5elen018w_tutorial7_2025_code

December 12, 2024

5ELEN018W - Tutorial 7 2025 Solutions

Exercise 2

[ ]:
```python
import matplotlib.pyplot as plt

# Initialisation
v = 0  # current speed initialised to 0
previous_v = 0  # the speed at the previous time step
dt = 0.001  # time step for the simulation


''' Implementnts the dynamic system (plant) - system input is action u
    and the method returns the output of the plant '''
def plant(action_u):
    # m \dot{v} + b v  = u
    # m=1000, b=50, u = 500
    m = 1000.0
    b = 50

    v_dot = (action_u - b*v)/m

    new_speed = v + v_dot*dt
    return new_speed


# open a file for writing
pw = open('myfile.txt', 'w')

start_time = 0
end_time = 10.0

current_time = start_time

v_ref = 10  # the desired speed

K_p = 800  # proportional gain
K_i = 40    # integral gain
```

```python
K_d = 40   # derivative gain

previous_error = 0
integral = 0

# list containing time for the simulation
time = []

# list containing speed for the simulation
speed = []

#   simulate the system operation from the beginning till
#   the end of the simulation
while current_time <= end_time:
    error = v_ref - v

    # I(ntegral) component of the PID controller
    integral = integral + error*dt

    # D(erivative) component of the PID controller
    deriv = (error - previous_error)/dt

    # the output (action) of the PID controller
    action = K_p*error + K_i*integral + K_d*deriv

    # remember the last error when the previous action
    # was applied to the plant
    previous_error = error

    # apply the new action to the plant to calculate
    # the new (current) speed
    v = plant(action)

    print(f"Time: {current_time} Action: {action}, Speed={v}")
    pw.write(f'{v} {current_time}\n')

    # save v and current_time in the corresponding lists
    time.append(current_time)
    speed.append(v)

    # advance the time
    current_time += dt


pw.close()

# plot speed vs time
```

```
plt.plot(time, speed)
plt.ylim(0, 10)  # set the limits of the range for the y-axis
plt.show()  # display the plot
```

Exercise 3

[ ]:
```
import matplotlib.pyplot as plt

# Initialisation
v = 0  # current speed initialised to 0
x = 0 # initial position is 0

dt = 0.001  # time step for the simulation


''' Implementnts the dynamic system (plant) - system input is action u
    and the method returns the next state of the plant (system),
    i.e. (position x, speed v) '''
def plant(action_u):
    # m \dotdot{x} + b xdot + k x = u
    # m=1 b=6, k = 9.86960
    m = 1
    b = 6
    k = 9.86960

    x_dot = v  # xdot is speed
    x_dotdot = (action_u - b*x_dot - k*x)/m

    v_dot = x_dotdot
    new_speed = v + v_dot*dt

    new_x = x + new_speed*dt
    return new_x, new_speed


# open a file for writing
pw = open('myfile_ex3.txt', 'w')

start_time = 0
end_time = 10.0

current_time = start_time

x_ref = 1  # the desired position

K_p = 50  # proportional gain
K_i = 40    # integral gain
```

```python
K_d = 8   # derivative gain

previous_error = 0
integral = 0

# list containing time for the simulation
time = []

# list containing positions x for the simulation
position = []

#   simulate the system operation from the beginning till
#   the end of the simulation
while current_time <= end_time:
    error = x_ref - x

    # I(ntegral) component of the PID controller
    integral = integral + error*dt

    # D(erivative) component of the PID controller
    deriv = (error - previous_error)/dt

    # the output (action) of the PID controller
    action = K_p*error + K_i*integral + K_d*deriv

    # remember the last error when the previous action
    # was applied to the plant
    previous_error = error

    # apply the new action to the plant to calculate
    # the new (current) speed
    x, v = plant(action)

    print(f"Time: {current_time} Action: {action}, Position={x}")
    pw.write(f'{x} {current_time}\n')

    # save v and current_time in the corresponding lists
    time.append(current_time)
    position.append(x)

    # advance the time
    current_time += dt


pw.close()

# plot speed vs time
```

```python
plt.plot(time, position)
plt.ylim(0, 1.2)  # set the limits of the range for the y-axis
plt.show()  # display the plot
```

[ ]:

```python

```