Nama Anggota Kelompok

Salma Fathiyatur Rizky Munir 20081010025 Farrel Adel Mohammad 20081010138 Sabrina Laila Sari 20081010224 Muhammad Abi Prakosa 20081010232

Penjelasan Kode

Berikut adalah ringkasan singkat tentang kode tersebut:

Fungsi process(y, t, u): Fungsi ini mewakili model matematis dari suatu sistem yang akan dikendalikan dengan PID. Model ini merupakan model proses dengan respons yang ditunda, di mana suatu konstanta proporsional (Kp) dan waktu respons (taup) digunakan untuk menghitung turunan dari variabel terkendali (y). Fungsi ini mengembalikan turunan dari y terhadap waktu (dy/dt).

Fungsi pidPlot(Kc, tauI, tauD): Fungsi ini mengimplementasikan kontrol PID untuk sistem yang diberikan. Fungsi ini menggunakan pendekatan numerik untuk menghitung respons sistem terhadap kontrol PID. Respons tersebut kemudian diplot dalam beberapa grafik yang mencakup setpoint, variabel proses, komponen PID, dan keluaran kontrol. Fungsi ini juga menerima tiga argumen, yaitu Kc (konstanta proporsional), tauI (konstanta waktu integral), dan tauD (konstanta waktu derivatif).

Fungsi wg.interact(pidPlot, Kc=Kc_slide, tauI=tauI_slide, tauD=tauD_slide): Fungsi ini menggunakan pustaka ipywidgets untuk membuat kontrol geser (slider) yang berinteraksi dengan fungsi pidPlot. Slider tersebut memungkinkan pengguna untuk mengubah nilai Kc, tauI, dan tauD secara interaktif, dan menampilkan respons PID yang diperbarui saat slider digerakkan.

In [4]:
```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.integrate import odeint
import ipywidgets as wg
from IPython.display import display
n = 100 # time points to plot
tf = 20.0 # final time
SP_start = 2.0 # time of set point change

def process(y,t,u):
    Kp = 4.0
    taup = 3.0
    thetap = 1.0
    if t<(thetap+SP_start):
        dydt = 0.0  # time delay
    else:
        dydt = (1.0/taup) * (-y + Kp * u)
    return dydt

def pidPlot(Kc,tauI,tauD):
    t = np.linspace(0,tf,n) # create time vector
    P= np.zeros(n)          # initialize proportional term
    I = np.zeros(n)         # initialize integral term
    D = np.zeros(n)         # initialize derivative term
    e = np.zeros(n)         # initialize error
    OP = np.zeros(n)        # initialize controller output
    PV = np.zeros(n)        # initialize process variable
    SP = np.zeros(n)        # initialize setpoint
    SP_step = int(SP_start/(tf/(n-1))+1) # setpoint start
    SP[0:SP_step] = 0.0     # define setpoint
    SP[SP_step:n] = 4.0     # step up
    y0 = 0.0                # initial condition
    # loop through all time steps
    for i in range(1,n):
        # simulate process for one time step
        ts = [t[i-1],t[i]]         # time interval
        y = odeint(process,y0,ts,args=(OP[i-1],))  # compute next step
        y0 = y[1]                  # record new initial condition
        # calculate new OP with PID
        PV[i] = y[1]               # record PV
        e[i] = SP[i] - PV[i]       # calculate error = SP - PV
        dt = t[i] - t[i-1]         # calculate time step
        P[i] = Kc * e[i]           # calculate proportional term
        I[i] = I[i-1] + (Kc/tauI) * e[i] * dt  # calculate integral term
        D[i] = -Kc * tauD * (PV[i]-PV[i-1])/dt # calculate derivative term
        OP[i] = P[i] + I[i] + D[i] # calculate new controller output

    # plot PID response
    plt.figure(1,figsize=(15,7))
    plt.subplot(2,2,1)
    plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')
    plt.plot(t,PV,'r:',linewidth=2,label='Process Variable (PV)')
    plt.legend(loc='best')
    plt.subplot(2,2,2)
    plt.plot(t,P,'g.-',linewidth=2,label=r'Proportional = $K_c \; e(t)$')
    plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\frac{K_c}{\tau_I} \int_{i=0}^{n_t} e(t) \; dt $')
    plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c \tau_D \frac{d(PV)}{dt}$')
    plt.legend(loc='best')
    plt.subplot(2,2,3)
    plt.plot(t,e,'m--',linewidth=2,label='Error (e=SP-PV)')
    plt.legend(loc='best')
    plt.subplot(2,2,4)
    plt.plot(t,OP,'b--',linewidth=2,label='Controller Output (OP)')
    plt.legend(loc='best')
    plt.xlabel('time')

Kc_slide = wg.FloatSlider(value=0.1,min=-0.2,max=1.0,step=0.05)
tauI_slide = wg.FloatSlider(value=4.0,min=0.01,max=5.0,step=0.1)
tauD_slide = wg.FloatSlider(value=0.0,min=0.0,max=1.0,step=0.1)
wg.interact(pidPlot, Kc=Kc_slide, tauI=tauI_slide, tauD=tauD_slide)
```

Out[4]: <function __main__.pidPlot(Kc, tauI, tauD)>

In [ ]: