Instituto Tecnológico de Aeronáutica – ITA

Sistemas de Controle Contínuos e Discretos – CMC-12

Laboratório 4 – Projeto de Controlador para Multicóptero (Adaptado de laboratório de MPS-43 do Prof. Davi Santos)

Professor: Marcos Ricardo Omena de Albuquerque Maximo

9 de maio de 2022

Observação: por questões de compatibilidade, este laboratório deve ser feito utilizando Simulink R2021b.

1 Introdução

Neste laboratório, projetar-se-á um controlador para um multicóptero 2D com apenas duas hélices (*drone*), de modo que se move apenas no plano com 3 graus de liberdade. Embora seja um sistema fictício, destaca-se que uma abordagem popular para projeto de sistema de controle de multicópteros envolve separar as dinâmicas de movimento de cada eixo, como se fossem desacopladas. Assim, nosso modelo fictício de multicóptero 2D é um bom modelo didático, pois é possível estender as ideias de projeto discutidas aqui para projetar o sistema de controle de um multicóptero real. Conforme já discutido durante o curso, essa abordagem de transformar um sistema intrinsicamente MIMO em vários sistemas SISOs é necessária para que técnicas clássicas de projeto sejam usadas.

Na Figura 1, apresenta-se o multicóptero 2D, que está restrito a se mover no plano $x-z$, com os graus de liberdade x, y e θ (dois de translação e um de rotação). Note que dá-se o nome de **arfagem** (*pitch*, em inglês) para o ângulo θ no plano $x-z$ na nomenclatura aeronáutica. A massa e inércia do *drone* são m e J, respectivamente. Além disso, a separação entre os rotores (atuadores) é l. Com isso, atuam no multicóptero a força peso $m\mathbf{g}$ e as forças $\mathbf{f}_1$ e $\mathbf{f}_2$ geradas pelos rotores. As forças $\mathbf{f}_1$ e $\mathbf{f}_2$ estão relacionadas às velocidades de rotação v_1 e v_2 das hélices por

$$\begin{cases} f_1 = kv_1^2, \\ f_2 = kv_2^2, \end{cases} \tag{1}$$

em que k é uma constante que depende da geometria da hélice. Ademais, a força f e o torque τ resultantes que atuam no *drone* são

$$f = f_1 + f_2, \tag{2}$$

$$\tau = \frac{l}{2}\left(f_2 - f_1\right). \tag{3}$$

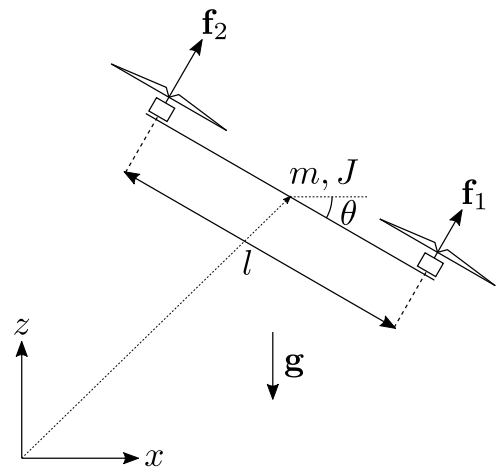

Figura 1: Multicóptero 2D com três graus de liberdade.

Portanto, pelas leis de Newton e Euler, pode-se determinar as dinâmicas de cada grau de liberdade como

$$\ddot{\theta} = \frac{1}{J}\tau, \tag{4}$$

$$\ddot{x} = \frac{1}{m} f \sin\theta, \tag{5}$$

$$\ddot{z} = \frac{1}{m} f \cos\theta - g, \tag{6}$$

em que os efeitos de resistência do ar foram desprezados. Com isso, o sistema não-linear completo fica

$$\begin{cases} \dfrac{d}{dt}x = \dot{x}, \\ \dfrac{d}{dt}\dot{x} = \dfrac{1}{m} f \sin\theta, \\ \dfrac{d}{dt}z = \dot{z}, \\ \dfrac{d}{dt}\dot{z} = \dfrac{1}{m} f \cos\theta - g, \\ \dfrac{d}{dt}\theta = \dot{\theta}, \\ \dfrac{d}{dt}\dot{\theta} = \dfrac{1}{J}\tau, \end{cases} \tag{7}$$

