Control Theory Assignment № 3

Danat Ayazbayev, BS-18-03 01.04.2020

1. Selecting Variant

Name: Danat
Email: d.ayazbaev@innopolis.university
Variant: g

2. In this task use python for calculations

A. Design PD-controller that tracks time varying reference states i.e. $[x^*(t), \dot{x}^*(t)]$ **as closely as possible. Test your controller on different trajectories, at least two. System:** $\ddot{x} + \mu\dot{x} + kx = u$**, see variants below.**

My system: $\ddot{x} + 7\dot{x} + 25x = u$

Code:

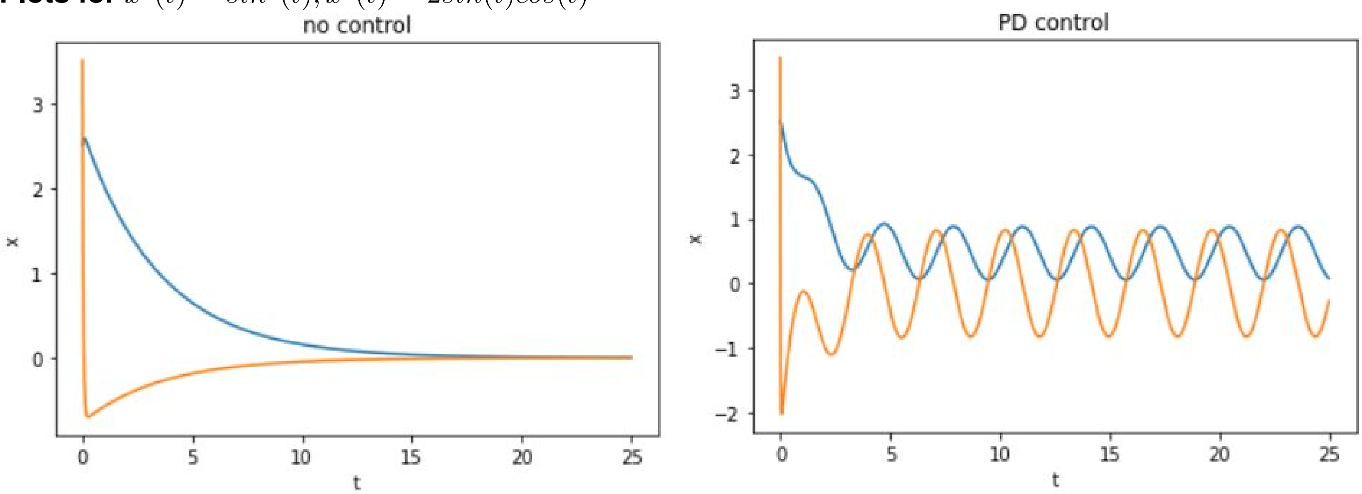
```python
#importing needed libraries
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.integrate import odeint

time = np.linspace(0, 25, 10000) #defining in which bounds to plot
kp = 100 #proportional control coefficient
kd = 100 #derivative control coefficient
k = 7     #given constant
mu = 25   #given constant
init = [2.5, 3.5] #you can write here any initial conditions

def ref(t):  #desired reference state
  return math.sin(t) * math.sin(t) #function of desired reference state
def dref(t): #rate of change of desired reference state
  return 2.0 * math.sin(t) * math.cos(t) #derivative of previous function
def no_control(x, t): #model without any control
  y = x[0]
  dy = x[1]
  xdot = [[], []]
  xdot[0] = dy
  xdot[1] = -mu * dy - k * y
```

```
  return xdot
def pd_control(x, t): #model with PD control
  err = ref(t) - x[0] #error
  derr = dref(t) - x[1] #rate of change of error
  u = kd * derr + kp * err
  y = x[0]
  dy = x[1]
  xdot = [[], []]
  xdot[0] = dy
  xdot[1] = u - mu * dy - k * y
  return xdot

plt.subplot(221) #positioning the plotted graph
plt.title("no control") #title of graph
plt.xlabel("t")   #what means x-axis (time)
plt.ylabel("x")   #what means y-axis (x)
plt.plot(time, odeint(no_control, init, time)) #plotting ODE solution for model without control
plt.subplot(224)
plt.title("PD control")
plt.xlabel("t")
plt.ylabel("x")
plt.plot(time, odeint(pd_control, init, time)) #plotting ODE solution for model with PD control
```

