Instituto Tecnológico de Aeronáutica – ITA
Controle para Sistemas Computacionais – CMC-12
Laboratório 1 – Simulação de Sistema de *Cruise Control*

Professor: Marcos Ricardo Omena de Albuquerque Maximo

13 de março de 2020

Observação: por questões de compatibilidade, este laboratório deve ser feito utilizando Simulink R2019b.

1 Introdução

O objetivo desse laboratório é familiarizar o aluno com simulação de sistemas dinâmicos através do Simulink. Em específico, utilizar-se-á o sistema de *cruise control*, que foi extensivamente discutido em sala de aula. Como a maior parte da teoria relativa a esse laboratório já foi apresentada em sala de aula, há pouca discussão de teoria neste roteiro.

2 Tarefas

2.1 Identificação do Sistema

Como visto em sala de aula, a dinâmica da planta do *cruise control* é

$$m\dot{v} + bv = u, \tag{1}$$

em que m é a massa do carro, b é a constante de amortecimento devido à resistência do ar, v é a velocidade do carro e u é a força gerada através do acelerador ou do freio. Como visto em aula, a solução dessa EDO para uma condição inicial $v(0) = 0$ é

$$v(t) = \frac{f}{b}\left(1 - e^{-\frac{b}{m}t}\right). \tag{2}$$

Suponha que não se sabe os parâmetros m e b do carro para o qual se deseja projetar um sistema de *cruise control*. Uma forma de se determinar esses parâmetros é através de uma técnica chamada **identificação de sistemas**, que consiste em executar um experimento e determinar os parâmetros através da resposta dinâmica encontrada. Há diversas formas de se realizar identificação de sistemas, algumas bem sofisticadas. Nesse laboratório, será utilizada uma forma bem básica, baseada no conhecimento do comportamento de um sistema de 1ª ordem.

Considere um experimento em que o carro começa em repouso e uma força f constante é aplicada através do acelerador. Então, um velocímetro de altíssima precisão é utilizado para medir a velocidade do carro a cada $0,1$ s. Conforme visto em aula, a constante de tempo do *cruise control* é

$$\tau = \frac{m}{b}, \tag{3}$$

enquanto a velocidade em regime permanente é

$$\lim_{t\to\infty} v(t) = v_\infty = \frac{f}{b}. \tag{4}$$

Desenvolva uma função em MATLAB que utiliza (3) e (4) e os dados obtidos no experimento para determinar os parâmetros m e b do sistema. Para isso, foi entregue um arquivo chamado `identificarCruiseControl.m` como uma base para essa implementação. Nesse arquivo, deve-se implementar a seguinte função:
`[m, b] = identificarCruiseControl(f, t, v)`: utiliza identificação de sistemas para determinar parâmetros da planta do *cruise control*. Assume que foi realizado um experimento com força `f` constante (escalar) com o carro começando em repouso. Nesse experimento, a velocidade do carro foi medidas nos tempos `t` (`t` é um vetor) e os resultados foram armazenadas no vetor `v`. As saídas da função são os parâmetros da planta: a massa `m` e o fator de amortecimento `b`. **Observação:** perceba que o objetivo dessa função é apenas determinar `m` e `b` isoladamente, ela não deve carregar o arquivo de dados nem gerar gráficos.

Além disso, foi entregue o arquivo `data.mat` com os dados do experimento. Considere todos os dados em SI (sistema internacional de unidades). Assim, realize a identificação de sistemas com esses dados e compare as respostas do experimento e do modelo através de gráficos de $v(t)$ para cada caso. Trace os dois gráficos (experimento e modelo) utilizando o MATLAB, preferencialmente numa única figura. Lembre-se de deixar o gráfico minimamente apresentável: nomes nos eixos, legenda etc.

Para esse item, entregue o arquivo `identificarCruiseControl.m` e inclua o gráfico da comparação no seu relatório.