em que

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & z & \dot{z} & \theta & \dot{\theta} \end{bmatrix}^T \tag{8}$$

e

$$\mathbf{u} = \begin{bmatrix} f & \tau \end{bmatrix}^T \tag{9}$$

são os vetores de estado e de entrada, respectivamente. Embora os comandos do *drone* para os atuadores sejam na realidade v_1 e v_2, pode-se trabalhar com $\mathbf{u}$ conforme definido em (9) e então obter v_1 e v_2 a partir de (1), (2) e (3), ou seja

$$\begin{cases} v_1 = \sqrt{\dfrac{1}{k}\left(\dfrac{f}{2} - \dfrac{\tau}{l}\right)}, \\ v_2 = \sqrt{\dfrac{1}{k}\left(\dfrac{f}{2} + \dfrac{\tau}{l}\right)}. \end{cases} \tag{10}$$

Trabalhar com $\mathbf{u} = [f\ \tau]^T$ evita introduzir a não-linearidade presente em (1) no sistema. Um ponto de equilíbrio do sistema é

$$\begin{cases} \dot{x} = 0, \\ \dot{z} = 0, \\ \dot{\theta} = 0, \\ \theta = 0, \\ f = mg, \\ \tau = 0. \end{cases} \tag{11}$$

Nesse equilíbrio, o veículo aéreo permanece em voo nivelado numa posição (x, z) qualquer. Perceba que $f = mg$ atua como uma espécie de termo de *feedforward* para cancelar a gravidade. Linearizando o sistema em torno de (11) por análise de pequenos sinais, obtém-se

$$\frac{d}{dt}\begin{bmatrix} \delta x \\ \delta\dot{x} \\ \delta z \\ \delta\dot{z} \\ \delta\theta \\ \delta\dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\dot{x} \\ \delta z \\ \delta\dot{z} \\ \delta\theta \\ \delta\dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/m & 0 \\ 0 & 0 \\ 0 & 1/J \end{bmatrix} \begin{bmatrix} \delta f \\ \delta\tau \end{bmatrix} \tag{12}$$

Para facilitar o projeto do sistema de controle, pode-se dividi-lo em três malhas, conforme mostram as figuras 2, 3 e 4, em que são mostradas as malhas de arfagem, vertical e horizontal, respectivamente. Nessas figuras, destaca-se que:

- $K_{p\theta}$ e $K_{v\theta}$ são ganhos proporcional e de velocidade do controlador P+V de arfagem, respectivamente.
- $C_z(s)$ e $C_x(s)$ são compensadores PID das malhas vertical e horizontal, respectivamente.
- $F_z(s)$ e $F_x(s)$ são pré-filtros das malhas vertical e horizontal, respectivamente.
- $G_{f\theta}$ é a dinâmica de malha fechada de arfagem, interna à malha horizontal.
- $\Theta_r(s)$, $Z_r(s)$ e $X_r(s)$ são referências de arfagem, altura e posição horizontal, respectivamente.

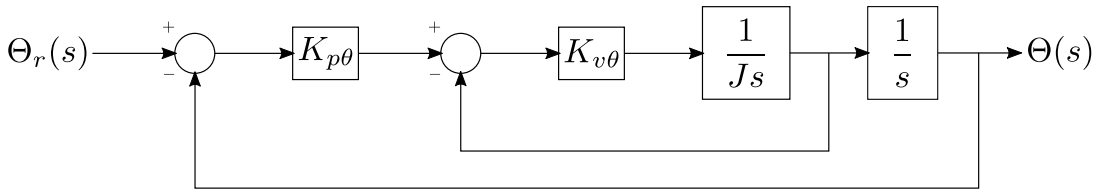

Figura 2: Malha interna de arfagem para controlar θ.

