Instituto Tecnológico de Aeronáutica – ITA
Planejamento e Controle para Robótica Móvel – CM-202
Laboratório 3 – Projeto de Servomotor de Velocidade

Professor: Marcos Ricardo Omena de Albuquerque Maximo

13 de setembro de 2021

Observação: por questões de compatibilidade, este laboratório deve ser feito utilizando Simulink R2021a.

1 Introdução

Neste laboratório, projetar-se-á um controlador PI para um servomotor de velocidade. Este laboratório trabalha com um modelo de um servomotor de velocidade, que é um conjunto formado por um motor elétrico, um sensor de velocidade, uma caixa de redução (formada por engrenagens) e um controlador. Também é possível incrementar mais ainda o servomotor dependendo da aplicação, e.g. pode-se incluir um sensor de corrente para adição de uma malha de corrente. Com isso, tem-se um sistema que autonomamente rastreia uma referência de velocidade angular. Um servomotor de velocidade pode ser usado em diversas aplicações que requerem controlar a velocidade de rotação de um eixo:

- Controlar a velocidade de rotação de uma hélice de propulsão de um avião ou de um barco.
- Controlar as velocidades de rotação das hélices de um quadricoptero.
- Controlar a velocidade de rotação de uma roda de um robô ou de um carro autônomo.

No caso de um sistema autônomo, o servomotor de velocidade recebe referência de um outro sistema de controle. Por exemplo, no caso de um robô, os comandos de velocidade calculados por uma malha de controle de posição criam referências para os servomotores das rodas do robô. Neste laboratório, implementar-se-á um servomotor de velocidade para a roda de um robô. No caso, o exemplo considerado é o servomotor utilizado na roda de um robô diferencial usado na liga Very Small Size (VSS) em competições de futebol de robôs, mostrado na Figura 1.

Conforme já visto nas aulas de teoria, um motor elétrico é um sistema eletromecânico, conforme apresentado no diagrama da Figura 2, e é regido por

$$\begin{cases} J_m\dot{\omega}_m + B_m\omega_m = K_t i = \tau_m, \\ V = L\dot{i} + Ri + K_t\omega_m. \end{cases} \tag{1}$$

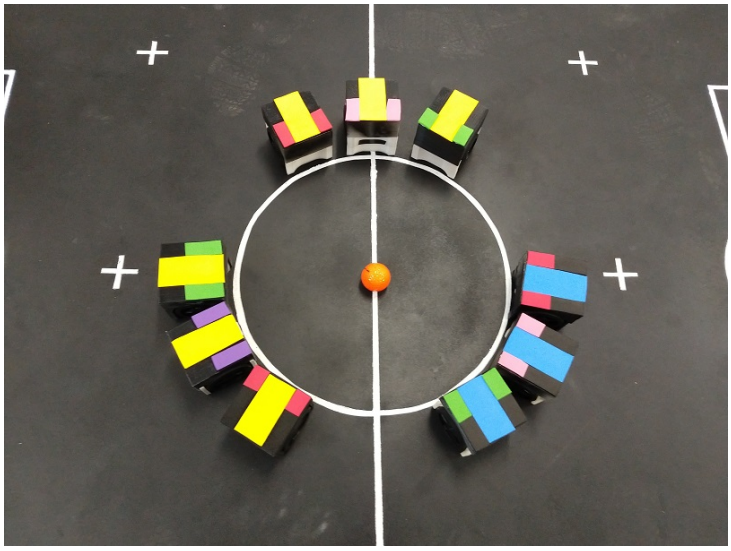

Figura 1: Robôs diferenciais usados na liga Very Small Size (VSS).

em que ω_m é a velocidade de rotação do motor, i é a corrente que circula pelo motor, J_m é a inércia do motor, B_m é o coeficiente de atrito viscoso do eixo do motor, K_t é a constante de torque, τ_m é o torque gerado pelo motor, V é a tensão aplicada nos terminais do motor, L é a indutância e R é a resistência.

