference between the desired value R and the current system output Y.

Mass-spring-damper example: PID controller

We will design a PID controller that will reduce the rise time, reduce the settling time, and eliminate the steady-state error for the mass-spring-dumper system.

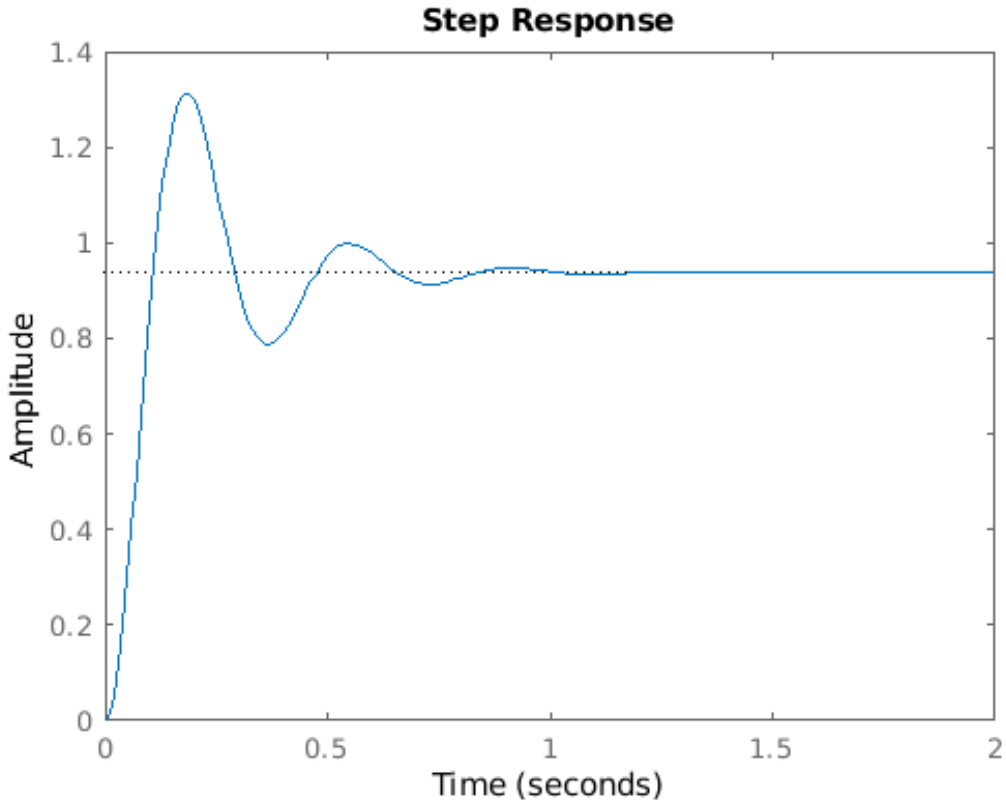
$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t) + K_i \int_0^t e(\tau) d\tau$$

The feedback controller generates a command signal u based on the error signal e, that is, the difference between the desired value R and the current system output Y.

Mass-spring-damper example: PID controller

Proportional control

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
Kp = 300;
C = pid(Kp)
T = feedback(C*P,1)
