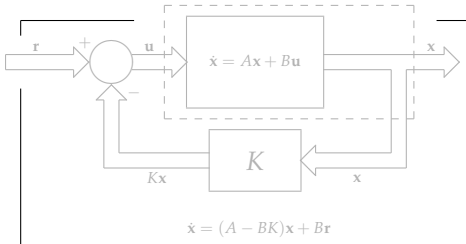

PROF. B. J. MASS – *bjmass@sc.usp.br* – tel. (16) 33 73 93 61 – sala 3005 – campus 1

Nota Didática nd/tdm11 - 01

Versão 1.0

Distribuida em 12/09/2011 - terça-feira

Uma Sinopse da Matéria Coberta: Quatro Semanas

1. Realimentação de Estado

1.1 - A ideia básica consiste em impor a condição: $\mathbf{u} = -K\mathbf{x}$, ou seja forçar $\mathbf{u}$ a ser "proporcional" a $\mathbf{x}$.

1.2 - Como o sistema de malha fechada precisa de uma entrada, a ideia completa consiste em impor $\mathbf{u} = \mathbf{v} - K\mathbf{x}$, onde $\mathbf{v}$ é uma nova entrada externa, que toma o lugar de $\mathbf{u}$. As dimensões envolvidas são as seguintes:

$$
\begin{array}{ll}
\mathbf{u} & \cdots\cdots \; l \times 1 \\
\mathbf{y} & \cdots\cdots \; m \times 1 \\
\mathbf{x} & \cdots\cdots \; n \times 1 \\
A & \cdots\cdots \; n \times n \\
B & \cdots\cdots \; n \times l \\
C & \cdots\cdots \; m \times n \\
D & \cdots\cdots \; m \times l \\
K & \cdots\cdots \; l \times n \\
\mathbf{v} & \cdots\cdots \; l \times 1
\end{array}
$$

1.3 - MATLAB e realimentação de estado

– Declaração de um vetor de n polos (ou autovalores) dados:

```
p = [p1, p2,...,pn];
```

– Comandos para impor polos (ou atovalores) de malha fechada:
"`acker`" - Calcula a matriz K de realimentação, dado o sistema e os autovalores desejados.
Sintaxe: `k = acker(a, b, p);`
Características: Só se aplica a sistemas com uma entrada e uma saída. É necessário que o par A, B seja controlável.

"`place`" - Uma alternativa (melhor) para calcular a matriz K.
Sintaxe: `k = place(a, b, p);`
Características: Serve para sistemas de múltiplas variáveis; recomendado pelo MATLAB.

"`lqr`" - Uma alternativa para calcular a matriz K, que não precisa que especifiquemos os polos desejados para malha fechada.
Sintaxe: `k = lqr(a, b, q, r);`
Caracerísticas: Não precisamos do vetor de polos mas precisamos de Q e R, duas matrizes de ponderação.

1.4 As equações dinâmicas de malha fechada

$$\left.\begin{array}{l}\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}\\ \mathbf{y} = C\mathbf{x} + D\mathbf{u}\\ \mathbf{u} = \mathbf{v} - K\mathbf{x}\end{array}\right\} \implies \left.\begin{array}{l}\dot{\mathbf{x}} = (A - BK)\mathbf{x} + B\mathbf{v}\\ \mathbf{y} = (C - DK)\mathbf{x} + D\mathbf{v}\end{array}\right\}$$

Os autovalores (polos) de malha fechada são os autovalores de $A - BK$.