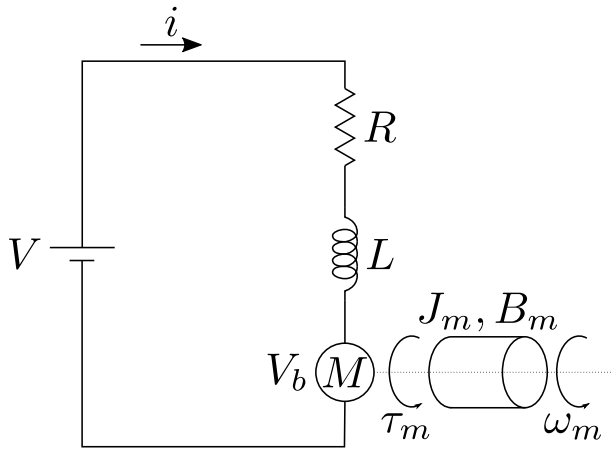

Figura 2: Motor elétrico.

Entretanto, um motor é muito fraco para a maioria das aplicações, de modo que é comum o uso de engrenagens para aumentar o torque na saída. Para compreender esse fenômeno, considere o par de engrenagens apresentado na Figura 3. Conforme a figura sugere, numere como 1 e 2 as engrenagens da esquerda e da direita, respectivamente. As velocidades lineares das duas engrenagens nos pontos de contato dos dentes devem ser iguais:

$$v_1 = v_2, \tag{2}$$

logo

$$\omega_1 R_1 = \omega_2 R_2, \tag{3}$$

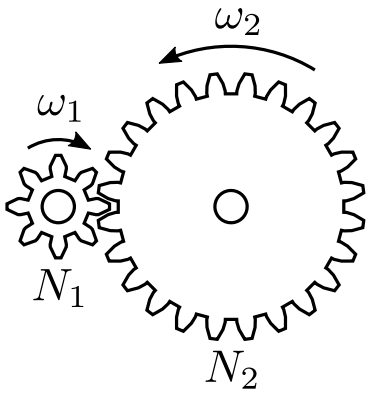

Figura 3: Par de engrenagens.

em que ω_1 e ω_2 são as velocidades angulares das engrenagens, enquanto R_1 e R_2 representam seus respectivos raios. Já N_1 e N_2 denotam os números de dentes das engrenagens. Com os dentes devem ter o mesmo tamanho, tem-se que R_1 e R_2 proporcionais a N_1 e N_2, respectivamente. Portanto

$$\omega_1 N_1 = \omega_2 N_2 \Rightarrow \frac{\omega_2}{\omega_1} = \frac{N_1}{N_2} = \frac{1}{N}, \tag{4}$$

em que

$$N = \frac{N_2}{N_1} \tag{5}$$

é chamado fator de redução, pois o par de engrenagens em geral é utilizado para reduzir a velocidade de rotação do motor. Para entender a vantagem em reduzir a velocidade de rotação, perceba que, por conservação da energia, deve-se ter a mesma potência mecânica nas duas engrenagens:

$$P_1 = P_2 \Rightarrow \tau_1 \omega_1 = \tau_2 \omega_2 \Rightarrow \frac{\tau_2}{\tau_1} = \frac{\omega_1}{\omega_2} = N, \tag{6}$$

ou seja, tem-se uma troca de velocidade por torque (reduz-se a velocidade, mas aumenta-se o torque). Na realidade, há perdas na transmissão de torque, geralmente refletida num parâmetro chamado eficiência da transmissão η, de modo que as relações tornam-se

$$\frac{\omega_2}{\omega_1} = \frac{1}{N}, \ \frac{\tau_2}{\tau_1} = N\eta. \tag{7}$$

Dependendo da aplicação, pode ser necessário o uso de várias engrenagens em série para atingir a redução N desejada, que são organizadas no que se chama de caixa de redução. Com isso, considere que o motor foi acoplado a uma carga (em inglês, *load*) com inércia J_l. No nosso caso, a carga é a roda do robô. Além disso, seja B_l o coeficiente de atrito viscoso do eixo de saída. Como (1) foi escrita sem considerar a presença de carga, deve-se analisar como essa equação deve ser adaptada para levar em conta a carga.

