Instituto Tecnológico de Aeronáutica – ITA

Sistemas de Controle Contínuos e Discretos – CMC-12

Laboratório 6 – Projeto de Servomotor de Posição com Malha de Corrente

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 servomotor de posição. Além da verificação por simulação da teoria aprendida nas aulas teóricas, o laboratório tem como objetivo abordar as seguintes questões:

- Levar em conta efeitos de malhas internas, filtros e atrasos introduzidos por discretização do projeto do sistema de controle.
- Projetar controladores através de otimização paramétrica.
- Avaliar por simulação o efeito de quantização sobre o sistema de controle.
- Discretização de controladores através do método de Tustin.

Este laboratório trabalha com um modelo de um servomotor de posição, que é um conjunto formado por um motor elétrico, um sensor de posição, 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 velocidade para obtenção de medida de velocidade com maior qualidade. Com isso, tem-se um sistema que autonomamente rastreia uma referência de posição angular. Um servomotor de posição pode ser usado em diversas aplicações que requerem controlar a posição de um eixo:

- Controlar as posições das superfícies de controle (aileron, profundor e leme) de um avião.
- Controlar as posições das juntas de um manipulador robótico.
- Controlar as posições das juntas de um robô humanoide.

No caso de um sistema autônomo, o servomotor de posição recebe referência de um outro sistema de controle. Por exemplo, no caso do avião, como o do Laboratório 4, os comandos de posição das superfícies de controle são calculados pelo autopiloto.

Conforme visto no Laboratório 3, um motor elétrico é um sistema eletromecânico, conforme apresentado no diagrama da Figura 1. Além disso, conforme foi mencionado, pode-se considerar as equações do motor conforme visto "pelo motor" ou "pela carga". Qual das duas abordagens de modelagem deve ser considerada para projeto do sistema de controle depende geralmente de onde se encontra o sensor.

No caso do Laboratório 3, usou-se as equações conforme vistas "pelo motor", pois o sensor de velocidade (*encoder*) no robô GLaDOS da ITAndroids está de fato "do lado do motor". Porém, no caso de um servomotor de posição, é comum colocar o *encoder* no lado da carga para que a medida de posição leve em conta imperfeições na transmissão, como flexibilidade e folga mecânica (transmissão realizada por engrenagens costuma ter muita folga mecânica). Portanto, as equações utilizadas neste laboratório são

$$\begin{cases} \underbrace{\left(N^2\eta J_m + J_l\right)}_{J_{eq}}\ddot{\theta}_l + \underbrace{\left(N^2\eta B_m + B_l\right)}_{B_{eq}}\dot{\theta}_l = \tau_l = N\eta\tau_m = N\eta K_t i, \\ V - K_t N\dot{\theta}_l = L\dot{i} + Ri, \end{cases} \tag{1}$$

em que θ_l é a posição da carga (*load*), $\dot{\theta}_l = \omega_l$ é a velocidade da carga, i é a corrente que circula pelo motor, J_m é a inércia do motor, J_l é a inércia da carga, B_m é o coeficiente de atrito viscoso do eixo do motor, B_l é o coeficiente de atrito viscoso do eixo da carga, J_{eq} é a inércia equivalente vista do lado da carga, B_{eq} é o coeficiente de atrito viscoso equivalente visto do lado da carga, K_t é a constante de torque, τ_m é o torque gerado pelo motor, τ_l é o torque transmitido para a carga, V é a tensão aplicada nos terminais do motor, L é a indutância do motor, R é a resistência do motor, N é a redução da transmissão e η é a eficiência da transmissão. A Figura 2 apresenta um diagrama de blocos da planta. Caso deseje relembrar a dedução dessas equações, reveja a Introdução do Laboratório 3.

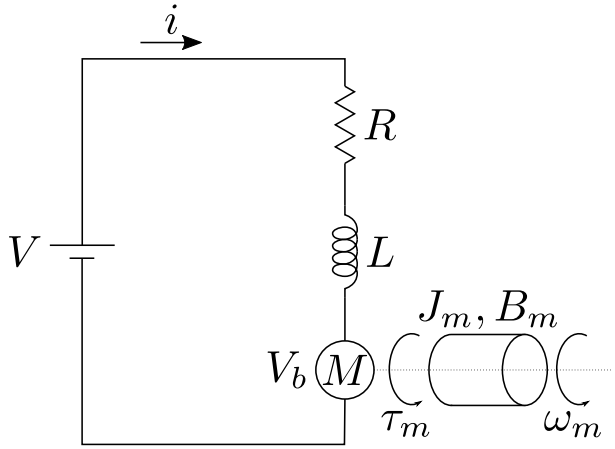

Figura 1: Motor elétrico.