t = 0:0.01:2;
step(T,t)
```



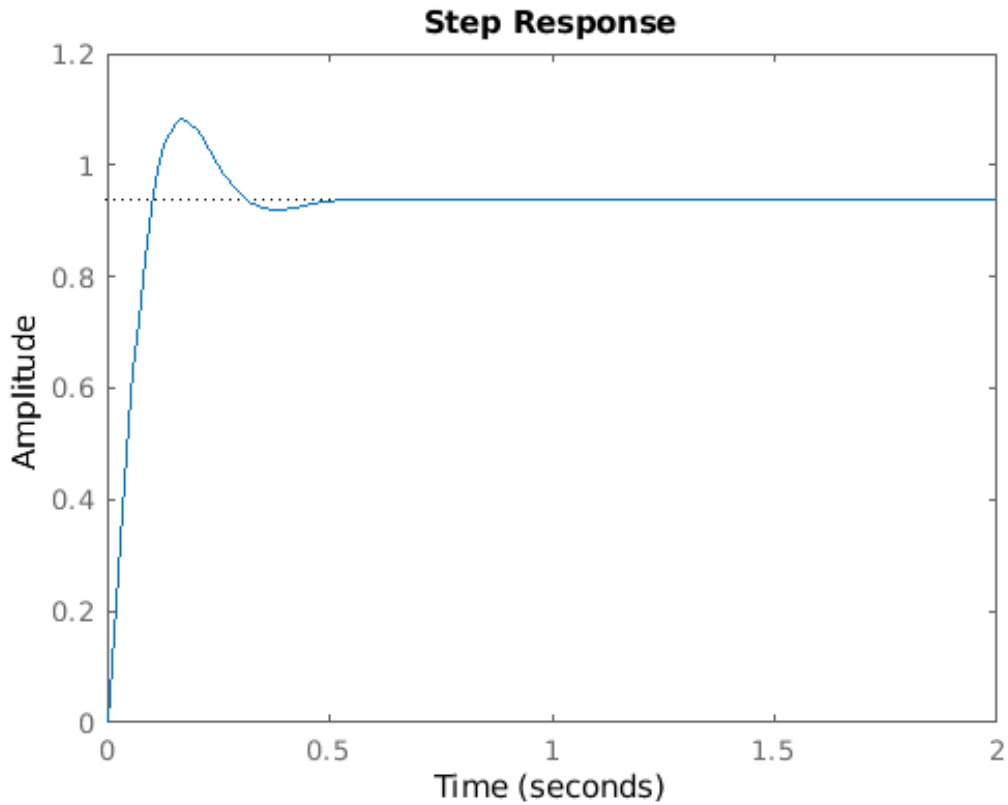
pid create a pid controller

feedback feedback connection of two input/output systems

Mass-spring-damper example: PID controller

Proportional-derivative control

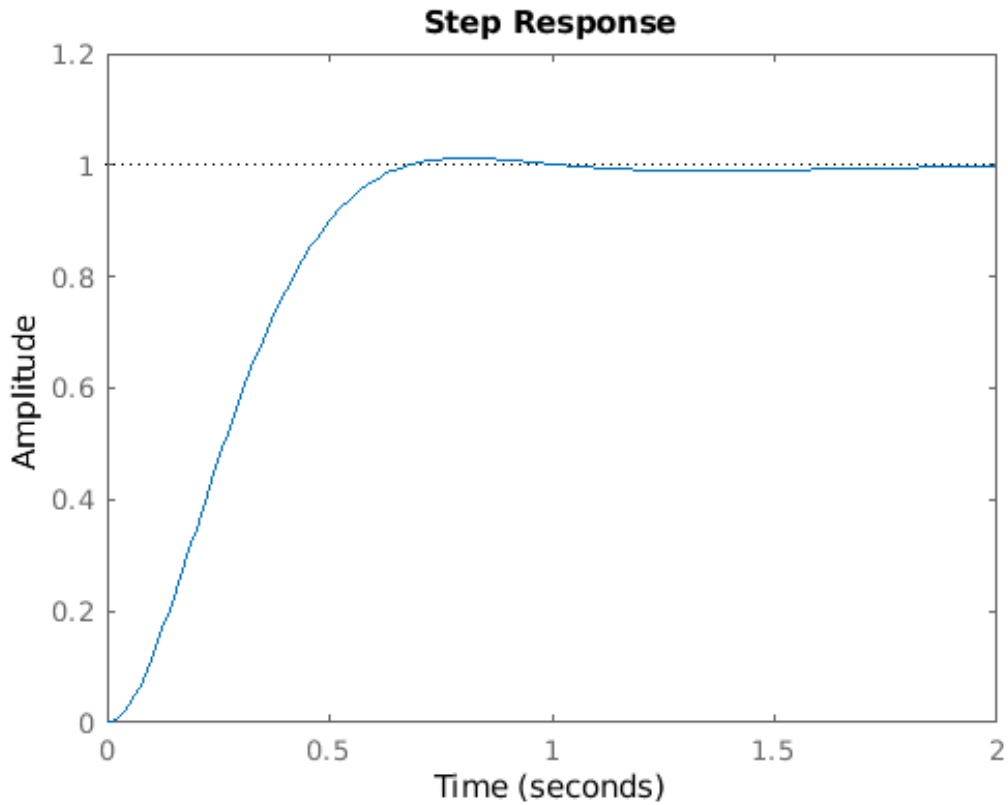
```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
Kp = 300;
Kd = 10;
C = pid(Kp,0,Kd)
T = feedback(C*P,1)
t = 0:0.01:2;
step(T,t)
```



Mass-spring-damper example: PID controller

Proportional-integral control

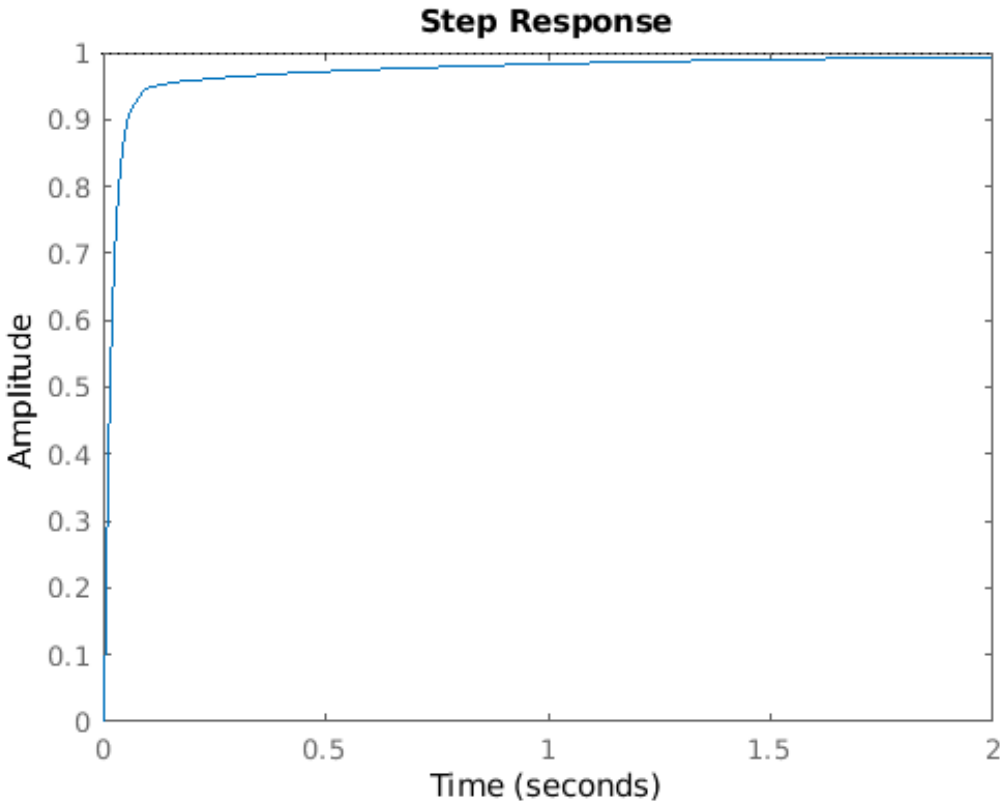
```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
Kp = 30;
Ki = 70;
C = pid(Kp,Ki)
T = feedback(C*P,1)
t = 0:0.01:2;
step(T,t)
```



Mass-spring-damper example: PID controller