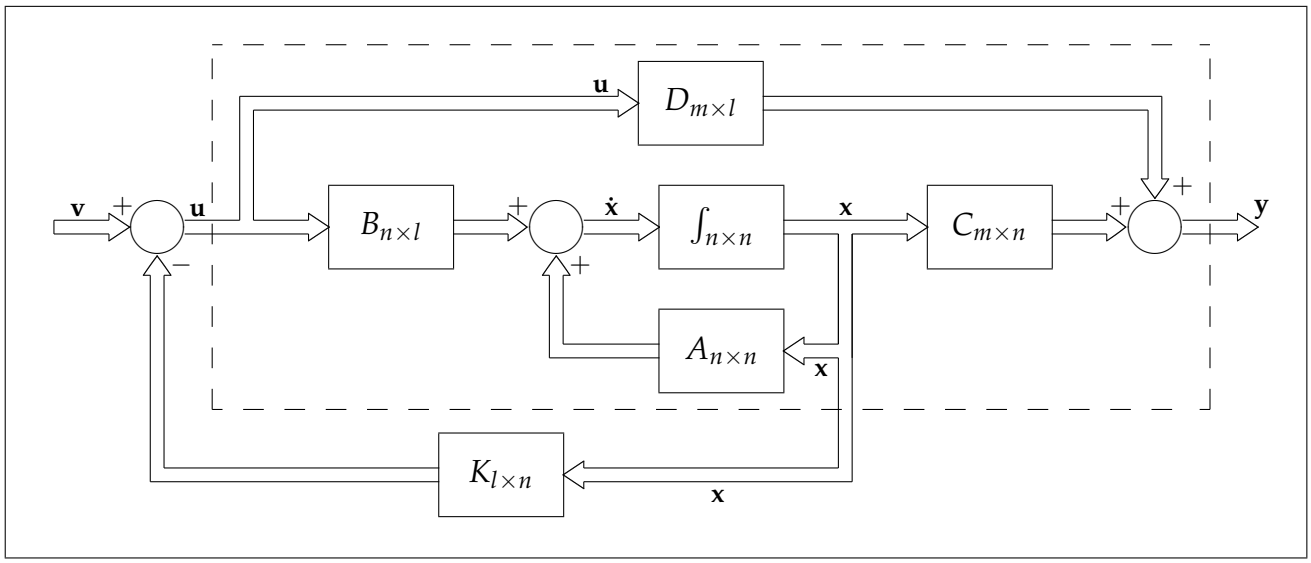

Fig. 1: Estrutura completa de um sistema linear com realimentação de estado. A moldura tracejada contém o sistema de malha aberta, cuja matriz D é geralmente nula.

2. Regulador Linear Quadrático ("LQR")
Dadas

$$\left.\begin{array}{l}\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}\\ \mathbf{u} = -K\mathbf{x}\end{array}\right\}$$

o regulador linear quadrático - LQR (do inglês *linear quadratic regulator*) pode ser interpretado como um algoritmo que determina a incógnita K, que minimiza a integral

$$J = \int_0^\infty (\mathbf{x}^T Q\mathbf{x} + \mathbf{u}R\mathbf{u})dt$$

As matrizes $Q_{n\times n}$ e $R_{l\times l}$ são chamadas matrizes de ponderação e devem ser dadas; a integral J (um número) é chamada índice de desempenho.

Nestas primeiras semanas de SEL0326 não nos envolveremos com o cálculo de J, nem com as razões para a sua introdução. Nosso interesse reside em utilizar o algoritmo para obter uma matriz K de realimentação de estado.

2.1 - MATLAB e o algoritmo LQR

- "`lqr`" - Calcula a matriz K dadas A, B, Q e R.
  Sintaxe: `k = lqr(a, b, q, r);`
  Características: Calcula a matriz sem que precisemos informar os polos de malha fechada desejados.
  Limitação: Temos que informar as matrizes Q e R, cuja escolha é um assunto pessimamente tratado na literatura para graduação.
  As matrizes Q e R: Têm que ser $n \times n$ e $l \times l$ respectivamente mas devem atender outros requisitos; estes são atendidos automaticamente se Q e R forem respectivamente simétricas. Na falta de maiores informações, Q e R podem ser as matrizes identidades com as dimensões apropriadas.
- **Exemplo 1 (Ogata, 4a. ed., p. 779)** Para