2.2 Controle em Malha Aberta

Conforme visto em sala, pode-se considerar também uma perturbação no sistema através de uma força constante d. O diagrama de blocos associado a esse sistema de controle é mostrado na Figura 1. Conforme visto em aula, dada uma velocidade de referência v_r, pode-se atingir v_r através de um algoritmo de controle em malha aberta que escolhe $u = b_c v_r$. Faça uma simulação em Simulink para implementar esse diagrama de blocos. Para isso, use o arquivo `cruise_control_aberta.slx`. **Não** altere os nomes ou tipos dos blocos ou o nome do arquivo do Simulink (por conta do auto-corretor).

Utilizando a simulação construída no Simulink (use `Stop Time` como 100 s) (caso não seja mencionado, considere uma estratégia de malha aberta com u constante e que não há erro de modelo nem distúrbio):

(a) Considerando que não há erro de modelo e $d = 0$, faça simulações de controle em malha aberta para atingir $v_r = 10$ m/s, $v_r = 20$ m/s e $v_r = 30$ m/s. Trace os 3 gráficos numa mesma figura e comente o resultado obtido com base no que foi visto em aula.

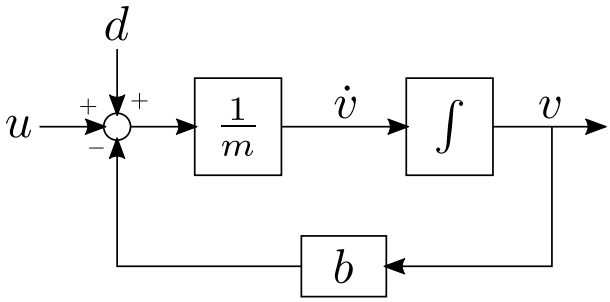

Figura 1: Diagrama do sistema de *cruise control* em malha aberta com perturbação.

(b) Considerando que não há erro de modelo e $d = 0$ e que a força máxima gerada pelo motor é $f_{max} = 2000\ N$, implemente a **estratégia de tempo mínimo** mostrada em sala para atingir $v_r = 10\ m/s$ no menor tempo possível. Faça um gráfico mostrando a resposta do sistema nesse caso. É provável que sua simulação fique esquisita com o *solver* de passo variável usado por padrão no Simulink, assim veja na Seção 4 como alterar a simulação para usar um *solver* de passo fixo.

(c) Considerando que não há erro de modelo e $v_r = 10\ m/s$, faça simulações com os seguintes valores de perturbação: $d = 100\ N$, $d = 200\ N$ e $d = 300\ N$. Trace os 3 gráficos numa mesma figura e comente o resultado obtido conforme o discutido em aula.

(d) Considere que um passageiro de 100 kg entrou no carro. Simule o efeito provocado por esse aumento de massa e discuta a diferença em relação ao caso nominal quando $v_r = 10\ m/s$.

(e) Considere que após o sistema de controle já ter sido projetado, o carro sofreu uma colisão e por conta disso tornou-se menos aerodinâmico, de modo que seu coeficiente de amortecimento aumentou em $\Delta b = 10\ Ns/m$. Simule o efeito provocado por esse aumento no coeficiente de amortecimento e discuta a diferença em relação ao caso nominal quando $v_r = 10\ m/s$.

Inclua todos os gráficos solicitados no seu relatório.

2.3 Controle em Malha Fechada

Nessa seção, você implementará os algoritmos de controle em malha fechada para *cruise control* apresentados durante as aulas. Para isso, deve-se implementar a lei de controle "*feedforward* + P":

$$u(t) = u_{ff}(t) + u_{fb}(t) = b_c v_r + K_p e(t), \tag{5}$$

em que $u_{ff}(t)$ é o termo de *feedfoward* e $u_{fb}(t)$ é o termo de *feedback*, $e(t) = v_r - v(t)$ é o erro de velocidade, K_p é o ganho proporcional e b_c é o coeficiente de amortecimento usado no projeto do controlador. Implemente um diagrama no Simulink conforme mostrado na Figura 2. Para isso, foi fornecido o arquivo `cruise_control_fechada.slx`.