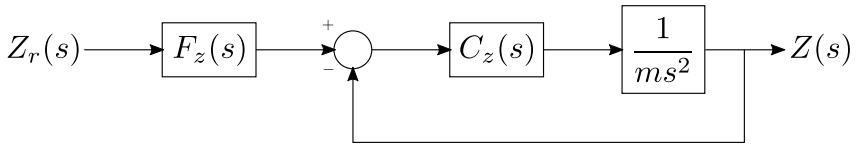

Figura 3: Malha vertical para controlar z.

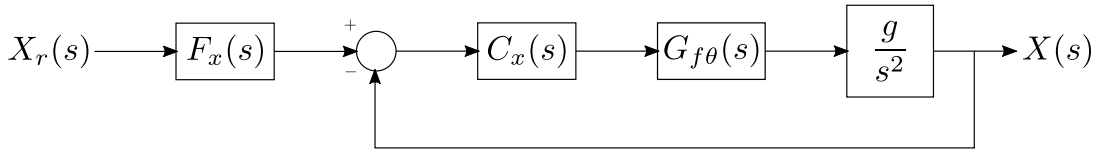

Figura 4: Malha horizontal para controlador x.

- $\Theta(s)$, $Z(s)$ e $X(s)$ representam o ângulo de arfagem, a altura e a posição horizontal (no domínio s), respectivamente.

Como exemplo neste laboratório, usa-se um multicóptero com os seguintes parâmetros: $m = 0{,}5\ kg$, $J = 0{,}04\ kgm^2$ e $l = 0{,}2\ m$. Além disso, adota-se $g = 9{,}81\ m/s^2$. Esses parâmetros já estão configurados na função de MATLAB `obterPlantaMulticoptero`. Além disso, a função `obterRequisitos` fornece os requisitos de cada malha, conforme discutido mais adiante. Os requisitos foram escolhidos de modo que a malha de arfagem, interna à malha horizontal, seja muito mais rápida que a malha horizontal. Ademais, decidiu-se que as malhas horizontal e vertical tivessem comportamentos semelhantes, sem dar preferência para alguma das coordenadas.

2 Tarefas

2.1 Projeto do Controlador de Arfagem

Para a malha de arfagem, adota-se um controlador P+V conforme mostrado na Figura 2. Nesse caso, solicita-se:

- Obtenha a função de transferência de malha fechada $G_{f\theta}(s) = \Theta(s)/\Theta_r(s)$.
- Determine $K_{p\theta}$ e $K_{v\theta}$ para que o sistema tenha tempo de subida (de 0 a 100%) $t_r|_0^{100\%} = 0{,}1\ s$ e sobressinal $M_p = 0{,}05$.
- Implemente o cálculo dos ganhos do controlador de arfagem na função de MATLAB `projetarControladorArfagem`.

- Avalie o sistema de controle projetado através de `avaliarMalhaArfagem`, que apresenta a resposta ao degrau unitário do sistema. Para que essa função funcione corretamente, deve-se implementar a expressão da função de transferência em malha fechada $G_{f\theta}$ em `obterMalhaArfagem`. Apresente o gráfico gerado no relatório.

Apresente as deduções matemáticas realizadas no seu relatório. Além disso, discuta o atendimento aos requisitos pelo sistema de controle.

2.2 Projeto do Controlador Vertical

Para a malha vertical, adota-se um controlador PID com pré-filtro conforme apresenta a Figura 3. Considere

$$C_z(s) = \frac{K_{dz}s^2 + K_{pz}s + K_{iz}}{s}. \tag{13}$$

Desse modo, pede-se:

- Obtenha a função de transferência de malha fechada $G_{fz}(s) = Z(s)/Z_r(s)$. Mantenha o pré-filtro apenas como $F_z(s)$ por enquanto.