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

  deseja-se encontrar o sinal de controle ótimo u, de modo que o índice de desempenho

$$J = \int_0^\infty (\mathbf{x}^T Q \mathbf{x} + u^2) dt, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & \mu \end{bmatrix}$$

  seja minimizado. Para $\mu = 0,01$ determine o sinal ótimo $u(t)$.
  Resposta: $u(t) = -K\mathbf{x}(t)$, $R = [1]$, $K \approx [1 \quad 1,4177]$.
- **Exemplo 2** Escreva um trecho de MATLAB que retorne a matriz K procurada no Exemplo 1 acima.
  Solução:

```
≫ a = [0   1; 0   0]; b = [0;   1];
≫ mi = 0.01; q = [1   0; 0   mi]; r = [1];
≫ k = lqr(a, b, q, r);
```

3. A Forma Canônica Controlável
   Para um sistema com apenas uma entrada e uma saída podemos passar facilmente da função de transferência para um quarteto de matrizes representando o sistema no espaço de estados, i.e., por variáveis de estados.

$$G(s) \longrightarrow (A, B, C, D) \quad \text{trivial}$$

Para sistemas com mais de uma entrada ou mais de uma saída essa conversão **não é simples**, podendo porém ser obtida (numericamente) com facilidade via MATLAB.

$$\mathbf{G}(s) \longrightarrow (A, B, C, D) \quad \text{trivial apenas via MATLAB}$$

Lembrete:
Para qualquer sistema, multivariável ou não, temos o seguinte

$$(A, B, C, D) \mapsto \mathbf{G}(s) = C(sI - A)^{-1}B + D$$

3.1 Funções de transferência (uma só entrada/uma só saída); é útil recapitular a seguinte classificação.

$$\begin{array}{lll} G(s) & \textbf{estritamente própria} & \iff D = [\,0\,] \\ G(s) & \textbf{própria} & \iff D \neq [\,0\,] \\ G(s) & \textbf{imprópria} & \iff y = C\mathbf{x} + D_1 u + D_2 \dot{u} + D_3 \ddot{u} + \cdots \end{array}$$

Quando a função de transferência é imprópria, a implicação é que não podemos representar o sistema pela forma usual

$$\left.\begin{array}{l} \dot{\mathbf{x}} = A\mathbf{x} + Bu \\ y = C\mathbf{x} + Du \end{array}\right\} \tag{1}$$

porque entram em cena derivadas da entrada que não podem ser desprezadas.

3.2 A forma canônica controlável de uma função de transferência **estritamente própria**, na forma

$$G(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \cdots + a_1 s + a_0}{s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0}$$

é simplesmente a representação (1) acima com as matrizes

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \\ -b_0 & -b_1 & -b_2 & \cdots & -b_{n-1} \end{bmatrix}_{n\times n}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}_{l\times 1} \tag{2}$$

$$C = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{m-1} & a_m & 0 & \cdots & 0 \end{bmatrix}_{1\times m}, \quad D = [\,0\,]_{1\times 1} \tag{3}$$

- A matriz D é sempre nula e escalar, i.e., 1×1.
- A matriz A é chamada *companheira*. Os elementos da última linha são os coeficientes do polinômio característico na ordem inversa e com os sinais trocados; os elementos da superdiagonal unitários e todos os outros nulos.
- A matriz B tem o último elemento (b_{l1}) igual a um e todos os outros nulos.
- Os elementos da matriz C são os coeficientes do polinômio do numerador de $G(s)$ na ordem inversa; os elementos que faltarem preenchidos com zeros.

Erratum: Na lista de exercícios LE-02/SEL0326-2011 onde se lê b_n leia-se b_{n-1}.

