Universidade Federal de Uberlândia
João Gabriel Amaral de Moura - 11821ETE008
FEELT - Faculdade de Engenharia Elétrica

Sistemas Embarcados 2

Trabalho Final 1
Simulação do drone 2D

Professor: Éder Alves de Moura
Grupo: João Gabriel Amaral de Moura - 11821ETE008, Erick Damasceno - 11811EMT001, Talles Silva Rodrigues - 11611EAU012, Amanda Lópes Gonçalves - 11821ETE005;

07/10/21

SUMÁRIO

1 - Introdução

O principal objetivo deste trabalho prático foi consolidar os conhecimentos assimilados na disciplina de controle de sistemas lineares. Fizemos isso por meio do desenvolvimento de um sistema de controle capaz de conduzir um drone de funcionamento bidimensional. Em nosso sistema de controle 2D, é possível movimentar o drone pelo teclado da máquina e também por waypoints. Waypoint se trata de um determinado ponto no globo terrestre definido por coordenadas geográficas através do sistema GPS.

2 - Método

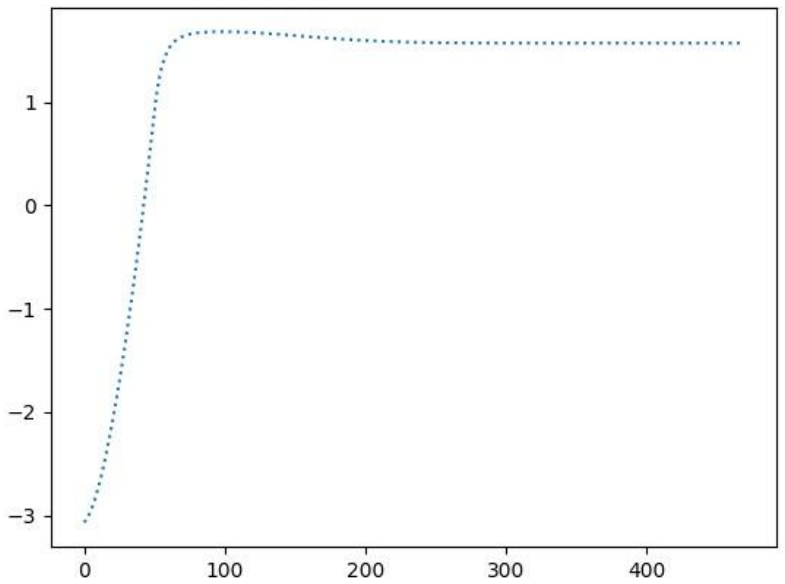
A metodologia utilizada consistiu na aplicação direta dos conceitos passados em aula, onde a formulação dinâmica do sistema foi feita utilizando uma malha de realimentação e um controle PID. Após ajustes manuais de ganhos do PID, o sistema foi implementado utilizando linguagem Python, por meio da Engine Pygame.

A linguagem Python se trata de uma linguagem de programação de código aberto, otimizada para qualidade, integração, produtividade e portabilidade. Nessa linguagem, os procedimentos estão submetidos à classes, o que possibilita maior controle e estabilidade de códigos para projetos de grandes proporções. Um dos seus maiores atrativos, é o fato de possuir diversas bibliotecas, tanto nativas como de terceiros, o que a torna muito difundida e útil em uma grande quantidade de setores.