Para isso, perceba que o torque gerado pelo motor deve mover tanto o próprio motor quanto a carga, de modo que pode-se escrever

$$\begin{cases} \tau_m = J_m \dot{\omega}_m + B_m \omega_m + \tau_{t,i}, \\ \tau_{t,o} = J_l \dot{\omega}_l + B_l \omega_l. \end{cases} \tag{8}$$

em que ω_m e ω_l são as velocidades de rotação do motor e da carga, respectivamente, e $\tau_{t,i}$ e $\tau_{t,o}$ são torques de transmissão na entrada e na saída da caixa de redução, respectivamente. Seja N a relação de redução, tem-se

$$\omega_l = \frac{\omega_m}{N}, \ \tau_{t,o} = N\eta\tau_{t,i}. \tag{9}$$

Substituindo (9) em (8), tem-se

$$\begin{cases} \tau_m = J_m\dot{\omega}_m + B_m\omega_m + \tau_{t,i} \\ N\eta\tau_{t,i} = J_l\dfrac{\dot{\omega}_m}{N} + B_l\dfrac{\omega_l}{N} \end{cases} \Rightarrow \underbrace{\left(J_m + \frac{J_l}{N^2\eta}\right)}_{J_{eq}}\dot{\omega}_m + \underbrace{\left(B_m + \frac{B_l}{N^2\eta}\right)}_{B_{eq}}\omega_m = \tau_m, \tag{10}$$

em que J_{eq} e B_{eq} são chamados inércia e coeficiente de atrito viscoso equivalentes, conforme visto pelo motor, respectivamente. Perceba que também há também a equação equivalente conforme vista pela carga:

$$\left(N^2\eta J_m + J_l\right)\dot{\omega}_l + \left(N^2\eta B_m + B_l\right)\omega_l = \tau_l, \tag{11}$$

em que $\tau_l = N\eta\tau_m$ é o torque gerado pelo motor na carga. Nesse caso, define-se também inércia e coeficiente de atrito viscoso equivalentes conforme visto pela carga. Em geral, o que dita se (10) ou (11) será usada para projeto do sistema de controle é se o sensor de velocidade está no eixo do motor ou da carga, i.e. se a velocidade medida para retroalimentação é ω_m ou ω_l. Há vantagens e desvantagens em colocar o sensor no motor ou na carga. No caso do robô considerado, o sensor (*encoder*) está no motor, logo (10) será usada, de modo que tem-se

$$\begin{cases} J_{eq}\dot{\omega}_m + B_{eq}\omega_m = \tau_m, \\ V - K_t\omega_m = L\dot{i} + Ri. \end{cases} \tag{12}$$

Finalmente, seja τ_e um torque externo (distúrbio) aplicado na carga. Transferindo para o motor, as equações que regem a dinâmica do servomotor são

$$\begin{cases} J_{eq}\dot{\omega}_m + B_{eq}\omega_m = \tau_m + \dfrac{\tau_e}{N\eta}, \\ V - K_t\omega_m = L\dot{i} + Ri. \end{cases} \tag{13}$$

em que V é a entrada, ω_m é a saída e τ_e é um entrada de perturbação. De acordo com o discutido, a Figura 4 mostra um diagrama da planta (dinâmica) do servomotor.

Como dito anteriormente, o exemplo considerado é o servomotor de velocidade de um robô diferencial projetado para a liga Very Small Size (VSS) de competições de futebol de robôs. O motor utilizado é um Pololu Micro Metal Gearmotor HP com tensão nominal de 6 V. A redução é implementada por uma caixa de redução com várias engrenagens em série, de modo que $N = 51{,}45$. A carga considerada é uma roda de 60 mm. O sensor de velocidade utilizado é um *encoder* magnético de quadratura Pololu com 12 contagens por revolução (CPR). Todos os parâmetros associado a essa planta são fornecidos em código pela função `getServoParams()`.