4. Transformações de similaridade
É importante lembrar que para um dado sistema o quarteto de matrizes que o representa no espaço de estados **não é único**. Há uma infinidade de quartetos, todos equivalentes e perfeitamente válidos:

$$\begin{array}{ccc} G(s) & \Longleftrightarrow & (A,B,C,D) \\ G(s) & \Longleftrightarrow & (\hat{A},\hat{B},\hat{C},\hat{D}) \\ G(s) & \Longleftrightarrow & (\bar{A},\bar{B},\bar{C},\bar{D}) \\ G(s) & \Longleftrightarrow & (\tilde{A},\tilde{B},\tilde{C},\tilde{D}) \\ \vdots & \vdots & \vdots \end{array}$$

Todos esses quartetos são aparentados e o relacionamente é o seguinte

4.1 Transformação de **similaridade** entre matrizes quadradas.
Dada uma matriz quadrada A qualquer, consideremos uma matriz quadrada T, com as mesmas dimensões que A, mas inversível, com inversa T^{-1}. Podemos prémultiplicar A por T e em seguida posmultiplicar o resultado por T^{-1}; obtemos outra matriz quadrada, com as dimensões de A:

$$TAT^{-1} = \breve{A} \tag{4}$$

Uma transformação desse tipo é chamada *transformação de similaridade (ou de semelhança)*. Matrizes que podem ser associadas através de uma transformação de similaridade são ditas matrizes *similares* ou *semelhantes*. Na expressão (4) acima as matrizes A e $\breve{A}$ (A breve) são similares.

4.2 Infinitas transformações de similaridade.
Claramente, dada uma matriz quadrada qualquer A, são infinitas as matrizes inversíveis (com as dimensões de A) que podemos encontrar e consequentemente há uma infinidade de matrizes similares à A.

4.3 Transformações de similaridade e mudanças de variáveis.
Transformações de similaridade representam mudanças de variáveis e como consequência, se T é inversível, podemos por exemplo escrever as seguintes relações entre duas representações de um mesmo sistema no espaço de estados:

$$\begin{array}{ccc} \breve{A} & = & TAT^{-1} \\ \breve{B} & = & TB \\ \breve{C} & = & CT^{-1} \\ \breve{D} & = & D \end{array}$$

$$(\breve{A},\breve{B},\breve{C},\breve{D}) \Longleftrightarrow (A,B,C,D)$$

4.4 A natureza das variáveis de estados.
As matrizes de um dado sistema não garantem a identificação das variáveis de estado; conhecer (A,B,C,D) por exemplo, indica que podemos escrever

$$\left.\begin{array}{l} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} = C\mathbf{x} + D\mathbf{u} \end{array}\right\}$$

mas não garante que possamos identificar a natureza física de cada uma das variáveis em **u**, **y** ou **x**. Uma maneira de conhecer a natureza de cada variável de uma representação de estados, consiste em obter as equações dinâmicas modelando diretamente o sistema original, o que quase sempre envolve equações diferenciais ou de diferenças finitas.

4.5 A representação de estados via MATLAB.
Dada a função de transferência $G(s)$ de um sistema (com uma entrada e uma saída), podemos obter as matrizes correspondentes via

```
≫ [numg, deng] = tfdata(g, 'v');
≫ [a, b, c, d] = tf2ss(numg, deng);
```

Para $G(s)$ da forma

$$G(s) = \frac{a_m s^m + a_{m-1}s^{m-1} + \cdots + a_1 s + a_0}{s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}$$

as matrizes retornadas pelo MATLAB têm sempre a forma

$$A = \begin{bmatrix} -b_{n-1} & \cdots & -b_2 & -b_1 & -b_0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}_{n\times n}, \quad B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}_{l\times 1} \tag{5}$$

$$C = \begin{bmatrix} 0 & \cdots & 0 & a_m & a_{m-1} & \cdots & a_1 & a_0 \end{bmatrix}_{1\times m}, \quad D = \begin{bmatrix} 0 \end{bmatrix}_{1\times 1} \tag{6}$$

claramente associada à forma canônica controlável (2)-(3) acima; na verdade uma variante da mesma. A matriz A em (5) também apresenta uma *forma companheira*.

