Objetos simbólicos, linearização, transformada de Laplace

Vilma A. Oliveira

USP São Carlos

Aula 1b

Colaboradores

Elmer Alexis Gamboa Peñaloza
Rodolpho Vilela Alves Neves
Rafael Fernando Quirino Magossi
Rafael Mariano

Introdução

Esta aula é para ser ministrada de forma interativa utilizando um microcomputador com o aplicativo Matlab instalado. Tem por objetivo apresentar aos alunos o ambiente e funções básicas do Matlab. Na apresentação encontram-se diversos exercícios a serem resolvidos em sala sob a supervisão do professor. No final será proposta uma tarefa extraclasse a ser iniciada na sala de aula. Pode ser atribuída nota à tarefa realizada desde que a sua solução seja encaminhada e a aluna ou aluno tenha participado da aula. A solução pode ser enviada à professora ou professor via a plataforma Moodle de disciplinas, por exemplo.

Objetos simbolicos e suas operações

Objetos simbólicos podem ser variáveis ou números. Podem ser criados com **sym** ou **syms**:

```
object_name = sym('string')
```

```
a=sym('a')
a =
a
» bb=sym('bb')
bb =
bb
» x=sym('x');
```

```
syms variable_name variable_name variable_name
```

```
» syms y z d
» y
y =
y
```

Álgebra simbólica

Uma equação algébrica simples pode ser resolvida para uma variável e o sistema de equações pode ser resolvido para várias variáveis usando a função *solve*

```
h = solve(eq)
```

or

```
h = solve(eq,var)
```

$$ax^2 + bx + c = k$$

» syms a b c k x
» $eq = a * x\hat{}2 + b * x + c = k$
» pretty(eq)
» X = solve(eq,x);
» pretty(X)

Tarefa!

Considere o sistema linear:

$$\begin{cases} x - 2y + z = 12 \\ 3x + 4y + 5z = 20 \\ -2x + y + 7z = 11 \end{cases}$$

Obter a solução usando o comando Matlab [x1,x2,x3]=solve(eq1,eq2,eq3)

Solução

```
» syms x y z;
» eq1 = x - 2*y+z-12;
» eq2 = 3*x+4*y+5*z-20;
» eq3 = -2*x+y+7*z-11;
»[X,Y,Z] = solve(eq1,eq2,eq3)
```

Plotando expressões simbólicas

Em muitos casos, queremos plotar uma expressão simbólica. Pode ser feito com o comando *ezplot*.

```
ezplot(function)
ezplot(function,[xmin,xmax])
ezplot(function,[xmin,xmax,ymin,ymax])
ezplot(funx,funy)
ezplot(funx,funy,[tmin,tmax])
```

```
» syms x
» S=(3*x+2)/(4*x-1)
» ezplot(S)
```

Tarefa!

Plotar as equações seguintes:

- Círculo:

$$x^2 + y^2 = 1$$

- Elipse:

$$4x^2 + 18x + 4y^2 + 12y - 11 = 0$$

- Plot **X** x **Y**:

$$X = cos(2t)$$
$$Y = sin(2t)$$

Solução

Círculo:

»syms x y
»$S = x\hat{}2 + y\hat{}2 - 1$
»ezplot(S)

Elipse:

»syms x y
»$S = 4 * x\hat{}2 - 18 * x + 4 * y\hat{}2 + 12 * y - 11$
»ezplot(S)

X x Y:

»syms t
»$X = cos(2 * t)$
»$Y = sin(2 * t)$
»ezplot(S)

Transformada de Laplace

No Matlab tem um comando para computar a transformada de Laplace no domínio do tempo. A sintaxe é:

laplace(f)
laplace(f, t)

Exemplo:

```
»syms t a;
»f = exp(-a*t);
»laplace(f)
1/(a + s)
```

Tarefa!

Calcular a transformada de de Laplace:

- Step unitário $\mu(t)$ (dica: no Matlab usar heaviside(t))
- $sin(w * t)$
- Impulse unitário $\delta(t)$ (dica: no Matlab usar dirac(t))
- $cos(w * t)$

Transformada inversa de Laplace

Existe uma outra maneira de computar a inversa de Laplace. A sintaxe é:

F = ilaplace(L)

Exemplo:

```
» syms s a;
»L = 1/(s+a);
» ilaplace(L);
ans =
exp(-a*t)
```

Tarefa!

Calcular a inversa da transformada de Laplace:

- $1/s$
- $1/(s\hat{}2 + w)$
- 1
- $s/(s\hat{}2 + w\hat{}2)$
- $1/(s + a)\hat{}2$

Diferenciação

Diferenciação simbólica pode ser realizada usando o comando *diff*. A sintáxe é:

$$\text{diff(S)} \quad \text{ou} \quad \text{diff(S,var)}$$

Exemplo:

» *symsx*
» $S = exp(x\hat{\ }4);$
» $diff(S)$
$4 * x\hat{\ }3 * exp(x\hat{\ }4)$

»syms x
»$S = exp(x\hat{\ }4);$
»$diff(S, 2)$
$12 * x\hat{\ }2 * exp(x\hat{\ }4) + 16 * x\hat{\ }6 * exp(x\hat{\ }4)$

Diferenciação: transformada de Laplace

Exemplo:

$$y^{''}(t) + 7y^{'}(t) + 12y(t) = 0$$

» syms y(t) t;
»
laplace(diff(diff(y(t),t))+diff(y(t),t)*7+y(t)*12);
ans =
7*s*laplace(y(t), t, s) - D(y)(0) - 7*y(0) - s*y(0) + s^2*laplace(y(t), t, s) + 12*laplace(y(t), t, s)

obs: laplace(y(**t**), **t**, **s**) significa Y(**s**)

Linearização local

Considere um sistema da forma

$$\dot{x} = F(x; u) \tag{1}$$

com x estado, u controle e $F(.,.)$ composto de funções suaves.