Os parâmetros de servomotor considerados neste laboratório são os mesmos do Laboratório 3. A diferença está na aplicação do servomotor, dado que neste laboratório,

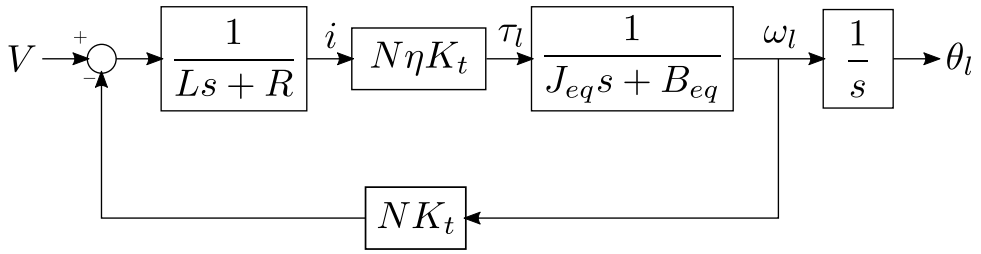

Figura 2: Diagrama de blocos da planta do servomotor de posição, conforme visto pela carga.

está-se interessado em controle de posição. Considera-se que o sensor utilizado é um *encoder* absoluto localizado no lado da carga. Não há sensor dedicado para medida da velocidade, o que é uma situação comum. Por outro lado, assume-se disponível um sensor de corrente para implementação de malha de corrente. Todos os parâmetros associados à planta são fornecidos em código pela função `obterPlantaServoPosicao`. Esta função calcula J_{eq} e B_{eq} conforme vistos pela carga.

2 Tarefas

2.1 Projeto da Malha de Corrente

A principal vantagem em se utilizar uma malha de corrente é a possibilidade de aumentar a banda passante da dinâmica da corrente em relação à banda passante natural ditada pelo circuito elétrico. Com isso, pode-se utilizar uma banda passante maior na malha de posição, sem que a dinâmica da corrente se torne limitante. Pede-se usar o seguinte compensador para a malha de corrente:

$$C_c(s) = K\frac{T_l s + 1}{s\left(\alpha T_l s + 1\right)} = K\left(\frac{T_l s + 1}{\alpha T_l s + 1}\right)\left(\frac{1}{s}\right), \tag{2}$$

em que K, α e T_l são parâmetros do compensador. Trata-se de um compensador *lead* combinado em série com um integrador. Perceba que a planta da corrente é

$$G_c(s) = \frac{1}{Ls + R}, \tag{3}$$

que não possui polo na origem, logo o integrador em $C_c(s)$ é interessante para tornar o sistema do tipo 1 e garantir erro nulo em regime para entrada degrau unitário. Ademais, para a malha de corrente, a força contraeletromotriz atua como uma perturbação, assim o integrador também elimina o erro em regime que seria introduzido pela força contraeletromotriz. Por outro lado, o integrador tende a instabilizar o sistema e o *lead* é usado para recuperar parte da margem de fase perdida.

Um diagrama de blocos da malha de corrente (com controlador discreto $C_c(z)$) é apresentado na Figura 3. Como as técnicas de projeto de CMC-12 trabalham apenas com sistemas contínuos, o equivalente contínuo dessa malha é apresentado na Figura 4, em que $A_c(s) = e^{-sT_c/2}$ representa o atraso introduzido pela discretização do controlador com período de amostragem T_c.

Considere os seguintes requisitos para a malha da corrente:

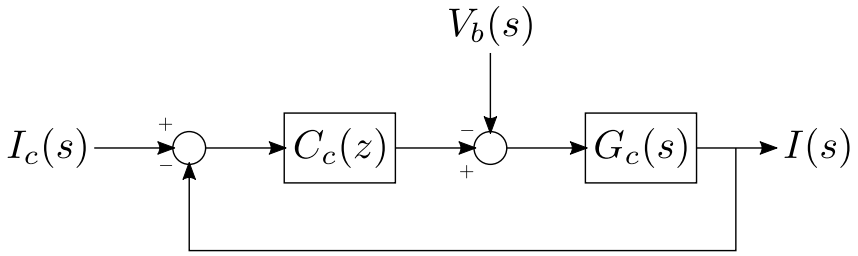

Figura 3: Diagrama de blocos da malha de corrente com controlador discreto.