<u>*Lembrete*</u>:
O comando "`tf2ss`" é aplicável tanto a sistemas com uma entrada e uma saída (funções de transferência), como a sistemas multivariáveis (matrizes de transferência).

5. Funções matriciais de matrizes quadradas.
Matrizes quadradas podem ser multiplicadas infinitas vezes, com o produto mantendo as mesmas dimensões; essa é uma das propriedades que permitem construir, através de séries de potências, funções que levam de $M_n(\mathbf{R})$ — o espaço das matrizes com $n \times n$ elementos reais — ao $M_n(\mathbf{R})$. Assim introduzimos

$$e^A = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \frac{1}{4!}A^4 + \cdots$$

onde $A^2 = AA$, $A^3 = AAA$, $A^4 = AAAA$ e assim por diante. Introduzindo também $A^{\mathbf{0}} = I$, onde I é a matriz identidade apropriada, temos uma definição consistente.

5.1 Propriedades e características da função e^A.

$$\begin{array}{l}
\text{Domínio} = M_n(\mathbf{R}) \\
\text{Contradomínio} = M_n(\mathbf{R}) \\
e^{\mathbf{0}} = I \quad \text{onde} \quad \mathbf{0} \in M_n(\mathbf{R}) \quad \text{é a matriz nula.} \\
e^{-A}e^{A} = I \\
e^{\alpha A}e^{\beta A} = e^{(\alpha+\beta)A} \quad \text{se} \quad \alpha,\, \beta \in \mathbf{R} \\
e^{A}e^{B} = e^{B}e^{A} = e^{(A+B)} \quad \text{se} \quad AB = BA \\
e^{I} = eI
\end{array}$$

5.2 Função matricial do tempo e^{tA}.
Fixada uma matriz quadrada A podemos introduzir uma função de $\mathbf{R}$ em $M_n(\mathbf{R})$, definida pela série de potências

$$e^{tA} = I + tA + \frac{t^2}{2!}A^2 + \frac{t^3}{3!}A^3 + \frac{t^4}{4!}A^4 + \cdots$$

Uma notação esclarecedora seria

$$\Phi_A(t) = e^{tA}$$

por economia porém, a notação normalmente usada é simplesmente

$$\Phi(t) = e^{tA} \quad (\text{ou} \quad \Phi(t) = e^{At})$$

5.3 Propriedades e características da função e^{tA}.

$$\begin{array}{l}
\text{Domínio} = \mathbf{R} \\
\text{Contradomínio} = M_n(\mathbf{R}) \\
\dfrac{d\,e^{tA}}{dt} = Ae^{tA} = e^{tA}A \\
\\
\displaystyle\int e^{tA}dt = A^{-1}(e^{tA} - I) = A^{-1}e^{tA} - A^{-1} \\
\mathcal{L}[e^{tA}] = (sI - A)^{-1} = \Phi(s) - \text{Transformada de Laplace}
\end{array}$$

5.4 Recursos numéricos e simbólicos (MATLAB) para e^A, e^{tA}, etc.

- "`expm`" - Calcula e^A para uma matriz quadrada A.
  Sintaxe: `expm(a)`;
  Características: Pode ser empregado para calcular e^{tA} desde que t seja um número real.

– "exp" - Dada uma matriz A com elementos a_{ij}, calcula a matriz $A^{'}$ cujos elementos são $a^{'}_{ij} = e^{a_{ij}}$.
Sintaxe: `exp(a);`
Características: É muito diferente de "expm", com o qual **não deve ser confundido**.

– **Exemplo 3** Para "`a = [0 0; -1 0];`" temos

```
expm(a) = [1 0; -1 1];
exp(a) = [1 1; 0.3679 1];
```

Obs.: Na exponencial escalar de A **todos os elementos são positivos**; na exponencial matricial de A isso não é sempre verdade.
Obs.: Para a exponencial matricial a matriz A tem que ser quadrada; para a exponencial escalar A **pode ser qualquer**.

