Resolução explicada dos exercícios 1, 2 e 3 da folha 4 (resolvidos nas aulas PL dos dias 9, 10 e 11 de dezembro)

exercício 1.a)

```
>> p3=@(x) x^3-3.1*x^2+2.3*x-0.3

p3 =

    @(x)x^3-3.1*x^2+2.3*x-0.3

>> [p3(-0.5), p3(-1), p3(0), p3(3)]

ans =

   -2.3500   -6.7000   -0.3000    5.7000
```

exercício 1.c) Não existe outro polinómio de grau não superior a 3 que interpole f nos 4 pontos dados. Com efeito, dados n+1 pontos $(x_i, y_i),\ i = 0, ...n$, existe e é único o polinómio, digamos p_n, de grau não superior a n, tal que $p_n(x_i) = y_i,\ i = 0, ...n$.

exercício 2.a) Com

$$p_(x) = a_1x^3 + a_2x^2 + a_3x + a_4,$$

os coeficientes $a_1,\ a_2,\ a_3$ e a_4 são a solução do sistema

$$\begin{bmatrix} x_0^3 & x_0^2 & x_0 & 1 \\ x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ x_3^3 & x_3^2 & x_3 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}.$$

A matriz do sistema, que depende apenas dos nós de interpolação,é designada por matriz de Vandermonde. Usaremos a função **vander** do Matlab para construir a matriz correspondente aos nós dados

```
>> x=[-0.5,-1,0,3]; V=vander(x)

V =

   -0.1250    0.2500   -0.5000    1.0000
   -1.0000    1.0000   -1.0000    1.0000
         0         0         0    1.0000
   27.0000    9.0000    3.0000    1.0000
```

A função **det** calcula o determinante

```
>> det(V)

ans =

  -10.5000
```

Pode mostrar-se que o determinante de uma matriz de Vandermonde é igual ao produto das diferenças dos nós. Verifiquemos que assim acontece neste caso

```
>> prod=1; for i=1:4, for j=i+1:4, prod=prod*(x(i)-x(j)); end, end, prod

prod =

  -10.5000
```

exercício 2.b) A solução do sistema é o vetor dos coeficientes do polinómio p_3 dado antes. No Matlab resolvemos o sistema $Va = y$, onde y é o vetor dos valores nodais, como se indica a seguir

```
>> y=[-2.35; -6.7; -0.3; 5.7]; a=V\y

a =

    1.0000
   -3.1000
    2.3000
   -0.3000
```

exercício 3.a) O polinómio expressa-se na forma

$$1\times\frac{(x-1)(x-2)(x-4)}{(0-1)(0-2)(0-4)}-2\times\frac{(x-0)(x-2)(x-4)}{(1-0)(1-2)(1-4)}+0\times\frac{(x-0)(x-1)(x-4)}{(2-0)(2-1)(2-4)}+3\times\frac{(x-0)(x-1)(x-2)}{(4-0)(4-1)(4-2)}$$

exercício 3.b) Dado x, o cálculo do valor da expressão anterior requer exatamente 51 operações aritméticas. Em geral, para $n+1$ pontos, a fórmula interpoladora de Lagrange requer $4n^2+5n$ operações (ver p. 85 das notas das aulas TP).

exercício 3.c) A função **poLagrange** está disponível na área de conteúdo "Matlab"da Blackboard. Devem os estudantes estudar o código que será usado na próxima aula PL.

exercício 3.d)