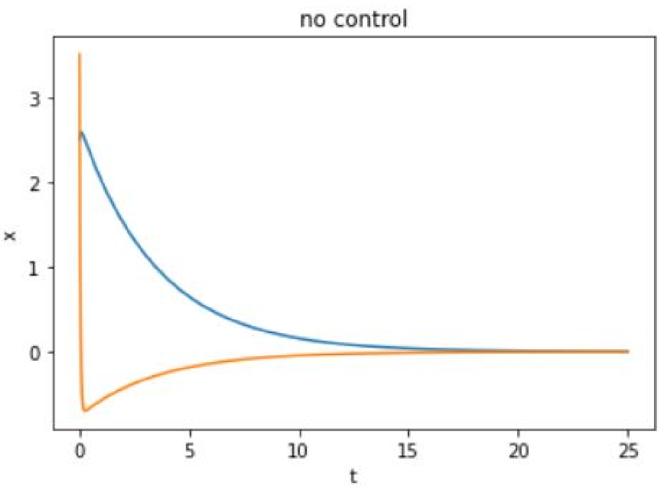
Notes

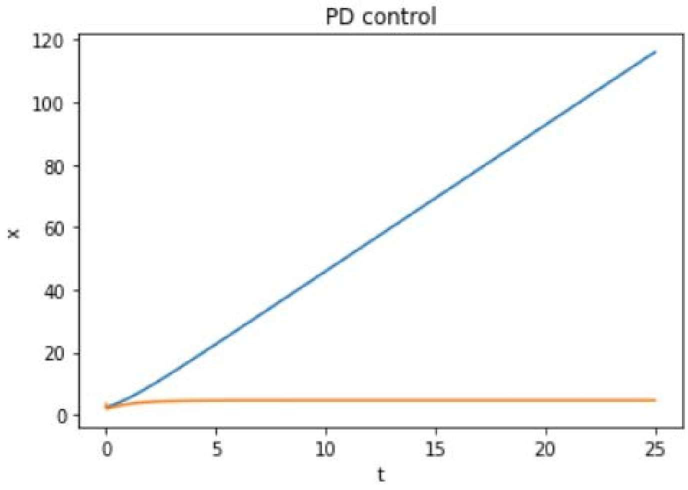
I chose $kp = 100$, $kd = 100$.

Plots for $x^*(t) = sin^2(t), \dot{x}^*(t) = 2sin(t)cos(t)$



Plots for $x^*(t) = 5t, \dot{x}^*(t) = 5$





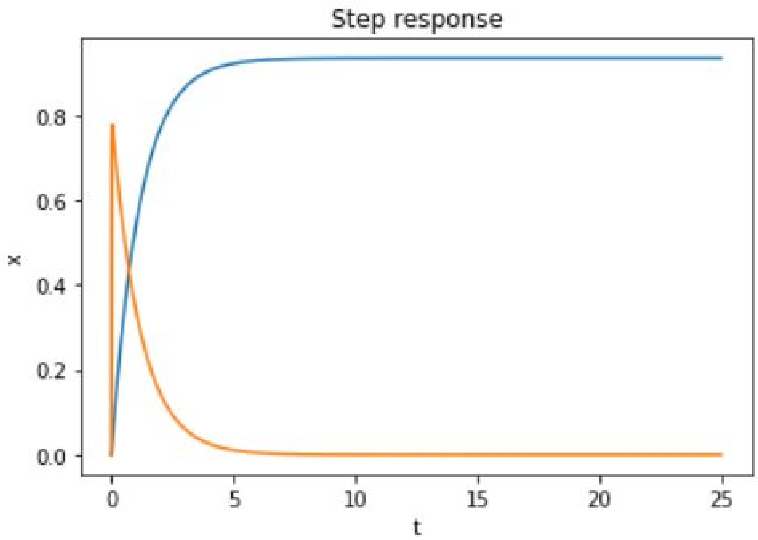
B. Tune controller gains k_p and k_d. Find gains that provide no oscillations and no overshoot. Prove it with step input.