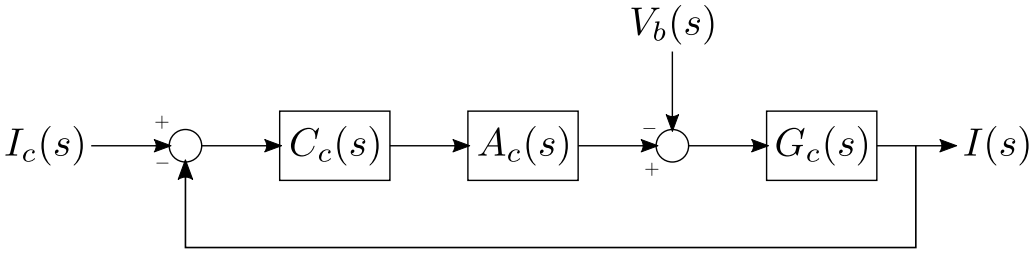

Figura 4: Diagrama de blocos do equivalente contínuo da malha de corrente para fins de projeto com técnicas de tempo contínuo.

- Banda passante $\omega_b \geq 1000\ Hz$.
- Margem de ganho $GM \geq 11\ dB$.
- Margem de fase $PM \geq 60°$.
- Taxa de amostragem $f_s = 20\omega_b = 20\ kHz$ (regra de bolso) para discretização.

Os requisitos para ambas as malhas (corrente e posição) podem ser obtidos através da função `obterRequisitos`. Sobre o projeto, pede-se:

- Inicialmente, projete o compensador de forma analítica de forma semelhante ao visto em sala. Para facilitar, considere que o integrador faz parte da planta e então projete o *lead* através do procedimento visto em sala. Primeiramente, determine K para atendimento ao requisito de banda passante. Então, projete o compensador *lead* para fornecer o avanço de fase ϕ_{max} necessário na frequência de cruzamento ω_{cp} para atendimento ao requisito de margem de fase. **Não** inclua um valor adicional a ϕ_{max}, pois o método de projeto ainda será refinado por um método numérico posteriormente. Implemente o método de projeto na função `projetarControladorCorrenteAnalitico`. Ademais, despreze o atraso introduzido pela discretização. Dado que este é um laboratório, fique livre para usar comandos do MATLAB para o projeto (e.g. `margin`).
- Implemente a função `obterMalhaCorrente`, que retorna as funções de transferência de malha aberta $G_a(s)$ e malha fechada $G_f(s)$ do equivalente contínuo da malha de corrente (Figura 4). Inclua no seu relatório as expressões de $G_a(s)$ e $G_f(s)$ considerando os parâmetros e funções de transferência presentes na Figura 4. Neste caso, você deve considerar todos os efeitos relevantes. Use aproximação de Padé de 2ª ordem (função `pade` do MATLAB) para obter a função de transferência do

atraso introduzido pela discretização. Assuma que o tempo de computação do compensador é insignificante e que o sensor de corrente é ideal. **Dicas:**

  - Aproveite-se de operações com objetos de função de transferência e de funções como `feedback` para facilitar.
  - Após operações usando objetos do tipo função de transferência, use o comando `minreal` para obrigar o MATLAB a fazer cancelamentos polo-zero.

- Com `obterMalhaCorrente` implementada, use a função `avaliarMalhaCorrente` para avaliar o projeto do controlador de corrente através do método analítico. Inclua no seu relatório os gráficos gerados. Comente quais requisitos foram atendidos e quais não foram.

