Controle 1D e 2D para quadrotors

1. Exercícios complementares

Para fixar o conteúdo visto até o momento, segue alguns exemplos interativos para vocês se divertirem ao passo que exercitam a mente.

1.1 Ajustando o ganho proporcional

Baixe a simulação GUI intitulada GainTuningQuiz e descompacte seu conteúdo para o diretório em que deseja completar o exercício. Abra o MATLAB nesse diretório e digite "runsim" na janela de comando do MATLAB para iniciar a GUI.

Dado o ganho derivativo $K_v = 18$, encontre o ganho proporcional K_p, valor tal que o tempo de subida seja menor que 1s e o overshoot seja menor que 5%.

1.2 Influência do peso (Impulso de peso)

Baixe a simulação GUI intitulada ThrustWeightExercise e descompacte seu conteúdo para o diretório em que deseja completar o exercício. Abra o MATLAB nesse diretório e digite "runsim" na janela de comando do MATLAB para iniciar a GUI.

Dado o empuxo máximo de 11,77N (1,2Kgf), qual é a massa máxima (kg) de forma que o tempo de subida seja menor que 1s? (Dê uma resposta com 2 casas decimais).

1.3 Distância de parada

Baixe a simulação GUI intitulada StoppingDistanceExercise e descompacte seu conteúdo para o diretório em que deseja completar o exercício. Abra o MATLAB nesse diretório e digite "runsim" na janela de comando do MATLAB para iniciar a GUI.

Dada a desaceleração máxima de $15 m/s^2$, qual é a velocidade máxima para que o quadrotor possa parar dentro de 6m? (Dê uma resposta com uma casa decimal).

2. Exercícios de Simulação

2.1 Exercício 1: Quadrotor e Controlador PD (Controle 1D)

Nesta tarefa, você começará a usar um simulador de quadrotor 1D e implementará um controlador para ele. Estaremos testando seu controlador com dois casos.

No primeiro caso de teste, o quadrotor simplesmente precisa se estabilizar a uma altura de 0. Já o segundo caso de teste dá ao quadrotor uma entrada de passo de 1 metro; ou seja, seu quadrotor será solicitado a subir até uma altura de 1 metro.

O objetivo desta tarefa é fazer com que seu controlador atinja a resposta desejada para cada caso de teste.

Em outras palavras desejamos familiarizá-lo com o trabalho com o simulador quadrotor e com a implementação de um controlador Proportional Derivative (PD). No capítulo 1, nós fornecemos a você um quadrotor GUI para ajustar os ganhos de controle PD. Neste exercício, você terá que implementar seu próprio controlador PD para controlar a altura de um quadrotor, além de sintonizar seus ganhos. Para começar, você precisará baixar o código inicial disposto na pasta intitulada *1D Control Quadrotor* e descompactar seu conteúdo no diretório no qual você deseja concluir o exercício.

Utilizamos um dos solucionadores de ODE do MATLAB, chamado ode45, para simular o comportamento do quadrotor. Você pode ler mais detalhes no site da Mathworks ou em outros recursos online. Usamos também a função plot / plot3 para ajudar a visualizar o estado atual do quadrotor a cada vez passo. Você pode dar uma olhada no arquivo *height_control.m* para o código de simulação.

Antes de implementar sua própria função, você deve primeiro tentar executar *runsim.m* em seu ambiente do MATLAB. Se você vir um quadrotor caindo da altura 0, o simulador está funcionando perfeitamente em seu computador e você pode continuar com outras tarefas. O segmento suplementar ***"Material Suplementar: Introdução à Primeira Programação Atribuição"*** percorre essas etapas. O código de partida para o controlador (controlador.m) produz entradas para o robô que são todas de empuxo zero e, portanto, o quadrotor cai devido à gravidade.

2.1.1 Controlador PD

A equação dinâmica para o movimento do quadrotor na direção Z é dada pela Eq. 1

$$\ddot{z} = \frac{u}{m} - g \tag{1}$$

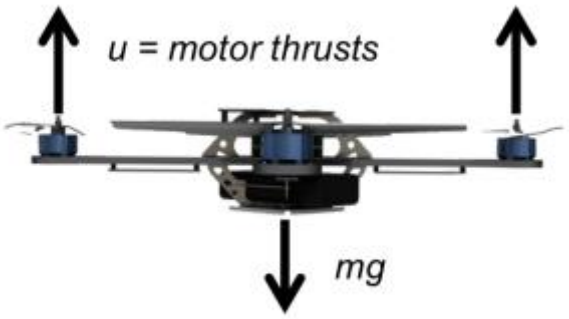

Figura 1 – Modelo 1D do Quadrotor

Portanto, a entrada de controle para um controlador PD é

$$u = m(\ddot{z}_{des} + K_p e + K_v \dot{e} + g \quad (2)$$

Onde e e $\dot{e}$ podem ser obtidos a partir dos estados atual e desejado $(z, z_{des}, \dot{z}, \dot{z}_{des})$.