Definição: Um estado x_e é um estado de equilíbrio ou ponto de equilíbrio do sistema se uma vez igual $x(t)$ igual a $x_e, x(t)$ permanece em x_e para todo tempo futuro.

O comportamento de sistemas na vizinhança do ponto de equilíbrio pode ser analisado em torno do ponto de equilíbrio. Sejam x_e, u_e um ponto de equilíbrio de (1). Para uma pequena perturbação z e v tem-se

$$x = x_e + z, u = u_e + v. \tag{2}$$

Substituindo agora (2) em (1) obtém-se

$$\dot{z} = F(x_e + z, u_e + v) \tag{3}$$

Aproximação de Taylor

Supondo F na classe de funções diferenciáveis C^2, pode-se expandir (3) em uma série de Taylor em torno do ponto (x_e, u_e), e então

$$\begin{aligned} \dot{z} &= F(x_e, u_e) + [grad_x F]^T \Big|_{x_e, u_e} z + [grad_u F]^T \Big|_{x_e, u_e} v \\ &+ (termos\ ordem\ superior) \end{aligned} \tag{4}$$

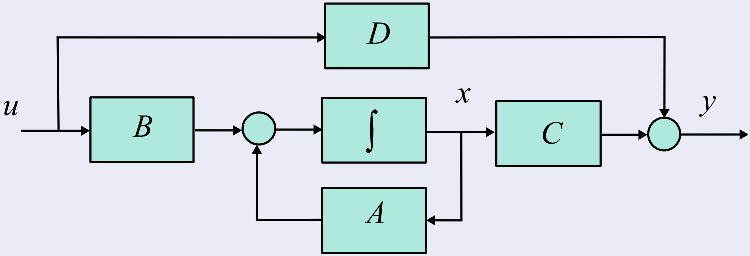
$[grad_x F]^T = \frac{\partial F}{\partial x}$, $[grad_u F]^T = \frac{\partial F}{\partial x}$ são as matrizes Jacobianas de $F(x, u)$ em relação a x e u, respectivamente.

Sistema linearizado

$$\dot{z} \approx [grad_x F]^T\Big|_{x_e,u_e} z + [grad_u F]^T\Big|_{x_e,u_e} v = Az + Bv \qquad (5)$$

$$A = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \cdots & \cdots & \vdots & \cdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}_{x_e,u_e} \qquad B = \begin{bmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \cdots & \frac{\partial F_1}{\partial u_m} \\ \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} & \cdots & \frac{\partial F_2}{\partial u_m} \\ \cdots & \cdots & \vdots & \cdots \\ \frac{\partial F_n}{\partial u_1} & \frac{\partial F_n}{\partial u_2} & \cdots & \frac{\partial F_n}{\partial u_m} \end{bmatrix}_{x_e,u_e}$$

A equação da saída linearizada pode ser obtida de forma análoga, ou seja, $y \approx [grad_x H]^T|_{x_e,u_e} + [grad_u H]^T|_{x_e,u_e}$.



Exemplo Maglev

```
syms x1 x2 x3 v g L0 a m R L ka c1
BL=[0;0;ka/L]
CL=[-c1 0 0]; DL=0;
P=[g m a R L ka L0];
f=[x2;g-L0/(2*a*m)*(x3^2/(1+x1/a)^2);-R/L*x3+1/L*v]
As=jacobian(f,[x1 x2 x3]);
```

Parâmetros Maglev

```
% Parâmetros do sistema Maglev de protótipo de laboratório
g=9.8;m=22.6e-3;a=6.72e-3;R=19.9;L=0.52;ka=2.4;L0=0.0249;
c1=173.61e+1;
% ponto de equilíbrio
x1e=4.5e-3;
Calcular valor de x3e e ve
x3e=sqrt(g*2*a*m*(1+x1e/a)^2/L0);
ve=R*x3e/ka;
Matrizes do sistema linearizado em [x1e x2e x3e]
AL=simplify(subs(As,[x1 x2 x3 v],[x1e 0 x3e ve]));
pretty (AL)
Uso do comando eval para substituir os valores dos AL = eval(AL); BL =
eval(BL); CL = eval(CL); DL = DL;
```

Função de transferência

Relação entrada-saída pode ser descrita via funções de transferência. Os comandos tf(**num**,**den**) and tf('**s**') criam um objeto TF que pode ser usado nas rotinas Matlab.

Cria espaço de estado

»Gmaglev=ss(AL,BL,CL,DL);

Cria função de transferência

»Gmaglev=tf(Gmaglev);

Plota polos e zeros

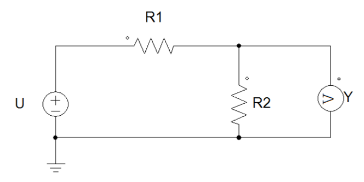
»pzmap(Gmaglev);

Calcula polos

»pole(Gmaglev);

Exemplo

Função de transferência de um circuito elétrico resistivo



Sabemos que:

$$Y = \frac{R2}{R1 + R2} U$$

Assim,

$$G(s) = \frac{Y}{U} = \frac{R2}{R1 + R2}$$

Tarefa para nota

A solução da tarefa deve conter o que foi estudado, o que foi feito e analisado.

Tarefa

Usando os valores dos parâmetros dados anteriormente, obter a função de transferência e os polos e zeros do sistema Maglev.

Referências

[1] Matlab Product Help.
[2] Matlab Demystified. A Self-Teaching Guide, David McMahon, McGraw Hill.