- Apesar da dinâmica da corrente possuir apenas um polo, quando considera-se as funções de transferência do compensador e da aproximação de Padé do atraso, tem-se três zeros e cinco polos em $G_a(s)$ e $G_f(s)$, de modo que um projeto que leve em conta todas as dinâmicas envolvidas através de técnicas analíticas torna-se muito complicado. Uma ideia inicial seria refinar o projeto analítico realizado de forma aproximada através de busca em grade, conforme foi feito no Laboratório 4, porém essa é uma estratégia muito ineficiente. Ao invés disso, uma estratégia muito melhor é fazer a busca pelos parâmetros adequados através de um algoritmo de otimização. Para projeto de sistemas de controle, um algoritmo de otimização popular é o Nelder-Mead, pois ele já está implementado na função `fminsearch` do MATLAB. O Nelder-Mead pertence à classe dos algoritmos de otimização metaheurísticos, que são geralmente estudados pela área de Inteligência Artificial (IA). **Recomenda-se estudar o tutorial em separado entregue com esse roteiro sobre projeto com otimização**. Assim, implemente o projeto com otimização na função `projetarControladorCorrenteOtimizacao`. Considere as funções de transferência $G_a(s)$ e $G_f(s)$ retornadas por `obterMalhaCorrente`. Como o algoritmo Nelder-Mead facilmente fica preso em mínimo local, use o resultado do projeto analítico como chute inicial para a otimização. Use uma função de custo que penaliza o erro quadrático em relação à banda passante e à margem de fase desejadas (ignore o requisito de margem de ganho no projeto):

$$J_c(K, \alpha, T_l) = (\omega_{b,req} - \omega_b(K, \alpha, T_l))^2 + (PM_{req} - PM(K, \alpha, T_l))^2 . \qquad (4)$$

  em que $\omega_{b,req}$ é o requisito de banda passante, PM_{req} é o requisito de margem de fase, ω_b é a banda passante determinada com $G_f(s)$ e PM é a margem de fase obtida por $G_a(s)$. **Dica:** use as funções `bandwidth` e `margin` para determinar banda passante e margens de estabilidade, respectivamente.

- Use a função `avaliarMalhaCorrente` para avaliar o projeto do controlador de corrente através do método com otimização. Inclua no seu relatório os gráficos gerados. Verifique atendimento a todos os requisitos.

2.2 Projeto da Malha de Posição

No caso de estar disponível também um sensor de velocidade, é interessante a implementação do sistema de controle do servomotor de posição com 3 malhas aninhadas:

corrente, velocidade e posição. Porém, como foi considerado que há apenas sensor de posição disponível, pede-se usar um controlador PD com filtro no termo derivativo para a malha de posição:

$$C_p(s) = K_p + K_d s \left(\frac{a}{s+a} \right) \tag{5}$$

Um diagrama de blocos do servomotor (com sistema de controle em tempo discreto) é apresentado na Figura 5. Ademais, um equivalente de tempo contínuo é apresentado na Figura 6, em que $A_p(s) = e^{-sT_p/2}$ representa o atraso introduzido pela discretização do controlador de posição com período de amostragem T_p.

Considere os seguintes requisitos para a malha de posição:

- Banda passante $\omega_b \geq 10\ Hz$.
- Margem de ganho $GM \geq 11\ dB$.
- Margem de fase $PM \geq 60°$.
- Taxa de amostragem $f_s = 1000\ Hz$.

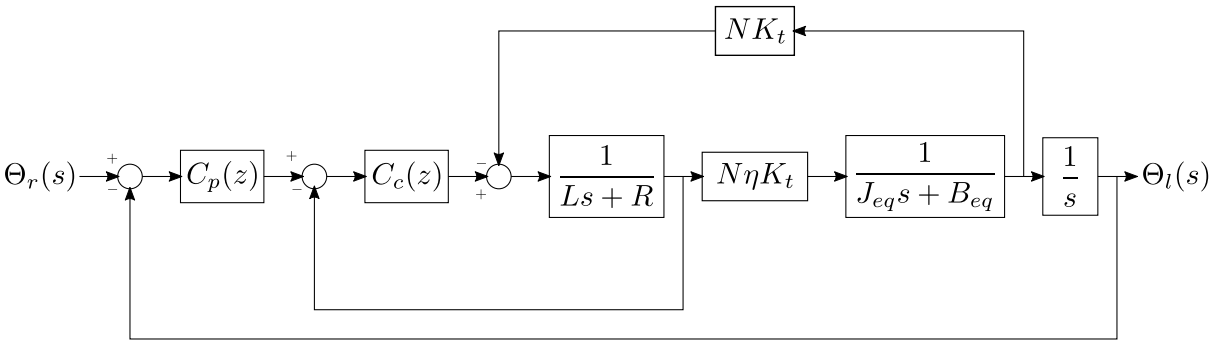

Figura 5: Diagrama de blocos da malha de controle do servomotor com controladores discretos.