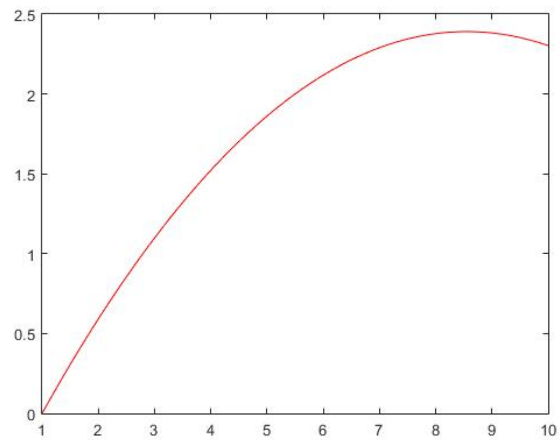
```
>> xi=rand(1,5), yi=rand(1,5)

xi =

    0.6557    0.0357    0.8491    0.9340    0.6787


yi =

    0.7577    0.7431    0.3922    0.6555    0.1712

>> poLagrange(xi,yi,xi(1))

ans =
```

```
    0.7577

>> poLagrange(xi,yi,xi(2))

ans =

    0.7431

>> poLagrange(xi,yi,xi(3))

ans =

    0.3922

>> poLagrange(xi,yi,xi(4))

ans =

    0.6555

>> poLagrange(xi,yi,xi(5))

ans =

    0.1712
```

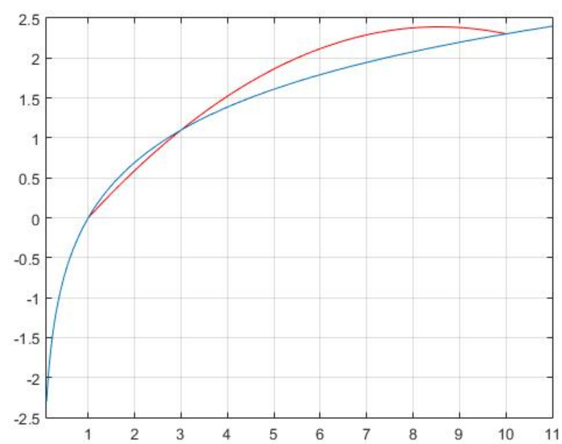
Resolução explicada dos exercícios 4, 5 e 6 da folha 4 (resolvidos nas aulas PL dos dias 15, 16, 17 e 18 de dezembro)

exercício 4.a) Tal como ficou ilustrado na resposta à questão 3.d), a função poLagrange calcula o valor do polinómio interpolador num ponto. No código seguinte, vamos usar aquela função para calcular o polinómio interpolador em cada um dos pontos $1, 1.1, 1.2, 1.3, ..., 10$.

```
xi=[1,3,10]; yi=log(xi);
>> x=1:0.1:10; for k=1:length(x),y(k)=poLagrange(xi,yi,x(k));end,plot(x, y,'r')
```



exercício 4.b) `>> hold on, fplot('log(x)',[0.1,11]), grid on`



(nota: fizeram-se algumas alterações ao código apresentado no enunciado). A figura permite vizualizar os gráficos da função e do polinómio interpolador, os quais, como esperado, se intersetam nos pontos cujas abcissas são os nós de interpolação. Observe-se como o polinómio interpolador tem erros maiores nos pontos mais afastados dos nós

exercício 5.a) Neste caso é $n = 1$ (interpolação linear) uma vez que vão ser usados dois nós. Tem-se

$$(log(x))'' = -1/x^2$$

e a expressão do erro neste caso, para qualquer ponto x entre os nós $x_0 = 1$ e $x_1 = 2$, é

$$log(x) - p_1(x) = (x-1)(x-2)\frac{-1/\xi_x^2}{2!},$$

onde ξ_x é um ponto que está entre 1 e 2 (e depende de x). Uma vez que

$$|-1/\xi_x^2| \leq 1,$$

podemos escrever, com $x = 1.5$ e tomando módulos,

$$|log(1.5) - p_1(1.5)| \leq |(1.5-1)(1.5-2)|\frac{1}{2} = 0.125$$

exercício 5.b) Para $n = 2$ (os nós são $x_0 = 1, x_1 = 2$ e x_2) tem-se

$$(log(x))''' = 2/x^3$$

e a expressão do erro neste caso, para qualquer ponto x entre 1 e 3, é

$$log(x) - p_1(x) = (x-1)(x-2)(x-3)\frac{2/\xi_x^3}{3!},$$

onde ξ_x é um ponto que está entre 1 e 3 (e depende de x). Uma vez que

$$|2/\xi_x^3| \leq 1$$

(o máximo, em módulo, da derivada continua a ocorrer no ponto 1), podemos escrever, com $x = 1.5$ e tomando módulos,

$$|log(1.5) - p_2(1.5)| \leq |(1.5-1)(1.5-2)(1.5-3)|\frac{2}{3!}.$$

O valor do polinómio nodal pode calcular-se no Matlab com

```
>> xi=[1,2,3]; x=1.5; abs(prod(x-xi))
```

e assim o majorante para o erro de truncatura é dado por

```
>> xi=[1,2,3]; x=1.5; abs(prod(x-xi))/3

ans =

    0.1250
```