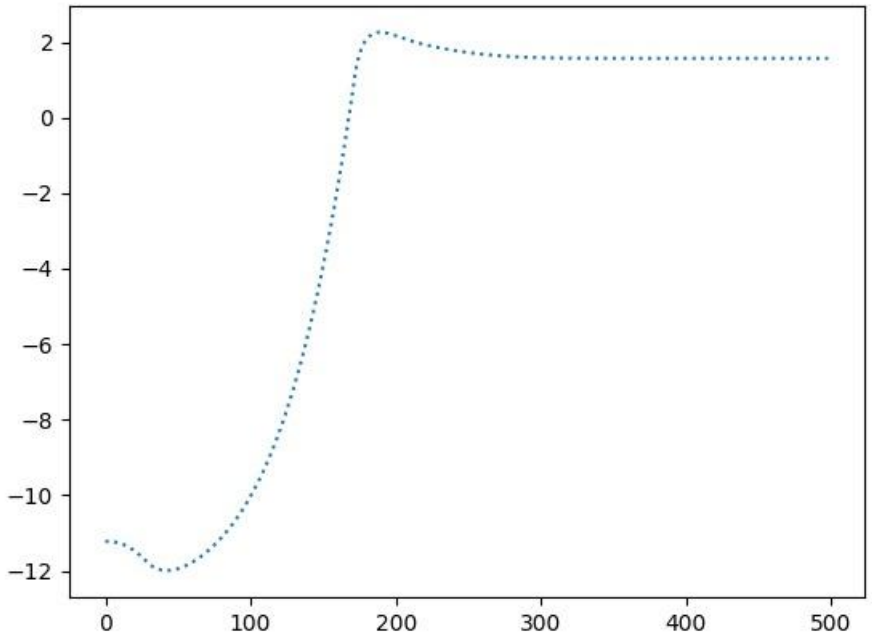
- Pygame: Game Engine utilizada para criar jogos em geral, independente de qual plataforma for escolhida. Possibilita criar “desenhos” em um display e a atualização constante destes cria a fluidez necessária para o desenvolvimento do jogo.
- Numpy: Biblioteca utilizada para cálculos que utilizam arrays multidimensionais (Matrizes).
- Matplotlib: Biblioteca capaz de gerar os gráficos necessários para revisão dos resultados a partir de matrizes.

3 - Resultados

Respostas do sistema:



Aceleração



Controle de ângulo

Código:

```
import pygame
import sys
from pygame.locals import *
import numpy as np
import matplotlib.pyplot as plt

class Drone:

    def __init__(self):
        self.m = 0.3 #kg
        self.g = 9.8 #gravidade
        self.F_max = 6
        self.F_min = 0

    def gravity(self): #gravidade é massa * aceleração * F_angular 0
        v=self.m*self.g*np.sin(np.pi/2)
        return v

    def F_motor1(self, accelerate, fi): #força vertical
        #-----cálculo da força vertical do motor 2
        v=self.m*accelerate*np.sin(fi)*2
        #-----condiçaõ max de força do motor
        if v>=self.F_max:v=self.F_max
        elif v<=self.F_min:v=self.F_min
        return -v

    def F_angular(self, accelerate, fi): #força horizontal dos motores
        #-----Calculo da Força horizontal
        f_motor=drone.F_motor1(accelerate, pi/2)*2
        v=np.cos(fi)*f_motor
        return v


if __name__ == "__main__":
    pygame.init() # initialize pygame
    clock = pygame.time.Clock() #inicializa o tempo no jogo
    screen = pygame.display.set_mode((900,550)) #inicia a janela
```

```
programIcon = pygame.image.load(r"Images/icon.png") #carrega novo icone do programa
pygame.display.set_icon(programIcon) #seta o icone
pointerImg = pygame.image.load(r"Images/cursor.png")
bg = pygame.image.load(r"Images/95.jpeg") #chama o backgroud para bg
ship = pygame.image.load(r"Images/drone.png") #carrega a imagem do drone
control = pygame.image.load(r"Images/control.png")
analog = pygame.image.load(r"Images/analog button.png")
pygame.mouse.set_visible(0) #mouse invisivel
pygame.display.set_caption('Drone Simulator')
pygame.display.flip()
drone = Drone()

autonomo =False
pi = np.pi
y_force = 0
fi = pi/2
aceleracao = 0
data = []
x = 500
y = 435
y1= 0
x1 = 0
erro_acumulado_y = 0
erro_acumulado_x = 0
iteracao=0
last_error =0
derivativo_x=0

running = True
while running:
    clock.tick(60)
    screen.blit(bg, (0,0))
    screen.blit(pygame.transform.rotate(ship, -np.degrees(fi)), (x, y))
    screen.blit(control, (50,400))
    screen.blit(analog, (78,(-aceleracao*0.6)+480))
    screen.blit(analog, ((fi*10)+185,480))
    key = pygame.key.get_pressed()
    pos = pygame.mouse.get_pos()
    screen.blit(pointerImg, (pos[0],pos[1]))

    #-----------Comando de encerrar o programa----------#
    for event in pygame.event.get(): #procura um evento
        if event.type == pygame.QUIT or key[pygame.K_ESCAPE]: #se o evento for do tipo quit
            running = False #fecha o looping
            #ypoints = data
            #plt.plot(ypoints, linestyle = 'dotted')
            #plt.show()
            pygame.display.quit() #fecha o game
            sys.exit() #fecha o sistema


    #------------Comando Horizontal-----------------#
    if key[pygame.K_LEFT]: #se clicar pra esquerda
        fi -= pi/180  #decresce a posiçaõ no eixo horizontal para o sentido da direita
    elif key[pygame.K_RIGHT]: #se clicar para direita
        fi += pi/180 #incrementa a posição no eixo horizontal para o sentido da esquerda
        autonomo = False

    #------------Comando Vertical---------------#
    if key[pygame.K_UP]: #se clicar pra cima
        aceleracao += 0.5#decresce a posição no eixo vertical para cima
    elif key[pygame.K_DOWN]: #se clicar para baixo
        aceleracao -= 0.5 #decresce a posiçaõ no eixo vertical para baixo
        autonomo = False

    #------------Comando Waypoint-----------------#
    if event.type == MOUSEBUTTONDOWN: #se o evento for clique do mouse
        x1,y1 = pygame.mouse.get_pos() #transforma x e y na posiçao do clique
        iteracao = 0
        autonomo = True

    if autonomo == True:
        iteracao += 1
        erro_x = (x-x1)
        erro_y = (y-y1)
        erro_acumulado_y += erro_y * iteracao
        #erro_acumulado_x += (fi-np.arctan(erro_y/erro_x))*iteracao
        erro_acumulado_x += ((-erro_x/1800)/90)*iteracao
        derivativo_x = (erro_acumulado_x - last_error)/iteracao
        last_error = erro_acumulado_x
        aceleracao = (erro_acumulado_y * 0.000005) + (erro_y * 0.1)
        fi = ((-erro_x/1800)/90)*2500 + erro_acumulado_x*0.5 + derivativo_x*1000
        data.append(fi)
```

```
#-----------Limites adotados para o Drone----------#
if fi<=pi/4: fi=pi/4
elif fi>=3*pi/4: fi=3*pi/4

if aceleracao>=10: aceleracao=10
elif aceleracao<=-10: aceleracao=-10

#-----------Forças aplicadas no sistema------------#
x += drone.F_angular(aceleracao, fi)
y_force = drone.F_motor1(aceleracao,fi)
y = y_force + y
y += drone.gravity()


#-----------Limites do Mapa------------------#
if y <= 0: y = 0
elif y >= 420: y = 420

if x <= 0: x = 0
elif x >= 800: x = 800
#--------------------------------------------#

pygame.display.update()
```

4 - Conclusão

Para a construção desse sistema foi necessário conhecimentos específicos da linguagem Python para entender como cada função se encaixaria no código. Além disso, com todo o conhecimento adquirido em aula foi possível desenvolver a malha de realimentação e o controle PID fundamentais para a construção do projeto. Durante o desenvolvimento foram realizados diversos testes para verificar erros de programação no código escrito.

Em suma, através do trabalho de todos os participantes, podemos entender na prática o funcionamento de um sistema embarcado, que nada mais é do que um sistema microprocessado em que uma máquina está anexada ao sistema que a controla.

7 - Bibliografia

O que é Python, para que serve e por que aprender? - https://kenzie.com.br/blog/o-que-e-python/

Waypoint - https://pt.wikipedia.org/wiki/Waypoint

Sistema Embarcado - https://pt.wikipedia.org/wiki/Sistema_embarcado#:~:text=Um%20sistema%20embarcado%20(ou%20sistema,ou%20sistema%20que%20ele%20controla.&text=Em%20geral%20tais%20sistemas%20n%C3%A3o,funcionalidade%20alterada%20durante%20o%20uso.

Introdução aos sistemas embarcados e microcontroladores - https://www.embarcados.com.br/sistemas-embarcados-e-microcontroladores/