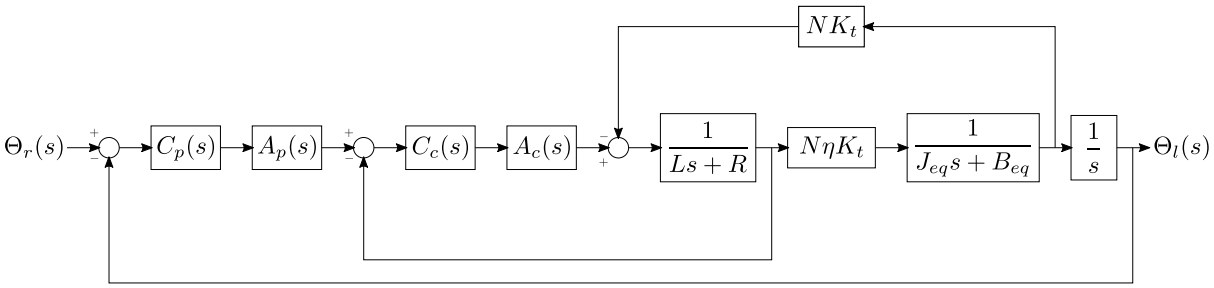

Figura 6: Diagrama de blocos do equivalente contínuo da malha de controle do servomotor para fins de projeto com técnicas de tempo contínuo.

Pede-se:

- Inicialmente, projete o compensador de forma analítica. Considere que a malha interna de corrente é instantânea (i.e. $I(s)/I_c(s) \approx 1$) e elimina completamente o efeito da força contraeletromotriz. Além disso, ignore o fato que o PD introduz um zero (i.e. ignore o termo $K_d K_t s$ que surge no numerador da malha fechada), o filtro do termo derivativo e o atraso da discretização, de modo que o sistema em malha fechada torne-se de segunda ordem padrão. Com isso, projete os ganhos K_p e K_d para que o sistema tenha $\omega_{n,req}$ e ξ_{req} tal que

$$\xi_{req} = \frac{PM_{req}}{100}, \tag{6}$$

$$\omega_{n,req} = \frac{\omega_{b,req}}{\sqrt{1 - 2\xi_{req}^2 + \sqrt{4\xi_{req}^4 - 4\xi_{req}^2 + 2}}}, \tag{7}$$

  em que $\omega_{b,req}$ e PM_{req} são os requisitos de banda passante e de margem de fase, respectivamente. Finalmente, faça $a = 10\omega_{b,req}$ para que o filtro não afete muito a dinâmica do sistema.

- Implemente a função `obterMalhaPosicao`, que retorna as funções de transferência de malha aberta $G_a(s)$ e malha fechada $G_f(s)$ da malha de posição. Neste caso, você deve considerar todos os efeitos relevantes (vide Figura 6): atraso devido a discretizações (posição e corrente), presença do zero do PD, presença do filtro no termo derivativo e malha interna de corrente (**não** despreze o efeito da força contraeletromotriz). Inclua no seu relatório as expressões de $G_a(s)$ e $G_f(s)$ considerando os parâmetros e as funções de transferências mostradas na Figura 6. Use aproximação de Padé de 2ª ordem (função `pade` do MATLAB) para obter as funções de transferência dos atrasos. Assuma que o tempos de computação são insignificantes e que o sensores são ideais. **Dica:** além da dica do `minreal`, separe $A_c(s) = N_c(s)/D_c(s)$ em $N_c(s)$ e $D_c(s)$ na escrita da função de transferência da malha interna de corrente (caso contrário, o MATLAB reclama que o sistema é instável durante o projeto com otimização).

- Com `obterMalhaPosicao` implementada, use a função `avaliarMalhaPosicao` para avaliar o projeto do controlador de posição através do método analítico (projete a malha de corrente com otimização). Inclua no seu relatório os gráficos gerados. Comente quais requisitos foram atendidos e quais não foram.

- A dinâmica completa do servomotor de posição, considerando todos efeitos presentes na malha é muito complicada, logo recorre-se novamente ao uso de otimização metaheurística para sintonia dos ganhos. Assim, implemente o projeto com otimização na função `projetarControladorPosicaoOtimizacao`. Considere as funções de transferência $G_a(s)$ e $G_f(s)$ retornadas por `obterMalhaPosicao` (e um controlador de corrente já projetado). Como o algoritmo Nelder-Mead facilmente fica preso em mínimo local, use o resultado do projeto analítico como chute inicial para a otimização. Use uma função de custo que penaliza o erro quadrático em relação à banda passante e à margem de fase desejadas (ignore o requisito de margem de ganho no projeto):

$$J_p(K_p, K_d) = (\omega_{b,req} - \omega_b(K_p, K_d))^2 + (PM_{req} - PM(K_p, K_d))^2 . \tag{8}$$

em que $\omega_{b,req}$ é o requisito de banda passante, PM_{req} é o requisito de margem de fase, ω_b é a banda passante determinada com $G_f(s)$ e PM é a margem de fase obtida por $G_a(s)$.