(nota: embora o majorante do erros de $p_1(1.5)$ e de $p_2(1.5)$ seja o mesmo, tal não significa que os erros sejam iguais. O erro $|log(1.5) - p_1(1.5)|$ é

```
 >> abs(log(1.5)-poLagrange([1,2],log([1,2]),1.5))

ans =

    0.0589
```

e o erro $|log(1.5) - p_2(1.5)|$ é

```
>> abs(log(1.5)-poLagrange([1,2,3],log([1,2,3]),1.5))

ans =

    0.0229

)
```

Pode proceder-se de forma análoga para os restantes valores de n, tratando cada caso separadamente. Vamos proceder de forma diferente. Observando que para $f(x) = log(x)$ é

$$\begin{array}{rcl} f'(x) & = & 1/x \\ f''(x) & = & -1/x^2 \\ f'''(x) & = & 1 \times 2/x^3 \\ f^{(iv)}(x) & = & -1 \times 2 \times 3/x^4 \\ \ldots & & \\ f^{(n+1)}(x) & = & -1 \times 2 \times 3 \times \cdots \times n/x^{n+1} \end{array}$$

e que o máximo de $|f^{(n+1)}(x)|$, no intervalo $[1, n+1]$ é $n!$, atingido no ponto 1, podemos escrever

$$|log(1.5) - p_n(1.5)| \leq |(1.5-1)(1.5-2)(1.5-3)...(1.5-n+1)| \frac{n!}{(n+1)!}.$$

Tem-se

```
>> n=3; xi=1:n+1; x=1.5; abs(prod(x-xi))/(n+1)

ans =

    0.2344

>> n=4; xi=1:n+1; x=1.5; abs(prod(x-xi))/(n+1)

ans =

    0.6563

>> n=5; xi=1:n+1; x=1.5; abs(prod(x-xi))/(n+1)

ans =

    2.4609

>> n=6; xi=1:n+1; x=1.5; abs(prod(x-xi))/(n+1)

ans =

   11.6016
```

o que mostra que não é adequado fazer interpolações com os nós muito afastados do ponto onde se quer aproximar a função.

exercício 6.a) A tabela das diferenças divididas é (ver p. 86 e 87 das notas das aulas)

```
x         f(x)    f[,]    f[,,]

0           0

0.2      0.2013  1.0067

0.4      0.4108  1.0471    0.1010
```

e a fórmula interpoladora de Newton com diferenças divididas neste caso é

$$p_2(x) = 0 + 1.0067 \times (x - 0) + 0.1010 \times (x - 0)(x - 0.2)$$

e para $x = 0.3$, temos

```
>> x=0.3; 0+1.0067*(x-0)+0.1010*(x-0)*(x-0.2)

ans =

    0.3050
```

O mesmo resultado pode ser obtido com a fórmula interpoladora de Newton que requer menos operações aritméticas

```
>> poLagrange([0,0.2,0.4], sinh([0,0.2,0.4]),0.3)

ans =

    0.3050
```

exercício 6.b) Teremos que acrescentar o novo nó 0.6 e correspondente valor nodal sinh(0.6) à tabela anterior e fazer o cáculo das diferenças divididas. Resulta a tabela

```
x         f(x)    f[,]    f [,,]   f[,,,]

0           0

0.2      0.2013  1.0067

0.4      0.4108  1.0471    0.1010

0.6      0.6367  1.1295    0.2061    0.1751
```

e agora tem-se

```
>> x=0.3; 0+1.0067*(x-0)+0.1010*(x-0)*(x-0.2)+0.1751*(x-0)*((x-0.2)*(x-0.4))

ans =

    0.3045
```