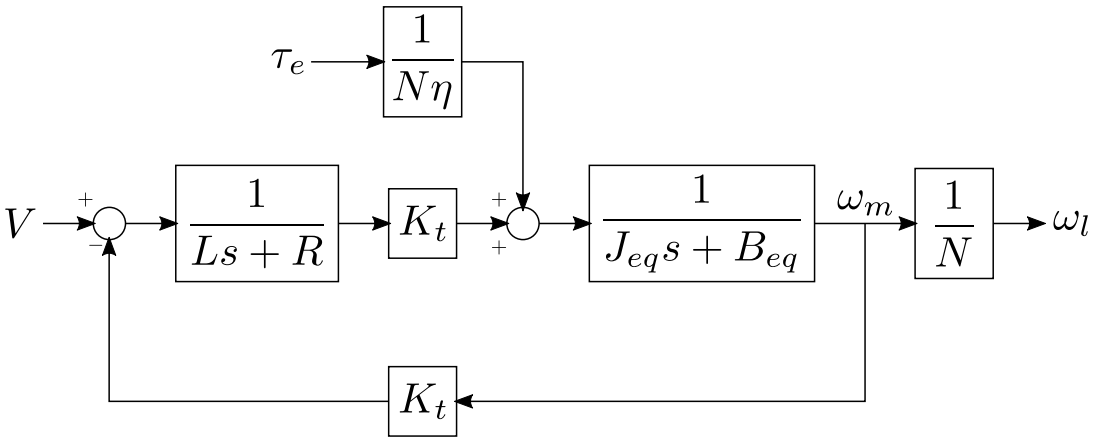

Figura 4: Diagrama de blocos da planta do servomotor.

2 Tarefas

2.1 Obtenção da Função de Transferência da Planta

A primeira tarefa consiste em determinar a função de transferência da planta, com base no diagrama mostrado na Figura 4. Para facilitar, considere que, para o motor em questão, a dinâmica da corrente é muito mais rápida que a dinâmica da mecânica, de modo que pode-se negligenciar o efeito do indutor e considerar $L \approx 0$. Destaca-se que, de fato, para o motor Pololu considerado, esta aproximação é bem válida. Assim, como o sensor (*encoder*) está no eixo do motor, encontre a expressão da seguinte função de transferência:

$$G_m(s) = \frac{\Omega_m(s)}{V(s)} \tag{14}$$

Inclua a expressão no seu relatório e implementa-a na função `getSpeedDynamics()`. A função espera que seu retorno seja um objeto do tipo função de transferência do MATLAB (gerado com a função `tf`). Note que, pelo princípio da superposição, você pode considerar $\tau_e = 0$ durante na obtenção de $G_m(s)$. Caso tenha dificuldade, veja a seção de dicas.

2.2 Controlador PI Analítico

Inicialmente, você implementará um controlador PI para o servomotor de velocidade usando técnicas analíticas geralmente estudadas em cursos básicos de Controle. Lembre que a função de transferência de um compensador PI é

$$C(s) = K_p + \frac{K_i}{s}, \tag{15}$$

em que K_p e K_i são os ganhos proporcional e integrativo, respectivamente. A implementação do controlador do motor no microcontrolador do robô é uma implementação digital. Como estudado em um curso de Controle Digital, a discretização devido à implementação digital introduz um atraso de $T/2$, em que T é o tempo de amostragem. Além

disso, o *encoder* introduz um atraso adicional de $T/2$. Entretanto, para a implementação usando técnicas analíticas, pede-se ignorar os atrasos por enquanto.

O diagrama de blocos do sistema de controle considerado é mostrado na Figura 5. Perceba que, embora a malha fechada considere grandezas no motor, a referência r_l para o servomotor é em termos de velocidade da roda, a qual é convertida para referência de velocidade do motor $r_m = Nr_l$.

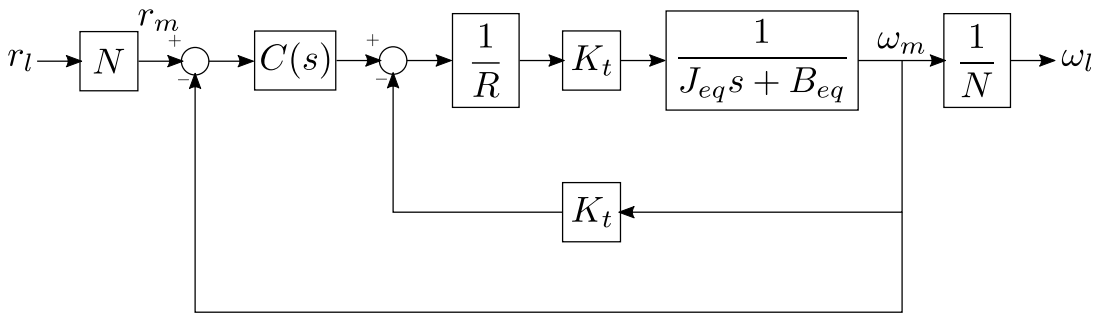

Figura 5: Diagrama de blocos do sistema em malha fechada para projeto analítico.