2.1.2 Assignment

• Arquivos inclusos neste exercício

1. [❤] controller.m – Controlador para um quadrotor;
2. runsim.m – Script para chamar e testar o controle implementado;
3. height_control.m – Código de simulação que será chamado pelo runsim;
4. submit.m - Um script a ser chamado para gerar os arquivos enviados;
5. evaluate.p - Script de avaliação a ser chamado por submit.m;
6. fixed_set_point.m - Função de resposta ao degrau;
7. utils/ - Funções auxiliares para simulador quadrotor.

[❤] Indica arquivos que você precisará implementar.

2.1.3 Tarefa

Você precisará primeiro implementar um controlador PD para controle de altura do quadrotor. Então, ajuste o ganho proporcional (Kp) e o ganho derivativo (Kv) no arquivo controller.m até o quadrotor convergir suavemente para uma entrada de resposta ao degrau.

Submissão

Para enviar seus resultados, você precisa executar o comando submit em seu MATLAB (janela de comando). Um script irá avaliar seu controlador em dois casos de teste e gerar arquivos de saída (arquivos com o tipo .mat). Haverá um arquivo de saída para cada caso de teste. No primeiro caso de teste, o quadrotor simplesmente precisa se estabilizar a uma altura de 0. O segundo caso de teste dá ao quadrotor uma entrada em degrau de 1 metro; isso é, seu quadrotor deverá subir a uma altura de 1 metro. A resposta a esta entrada deve têm um tempo de subida de menos de 1s e um excesso máximo de menos de 5%. Lembre-se disso, o tempo de subida é o tempo que leva para atingir 90% do valor do estado estacionário, então, neste caso, você deve atingir 0,9 metros em menos de um segundo. Observe que a resposta ao degrau é diferente dos exercícios que você já completou. Assim, ao usar sua posição e derivada, ganhos de uma tarefa anterior pode ser um bom lugar para começar, você pode precisar ajustá-los mais adiante para completar com sucesso esta tarefa.

2.2 Exercício 2: Controle do quadrotor no plano (Controle 2D)

Neste exercício, você implementará o controlador PD para controlar o movimento do quadrotor no plano Y-Z.

2.2.1 Sistema de Coordenadas

Os sistemas de coordenadas e o diagrama de corpo livre para o modelo planar de um quadrotor são mostrados na Fig. 2. O referencial inercial, ***A***, é definido pelos eixos a2 e a3. A estrutura do corpo, ***B***, está ligado ao centro de massa do quadrotor com b2 coincidindo com a direção para frente e b3 perpendicular aos rotores apontando verticalmente para cima (ver Fig. 2).

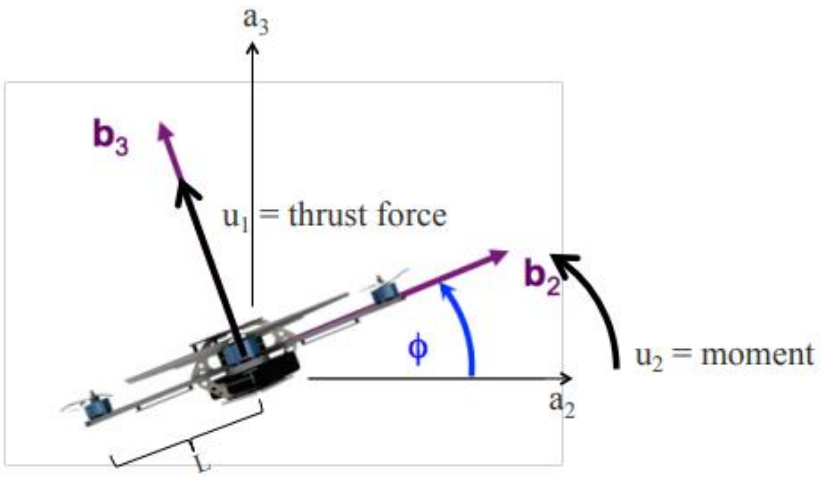

Figura 2 - Modelo quadrotor plano e os sistemas de coordenadas.

