Interpolation polynomiale de Lagrange

L'interpolation consiste à trouver l'expression générale d'une fonction à partir d'un nombre limité de points.
Quand la fonction recherchée est un polynôme l'interpolation est dite polynomial.
Dans notre TP on va arborer l'implémentation de la méthode d'interpolation polynomiale de Lagrange.

Principe :

Soit ***f*** une fonction donnée définie sur$\mathbb{R}$. Interpoler la fonction ***f*** par un polynôme ***P*** de degré ***n*** sachant les ***n+1*** points : (x_0, $f(x_0)$), (x_1, $f(x_1)$), . . . , (x_n, $f(x_n)$) consiste à résoudre le problème suivant :
Trouver un polynôme ***P*** de degré ***≤ n*** tel que : $y_i=P(x_i) = f(x_i), 0 \leq i \leq n$.
Ce polynôme est donnée par :

$$P(x) = \sum_{j=1}^{n} P_j(x),$$

P_j est un polynôme qui passe par le point $(x_j, f(x_j))$ et qui s'annule dans tous les autres points, et on peut le calculer par :

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

Construction du polynôme d'interpolation de Lagrange

Supposons que le polynôme d'interpolation est donné par :

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

Afin que ce dernier passe par l'ensemble des points à interpoler, il faut que les coefficients $a_0 \ldots a_n$ respectent le système d'équation linéaire suivant :

$$\begin{pmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Donc pour trouver la formule du polynôme P il suffit de résoudre ce système d'équation.

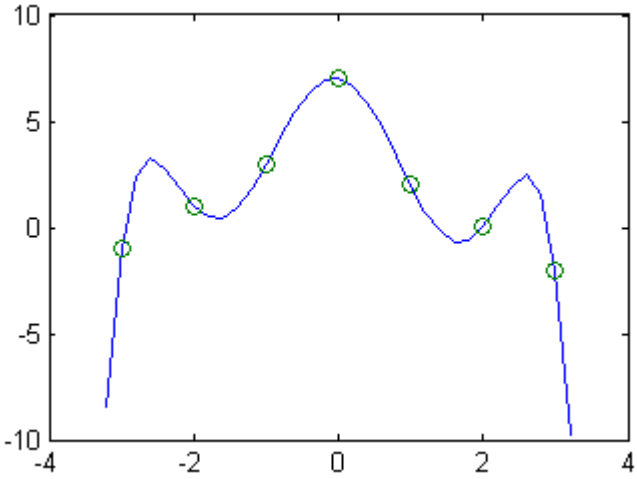
```
function P = lagrange(X,Y)

    N = length(X)-1;

    A = ones(N+1,N+1);
    for i=1:N+1
        for j=1:N
            A(i,j) = X(i)^(N+1-j);
        end
    end
    P = A\Y';
```

```
>> x = [-3,-2,-1,0,1,2, 3];
>> y = [-1, 1, 3,7,2,0,-2];
>> P = lagrange(x,y)
     P =
          -0.1028
          -0.0083
           1.4722
           0.1250
          -5.8694
          -0.6167
           7.0000

>> a = -3.2:0.2:3.2;
>> b = polyval(P, a);
>> plot(a,b,x,y,'o')
```



On aurait pu faire le même travail en utilisant la fonction Matlab **polyfit** comme suit :

```
>> x = [-3,-2,-1,0,1,2, 3];
>> y = [-1, 1, 3,7,2,0,-2];
>> P = polyfit(x,y,6)
     P =
      -0.1028   -0.0083    1.4722    0.1250   -5.8694   -0.6167    7.0000

>> a = -3.2:0.2:3.2;
>> b = polyval(P, a);
>> plot(a,b,x,y,'o')
```

Pour programmer une fonction Matlab qui permet l'évaluation du polynôme mais sans donner son expression (les coefficients) on peut utiliser le code suivant :

```
function y = lagrangeEval(x,PX,PY)

N = length(PX);

y = 0;
for j=1:N
    prod = 1;
    for k = [1:j-1,j+1:N]
        prod = prod*(x-PX(k))/(PX(j)-PX(k));
    end
    y = y + PY(j)*prod;
end
```

```
>> x = [-3,-2,-1,0,1,2, 3];
>> y = [-1, 1, 3,7,2,0,-2];
>> lagrangeEval(5,x,y)
     ans =
          -839.0000
```

Pour vérifier si le résultat est correct, on utilise la fonction Matlab ***polyval*** comme suit :

```
>> P = polyfit(x,y,6);
>> polyval(P,5)
     ans =
          -839.0000
```