Capítulo 19
Polinômios

Resumo do capítulo

- Polinômios são representados em **MATLAB** por vetores de seus coeficientes em ordem descrescente dos graus de seus termos.
- Como exemplo, o polinômio $x^4 - 12x^3 + 25x + 116$ é criado com o comando `p= [1 -12 0 25 116]`. Note que termos nulos devem ser explicitamente representados.
- Assim, as funções de manipulação de polinômios são combinações de operações em vetores com propósitos específicos: encontrar raízes, operações aritméticas, derivação e integração, cálculo do seu valor em pontos dados, ajuste de curvas. Veremos tais funções neste capítulo.
- Veja também Help –> **MATLAB** Help –> **MATLAB** –> Mathematics –> Polynomials and Interpolation –> Polynomials

Busca por raízes

- Use a função *root(p)* para encontrar raízes do polinômio p.
- Use a função *poly(r)* para encontrar o polinômio dadas as raízes no vetor r. Raízes são sempre vetores-coluna.

```
>> p=[1 -12 0 25 116]
p =
    1   -12     0    25   116
>> r=roots(p)'
r =
  11.7473   2.7028    -1.2251 - 1.4672i    -1.2251 + 1.4672i
>> pp=poly(r')
pp =
   1.0000  -12.0000   -0.0000   25.0000  116.0000
```

Produto

- É implementado pela função *conv(p,q)*, que calcula a convolução dos polinômios p e q:

```
>> a=[1 2 3 4]; b=[1 4 9 16];
>> c=conv(a,b)
c =
    1     6    20    50    75    84    64
```

- Para multiplicar três ou mais polinômios aplique repetidamente a função *conv*.

Adição

- Não há uma função específica; deve-se usar adição de vetores.
- É necessário, porém, ajustar os tamanhos dos vetores para que a soma de coeficientes de termos de mesmo grau seja feita corretamente. Isso é feito adicionando-se “zeros à esquerda” ao polinômio de menor grau.
- Veja a descrição da função *mmpadd(p,q)* que já faz o enchimento de zeros e soma os polinômios p e q.

```
>> c = [1 6 20 50 75 84 64]; d=[2 6 12 20];
>> f = mmpadd(c,d)
f =
  1   6   20   52   81   96   84
>> g = mpadd(c,-d)
g =
  1   6   20   48   69   72   44
```

Divisão

- É feita com a função *deconv(a,b)* que devolve polinômios quociente e resto, q e r.

```
>> c = [1 6 20 50 75 84 64]; b=[1 4 9 16];
>>[q,r]=deconv(c,b)
q =
    1     2     3     4
r =
    0     0     0     0     0     0     0
>> f = [1 6 20 52 81 96 84];
>> [q,r]=deconv(f,b)
q =
    1     2     3     6
r =
    0     0     0     0    -2    -6   -12
```

Derivadas e integrais

- Use as funções *polyder* e *poliint*. (Veja as descrições delas nos arquivos .m também!)

```
>> g = [1 6 20 48 69 72 44]
>> h = polyder(g)
h =
    6    30    80   144   138    72
>> polyint(h,44) % termo constante = 44
ans =
    1     6    20    48    69    72    44
```

Cálculo de polinômios em pontos

- É feito com a função *polyval(p,x)* que calcula o valor do polinômio p nos pontos em x.
- Olhe e execute o arquivo *mm1901.m*.

Ajuste de curva

- Dados pontos no plano, às vezes queremos uma curva que passe “perto” desses pontos, mas não necessariamente nos pontos. Isso pode resultar numa curva mais suave, ao custo de erros de aproximação.
- O método dos quadrados mínimos, por exemplo, minimiza a soma dos quadrados dos erros, para uma curva qualquer. Se escolhermos fazer a aproximação usando um polinômio de grau n, a função *polyfit* do **MATLAB** pode ser usada.
- Veja um exemplo de seu uso no arquivo *mm1902.m*, que usa um polinômio de grau 2 para um ajuste de 11 pontos e o arquivo *mm1903.m* que usa um polinômio de grau 10. Compare a suavidade das duas curvas e suas propriedades numéricas.