Proportional-integral-derivative control

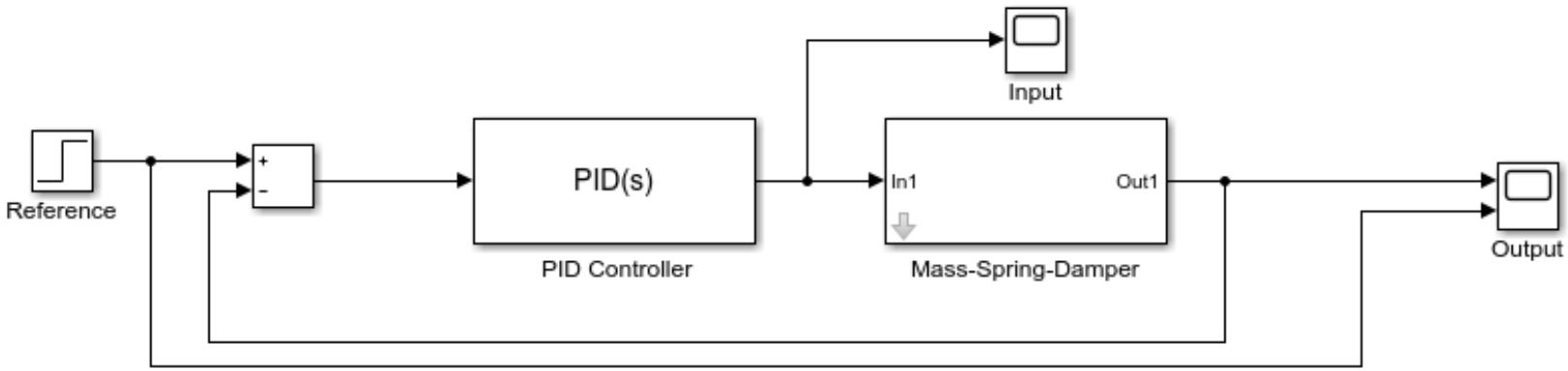
```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
Kp = 350;
Ki = 300;
Kd = 50;
C = pid(Kp,Ki,Kd)
T = feedback(C*P,1);
t = 0:0.01:2;
step(T,t)
```



PID Controller design in Simulink

PID control in continuous-time

The Simulink model for the PID control of the mass-spring damper system is as shown below:



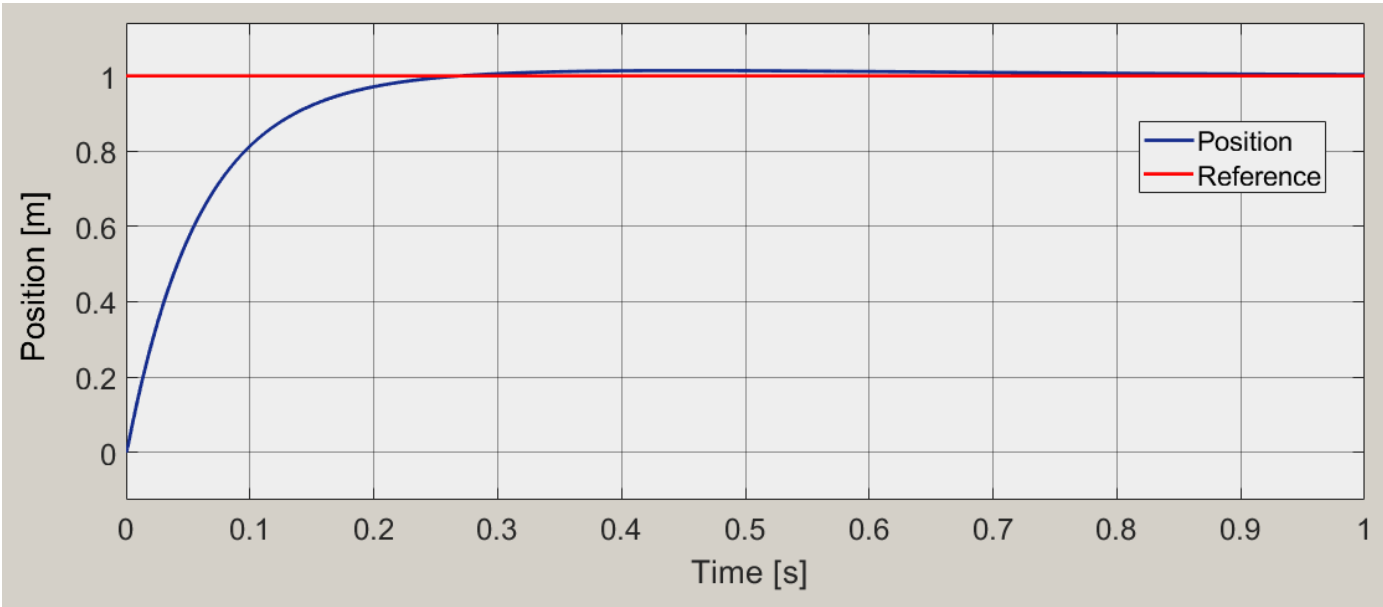
The physical system comprising the mass-spring-damper evolves continuously in time.
Here, we also assume that the control inputs can be updated continuously, i.e., controller evolves continuously. This is implied by 's' in **PID(s)** in the above model.

PID control in continuous-time

We tune the continuous-time PID controller to get the following gain values:

$$K_p = 163.23; \qquad K_I = 394.07; \qquad K_D = 16.8;$$

And we get the following response.



Assignment 4 (due: 17.06.2021)

Task 1: DC-Motor Control in Matlab

Let us consider a DC motor speed control system for which the plant dynamics is given by the following differential equations:

$$J\frac{d^2\theta}{dt} + b\frac{d\theta}{dt} = K_t i$$
$$L\frac{di}{dt} + Ri = V - K_e\frac{d\theta}{dt}$$

Here, θ is the angular position, i is the motor current and V is the terminal voltage of the motor. The output of the system is the *angular speed* of the motor, i.e., y=$\frac{d\theta}{dt}$. And the control input is the terminal *voltage* of the motor, i.e., $u = V$.