- Assumindo polos dominantes, determine K_{pz}, K_{iz} e K_{dz} para que o sistema tenha aproximadamente tempo de subida $t_r|_0^{100\%} = 1{,}0$ s e sobressinal $M_p = 0{,}1$. Para isso, escolha os ganhos para alocar os polos de malha fechada em

$$p_1 = -\xi\omega_n + j\omega_n\sqrt{1-\xi^2},\ p_2 = -\xi\omega_n - j\omega_n\sqrt{1-\xi^2},\ p_3 = a = -5\xi\omega_n, \tag{14}$$

  em que a foi escolhido de modo que os polos complexo-conjugados p_1 e p_2 sejam dominantes. Além disso, determine uma expressão para o pré-filtro $F_z(s)$ para eliminar os zeros introduzidos pelo compensador PID. **Dica:** iguale o denominador de $G_{fz}(s)$ ao polinômio produzido pelos polos p_1, p_2 e p_3.

- Implemente o cálculo dos ganhos do controlador vertical na função de MATLAB `projetarControladorVerticalAnalitico`.

- Para avaliar a necessidade de pré-filtro, obtenha respostas ao degrau do sistema sem e com pré-filtro. Discuta qualitativamente a influencia dos zeros na resposta do sistema, destacando a dificuldade que o sistema sem pré-filtro tem para atendimento aos requisitos. Apresente um único gráfico em seu relatório comparando as duas respostas. Ademais, implemente a expressão da função de transferência em malha fechada G_{fz} (com pré-filtro) em `obterMalhaVertical`. **Não** foi fornecida função pronta para realizar a comparação solicita nesse item.

- Mesmo com o pré-filtro, o sistema não atende tão bem aos requisitos devido à influência do polo em $s = -a$. Para refinar o projeto (no sentido de melhor atendimento aos requisitos) sem precisar deixar o polo em $s = -a$ ainda mais distante da origem, o que necessitaria de ganhos maiores, pode-se usar um procedimento numérico de busca força bruta para um ajuste fino dos ganhos. Sejam $t_{r,req}|_0^{100\%}$ e $M_{p,req}$ os valores de tempo de subida e sobressinal desejados, respectivamente, a ideia sugerida consiste em fazer uma grade (*grid*) em torno de $t_{r,req}|_0^{100\%}$ e $M_{p,req}$,

com vários valores de projeto $t_{r,proj,ij}|_0^{100\%}$ e $M_{p,proj,ij}$ para tempo de subidade e sobressinal, respectivamente, em que os índices i e j indicam diferentes posições na grade bidimensional. Perceba que também é possível fazer uma busca diretamente sobre os ganhos do controlador K_{pz}, K_{iz} e K_{dz}, porém deu-se preferência para uma busca sobre os requisitos $t_{r,req}|_0^{100\%}$ e $M_{p,req}$ para reduzir o número de parâmetros de 3 para 2. Para a busca, adote uma grade com 20 valores igualmente espaçados de 0,8 $t_{r,req}|_0^{100\%}$ a 1,2 $t_{r,req}|_0^{100\%}$ e 20 valores igualmente espaçados de $0{,}8M_{p,req}$ a $1{,}2M_{p,req}$. Então, para decidir qual par de parâmetros é o melhor, use o seguinte critério (medida de erro em relação ao comportamento desejado):

$$J_{ij}(t_{r,proj,ij}|_0^{100\%}, M_{p,proj,ij}) = \frac{|t_{r,req}|_0^{100\%} - t_{r,ij}|_0^{100\%}|}{|t_{r,req}|_0^{100\%}|} + \frac{|M_{p,req} - M_{p,ij}|}{|M_{p,req}|}, \qquad (15)$$

em que $t_{r,ij}|_0^{100\%}$ e $M_{p,ij}$ são os valores de tempo de subida e do sobressinal, respectivamente, determinados por simulação após projeto do controlador usando $t_{r,proj,ij}|_0^{100\%}$ e $M_{p,proj,ij}$. A ideia é escolher a posição (i, j) na grade com menor valor de J_{ij}. Uma estrutura para auxiliar a implementar esse procedimento de projeto foi fornecida na função `projetarControladorVerticalBusca`. Para facilitar a implementação use a função `stepinfo` (veja Seção 4). Note que há várias maneiras de melhorar esse procedimento, porém o algoritmo "força bruta" adotado já produz bons resultados para o caso considerado.

- Avalie o sistema de controle projetado com os métodos analítico e de busca através de `avaliarMalhaVertical`.