Determine a função de transferência de malha fechada para o sistema de controle

$$G_f(s) = \frac{\Omega_m(s)}{R_m(s)}. \tag{16}$$

O projeto de sistema de controle considera os seguintes requisitos:

- Banda passante $\omega_b = 30\ Hz$.
- Margem de fase $PM = 50°$.

Para obter os requisitos em código, use a função `getRequirements()`. A função de transferência encontrada é semelhante a um sistema de segunda ordem padrão, cuja expressão é

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}, \tag{17}$$

em que ω_n é a frequência natural e ξ é o fator de amortecimento. Entretanto, $G_m(s)$ tem um zero a mais devido ao controlador PI. Para permitir o projeto usando uma comparação com um sistema de segunda ordem padrão, ignore o zero introduzido pelo PI (i.e. ignore o termo com s no numerador). Além disso, utilize as seguintes fórmulas para converter os requisitos para os parâmetros de um sistema de segunda ordem padrão:

$$\omega_b = \omega_n\sqrt{1 - 2\xi^2 + \sqrt{4\xi^4 - 4\xi^2 + 2}}, \tag{18}$$

$$\xi = \frac{PM(^\circ)}{100^\circ}, \tag{19}$$

em que PM é dada em graus na fórmula acima. Com isso, encontre as expressões para os ganhos K_p e K_i em função dos parâmetros do sistema de segunda ordem (ω_n e ξ) e dos parâmetros do servomotor. Com isso, pede-se:

- Coloque as expressões obtidas para K_p e K_i no seu relatório.
- Implemente o método de projeto no arquivo `designCompensatorAnalytic.m`.
- Avalie o controlador analítico usando a função `analyzeDesignNoDelay()`. Embora esta função não considere os atrasos na malha, o efeito do zero introduzido pelo PI não é desprezado. Inclua os gráficos no relatório e comente sobre atendimento (ou não) aos requisitos.
- Avalie o controlador analítico usando a função `analyzeDesign()`. Neste caso, considera-se também os atrasos introduzidos pela discretização e pelo *encoder*. Inclua os gráficos no relatório e comente sobre atendimento (ou não) aos requisitos.

Para compreender melhor onde os atrasos são introduzidos na malha, verifique o diagrama equivalente contínuo apresentado na Figura 6, em que as funções de transferência dos atrasos são dadas por

$$A_c(s) = A_e(s) = e^{-sT/2}. \tag{20}$$

Lembre que a função de transferência de um atraso τ é $e^{-\tau s}$. Ademais, perceba que os atrasos em código foram gerados com uso de aproximação de Padé de segunda ordem:

$$e^{-\tau s} \approx \frac{s^2 - (6/\tau)s + 12/\tau^2}{s^2 + (6/\tau)s + 12/\tau^2}. \tag{21}$$

Na verdade, em código, a aproximação de Padé está sendo obtida através da função `pade` do MATLAB.

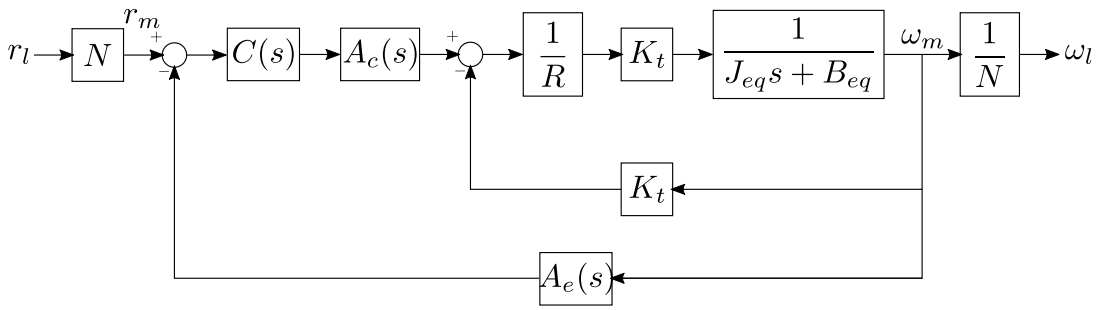

Figura 6: Diagrama de blocos do equivalente contínuo considerando atrasos de discretização e do *encoder*.