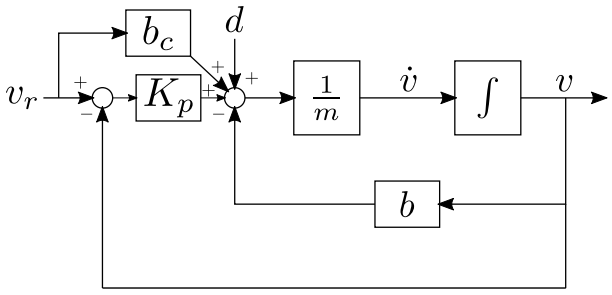

Figura 2: Diagrama do sistema de *cruise control* em malha fechada com *feedfoward* e perturbação.

Calcule K_p para que o sistema em malha fechada tenha constante de tempo de $\tau = 10\ s$. Utilizando a simulação construída no Simulink (assuma que não deve incluir erro de modelo ou perturbação a não ser que seja explicitamente solicitado):

(a) Simule o sistema em malha fechada com $b_c = b$ para $v_r = 10\ m/s$, $v_r = 20\ m/s$ e $v_r = 30\ m/s$. Trace os 3 gráficos numa mesma figura e comente o resultado obtido com base no que foi visto em aula.

(b) Simule o sistema em malha fechada, sem *feedforward* (i.e. $b_c = 0$), para $v_r = 10\,m/s$. Trace o gráfico e comente o que foi obtido.

(c) Considerando que $v_r = 10\ m/s$, faça simulações com os seguintes valores de perturbação: $d = 100\ N$, $d = 200\ N$ e $d = 300\ N$. Trace os 3 gráficos numa mesma figura e comente o resultado obtido conforme o discutido em aula.

(d) Considere que após o sistema de controle já ter sido projetado, o carro sofreu uma colisão e por conta disso tornou-se menos aerodinâmico, de modo que seu coeficiente de amortecimento aumentou em $\Delta b = 10\ Ns/m$. Para $v_r = 10\ m/s$, simule o efeito provocado por esse aumento no coeficiente de amortecimento e discuta.

(e) Simule o sistema em malha fechada com $b_c = b$ apenas para $v_r = 10\ m/s$. Trace um gráfico do esforço de controle $u(t)$ nesse caso e comente o resultado obtido. Faça um paralelo com a estratégia usada em malha aberta para tornar o sistema mais rápido. Observação: você precisará adicionar um bloco do tipo `To Workspace`. Configure o `Save format` como `Structure with Time`. Dê o nome de `out.u` para a sua variável de saída.

Inclua todos os gráficos solicitados no seu relatório.

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 esse código.

- **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

- Para carregar um arquivo `.mat` no MATLAB, basta usar:

  ```
  load('arquivo.mat')
  ```

- Configure `Stop Time` do Simulink como 100 s em todas as simulações.

- É provável que a simulação usando a estratégia de tempo mínimo fique esquisita com o *solver* de passo variável usado por padrão no Simulink, assim veja as dicas de como alterar a simulação para usar um *solver* de passo fixo. Para alterar as configurações do *solver*, vá em `Model Settings`, então em `Solver`. Dentro de `Solver selection`, troque `Type:` para `Fixed-step`. Assim, em `Solver details`, troque o passo (`Fixed-step size`) para `1e-3`.

- Os seguintes blocos de Simulink são úteis nesse laboratório:

(a) `Constant`: define uma constante.

(b) `From Workspace`: recebe variáveis do `workspace` do MATLAB.

(c) `To Workspace`: envia variáveis para o `workspace` do MATLAB.

(d) `Gain`: multiplica um sinal por uma certa constante.

(e) `Sum`: faz soma (e subtração) de vários sinais da simulação.

(f) `Integrator`: integra de um sinal. Observação: $1/s$ significa integração na Transformada de Laplace.

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

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

- Macaco para definir uma entrada usando o bloco `From Workspace` (adapte o macaco):

```
u.time = (0:0.1:100)'
u.signals.values = ones(length(u.time), 1)
u.signals.dimensions = 1
```

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