Apresente as deduções matemáticas realizadas no seu relatório. Além disso, discuta o atendimento aos requisitos pelo sistema de controle com os diferentes métodos de projeto.

2.3 Projeto do Controlador Horizontal

Para a malha horizontal, adota-se um controlador PID com pré-filtro conforme apresenta a Figura 4. Considere

$$C_x(s) = \frac{K_{dx}s^2 + K_{px}s + K_{ix}}{s}. \qquad (16)$$

Desse modo, pede-se:

- Obtenha a função de transferência de malha fechada $G_{fx}(s) = X(s)/X_r(s)$. Adote um pré-filtro $F_x(s)$ semelhante ao adotado no projeto da malha vertical, de modo a cancelar os zeros provenientes do compensador PID.

- Desprezando a influência da malha de arfagem $G_{f\theta}$ (i.e. considere $G_{f\theta} = 1$) e assumindo polos dominantes, determine K_{px}, K_{ix} e K_{dx} para que o sistema tenha aproximadamente tempo de subida $t_r|_0^{100\%} = 1{,}0$ s e sobressinal $M_p = 0{,}1$. Para isso, escolha os ganhos para alocar os polos de malha fechada em

$$p_1 = -\xi\omega_n + j\omega_n\sqrt{1-\xi^2},\ p_2 = -\xi\omega_n - j\omega_n\sqrt{1-\xi^2},\ p_3 = a = -5\xi\omega_n, \qquad (17)$$

em que a foi escolhido de modo que os polos complexo-conjugados p_1 e p_2 sejam dominantes.

- Implemente o cálculo dos ganhos do controlador vertical na função de MATLAB `projetarControladorHorizontalAnalitico`.
- Implemente a expressão da função de transferência em malha fechada G_{fx} em `obterMalhaHorizontal`. Nesse caso, você deve considerar a influência da dinâmica da malha de arfagem.
- Faça um procedimento semelhante ao já feito para o controlador vertical para refinar o controlador horizontal. Nesse caso, o ajuste fino compensará tanto a influência do polo em $s = -a$ quanto a dinâmica da malha de arfagem $G_{f\theta}$. Adote o mesmo espaçamento e o critério de desempenho (também chamado de "função de custo") adotado no projeto do controlador vertical. Uma estrutura para auxiliar a implementar esse procedimento de projeto foi fornecida na função `projetarControladorHorizontalBusca`.
- Avalie o sistema de controle projetado com os métodos analítico e de busca através de `avaliarMalhaHorizontal`.

Apresente as deduções matemáticas realizadas no seu relatório. Além disso, discuta o atendimento aos requisitos pelo sistema de controle com os diferentes métodos de projeto.

2.4 Avaliação do Sistema de Controle do Multicóptero

Nesta seção, avalia-se o sistema de controle do multicóptero considerando a dinâmica não-linear. Foram fornecidos os seguintes arquivos:

- `multicoptero.slx`: modelo Simulink **parcialmente** implementado de simulação não-linear de voo do multicóptero.
- `projetarControladorMulticoptero.m`: projeta os controladores para as malhas de arfagem, vertical e horizontal.
- `simularMulticoptero.m`: simula o voo do multicóptero considerando parâmetros de entrada.
- `simularExperimentoMulticoptero.m`: simula os experimentos de 'a' a 'g' descritos a seguir neste roteiro.
- `tracarGraficosSimulacao.m`: traça gráficos a partir do resultado da simulação. Os seguintes gráficos são exibidos:
  1. $x_r \times z_r$ e $x \times z$.
  2. $t \times x_r$ e $t \times x$, em que t significa tempo.
  3. $t \times z_r$ e $t \times z$.
  4. $t \times \theta_r$ e $t \times \theta$.
  5. $t \times f$.
  6. $t \times \tau$.

- `fazerAnimacaoMulticoptero.m`: produz uma animação mostrando o voo do multicoptero realizado durante uma simulação.

Conclua a implementação do modelo de Simulink `multicoptero.slx`. Além das questões já discutidas até aqui, destaca-se que adicionou-se saturações de f e de θ para evitar que o sistema entre em situações em que a não-linearidade instabilize o sistema. Ademais, perceba que o subsistema da malha de arfagem também deve ser implementado.