2.2.2 Dinâmica

Para um quadrotor modelado no plano Y - Z, sua orientação é definida por um ângulo de rotação,ϕ. Supõe-se que seus ângulos de inclinação e guinada são 0. Você precisará da matriz de rotação para transformar componentes de vetores em ***B*** em componentes de vetores em ***A***:

$$ {}^{A}[R]_{B} = \begin{bmatrix} \cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \tag{3} $$

Vamos denotar os componentes da velocidade angular do robô na estrutura do corpo por $\dot{\phi}$:

$$ {}^{A}\omega_{B} = \dot{\phi} \tag{4} $$

No modelo da planar do quadrotor, consideramos apenas a força de empuxo em dois dos rotores. O quadrotor tem duas entradas: a força de impulso (u_1) e o momento (u_2), u_1 é a soma de os impulsos em cada rotor

$$u_1 = \sum_{i=1}^{2} F_i,$$

enquanto u_2 é proporcional à diferença entre os impulsos de dois rotores

$$u_2 = L(F_1 - F_2).$$

Aqui, L é o comprimento do braço do quadrotor.

Seja $r = [y, z]^T$ o vetor de posição do quadrotor planar em $\boldsymbol{A}$. As forças no sistema são a gravidade, na direção −a3, e a força de empuxo, na direção b3.
Consequentemente, pelas equações de movimento de Newton,

$$m\ddot{\boldsymbol{r}} = m\begin{bmatrix}\ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\ -mg\end{bmatrix} + {}^{A}[R]_{\boldsymbol{B}}\begin{bmatrix}0\\ u_1\end{bmatrix} = \begin{bmatrix}0\\ -mg\end{bmatrix} + \begin{bmatrix}-u_1\sin(\phi)\\ u_1\cos(\phi)\end{bmatrix} \tag{5}$$

A aceleração angular pode ser determinada pela equação de movimento de Euler

$$I_{xx}\ddot{\phi} = L(F_1 - F_2) = u_2$$

Como resultado, podemos escrever o modelo do sistema na forma matricial como,

$$\begin{bmatrix}\ddot{y}\\ \ddot{z}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0\\ -g\\ 0\end{bmatrix} + \begin{bmatrix}-\frac{1}{m}\sin(\phi) & 0\\ \frac{1}{m}\cos(\phi) & 0\\ 0 & 1/I_{xx}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{6}$$

2.2.3 Controlador

2.2.3.1 Linearização

O modelo dinâmico do quadrotor (Eq. 6) é não linear. No entanto, um controlador PD é projetado para um sistema linear. Para usar um controlador linear para este sistema não linear, primeiro devemos linearizar a equação de movimentos sobre uma configuração de equilíbrio. No caso do quadrotor, a configuração de equilíbrio é a configuração de foco em qualquer posição arbitrária y_0, z_0, com ângulo de rotação zero. A força de empuxo correspondente necessária para pairar nesta configuração é exatamente *mg*, enquanto o momento deve ser zero. Explicitamente, os valores de as variáveis relacionadas na configuração de foco são

$$y_0, z_0, \phi_0 = 0, u_{1,0} = mg, u_{2,0} = 0$$

Para linearizar a dinâmica, substituímos todas as funções não lineares das variáveis de estado e controle com suas aproximações de Taylor de primeira ordem no local de equilíbrio. Neste caso, as funções não lineares são sin (ϕ) e cos(ϕ).

Perto de $\phi = 0,\ \sin(\phi) \approx \phi\ e \cos(\phi) \approx 1$

$$\ddot{y} = -g\phi$$

$$\ddot{z} = -g + \frac{u_1}{m}$$

$$\ddot{\phi} = \frac{u_2}{I_{xx}}$$

Deixe **r** denotar uma variável de estado, y, z ou ϕ. Podemos encontrar a aceleração comandada daquele estado, $\ddot{r}_c$, correspondendo a um controlador (PD) como segue. Defina a posição e a velocidade erros como

$$e_p = r_T(t) - r$$

$$e_v = \dot{r}_T(t) - \dot{r}$$

Queremos que o erro satisfaça a seguinte equação diferencial, o que resultará em convergência do erro para algum valor de k_p e k_d.

