TECHNISCHE UNIVERSITÄT CHEMNITZ

Professur Betriebssysteme

Assignment 2

SWES WS 15/16

Jafar Akhundov, Dr. Peter Tröger

General Rules

All five assignments in the course sum up to 50 points. You need to have at least 25 points to be allowed to enter the exam.

The solutions have to be submitted at:

`https://osg.informatik.tu-chemnitz.de/submit/`

Re-uploads are possible until the shown deadline. Your submitted solution must be a single PDF file, containing the answers for all tasks.

Introduction

Imagine that you have to design the cruise control functionality for a modern car. It allows the driver to set a desired velocity, which is then automatically maintained by the system.

Engine power control, and therefore car speed regulation, works by setting the angle of the throttle body regulating the air inflow [1]. The current speed is computed by constantly measuring the angular velocity of the car wheels.

Task 1 (1 point)

Describe the cruise control functionality in terms of control systems theory:

- What is the set point? Where does it come from?
- What is the control variable? Who reads it? Who writes it?
- What is the controlled variable? Who reads it? Who writes it?
- What is the measured variable? Who reads it? Who writes it?

[1] `https://en.wikipedia.org/wiki/Throttle#Throttle_body`

Task 2 (1 point)

Draw the feedback diagram for the cruise control functionality, containing the following parts:

- Cruise controller, the software evaluating sensors and triggering actuators
- Car engine, trottle body, and wheels
- Air Intake / Pressure
- Environmental disturbance (wind, slope, etc)
- Desired velocity, as set by the driver
- Angular velocity, as measured by wheel-attached sensors
- Actual velocity

Task 3 (4 points)

For this task, we rely on RStudio[2] and R[3]. Both tools together allow the creation of interactive PID controller simulations. They can be installed, free of charge, on Windows, MacOS X or Linux.

After loading an *.rmd* script, you can run it by clicking on the 'Run Document' button. Running a script may demand the (automatic) installation of package dependencies in RStudio.

The following script[4] is a broken simulation of a PID controller in RStudio.

~~~
---
runtime: shiny
---
```{r, echo=FALSE}
inputPanel(
  sliderInput("P", label = "P-term␣adjustment:", min = 0, max = 10,
                   value = 2, step = 0.01),
  sliderInput("I", label = "I-term␣adjustment:", min = 0, max = 20,
                   value = 0.5, step = 0.01),
  sliderInput("D", label = "D-term␣adjustment:", min = 0, max = 1,
                   value = 1, step = 0.001)
)
############## Your changes start here! (Block A) ##############
pv = function(pv.prev, u, tt) {
  out = pv.prev + 2                # linear growth
  out = out - 0.1*u                # the control response
  if (out < 0) out = 0             # keep values positive
  return(out)
}
############## Your changes end here!             ##############
############## Your changes start here! (Block B) ##############
#pv = function(pv.prev, u, tt) {
#  out = pv.prev*1.1 + 3           # exponential growth + linear growth
#  out = out - 0.1*u               # the control response
#  if (out < 0) out = 0            # keep values positive
~~~

[2]https://www.rstudio.com/products/rstudio/download/

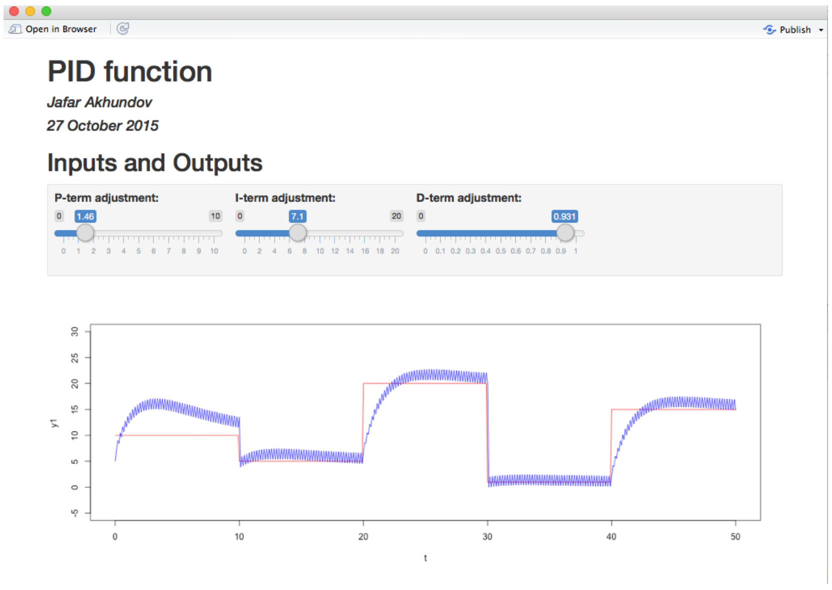
[3]https://cran.rstudio.com

[4]The script is available at https://osg.informatik.tu-chemnitz.de/lehre/swes/PID_broken.Rmd

```
#  return(out)
#}
############## Your changes end here!             ##############
############## Your changes start here! (Block C) ##############
#pv = function(pv.prev, u, tt) {
#  out = pv.prev*1.15 + 0.8         # exponential growth + linear growth
#  out = out - 0.1*u                # the control response
#  out = out + .5*rnorm(length(tt), mean = 0, sd = 1)  # little noise
#  if (out < 0) out = 0             # keep values positive
#  return(out)
#}
############## Your changes end here!             ##############
dt = 0.1
t <- seq(0, 50, by=dt)
PV = U = E = EI = ED = rep(0, length(t))
PV[1] = 5
renderPlot({
  Kp = input$P
  Ti = input$I
  Td = input$D
  SP = rep(10, length(t))
  SP[which(t >= 10)] = 5
  SP[which(t >= 20)] = 20
  SP[which(t >= 30)] = 1
  SP[which(t >= 40)] = 15
  for (k in 2:length(t)) {
    PV[k] = pv(PV[k-1], U[k-1], t[k])
    E[k] = PV[k] - SP[k]
    ############## Your changes start here! (Block D)  ##############
    EI[k] = EI[k+1] + E[k]*dt                          # integral
    ED[k] = (E[k-1] + E[k])/dt                         # derivative
     U[k] = Kp/(E[k] + (1*Ti)*sum(E*dt) + Td + ED[k])  # control output
    ############## Your changes end here!              ##############
    if (U[k] <= 0) U[k] = 0
  }
  y1 = SP
  y2 = PV
  plot(t,y1,type="l",col="red",ylim = c(-5,30))
  lines(t,y2,col="blue")
})
```
```

The code contains variables for the process (PV), the control output (U), the error (E), the integral error (EI), the derivative error (ED), the set points (SP), the time (t), the time step (dt) and the control coefficients (Kp,Ti,Td),

Answer the following question in your solution document:

(a) What corrections must be done in Block D to implement a correct PID controller? You can check that your corrections are valid by comparing the resulting simulation with the screenshot given in Figure 1.

(b) Make sure that Block A is uncommented and Block B and C are commented out. What are P, I and D values so that overshoot, stabilization time and noise are minimized?

(c) Make sure that Block B is uncommented and Block A and C are commented out.
```

What are P, I and D values so that overshoot, stabilization time and noise are minimized?

(d) Make sure that Block C is uncommented and Block A and B are commented out.
What are P, I and D values so that overshoot, stabilization time and noise are minimized?



PID function

*Jafar Akhundov*

*27 October 2015*

Inputs and Outputs

P-term adjustment: 1.46 (0–10)

I-term adjustment: 7.1 (0–20)

D-term adjustment: 0.931 (0–1)

Figure 1: Correct PID Simulator in RStudio