Para avaliar o controlador projetado, use `simularExperimentoMulticoptero.m` passando um caractere de 'a' a 'g' como argumento para escolher qual experimento executar. Em todos os experimentos, o *drone* começa com $x = 0\ m$, $z = 0\ m$ e $\theta = 0\ rad$. Analise os resultados dos experimentos conforme indicado a seguir. Fique à vontade para incluir gráficos além dos solicitados no seu relatório. Os experimentos são os seguintes:

(a) Voo de 10 s com $x_r = 0\ m$ e $z_r = 1\ m$. Analise principalmente o gráfico $t \times z$.

(b) Inicialmente, preparação de 1 s com $x_r = 0\ m$ e $z_r = 1\ m$. Então, comandado para $x_r = 1\ m$ e $z_r = 1\ m$. Duração total é 10 s. Analise principalmente os gráficos $x \times z$ e $t \times x$.

(c) Inicialmente, preparação de 1 s com $x_r = 0\ m$ e $z_r = 1\ m$. Então, comandado com velocidade horizontal constante $\dot{x}_r = 1\ m/s$. Analise principalmente os gráficos $x \times z$ e $t \times x$.

(d) Comando igual ao experimento 'a', mas uma carga de 0,2 kg é colocada em cima do *drone* em $t = 3\ s$. Analise principalmente o gráfico $t \times z$.

(e) Comando igual ao experimento 'c', mas um vento começa a empurrar o *drone* com $-2\ N$ em x a partir de $t = 3\ s$. Analise principalmente os gráficos $x \times z$, $t \times x$ e $t \times \theta$.

(f) Preparação inicial de 1 s com $x_r = 0\ m$ e $z_r = 2\ m$. Então, o multicóptero é comandado a seguir uma curva de Lissajous com expressão

$$\begin{cases} x_r(t) = \cos\left(\dfrac{3}{8}(t-1) + \dfrac{\pi}{2}\right)\ m, \\ z_r(t) = \left[\sin\left(\dfrac{2}{8}(t-1)\right) + 2\right]\ m, \end{cases} \tag{18}$$

por 8 s. Finalmente, comanda-se $x_r = 0\ m$ e $z_r = 2\ m$ por 1 s. Analise principalmente o gráfico $x \times z$.

(g) Semelhante ao experimento 'f', mas o período da curva de Lissajous é de 18 s.

Discuta os resultados obtidos em relação a propriedades de um sistema de controle, especialmente tempo de subida, sobressinal, erro em regime, rejeição à perturbação etc. Ademais, recomenda-se que o aluno crie animações de todos os experimentos para entendê-los melhor. Pede-se discutir qualitativamente no relatório o que é observado nas animações relativas aos experimentos 'd', 'e' e 'f'. Inclua na sua entrega as animações desses três experimentos.

Perceba que o sistema completo é não-linear, logo é esperado um descasamento de modelo entre o modelo de simulação e os modelos lineares usados para projeto. Além disso, o sistema não-linear apresenta acoplamento entre x e z não observado nos modelos lineares de projeto.

3 Instruções

- A primeira etapa do processo de correção consistirá em submeter as funções implementadas a vários casos de teste de forma automatizada. Assim, os cabeçalhos das funções devem ser seguidos **rigorosamente**. Arquivos `.m` com os nomes destas funções e os cabeçalhos já implementados foram fornecidos juntamente com este roteiro. Dê preferência a implementar seu laboratório a partir destes arquivos `.m` fornecidos para evitar erros.

- A entrega da solução desse laboratório consiste de arquivos de código (MATLAB e Simulink) e de um relatório (em `.pdf`), que devem ser submetidos no Google Classroom.

- Compacte todos os arquivos a serem submetidos em um único `.zip` (use obrigatoriamente `.zip`, e **não** outra tecnologia de compactação de arquivos) e anexe esse `.zip` no Google Classroom. Para o `.zip`, use o padrão de nome `<login_ga>_labX.zip`. Por exemplo, se seu login é `marcos.maximo` e você está entregando o laboratório 1, o nome do arquivo deve ser `marcos.maximo_lab1.zip`. **Não** crie subpastas, deixe todos os arquivos na "raiz" do `.zip`.