2.3 Controlador PI com Otimização

Para refinar a implementação do controlador PI, usa-se otimização com o algoritmo Nelder-Mead através da função `fminsearch` do MATLAB. Faça a implementação no arquivo `designCompensatorOptimization.m`. **Caso não saiba como usar esta função para isso, recomenda-se estudar primeiro o roteiro entregue juntamente com este laboratório**. Use como chute inicial para a otimização, os valores de ganhos calculados na seção anterior. Ademais, use como função de custo:

$$J(K_p, K_i) = \left(\omega_{b,req} - \omega_b(K_p, K_i)\right)^2 + \left(PM_{req} - PM(K_p, K_i)\right)^2, \tag{22}$$

em que $\omega_{b,req}$ e PM_{req} são os requisitos de banda passante e margem de fase, respectivamente. A ideia da otimização é minimizar $J(K_p, K_i)$, de modo a obter os valores de K_p e K_i que fazem o sistema atender aos requisitos o melhor possível.

Para dicas de como obter a banda passante e a margem de fase de uma função de transferência no MATLAB, veja a seção de dicas. Como a função de transferência em questão é relativamente complicada, ao invés de tentar obter a expressão analiticamente e escrevê-la no MATLAB, recomenda-se usar as operações com objetos do tipo função de transferência que já estão implementadas (i.e. pode-se operar normalmente com objetos do tipo função de transferência no MATLAB). Caso o MATLAB fique gerando *warnings* durante a otimização devido ao não cancelamento de polos e zeros, use a função `minreal()` nas funções de transferência geradas.

Finalmente, avalie o controlador projetado aqui usando a função `analyzeDesign()`. Comente sobre o atendimento aos requisitos.

2.4 Implementação Digital

Para discretização do controlador, pede-se usar a transformação de Tustin (bilinear):

$$s \approx \frac{2}{T}\left(\frac{z-1}{z+1}\right), \tag{23}$$

de modo que a expressão discreta do controlador PI se torna

$$C(z) = K_p + K_i\frac{T}{2}\left(\frac{z+1}{z-1}\right), \tag{24}$$

o que gera a seguinte lei de controle discreta:

$$u[k] = u[k-1] + \left(K_p + \frac{K_iT}{2}\right)e[k] + \left(-K_p + \frac{K_iT}{2}\right)e[k-1]. \tag{25}$$

Para definir T no código, cria-se $f_s = 200\ Hz$ em `getRequirements()`, que significa um requisito de taxa de amostragem. Assim, para obter T, basta calcular $T = 1/f_s$.

No projeto do controlador PI, ignorou-se o efeito do indutor. Essa aproximação remove um polo da função de transferência em malha fechada, de modo que essa assume uma forma de sistema de segunda ordem padrão, o que é muito conveniente para projeto. Pode-se justificar negligenciar o indutor como uma aproximação por polos dominantes.

Além disso, até o momento ignorou-se o fato de que a tensão comandada para o motor é limitada pela fonte de tensão utilizada, que no caso do robô é uma bateria de lítio polímero (LiPo) de 2 células. Dependendo da carga da bateria, sua tensão varia de 7,4 a 8,4 V. No código, considera-se a bateria cheia, i.e. usa-se $V_{max} = 8{,}4\ V$.

Como a técnica de projeto assume um modelo linear, não é possível levar em conta essa saturação durante o projeto. Esse descasamento é acentuado quando há um integrador na malha, de modo que surge um efeito chamado *windup*. O *windup* surge porque o integrador acumula mais erro do que deveria, dado que a saturação faz com que a referência seja atingida mais lentamente do que o esperado pelo modelo linear.

Esse acúmulo excessivo de erro no integrador aumenta o *overshoot* e pode até mesmo desestabilizar o controlador. Para mitigar o problema, deve-se implementar um *anti-windup*. Quando se tem um controlador em formato digital como mostrado em (25), uma

forma muito simples de se implementar um *anti-windup* consiste em limitar explicitamente o valor do comando $u[k]$. No caso, aplica-se

$$u[k] = \begin{cases} u[k], & -V_{max} \leq u[k] < V_{max}, \\ -V_{max}, & u[k] < -V_{max}, \\ V_{max}, & u[k] > V_{max}. \end{cases} \tag{26}$$

logo após o cálculo de $u[k]$ de acordo com (25). Com isso, esse valor limitado é usado como $u[k-1]$ na próxima iteração, evitando o acúmulo de integral. Implemente o controlador digital, incluindo a estratégia de *anti-windup*, na `MATLAB Function` `compensator` de `servomotor.slx`. O restante do Simulink `servomotor.slx` já está completamente implementado. Com isso, pede-se:

- Use a função `simulateServo()` para simular o servomotor com degraus de 40, 70 e 100 rad/s (referências para velocidade da roda). Inclua os gráficos gerados no seu relatório e comente o que observa, em especial em relação ao degrau de 100 rad/s.
- Use a função `evaluateAntiWindup()` para avaliar a implementação do anti-windup. Comente sobre o que é observado, especialmente levando em conta o limite de tensão da bateria.