I chose $k_p = 100$, $k_d = 100$.

```python
#importing needed libraries
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.integrate import odeint

time = np.linspace(0, 25, 10000) #defining in which bounds to plot
kp = 100 #proportional control coefficient
kd = 100 #derivative control coefficient
k = 7     #given constant
mu = 25   #given constant
init = [0, 0] #zero initial conditions

def ref(t):  #desired reference state (step function)
  if t == 0:
    return 0.5
  elif t < 0:
    return 0
  else:
    return 1
def dref(t): #rate of change of desired reference state
  return 0   #derivative of previous function
def step(x, t): #function for defining ODE for step response
  err = ref(t) - x[0] #error
  derr = dref(t) - x[1] #rate of change of error
  u = kd * derr + kp * err #control
  y = x[0]
```

```
  dy = x[1]
  xdot = [[], []]
  xdot[0] = dy
  xdot[1] = u - mu * dy - k * y
  return xdot
plt.subplot(111)
plt.title("Step response")
plt.xlabel("t")
plt.ylabel("x")
plt.plot(time, odeint(step, init, time)) #solving ODE and plotting step response
```

No overshooting and no oscillations proof:



C. Prove that controlled oscillator dynamics is stable for your choice of k_p and k_d.

$$\begin{bmatrix}\ddot{x}\\ \dot{x}\end{bmatrix} = \begin{bmatrix}-7 & -25\\ 1 & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ x\end{bmatrix} + \begin{bmatrix}k_d & k_p\\ 0 & 0\end{bmatrix}\begin{bmatrix}\dot{x}^* - \dot{x}\\ x^* - x\end{bmatrix}$$

$$\begin{bmatrix}\ddot{x}\\ \dot{x}\end{bmatrix} = \begin{bmatrix}-7 & -25\\ 1 & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ x\end{bmatrix} + \begin{bmatrix}k_d & k_p\\ 0 & 0\end{bmatrix}\begin{bmatrix}\dot{x}^*\\ x^*\end{bmatrix} - \begin{bmatrix}k_d & k_p\\ 0 & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ x\end{bmatrix}$$

$$\begin{bmatrix}\ddot{x}\\ \dot{x}\end{bmatrix} = \begin{bmatrix}-7 - k_d & -25 - k_p\\ 1 & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ x\end{bmatrix} + \begin{bmatrix}k_d & k_p\\ 0 & 0\end{bmatrix}\begin{bmatrix}\dot{x}^*\\ x^*\end{bmatrix}$$

$$A = \begin{bmatrix}-7 - k_d & -25 - k_p\\ 1 & 0\end{bmatrix}$$

Let's find eigenvalues of A matrix

$$det(A - \lambda I) = 0$$

$$(-7 - k_d - \lambda)(-\lambda) + 25 + k_p = 0$$

$$7\lambda + k_d\lambda + \lambda^2 + 25 + k_p = 0$$

$$\lambda^2 + \lambda(7 + k_d) + 25 + k_p = 0$$

$$\lambda_1 + \lambda_2 = -(7 + k_d), \lambda_1\lambda_2 = 25 + k_p$$

If system if stable, then

$$\lambda_1 < 0 \text{ and } \lambda_2 < 0$$

$$-7 - k_d < 0, k_d > -7$$

$$25 + k_p > 0, k_p > -25$$

As I chose k_d = 100, k_p = 100, inequalities holds, and system is stable

E. Implement a PI/PID controller for the system: $\ddot{x} + \mu\dot{x} + kx + 9.8 = u$. Test your controller on different trajectories, at least two.