– *Comandos ou funções simbólicas:* Alguns comandos no MATLAB admitem argumentos simbólicos, se o subprograma "*Symbolic Math Toolbox*" estiver presente e se o contexto simbólico estiver em vigor.
P/ abrir o contexto simbólico: Digitar "syms" seguido das variáveis simbólicas de interesse, separadas por espaços.

– **Exemplo 4** Podemos utilizar "expm" no contexto simbólico, seguindo

```
≫ a = [2 0; 0 5];
≫ syms t
≫ expmta = expm(t*a)
ans =  [exp(2*t) 0; 0 exp(5*t)]
```

– **Exemplo 5** Um exemplo envolvendo duas variáveis simbólicas seria

```
≫ a = [2 0; 0 5];
≫ syms t s
≫ phiesse = inv(s*eye(2) - a);
≫ phite = ilaplace(phiesse)
phite =  [exp(2*t) 0; 0 exp(5*t)]
```

Obs.: Aqui "eye(2)" é a matriz identidade 2×2 e "ilaplace" o comando para a transformada inversa de Laplace.

– *Limitações dos comandos simbólicos:* Processamento lento; travamento (dependendo da máquina) no caso de sistemas de ordem acima de algumas unidades; resposta em aritmética racional, que pode ser inconveniente. De uma forma geral os comandos simbólicos exigem mais experiência e mais cuidado para serem utilizados satisfatoriamente.

5.5 A função $\Phi(s) = (sI - A)^{-1}$.
Um dos recursos preferidos para a determinação manual de $\Phi(t) = e^{tA}$. A inversão matricial envolvida pode desencorajar, mas exige apenas operações algébricas conceitualmente simples: a transposta da cofatora divida pelo determinante; mesmo assim, para $A_{n\times n}$ com n bem maior que 3, o trabalho braçal pode ser insuportável.

6 A solução da equação de estado.
Como se trata de uma equação diferencial **ordinária**, se A e B forem matrizes constantes, é relativamente simples (via transformada de Laplace) escrever a seguinte expressão para a solução:

$$\mathbf{x}(t) = e^{tA}\mathbf{x}(0) + \int_0^t e^{(t-\tau)A}B\mathbf{u}(\tau)d\tau \tag{7}$$

Claro que para o uso efetivo desta expressão temos que conhecer $\mathbf{u}(t)$ no intervalo de interesse; ainda assim o cálculo da integral pode ser trabalhoso e em geral recorremos a aproximações numéricas, supostamente satisfatórias do ponto de vista de engenharia. Aproximações numéricas satisfatórias estão implícitas em diferentes comandos do MATLAB (por exemplo) mas não serão consideradas por ora.

6.1 Duas soluções aditivas.
Felizmente a expressão (7) é composta de duas parcelas separadas, conhecidas por nomes sugestivos

$$e^{tA}\mathbf{x}(0) \quad \longrightarrow \quad \text{Resposta de entrada nula}$$

$$\int_0^t e^{(t-\tau)A}B\mathbf{u}(\tau)d\tau \quad \longrightarrow \quad \text{Resposta de estado (inicial) zero}$$

É de interesse o caso particular da *resposta de estado zero* em que $\mathbf{u}$ é constante pelo menos no intervalo de interesse. Neste caso o produto $B\mathbf{u}$ é uma matriz constante $n \times 1$, que pode ficar fora da integração.

6.2 A resposta de entrada nula.
Quando $\mathbf{u} = \mathbf{0}$ podemos dizer que o sistema em questão é "largado à própria sorte", i.e., evolui de acordo com a energia interna que tiver, sem estímulo externo. O comportamento do sistema nessas condições é fundamental do ponto de vista de estabilidade, a razão principal pelo nosso interesse na *resposta de entrada nula*.

- *Trajetória de estado:* Partindo de um estado inical $\mathbf{x}(0)$ não nulo, a expressão

$$\mathbf{x}(t) = e^{tA}\mathbf{x}(0)$$

calcula um vetor de estado (ou simplesmente um estado) para cada valor de t. Em outras palavras o estado percorre uma **trajetória** no espaço dos estados.