3 Instruções

- 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 como um único arquivo `.zip`.
- Compactar 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.
- Use o padrão de nome `<login_ga>_labX.zip`, em que `<login_ga>` é seu login `@ga.ita.br` no Classroom e X é o número do laboratório em questão. Por exemplo, se o login do aluno for `marcos.maximo` e estiver entregando o laboratório 1, o nome do arquivo deve ser `marcos.maximo_lab1.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), usar 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 códigos 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.
- Os seguintes blocos do Simulink são utilizados:
  - `Step`: implementa degrau.
  - `Sum`: soma/subtrai sinais.
  - `Gain`: multiplica um sinal por uma constante.
  - `MATLAB Function`: permite escreve um código em MATLAB para ser executado dentro do Simulink.
  - `To Workspace`: copia sinais do Simulink para o `workspace` do MATLAB.
  - `Saturation`: satura um sinal.
  - `Quantizer`: quantiza o valor de um sinal.
  - `Transport Delay`: implementa atraso de transporte.
  - `Delay`: implementa atraso discreto (de um tempo de amostragem).
  - `Transfer Fcn`: define uma função de transferência contínua.
- 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.
- A função `pade` do MATLAB retorna o numerador e denominador da função de transferência. Assim, para que fique no formato de um objeto do tipo função de transferência, faça:

  ```
  [num, den] = pade(T, 2);
  atraso = tf(num, den);
  ```

- Muitas vezes, o MATLAB não realiza cancelamento polo-zero durante operações com funções de transferência. Para forçar que ele o faça, use `G = minreal(G)`. Isso é importante para evitar que o MATLAB fique emitindo vários *warnings* durante a otimização.

- Durante a otimização, pode ser que o algoritmo escolha ganhos que tornam o sistema instável. Isso fará com que o MATLAB emita um *warning*. Porém, isso em geral não gera problemas, pois o otimizador naturalmente encontra ganhos adequados à medida que prossegue. Basicamente, preocupe-se se a solução encontrada atende aos requisitos, mesmo que o MATLAB emita *warnings* durante a otimização.

- Para definir uma função de transferência no MATLAB, use `sys = tf(num, den)`, em que `num` e `den` são os polinômios do numerador e do denominador.

- Outra forma conveniente de criar função de transferência no MATLAB é fazer:

```
s = tf('s');
G = (5 * s + 1) / (s^2 + 2 * s + 10);
```

- Para obter banda passante de uma função de transferência no MATLAB, use `wb = bandwidth(G)`.

- A função `margin` retorna `GM`, `PM`, `Wcg` e `Wcp`. Caso precise apenas de `PM` e `Wcp`, use `[~, PM, ~, Wcp] = margin(Ga)`. Caso precise apenas de `PM`, faça `[~, PM, ~, ~] = margin(Ga)`.

- No caso de ter-se um sistema como representado na Figura 7, lembre-se que a função de transferência da malha fechada é dada por

$$G_f(s) = \frac{Y(s)}{R(s)} = \frac{G_a(s)}{1 + H(s)G_a(s)}. \tag{27}$$

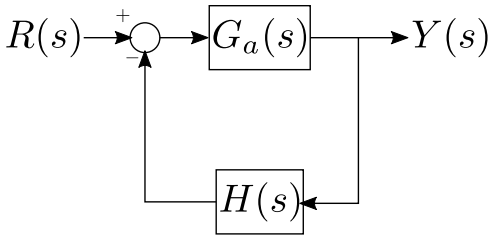

Figura 7: Diagrama de blocos de um sistema em malha fechada com dinâmica de sensor.

- Levando em conta a notação do item anterior, lembre-se que a banda passante é medida usando $G_f(s)$, enquanto a margem de fase é medida usando $G_a(s)H(s)$.