Task 1: DC-Motor Control in Matlab

The constants are given as follows:

$J = 0.01\,\mathrm{kgm}^2$ (moment of inertia) 转动惯量

$b = 0.1\,\mathrm{Nms}$ (motor viscous friction constant) 马达粘性摩擦常量

$K_t = 0.01\,\mathrm{Nm/A}$ (motor torque constant) 马达转矩常量

$L = 0.05\,\mathrm{H}$ (motor inductance) 马达电感

$R = 0.75\,\Omega$ (motor resistance)

$K_e = 0.01\,\mathrm{V/rad/s}$ (electromotive force constant) 电动势常数

Task 1: DC-Motor Control in Matlab

- Find the transfer function of the DC-Motor system
- Prove that the open-loop system is stable
- Find the step-response of the open-loop system
- Design the following controllers:
  - proportional (steady-state error below 5%)
  - proportional-derivative (reduce the overshoot)
  - proportional-integral-derivative (improve at least one characteristic of PD)
- Compare the controllers (rise time, overshoot, steady-state error[1])

稳态误差

Parameters of each controller can be assigned arbitrarily (e.g., you can use different values of proportional gain for each controller).

Please provide the Matlab code (remember to adjust the time-axis on the plots). Consider continuous-time only.

[1] Right click on the plot and select the Characteristics

Task 2: PID Control in C - Drone Control

Implementation in C

Implement a PID controller in C and apply it in the provided simulation.

Your program should

- implement a PID controller,
- include a main function that starts the simulation,
- in a loop
  - advances the simulation time (in seconds),
  - fetches the current drone height (in meters),
  - update the drone acceleration (`MIN_ACCELERATION` to `MAX_ACCELERATION`) and
- uses the PID controller for calculating the throttle from the drone height.

Check out the `drone_simulation.h` on how to interface with the simulation.

Task 2: PID Control in C - Drone Control

PID Tuning

As start values for the PID controller you can use:
$K_p = 2.0, K_d = 0.2, K_i = 0.5$.

Use Matlab to visualize the PID controller's performance (from the flight log) and try to tune the gains on a 0-to-1 step response to achieve

- no oscillation,
- no overshoot, and
- a settling time ≤ 2 seconds.

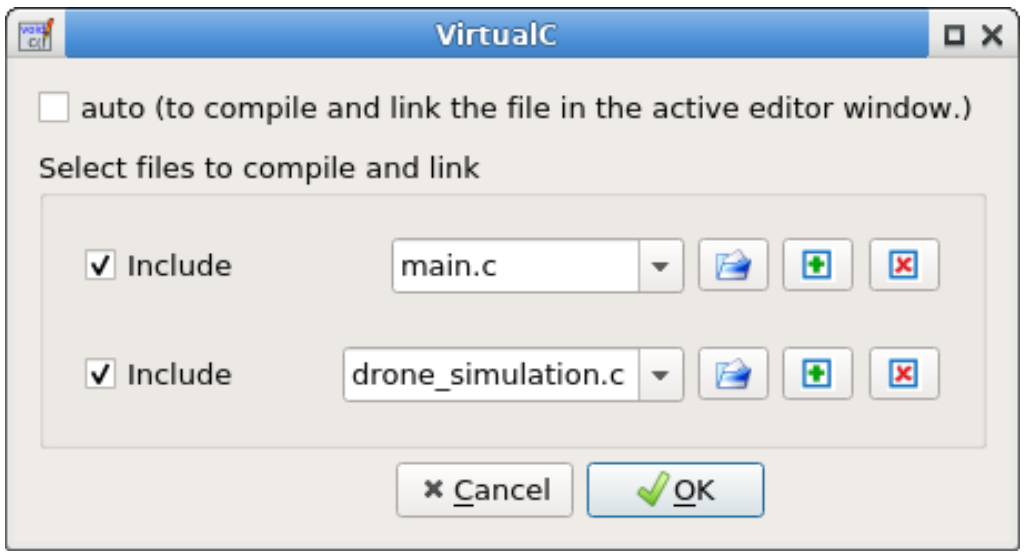
To find the first/last index in a vector that fulfills a condition (e.g., greater than some value), you can use:

```
first_index = find( vector > some_value, 1 )
last_index = find( vector > some_value, 1, 'last' )
```

Task 2: PID Control in C - Drone Control

Multiple Source Files in VirtualC

As you learned in Lab 1, when compiling larger projects we need to provide a list of all source files to the compiler (remember header files (`*.h`) are only included by the preprocessor and not compiled on their own). In VirtualC this can be done in `Build > Build Options`:



Next Lab: Lab 5 on 17.06.2021

Public holiday on 03.06.2021
Q&A Meetings on 10.06.2021

Next Lab: Lab 5 on 17.06.2021

- Topic: POSIX
- **Feedback meeting: Assignment 4**
- Hand out Assignment 5 (POSIX)