– <u>*Estabilidade e instabilidade:*</u> Quando uma trajetória de estado fica constrangida a um subconjunto limitado do espaço dos estados (o sistema evolui sem deixar o subconjunto), é natural associar a essa trajetória uma certa *estabilidade*. Se por outro lado uma trajetória cruzar qualquer fronteira sem ser constrangida a subconjunto limitado algum do espaço de estados, somos tentados a associar a essa trajetória uma certa *instabilidade*. Âmbos os conceitos, *estabilidade* e *instabilidade* serão ainda definidos com rigor; as definições apoiadas na resposta de entrada nula.
– <u>*Velocidade do estado:*</u> A equação de estado efetivamente é uma expressão para a velocidade vetorial do estado (dentro do espaço dos estados) em qualquer instante. No caso de resposta de entrada nula podemos escrever

$$\mathbf{v} = \dot{\mathbf{x}} = Ae^{tA}\mathbf{x}(0)$$

6.3 Uma generalização útil.
A expressão (7) foi deduzida considerando-se instante inicial nulo, i.e., $t_0 = 0$. Esse detalhe pode ser facilmente alterado, levando à forma generalizada

$$\mathbf{x}(t) = e^{(t-t_0)A}\mathbf{x}(t_0) + \int_{t_0}^{t} e^{(t-t_0-\tau)A}B\mathbf{u}(\tau)d\tau \tag{8}$$

válida para qualquer t_0, até mesmo negativo.

7 Equações dinâmicas com o tempo discretizado.
Temos grande interesse em considerar as equações de estado e de saída para tempo da forma $t = kT$, onde k é um inteiro e T (ou T_{am}) um intervalo finito chamado *intervalo de amostragem*. Com a manipulação algébrica apropriada e uma notação econômica podemos transformar as usuais equações

$$\left.\begin{array}{l} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} = C\mathbf{x} + D\mathbf{u} \end{array}\right\} \tag{9}$$

válidas para tempo contínuo, nas equações

$$\left.\begin{array}{rcl} \mathbf{x}(k+1) & = & A_d\mathbf{x}(k) + B_d\mathbf{u}(k) \\ \mathbf{y}(k) & = & C_d\mathbf{x}(k) + D_d\mathbf{u}(k) \end{array}\right\} \tag{10}$$

válidas para tempo discreto.

O correto é dizer que as equações (10) representam um **sistema discretizado**, já que representam apenas parte da realidade contida em (9). Por simplicidade dizemos frequentemente que (10) são ou representam um **sistema discreto**, ainda que o sistema original seja de fato contínuo.

7.1 As relações entre a s matrizes.
Já que a representação (10) descende de (9), podemos demonstrar uma conveniente relação de primeira ordem entre (A, B, C, D, T_a) e $(A_d, B_d, C_d, D_d, T_a)$:

$$
\begin{array}{lcl}
A_d & = & e^{T_a A} \\
B_d & = & \int_0^{T_a} e^{\eta A} d\eta B = A^{-1}(e^{T_a A} - I) = A^{-1}(A_d - I) \\
C_d & = & C \\
D_d & = & D
\end{array}
$$

$$
(A, B, C, D, T_a) \longrightarrow (A_d, B_d, C_d, D_d, T_a)
$$

Essa conveniência depende de A; se A não for inversível não conseguimos calcular B_d. O caminho inverso, i.e., obter o sistema contínuo a partir do discreto é mais espinhoso, exigindo considerações sobre o logaritmo de uma matriz, um assunto fora do nosso interesse imediato. Transformaçõe numéricas (e.g. via MATLAB) são perfeitamente possíveis.

7.2 Conversões *contínuo-discreto-contínuo* pela via numérica.
O MATLAB por exemplo oferece os comandos `c2d` (contínuo para discreto) e `d2c` (discreto para contínuo).