$$(\ddot{r}_T(t) - r_c) + k_p e_p \ddot{} + k_v e_v = 0 \tag{7}$$

A partir disso, podemos ver que

$$\ddot{r}_c = \ddot{r}_T(t) + k_p e_p + k_v e_v, \tag{8}$$

onde k_p e k_v são ganhos proporcionais e derivados, respectivamente. Como resultado, as entradas u_1, u_2 podem ser derivadas como:

$$u_1 = mg + m\ddot{z}_c = m\{g + \ddot{z}_T(t) + k_{v,z}(\dot{z}_T(t) - \dot{z}) + k_{p,z}(z_T(t) - z)\} \tag{9}$$

$$u_2 = I_{xx}\ddot{\phi}_T(t) = I_{xx}\{\ddot{\phi}_c + k_{v,\phi}(\dot{\phi}_c - \dot{\phi}) + k_{p,\phi}(\phi_c - \phi)\} \tag{10}$$

$$\phi_c = -\frac{\ddot{y}_c}{g} = -\frac{1}{g}\{\ddot{y}_T(t) + k_{v,y}(\dot{y}_T(t) - \dot{y}) + k_{p,y}(y_T(t) - y)\} \tag{11}$$

2.2.3.2 Controle de Estabilidade

Pairar é o caso especial em que a posição desejada é constante e o *rolamento* (roll) desejado é zero. Para $\boldsymbol{r}_T(t) = \boldsymbol{r}_0 = [y_0, z_0]^T, \phi_T(t) = \phi_0, \dot{\boldsymbol{r}}_T(t) = \ddot{\boldsymbol{r}}_T(t) = 0$ nós obtemos:

$$u_1 = m[g - k_{v,z}\dot{z} + k_{p,z}(z_0 - z)]$$

$$u_2 = I_{xx}[\ddot{\phi}_c + k_{v,\phi}(\dot{\phi}_c - \dot{\phi}) + k_{p,\phi}(\dot{\phi}_c - \dot{\phi})]$$

$$\phi_c = -\frac{1}{g}(k_{v,y}(-\dot{y}) + k_{p,y}(y_0 - y)$$

2.2.3.3 Controle de Trajetória

Para acompanhamento de trajetória, dadas as trajetórias desejadas para cada estado e seus derivados, $r_T(t), \dot{r}_T(t), \ddot{r}_T(t)$, as entradas u_1, u_2 podem ser calculadas usando as Equações 9-11.

2.2.4 Atividade

• Arquivos inclusos neste exercício

1. [❤] controller.m – Controlador para um quadrotor;
2. runsim.m – Script para chamar e testar o controle implementado;
3. simulation_2d.m – Código de simulação que será chamado pelo runsim;
4. submit.m - Um script a ser chamado para gerar os arquivos enviados;
5. evaluate.p - Script de avaliação a ser chamado por submit.m;
6. trajectories/ - Inclui trajetórias de exemplo para testar seu controlador.;
7. utils/ - Funções auxiliares para simulador quadrotor.

[❤] Indica arquivos que você precisará implementar.

2.2.5 Tarefas

Você precisará concluir a implementação de um controlador PD no arquivo *controller.m* para que o quadrotor simulado possa seguir uma trajetória desejada. Antes de implementar as suas próprias funções, você deve primeiro tentar executar o *runsim.m* em seu ambiente MATLAB. Se você ver um quadrotor caindo da posição (0, 0), então o simulador funciona no seu computador e você pode continuar com outras tarefas. Novamente, o quadrotor cai porque as saídas padrão da função *controller.m* são todos zeros e nenhum impulso é aplicado.

Para testar diferentes trajetórias, você precisará modificar a variável trajetória dentro do runsim.m para apontar para a função apropriada.

2.2.6 Submissão

Para enviar seus resultados, você precisa executar o comando submit na janela de comando do MATLAB. Um script irá avaliar o seu controlador em duas trajetórias e gerar arquivos de saída (arquivos com o tipo .mat). Haverá um arquivo de saída para cada caso de teste. A avaliação é baseada em quão bem o seu controlador pode seguir as trajetórias desejadas. Para cada trajetória, temos limites superiores e inferiores

para o erro de posição cumulativo ao seguir a trajetória. Se o seu erro estiver abaixo do inferior limite, você obteve resultados positivos enquanto se o erro for maior do que o limite superior, você cometeu algum erro mais grave. Se o erro estiver entre os dois, você está no caminho certo, tente entender, entre em contato e tire suas dúvidas.