- O relatório deve ser sucinto, preocupe-se apenas em incluir discussões e entregáveis solicitados no roteiro. Pede-se apenas um cuidado mínimo na elaboração do relatório: responder adequadamente as perguntas, incluir figuras diretamente no relatório (ao invés de deixar como arquivos separados), figuras de boa qualidade, colocar nomes nos eixos dos gráficos, colocar legenda para diferenciar curvas num mesmo gráfico etc.

- **Não** é permitido o uso de funções ou comandos prontos do MATLAB que realizem toda a funcionalidade atribuída a uma certa função cuja implementação foi solicitada. Entretanto, o uso destas funções para verificação das implementações realizadas é encorajado. Em caso de dúvida, consulte o professor.

- A criação de *scripts* e funções auxiliares no MATLAB para execução de experimentos e geração de gráficos é fortemente recomendada para facilitar seu trabalho. Porém, não há necessidade de entregar código auxiliares.

- **Não** há necessidade de copiar e colar o código no seu relatório, dado que você também submeterá os arquivos de código. Também **não** há necessidade de explicar sua implementação no relatório, a **não** ser que o roteiro tenha solicitado explicitamente. Porém, organizar e comentar o código é muito salutar, pois ele será o foco da correção.

4 Dicas

- Nos modelos de simulação entregues, todos os blocos e os parâmetros de simulação já estão adequadamente configurados.

- Além dos blocos Simulink conhecidos de laboratórios anteriores, neste laboratório, usa-se o bloco `PID`, que implementa um compensador PID. Ao abrir a caixa de diálogo do bloco, observa-se que a equação utilizada para o PID é

$$C(s) = P + I\frac{1}{s} + D\frac{N}{1 + N/s}. \tag{19}$$

  Para facilitar a ligação com o visto em teoria, pode-se reescrever essa equação como

$$C(s) = K_p + \frac{K_i}{s} + K_d s\left(\frac{a}{s+a}\right), \tag{20}$$

  em que $K_p = P$, $K_i = I$, $K_d = D$ e $N = a$. Com isso, conclui-se que o MATLAB implementa o termo derivativo da forma como foi comentado na aula sobre controlador PID: coloca-se um filtro passa-baixas de 1ª ordem em série para mitigar o ruído amplificado pelo derivativo. Além disso, perceba que o controlador PID "puro" é não-implementável na prática, porque sua função de transferência é imprópria (tem mais zeros do que polos). Por outro lado, a presença do filtro passa-baixas modifica o comportamento do compensador PID. Para reduzir a influência desse efeito, pode-se a bem distante da origem, de acordo com a ideia de modos dominantes. No caso deste laboratório, usou-se o valor padrão do MATLAB de $N = a = 100\ rad/s$, pois esse já é um valor alto o suficiente a aplicação em questão.

- Para determinar $t_r|_0^{100\%}$ e M_p de um sistema qualquer no MATLAB, use:

  ```
  info = stepinfo(dinamica, 'RiseTimeLimits', [0, 1]);
  ```

  No caso, $t_r|_0^{100\%}$ e M_p são dados pelos campos `RiseTime` e `Overshoot` da *struct* `info` retornada, respectivamente. Observação: perceba que o valor do campo `Overshoot` está em percentagem (i.e. na escala $0 - 100$), enquanto no curso tem-se usado a escala $0 - 1$.

- Para colocar tempo de amostragem em blocos Simulink, deve-se editar a propriedade `Sample Time` desses. Não se preocupe, pois todos os blocos já foram configurados.

- Para usar a saída do bloco `To Workspace` no formato `Structure with Time`:

  ```
  plot(out.x.time, out.x.signals.values)
  ```

- Caso queira chamar o Simulink dentro do MATLAB, use `out = sim('arquivo.slx')`. A saída do Simulink ficará na variável `out` nesse caso.