- "`c2d`" - Calcula as matrizes A_d, B_d, C_d e D_d dadas A, B, C, D e T_a na forma apropriada.
  Sintaxe: Indireta, através de "`ssdata`", e.g.:

```
≫ sist1 = ss(a, b, c, d);
≫ ta = <   >;
≫ sistd1 = c2d(sist1, ta);
≫ [ad, bd, cd, dd] = ssdata(sistd1);
```

  Observações: Temos que declarar o tempo de amostragem T_a.
  Podemos optar por um algoritmo mais sofisticado, com um argumento a mais em "`c2d`"; para primeira ordem esse argumento é "`zoh`", que pode ser omitido.
- "`d2c`" - Calcula as matrizes A, B, C e D, dadas A_d, B_d, C_d e D_d na forma apropriada.
  Sintaxe: Indireta, através de "`ssdata`", e.g.:

```
≫ ta = <   >;
≫ sisd1 = ss(ad, bd, cd, dd, ta);
≫ sist1 = d2c(sistd1);
≫ [a, b, c, d] = ssdata(sist1);
```

  Observações: **Não** temos que declarar o intervalo de amostragem; o MATLAB reconhece que o argumento de "`d2c`" é um sistema discretizado e "lê" o valor de T_a que acompanha o mesmo.

7.3 O tempo de amostragem.
Para **um sistema linear estável** a fidelidade da discretização é tanto maior quanto maior for a frequência de amostragem (o recíproco de T_a). Em geral escolhemos uma frequência de amostragem bem maior (20 vezes, digamos) que a maior frequência que o sistema a ser discretizado (o sistema contínuo) deixa passar sem atenuar ou distorcer (frequência de corte superior). Para **um sistema linear não estável** ou **um sistema não linear**, a escolha de um intervalo de amostragem adequado exige uma discussão que está fora do escopo de nossa matéria.

8 Sistemas não lineares e simulação.
Para um sistema dinâmico não linear genérico as equações dinâmicas têm a forma

$$\left.\begin{array}{l} \dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} = \mathcal{G}(\mathbf{x}, \mathbf{u}) \end{array}\right\} \tag{11}$$

onde $\mathcal{F}$ e $\mathcal{G}$ são funções (pelo menos uma delas não linear) de todas as variáveis de estado e de todas as variáveis de entrada. Uma forma alternativa de escrevê-las seria

$$\left.\begin{array}{l} \dot{\mathbf{x}} = \mathcal{F}(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_l) \\ \mathbf{y} = \mathcal{G}(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_l) \end{array}\right\} \tag{12}$$

Ainda que a primeira equação permita, em tese, calcular a velocidade do estado em cada instante, em muitos casos isso seria elaborado ou difícil demais para alguma aplicação prática. Uma discretização de (11) levaria a equações da forma

$$\left.\begin{array}{rcl} \mathbf{x}(k+1) & = & \mathcal{F}_d(x_1(k), x_2(k), \ldots, x_n(k), u_1(k), u_2(k), \ldots, u_l(k)) \\ \mathbf{y} & = & \mathcal{G}_d(x_1(k), x_2(k), \ldots, x_n(k), u_1(k), u_2(k), \ldots, u_l(k)) \end{array}\right\} \tag{13}$$

ou na forma compacta

$$\left.\begin{array}{rcl} \mathbf{x}(k+1) & = & \mathcal{F}_d(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y} & = & \mathcal{G}_d(\mathbf{x}(k), \mathbf{u}(k)) \end{array}\right\} \tag{14}$$

Estas últimas permitem calcular numericamente trajetórias de estados de sistemas elaborados, viabilizando simulações extraordinárias (fluidodinâmica, comportamento climático, colisões, etc., etc.).

Final da ND/TDM11-01

Arquivo original: ..... "tdm11nd01.tex"
Arquivo p/ impressão: ... "tdm11nd01.pdf"
Versão: 1.0
No. de páginas: 12
Concluído em: ..... 12/09/2011 - 16:00h