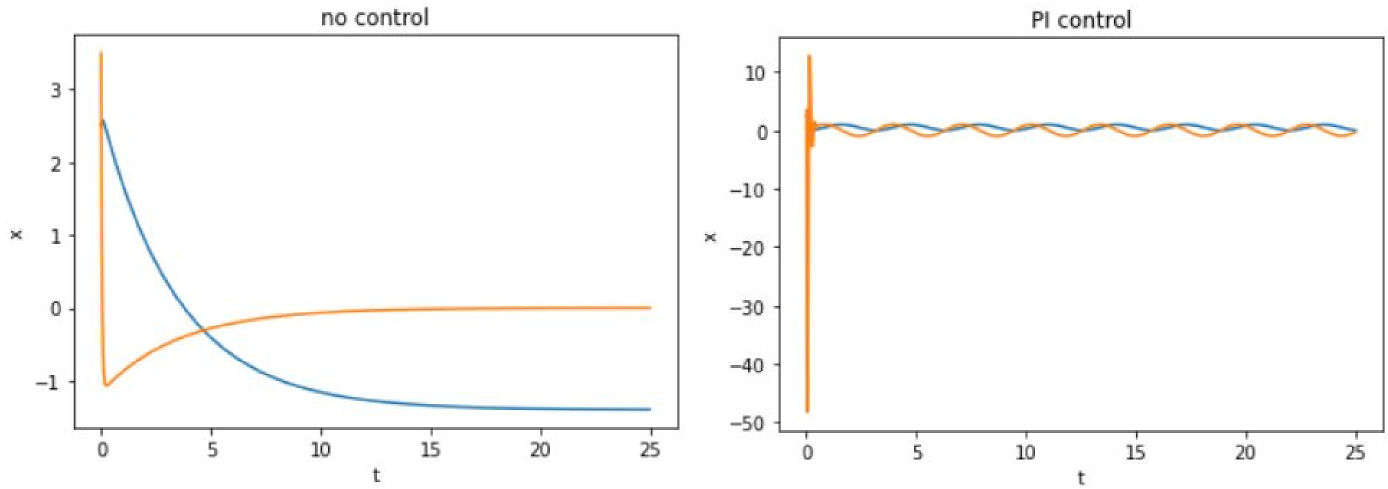
```
#importing needed libraries
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import math
from scipy.integrate import odeint

time = np.linspace(0, 25, 10000) #defining in which bounds to plot
kp = 1000 #proportional control coefficient
ki = 1 #integral control coefficient
k = 7     #given constant
mu = 25   #given constant
init = [2.5, 3.5] #you can write here any initial conditions

def ref(t):  #desired reference state
  return math.sin(t)*math.sin(t) #function of desired reference state
def no_control(x, t): #model without any control
  y = x[0]
  dy = x[1]
  xdot = [[], []]
  xdot[0] = dy
  xdot[1] = -mu * dy - k * y - 9.8
  return xdot
def pi_control(x, t): #model with PD control
  err = ref(t) - x[0] #error
  diff = lambda t: ref(t) - x[0]
  ierr = integrate.quad(diff, 0, t)[0] #integral of error
  u = ki * ierr + kp * err #control
  y = x[0]
  dy = x[1]
  xdot = [[], []]
  xdot[0] = dy
  xdot[1] = u - mu * dy - k * y - 9.8
```

```
    return xdot

plt.subplot(221) #positioning the plotted graph
plt.title("no control") #title of graph
plt.xlabel("t")   #what means x-axis (time)
plt.ylabel("x")   #what means y-axis (x)
plt.plot(time, odeint(no_control, init, time)) #plotting ODE solution for model without control

plt.subplot(224)
plt.title("PI control")
plt.xlabel("t")
plt.ylabel("x")
plt.plot(time, odeint(pi_control, init, time)) #plotting ODE solution for model with PI control
```

Notes

I chose $k_p = 1000$, $k_i = 1$

Plots with desired reference state $sin^2(t)$



Plots with desired reference state $5t$