- Use a função `avaliarMalhaPosicao` para avaliar o projeto do controlador de corrente através do método com otimização. Inclua no seu relatório os gráficos gerados. Verifique atendimento a todos os requisitos.

2.3 Implementação Digital

Para implementação digital, é necessário fazer a conversão das funções de transferência dos controladores para o domínio discreto. Use o método de Tustin para discretizar os controladores de posição e de corrente nas funções `discretizarControladorPosicao` e `discretizarControladorCorrente`, respectivamente. As saídas dessas funções são funções de transferência no domínio z (da transformada Z), i.e. $C_p(z)$ e $C_c(z)$. **Dica:** use `c2d(C, T, 'Tustin')` no MATLAB para discretizar uma função de transferência `C` através do método de Tustin com período de amostragem `T`. Não há necessidade de realizar a transformação analiticamente.

Ademais, destaca-se que a rigor é necessário implementar estratégias de *anti-windup* nos controladores (em especial, no de corrente, que possui integrador), porém isso não será feito nesse caso por simplicidade. A função `projetarControladorServoOtimizacao` projeta o controlador discreto completo (posição e corrente) através de otimização. Essa função também salva o arquivo `controlador.mat`, que deve ser entregue juntamente com os demais arquivos.

2.4 Implementação do Modelo Simulink

Para simulação do servomotor de posição com controlador digital, implemente o modelo Simulink `servomotor_posicao.slx`. Use a Figura 5 para guiar sua implementação. Além disso, esse modelo inclui um bloco do tipo `Quantizer`, usado para simular o efeito de quantização introduzido pelo *encoder*. No caso de um *encoder* absoluto de posição, sua quantização é ditada pelo número de *bits* usado na conversão analógico-digital. Como o objetivo do sensor é medir todo o intervalo de rotação de 2π (360°), a quantização do *encoder* é dada por

$$Q = \frac{2\pi}{2^{N_b}}, \tag{9}$$

em que N_b é o número de bits usado na conversão analógico-digital. Há também um bloco de saturação para limitar o **comando** de tensão. A função `obterPlantaServoPosicao` considera $N_b = 10$ bits. Note que a quantização afeta a **medida de posição**. Finalmente, destaca-se que o modelo possui os seguintes blocos do tipo `To Workspace`:

- `thetal`: posição angular na carga.
- `thetam`: posição angular medida pelo *encoder* (após quantização).
- `wl`: velocidade angular na carga.
- `ic`: corrente comandada pelo compensador de posição.

- `i`: corrente no circuito do motor.
- `Vc`: tensão comandada pelo compensador de corrente.
- `V`: tensão aplicada nos terminais do motor (após saturação).

A função `simularRespostaDegrau` realiza a simulação desse sistema para uma entrada degrau unitário. Para cada um dos gráficos, comente se o comportamento está de acordo com o esperado qualitativamente. Comente sobre o efeito da quantização nos comportamentos das curvas (pode ser interessante dar um *zoom* na figura para avaliar melhor). Também comente sobre a diferença entre `Vc` e `V`, destacando a relação com o efeito de *windup*.

Observação: devido à alta banda passante da malha de corrente, essa simulação deve ser executada com um passo de simulação pequeno. O modelo Simulink entregue já está configurado adequadamente. Ademais, o pequeno passo de simulação faz com que a simulação seja um pouco lenta.

2.5 Avaliação da Quantização do *Encoder*

A função `avaliarQuantizacaoEncoder` realiza uma comparação da reposta ao degrau unitário para diferentes valores de N_b. Pelos gráficos gerados, comente qualitativamente como a resposta do sistema é degradada à medida que a quantização do *encoder* aumenta. Como foi destacado em sala, a quantização é um efeito não-linear, logo sua influência sobre o sistema dinâmico geralmente é avaliada através de simulação, como feito aqui.

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 `Quantizer`, que produz quantização de um sinal.

- 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.

- 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)`.

- 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);
```